LivingApps-Dokumentation

LivingLogic AG

16.05.2025

Inhaltsverzeichnis

1 Allgemeines **2**
1.1 Erweiterte Funktionen . . . 2
1.2 Felder und deren Identifizierer . . . 7
1.3 Anwendungsbeispiel . . . 7

2 Übersicht der Feldtypen **13**
2.1 Feldtyp `string` . . . 13
2.1.1 Untertyp `text` (Standard) . . . 13
2.1.2 Untertyp `textarea` . . . 14
2.1.3 Untertyp `email` . . . 14
2.1.4 Untertyp `url` . . . 15
2.1.5 Untertyp `password` . . . 15
2.1.6 Untertyp `tel` . . . 15
2.2 Feldtyp `int` . . . 16
2.3 Feldtyp `number` . . . 16
2.4 Feldtyp `date` . . . 16
2.4.1 Untertyp `date` (Standard) . . . 17
2.4.2 Untertyp `datetimeminute` . . . 18
2.4.3 Untertyp `datetimesecond` . . . 18
2.5 Feldtyp `lookup` . . . 18
2.5.1 Untertyp `select` . . . 18
2.5.2 Untertyp `choice` . . . 18
2.5.3 Untertyp `radio` . . . 18
2.6 Feldtyp `applookup` . . . 20
2.6.1 Untertyp `select` . . . 20
2.6.2 Untertyp `choice` . . . 20
2.7 Feldtyp `multiplelookup` . . . 21
2.7.1 Untertyp `select` . . . 21
2.7.2 Untertyp `choice` . . . 21
2.7.3 Untertyp `checkbox` . . . 21
2.8 Feldtyp `multipleapplookup` . . . 22
2.8.1 Untertyp `select` . . . 22
2.8.2 Untertyp `choice` . . . 23
2.9 Feldtyp `bool` . . . 23
2.10 Feldtyp `file` . . . 24
2.10.1 Untertyp `signature` . . . 24
2.11 Feldtyp `geo` . . . 25

3 Maximale Feldanzahl **37**

4 URLs für Formulare **38**

4.1 Systemparameter . . . 38

5 Anzeige-Templates **39**
5.1 Erstellung eines Anzeige-Templates . . . 39
5.1.1 Formulierung des Templates . . . 42
5.1.2 Prüfung von Feldwerten auf Inhalt . . . 44
5.1.3 Whitespace . . . 45
5.1.4 Formulierung des HTML-Codes . . . 45
5.1.5 CSS-Template . . . 45
5.1.6 Ergebnisseite . . . 45
5.1.7 Aufruf eines Anzeige-Templates . . . 47
5.2 Anlegen von Datenquellen . . . 49
5.2.1 Sortierung . . . 53
5.2.2 Untergeordnete Datensätze . . . 55
5.2.3 Felder . . . 55
5.3 Zugriff auf System-Apps . . . 55
5.3.1 System-App „Aktivitäten“ . . . 57
5.3.2 System-App „Apps“ . . . 60
5.3.3 System-App „Controls“ . . . 60
5.3.4 System-App „Emailtemplates“ . . . 60
5.3.5 System-App „Versendungen“ . . . 60
5.4 Anwendungsbeispiel . . . 60
5.5 Verwendung von Template-Vorlagen . . . 64
5.5.1 Einzelne Template-Vorlagen übernehmen . . . 64
5.5.2 Anwendungsbeispiele . . . 68
5.5.3 Einen Template-Satz für einen Anwendungsfall übernehmen . . . 90

6 Interne Templates **97**
6.1 Erstellung eines Internen Templates . . . 97
6.2 Anwendungsbeispiel . . . 99

7 E-Mail-Templates **102**
7.1 Erstellung eines E-Mail-Templates . . . 102
7.1.1 E-Mail-Header - letzte Änderungsoptionen . . . 105
7.1.2 Zugriff auf Elemente des Datensatzes und Ausgabe von Feldwerten . . . 107
7.1.3 Einbinden eines Links in das E-Mail-Template . . . 108
7.1.4 Einbinden eines Bildes in der HTML-E-Mail . . . 108
7.2 Anwendungsbeispiel . . . 109
7.2.1 Datenquelle . . . 113
7.3 Verwendung von Template-Vorlagen . . . 114
7.3.1 Benachrichtigung an mich bei neuem Datensatz . . . 116
7.3.2 Bestätigung an den Benutzer bei neuem Datensatz . . . 120
7.4 Versendete E-Mails . . . 121

8 E-Mail-Templates (vereinfacht) **124**
8.1 Erstellung eines vereinfachten E-Mail-Templates . . . 124
8.2 Anwendungsbeispiel . . . 128

9 Formular-Templates **133**
9.1 Ablaufplan bei „Neu“-Formularen . . . 133
9.2 Ablaufplan bei „Bearbeiten“-Formularen . . . 134
9.3 Erstellung eines Formular-Templates . . . 135
9.4 Bearbeiten eines Formular-Templates . . . 136
9.5 Anwendungsbeispiele . . . 136

10 Update-Templates **143**
10.1 Erstellung eines Update-Templates . . . 143
10.2 Anwendungsbeispiel . . . 145

11 Parameter **152**
11.1 Erstellung von App-Parametern . . . 152
11.2 Zugriff in E-Mail- und Anzeige-Templates . . . 154
11.3 Anwendungsbeispiel . . . 155

12 Datenquellen **157**

13 App-Menüs **159**
13.1 Erstellung von App-Menüs . . . 159
13.2 Anwendungsbeispiel . . . 162
13.2.1 Link zu einem Anzeige-Template . . . 162

14 App-Panels **166**
14.1 Erstellung von App-Panels . . . 166
14.2 Anwendungsbeispiel . . . 169

15 Datenmanagement **172**
15.1 Farben . . . 172
15.1.1 Erstellung . . . 172
15.1.2 Formulieren der Bedingung . . . 174
15.2 Sortierung . . . 174
15.2.1 Erstellung . . . 174
15.2.2 Anwendungsbeispiel . . . 176

16 Formularvarianten **179**
16.1 Erstellung . . . 179

17 Daten-Auswertungen **182**
17.1 Erstellung von Daten-Auswertungen . . . 182
17.1.1 Haupt-Felder . . . 182
17.1.2 Filterbedingungen . . . 187
17.2 Anwendungsbeispiel . . . 187

18 Daten-Aktionen **193**
18.1 Erstellung von Aktionen . . . 193
18.1.1 Formulieren der Bedingung . . . 196
18.1.2 Anweisungen . . . 196
18.1.3 Felderbelegung . . . 198
18.2 Anwendungsbeispiele . . . 201
18.2.1 Datensatz verändern . . . 201
18.2.2 Link für E-Mails . . . 201
18.2.3 Datensatz löschen . . . 208
18.2.4 Datensatz über Fomular anlegen . . . 208
18.2.5 Arbeitsaufgaben Datensatz E-Mails . . . 211
18.2.6 E-Mail-Versendung . . . 226
18.2.7 Einladung zu einer App . . . 231

19 Meldungen **238**

20 Konfiguration **239**
20.1 Basisdaten . . . 239
20.2 Bezeichnungen . . . 240
20.3 Felder . . . 243
20.3.1 Erstellung . . . 243
20.3.2 Anwendungsbeispiele . . . 246
20.4 Startlink . . . 249
20.5 Zugewiesene Daten . . . 250
20.6 Uploads schützen? . . . 252
20.7 Aktivitäten anlegen? . . . 253
20.8 Feldreihenfolge beibehalten? . . . 253

21 Account **254**
21.1 Benutzer-Menüs . . . 254
21.1.1 Erstellung von Benutzer-Menüs . . . 254
21.1.2 Anwendungsbeispiel . . . 259
21.2 Benutzer-Panels . . . 259
21.2.1 Erstellung von Benutzer-Panels . . . 259
21.2.2 Anwendungsbeispiel . . . 264
21.3 Login-Token . . . 266
21.4 App-Kategorien . . . 266
21.5 App-Zuordnungen . . . 268
21.6 Kalender . . . 268
21.7 SMTP-Server . . . 276
21.8 Installationen . . . 276

22 Meine LivingApps **278**
22.1 Living-Apps . . . 278
22.2 Alle Meldungen . . . 278
22.3 Alle Anzeige-Templates . . . 278
22.4 Alle Internen Templates . . . 279
22.5 Alle E-Mail-Templates . . . 279
22.6 Alle E-Mail-Templates (vereinfacht) . . . 279
22.7 Alle Formular-Templates . . . 279
22.8 Alle Update-Templates . . . 279
22.9 Alle Parameter . . . 279
22.10 Alle Datenquellen . . . 279
22.11 Alle App-Menüs . . . 279
22.12 Alle App-Panels . . . 279
22.13 Farben (Datenmanagement) . . . 279
22.14 Alle Felder (Datenmanagement) . . . 279
22.15 Sortierung (Datenmanagement) . . . 279
22.16 Alle Formularvarianten . . . 280
22.17 Alle Daten-Aktionen . . . 280
22.18 Alle Daten-Auswertungen . . . 280

23 Werkzeuge **281**
23.1 LivingApps kopieren . . . 281

24 Library-Templates **285**
24.1 Übersicht über die Templates . . . 285
24.2 Einbinden eines Templates . . . 285
24.3 Überschreiben eines Templates . . . 285
24.4 Library-Templates als Methoden eines Typs . . . 287
24.5 Namens-Konvention . . . 288

25 Library-Parameter **289**
25.1 Übersicht über die Parameter . . . 289
25.2 Verwenden eines Parameters . . . 289
25.3 Überschreiben eines Parameters . . . 289
25.4 Namens-Konvention . . . 291

26 vSQL **293**
26.1 Verwendung . . . 293
26.2 Beispiele . . . 293
26.3 Variablen . . . 294
26.4 Objekte . . . 296
26.4.1 Benutzer . . . 296
26.4.2 App . . . 297
26.4.3 App-Parameter . . . 298
26.4.4 Datensätze . . . 298

26.5 Datentypen, Attribute und Methoden . . . 299
26.5.1 None (NONE) . . . 299
26.5.2 Boolsche Werte (BOOL) . . . 299
26.5.3 Ganze Zahlen (INT) . . . 299
26.5.4 Zahlen mit Nachkommastellen (NUMBER) . . . 299
26.5.5 Strings (STR) . . . 299
26.5.6 Datum (DATE) und Datum/Uhrzeit (DATETIME) . . . 303
26.5.7 Zeiträume (DATEDELTA und DATETIMEDELTA) . . . 304
26.5.8 Monatszeiträume (MONTHDELTA) . . . 304
26.5.9 Farben (COLOR) . . . 305
26.5.10 Geo-Koordinaten (GEO) . . . 306
26.5.11 Listen . . . 307
26.6 Operatoren . . . 307
26.6.1 x == y (Vergleich auf Gleichheit) . . . 308
26.6.2 x != y (Vergleich auf Ungleichheit) . . . 308
26.6.3 x < y (Vergleich auf kleiner) . . . 308
26.6.4 x <= y (Vergleich auf kleiner/gleich) . . . 308
26.6.5 x > y (Vergleich auf größer) . . . 308
26.6.6 x >= y (Vergleich auf größer/gleich) . . . 308
26.6.7 x + y (Addition) . . . 308
26.6.8 x - y (Subtraktion) . . . 309
26.6.9 x * y (Multiplikation) . . . 309
26.6.10 x / y (Division) . . . 309
26.6.11 x // y (Ganzzahl-Division) . . . 310
26.6.12 x % y (Modulo-Operator) . . . 310
26.6.13 x and y (Boolsche „Und“-Verknüpfung) . . . 310
26.6.14 x or y (Boolsche „Oder“-Verknüpfung) . . . 311
26.6.15 x in y (Enthalten-Test) . . . 311
26.6.16 x not in y (Nicht-Enthalten-Test) . . . 311
26.6.17 x is y (Identitäts-Test) . . . 311
26.6.18 x is not y (Negativer Identitäts-Test) . . . 311
26.6.19 x[index] (Index-Zugriff) . . . 311
26.6.20 x[start:stop] (Slice-Zugriff) . . . 312
26.6.21 not x (Boolsche Negation) . . . 312
26.6.22 -x (Unäres Minus) . . . 313
26.6.23 x if cond else y (Wenn/Dann-Ausdruck) . . . 313
26.6.24 x.attrname (Attribut-Zugriff) . . . 313
26.6.25 funktion(...) (Funktionsaufruf) . . . 313
26.6.26 x.methode(...) (Methodenaufruf) . . . 313
26.7 Funktionen . . . 313
26.7.1 today() . . . 313
26.7.2 now() . . . 313
26.7.3 bool() . . . 313
26.7.4 bool(obj) . . . 313
26.7.5 int() . . . 314
26.7.6 int(obj) . . . 314
26.7.7 float() . . . 314
26.7.8 float(obj) . . . 314
26.7.9 str() . . . 315
26.7.10 str(obj) . . . 315
26.7.11 date(jahr, monat, tag) . . . 315
26.7.12 datetime(ja, mo, ta, st, mi, se) . . . 315
26.7.13 len(str) . . . 315
26.7.14 len(list) . . . 316
26.7.15 timedelta() . . . 316
26.7.16 timedelta(tage) . . . 316
26.7.17 timedelta(tage, sekunden) . . . 316
26.7.18 monthdelta() . . . 316

26.7.19 monthdelta(n) . . . 316
26.7.20 years(n) . . . 317
26.7.21 months(n) . . . 317
26.7.22 days(n) . . . 317
26.7.23 hours(n) . . . 317
26.7.24 minutes(n) . . . 317
26.7.25 seconds(n) . . . 318
26.7.26 md5(s) . . . 318
26.7.27 random() . . . 318
26.7.28 randrange(start, stop) . . . 318
26.7.29 cos(x) . . . 318
26.7.30 sin(x) . . . 318
26.7.31 tan(x) . . . 318
26.7.32 sqrt(x) . . . 318
26.7.33 abs(x) . . . 318
26.7.34 geo(lat, long) . . . 319
26.7.35 geo(lat, long, info) . . . 319
26.7.36 dist(x, y) . . . 319

27 LivingAPI **320**
27.1 Objekt-Typen . . . 320
27.1.1 Globals . . . 320
27.1.2 HTTPRequest . . . 330
27.1.3 HTTPResponse . . . 331
27.1.4 EmailRequest . . . 332
27.1.5 EmailResponse . . . 332
27.1.6 FlashMessage . . . 333
27.1.7 DataSource . . . 334
27.1.8 DataSourceChildren . . . 334
27.1.9 App . . . 335
27.1.10 AppGroup . . . 343
27.1.11 Control . . . 343
27.1.12 Record . . . 348
27.1.13 RecordChildren . . . 355
27.1.14 Field . . . 355
27.1.15 LayoutControl . . . 360
27.1.16 AppParameter . . . 361
27.1.17 MenuItem . . . 363
27.1.18 Panel . . . 365
27.1.19 LookupItem . . . 368
27.1.20 View . . . 368
27.1.21 ViewControl . . . 370
27.1.22 ViewLookupItem . . . 372
27.1.23 Attachment . . . 372
27.1.24 User . . . 373
27.1.25 File . . . 374
27.1.26 Geo . . . 375
27.1.27 Installation . . . 376
27.1.28 Category . . . 376
27.1.29 KeyView . . . 377
27.1.30 ViewTemplateInfo . . . 377
27.2 Sonstiges . . . 378
27.2.1 Variablen . . . 378
27.2.2 HTTP-Request-Parameter . . . 378
27.2.3 E-Mails über die LivingAPI versenden . . . 379

28 Statische Resourcen **382**

29 Bilderskalierung **383**

30 QR-Codes **384**

31 Externe Pakete **386**

32 Programmierschnittstellen **387**
32.1 REST-Service . . . 387
32.1.1 Autorisierung . . . 387
32.1.2 Verfügbare Endpoints . . . 389
32.1.3 Fehlermeldungen . . . 389
32.1.4 Interne Fehlermeldungen . . . 399
32.2 GraphQL LivingAPI . . . 399
32.2.1 Einloggen . . . 400
32.2.2 Login mit Nutzernamen und Passwort . . . 400
32.2.3 Login mit OIDC . . . 400
32.2.4 Dokumentation . . . 401
32.2.5 GraphQL und Maps . . . 401
32.2.6 Antworten Caching . . . 401
32.3 Python-SDK . . . 402
32.3.1 Überblick . . . 402
32.3.2 Installation . . . 402
32.3.3 Beispiel-App . . . 402
32.3.4 Vorbereitungen in der App . . . 403
32.3.5 Verwendung des SDKs . . . 403

33 Javascript Helfer Bibliothek **408**
33.1 Funktionen . . . 408
33.1.1 Teilen . . . 408
33.1.2 E-Mail-Adressen validieren . . . 410

34 Importfunktion **411**
34.1 Grundsätzliches . . . 411
34.2 Unterschiedliche Szenarien für einen Import . . . 411
34.3 Übernahme von Daten in eine bestehende LivingApp . . . 411
34.4 Übernahme von Daten in eine bestehende LivingApp (Daten ergänzen) . . . 413
34.5 Import in eine neue LivingApp . . . 416
34.5.1 Begutachtung der neuen App . . . 418
34.5.2 Konfiguration der neuen App . . . 419
34.6 Unterstützung des Imports durch aktive Hinweise auf Fehler . . . 419

35 Howtos **421**
35.1 Wie bekomme ich einen LivingApps-Account? . . . 421
35.1.1 Vorgehen bei Eigeninitiative . . . 421
35.1.2 Vorgehen bei Einladung zur Mitarbeit an einer LivingApp . . . 421
35.2 Wie bekomme ich Administrationsrechte an einer LivingApp? . . . 421
35.2.1 Voraussetzung . . . 421
35.2.2 Vorgehen . . . 421
35.3 Wie kann ich Personen zur Mitarbeit an einer LivingApp einladen? . . . 423
35.3.1 Voraussetzungen . . . 423
35.3.2 Vorgehen . . . 423
35.4 Wie kann ich mich als Experten-User freischalten lassen? . . . 424
35.4.1 Voraussetzung . . . 424
35.4.2 Vorgehen . . . 424
35.5 Wie kann ich mehrere Datensätze auf einmal ändern? . . . 424
35.5.1 Voraussetzungen . . . 424
35.5.2 Vorgehen . . . 426
35.6 Wie kann ich mehrere Datensätze auf einmal kopieren? . . . 428
35.6.1 Voraussetzungen . . . 428
35.6.2 Vorgehen . . . 428
35.7 Wie kann ich mehrere Datensätze auf einmal löschen? . . . 429

35.7.1 Voraussetzungen . . . 429
35.7.2 Vorgehen . . . 429
35.8 Wie kann ich Anhänge (Dateien, URLs, Notizen) für einen Datensatz hochladen? . . . 431
35.8.1 Voraussetzung . . . 431
35.8.2 Vorgehen . . . 431
35.9 Wie kann ich die Tab-Reihenfolge für die Dateneingabe im Formular ändern? . . . 433
35.9.1 Voraussetzungen . . . 433
35.9.2 Vorgehen . . . 433
35.10 Wie kann ich bei Verwendung eines Formulars Feldwerte vorbelegen? . . . 433
35.10.1 Vorgehen . . . 433
35.11 Welche Möglichkeiten habe ich, Einfluss auf Eingabeformulare zu nehmen? . . . 438
35.11.1 Vorgehen . . . 439
35.12 Wie kann ich mehrere zusammengehörige LivingApps kopieren? . . . 442
35.12.1 Voraussetzungen . . . 442
35.12.2 Vorgehen . . . 442
35.13 Wie kann ich die Bezeichnung eines Feldes im Formular ändern? . . . 443
35.13.1 Voraussetzungen . . . 443
35.13.2 Vorgehen . . . 443
35.14 Wie kann ich Uploads in Datensätzen vor dem Zugriff von Dritten schützen? . . . 445
35.14.1 Voraussetzungen . . . 445
35.14.2 Vorgehen . . . 445
35.15 Wie kann ich E-Mails über das Datenmanagement verschicken? . . . 445
35.15.1 Voraussetzungen . . . 445
35.15.2 Vorgehen . . . 446
35.15.3 Ergebnis . . . 451
35.16 Wie kann ich, wenn ein Hauptdatensatz gelöscht wird, auch alle untergeordneten Datensätze löschen? 451
35.16.1 Voraussetzungen . . . 451
35.16.2 Vorgehen . . . 451
35.17 Wie kann ich im Datenmanagement für Zahlenfehler eine Summe anzeigen lassen? . . . 455
35.17.1 Voraussetzungen . . . 455
35.17.2 Vorgehen . . . 455
35.18 Wie kann ich Interne Templates und Parameter vererben? . . . 457
35.18.1 Voraussetzungen . . . 457
35.18.2 Vorgehen . . . 457
35.19 Wie kann ich in Anzeige-Templates Font Awesome-Icons verwenden? . . . 459
35.19.1 Voraussetzungen . . . 459
35.19.2 Vorgehen . . . 459
35.19.3 Standard-Icon-Stil . . . 459
35.19.4 Verwendung eines Nicht-Standard-Stils . . . 461
35.19.5 Unterschiedliche Icons je nach Font Awesome-Version . . . 462
35.20 Wie kann ich in einem Anzeige-Template ein HTML-„Akkordeon“ verwenden? . . . 462
35.21 Wie kann ich in meinen Anzeige-Templates den Dunkel-Modus des Browsers unterstützen? . . . 463
35.21.1 Voraussetzungen . . . 463
35.21.2 Vorgehen . . . 464

Stichwortverzeichnis **466**

In dieser Dokumentation werden die erweiterten Funktionen von LivingApps erklärt, die Erstellung und Anwendung demonstriert und anhand eines Anwendungsbeispiels vorgestellt.

Für die Nutzung der erweiterten Funktionen müssen Sie Administrationsrechte an der App besitzen.

KAPITEL 1

Allgemeines

In dieser Dokumentation werden die *erweiterten Funktionen* von LivingApps erklärt, die Erstellung und Anwendung demonstriert und anhand eines Anwendungsbeispiels vorgestellt.

Für die Nutzung der erweiterten Funktionen müssen Sie *Administrationsrechte* an der LivingApp besitzen und für manche Anwendungen müssen Sie zusätzlich als *Experten-User* freigeschaltet sein.

1.1 Erweiterte Funktionen

Wählen Sie im Menü Ihrer App *Konfiguration* → *Erweitert*. In der linken Spalte finden Sie alle verfügbaren erweiterten Funktionen. Eine ausführliche Beschreibung der einzelnen Funktionen können Sie im jeweiligen Kapitel nachlesen.

Wenn Sie in einer der erweiterten Funktionen auf Ihre Datensätze zugreifen möchten, so muss hierzu die Templatesprache UL4 verwendet werden. Die UL4-Dokumentation finden Sie unter https://python.livinglogic.de/UL4.html. Dort finden Sie unter anderem auch eine Beschreibung, wie Sie lokale Variablen, Dictionaries und Listen anlegen. Die Formulierung der UL4-Ausdrücke wird jedoch auch im Folgenden an den entsprechenden Stellen erläutert.

Die Ausgabe der Daten Ihrer App erfolgt über die *LivingAPI*. Hierüber können Sie im entsprechenden Kapitel und in der nachfolgenden Dokumentation mehr lesen.

Anzeige-Templates
: Erstellen Sie hier sogenannte „UL4-Templates“, die es Ihnen ermöglichen, die Daten Ihrer LivingApp in jeder denkbaren Form darzustellen und aufzubereiten.

Interne Templates
: Sie haben hier die Möglichkeit, sich eine Bibliothek von „UL4-Templates“ anzulegen. Diese Templates können Sie dann in Ihren Anzeige- und E-Mail-Templates aufrufen.

E-Mail-Templates
: Gestalten Sie hier Vorlagen für E-Mails, die von LivingApps oder vom Benutzer in bestimmten Situationen automatisch verschickt werden:

  Beim Anlegen eines neuen Datensatzes, beim Ändern eines Datensatzes, oder wenn ein Benutzer im Datenmanagement eine E-Mail an die in den Datensätzen hinterlegten E-Mail-Adressen verschickt.

E-Mail-Templates (vereinfacht)
: Einfachere Erstellung, dadurch auch weniger Gestaltungsmöglichkeiten.

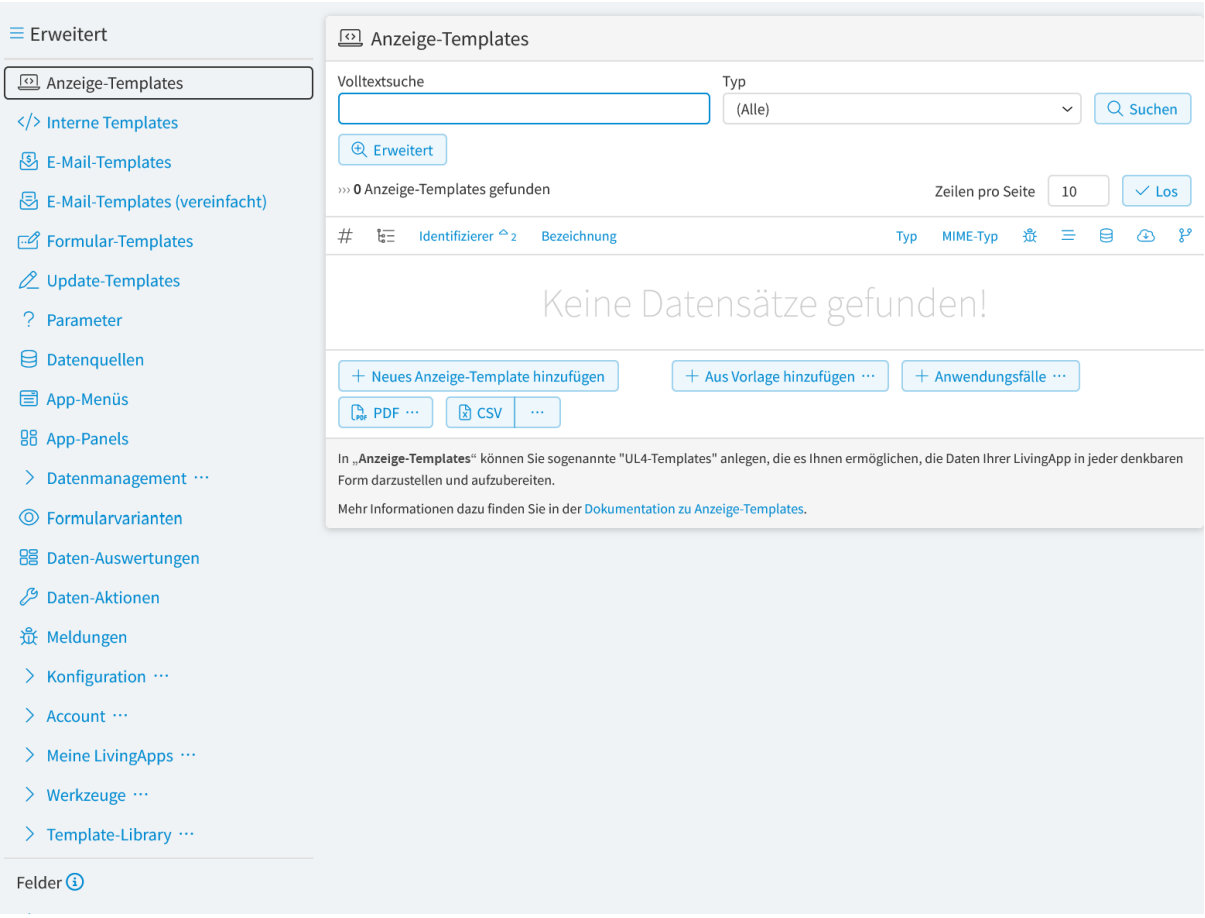

Abb. 1: Anmeldeformular Konfiguration - Erweitert

Formular-Templates
Mit Formular-Templates können Sie Aktionen definieren, die im Eingabeformular der jeweiligen Ansicht beim Aufruf, beim Speichern, nach dem Speichern oder im Fehlerfall des Formulars ausgeführt werden.

So können z. B. Feldwerte vorbelegt, die Auswahl von Optionen eingeschränkt usw. . . .

Update-Templates
Mit Update-Templates können Sie Aktionen definieren, die im Eingabeformular der jeweiligen Ansicht ausgeführt werden, während der Benutzer Daten eingibt.

Damit ist es möglich — in Abhängigkeit von den Eingaben des Benutzers — Werte in Eingabefeldern zu manipulieren, sowie Felder zu deaktivieren, oder komplett auszublenden. Außerdem kann der Absendebutton in Abhängigkeit von bestimmten Bedingungen aktiviert oder deaktiviert werden.

Parameter
Legen Sie hier Konfigurationswerte fest, auf die Sie dann in den Anzeige- und E-Mail-Templates zugreifen können.

Datenquellen
Unter *Datenquellen* sehen Sie alle Datenquellen die einem Anzeige-, E-Mail-, oder Formular-Template dieser LivingApp zugeordnet sind oder die diese LivingApp als Ziel-App haben. Hierüber können die Datenquellen gelöscht werden.

App-Menüs
In *App-Menüs* können Sie Menüs und Menüeinträge anlegen, die das Navigationsmenü auf der Detailseite Ihrer LivingApp erweitern.

App-Panels
In *App-Panels* können Sie Panels anlegen, die auf der rechten Seite der Detailseite Ihrer LivingApp angezeigt werden und weiterführende Links enthalten.

Datenmanagement. . .
Konfigurieren Sie hier die Anzeige der Daten in der Datenliste.

Farben
So können Sie in der Datenliste Datensätze einfärben, wenn diese bestimmte Bedingungen erfüllen.

Felder
Legen Sie hier fest, in welcher Reihenfolge die Felder Ihrer App in der Datenliste (Spalten) angezeigt werden und ob sie in der Datenliste sichtbar sein sollen.

Sortierung
Legen Sie hier die Sortierung der Datensätze im Datenmanagement (Zeilen) fest. Wird nichts angegeben, so werden die Datensätze absteigend nach dem Erzeugungsdatum sortiert (d. h. die neuesten Datensätze sind oben).

Formularvarianten
Hier können Sie für die einzelnen Formular-Varianten festlegen, in welchem Zeitraum sie angezeigt werden oder wie oft darüber ein Datensatz geändert werden darf.

Daten-Auswertungen
Hier können zusätzliche Darstellungen der Daten in der Liste. Z. B. Monatsauswertungen, Summen, usw. erstellt werden.

Welche Felder in diesen Darstellungen angezeigt werden, lässt sich flexibel konfigurieren. Weiterhin ist es möglich, die Daten zu aggregieren (d.h. die Datensätze anhand bestimmter Kriterien zu gruppieren).

Daten-Aktionen
Erstellen Sie hier Aktionen, die automatisch (z. B. wenn ein neuer Datensatz vom Benutzer angelegt wird) oder manuell (z. B. durch Anklicken einer URL in einer E-Mail oder eines Buttons im Datenmanagement) Datensätze verändern, löschen, neue Datensätze generieren, oder E-Mails verschicken.

Meldungen
In *Meldungen* werden Fehler- oder Log-Meldungen Ihrer LivingApps gespeichert.

Konfiguration. . .

Basisdaten
In *Basis-Daten* können Sie den Name, die Beschreibung und das Icon der LivingApp ändern.

Bezeichnungen
In *Bezeichnungen* können Sie festlegen, wie die Dinge in Ihrer LivingApp bezeichnet werden (also z.B. „Mitarbeiter“ oder „Anmeldungen“ usw. statt nur „Datensätze“).

Startlink
Legen Sie hier fest, zu welcher Seite der Benutzer kommt, wenn er diese LivingApp in der App-Übersichtsseite anklickt.

Zugewiesene Daten
Hier können Sie festlegen, welche Bedingung erfüllt sein muß, damit ein Datensatz einem Benutzer zugewiesen wird.

Uploads schützen?
In *Uploads* können Sie den Zugriff auf die hochgeladenen Dateien konfigurieren.

Aktivitäten anlegen?
In *Aktivitäten* können Sie konfigurieren, ob für diese App bei Ereignissen (Datensatzänderungen, versendete E-Mails, Chat-Nachrichten) Aktivitäten für beteiligte Benutzer angelegt werden sollen.

Feld-Reihenfolge beibehalten?
In *Feld-Reihenfolge beibehalten?* können Sie festlegen, ob eine geänderte Feld-Reihenfolge im Datenmanagement beibehalten werden soll, auch wenn Sie die LivingApp ändern.

Account…
Die Funktionen unter *Account* sind nicht app-spezifisch.

Benutzer-Menüs
In *Benutzer-Menüs* können Sie Menüs und Menüeinträge anlegen, die das Navigationsmenü auf der App-Übersichtsseite oder den Detailseiten aller Ihrer LivingApps erweitern.

Menüs, die sie hier anlegen, werden nur für Sie angezeigt, nicht für andere Benutzer.

Benutzer-Panels
In *Benutzer-Panels* können Sie Panels anlegen, die auf der rechten Seite der App-Übersichtsseite oder den Detailseiten aller Ihrer LivingApp angezeigt werden und weiterführende Links enthalten.

Panels, die sie hier anlegen, werden nur für Sie angezeigt, nicht für andere Benutzer.

Login-Token
In *Login-Token* können Sie einen alternativen Login-Mechanismus aktivieren.

App-Kategorien
In *App-Kategorien* können Sie hierarchische Kategorien anlegen, und Ihren Apps, zur besseren Übersicht, Kategorien zuweisen. Diese Kategorien tauchen auf der App-Übersichtsseite als Navigation auf.

App-Zuordnungen
App-Zuordnungen zeigen die Zuordnungen Ihrer Apps zu den unter *App-Kategorien* aufgelisteten Kategorien.

Kalender
Hier können Sie konfigurieren, wie die Termine der Arbeitsaufgaben in Ihren Apps publiziert werden. Dies ermöglicht es Ihnen diese Termine mit Ihrem Kalender-Program zu synchronisieren (z.B. „Outlook Express“ (auf Windows) oder „Kalender“ (auf Mac OS X)).

SMTP-Server
Unter *SMTP-Server* können Sie den Zugang zu einem eigenen E-Mail-Server für E-Mails einrichten, die über LivingApps versendet werden. Dieser E-Mail-Server kann dann bei Versendungen und E-Mail-Templates verwendet werden.

Installationen
Hier werden alle unter diesem Account durchgeführten Installtionen aufgeführt.

Meine Living-Apps…

Hier finden Sie eine Übersicht und alle Konfigurationsmöglichkeiten der LivingApps, bei denen Sie Administrationsrechte haben.

Living-Apps

Die Liste aller LivingApps, die ich administriere.

Über diese Maske können Sie, nach Auswahl einer LivingApp, alle Informationen zu dieser LivingApp einsehen. Außerdem gelangen Sie über *Konfiguration…*, *Templates…*, *Sonstiges…* und *Datenmanagment…* in alle erweiterten Funktionen, die für diese LivingApp angelegt wurden.

Alle Anzeige-Templates

Die Anzeige-Templates aller LivingApps, die ich administriere.

Alle Internen Templates

Die Internen Templates aller LivingApps, die ich administriere.

Alle E-Mail-Templates

Die E-Mail-Templates aller LivingApps, die ich administriere.

Alle E-Mail-Templates (vereinfacht)

Die vereinfachten E-Mail-Templates aller LivingApps, die ich administriere.

Alle Formular-Templates

Die Formular-Templates aller LivingApps, die ich administriere.

Alle Update-Templates

Die Update-Templates aller LivingApps, die ich administriere.

Alle Parameter

Die konfigurierten Parameter aller LivingApps, die ich administriere.

Alle Datenquellen

Die Datenquellen der Templates aller LivingApps, die ich administriere.

Alle App-Menüs

Die erstellten App-Menüs aller LivingApps, die ich administriere.

Alle App-Panels

Die erstellten App-Panels aller LivingApps, die ich administriere.

Farben (Datenmanagement)

Die Farbkonfigurationen im Datenmanagement aller LivingApps, die ich administriere.

Alle Felder (Datenmanagement)

Die Felderkonfigurationen im Datenmanagement aller LivingApps, die ich administriere.

Sortierung (Datenmanagement)

Die Sortierkonfigurationen im Datenmanagement aller LivingApps, die ich administriere.

Alle Formularvarianten

Die Formularvarianten aller LivingApps, die ich administriere.

Alle Daten-Aktionen

Die Daten-Aktionen aller LivingApps, die ich administriere.

Alle Daten-Auswertungen

Die Daten-Auswertungen aller LivingApps, die ich administriere.

Werkzeuge

LivingApps kopieren

Mit diesem Werkzeug können Sie mehrere zusammengehörende LivingApps kopieren.

Template-Library…

Templates
Hier finden Sie eine Auswahl an Hilfstemplates, die Sie kopieren und für Ihre LivingApps einsetzen können.

Parameter
Hier finden Sie eine Auswahl an App-Parametern, die Sie für Ihre LivingApps übernehmen können.

1.2 Felder und deren Identifizierer

Die Daten einer App werden in *Feldern* mit unterschiedlichen Feldtypen organisiert. Diese Felder legen Sie an, wenn Sie in Ihrer App unter *Konfiguration* → *Eingabe* ein Formular erstellen.

Möchten sie diese Feldinformationen einsehen, wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken dann unten auf das ? neben *Felder*.

Es öffnete sich folgendes Fenster:

Hier bekommen Sie Informationen über die *Bezeichnung*, den *Identifizierer*, den *Typ*, das *Ziel* der Felder Ihrer App und unter *Liste?*, ob das Feld im Datenmanagement angezeigt wird.

Jedes der Felder hat eine bestimmte *Bezeichnung*, die Sie selbst wählen. Da diese Bezeichnung Sonder- und Leerzeichen enthalten oder mehrfach vergeben werden kann, wird zusätzlich für jedes Feld ein eindeutiger *Identifizierer* erstellt. Wenn Sie in einer der erweiterten Funktionen auf die Daten in Ihren Feldern zugreifen möchten, benötigten Sie diese Identifizierer.

Zum *Typ* können Sie im Kapitel *Übersicht der Feldtypen* mehr lesen.

Ist Ihr Feld vom Typ `lookup` finden Sie unter *Ziel* die entsprechende *Auswahl* und für den Typ `applookup` die *verknüpfte App* mit der entsprechenden Auswahl.

Im Anschluss an die Feldinformationen werden die Formularvariationen mit der entsprechenden Formular-ID aufgeführt.

1.3 Anwendungsbeispiel

Die erweiterten Funktionen von LivingApps werden im Folgenden anhand der Beispiel-LivingApp „Anmeldung“ veranschaulicht. Über das *Formular* der App soll man sich zu Veranstaltungen anmelden und gleichzeitig Angaben zu Übernachtungen machen können. Die anmeldende Person soll nach der Anmeldung automatisch über die erfolgreiche Anmeldung informiert werden und gleichzeitig eine Bestätigung per E-Mail erhalten, in der nochmal alle gemachten Angaben aufgelistet sind. Außerdem soll das Anzeigetemplate „Teilnehmerliste“ erstellt werden, das alle Teilnehmer und die jeweils gewählte Veranstaltung auflistet.

Für die folgenden Kapitel ist es notwendig, dass die *Identifizierer der Felder* des Formulars bekannt sind. Wie bereits erwähnt, können diese durch Klicken auf *Felder* eingesehen werden. Hier können auch die Typen der Felder abgelesen werden. Die *Veranstaltung*-Auswahlbox hat z. B. den Identifizierer `veranstaltung2` und ist vom Typ `applookup/select`, das Feld zur Eingabe der *E-Mail-Adresse* hat den Identifizierer `e_mail_adresse` und ist vom Typ `string/email` und das Feld *Einzelzimmer* hat den Identifizierer `einzelzimmer` und ist vom Typ `number`. Über Feldtypen mit Ihren Untertypen können Sie im Kapitel *Übersicht der Feldtypen* mehr lesen.

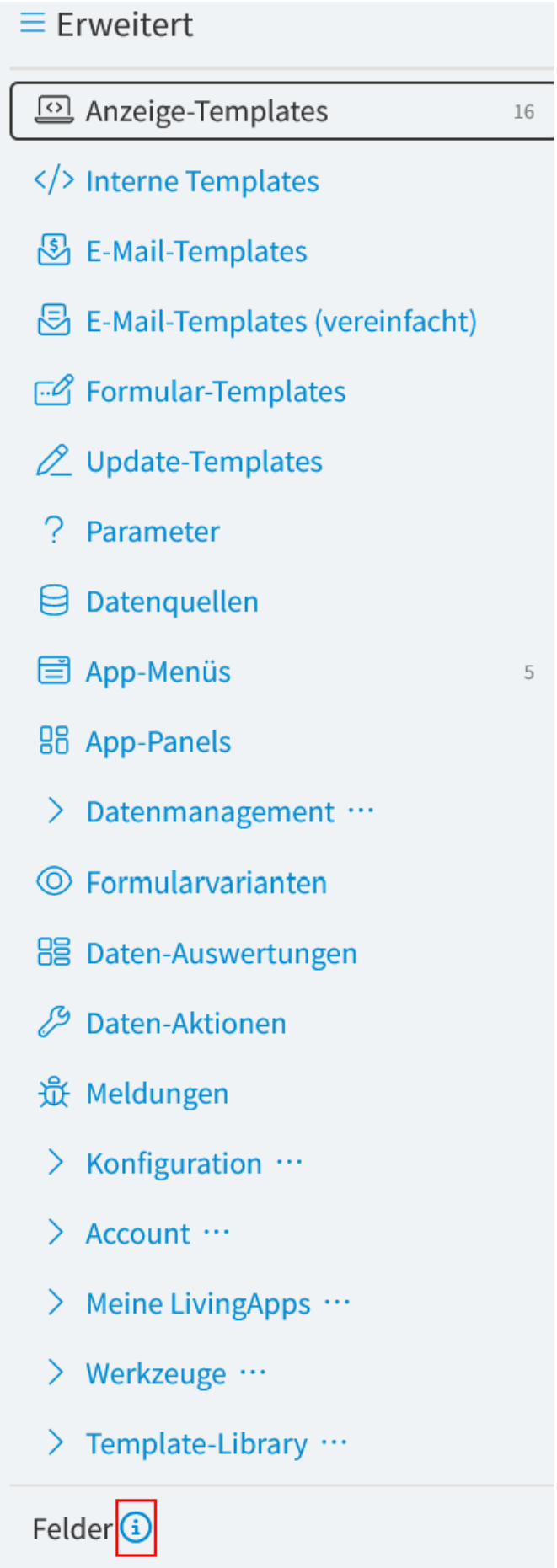

Abb. 2: Felder einsehen

Feld-Information

Anmeldung
Computerkurse

App-ID da11219bb494aff28062f50f

Felder

Bezeichnung	Identifizierer	Typ	Liste?	Ziel
Anrede	anrede	lookup/radio		Auswahl
Zuständiger Mitarbeiter	zustaendiger_mitarbeiter	string/email	✓	
abgemeldet	abgemeldet	bool		
Zimmer sind gebucht	zimmer_sind_gebucht	bool		
Teilnahmegebühr	teilnahmegebuehr	number		
bezahlt	bezahlt	bool	✓	
bezahlt am	bezahlt_am	date/date		
Veranstaltung	veranstaltung2	applookup/select		Veranstaltungen (Doku) Beispiel-App für Doku
Vorname	vorname	string/text	✓	
Nachname	nachname	string/text	✓	
Straße Hsnr.	strasse_und_hsnr2	string/text		
PLZ Ort	plz_ort	string/text		
E-Mail-Adresse	e_mail_adresse	string/email		
Telefon	telefon	string/tel		
Anzahl Personen	anzahl_personen2	number	✓	
Wird Übernachtung benötigt?	wird_uebernachtung_benoetigt	lookup/radio	✓	Auswahl
Einzelzimmer	einzelzimmer	number		
Euro / Nacht	euro_nacht	number		
Doppelzimmer	mehrbettzimmer	number		
Euro / Nacht	euro_nacht2	number		

Formularvarianten

Anmeldung aktuell	1d93748d3bb8548b7524b9a2
Anmeldung Sommer	4208320cc4b29dcf3642466e

Abb. 3: Feldinformationen

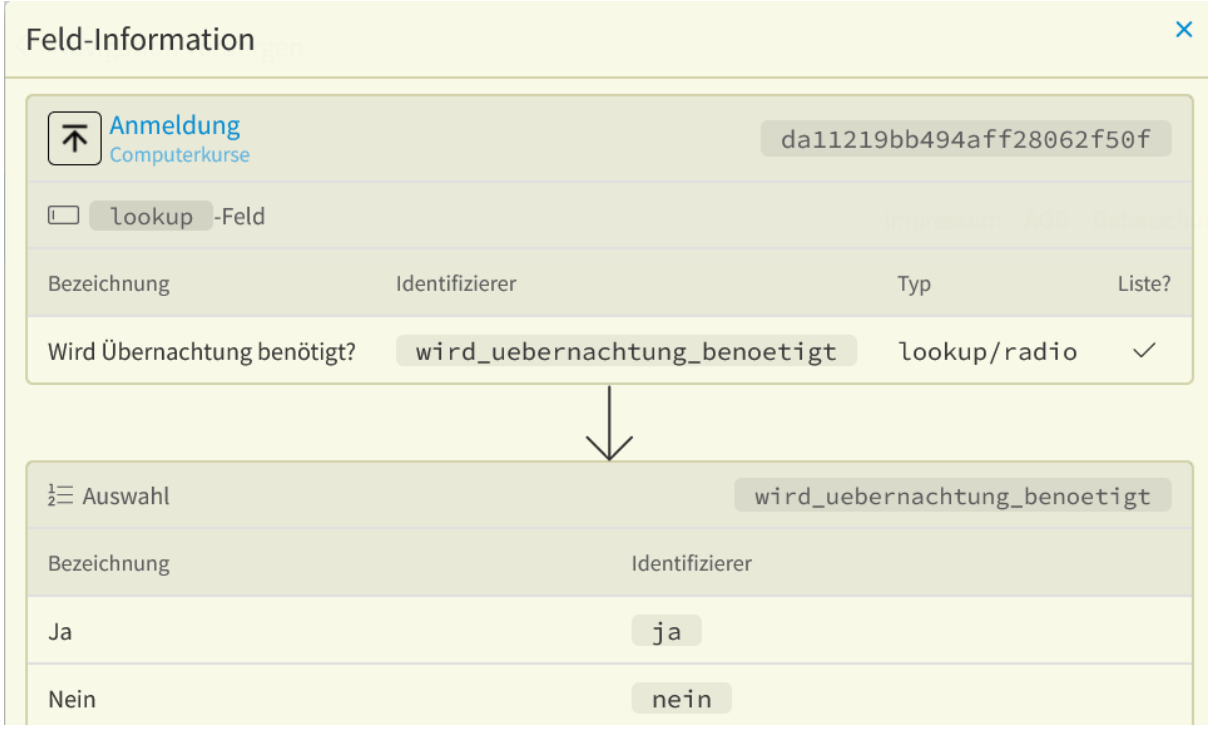

Abb. 4: Feld-Information Ziel - lookup

Veranstaltung	veranstaltung	string/text	✓
Ort	ort	string/text	✓
Datum	datum	date/date	✓
von - bis	von_bis	string/text	✓
E-Mail-Verteiler	e_mail_verteiler	string/email	✓
Icon	icon	file	

Formularvarianten

Neue Ansicht	6a00c31235aafde748a83c45

Applookups auf diese App

App	Feld-Name	Identifizierer	Typ
Anmeldung Computerkurse	Veranstaltung	veranstaltung2	applookup/select
Doku der Feldtypen App zur Dokumentation der F...	Auswahlbox (einfach)	auswahlbox	applookup/select
Doku der Feldtypen App zur Dokumentation der F...	Veranstaltung	veranstaltung	applookup/choice
Doku der Feldtypen App zur Dokumentation der F...	Veranstaltungen	veranstaltungen	applookup/choice
Vereinsveranstaltungen Bitte geben Sie eine aussagekräft...	Veranstaltung	veranstaltung	applookup/choice

Abb. 5: Feld-Information Ziel - applookup
Im Anschluss an die Feldinformationen der verknüpften LivingApp wird noch aufgeführt, welche LivingApps über ein Applookupfeld mit dieser LivingApp verbunden sind.

Anmeldung

Veranstaltung * bitte auswählen

Ihre persönlichen Daten:

Vorname * Nachname *

Straße Hsnr. * PLZ Ort *

E-Mail-Adresse * Telefon

Mit mir melde ich folgende Anzahl an Personen zu oben genannter Veranstaltung an. Personen

Wird Übernachtung benötigt?

Ja ○ Nein ○

Verbindlich anmelden

Abb. 6: Anmeldeformular

KAPITEL 2

Übersicht der Feldtypen

Die Daten einer App werden in Feldern mit unterschiedlichen Feldtypen organisiert. Felder können bei der Erstellung einer neuen App, oder bei einer bereits bestehenden App unter *Konfiguration* → *Eingabe* im Formulardesigner angelegt werden. Der Feldtyp (`type`) legt grundsätzlich die Art des Inhaltes fest. Untertypen (`subtypes`) stellen entweder eine Einschränkung des erlaubten Feldtyps dar, oder legen fest, wie das Feld dem Benutzer präsentiert wird. Wird kein Untertyp gewählt, wird der Standardwert verwendet.

Folgende Feldtypen mit ihren Untertypen stehen zur Auswahl:

2.1 Feldtyp `string`

Feld für die Eingabe von Text, E-Mail-Adressen, Internet-Adressen, Telefonnummern.

2.1.1 Untertyp `text` (Standard)

Feld für die Eingabe von Text (bis zu 4000 Zeichen) Z. B. Name, Straße, Ort, Beschreibung, usw. Wird im *Formulardesigner* angelegt durch Auswahl von *Eingabefeld (einzeilig)* bei *Formularelemente* und *Text* bei *Allgemein* → *Art des Inhalts*. So sieht ein `string/text`-Feld in der *Dateneingabe* aus.

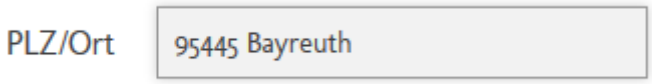

Abb. 1: `string/text`-Feld in der Dateneingabe

Auf folgende Weise können Sie den Inhalt eines `string/text`-Feldes in Templates ausgeben:

```
<?printx record.fields.identifier.value?>
```

Shortcut:

```
<?printx record.v_identifier?>
```

Wobei `identifier` für den *Identifizierer des Feldes* steht.

2.1.2 Untertyp `textarea`

Feld für die Eingabe von mehrzeiligem Text (unbegrenzt). Z. B. für ausführliche Beschreibungen, Kommentare, Notizen, usw. Wird im *Formulardesigner* angelegt als *Textfeld (mehrzeilig)*. So sieht ein `string/textarea`-Feld in der *Dateneingabe* aus.

Abb. 2: `string/textarea`-Feld in der Dateneingabe

Sie haben die Möglichkeit, den Inhalt Ihres mehrzeiligen Textfeldes zu verschlüsseln. Wählen sie dafür im *Formulardesigner* unter *Feldinhalt verschlüsseln…* die entsprechende Option aus.

Nicht anwenden
: Der Inhalt wird nicht verschlüsselt.

Erzwingen
: Der Inhalt muss beim Speichern mit einem Passwort verschlüsselt werden.

Optional
: Der Inhalt kann nach dem Speichern mit einem Passwort verschlüsselt werden.

Auf folgende Weise können Sie den Inhalt eines `string/textarea`-Feldes in Templates ausgeben:

```
<?printx record.fields.identifier.value?>
```

Shortcut:

```
<?printx record.v_identifier?>
```

Wobei `identifier` für den *Identifizierer des Feldes* steht.

2.1.3 Untertyp `email`

Textfeld für die Eingabe von E-Mail-Adressen, z. B. `max.mustermann@beispiel.com`. Die Eingabe der E-Mail-Adresse wird auf Gültigkeit überprüft. Wird im *Formulardesigner* angelegt als *Eingabefeld (einzeilig)* bei *Formularelemente* und *E-Mail* bei *Allgemein* → *Art des Inhalts*. Ein E-Mail-Feld ist z. B. notwendig für E-Mail-Versendungen aus dem Datenmanagement oder für Bestätigungs-Mails über E-Mail-Templates. So sieht ein `string/email`-Feld in der *Dateneingabe* aus.

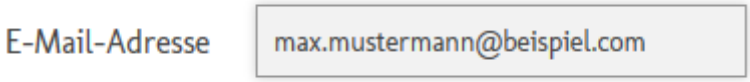

Abb. 3: `string/email`-Feld in der Dateneingabe

Auf folgende Weise können Sie den Inhalt eines `string/email`-Feldes in Templates als Link ausgeben:

```
<a href="mailto:<?printx record.fields.identifier.value?>">
   <?printx record.fields.identifier.value?>
</a>
```

Shortcut:

```
<a href="mailto:<?printx record.v_identifier?>">
   <?printx record.v_identifier?>
</a>
```

Wobei `identifier` für den *Identifizierer des E-Mail-Feldes* steht. Beim Anklicken des Links im Template öffnet sich Ihr E-Mail-Programm.

2.1.4 Untertyp `url`

Textfeld für die Eingabe von Internet-Adressen, z. B. `http://www.livinglogic.de/`. Die Eingabe der URL wird auf Gültigkeit überprüft. Wird im *Formulardesigner* angelegt als *Eingabefeld (einzeilig)* bei *Formularelemente* und *URL* bei *Allgemein* → *Art des Inhalts*. So könnten z. B. bei Recherche-Arbeiten im Internet besonders interessante Links in URL-Feldern hinterlegt werden. Ansicht eines `string/url`-Feldes in der *Dateneingabe*.

Abb. 4: `string/url`-Feld in der Dateneingabe

Auf folgende Weise können Sie den Inhalt eines `string/url`-Feldes in Templates als Link ausgeben:

```
<a href="<?printx record.fields.identifier.value?>">
   <?printx record.fields.identifier.value?>
</a>
```

Shortcut:

```
<a href="<?printx record.v_identifier?>">
   <?printx record.v_identifier?>
</a>
```

Wobei `identifier` für den *Identifizierer des Feldes* steht.

2.1.5 Untertyp `password`

Passwortfeld - momentan nicht aktiv

2.1.6 Untertyp `tel`

Textfeld für die Eingabe von Telefonnummern. Die Eingabe wird auf Gültigkeit überprüft. Erlaubt sind Zahlen von `0`-`9`, `/`, Leerzeichen und `+`. Wird im *Formulardesigner* angelegt als *Eingabefeld (einzeilig)* bei *Formularelemente* und *Telefonnummer* bei *Allgemein* → *Art des Inhalts*. So sieht ein `string/tel`-Feld in der *Dateneingabe* aus.

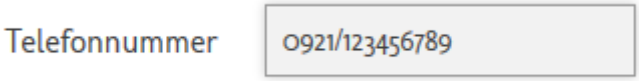

Abb. 5: `string/tel`-Feld in der Dateneingabe

Auf folgende Weise können Sie den Inhalt eines `string/tel`-Feldes in Templates als Link ausgeben:

```
<a href="tel:<?printx record.fields.identifier.value?>">
   <?printx record.fields.identifier.value?>
</a>
```

Shortcut:

```
<a href="tel:<?printx record.v_identifier?>">
   <?printx record.v_identifier?>
</a>
```

Wobei `identifier` für den *Identifizierer des Feldes* steht. Beim Anklicken des Links kann die Telefonnummer direkt angewählt werden (bei App auf dem Handy).

2.2 Feldtyp `int`

Textfeld mit Ganze-Zahlen-Validierung - momentan nicht aktiv

2.3 Feldtyp `number`

Textfeld für die Eingabe von Zahlen, z. B. `12345,6`. Die Eingabe wird auf Gültigkeit überprüft. Wird im *Formulardesigner* angelegt als *Eingabefeld (einzeilig)* bei *Formularelemente* und *Zahl* bei *Allgemein → Art des Inhalts*. Sollen für ein Feld Summierungen oder sonstige Berechnungen erstellt werden, dann muss das Feld ein Zahlenfeld sein. Z. B. für Berechnungen der Summe oder des Durchschnitts in Daten-Auswertungen. So sieht ein `number`-Feld in der *Dateneingabe* aus.

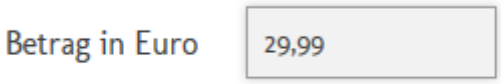

Abb. 6: `number`-Feld in der Dateneingabe

Auf folgende Weise können Sie den Inhalt eines `number`-Feldes in Templates ausgeben:

```
<?printx record.fields.identifier.value?>
```

Shortcut:

```
<?printx record.v_identifier?>
```

Wobei `identifier` für den *Identifizierer des Feldes* steht.

Mit folgender Funktion werden die Zahlen mit 2 Nachkommastellen ausgegeben:

```
<?def formatfloat(v)?>
   <?if v is not None?>
      <?code v = str(v)?>
      <?if "." not in v?>
         <?code v += "."?>
      <?end if?>
      <?code v += "00"?>
      <?code v = v[:v.find(".")+3]?>
   <?end if?>
   <?return v.replace(".", ",")?>
<?end def?>
<?printx formatfloat(record.v_identifier)?>
```

2.4 Feldtyp `date`

Textfeld für die Eingabe von Datum und Uhrzeit. Eingabe wird auf Gültigkeit überprüft. Wird im *Formulardesigner* angelegt als *Eingabefeld (einzeilig)* bei *Formularelemente* und *Datum und Uhrzeit* bei *Allgemein → Art des*

Inhalts. Bei einem Datumsfeld öffnet sich bei der Dateneingabe ein Kalender zur Auswahl von Datum und Uhrzeit. Datumsfelder sind sinnvoll um Zeiträume darzustellen, Zeiterfassungen zu machen, usw. So sieht ein Datumsfeld mit dem Untertyp `datetimeminute` in der *Dateneingabe* aus.

Abb. 7: `date/datetimeminute` in der Dateneingabe

Folgende *Datumsformate* stehen zur Auswahl:

2.4.1 Untertyp `date` (Standard)

Tag, Monat, Jahr (z. B. 29.02.2000)

Auf folgende Weise können Sie den Inhalt eines `date/date`-Feldes in Templates ausgeben:

```
<?if record.fields.identifier.value?>
   <?printx format(record.fields.identifier.value, "%d.%m.%Y", "de")?>
<?end if?>
```

Shortcut:

```
<?if record.v_identifier?>
   <?printx format(record.v_identifier, "%d.%m.%Y", "de")?>
<?end if?>
```

2.4.2 Untertyp `datetimeminute`

Tag, Monat, Jahr, Stunden, Minuten (z. B. 29.02.2000 15:30)

Auf folgende Weise können Sie den Inhalt eines `date/datetimeminute`-Feldes in Templates ausgeben:

```
<?if record.fields.identifier.value?>
   <?printx format(record.fields.identifier.value, "%d.%m.%Y %H:%M", "de")?>
<?end if?>
```

Shortcut:

```
<?if record.v_identifier?>
   <?printx format(record.v_identifier, "%d.%m.%Y %H:%M", "de")?>
<?end if?>
```

2.4.3 Untertyp `datetimesecond`

Tag, Monat, Jahr, Stunden, Minuten, Sekunden (z. B. 29.02.2000 15:30:45)

Auf folgende Weise können Sie den Inhalt eines `date/datetimesecond`-Feldes Templates ausgeben:

```
<?if record.fields.identifier.value?>
   <?printx format(record.fields.identifier.value, "%d.%m.%Y %H:%M:%S", "de")?>
<?end if?>
```

Shortcut:

```
<?if record.v_identifier?>
   <?printx format(record.v_identifier, "%d.%m.%Y %H:%M:%S", "de")?>
<?end if?>
```

Wobei `identifier` jeweils für den *Identifizierer des Feldes* steht.

2.5 Feldtyp `lookup`

Feld zur Auswahl einer Option aus einer Liste von Auswahloptionen.

Hier ist in der Dateneingabe die Auswahl maximal einer Option möglich.

2.5.1 Untertyp `select`

Auswahl-Box einfach (Standard)

Wird im *Fomulardesigner* angelegt als *Auswahlbox*. Hier können unter *Auswahlen* → *Optionen Standard* beliebig viele Auswahl-Optionen angelegt werden. Z. B. verschiedene Kategorien, denen Datensätze zugeordnet werden sollen. So sieht ein `lookup/select`-Feld in der *Dateneingabe* aus.

2.5.2 Untertyp `choice`

Dieser Untertyp ist sinnvoll für große Auswahlmengen. Allgemeine Beschreibung siehe Untertyp `select`. Zusätzlich muss hier bei *Auswahlen* → *Optimieren für große Auswahlmenge* das Häckchen gesetzt werden. In der Dateneingabe werden dann erst nachdem man Zeichen eingegeben hat, die passenden Auswahlen vorgeschlagen. Groß-/Kleinschreibung ist dabei von Bedeutung. So sieht ein `lookup/choice`-Feld in der *Dateneingabe* aus.

2.5.3 Untertyp `radio`

Wird im *Fomulardesigner* angelegt als *Optionen*. Z. B. soll als Anrede `Herr` oder `Frau` gewählt werden. So sieht ein `lookup/radio`-Feld in der *Dateneingabe* aus.

Auf folgende Weise können Sie den Inhalt eines `lookup`-Feldes in Templates ausgeben:

Abb. 8: `lookup/select`-Feld in der Dateneingabe

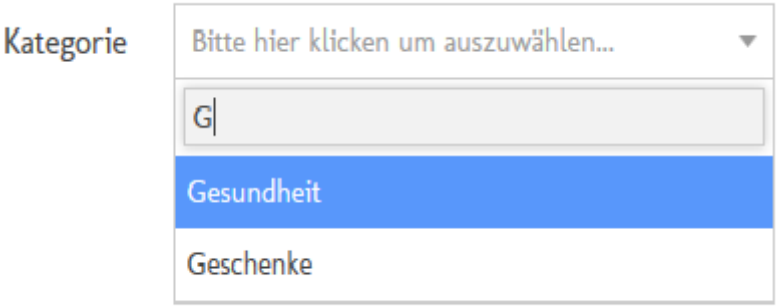

Abb. 9: `lookup/choice`-Feld in der Dateneingabe

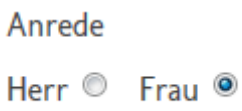

Abb. 10: `lookup/radio`-Feld in der Dateneingabe

```
<?if record.fields.identifier.value?>
   <?printx record.fields.identifier.value.label?>
<?end if?>
```

Shortcut:

```
<?if record.v_identifier?>
   <?printx record.v_identifier.label?>
<?end if?>
```

Wobei `identifier` für den *Identifizierer des Feldes* steht.

2.6 Feldtyp `applookup`

Dieser Typ ist ähnlich zum Typ `lookup` jedoch legen Sie hier die Auswahloptionen nicht beim Feld fest, sondern die Datensätze einer festgelegten, ausgewählten App werden als Auswahloptionen verwendet.

Hier ist in der Dateneingabe die Auswahl maximal einer Option möglich.

2.6.1 Untertyp `select`

Wird im *Fomulardesigner* angelegt als *Auswahlbox*. Hier kann unter *Auswahlen* → *Optionen einer App nutzen* eine App gewählt werden. Weiterhin können Sie festlegen welche Felder dieser App in der Auswahl angezeigt werden. Z. B. könnten einer App „Anmeldung“ die Daten der App „Veranstaltungen“ mit den Feldern „Veranstaltung“ und „Datum“ als Auswahl zur Verfügung gestellt werden. So sieht ein `applookup/select`-Feld in der *Dateneingabe* aus.

Abb. 11: `applookup/select`-Feld in der Dateneingabe

2.6.2 Untertyp `choice`

Dieser Untertyp ist sinnvoll für große Auswahlmengen. Allgemeine Beschreibung siehe Untertyp `select`. Zusätzlich muss hier bei *Auswahlen* → *Optimieren für große Auswahlmenge* das Häckchen gesetzt werden. In der Dateneingabe werden dann erst nachdem man Zeichen eingegeben hat, die passenden Auswahlen vorgeschlagen. Groß-/Kleinschreibung ist dabei von Bedeutung.

Auf folgende Weise können Sie den Inhalt eines `applookup`-Feldes in Anzeige- und internen Templates ausgeben:

```
<?code field = record.f_identifier?>
<?if field.value?>
   <?for reffieldidentifier in field.control.lookupcontrols?>
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
      <?code reffield = field.value.fields[reffieldidentifier]?>
      <?printx reffield.control.label?>: <?printx reffield.value?>
   <?end for?>
<?end if?>
```

Um alle anzuzeigenden Felder des zugeordneten Datensatzes auszugeben, muss man eine Schleife über diese Felder benutzen. Dabei steht `identifier` für den *Identifizierer des anzuzeigenden Feldes* und `reffieldidentifier` durchläuft die Feld-Identifizierer die für die Auswahl konfiguriert wurden. Im oben genannten Beispiel werden so die Inhalte der Felder „Veranstaltung“ und „Datum“ aus der verknüpften App „Veranstaltungen“ ausgegeben.

2.7 Feldtyp `multiplelookup`

Feld zur Auswahl mehrerer Optionen aus einer Liste von Auswahloptionen. Hier ist in der Dateneingabe die Auswahl mehrerer Optionen möglich. Dafür muss bei der Auswahl die `Strg`-Taste der Tastatur gedrückt werden.

2.7.1 Untertyp `select`

Auswahl-Box mehrfach (Standard)

Wird im *Fomulardesigner* angelegt als *Auswahlbox*. Hier können unter *Auswahlen* → *Optionen Standard* beliebig viele Auswahl-Optionen angelegt werden. Bei *Auswahlen* → *Mehrfachauswahl zulassen* muss das Häkchen gesetzt werden. Es können z. B. verschiedene Kategorien angelegt werden, denen Datensätze zugeordnet werden sollen. Ein Datensatz kann hier mehreren Kategorien zugeordnet werden. So sieht ein `multiplelookup/select`-Feld in der *Dateneingabe* aus.

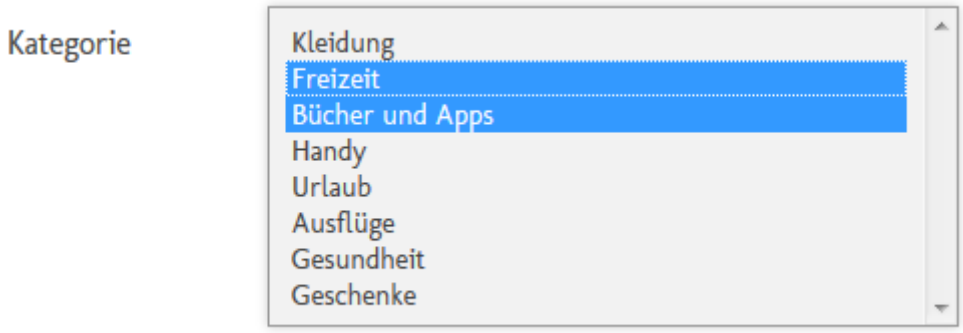

Abb. 12: `multiplelookup/select`-Feld in der Dateneingabe

2.7.2 Untertyp `choice`

Dieser Untertyp ist sinnvoll für große Auswahlmengen. Allgemeine Beschreibung siehe Untertyp `select`. Zusätzlich muss hier bei *Auswahlen* → *Optimieren für große Auswahlmenge* das Häckchen gesetzt werden. In der Dateneingabe werden dann erst nachdem man Zeichen eingegeben hat, die passenden Auswahlen vorgeschlagen. Groß-/Kleinschreibung ist dabei von Bedeutung. So sieht ein `multiplelookup/choice`-Feld in der *Dateneingabe* aus.

2.7.3 Untertyp `checkbox`

Wird im *Fomulardesigner* angelegt als *Optionen (mehrfach)*. Kann z. B. verwendet werden für Multiple Choice-Fragen. So sieht ein `multiplelookup/checkbox`-Feld in der *Dateneingabe* aus.

Auf folgende Weise können Sie den Inhalt eines `multiplelookup`-Feldes in Templates ausgeben:

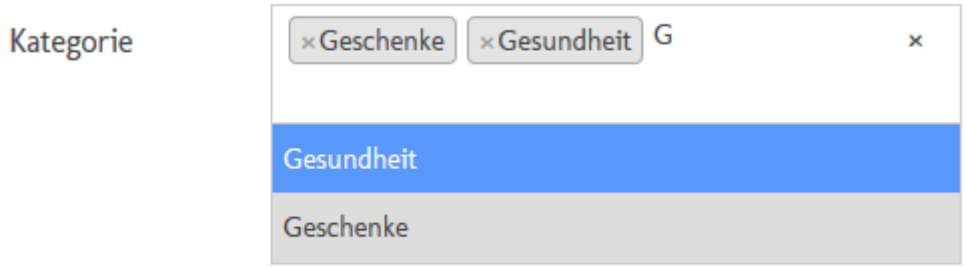

Abb. 13: `multiplelookup/choice`-Feld in der Dateneingabe

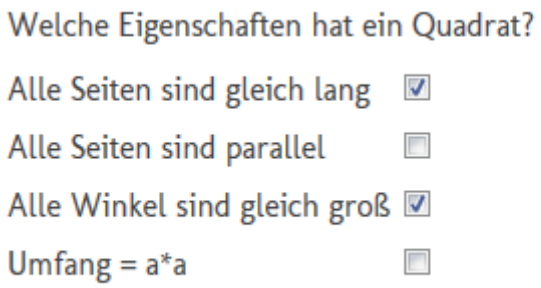

Abb. 14: `multiplelookup/checkbox`-Feld in der Dateneingabe

```
<?for lookup in record.fields.identifier.value?>
   <?printx lookup.label?>
<?end for?>
```

Shortcut:

```
<?for lookup in record.v_identifier?>
   <?printx lookup.label?>
<?end for?>
```

Um alle zugeordneten Optionen anzuzeigen, muss man eine Schleife über der Wert des Feldes benutzen. Wobei `identifier` für den *Identifizierer des Feldes* steht.

2.8 Feldtyp `multipleapplookup`

Feld, das die Datensätze einer festgelegten, ausgewählten App zur Auswahl nimmt. Hier ist in der Dateneingabe die Auswahl mehrerer Optionen möglich. Dafür muss bei der Auswahl die `Strg`-Taste der Tastatur gedrückt werden.

2.8.1 Untertyp `select`

Wird im *Fomulardesigner* angelegt als *Auswahlbox*. Hier kann unter *Auswahlen* → *Optionen einer App nutzen* eine App gewählt werden. Und Sie können festlegen, welche Felder dieser App in der Auswahl angezeigt werden. Bei *Auswahlen* → *Mehrfachauswahl zulassen* muss das Häkchen gesetzt werden. Z. B. könnten hier für eine Veranstaltung in der App „Veranstaltungen“ mehrere zuständige Mitarbeiter aus der App „Mitarbeiter“ mit den Feldern „Nachname“, „Vorname“ und „E-Mail-Adresse“ ausgewählt werden. So sieht ein `multipleapplookup/select`-Feld in der *Dateneingabe* aus.

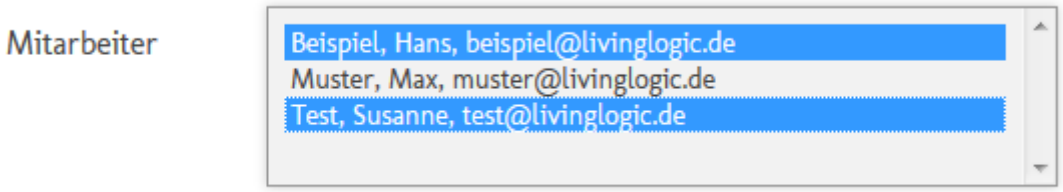

Abb. 15: `multipleapplookup/select`-Feld in der Dateneingabe

2.8.2 Untertyp `choice`

Dieser Untertyp ist sinnvoll für große Auswahlmengen. Allgemeine Beschreibung siehe Untertyp `select`. Zusätzlich muss hier bei *Auswahlen → Optimieren für große Auswahlmenge* das Häckchen gesetzt werden. In der Dateneingabe werden dann erst nachdem man Zeichen eingegeben hat, die passenden Auswahlen vorgeschlagen. Groß-/Kleinschreibung ist dabei von Bedeutung.

Auf folgende Weise können Sie den Inhalt eines `multipleapplookup`-Feldes in Templates ausgeben:

```
<?code field = record.f_identifier?>
<?for refrecord in field.value?>
   <?for reffieldidentifier in field.control.lookupcontrols?>
      <?code reffield = refrecord.fields[reffieldidentifier]?>
      <?printx reffield.control.label?>: <?printx reffield.value?>
   <?end for?>
<?end for?>
```

Um alle anzuzeigenden Felder der zugeordneten Datensätze auszugeben, muss man eine Schleife über den Wert dieses Feldes benutzen. Dabei steht `identifier` für den *Identifizierer des anzuzeigenden Feldes*, `refrecord` durchläuft die ausgewählten, verknüpften Datensätze und `reffieldidentifier` durchläuft die Felder, die für die Auswahl konfiguriert wurden. Im oben genannten Beispiel können so die Inhalte der Felder „Nachname“, „Vorname“ und „E-Mail-Adresse“ aus der verknüpften App „Mitarbeiter“ ausgegeben werden.

2.9 Feldtyp `bool`

Feld zur Eingabe und Anzeige von Ja/Nein-Werten (Boolesche Variable). Wird im *Formulardesigner* angelegt als *Häkchen*. `bool`-Felder werden verwendet um z. B. AGBs zu akzeptieren, Newsletter zu abonnieren, oder ähnliche Ja/Nein-Fragen zu beantworten. So sieht ein `bool`-Feld in der *Dateneingabe* aus.

Abb. 16: `bool`-Feld in der Dateneingabe

Auf folgende Weise können Sie den Inhalt eines `bool`-Feldes in Templates ausgeben:

```
<?if record.fields.identifier.value?>
   Ja
<?else?>
   Nein
<?end if?>
```

Shortcut:

```
<?if record.v_identifier?>
   Ja
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
<?else?>
   Nein
<?end if?>
```

Wobei `identifier` für den *Identifizierer des Feldes* steht. Gibt „Ja“ aus, wenn das Häkchen gesetzt ist, ansonsten „Nein“.

2.10 Feldtyp `file`

Feld für Datei-Uploads. Wird im Fomulardesigner angelegt als *Datei-Upload*. Hier können Sie Dateien sämtlicher Formate, Bilder, Videos, Audio-Dateien, usw. hochladen. Kann z. B. verwendet werden, um in Online-Bewerbungen das Anschreiben, Lebenslauf, Zeugnisse, usw. hochzuladen. So sieht ein `file`-Feld in der *Dateneingabe* aus.

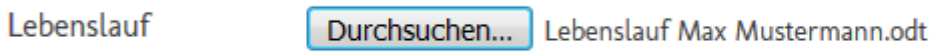

Abb. 17: `file`-Feld in der Dateneingabe

2.10.1 Untertyp `signature`

Hier können Sie direkt in der Dateneingabe mit der Maus im grauen Feld unterschreiben und diese Unterschrift dann in Templates einfügen. So sieht ein `file/signature`-Feld in der *Dateneingabe* aus.

Abb. 18: `file/signature`-Feld in der Dateneingabe

Auf folgende Weise können Sie den Inhalt eines `file`-Feldes entweder als `<img/>`-Element oder als Download-Link in Templates ausgeben:

```
<?code file = record.v_identifier?>
<?if file and file.mimetype and file.mimetype.startswith("image/")?>
   <img style="max-width: 150px; max-height: 150px" src="<?printx file.url?>"/>
   <?printx file.filename?>
<?elif file?>
   <a target="_top" href="<?printx file.url?>"><?printx file.filename?></a>
<?end if?>
```

Wobei `identifier` für den *Identifizierer des Feldes* steht. Bei `style` können Sie festlegen, in welcher Größe das Bild im Template angezeigt werden soll.

2.11 Feldtyp geo

Feld zur Erfassung von Adressen oder Geo-Koordinaten. Wird im *Fomulardesigner* angelegt als *Geodaten*. Hier können Sie unter *Allgemein* → *Dateneingabe* wählen zwischen *Adresse/Straße* (Standard) oder *Koordinaten*. Kann z. B. verwendet werden um fotografisch lohnenswerte Orte zu sammeln und zu katalogisieren. So sieht ein `geo`-Feld in der *Dateneingabe* aus.

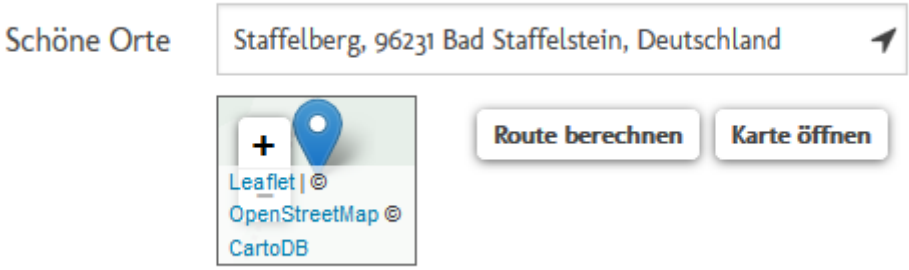

Abb. 19: `geo`-Feld in der Dateneingabe

Folgende Konfigurationen können Sie zusätzlich bei einem `geo`-Feld vornehmen:

Kartenausschnitt anzeigen
: Wird hier das Häkchen gesetzt, wird bei der Dateneingabe ein Kartenausschnitt angezeigt.

Kartenanzeige im Umkreis von
: Der Kartenausschnitt wird im Umkreis von der gewählten Option angezeigt.

Route berechnen anzeigen
: Wird hier das Häkchen gesetzt, kann bei der Dateneingabe eine Route berechnet werden.

Karte öffnen anzeigen
: Wird hier das Häkchen gesetzt, kann die Karte z. B. in google maps geöffnet werden.

Automatisch füllen mit…
: Das Feld wird automatisch mit *Aktuellem Standort* gefüllt.

Auf folgende Weise können Sie den Inhalt eines `geo`-Feldes als Link in Templates ausgeben:

```
<?if record.v_identifier?>
   <a href="https://www.google.de/maps?q=<?printx record.v_identifier.lat?>,<?printx␣
↪record.v_identifier.long?>">
      <?printx record.v_identifier.info?>
   </a>
<?end if?>
```

Wobei `identifier` für den *Identifizierer des Feldes* steht. Wird der Link im Template angeklickt, öffnet sich Google Maps mit dem ausgewählten Standort.

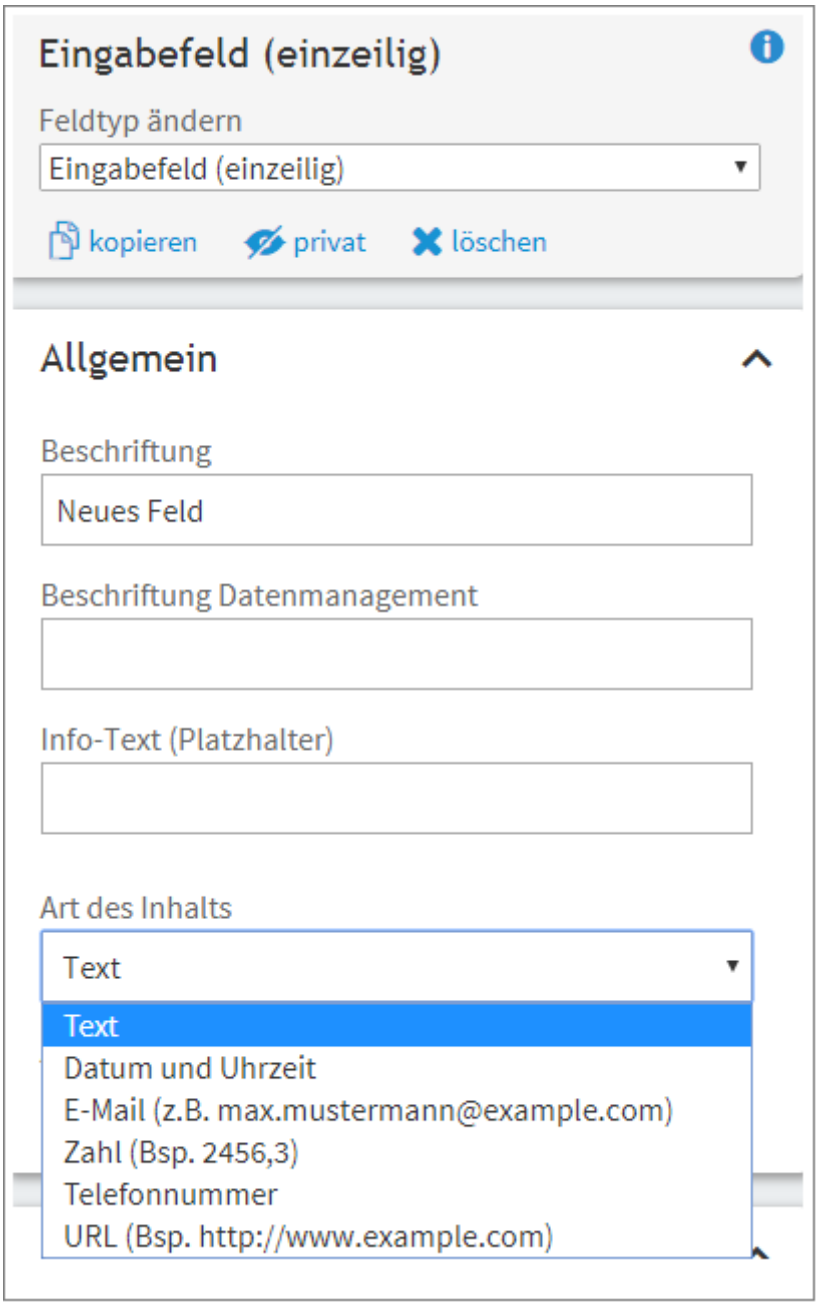

Abb. 20: Eingabefeld (einzeilig)

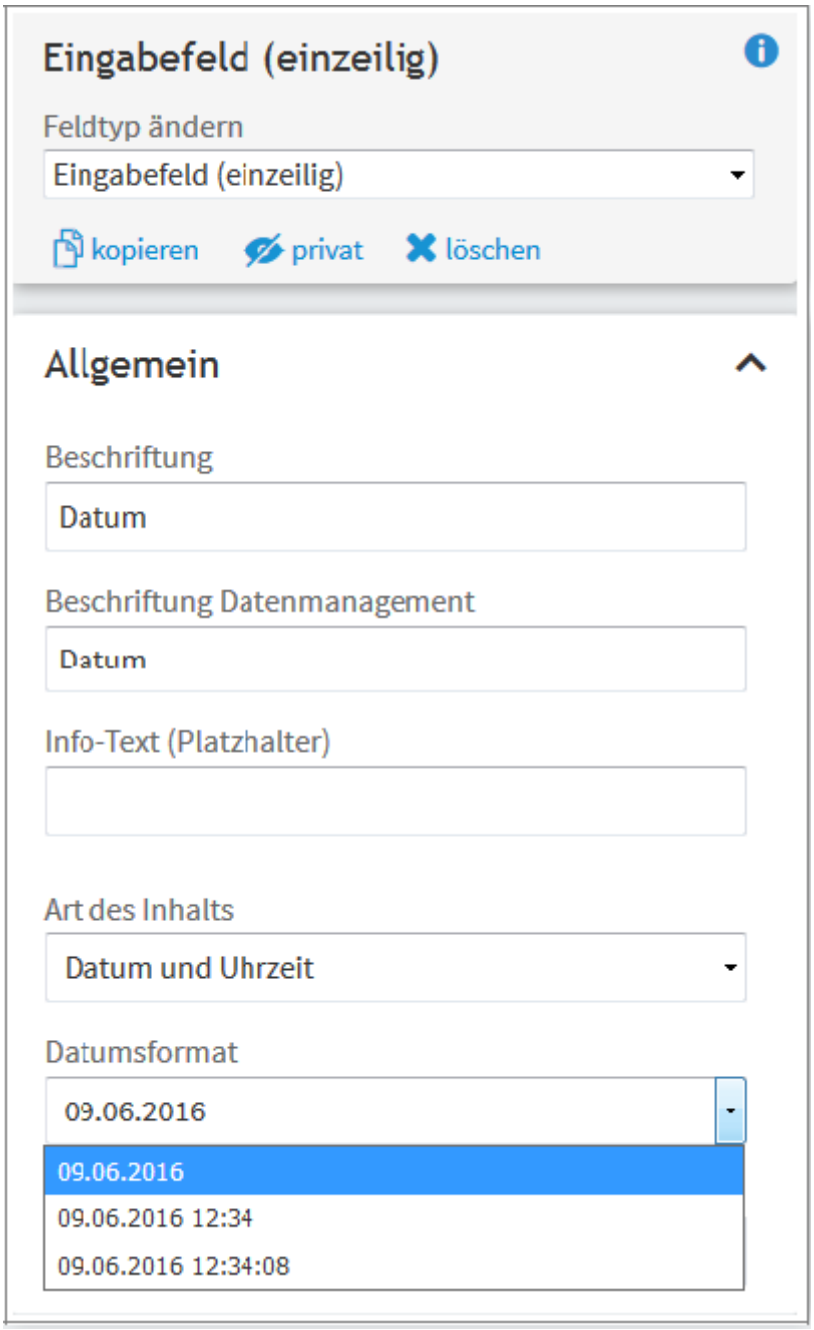

Abb. 21: Eingabefeld (einzeilig)- Datumsformat

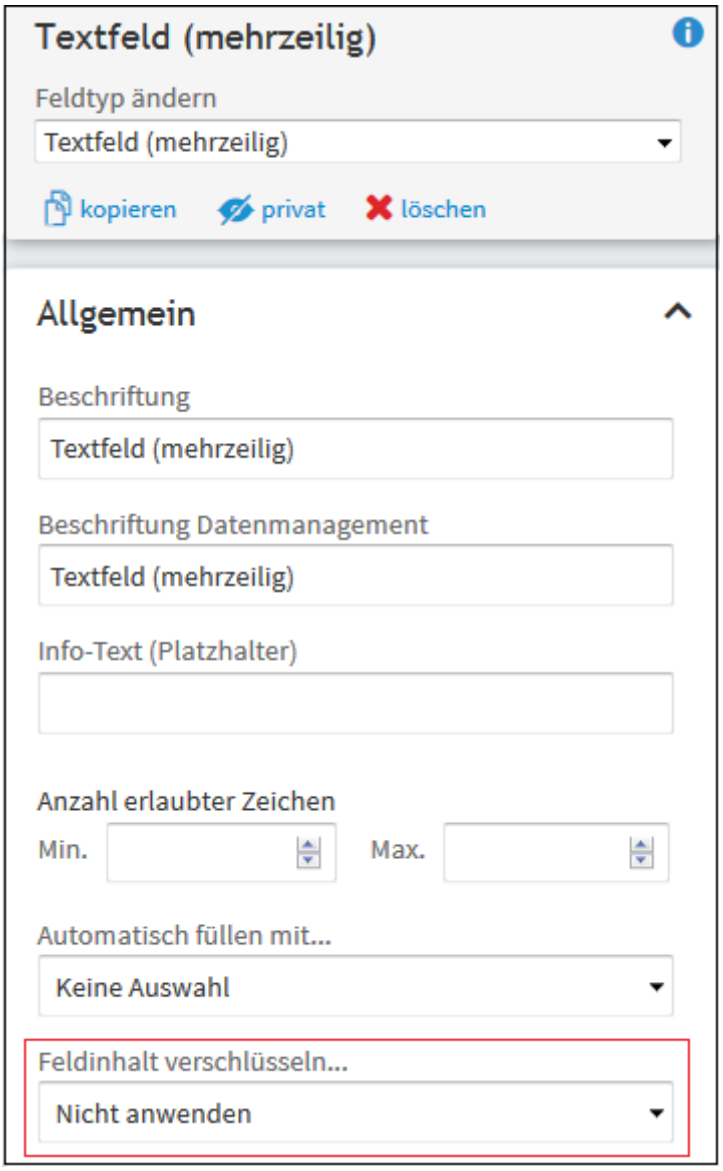

Abb. 22: Textfeld (mehrzeilig)

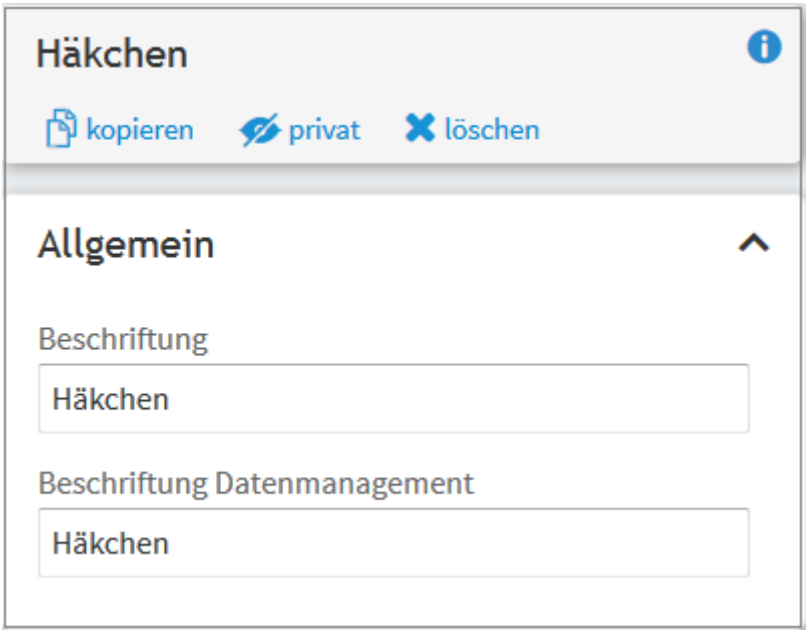

Abb. 23: Häkchen

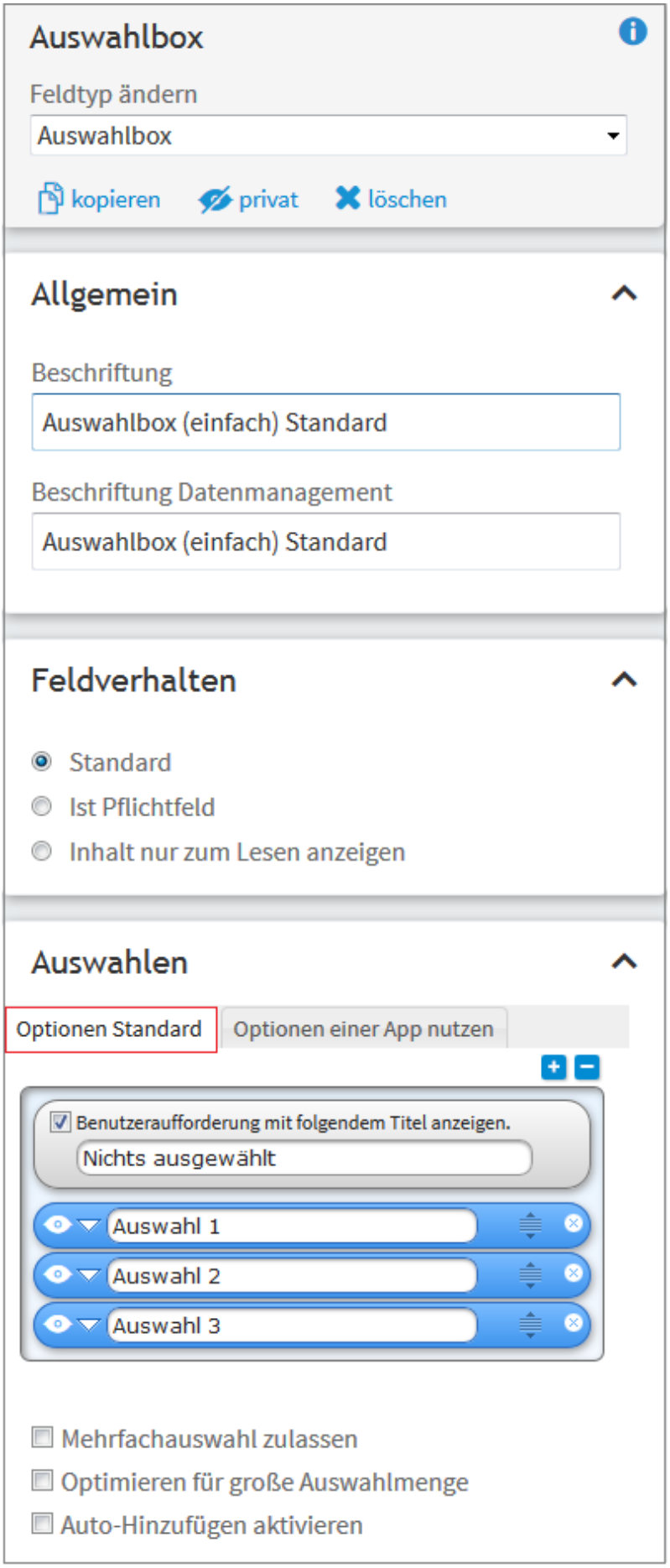

Abb. 24: Auswahlbox (einfach) - Optionen Standard

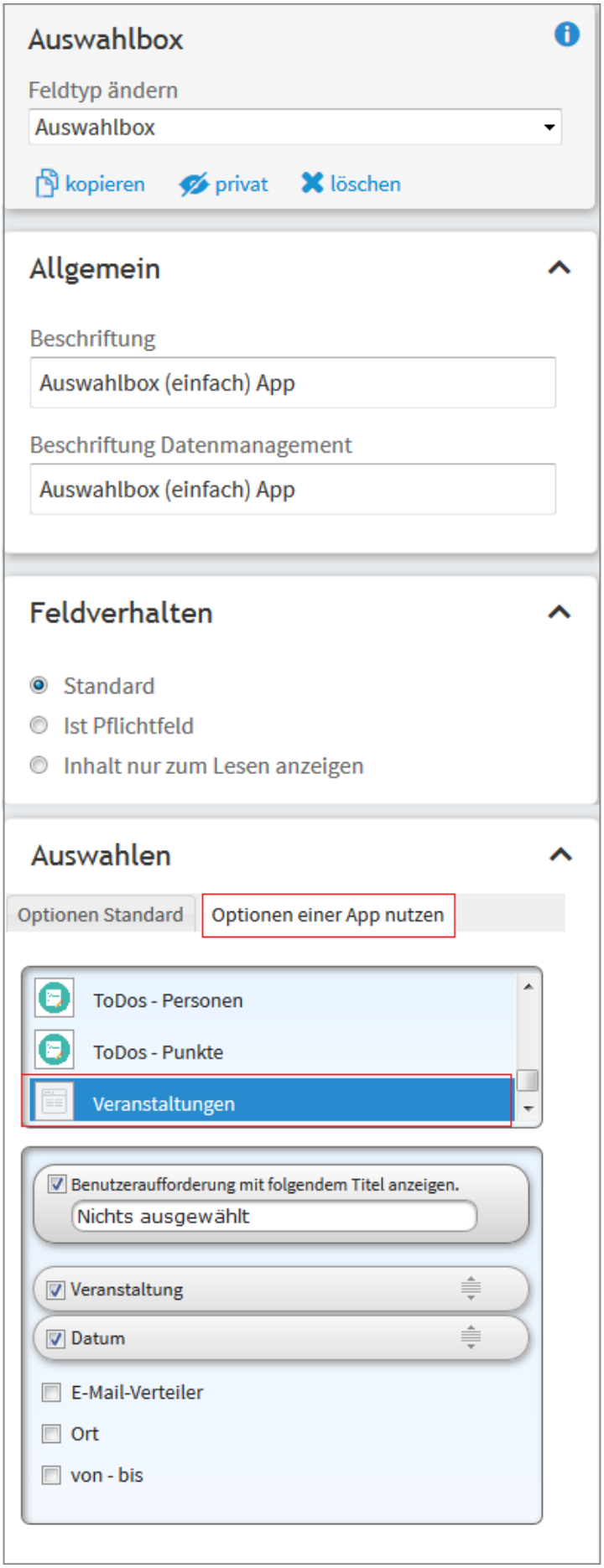

Abb. 25: Auswahlbox (einfach) - Optionen einer App nutzen

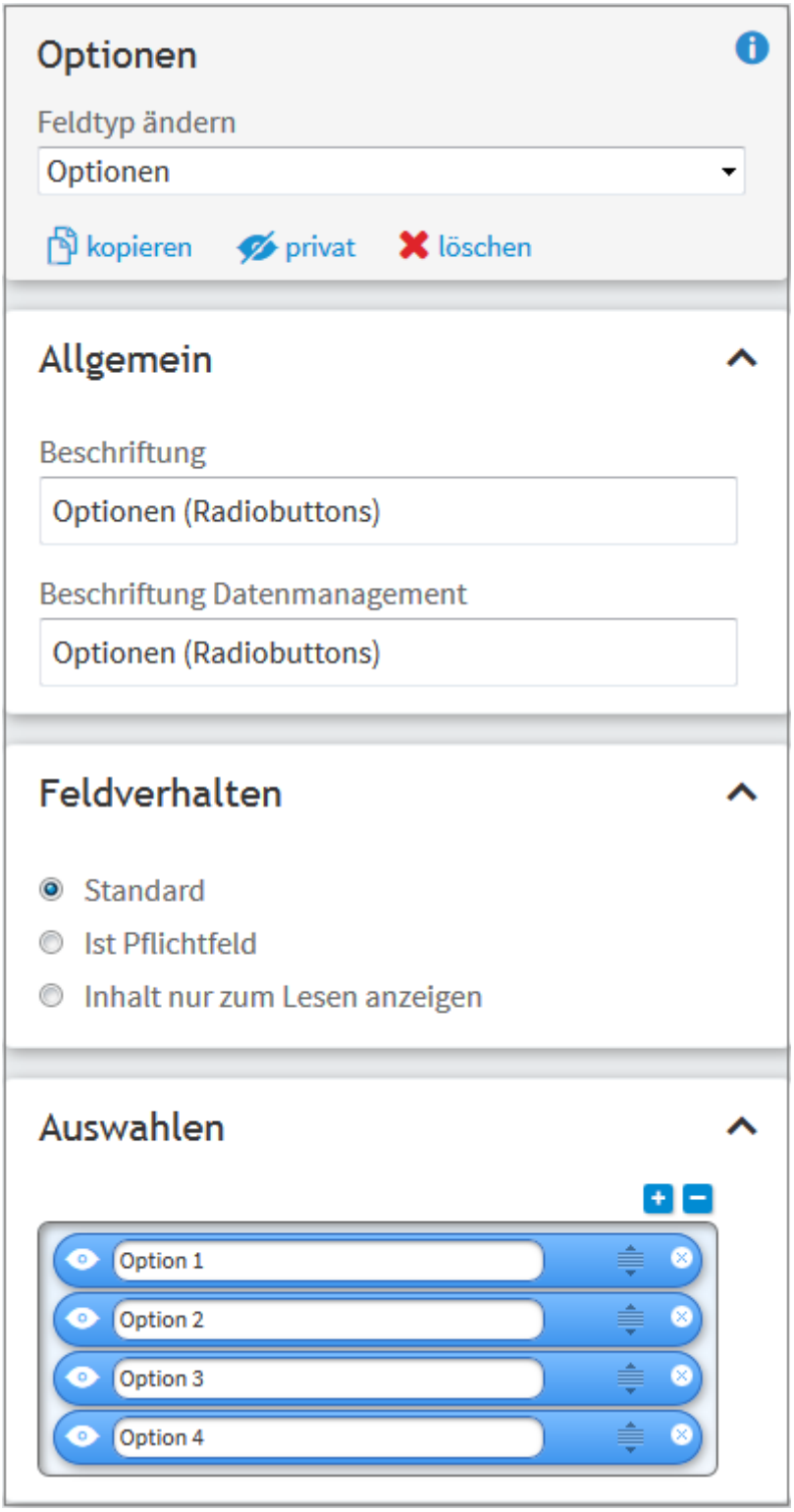

Abb. 26: Optionen einfach (Radiobuttons)

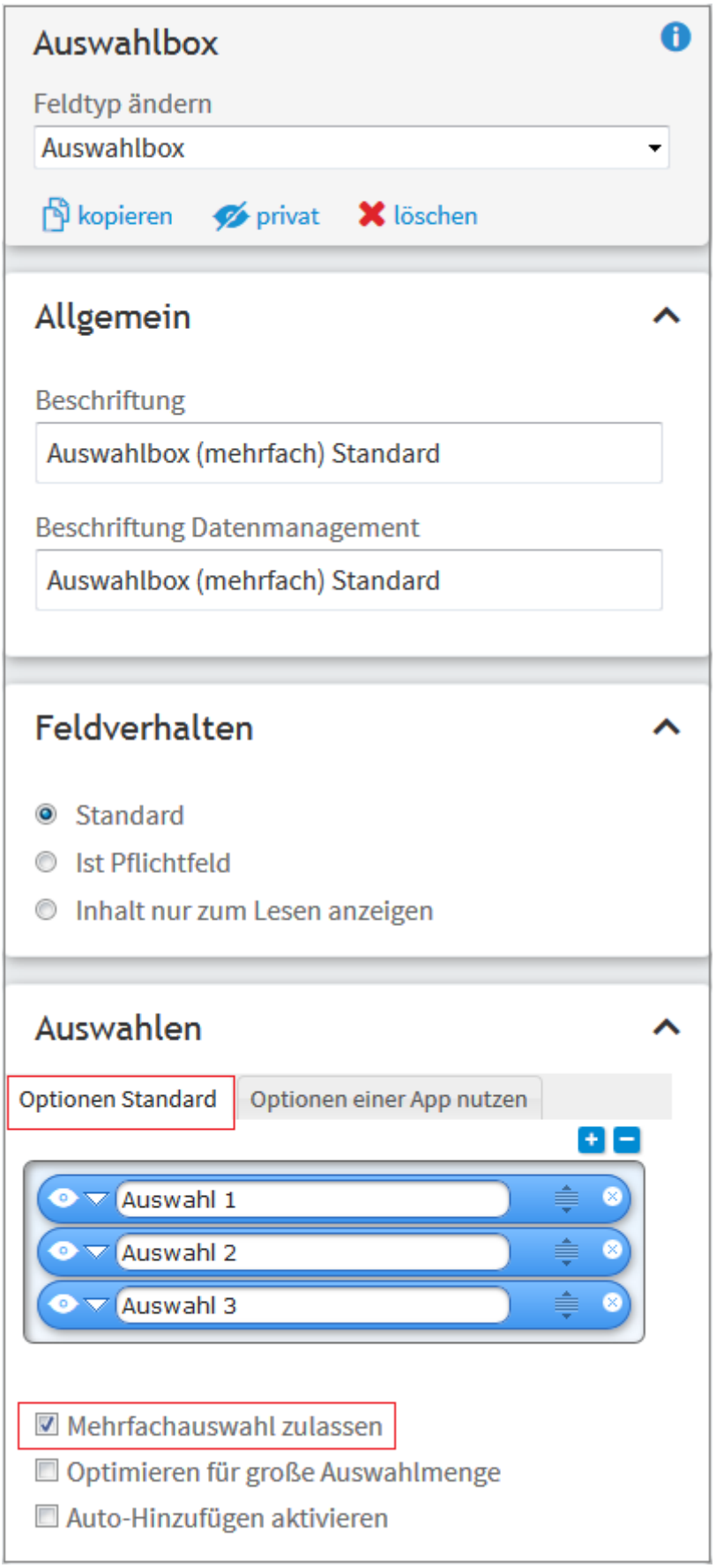

Abb. 27: Auswahlbox (mehrfach) - Optionen Standard

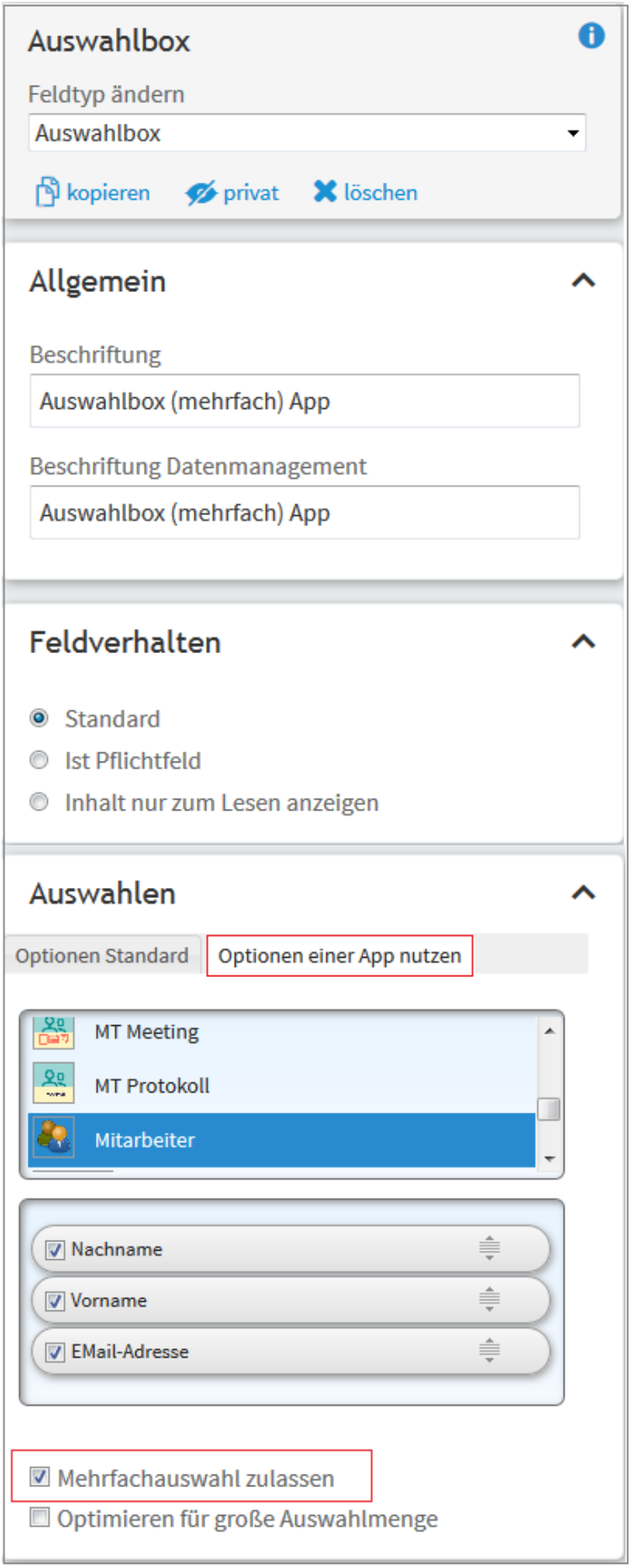

Abb. 28: Auswahlbox (mehrfach) - Optionen einer App nutzen

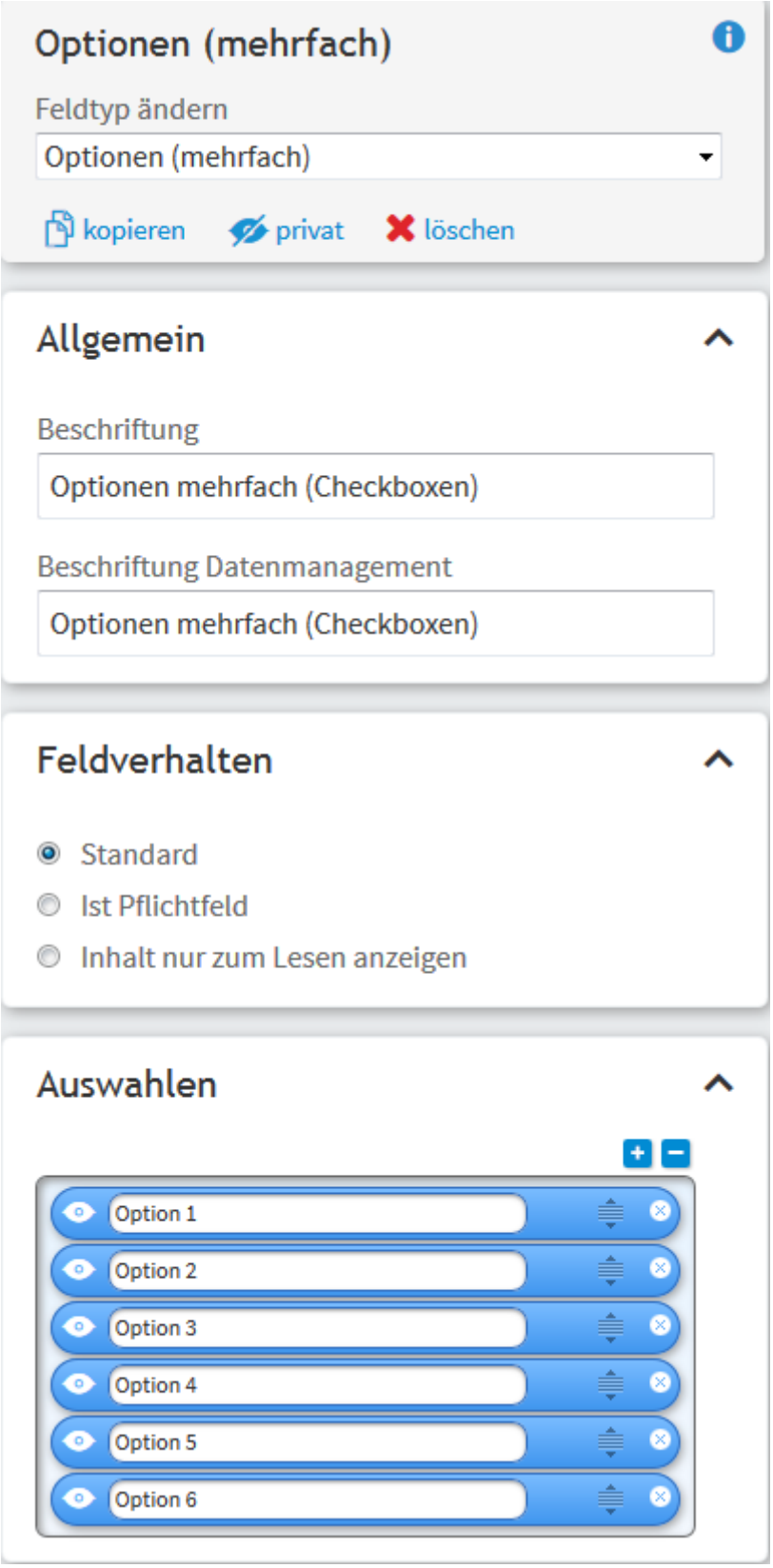

Abb. 29: Optionen mehrfach (Checkboxen)

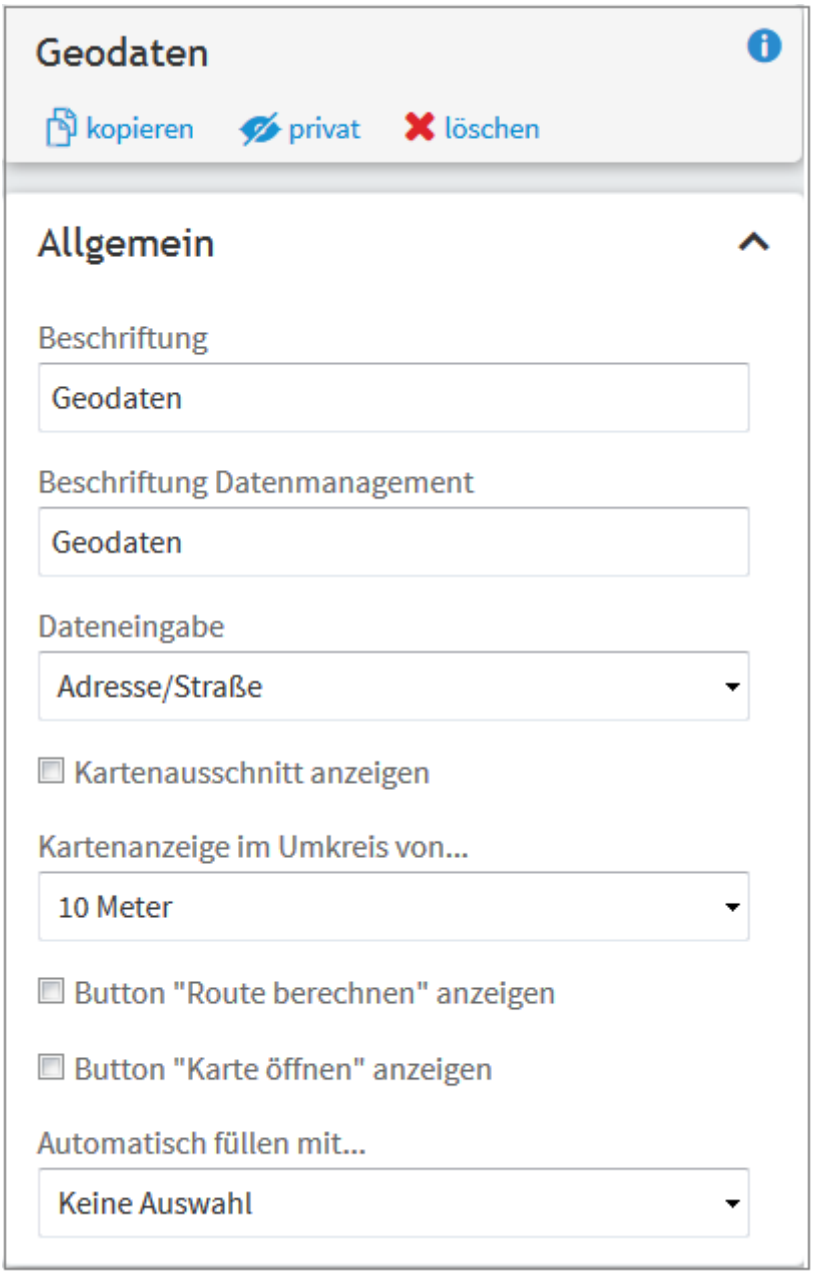

Abb. 30: Geodaten

KAPITEL 3

Maximale Feldanzahl

Pro App kann es nur eine bestimmte maximale Anzahl von Feldern eines bestimmten Typs geben. Aktuell gibt es folgende Beschränkungen:

Typen	Maximalanzahl
`string/textarea`, `string/html`	100
Sonstige `string`-Felder, `lookup`, `geo`	200
`applookup`	20
`multipleapplookup`	20
`file`	20
`bool`, `int`	100
`date`	50
`number`	100
`multiplelookup`	160

KAPITEL 4

URLs für Formulare

4.1 Systemparameter

Mit den folgenden Systemparametern im Request können Aussehen und Verhalten des Formulars gesteuert werden:

`la-header`
Wenn dieser Parameter nicht vorhanden ist oder den Wert *edit* hat, dann wird anhand der Benutzer-Berechtigungen geprüft, ob der User auf der Formularseite die Überschriften und das Bild ändern kann und falls ja, dann diese Möglichkeit auch angeboten. Hat er den Wert *yes* wird der Header lediglich angezeigt (ohne Editiermöglichkeit), bei *no* nicht angezeigt.

`la-frameless`
Wenn dieser Parameter den Wert *yes* hat, wird das Formular ohne umgebende Rahmen-Tags ausgeliefert. Wird beispielsweise bei der Einbettung in iframes genutzt.

`la-result-url`
Dieser Parameter kann genutzt werden, um eine andere URL nach der erfolgreichen Formulareingabe anzuzeigen. Es sind nur URLs mit dem gleichen Host und Port zulässig.

`la-error-url`
Dieser Parameter kann genutzt werden, um nach einem Fehler bei der Formulverarbeitung eine andere URL anzuzeigen. Es sind nur URLs mit dem gleichen Host und Port zulässig.

KAPITEL 5

Anzeige-Templates

In *Anzeige-Templates* können Sie beliebig viele sogenannte „UL4-Templates“ anlegen. In diesen Templates können Sie auf die vorhandenen Datensätze zugreifen und mit den Daten arbeiten, sie analysieren und ausgeben. Beispielsweise können die Daten mit Anzeige-Templates nach bestimmten Kriterien ausgewertet und die Ergebnisse in tabellarischer Form dargestellt werden.

Je nach Funktion Ihrer LivingApp sind die Anwendungsmöglichkeiten sehr vielfältig. Anzeige-Templates können sowohl für einfache Ausgaben von Datensätzen dienen, als auch alle möglichen Auswertungen der Daten vornehmen.

Anzeige-Templates sind also für alles nützlich, was mit Datenanalyse, -Übersicht und -Ausgabe zu tun hat.

5.1 Erstellung eines Anzeige-Templates

Um ein Anzeige-Template für Ihre LivingApp zu erstellen, wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Anzeige-Templates* und dann auf *Neues Anzeige-Template hinzufügen*.

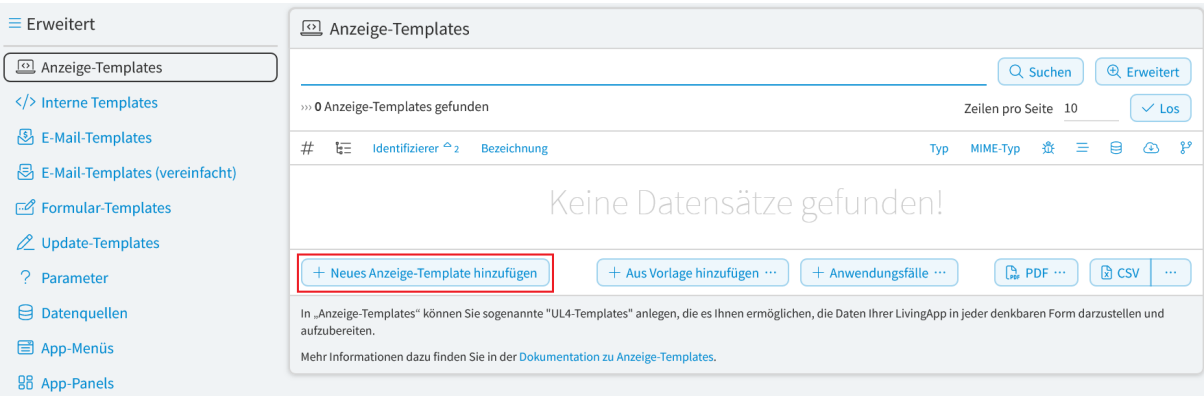

Abb. 1: Anzeige-Template hinzufügen

Im nun geöffneten Fenster sehen Sie die *Eingabeansicht* des Anzeige-Templates.

Identifizierer

Der Identifizierer ist eine eindeutige Kennung für das Template. Er kann verwendet werden, um das Template abzurufen. Der Identifizierer darf nur Buchstaben, Ziffern und _ enthalten.

Anzeige-Templates Bezeichnung:

+ Hinzufügen

Speichern Abbrechen Hilfe

* Identifizierer

Eine eindeutige Kennung für dieses Anzeige-Template. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Bezeichnung

Die Bezeichnung für den Link im Datenmanagement und im Menü.

Icon laptop-code

Das Icon für den Link im Datenmanagement und im Menü.

Quelltext

1

Hilfe – Element einfügen ··· Vollbild

Verfügbare Variablen: app, record, datasources. Verfügbare globale Variablen: globals, la

Anzeige-Eigenschaften

* Typ Liste Liste (Standard) Liste (mit Menüeintrag) Detail Detail (Ergebnisseite) Detail (mit Menüeintrag im Datenmanagement) Support

Ein „Listen-Template" zeigt eine Liste von Datensätzen an, ein „Detail-Template" einen einzelnen. Ein „Support-Template" kann für andere Anzeigen verwendet werden, die keine Datensätze benötigen.

MIME-Typ (Nichts ausgewählt) text/html text/plain text/css text/javascript text/xml text/calendar application/json

Das Dateiformat das von diesem Template ausgegeben wird.

Berechtigung

* Zugriff erlauben für alle Benutzer

Speichern Abbrechen Hilfe

Abb. 2: Anzeige-Template Eingabeansicht

Bezeichnung und Icon
Die Bezeichnung für den Link und das Icon zum Template im Datenmanagement und im Menü.

Quelltext
Hier erfolgt die *Formulierung des Templates*.

Wird im Quelltext ein `<?whitespace?>`-Tag oder ein `<?doc?>`-Tag eingegeben, dann erscheint unter *Template-eigenschaften* folgendes:

Whitespace
Whitespace konfiguriert, wie Einrückungen/Zeilenvorschübe im Template-Quelltext behandelt werden (wird automatisch aus dem `<?whitespace?>`-Tag im Template-Quelltext extrahiert).

Beschreibung
Beschreibung des Templates (aus dem `<?doc?>`-Tag im Template-Quelltext extrahiert)

Element einfügen ...
Unter *Element einfügen*, finden Sie vorgefertigte Bausteine in HTML-Format, die durch anklicken bei der aktuellen Cursor-Position eingefügt werden.

Abb. 3: Anzeige-Template - Element einfügen

Typ
Es muss ein Template-Typ ausgewählt werden.

Liste
Es kann auf alle Datensätze, die in der *Datenquelle* konfiguriert sind, zugegriffen werden. Soll das Template eine Auswertung aller Daten beinhalten, so muss dieser Typ gewählt werden.

Liste Standard
Beschreibung siehe oben unter *Liste*. Dieses Template wird ausgeliefert, wenn die URL keinen Parameter `template` hat. Darüber können Sie unter *Aufruf eines Anzeige-Templates* mehr lesen. (Es kann nur ein Anzeige-Template mit dem Typ Liste (Standard) geben.)

Liste (mit Menüeintrag)
Hier steht der Link zum Anzeige-Template auch in der Menüzeile und in der Datenliste (in der Fußzeile der Liste) zur Verfügung.

Detail
Es kann nur auf einen einzelnen Datensatz zugegriffen werden.

Detail (Ergebnisseite)
Hier wird dieses Anzeige-Template verwendet um den Inhalt der Ergebnisseite beim Abspeichern eines Formulars zu erzeugen. In dem Fall muß das Template nur den Seiteninhalt erzeugen, der Seitenrahmen wird automatisch erzeugt. (Es kann nur ein Anzeige-Template mit dem Typ Detail (Ergebnisseite) geben.)

Detail (mit Menüeintrag im Datenmanagement)
Hier steht der Link zum Anzeige-Template auch in der Datenliste zur Verfügung (bei jedem einzelnen Datensatz).

Support
Ein Support-Template kann für andere Anzeigen verwendet werden, die keine Datensätze benötigen. Z. B. für CSS oder Javascript.

MIME-Typ
Dateiformat, in welchem das Template ausgegeben wird. Hier kann z. B. das CSS das in einem Listen- oder Detail-Template verwendet wird ausgelagert und anschließend im HTML-Template eingebunden werden.

Berechtigung für
Hier kann ausgewählt werden, wer das Template aufrufen kann. In der jeweiligen LivingApp können unter *Konfiguration* → *Berechtigungen* verschiedenen Personen unterschiedliche Rechte zugewiesen werden. Je nachdem, wer welche Rechte hat und für wen das Template freigeschaltet wurde, haben unterschiedliche Personen Zugriff auf das jeweilige Template.

Alle Benutzer
Das Template ist frei zugänglich.

Eingeloggte Benutzer
Um das Template aufrufen zu können, muss diejenige Person auf LivingApps eingeloggt sein.

User mit Rechten auf der Applikation
Bei dieser Auswahl kann das Template nur durch Personen aufgerufen werden, die irgendein Recht an der App besitzen. Dies kann sowohl ein Einzelrecht sein, als auch eine Rolle (z. B. Minimal, Standard oder Administrator).

User mit Datenverarbeitungsrechten auf der Applikation
Bei dieser Auswahl kann das Template nur durch Personen aufgerufen werden, die die Berechtigung haben, in die Datenliste der App zu gelangen.

User mit Admin-Rechten an der Applikation
Bei dieser Auswahl kann das Template nur durch Personen aufgerufen werden, die Administrationsrechte an der App haben.

Ein Anzeige-Template ist in der Regel aus zwei Teilen aufgebaut. Der erste Teil führt die Datenaufbereitung durch. Hier wird auf die Datensätze zugegriffen und die gewünschte Auswertung berechnet. Der zweite Teil sorgt für die HTML-Ausgabe der ermittelten Daten. Dazu können sämtliche HTML-Elemente verwendet werden. Zur Formatierung kann zudem CSS eingesetzt werden. Hierzu können Sie im Kapitel *Formulierung des HTML-Codes* mehr lesen.

5.1.1 Formulierung des Templates

Um auf die Daten Ihrer App zugreifen zu können, benötigten Sie die Templatesprache UL4. Die UL4-Dokumentation finden Sie unter https://python.livinglogic.de/UL4.html. Die Ausgabe der Daten erfolgt über die *LivingAPI*. Im zugehörigen Kapitel sind die Objekte ausführlich beschrieben.

Durchlaufen von Datensätzen bei Templates vom Typ *Liste*
Die Datensätze der jeweiligen App sind im *Record*-Dictionary der *App* (`app.records`) gespeichert.

Damit die Datensätze dem Template zur Verfügung stehen, muss dafür eine Datenquelle angelegt werden. Über die Datenquelle können Sie dann auch z. B. auf *App-Parameter* zugreifen oder Daten mehrerer Apps ausgeben. Näheres können sie in der Dokumentation der *Datenquelle* nachlesen.

Wählen sie in der Datenquelle dieselbe App aus, in der das Template erstellt wird, durchlaufen Sie mit folgender `for`-Schleife alle vorhandenen Datensätze.

```
<?for r in app.records.values()?>
   Anweisungen
<?end for?>
```

Möchten Sie Datensätze aus anderen Apps in Ihrem Template anzeigen, durchlaufen Sie die Datensätze mit folgender `for`-Schleife:

```
<?for r in datasources.beispiel.app.records.values()?>
   Anweisungen
<?end for?>
```

Wobei `beispiel` für den Identifizierer der Datenquelle steht, die Sie in Ihrem Template angelegt haben.

Ausgabe von Feldwerten

Die Attribute der Datensätze, wie das Erstellungsdatum, die Felder oder Anhänge sind im *Record*-Objekt gespeichert. Im *Field*-Objekt befinden sich die Werte des Feldes. *Control*-Objekte beinhalten die Metadaten eines Feldes einer Applikation. Je nach Feld-Typ ist dies ein spezieller Objekt-Typ (z.B. TextControl für ein normales einzeiliges Text-Feld). Zu den Feldtypen mit Ihren Untertypen können Sie im Kapitel *Übersicht der Feldtypen* mehr lesen.

Auf die einzelnen Attribute kann per Punktnotation zugegriffen werden. Zur Ausgabe von Daten aus dem Datensatz verwenden Sie die Anweisung `printx` und geben das gewünschte Element des Datensatzes mit Punktnotation an.

Der Feldinhalt eines `string/text`-Feldes kann dann beispielsweise innerhalb der obigen `for`-Schleife und verkapselt in einer `if`-Anweisung wie folgt ausgegeben werden:

```
<?if r.fields.identifier.value?>
   <?printx r.fields.identifier.value?>
<?end if?>
```

Wie die Werte anderer Feldtypen ausgegeben werden, können Sie im Kapitel *Übersicht der Feldtypen* nachlesen.

Desweiteren stehen die Werte auch über „Shortcut“-Attribute zur Verfügung. `record.fields.identifier` steht auch direkt als `record.f_identifier` zur Verfügung. `record.fields.identifier.value` steht auch direkt als `record.v_identifier` zur Verfügung. Wobei `identifier` jeweils für den Identifizierer des Feldes steht, das sie ausgeben möchten.

Auf den Feldwert kann analog mit `<?printx r.fields['identifier'].value?>` zugegriffen werden.

Folgender Code durchläuft alle Datensätze und alle Felder und gibt, wenn ein Wert vorhanden ist, dann deren Werte aus:

```
<?for r in app.records.values()?>
   <?for f in r.fields.values()?>
      <?if f.value is not None?>
         <?printx f.value?>
      <?end if?>
   <?end for?>
<?end for?>
```

5.1.2 Prüfung von Feldwerten auf Inhalt

Bevor Sie den Inhalt eines Formularfeldes ausgeben, können Sie prüfen, ob ein Inhalt eingegeben wurde. Dafür ist die Ausgabe mit einer `if`-Bedingung zu umschließen, die prüft, ob eine Angabe im betrachteten Feld vorhanden ist.

Als „leer“ gelten:

- der boolsche Wert False
- leere Zeichenketten
- leere Listen
- leere Dictionaries
- numerische Null-Werte
- der Wert `None`

Es gibt folgende drei Möglichkeiten, den Feld-Wert zu überprüfen. Wobei `identifier` für den Identifizierer des jeweiligen Feldes steht.

Prüfen, ob ein Wert eingegeben wurde:

```
<?if record.fields.identifier.value?>
   Anweisung für den Fall, dass ein Wert vorhanden ist
<?end if?>
```

Shortcut:

```
<?if record.v_identifier?>
   Anweisung für den Fall, dass ein Wert vorhanden ist
<?end if?>
```

Besonderheiten:

Bei leeren Listen in `multiplelookup/multipleapplookup`-Feldern wird der `if`-Block in obiger Logik nicht ausgeführt, verhält sich also korrekt. Die Zahl „0“ in Feldern vom Typ `number` ist `False` und wird deshalb nicht ausgeführt.

Prüfen, ob der Wert nicht `None` ist:

```
<?if record.fields.identifier.value is not None?>
   Anweisung für den Fall, dass ein Wert vorhanden ist
<?end if?>
```

Shortcut:

```
<?if record.v_identifier is not None?>
   Anweisung für den Fall, dass ein Wert vorhanden ist
<?end if?>
```

Besonderheiten:

Bei leeren Listen in `multiplelookup/multipleapplookup`-Feldern wird der if-Block ausgeführt. Die Zahl „0“ in Feldern vom Typ `number` ist `True` und wird ausgeführt.

Methode `is_empty()` von *Field*-Objekt:

```
<?if not record.fields.identifier.is_empty()?>
   Anweisung für den Fall, dass ein Wert vorhanden ist
<?end if?>
```

Shortcut:

```
<?if not record.f_identifier.is_empty()?>
   Anweisung für den Fall, dass ein Wert vorhanden ist
<?end if?>
```

Je nach Typ ist das Richtige implementiert.

Bei leeren Listen in `multiplelookup/multipleapplookup`-Feldern wird der `if`-Block nicht ausgeführt. Die Zahl „0“ in Feldern vom Typ `number` ist `True` und wird ausgeführt.

Über die Verwendung der Methode `is_empty()` können Sie in der Dokumentation der LivingAPI - Objekt-Typen unter *Field* mehr nachlesen.

5.1.3 Whitespace

Befinden sich im Template UL4-Ausdrücke oder werden andere Templates eingebunden, empfiehlt es sich zur besseren Darstellung, einen `<?whitespace?>`-Tag ins Template einzufügen. Folgende drei Varianten stehen zur Auswahl:

`<?whitespace keep?>`
: Leerzeichen und Zeilenvorschübe werden 1:1 übernommen.

`<?whitespace strip?>`
: Leerzeichen und Zeilenvorschübe zwischen den Tags werden ignoriert.

`<?whitespace smart?>`
: Bei Zeilen die außer einem `<?print?>`-, `<?printx?>`- oder `<?render?>`-Tag nur Whitespace beinhalten, wird die Einrückung und der Zeilenvorschub ignoriert. Weiterhin wird die zusätzliche Einrückung ignoriert, die innerhalb eines `<?for?>`-, `<?if?>`-, `<?elif?>`-, `<?else?>`- oder `<?def?>`-Tags verwendet wird.

5.1.4 Formulierung des HTML-Codes

Abgesehen von Daten der Datensätze können Sie natürlich auch alle anderen HTML-Elemente in Ihre Auswertung integrieren. Sie können einen gesamten HTML-Quellcode mit `head`- und `body`-Bereich als Anzeige-Template eingeben und an den entsprechenden Stellen UL4 verwenden, um auf Datensätze zuzugreifen, sie zu verarbeiten und zu analysieren. Mit HTML-Notation werden diese Daten dann beispielsweise in Tabellen- oder Listenform dargestellt. Außerdem können Überschriften, Verlinkungen und Bilder in Ihre Auswertung eingefügt werden. Ihnen stehen alle Möglichkeiten der HTML-Notation zur Verfügung.

Um gewisse Formatierungen festzulegen und um die optische Darstellung der Auswertung ansprechend zu gestalten, kann zudem CSS verwendet werden.

Mehr Informationen zu HTML und CSS sowie ausführliche Anleitungen finden Sie unter https://wiki.selfhtml.org/

5.1.5 CSS-Template

Für das nachfolgende Anwendungsbeispiel wurde ein CSS-Template angelegt und im Anzeigetemplate `teilnehmerliste` eingebunden.

Um ein CSS-Template anzulegen, wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Anzeige-Templates*. Klicken Sie auf *neues Anzeige-Template hinzufügen*.

Im nun geöffneten Fenster wurden für das Anwendungsbeispiel folgende *Konfigurationen* vorgenommen:

Wie das CSS-Template im Anzeigetemplate `teilnehmerliste` eingebunden wird, können Sie im *Anwendungsbeispiel* nachlesen.

5.1.6 Ergebnisseite

Sie haben die Möglichkeit ein Anzeigetemplate zu erstellen, das dem Teilnehmer/ der Teilnehmerin nach der Eingabe eines Datensatzes oder nach einer Änderung über Formular, als Ergebnisseite angezeigt werden soll. Diese Ergebnisseite könnte folgendermaßen aussehen.

Um eine Ergebnisseite anzulegen, wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Anzeige-Templates*. Klicken Sie auf *Neues Anzeige-Template hinzufügen*.

Anzeige-Templates | CSS | Bezeichnung: CSS | Typ: Support | MIME-Typ: text/css | 1/4

Bearbeiten | Datenquellen | Externe Datenquellen | Parameter | Meldungen in aktueller Version | Versionen

Speichern | Wiederherstellen | Hilfe

* Identifizierer: CSS

Eine eindeutige Kennung für dieses Anzeige-Template. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Bezeichnung: CSS

Die Bezeichnung für den Link im Datenmanagement und im Menü.

Icon: laptop-code

Das Icon für den Link im Datenmanagement und im Menü.

Quelltext:

```
1  body
2  {
3      font-family: monospace;
4      text-align: center;
5  }
6
7  h1, h2
8  {
9      color: #bd0b20;
10 }
11
12 table
13 {
14     border-color: #999;
15     border-style: solid;
16     border-width: 1px 1px 0 0;
17 }
18
19 table th,
20 table td
21 {
22     padding: 0.2em 0.5em;
23     border-color: #999;
24     border-style: solid;
25     border-width: 0 0 1px 1px;
26 }
27
28 .zentriert
29 {
30     text-align: center;
31 }
```

Hilfe – Element einfügen ··· | Vollbild

Verfügbare Variablen: app, record, datasources. Verfügbare globale Variablen: globals, la

Anzeige-Eigenschaften

* Typ: ○ Liste ○ Liste (Standard) ○ Liste (mit Menüeintrag) ○ Detail ○ Detail (Ergebnisseite) ○ Detail (mit Menüeintrag im Datenmanagement) ◉ Support

Ein „Listen-Template" zeigt eine Liste von Datensätzen an, ein „Detail-Template" einen einzelnen. Ein „Support-Template" kann für andere Anzeigen verwendet werden, die keine Datensätze benötigen.

MIME-Typ: ○ (Nichts ausgewählt) ○ text/html ○ text/plain ◉ text/css ○ text/javascript ○ text/xml ○ text/calendar ○ application/json

Das Dateiformat das von diesem Template ausgegeben wird.

Berechtigung

* Zugriff erlauben für: alle Benutzer

Abb. 4: Anzeige-Template CSS - Konfiguration

Sehr geehrte(r) Teilnehmer(in),

Sie haben sich erfolgreich mit 10 Personen zur Veranstaltung: Word für Fortgeschrittene angemeldet.

Eine Anmeldebestätigung per E-Mail wird an Sie versendet.

Mit freundlichen Grüßen
Ihr Veranstaltungsteam

Abb. 5: Anzeige-Template Ergebnisseite

Im nun geöffneten Fenster werden folgende Konfigurationen vorgenommen:

Bei *Typ* muss *Detail (Ergebnisseite)* gewählt werden.

Wie eine Ergebnisseite aufgerufen wird, können Sie unter *Aufruf eines Anzeige-Templates* nachlesen.

5.1.7 Aufruf eines Anzeige-Templates

Wenn Sie das Formular einer beliebigen LivingApp öffnen, sehen Sie in der URL hinter `apps/` eine Kombination aus Buchstaben und Zahlen. Dies ist die App-UUID (Universally Unique Identifier), der eindeutige Identifizierer Ihrer App. Im Formular steht hinter der UUID `/new`, da hier ein neuer Datensatz angelegt werden kann. An das `/new` ist die View-ID angehängt. Der Link zum Formular der App könnte beispielsweise wie folgt lauten:

```
https://my.living-apps.de/gateway/apps/563a2616591c6ca7a9800ef2/new?
↪view=584f56445afcda56b7716252
```

Wobei `563a2616591c6ca7a9800ef2` die App-UUID ist und `584f56445afcda56b7716252` die View-ID. Ein Listen-Template, bei dem bei *Standard* das Häkchen gesetzt ist, erreichen Sie, indem Sie `/new?view=584f56445afcda56b7716252` aus der URL löschen und keinen weiteren Parameter anhängen. Im Beispiel wäre das folgende URL:

```
https://my.living-apps.de/gateway/apps/563a2616591c6ca7a9800ef2
```

Haben Sie hingegen bei Ihrem Listen-Template das Häkchen bei *Standard* nicht gesetzt, erreichen Sie dieses Template, indem Sie `?template=mein_identifizierer` hinter die App-UUID an die URL anhängen. Heißt der Template-Identifizierer z. B. `uebersicht`, so lautet die Beispiel-URL:

```
https://my.living-apps.de/gateway/apps/563a2616591c6ca7a9800ef2?template=uebersicht
```

Wird bei einem Detail-Template bei *Ergebnisseite* das Häkchen gesetz, dann wird nach Eingabe oder Änderung eines Datensatzes über Formular, dieses Template als Ergebnisseite angezeigt. Der Identifizierer wird in dem Fall automatisch an die URL angehängt. Diese würde dann mit dem Template-Identifizierer `beispiel` wie folgt lauten:

```
https://my.living-apps.de/gateway/apps/563a2616591c6ca7a9800ef2/
↪566832210d3450b5949b9c00?template=beispiel
```

Im vorherigen Beispiel wird außerdem sichtbar, dass auf Datensatz-Ebene hinter die App-UUID eine weitere ID (dat_id) gehängt wird, die für jeden Datensatz eindeutig ist.

Sie haben auch die Möglichkeit einen Link zum Aufruf Ihres Anzeige-Templates zu erstellen. Darüber können Sie in der Dokumentation der *App-Menüs*, *Benutzer-Menüs* bzw. *App-Panels* oder *Benutzer-Panels* mehr lesen.

Anzeige-Templates ergebnisseite Bezeichnung: ergebnisseite — Ergebnisseite Typ: Detail (Ergebnis 2/4

Bearbeiten | Datenquellen | Externe Datenquellen | Parameter | Meldungen in aktueller Version | Versionen

+ Neues Anzeige-Template hinzufügen | Kopieren | — Löschen | Meldungen ··· | Hilfe

* Identifizierer: ergebnisseite

Eine eindeutige Kennung für dieses Anzeige-Template. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Bezeichnung: ergebnisseite

Die Bezeichnung für den Link im Datenmanagement und im Menü.

Icon: laptop-code

Das Icon für den Link im Datenmanagement und im Menü.

Quelltext:

```
1  <?doc Ergebnisseite?>
2  <html>
3  ····<body>
4  ········<p>Sehr geehrte(r) Teilnehmer(in),</p>
5  ········<p>Sie haben sich erfolgreich mit <?printx record.v_anzahl_personen2?> Personen zur Veranstaltung: <?printx
       record.v_veranstaltung2.v_veranstaltung?> angemeldet.</p>
6  ········<p>Eine Anmeldebestätigung per E-Mail wird an Sie versendet.</p>
7  ········<p>
8  ············Mit freundlichen Grüßen<br />
9  ············Ihr Veranstaltungsteam
10 ········</p>
11 ····</body>
12 </html>
```

Hilfe — | Element einfügen ··· | Vollbild

Verfügbare Variablen: app, record, datasources. Verfügbare globale Variablen: globals, la

Template-Eigenschaften > Beschreibung

Anzeige-Eigenschaften

* Typ: ○ Liste ○ Liste (Standard) ○ Liste (mit Menüeintrag) ○ Detail ◉ Detail (Ergebnisseite) ○ Detail (mit Menüeintrag im Datenmanagement) ○ Support

Ein „Listen-Template" zeigt eine Liste von Datensätzen an, ein „Detail-Template" einen einzelnen. Ein „Support-Template" kann für andere Anzeigen verwendet werden, die keine Datensätze benötigen.

Berechtigung

* Zugriff erlauben für: alle Benutzer

Abb. 6: Anzeige-Template Ergebnisseite - Konfiguration

5.2 Anlegen von Datenquellen

Nachdem das Template das erste Mal gespeichert wurde, kann im Tab *Datenquellen* die Datengrundlage für das Template konfiguriert werden.

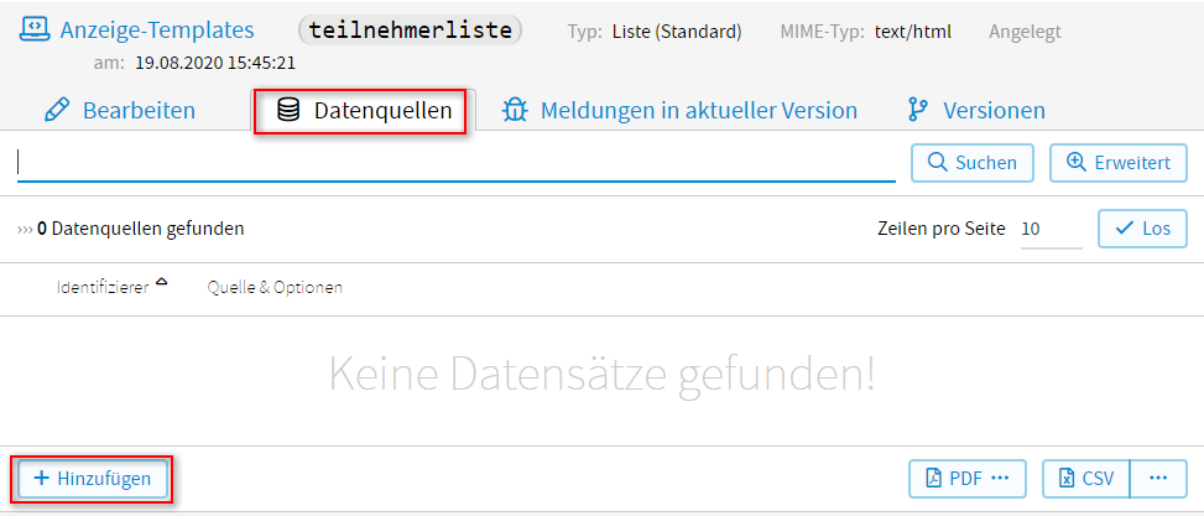

Abb. 7: Datenquelle hinzufügen

Dies ermöglicht es, auf Daten aus mehreren Apps zuzugreifen, die zur Verfügung stehenden Daten im Vorhinein einzuschränken und auf App-Parameter zuzugreifen. Außerdem haben Sie die Möglichkeit bei *Sortierung* Ihre *Daten zu sortieren* und bei *Felder* einzelne *Felder Ihrer App aus- oder abzuwählen*.

Möchten Sie auf Daten mehrerer unterschiedlicher Apps zugreifen, müssen Sie für jede App eine Datenquelle anlegen. Sie können im Anzeige-Template auf die Datenquelle zugreifen mit `datasources.beispiel`. Wobei `beispiel` für den Identifizierer der Datenquelle steht. Über die Attribute der Datenquelle können Sie in der Dokumentation der LivingAPI unter *DataSource* mehr lesen.

Klicken Sie auf *Hinzufügen* um eine Datenquelle anzulegen. Im nun geöffneten Fenster können folgende *Konfigurationen* vorgenommen werden.

Identifizierer

Der eindeutige Name dieser Datenquelle. Der Name darf nur Buchstaben, Ziffern und _ beinhalten. Eine Datenquelle mit dem Namen `beispiel` kann in den Anzeige-Templates mittels des Ausdrucks `datasources.beispiel` angesprochen werden.

App

Hier kann die App ausgewählt werden, deren Datensätze die Datenquelle dem Template zur Verfügung stellen soll. Möchten Sie die Informationen der *System-Apps* zur Verfügung stellen, müssen Sie hier die entsprechende System-App wählen.

Wird eine App ausgewählt, haben Sie die Moglichkeit bei *Kopien einbeziehen* das Häkchen zu setzen. Damit wird nicht nur diese App selbst, sondern auch alle ihre Kopien dem Template zur Verfügung gestellt.

App-Filter

Einen App-Filter können Sie nur dann eingeben, wenn Sie bei *App Alle Apps* gewählt oder bei einer bestimmten App das Häkchen bei *Kopien einbeziehen* gesetzt haben.

Diese Bedingung muss eine App `a` erfüllen, um in die Liste der Apps zu kommen, aus der die Datensätze in die Datenquelle aufgenommen werden. Es stehen die *Variablen* `a` (zu filternde App), `app` (App zu der diese Datenquelle gehört), `user` (eingeloggter Benutzer), `record` (Detail-Datensatz; nur in Detail-Templates) und `params` (Request-Parameter) zur Verfügung.

Mit `a.owner.id == user.id` können Sie sich zum Beispiel nur die Apps anzeigen lassen, bei denen der eingeloggte User der Besitzer ist. Mit `a.identifier == p.str.app` wird die App angezeigt, deren UUID als Parameter in der URL mitgegeben wird.

Anzeige-Templates teilnehmerliste Typ: Liste (Standard) MIME-Typ: text/html

Datenquellen teilnehmer Quelle & Optionen: App **Anmeldung** ; Alle Felder; Alle Datensätze; App-Parameter

Bearbeiten | Sortierung | Untergeordnete Datensätze | Felder

+ Hinzufügen | Kopieren | − Löschen | Hilfe

* Identifizierer: teilnehmer

Der eindeutige Name dieser Datenquelle. Der Name darf nur Buchstaben, Ziffern und "_" beinhalten. Die Datenquelle mit dem Namen beispiel kann in den Anzeige-Templates mittels des Ausdrucks `datasources.beispiel` angesprochen werden.

Quelle

* App: App Anmeldung (Computerkurse)

Wählen Sie hier die App aus deren Datensätze die Datenquelle dem Anzeige-Template zur Verfügung stellen soll. Wird hier „Alle Apps" ausgewählt, so werden die Datensätze aus **allen** Apps zur Verfügung gestellt.

Kopien einbeziehen? ☐

Ist „Kopien einbeziehen?" ausgewählt , so wird nicht nur die App selbst sondern auch alle ihre Kopien dem Template zur Verfügung gestellt.

Felder und Datensätze

* Felder/Datensätze: ○ Keine Daten ○ Felder ◉ Felder und Datensätze

Legt fest ob Informationen zu den Felder bzw. die Datensätze in die Datenquelle aufgenommen werden sollen.

* Felder: ○ Keine Felder ○ Prioritäts-Felder ◉ Alle Felder ○ Alle Felder und Layout-Felder

Legt fest welche Felder in die Datenquelle aufgenommen werden.

Anzahl Datensätze? ☐

Wird „Anzahl Datensätze?" gesetzt, so enthält das App-Attribut `recordcount` die Anzahl der Datensätze (ansonsten ist `recordcount` None).

* Datensätze: ○ Angelegte Datensätze ○ Zugewiesene Datensätze ○ Zugewiesene Datensätze bzw. alle Datensätze für Admins ◉ Alle Datensätze

Legt fest welche Datensätze dem Template in der Liste der Datensätze in die Datenquelle aufgenommen werden (bzw. für das App-Attribut `recordcount` gezählt werden, falls „Anzahl Datensätze?" gesetzt ist).

Datensatz-Filter

Diese Bedingung muß ein Datensatz `r` erfüllen, um in die Liste der Datensätze in der Datenquelle aufgenommen zu werden (bzw. um gezählt zu werden, falls „Anzahl Datensätze?" gesetzt ist).

Sonstige Datensatz-Informationen

Rechte? ☐

Wird „Rechte?" gesetzt, enthalten die `Record`-Objekte unter dem Attribut `permissions` Informationen zu den Zugriffsrechten auf diesem Datensatz (ansonsten ist `permissions` None) (**wird noch nicht unterstützt**).

Anhänge? ☐

Wird „Anhänge?" gesetzt, enthalten die `Record`-Objekte unter dem Attribut `attachments` Informationen zu den Anhängen an diesem Datensatz (ansonsten ist `attachments` None).

Sonstige App-Informationen

Parameter? ☑

Wird „Parameter?" gesetzt, enthält das App-Objekt unter dem Attribut `params` die App-Parameter (ansonsten ist `params` None).

Views? ☐

Wird „Views?" gesetzt, enthält das App-Objekt unter dem Attribut `views` die Views (ansonsten ist `views` None).

* Kategorien: ◉ Keine Kategorien ○ Kategorien-Pfade ○ Kategorien-Bäume ○ Kategorien-Bäume mit Apps

Legt fest welche Informationen zu den App-Kategorien in den App-Objekten zu Verfügung stehen.

Abb. 8: Datenquellen

Ist der Ausdruck nicht vom Typ `BOOL`, wird er mittels der Funktion bool konvertiert. Mehr Informationen zu vSQL-Ausdrücken finden Sie in der Dokumentation zu *vSQL*.

Felder/Datensätze
Hier können Sie festlegen, ob Informationen zu den Feldern (`controls`), oder die Datensätze (`records`) in die Datenquelle aufgenommen werden sollen.

keine Daten
Es werden keine Informationen zu den Feldern bzw. Datensätze in die Datenquelle aufgenommen, sondern nur die App-Information (Name der App, Beschreibung der App, Icon, ...). Dies kann z.B. sinnvoll sein, wenn Sie eine Übersicht über alle Ihre Apps erstellen wollen.

Felder
Hier werden nur Feldinformationen (Name des Feldes, Feldtyp, App zu der das Feld gehört, Info ob Listenfeld, Reihenfolge der Felder, ...) in die Datenquelle aufgenommen. Dies kann z.B. für reine Eingabemasken sinnvoll sein.

Felder und Datensätze
Zusätzlich zu den Feldinformationen werden hier die Datensätze der App mit in die Datenquelle aufgenommen. Diese ist die übliche Auswahl. Damit können Sie dann die Daten Ihrer App im Anzeigetemplate auswerten oder darstellen.

Felder
Erscheint bei der Auswahl von *Felder* oder *Felder und Datensätze*. Legt fest, welche Felder in die Datenquelle aufgenommen werden.

keine Felder
Es werden gar keine Felder aufgenommen.

Prioritätsfelder
Es werden nur die Felder aufgenommen, die in der Datenliste im Datenmanagement ausgewählt sind.

Alle Felder
Es werden alle Felder aufgenommen.

Alle Felder und Layoutfelder
Es werden alle Felder aufgenommen. Zusätzlich enthält `View.layout_controls` alle Layout-Felder, wie die „Formatierten Textfelder“, „Dekobilder“ und den „Absenden“-Button.

(Damit View-Objekte in die Datenquelle aufgenommen werden, muß der Haken bei *Views?* gesetzt sein.)

Anzahl Datensätze
Wird hier das Häkchen gesetzt, wird die Anzahl der Datensätze mit in die Datenquelle aufgenommen (als App-Attribut `recordcount`).

Datensätze
Erscheint bei der Auswahl von *Felder und Datensätze*. Legt fest, welche Datensätze dem Template in der Liste der Datensätze in die Datenquelle aufgenommen werden.

Angelegte Datensätze
Es werden nur die vom Benutzer angelegten Datensätze aufgenommen.

Zugewiesene Datensätze
Es werden nur Datensätze aufgenommen, die dem Benutzer über die Konfiguration *Zugewiesene Daten* zugewiesen wurden. Ohne diese Konfiguration wird diese Option wie *Angelegte Datensätze* behandelt.

Zugewiesene Datensätze bzw. alle Datensätze für Admins
Hat der Benutzer die Rolle Adminstrator oder Datenmanager, so werden alle Datensätze aufgenommen, sonst wird diese Option wie *Zugewiesene Datensätze* behandelt.

Alle Datensätze
Es werden alle Datensätze aufgenommen.

Datensatz-Filter
Diese Bedingung muss ein Datensatz `r` erfüllen, um in die Liste der Datensätze in der Datenquelle aufgenommen zu werden. Es stehen die *Variablen* `r` (zu filternder Datensatz), `app` (App zu der dieses Anzeige-

Template gehört), `user` (eingeloggter Benutzer) und `params` (Request-Parameter) zur Verfügung. Zum Beispiel könnten Sie nur die Daten, die in den letzten 30 Tagen eingegeben wurden anzeigen lassen mit `r.createdat >= now() - days(30)`. Folgenden Filter können Sie eingeben, wenn Sie zum Beispiel nur die Daten aus der App „Vorgänge“ anzeigen lassen wollen, bei denen als Vorgangstyp Rechnung in der Auswahl gewählt wurde `r.v_vorgangstyp == "rechnung"`. Wobei `vorgangstyp` für den identifier des Feldes Vorgangstyp steht.

Ist der Ausdruck nicht vom Typ `BOOL`, wird er mittels der Funktion bool konvertiert. Mehr Informationen zu vSQL-Ausdrücken finden Sie in der Dokumentation zu *vSQL*.

Rechte?
Wird hier das Häkchen gesetzt, wird das Attribut `permissions` gefüllt mit allen der App zugeordneten Benutzern und deren Berechtigungen. (Wird noch nicht unterstützt.)

Anhänge?
Wird hier das Häkchen gesetzt, werden auch die Anhänge (`attachments`) der jeweiligen Datensätze mit in die Datenquelle aufgenommen. Bei dem Datensatz der bei einem Detail-Template angezeigt wird, stehen die Anhänge immer zu Verfügung, unabhängig davon, ob das Häkchen bei *Anhänge?* gesetzt ist oder nicht.

Parameter?
Wird hier das Häkchen gesetzt, dann stehen dem Template die unter *Konfiguration → Erweitert → Parameter* erstellten *App-Parameter* (`params`) zur Verfügung.

Views?
Wird hier das Häkchen gesetzt, werden alle Formularvarianten (Formularansichten — `views`) dem Template zur Verfügung gestellt.

Kategorien
Legt fest, welche Informationen zu den *App-Kategorien* in den App-Objekten zur Verfügung stehen. Zur Demonstration der folgenden Möglichkeiten wird das Beispiel der unter *Konfiguration → Erweitert → Account → App-Kategorien angelegten Kategorien* verwendet.

Keine Kategorien
Es steht keine Information zur Verfügung, d.h. das Attribut `categories` der App-Objekte ist immer `None`.

Kategorien-Pfade
Hier ist das App-Attribut `categories` ein Dictionary mit den Kategorien, zu denen diese App gehört. Die Schlüssel sind der jeweilige Identifizierer der Kategorie und der Wert sind Kategorie-Objekte. Bei diesen verweist das `parent`-Attribut jeweils auf die übergeordnete Kategorie. D.h. über die den Apps zugeordneten Kategorien können über das parent-Attribut die Pfade zu diesen Kategorien im Kategorien-Baum rekonstruiert werden, aber nicht die Bäume selbst. Zum Beispiel *Familie → Privat*, oder *Verein → Privat*.

Kategorien-Bäume
Diese Option funktioniert ähnlich wie *Kategorien-Pfade*, jedoch enthält das Kategorien-Attribut `children` zusätzlich noch die dieser Kategorie untergeordneten Kategorien als Dictionary. (Bei *Kategorien-Pfade* ist `children` immer `None`). Zum Beispiel die übergeordnete Kategorie *Privat* mit den Unterkategorien *Familie* und *Verein*.

Kategorien-Bäume mit Apps
Diese Option funktioniert ähnlich wie *Kategorien-Bäume*, jedoch enthält das Kategorien-Attribut `apps` zusätzlich noch die dieser Kategorie zugeordneten Apps als Dictionary (bei *Kategorien-Pfade* und *Kategorien-Bäume* ist `apps` immer `None`). Zum Beispiel die übergeordnete Kategorie *Privat* mit der Unterkategorie *Familie* und deren Apps *Haushaltsbuch* und *Urlaubsplanung*, sowie der Unterkategorie *Verein* und deren Apps *Mitglieder* und *Spenden*.

Nach dem Speichern der Datenquelle erscheinen neben *Bearbeiten* die Tabs *Sortierung*, *Untergeordnete Datensätze* und *Felder*.

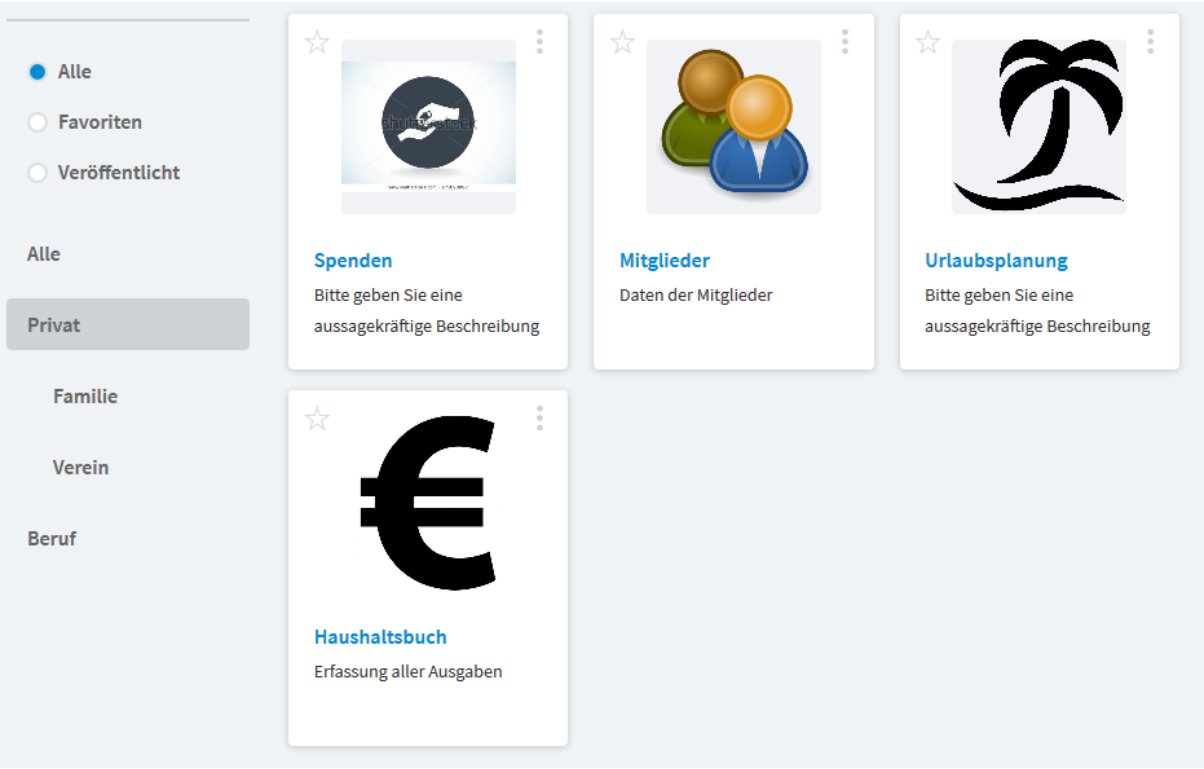

Abb. 9: Beispiel Kategorien

5.2.1 Sortierung

Bei *Sortierung* kann nach einem bestimmten Ausdruck *sortiert*, die Sortier-Richtung und das Handling von Null-Werten eingestellt werden. Wird hier keine Sortierung festgelegt, so wird die Sortierung, wie sie unter *Datenmanagement* → *Sortierung* für das Datenmanagement konfiguriert wurde, verwendet.

Klicken Sie auf den Tab *Sortierung* und dann auf *Hinzufügen*, um eine *Sortierung anzulegen*.

Im nun geöffneten Fenster können folgende Konfigurationen vorgenommen werden.

Reihenfolge
: Bei mehrerern Sortierungen werden die Datensätze in dieser Reihenfolge sortiert.

Ausdruck
: Nach dem Wert dieses Ausdrucks werden die Datensätze sortiert. Für die Formulierung wird *vSQL* verwendet. Es stehen die *Variablen* `r` (zu sortierender Datensatz), `app` (App zu der das zugehörige Anzeige-Template gehört) `record` (Detail-Datensatz; nur in Detail-Templates), `user` (eingeloggter Benutzer) und `params` (Request-Parameter) zur Verfügung.

Sortier-Richtung

Aufsteigend
: Aufsteigende Sortierung (A-Z)

Absteigend
: Absteigende Sortierung (Z-A)

Nullwerte

Zuerst
: Nullwerte werden zuerst angezeigt.

Zuletzt
: Nullwerte werden zuletzt angezeigt.

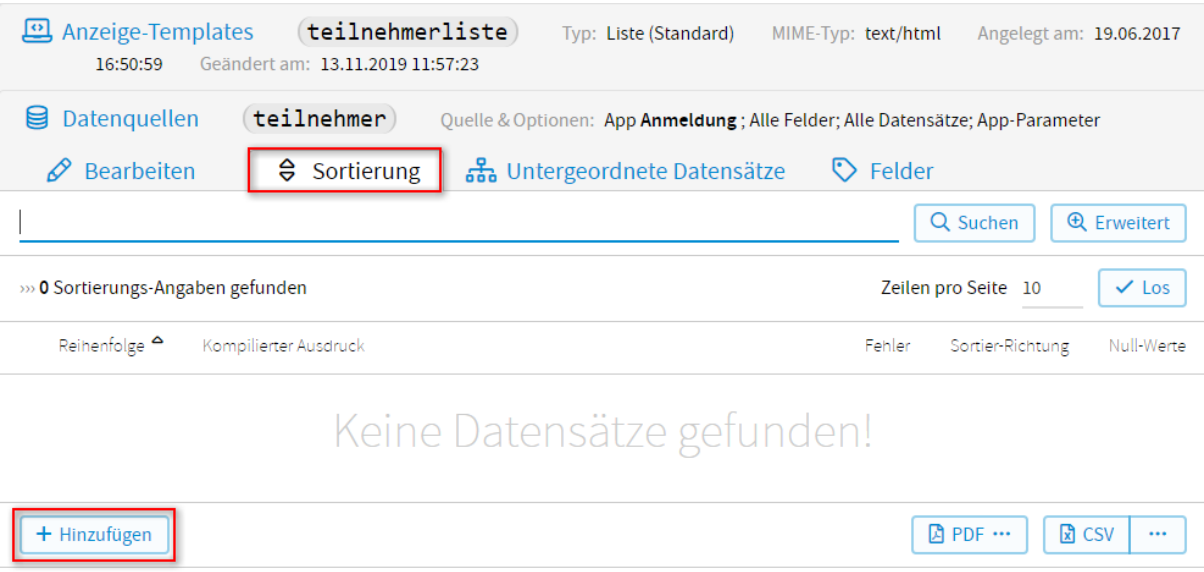

Abb. 10: Datenquellen - Sortierung hinzufügen

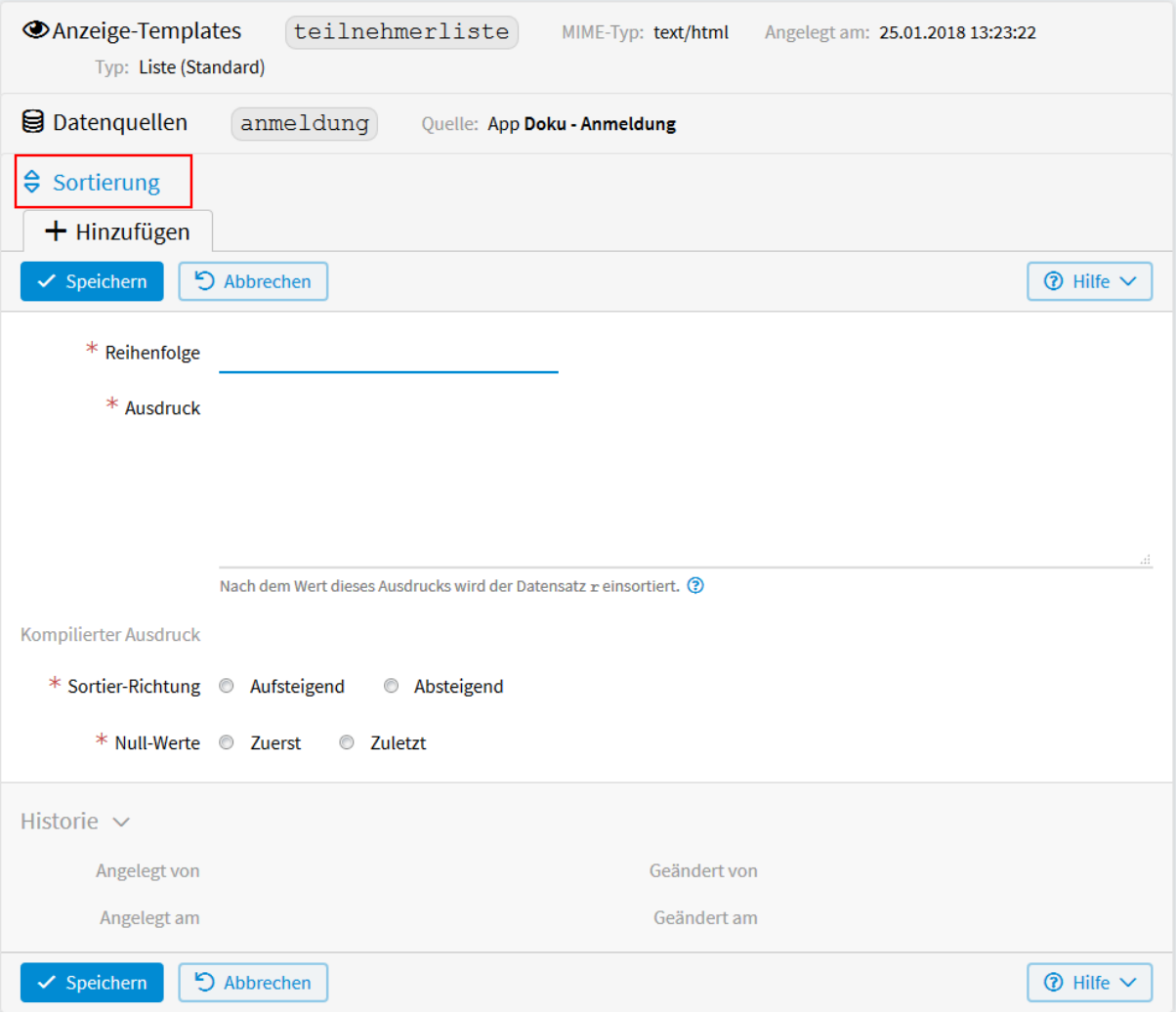

Abb. 11: Datenquellen - Sortierung

5.2.2 Untergeordnete Datensätze

Wenn Sie in einer App eine Auswahl auf eine andere App definieren, dann erzeugen Sie damit eine Eltern-Kind-Beziehung. In unserem Anwendungsfall z. B. greift das Feld *Veranstaltung* in der App *Anmeldung* auf die App *Veranstaltungen* zu. Das heißt die App *Veranstaltungen* hat eine Eltern-Kind-Beziehung zu der App *Anmeldung* (eine Veranstaltung kann mehrere Anmeldungen haben). Diese Beziehung können Sie für untergeordnete Datensätze nutzen. So könnten Sie sich z. B. die Teilnehmer zu Ihren Veranstaltungen in einem Template anzeigen lassen. Dafür müssen Sie in der App *Veranstaltungen* ein *Anzeigetemplate* und dazu eine *Datenquelle* anlegen. Für dieses Beispiel wurde folgende *Datenquelle* konfiguriert.

Nach dem Speichern der Datenquelle klicken Sie auf den Tab *Untergeordnete Datensätze* und dann auf *Hinzufügen*.

Im nun geöffneten Fenster werden folgende Konfigurationen vorgenommen.

Identifizierer
: Der eindeutige Identifizierer für diese Konfiguration. Der Identifizierer darf nur Buchstaben, Ziffern und _ beinhalten. Die Konfiguration mit dem Namen `beispiel` kann in den Anzeige-Templates für einen Datensatz `record` mittels des Ausdrucks `record.children.beispiel` angesprochen werden. Zu einem Veranstaltungsdatensatz wäre z. B. bei o.g. Beispiel ein Zugriff auf die zugehörigen Anmeldungen möglich über `record.children.anmeldungen`.

Ziel
: Hier können Sie die App auswählen, deren Datensätze den Datensätzen dieser Datenquelle zugeordnet sind. Angezeigt werden die Apps, die ausgewählt werden können, sowie die Felder worüber die beiden Apps miteinander verknüpft sind. Ist eine App mehrfach mit einer anderen App verbunden, dann können sie über das Feld, das Ihnen bei *Ziel* mitangezeigt wird, diese beiden Verknüpfungen unterscheiden.

Filterbedingung
: Diese Bedingung muß ein Datensatz `r` erfüllen, um in die Liste der untergeordneten Datensätze aufgenommen zu werden (zusätzlich zu der Bedingung, dass er dem Hauptdatensatz zugeordnet ist). Es stehen die *Variablen* `r` (untergeordneter zu filternder Datensatz), `app` (App zu der das zugehörige Anzeige-Template gehört), `record` (Detail-Datensatz; nur in Detail-Templates), `user` (eingeloggter Benutzer) und `params` (Request-Parameter) zur Verfügung. Ist der Ausdruck nicht vom Typ `BOOL`, wird er mittels der Funktion `bool` konvertiert. Für das *Formulieren der Bedingung* wird *vSQL* verwendet.

5.2.3 Felder

Hier haben Sie die Möglichkeit, einzelne Felder Ihrer App aus- oder abzuwählen, welche in die Datenquelle aufgenommen werden sollen. Das ist z. B. sinnvoll, wenn Sie nur wenige von vielen Feldern anzeigen lassen möchten oder wenn Sie entgegen der Angabe in der Datenquelle einzelne Felder mitaufnehmen oder ausschließen möchten.

Durch Klicken auf den Tab *Felder* gelangen Sie zu folgender Seite.

Nichts ausgewählt
: Das Feld wird entsprechend der Angabe in der Datenquelle behandelt.

Nein
: Das Feld wird nicht mit in die Datenquelle aufgenommen.

Ja
: Das Feld wird mit in die Datenquelle aufgenommen.

In der Übersicht Ihrer Anzeige-Templates können Sie durch Klicken auf den Pfeil neben dem entsprechenden Template einsehen, was in der *Datenquelle konfiguriert* wurde.

5.3 Zugriff auf System-Apps

Über System-Apps können Sie auf Informationen aus dem Living-Apps-System zugreifen, als wären diese Informationen Datensätze einer App (eben einer System-App). Überall wo Sie eine normale App als Datenquelle wählen können ist es auch möglich, statt dessen eine System-App zu wählen. Um auf eine System-App zugreifen zu können, müssen Sie dafür eine Datenquelle anlegen und dort als Quelle bei *Apps* die entsprechende System-App auswählen. Darüber können Sie in der Dokumentation der *Datenquelle* mehr lesen.

Anzeige-Templates beispiel Typ: Liste (Standard) MIME-Typ: text/html Angelegt am: 26.09.2019 11:33:19 Geändert am: 20.08.2020 10:37:02

Datenquellen veranstaltungen Quelle & Optionen: App **Veranstaltungen (Doku)**; Alle Felder; Alle Datensätze

Bearbeiten | Sortierung | Untergeordnete Datensätze | Felder

Speichern | Wiederherstellen | Hilfe

* Identifizierer: veranstaltungen

Der eindeutige Name dieser Datenquelle. Der Name darf nur Buchstaben, Ziffern und "_" beinhalten. Die Datenquelle mit dem Namen `beispiel` kann in den Anzeige-Templates mittels des Ausdrucks **`datasources.beispiel`** angesprochen werden.

Quelle

* App: App Veranstaltungen (Doku) (Beispiel-App für Doku)

Wählen Sie hier die App aus deren Datensätze die Datenquelle dem Anzeige-Template zur Verfügung stellen soll. Wird hier „Alle Apps" ausgewählt, so werden die Datensätze aus **allen** Apps zur Verfügung gestellt.

Kopien einbeziehen? ☐

Ist „Kopien einbeziehen?" ausgewählt , so wird nicht nur die App selbst sondern auch alle ihre Kopien dem Template zur Verfügung gestellt.

Felder und Datensätze

* Felder/Datensätze: ○ Keine Daten ○ Felder ◉ Felder und Datensätze

Legt fest ob Informationen zu den Felder bzw. die Datensätze in die Datenquelle aufgenommen werden sollen.

* Felder: ○ Keine Felder ○ Prioritäts-Felder ◉ Alle Felder

Legt fest welche Felder in die Datenquelle aufgenommen werden.

Anzahl Datensätze? ☐

Wird „Anzahl Datensätze?" gesetzt, so enthält das App-Attribut `recordcount` die Anzahl der Datensätze (ansonsten ist `recordcount` None).

* Datensätze: ○ Nur vom Benutzer angelegte Datensätze ○ Datensätze an denen der Benutzer Besitz-Rechte hat ◉ Alle Datensätze

Legt fest welche Datensätze dem Template in der Liste der Datensätze in die Datenquelle aufgenommen werden (bzw. für das App-Attribut `recordcount` gezählt werden, falls „Anzahl Datensätze?" gesetzt ist).

Datensatz-Filter

Diese Bedingung muß ein Datensatz `r` erfüllen, um in die Liste der Datensätze in der Datenquelle aufgenommen zu werden (bzw. um gezählt zu werden, falls „Anzahl Datensätze?" gesetzt ist).

Sonstige Datensatz-Informationen

Rechte? ☐

Wird „Rechte?" gesetzt, enthalten die `Record`-Objekte unter dem Attribut `permissions` Informationen zu den Zugriffsrechten auf diesem Datensatz (ansonsten ist `permissions` None) (**wird noch nicht unterstützt**).

Anhänge? ☐

Wird „Anhänge?" gesetzt, enthalten die `Record`-Objekte unter dem Attribut `attachments` Informationen zu den Anhängen an diesem Datensatz (ansonsten ist `attachments` None).

Sonstige App-Informationen

Parameter? ☐

Wird „Parameter?" gesetzt, enthält das App-Objekt unter dem Attribut `params` die App-Parameter (ansonsten ist `params` None).

Views? ☐

Wird „Views?" gesetzt, enthält das App-Objekt unter dem Attribut `views` die Views (ansonsten ist `views` None).

* Kategorien: ◉ Keine Kategorien ○ Kategorien-Pfade ○ Kategorien-Bäume ○ Kategorien-Bäume mit Apps

Legt fest welche Informationen zu den App-Kategorien in den App-Objekten zu Verfügung stehen.

Abb. 12: Datenquelle - untergeordnete Datensätze

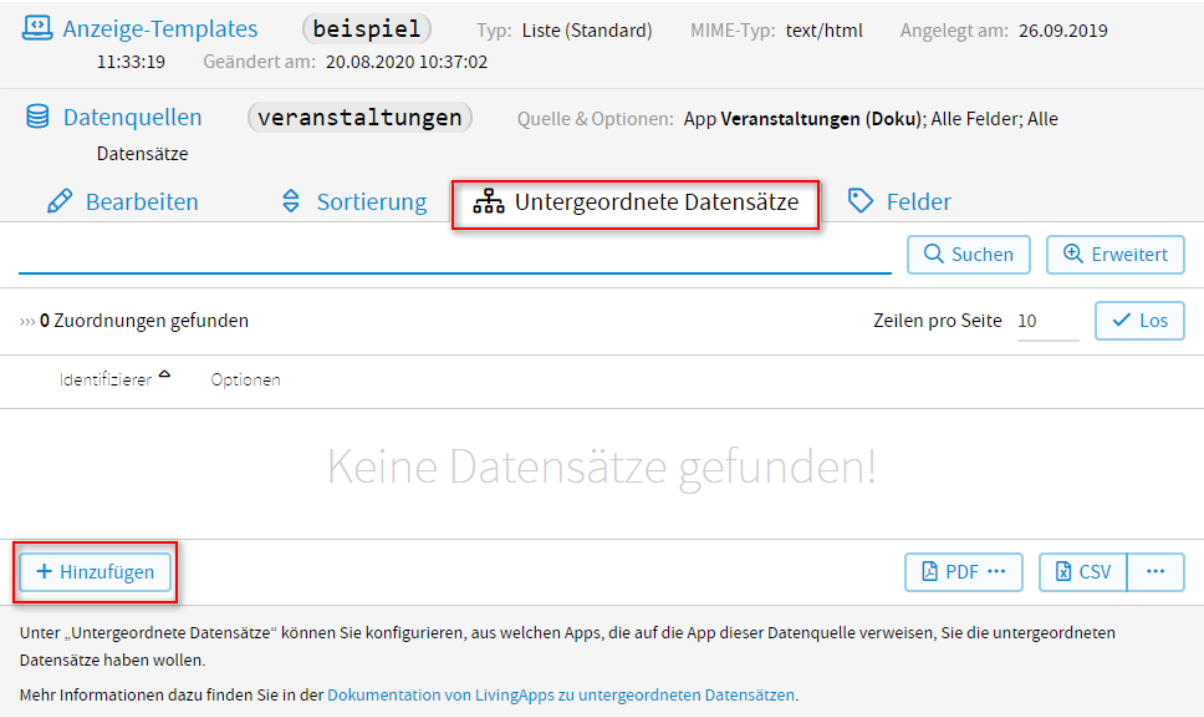

Abb. 13: Datenquelle - untergeordnete Datensätze hinzufügen

Folgende System-Apps stehen zur Verfügung.

5.3.1 System-App „Aktivitäten"

Die System-App „Aktivitäten" ermöglicht den Zugang zu dem im System protokollierten Aktivitäten (siehe *Aktivitäten anlegen?* zur Erklärung von Aktivitäten).

Die Sytem-App „Aktivitäten" hat folgende Felder:

Name	Identifizierer	Typ
Übergeordnete Aktivität	`parent`	`applookup/select`
Typ	`type`	`string/text`
App	`app`	`applookup/select`
Datensatz	`record`	`applookup/select`
Inhalt	`content`	`string/textarea`
Quittiert am	`seen`	`date/datetimeminute`

Das Feld `type` (d.h. der Aktivitäts-Typ) kann folgende Werte haben:

Anzeige-Templates beispiel Typ: Liste (Standard) MIME-Typ: text/html Angelegt am: 26.09.2019 11:33:19 Geändert am: 20.08.2020 10:37:02

Datenquellen veranstaltungen Quelle & Optionen: App **Veranstaltungen (Doku)**; Alle Felder; Alle Datensätze

Untergeordnete Datensätze

+ Hinzufügen

Speichern Abbrechen Hilfe

* Identifizierer anmeldungen

Der eindeutige Name für diese Konfiguration. Der Name darf nur Buchstaben, Ziffern und "_" beinhalten. Die Konfiguration mit dem Namen `beispiel` kann in den Anzeige-Templates für einen Datensatz **record** mittels des Ausdrucks `record.children.beispiel` angesprochen werden.

Ziel
- (Nichts ausgewählt)
- (●) App **Anmeldung** /Feld **Veranstaltung**
- System-App **Aktivitäten**/Feld **Datensatz**

Hier können Sie die App auswählen, deren Datensätze den Datensätzen dieser Datenquelle zugeordnet sind.

Filter

Filter-Bedingung

Diese Bedingung muß ein Datensatz **r** erfüllen, um in die Liste der untergeordneten Datensätze aufgenommen zu werden (zusätzlich zu der Bedingung, daß er dem Hauptdatensatz zugeordnet ist).

Abb. 14: Datenquellen - untergeordnete Datensätze - Konfiguration

Anzeige-Templates teilnehmerliste Typ: Liste (Standard) MIME-Typ: text/html Angelegt am: 19.06.2017 16:50:59 Geändert am: 13.11.2019 11:57:23

Datenquellen teilnehmer Quelle & Optionen: App **Anmeldung** ; Alle Felder; Alle Datensätze; App-Parameter

Bearbeiten | Sortierung | Untergeordnete Datensätze | Felder

Bezeichnung | Identifizierer | Typ | Suchen | Erweitert

›› 20 Felder gefunden « ‹ 1 2 › » Zeilen pro Seite 10 Los

	Bezeichnung	Identifizierer	Typ	Priorität?	In Datenquelle?		
1	Anrede	anrede	lookup/radio		(Nichts ausgewählt)	Nein	Ja
2	Zuständiger Mitarbeiter	zustaendiger_mitarbeiter	string/email		(Nichts ausgewählt)	Nein	Ja
3	abgemeldet	abgemeldet	bool		(Nichts ausgewählt)	Nein	Ja
4	Zimmer sind gebucht	zimmer_sind_gebucht	bool		(Nichts ausgewählt)	Nein	Ja
5	Teilnahmegebühr	teilnahmegebuehr	number		(Nichts ausgewählt)	Nein	Ja
6	bezahlt	bezahlt	bool		(Nichts ausgewählt)	Nein	Ja
7	bezahlt am	bezahlt_am	date/date		(Nichts ausgewählt)	Nein	Ja
8	Veranstaltung	veranstaltung2	applookup/select	✓	(Nichts ausgewählt)	Nein	Ja
9	Vorname	vorname	string/text	✓	(Nichts ausgewählt)	Nein	Ja
10	Nachname	nachname	string/text	✓	(Nichts ausgewählt)	Nein	Ja

Alle speichern | Wiederherstellen

Abb. 15: Datenquellen - Felderwahl

Anzeige-Templates

Bitte Suchbegriff eingeben | Suchen | Erweitert

›› 3 Anzeige-Templates gefunden Zeilen pro Seite 10 Los

	Identifizierer	Beschreibung	Typ	MIME-Typ			
1	Teilnehmer		Liste	text/html	0	1	25
2	css		Support	text/css	0	0	5
3	teilnehmerliste	Teilnehmerliste	Liste (Standard)	text/html	0	1	66

Datenquellen

	Identifizierer	Quelle	Optionen
1	anmeldung	App **Anmeldung**	Alle Felder; Alle Datensätze; App-Parameter

+ Hinzufügen | CSV | PDF

In Anzeige-Templates können Sie sogenannte "UL4-Templates" anlegen, die es Ihnen ermöglichen, die Daten Ihrer LivingApp in jeder denkbaren Form darzustellen und aufzubereiten.

Mehr Informationen dazu finden Sie in der Dokumentation zu Anzeige-Templates.

Abb. 16: Datenquellen - Ansicht in der Template-Übersicht

Wert	Beschreibung
attachment.insert	Anhang erzeugt
attachment.update	Anhang geändert
attachment.delete	Anhang gelöscht
massdata.insert	Mehrere Datensätze erzeugt
massdata.update	Mehrere Datensätze geändert
massdata.delete	Mehrere Datensätze gelöscht
data.insert	Datensatz erzeugt
data.update	Datensatz geändert
data.delete	Datensatz gelöscht
linkaction.execute	Link-Aktion ausgeführt
email.send	Email versendet
email.fail	Emailversendung fehlgeschlagen
task.insert	Aufgabe erzeugt
task.done	Aufgabe erledigt
task.delete	Aufgabe gelöscht
chat	Chat-Nachricht
app.chat	Chat-Nachricht zur App
data.chat	Chat-Nachricht zum Datensatz

So können Sie z. B. in der Datenquelle bei *Datensatz-Filter* folgende Bedingung angeben:

```
r.v_type != "data.insert"
```

Wenn ein Datensatz erzeugt wird, soll keine Aktivität angezeigt werden, ansonsten schon.

Folgende Bedingung zeigt nur die Aktivitäten zu den Chats an und keine anderen:

```
"chat" in r.v_type
```

5.3.2 System-App „Apps“

Bei dieser System-App werden die Apps selbst als Datensätze interpretiert. Diese Datensätze haben folgende Felder:

Name	Identifizierer	Typ
Name	`name`	`string/text`
Beschreibung	`description`	`string/text`
Kleines Icon	`icon_small`	*File*
Großes Icon	`icon_large`	*File*
Anzahl unquittierter Aktivitäten	`activity_count`	`int`

Dies ermöglicht es Ihnen z.B. ein Template zu erstellen, das alle ihre Apps anzeigt.

5.3.3 System-App „Controls“

5.3.4 System-App „Emailtemplates“

5.3.5 System-App „Versendungen“

5.4 Anwendungsbeispiel

Um die Darstellung eines Anzeige-Templates zu veranschaulichen, wird im Folgenden auf das Anwendungsbeispiel aus Kapitel 1 zurückgegriffen. Es soll das Anzeige-Template *Teilnehmerliste* vom Typ *Liste* erstellt werden, welches als Übersichtsseite über sämtliche Anmeldungen dient und verschiedene Links und Parameter einbindet. Die

Übersicht soll eine Tabelle anzeigen, mit den Spalten `Kurs`, `Teilnehmer`, `Anzahl Personen`, `E-Mail-Adresse` und `Bearbeiten`. Außerdem soll die Summe der angemeldeten Personen berechnet und ausgegeben werden.

Zusätzlich wird das Anzeige-Template *Ergebnis* vom Typ *Detail* (Ergebnisseite) erstellt, welches die Teilnehmer über eine erfolgreiche Anmeldung informieren soll.

Das zugehörige *CSS-Template* wird in einem separaten *Support-Template* ausgelagert und dann eingebunden.

Teilnehmerliste

Anmeldung (Computerkurse)

Kurs	Teilnehmer	Anzahl Personen	E-Mail-Adresse	Bearbeiten
Grundlagen Computerwissen (Aufbaukurs)	Beispiel Karl	2	karl.beispiel@example.org	
Grundlagen Computerwissen (Grundkurs)	Beispiel Hans	4	hans.beispiel@example.org	
Power Point Grundkurs	Muster Klara	4	klara.muster@example.org	
Power Point Grundkurs	Mustermann Max	1	max.mustermann@example.org	
Power Point Grundkurs	Sorglos Sabine	1	sabine.sorglos@example.org	
Power Point Grundkurs	Test Peter	2	peter.test@example.org	
Word für Einsteiger	Hinz Anne	1	anne.hinz@example.org	
Word für Fortgeschrittene	Mustermann Max	2	max.mustermann@example.org	

Insgesamt angemeldete Personen: 17

Anmeldung hinzufügen

Abb. 17: Anzeige-Template Teilnehmerliste

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Anzeige-Templates*. Klicken Sie auf *Hinzufügen* um ein neues Anzeige-Template anzulegen.

Im nun geöffneten Fenster werden folgende *Konfigurationen* vorgenommen.

Bei *Quelltext* wird folgender Code eingegeben.

```
<?doc?>
Teilnehmerliste
<?end doc?>
<?code teilnehmerapp = datasources.teilnehmer.app?>
<html>
   <head>
      <title>Teilnehmerliste</title>
      <?render globals.t_la_static_font_awesome()?>
      <link rel="stylesheet" href="<?printx teilnehmerapp.template_url('css')?>" />
   </head>
   <body>
      <h1><?printx app.pv_ueberschrift?></h1>
      <h2>
         <?if teilnehmerapp.image is not None?>
            <img src="<?printx teilnehmerapp.image.url?>" width="32"/>
         <?end if?>
         <?printx teilnehmerapp.name?> (<?printx teilnehmerapp.description?>)
      </h2>
      <table align="center" border="0" cellpadding="0" cellspacing="0">
         <thead>
            <tr>
               <th>Kurs</th>
               <th>Teilnehmer</th>
               <th>Anzahl Personen</th>
               <th>E-Mail-Adresse</th>
               <th>Bearbeiten</th>
            </tr>
         </thead>
```

(Fortsetzung auf der nächsten Seite)

Anzeige-Templates teilnehmerliste Typ: Liste (Standard) MIME-Typ: text/html

4/4

Bearbeiten | Datenquellen | Meldungen in aktueller Version | Versionen

+ Hinzufügen | Kopieren | − Löschen | Meldungen ··· | Hilfe

* Identifizierer teilnehmerliste

Eine eindeutige Kennung für das Template

Quelltext

```
1  <?doc Teilnehmerliste?>
2  <?code teilnehmerapp = datasources.teilnehmer.app?>
3  <html>
4  ----<head>
5  --------<title>Teilnehmerliste</title>
6  --------<?render globals.t_static_font_awesome()?>
7  --------<link rel="stylesheet" href="?template=css"/>
8  ----</head>
9  ----<body>
10 --------<h1><?print app.params.ueberschrift.value?></h1>
11 --------<h2>
12 ------------<?if teilnehmerapp.iconsmall is not None?>
13 ----------------<img src="<?printx teilnehmerapp.iconsmall.url?>" width="32"/>
14 ------------<?end if?>
15 ------------<?printx teilnehmerapp.name?> (<?printx teilnehmerapp.description?>)
16 --------</h2>
17 --------<table align="center" border="0" cellpadding="0" cellspacing="0">
18 ------------<thead>
19 ----------------<tr>
20 --------------------<th>Kurs</th>
21 --------------------<th>Teilnehmer</th>
22 --------------------<th>Anzahl Personen</th>
23 --------------------<th>E-Mail-Adresse</th>
24 --------------------<th>Bearbeiten</th>
25 ----------------</tr>
26 ------------</thead>
27 ------------<tbody>
28 ----------------<?code summe_personen = 0?>
29 ----------------<?for record in teilnehmerapp.records.values()?>
30 --------------------<?if record.v_abgemeldet is not True?>
31 ------------------------<tr>
32 ----------------------------<td>
33 --------------------------------<?if record.v_veranstaltung2?>
34 ----------------------------------------<?printx record.v_veranstaltung2.v_veranstaltung?>
35 --------------------------------<?end if?>
36 ----------------------------</td>
37 ----------------------------<td>
38 --------------------------------<?printx record.v_nachname?> <?printx record.v_vorname?>
39 ----------------------------</td>
40 ----------------------------<td>
41 --------------------------------<?if record.v_anzahl_personen2?>
42 ----------------------------------------<?printx record.v_anzahl_personen2?>
43 ----------------------------------------<?code summe_personen += record.v_anzahl_personen2?>
44 --------------------------------<?end if?>
```

Hilfe — Vollbild

Verfügbare Variablen: app, record, datasources. Verfügbare globale Variablen: globals, la

Template-Eigenschaften > Beschreibung

Anzeige-Eigenschaften

* Typ ○ Liste ◉ Liste (Standard) ○ Liste (inkl. Datenmanagement) ○ Detail ○ Detail (Ergebnisseite) ○ Detail (inkl. Datenmanagement) ○ Support

Ein „Listen-Template“ zeigt eine Liste von Datensätzen an, ein „Detail-Template“ einen einzelnen. Ein „Support-Template“ kann für andere Anzeigen verwendet werden, die keine Datensätze benötigen.

* Berechtigung für alle Benutzer

Abb. 18: Anzeige-Template Konfiguration

(Fortsetzung der vorherigen Seite)

```
29          <tbody>
30             <?code summe_personen = 0?>
31             <?for record in teilnehmerapp.records.values()?>
32                <?if not record.v_abgemeldet?>
33                   <tr>
34                      <td>
35                         <?if record.v_veranstaltung2?>
36                            <?printx record.v_veranstaltung2.v_veranstaltung?>
37                         <?end if?>
38                      </td>
39                      <td>
40                         <?printx record.v_nachname?> <?printx record.v_vorname?>
41                      </td>
42                      <td>
43                         <?if record.v_anzahl_personen2?>
44                            <?printx record.v_anzahl_personen2?>
45                            <?code summe_personen += record.v_anzahl_personen2?>
46                         <?end if?>
47                      </td>
48                      <td>
49                         <?printx record.v_e_mail_adresse?>
50                      </td>
51                      <td class="zentriert">
52                         <a target="_top" href="<?printx record.edit_embedded_url()?>">
↪<?render globals.t_la_icon("pencil", fw=True)?></a>
53                      </td>
54                   </tr>
55                <?end if?>
56             <?end for?>
57          </tbody>
58       </table>
59       <p>Insgesamt angemeldete Personen: <?printx summe_personen?></p>
60       <p><a target="_top" href="<?printx teilnehmerapp.new_embedded_url()?>">
↪Anmeldung hinzufügen</a></p>
61    </body>
62 </html>
```

Zeile 1
Die Beschreibung des Templates mit dem `<?doc?>`-Tag.

Zeile 4
Belegung der Variablen `teilnehmerapp` mit den Informationen aus der App der Datenquelle.

Zeilen 6 - 10
Der *title* der Seite wird festgelegt. Das interne Template `la_static_font_awesome` (Zeile 8) und das ausgelagerte CSS-Template (Zeile 9) werden eingebunden.

Zeile 12
Die Überschrift des Templates wird über den App-Parameter `ueberschrift` eingebunden und ausgegeben. Über App-Parameter können Sie in der Dokumentation der *Parameter* mehr lesen.

Zeilen 14 - 16
Wenn ein App-Icon hochgeladen wurde, wird dieses hier ausgegeben.

Zeile 17
Der Name und die Beschreibung der App wird ausgegeben.

Zeile 18 - 26
Der Kopf der Tabelle wird festgelegt, mit den Spalten-Überschriften `Kurs`, `Teilnehmer`, `Anzahl Personen`, `E-Mail-Adresse` und `Bearbeiten`.

Zeile 30
Die Variable `summe_personen` wird initialisiert, zu der in jedem Datensatz die Anzahl der angemeldeten Personen aufaddiert wird.

Zeile 31
Die `for`-Schleife durchläuft die Werte der Datensätze in der Datenquelle.

Zeile 32
Der Code im `if`-Block von Zeile 32 - 55 wird nur ausgeführt für Teilnehmer, die sich nicht wieder abgemeldet haben.

Zeilen 35 - 37
Wenn eine Veranstaltung ausgewählt wurde, wird der Wert des Feldes „Veranstaltung“ aus der verknüpften App „Veranstaltungen“ ausgegeben.

Zeile 40
Der Nachname und der Vorname des Teilnehmers wird ausgegeben.

Zeile 43 - 46
Wenn das Feld `Anzahl Personen` gefüllt ist, dann wird dieser Wert hier (Zeile 44) ausgegeben und anschließend auf die in Zeile 30 definierte Variable aufsummiert (Zeile 45).

Zeile 49
Die E-Mail-Adresse des Teilnehmers wird ausgegeben.

Zeile 52
Link zum Bearbeiten des jeweiligen Datensatzes. Mit `<?render globals.t_la_icon("pencil", fw=True)?>` wählen Sie das Icon zum Editieren aus Font-Awesome aus.

Zeile 59
Ausgabe der Variablen `summe_personen`.

Zeile 60
Link zur Dateneingabe, in dem ein neuer Datensatz angelegt werden kann.

Nachdem sie das Template gespeichert haben, können Sie Datenquellen dazu anlegen. Die Datenquelle, die für dieses Anwendungsbeispiel angelegt wurde, finden Sie in der Dokumentation der *Datenquelle*.

Wie ein Template aufgerufen wird, können Sie unter *Aufruf eines Anzeige-Templates* nachlesen.

5.5 Verwendung von Template-Vorlagen

Sie haben außerdem die Möglichkeit, *Template-Vorlagen* in HTML-Format für Tabellen-/Detail-Ansicht und als Übersichtsseite für Ihre LivingApp zu verwenden.

5.5.1 Einzelne Template-Vorlagen übernehmen

Um einzelne Template-Vorlagen für Ihre LivingApp zu übernehmen, wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Anzeige-Templates* und dann auf *Aus Vorlage hinzufügen* . . . .

Es öffnet sich folgendes Fenster:

Übersicht-Templates

Übersichts-Seite - `overview_page`
Eine *Übersichts-Seite* bietet eine Navigation durch alle Panels an, die Sie unter *App-Panels* und *Benutzer-Panels* angelegt haben und dort *Auf der eigenen Übersichtsseite?* ausgewählt wurde.

Listen-Templates

Listen-Seite - `list_page`
Dieses Template bietet eine einfache *Listen-Ansicht* der Daten in dieser App.

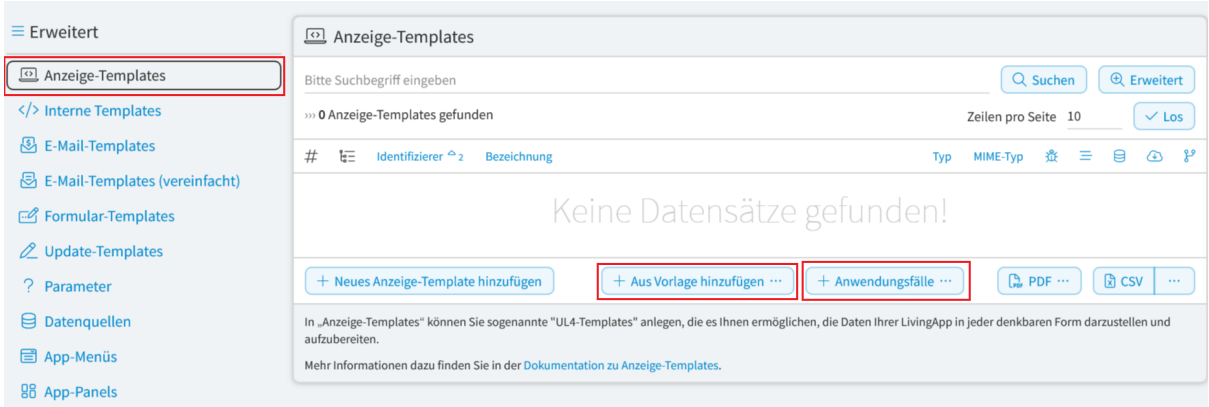

Abb. 19: Aus Vorlage hinzufügen

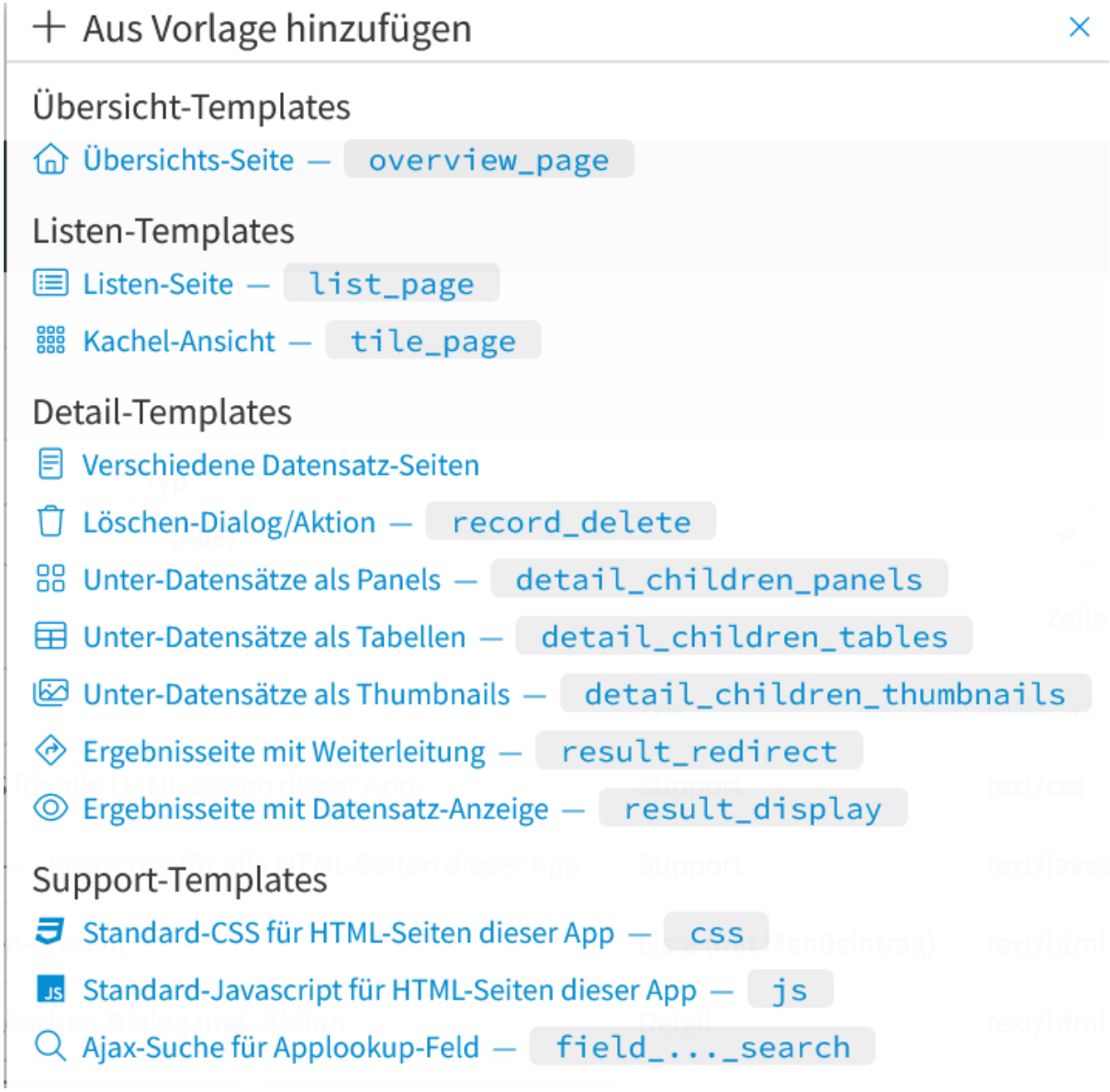

Abb. 20: Vorlage auswählen

Kachel-Ansicht - `title_page`
Dieses Template bietet eine einfache *Kachel-Ansicht* der Daten in dieser App.

Detail-Templates

Verschiedene Datensatz-Seiten
Dieses Anzeige-Template zeigt einen Datensatz in der einen oder anderen Form an.

Welche Form das ist, wird durch *Kontext* und *Modus* festgelegt.

Je nach *Kombination der Auswahl* bei *Kontext* und *Modus* können folgende Anzeige-Templates angelegt werden:

***Datensatz-Anzeige-Seite* - `detail_page`**
Dieses Template bietet eine einfache *Anzeige-Seite* für einen Datensatz in dieser LivingApp und etwaige per „Untergeordnete Datensätze“ zugordnete Details.

Der einzelne Datensatz wird als vollständige HTML-Seite angezeigt.

***Datensatz-Bearbeiten-Seite* - `detail_edit_page`**
Diese Template bietet eine einfache *Bearbeiten-Seite* für einen Datensatz in dieser LivingApp.

Der Datensatz wird als vollständige HTML-Seite bearbeitet.

***Datensatz-Anlegen-Seite* - `detail_new_page`**
Dieses Template bietet eine einfache *Anlegen-Seite* für neue Datensätze in dieser LivingApp.

Ein neuer Datensatz wird als vollständige HTML-Seite angelegt.

***Datensatz-Anzeige-Kachel* - `detail_display_panel`**
Eine *Detail-Anzeige-Kachel*, die für die Anzeige von untergeordneten Datensätzen verwendet wird (wird per Ajax nachgeladen, wenn das Bearbeiten eines Datensatzes abgebrochen wird).

***Datensatz-Bearbeiten-Kachel* - `detail_edit_panel`**
Eine *Detail-Kachel zum Bearbeiten* eines Datensatzes. Diese wird per Ajax nachgeladen, wenn der Benutzer in den untergeordneten Datensätzen auf „Bearbeiten“ klickt.

***Datensatz-Anlegen-Kachel* - `detail_new_panel`**
Eine *Detail-Kachel zum Anlegen* neuer Datensätze. Diese wird per Ajax nachgeladen, wenn der Benutzer in den untergeordneten Datensätzen auf „Neu“ klickt.

Nach dem Speichern wird dieses Panel durch das entsprechende Anzeige-Panel ersetzt.

***Datensatz-Anzeige-Kachel* - `detail_display_panel_tables`**
Zeigt einen Datensatz als *Kachel in der Tabellenansicht* der Unter-Datensätze an.

Zum Bearbeiten wird diese Kachel per Ajax durch die „Bearbeiten-Kachel“ ersetzt.

Dieses Template wird benutzt, wenn untergeordnete Datensätze als Reihe von Kacheln angezeigt werden.

***Datensatz-Bearbeiten-Kachel* - `detail_edit_panel_tables`**
Zeigt einen Datensatz zum *Bearbeiten in der Detail-Tabellen-Ansicht* an.

Nach dem Speichern wird dieses Panel geschlossen, und die Tabellenansicht neu geladen.

Dieses Template wird benutzt, wenn untergeordnete Datensätze als Reihe von Tabellen angezeigt werden.

Datensatz-Anlegen-Kachel - `detail_new_panel_tables`
Wird zum Anlegen eines *neuen Datensatzes in der Tabellenansicht* verwendet.

Nach dem Speichern wird die Tab-Leiste neu geladen.

Dieses Template wird benutzt, wenn untergeordnete Datensätze als Tabellen angezeigt werden.

Löschen-Dialog/Aktion - `record_delete`
Dieses Template bietet einen *Löschen-Dialog* sowie die zugehörige Löschen-Aktion für einen Datensatz in dieser LivingApp.

Unterdatensätze als Panels - `detail_children_panels`
Dieses Template erzeugt ein HTML-Fragment das die Unter-Datensätze eines Datensatzes als *Panels* anzeigt und die Navigationsmöglichkeiten zwischen verschiedenen Anzeige-Varianten bietet.

Unterdatensätze als Tabellen - `detail_children_tables`
Dieses Template erzeugt ein HTML-Fragment das die Unter-Datensätze eines Datensatzes als *Tabellen* anzeigt und die Navigationsmöglichkeiten zwischen verschiedenen Anzeige-Varianten bietet.

Unterdatensätze als Thumbnails - `detail_children_thumbnails`
Dieses Template erzeugt ein HTML-Fragment das die Unter-Datensätze eines Datensatzes als *Thumbnails* anzeigt und die Navigationsmöglichkeiten zwischen verschiedenen Anzeige-Varianten bietet.

Ergebnisseite mit Weiterleitung - `result_redirect`
Diese Template ist eine *Ergebnisseite*, die nach dem Abspeichern eines Datensatzes im Formular aufgerufen wird.

Dies wird dazu verwendet um zur `Listen-Seite` weiterzuleiten.

Ergebnisseite mit Datensatz-Anzeige - `result_display`
Diese Template ist eine *Ergebnisseite*, die nach dem Abspeichern eines Datensatzes im Formular aufgerufen wird.

Diese Seite zeigt den Datensatz an und bietet Links, um den Datensatz nochmals zu verändern, oder um einen neuen Datensatz anzulegen.

Support-Templates

Standard-CSS für HTML-Seiten dieser App - `css`
Diese CSS-Datei wird normalerweise in alle Ihre Seiten eingebunden wenn Sie das Standard-Library-Template `la_standard_page` für Ihre HTML-Seiten verwenden.

Sie definiert das Standard-Aussehen Ihrer HTML-Seiten.

Durch einen Aufruf des Library-Template `la_css` werden alle benötigten CSS-Regeln eingebunden. Sie können jedoch das erzeugte Template auch anpassen, um zusätzliche CSS-Regeln zu definieren, oder die CSS-Regeln der Template-Library selektiver einzubinden, um bestimmte CSS-Regeln wegzulassen.

Standard javascript für HTML-Seiten dieser App - `js`
Diese Javascript-Datei wird normalerweise in alle Ihre Seiten eingebunden wenn Sie das Standard-Library-Template `la_standard_page` für Ihre HTML-Seiten verwenden.

Sie definiert Standard-Javascript-Funktionalität für Popups, Overlays, Menüs und sonstiges dynamisches Verhalten.

Ajax-Suche für Applookup-Feld - `field_..._search`
Für Felder vom Typ `applookup/choice` oder `multipleapplookup/choice` kann der Benutzer nach den gewünschten Datensätzen suchen.

Mit dieser Vorlage können Sie für die in Ihrer LivingApp vorhandenen Felder des obigen Typs geeignete Anzeige-Templates anlegen, die dann über Ajax zur Suche in den jeweiligen Ziel-LivingApps verwendet werden.

Sonstiges

Beispiel Notizzettel für Ajax - `example_note`

Ein einfaches Beispiel für einen Notizzettel der per Ajax nachgeladen werden kann.

Es wird nur ein HTML-Schnipsel erzeugt, der dann per Javascript in die aktuelle Seite eingebunden wird.

Ein „Link“ um diesen Notizzettel beim Klicken anzuzeigen kann z.B. so aussehen:

```
<?code url = globals.app.template_url("example_note")?>
   <span class="la-look-link" onclick="la_popup(this, event)" data-url="<?
↪printx url?>">
      Wichtige Notiz
   </span>
```

Beispiel Menü für Ajax - `example_menu`

Ein einfaches Beispiel für ein Menü das per Ajax nachgeladen werden kann.

Es wird nur ein HTML-Schnipsel erzeugt, der dann per Javascript in die aktuelle Seite eingebunden wird.

Ein „Link“ um dieses Menü beim Klicken anzuzeigen kann z.B. so aussehen:

```
<?code url = globals.app.template_url("example_menu")?>
   <span class="la-look-link" onclick="la_popup(this, event)" data-url="<?
↪printx url?>">
      Menü
   </span>
```

5.5.2 Anwendungsbeispiele

In den folgenden Anwendungsbeispielen werden anhand der LivingApps *Veranstaltungen* und *Teilnehmer* die einzelnen Templatevorlagen veranschaulicht.

Zur *Auswahl der jeweiligen Templatevorlagen*, gehen Sie wie oben beschrieben vor.

Listen-Seite - `list_page`

Zuerst wird eine `Listen-Seite` erstellt, mit der die Daten der LivingApp in *Tabellenform* ausgegeben werden.

In unserem Beispiel sollen die Veranstaltungsdaten aufgelistet sein. Standardmäßig angezeigt werden alle Felder, die Sie unter *Konfiguration - Liste* als Listenfelder ausgewählt haben.

Mit der `Listen-Seite` können Sie Ihre Daten anzeigen und innerhalb dieser Liste suchen.

Damit der Link zur `Detail-Anzeigen-` bzw. `-Bearbeiten-Seite` funktioniert, sollten Sie auch die *Anzeige-Templates* `detail_page` und `detail_edit_page` anlegen.

Um neue Datensätze über die `Listen-Seite` anlegen zu können, ist das Anzeige-Template `detail_new_page` erforderlich.

Für die *Lösch-Funkionalität* benötigen Sie das Anzeige-Template `record_delete`.

Um die Listen-Seite zu erstellen, haben Sie folgende Konfigurationsmöglichkeiten:

Identifikation

Identifizierer

Die eindeutige Kennung für dieses Anzeige-Template.

Bezeichnung

Über diese Bezeichnung finden Sie das Template in der Menüzeile und in der Fußzeile der Datenliste (nur wenn Sie beim *Typ Liste mit Menüeintrag* gewählt haben).

Eigenschaften

Anzeige-Template aus Vorlage "Listen-Seite" hinzufügen

Diese Template bietet eine einfache Tabellen-Ansicht der Daten in dieser LivingApp.

Damit der Link zur Detail-Anzeigen- bzw. -Bearbeiten-Seite funktioniert, sollten Sie auch die Anzeige-Templates `detail_page` und `detail_edit_page` anlegen.

Für die Lösch-Funkionalität benötigen Sie das Anzeige-Template `record_delete`.

Identifikation

Identifizierer list_page
Eine eindeutige Kennung für dieses Anzeige-Template.

Bezeichnung Listen-Seite
Icon/Bezeichnung für den Link im Datenmanagement und im Menü.

Eigenschaften

* Typ ○ Liste ○ Liste (Standard) ◉ Liste (mit Menüeintrag)
Wo steht das Anzeige-Template zur Verfügung, und wie kann es angesprochen werden?

* Zugriff erlauben für User mit Rechten auf der Applikation

Erzeugung

* Erzeugen? ◉ Anlegen ○ Anlegen wenn nicht vorhanden ○ Anlegen und ggf. vorhandenes Template ersetzen
Soll das Anzeige-Template ersetzt werden, wenn es bereits existiert?

* Benötigte Templates erzeugen? ○ Nicht anlegen ◉ Anlegen wenn nicht vorhanden ○ Anlegen und ggf. vorhandenes Template ersetzen
`list_page` verlinkt auf die Anzeige-Templates `css` und `js`. Hiermit legen Sie fest, wann diese Anzeige-Templates erzeugt werden sollen.

Speichern Abbrechen

Abb. 21: Listen-Seite konfigurieren

Typ

Wie ein Template, je nach Typ, angesprochen werden kann, können Sie unter *Aufruf eines Anzeige-Templates* nachlesen.

Beim Typ *Liste (mit Menüeintrag)* steht der Link zum Anzeige-Template in der *Menüzeile* und in der Fußzeile der Datenliste zur Verfügung.

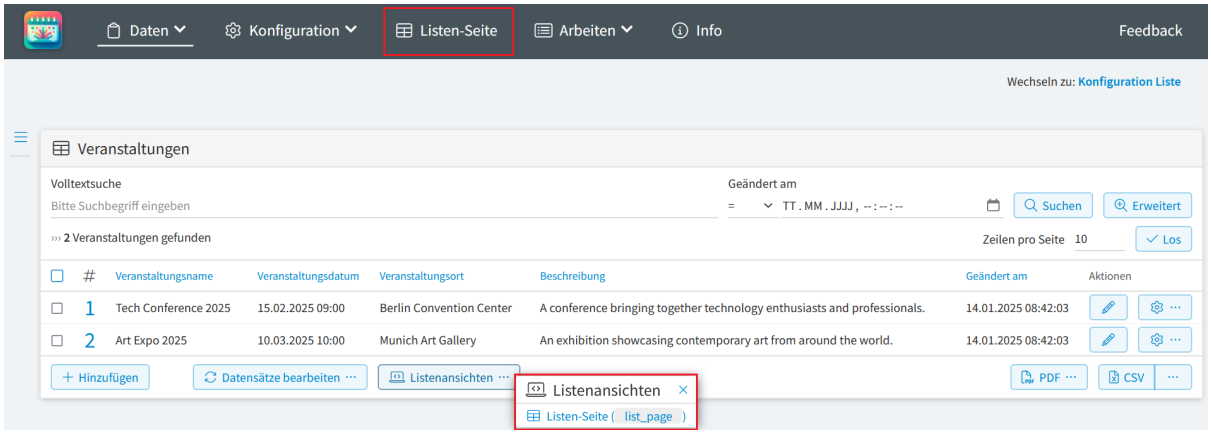

Abb. 22: Listen-Seite - Link in Menüzeile

Zugriff erlauben für

Hier legen Sie fest, welche Berechtigung ein Nutzer haben muss, um auf das Template zugreifen zu können.

Erzeugung

Erzeugen?

Legen Sie hier fest, ob das Anzeige-Template ersetzt werden soll, wenn es bereits existiert.

Benötigte Templates erzeugen?

`list_page` verlinkt auf die Anzeigetemplates `css` und `js`. Legen Sie hier fest, wann diese erzeugt werden sollen.

Das *Ergebnis* sieht dann folgendermaßen aus:

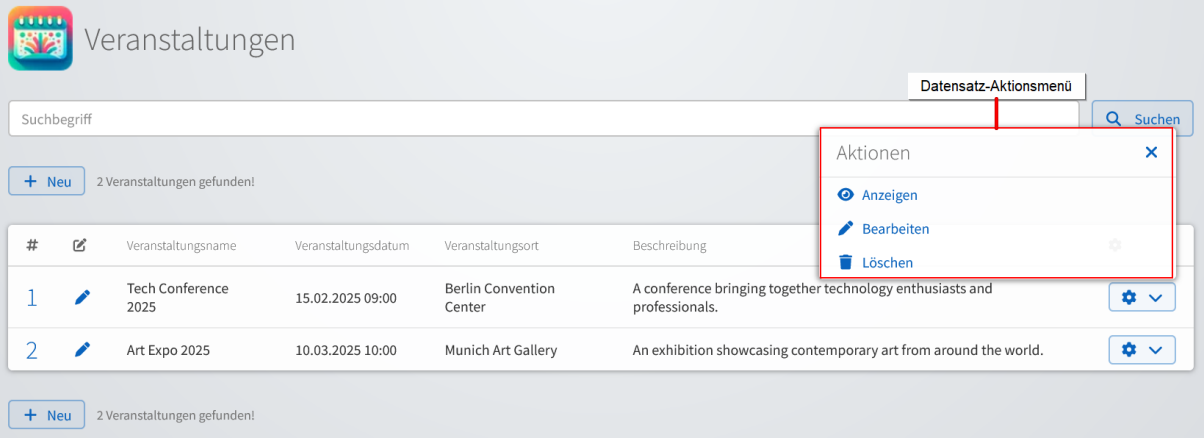

Abb. 23: Listen-Seite - Ergebnis

Damit die Links im Datensatz-Aktions-Menü zu *Anzeigen*, *Bearbeiten* und *Löschen* funktionieren, müssen Sie auch die *Anzeige-Templates* `detail_page`, `detail_edit_page` und `record_delete` anlegen. Um einen neuen Datensatz anzulegen, wird zusätzlich das Anzeige-Template `detail_new_page` benötigt.

Um die Anzeigetemplates `detail_page`, `detail_edit_page` und `detail_new_page` anzulegen, klicken Sie in der Auswahl der Templatevorlagen auf *Verschiedene Datensatz-Seiten*.

Es öffnet sich folgendes Fenster:

Anzeige-Template aus Vorlage hinzufügen für verschiedene Datensatz-Seiten

Diese Anzeige-Template zeigt einen Datensatz in der einen oder anderen Form an.

Welche Form das ist, wird durch „**Kontext**“ und „**Modus**“ festgelegt.

Verwendung

* Kontext
- (●) Vollständige HTML-Seite
- ( ) Panel als HTML-Fragment (in einer Reihe von Panels)
- ( ) Panel als HTML-Fragment (zu einer Tabelle von Datensätzen)

In welchen Kontext wird dieses Template verwendet?

* Modus
- (●) Anzeigen
- ( ) Bearbeiten
- ( ) Anlegen

Identifikation

Identifizierer `detail_page`

Eine eindeutige Kennung für dieses Anzeige-Template.

Bezeichnung Datensatz-Anzeige-Seite

Icon/Bezeichnung für den Link im Datenmanagement.

Eigenschaften

* Typ (●) Detail ( ) Detail (mit Menüeintrag im Datenmanagement)

Wo steht das Anzeige-Template zur Verfügung, und wie kann es angesprochen werden?

* Zugriff erlauben für User mit Rechten auf der Applikation

Erzeugung

* Erzeugen? (●) Anlegen ( ) Anlegen wenn nicht vorhanden ( ) Anlegen und ggf. vorhandenes Template ersetzen

Soll das Anzeige-Template ersetzt werden, wenn es bereits existiert?

* Benötigte Templates erzeugen? ( ) Nicht anlegen (●) Anlegen wenn nicht vorhanden ( ) Anlegen und ggf. vorhandenes Template ersetzen

Wird der Kontext „**Vollständige HTML-Seite**“ verwendet, so verlinkt dieses Anzeige-Template auf die Anzeige-Templates `css` und `js`. Hiermit legen Sie fest, wann diese Anzeige-Templates erzeugt werden sollen.

Speichern Abbrechen

Abb. 24: Datensatz-Seite konfigurieren

Verwendung

Kontext

Vollständige HTML-Seite
Datensatz wird als vollständige HTML-Seite ausgegeben.

Panel als HTML-Fragment (in einer Reihe von Panels)
Datensatz wird als Panel in einer Reihe von Panels ausgegeben.

Panel als HTML-Fragment (zu einer Tabelle von Datensätzen)
Datensatz wird als Panel zu einer Tabelle von Datensätzen ausgegeben.

Modus

Anzeigen
Datensatz wird angezeigt.

Bearbeiten
Datensatz kann bearbeitet werden.

Anlegen
Neuer Datensatz kann angelegt werden.

Identifikation

Identifizierer
Die eindeutige Kennung für diese Template.

Bezeichnung
Die Bezeichnung für den Link im Datenmanagement. Der Link wird nur angezeigt, wenn bei *Typ Detail (mit Menüeintrag im Datenmanagement)* gewählt wurde. Der Link kann dann im Datenmanagement, im Einstellungs-Menü rechts von jedem Datensatz ausgewählt werden.

Eigenschaften

Typ
Wo steht das Anzeige-Template zur Verfügung und wie kann es angesprochen werden?

Detail
Ein Detail-Template wird aufgerufen mit einer URL in der Form:

```
https://www.living-apps.de/gateway/apps/563a2616591c6ca7a9800ef2/
↪datensatz_id?template=mein_identifizierer
```

`datensatz_id` ist dabei die Datenbank-ID des Datensatzes und `mein_identifizierer` der Identifizierer des Detail-Templates.

Haben Sie in Ihrer LivingApp eine *Listen-Seite* - `list_page`, dann können Sie das *Anzeige-Template* auch über das *Datensatz-Aktions-Menü* aufrufen.

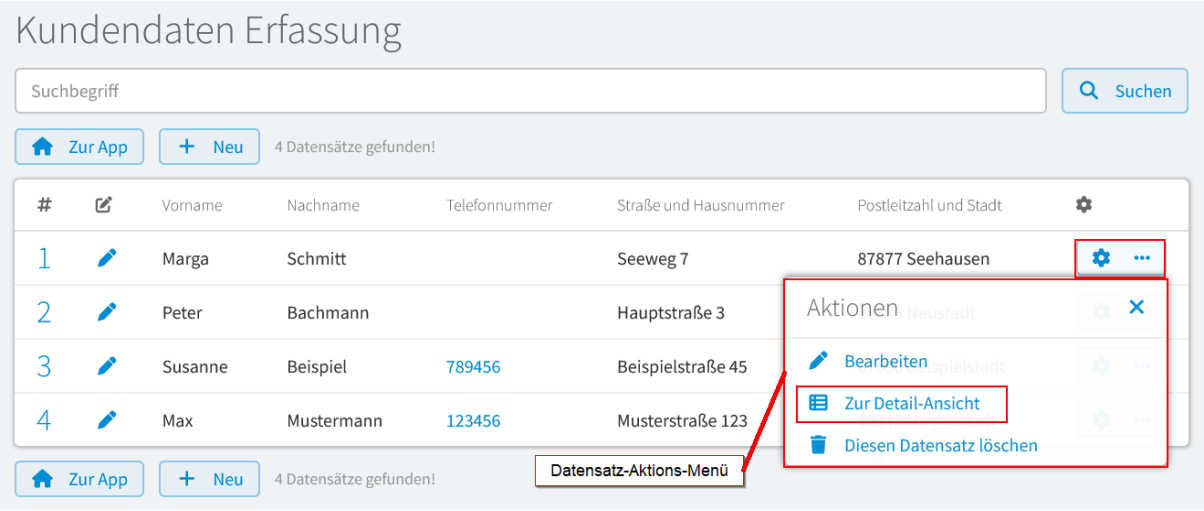

Abb. 25: Link im Aktionsmenü der Listen-Seite

Detail (mit Menüeintrag im Datenmanagement)
Das Anzeige-Template steht als Link im Datenmanagement zur Verfügung und zwar im Einstellungs-Menü rechts von jedem Datensatz.

Zugriff erlauben für
Hier legen Sie fest, welche Berechtigung ein Nutzer haben muss, um auf das Template zugreifen zu können.

Erzeugung

Erzeugen?
Legen Sie hier fest, ob das Anzeige-Template ersetzt werden soll, wenn es bereits existiert.

Benötigte Templates erzeugen?
Dieses Anzeige-Template verlinkt auf die Anzeigetemplates `css` und `js`. Legen Sie hier fest,

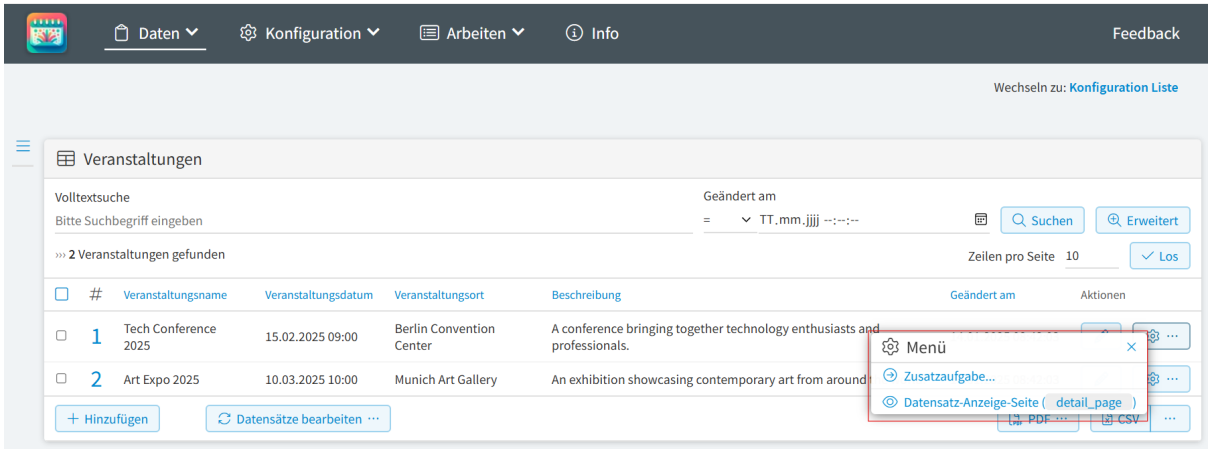

Abb. 26: Link im Datenmanagement

wann diese erzeugt werden sollen.

***Datensatz-Anzeige-Seite* - `detail-page`**

Um die *Datensatz-Anzeige-Seite* zu erstellen, wählen Sie bei *Kontext Vollständige HTML-Seite* und bei *Modus Anzeigen* aus und klicken dann auf *Speichern*.

Die *Datensatz-Anzeige-Seite* kann aufgerufen werden, indem Sie auf der *Listen-Seite* in der Tabelle im Datensatzaktionsmenü auf *Anzeigen* klicken.

Haben Sie in der Konfiguration der *Datensatz-Anzeige-Seite* bei *Typ Detail (mit Menüeintrag im Datenmanagement)* gewählt, steht das Anzeige-Template auch als *Link im Datenmanagement* zur Verfügung und zwar im Einstellungs-Menü rechts von jedem Datensatz.

Ergebnis Datensatz-Anzeige-Seite:

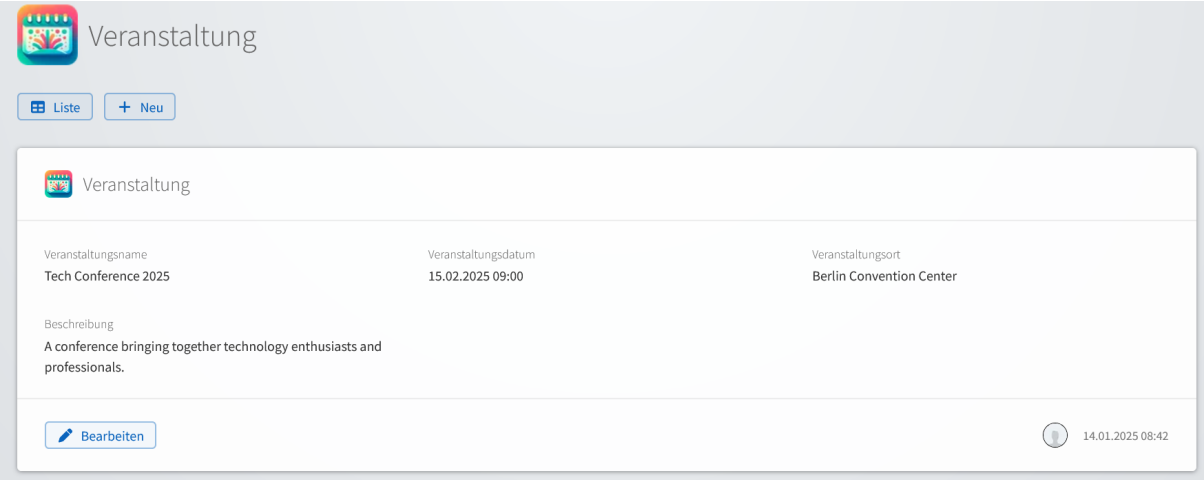

Abb. 27: Datensatz-Anzeige-Seite - Ergebnis

***Datensatz-Bearbeiten-Seite* - `detail_edit_page`**

Um die *Datensatz-Bearbeiten-Seite* zu erstellen, wählen Sie bei *Kontext Vollständige HTML-Seite* und bei *Modus Bearbeiten* aus und klicken dann auf *Speichern*.

Die *Datensatz-Bearbeiten-Seite* kann aufgerufen werden, indem Sie auf der *Listen-Seite* in der Tabelle links auf den Stift oder im Datensatzaktionsmenü auf *Bearbeiten* klicken.

Haben Sie in der Konfiguration der *Datensatz-Bearbeiten-Seite* bei *Typ Detail (mit Menüeintrag im Datenmanagement)* gewählt, steht das Anzeige-Template auch als *Link im Datenmanagement* zur Verfügung und zwar im Einstellungs-Menü rechts von jedem Datensatz.

Ergebnis Datensatz-Bearbeiten-Seite:

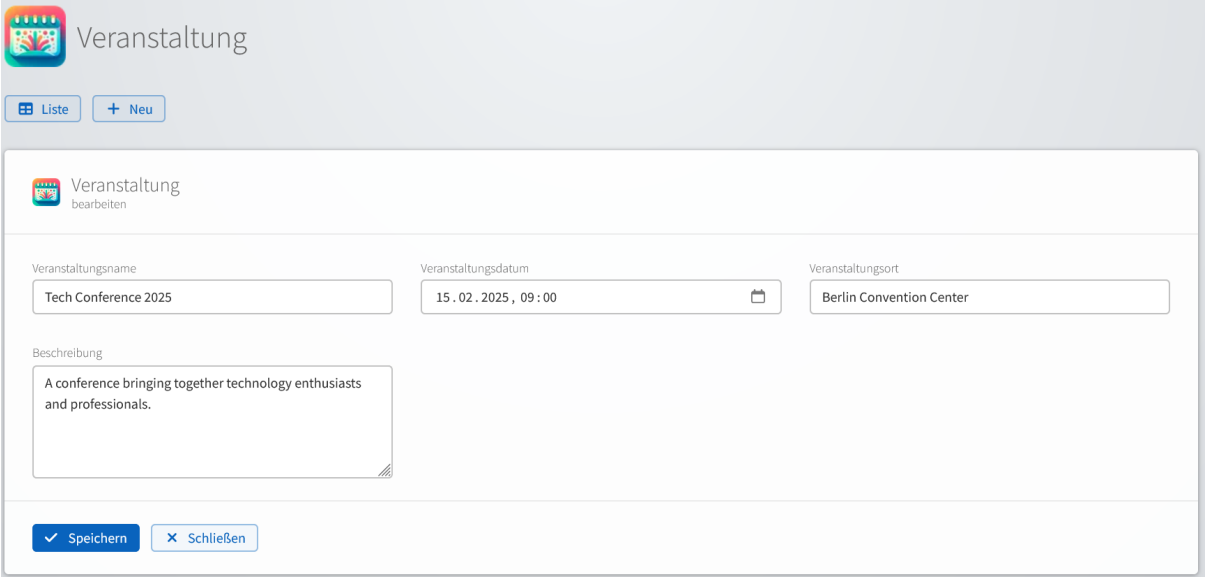

Abb. 28: Datensatz-Bearbeiten-Seite - Ergebnis

***Datensatz-Anlegen-Seite* - `detail_new_page`**

Um die *Datensatz-Anlegen-Seite* zu erstellen, wählen Sie bei *Kontext Vollständige HTML-Seite* und bei *Modus Anlegen* aus und klicken dann auf *Speichern*.

Die *Datensatz-Anlegen-Seite* kann aufgerufen werden, indem Sie auf der *Listen-Seite* auf + *Neu* klicken.

Haben Sie in der Konfiguration der *Datensatz-Anlegen-Seite* bei *Typ Detail (mit Menüeintrag im Datenmanagement)* gewählt, steht das Anzeige-Template auch als *Link im Datenmanagement* zur Verfügung und zwar im Einstellungs-Menü rechts von jedem Datensatz.

Ergebnis der Datensatz-Anlegen-Seite:

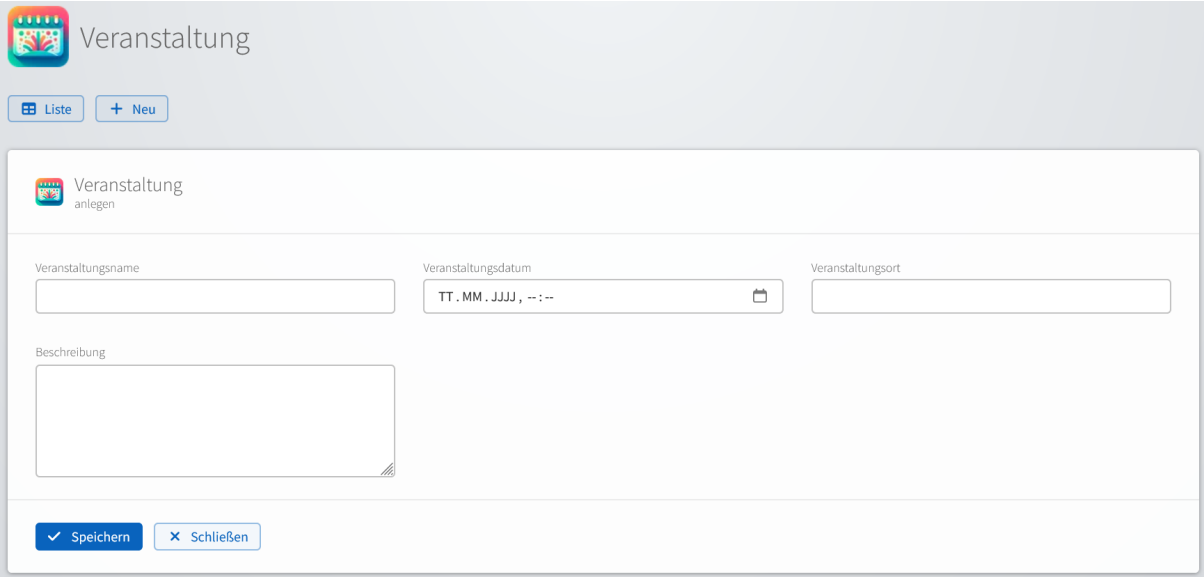

Abb. 29: Datensatz-Anlegen-Seite - Ergebnis

***Löschen-Dialog/Aktion* - `record_delete`**

Diese Template bietet einen Löschen-Dialog sowie die zugehörige Löschen-Aktion für einen Datensatzes in dieser LivingApp.

Um dieses Anzeige-Template anzulegen, wählen Sie in der Übersicht der Templatevorlagen *Löschen-Dialog/Aktion* aus.

Es öffnet sich folgendes Fenster:

Anzeige-Template aus Vorlage "Datensatz löschen" hinzufügen

Diese Template bietet einen Löschen-Dialog sowie die zugehörige Löschen-Aktion für einen Datensatzes in dieser LivingApp.

Identifikation

Identifizierer `record_delete`
Die eindeutige Kennung für das Anzeige-Template.

Bezeichnung Löschen-Dialog/Aktion
Icon/Bezeichnung für das Template.

Eigenschaften

* Zugriff erlauben für User mit Rechten auf der Applikation

Erzeugung

* Erzeugen? Anlegen / Anlegen wenn nicht vorhanden / Anlegen und ggf. vorhandenes Template ersetzen
Soll das Anzeige-Template ersetzt werden, wenn es bereits existiert?

* Benötigte Templates erzeugen? Nicht anlegen / Anlegen wenn nicht vorhanden / Anlegen und ggf. vorhandenes Template ersetzen
`record_delete` verlinkt auf die Anzeige-Templates `css` und `js`. Hiermit legen Sie fest, wann diese Anzeige-Templates erzeugt werden sollen.

Speichern Abbrechen

Abb. 30: Löschen-Dialog/Aktion konfigurieren

Hier müssen keine weiteren Konfigurationen vorgenommen werden. Klicken Sie auf *Speichern*.

Die *Löschen-Dialog/Aktion* kann aufgerufen werden, indem Sie auf der *Listen-Seite* im Datensatzaktionsmenü auf *Löschen* klicken.

Ergebnis der Löschen-Dialog/Aktion:

Möchten Sie bei verknüpften Apps auch die untergeordneten Datensätze anzeigen und bearbeiten können, müssen Sie noch folgende Anzeige-Templates anlegen und zwar in der LivingApp, die die untergeordneten Datensätze enthält.

In unserem Beispiel sollen alle Teilnehmer, die sich zu einer Veranstaltung angemeldet haben, mit angezeigt werden, wenn in der LivingApp „Veranstaltungen“ ein Detail-Datensatz angezeigt wird.

Dafür werden die Anzeige-Templates `detail_display_panel`, `detail_edit_panel` und `detail_new_panel` in der LivingApp „Teilnehmer“ angelegt.

Das Ergebnis wird ersichtlich, wenn Sie in der übergeordneten LivingApp (in unserem Beispiel „Veranstaltungen“) auf der *Listen-Seite* im Datensatz-Aktions-Menü eines Datensatzes auf *Anzeigen* klicken. Es erscheinen nun unter der Datensatz-Anzeige-Seite des gewählten Datensatzes, die verknüpften Informationen zu diesem Datensatz.

In unserem Beispiel werden unterhalb der Datensatz-Anzeige-Seite der Veranstaltung, die Sie gewählt haben, alle Teilnehmer angezeigt, die sich zu dieser Veranstaltung angemeldet haben.

***Datensatz-Anzeige-Kachel* - `detail_display_panel`**
Dieses Template bietet eine einfache Anzeige-Kachel für einen Datensatz in dieser LivingApp.

Abb. 31: Datensatz löschen

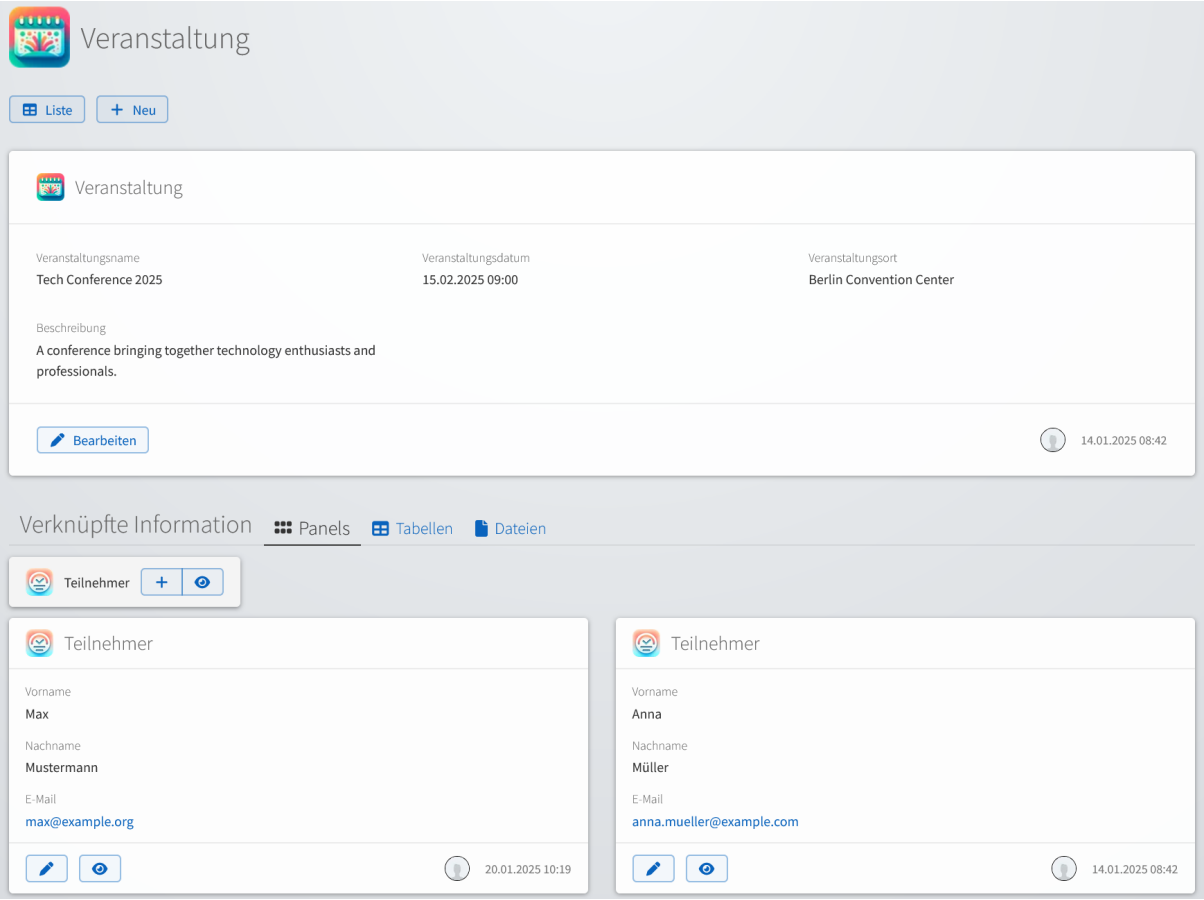

Abb. 32: Untergeordnete Datensätze - Kachel

Dieses Template erzeugt keine vollständige HTML-Seite sondern wird von `detail_edit_panel` durch einen Ajax-Request geholt, wenn Sie das Bearbeiten abbrechen.

Dazu müssen Sie das Anzeige-Template für das Bearbeiten-Panel (`detail_edit_panel`) erzeugt haben.

Um die *Datensatz-Anzeige-Kachel* zu erstellen, wählen Sie bei *Kontext Panel als HTML-Fragment (in einer Reihe von Panels)* und bei *Modus Anzeigen* aus und klicken dann auf *Speichern*.

***Datensatz-Bearbeiten-Kachel* - `detail_edit_panel`**

Dieses Template bietet ein einfache Bearbeiten-Kachel für einen Datensatz in dieser LivingApp.

Diese wird per Ajax nachgeladen, wenn der Benutzer in den untergeordneten Datensätzen auf „Bearbeiten“ klickt.

Dazu müssen Sie das Anzeige-Template für die Anzeige-Kachel (`detail_display_panel`) erzeugt haben.

Um die *Datensatz-Bearbeiten-Kachel* zu erstellen, wählen Sie bei *Kontext Panel als HTML-Fragment (in einer Reihe von Panels)* und bei *Modus Bearbeiten* aus und klicken dann auf *Speichern*.

Um die *Datensatz-Bearbeiten-Kachel* aufzurufen, klicken Sie in der Kachel, deren Daten Sie bearbeiten wollen auf den Stift.

Die Bearbeitung des Datensatzes wird in der Datensatz-Anzeigemaske ausgeführt.

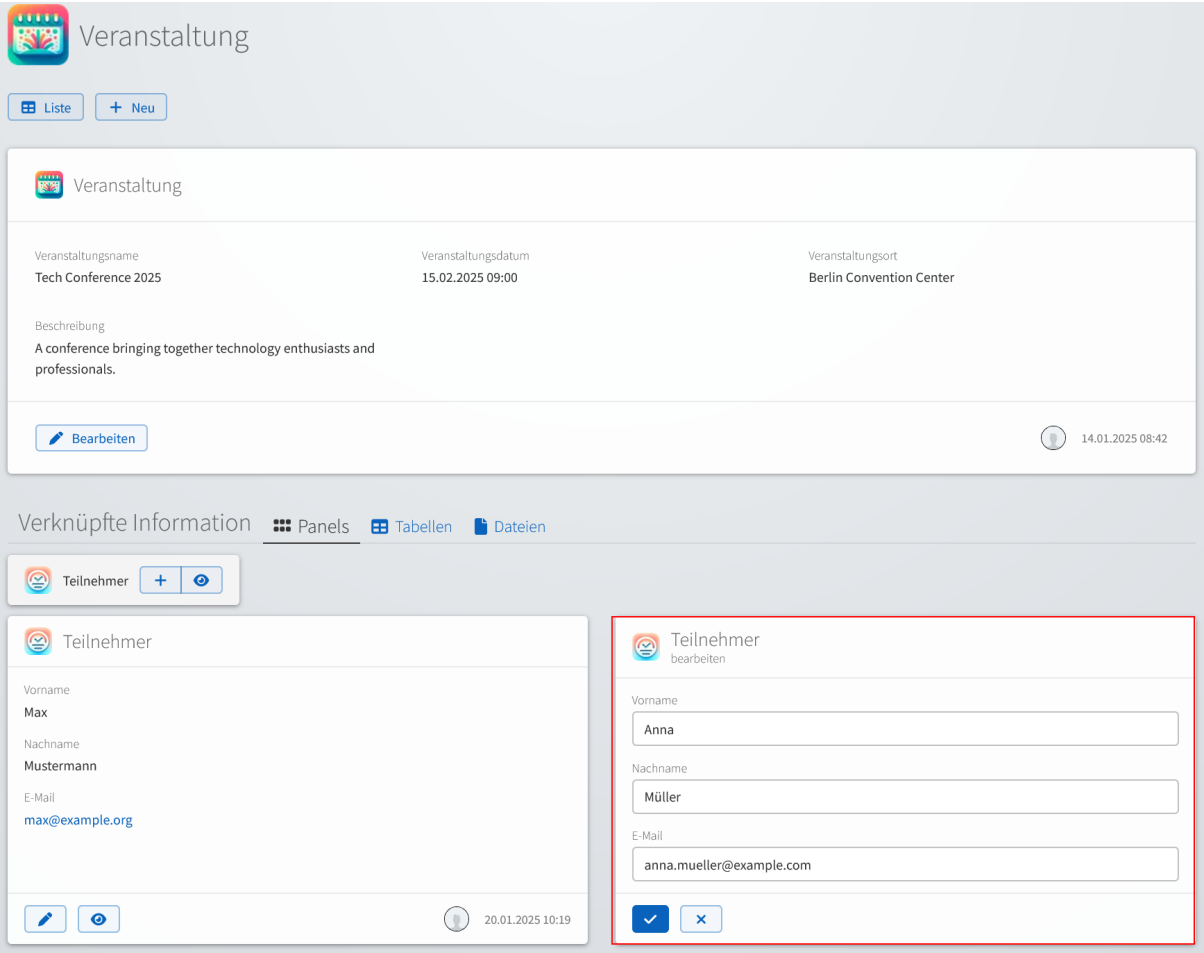

Abb. 33: Untergeordnete Datensätze - Kachel bearbeiten

***Datensatz-Anlegen-Kachel* - `detail_new_panel`**

Wird zum Anlegen eines neuen Datensatzes als Panel verwendet.

Nach dem Speichern wird dieses Panel durch das entsprechende Anzeige-Panel ersetzt.

Um die *Datensatz-Anlegen-Kachel* zu erstellen, wählen Sie bei *Kontext Panel als HTML-Fragment (in einer Reihe von Panels)* und bei *Modus Anlegen* aus und klicken dann auf *Speichern*.

Um das Anzeige-Template aufzurufen, klicken Sie auf das + bei der verknüpften LinvingApp. In unserem Beispiel, kann so ein neue Teilnehmer zu dieser Veranstaltung angelegt werden.

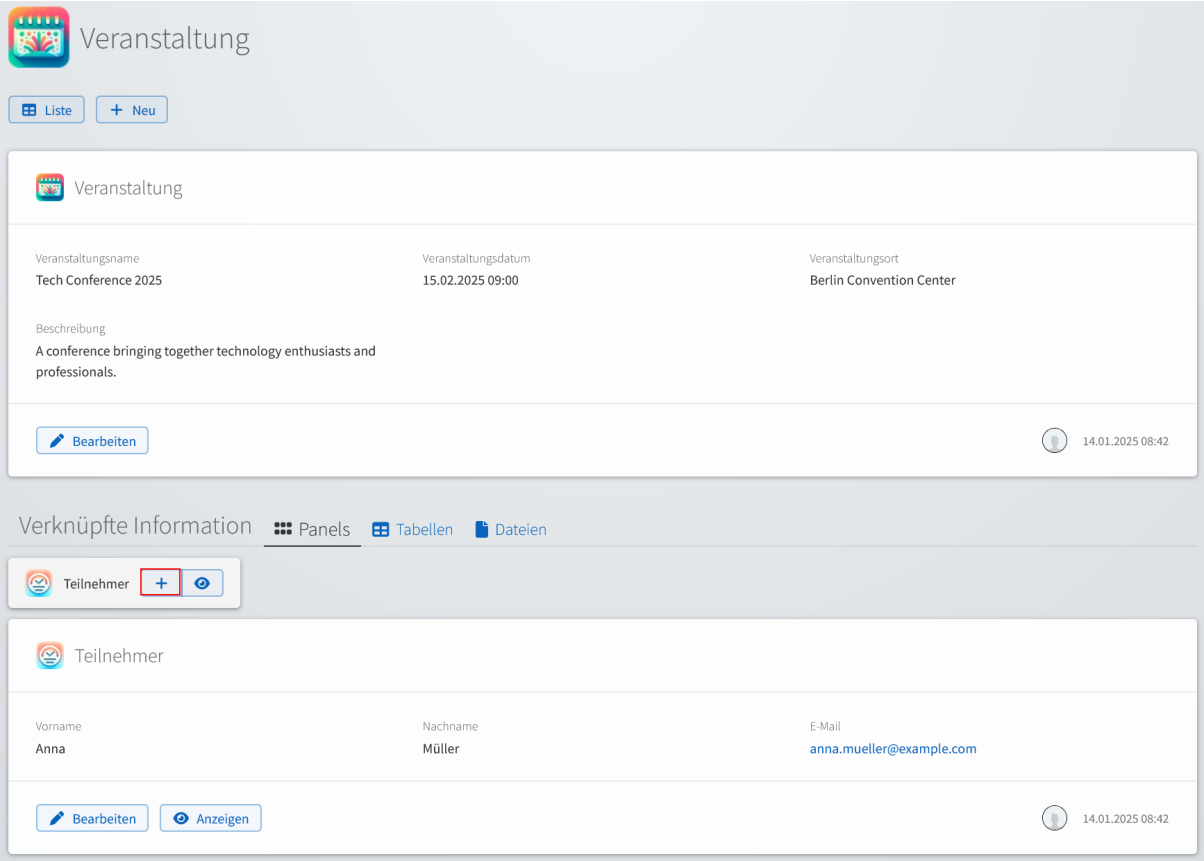

Abb. 34: Datensatz-Anlegen-Kachel aufrufen

Es öffnet sich ein Panel zur Dateneingabe.

In der Ansicht der untergeordneten Datensätze gibt es auch die Möglichkeit, die untergeordneten Datensätze als Tabelle anzuzeigen, zu bearbeiten und neue untergeordnete Datensätze in Tabellenform zu erfassen. Außerdem haben Sie die Möglichkeit, wenn es in Ihrer untergeordneten LivingApp ein „Upload“-Feld gibt, die hochgeladenen Dateien zu einem Datensatz anzeigen zu lassen.

Möchten Sie diese Ansichten und die Navigationsmöglichkeit nutzen, werden noch zusätzlich folgende Anzeige-Templates benötigt.

Diese müssen in der LivingApp angelegt werden, die die untergeordneten Datensätze enthält. In unserem Beispiel in der LivingApp „Teilnehmer“. Zur Auswahl der Templatevorlagen gehen Sie wie oben beschrieben vor. Wählen Sie *Verschiedene Datensatzseiten* aus der *Liste der Templatevorlagen* aus.

***Detail-Anzeigen-Kachel* für Tabellen-Ansicht - `detail_display_panel_tables`**

Eine Detail-Kachel zum Anzeigen eines untergeordneten Datensatzes in Tabellen- Ansicht. Diese wird per Ajax nachgeladen, wenn der Benutzer in den untergeordneten Datensätzen in der Tabellen-Ansicht auf „Anzeigen“ klickt.

Zur Erstellung dieses Anzeige-Templates wählen Sie bei *Kontext Panel als HTML-Fragment (in einer Tabelle von Datensätzen)* und bei *Modus Anzeigen* aus und klicken dann auf *Speichern*.

***Detail-Bearbeiten-Kachel* für Tabellen-Ansicht - `detail_edit_panel_tables`**

Eine Detail-Kachel zum Bearbeiten eines untergeordneten Datensatzes in Tabellen-Ansicht. Diese wird per Ajax nachgeladen, wenn der Benutzer in den untergeordneten Datensätzen auf „Bearbeiten“ klickt.

Zur Erstellung diese Anzeige-Templates wählen Sie bei *Kontext Panel als HTML-Fragment (in einer Tabelle von Datensätzen)* und bei *Modus Bearbeiten* aus und klicken dann auf *Speichern*.

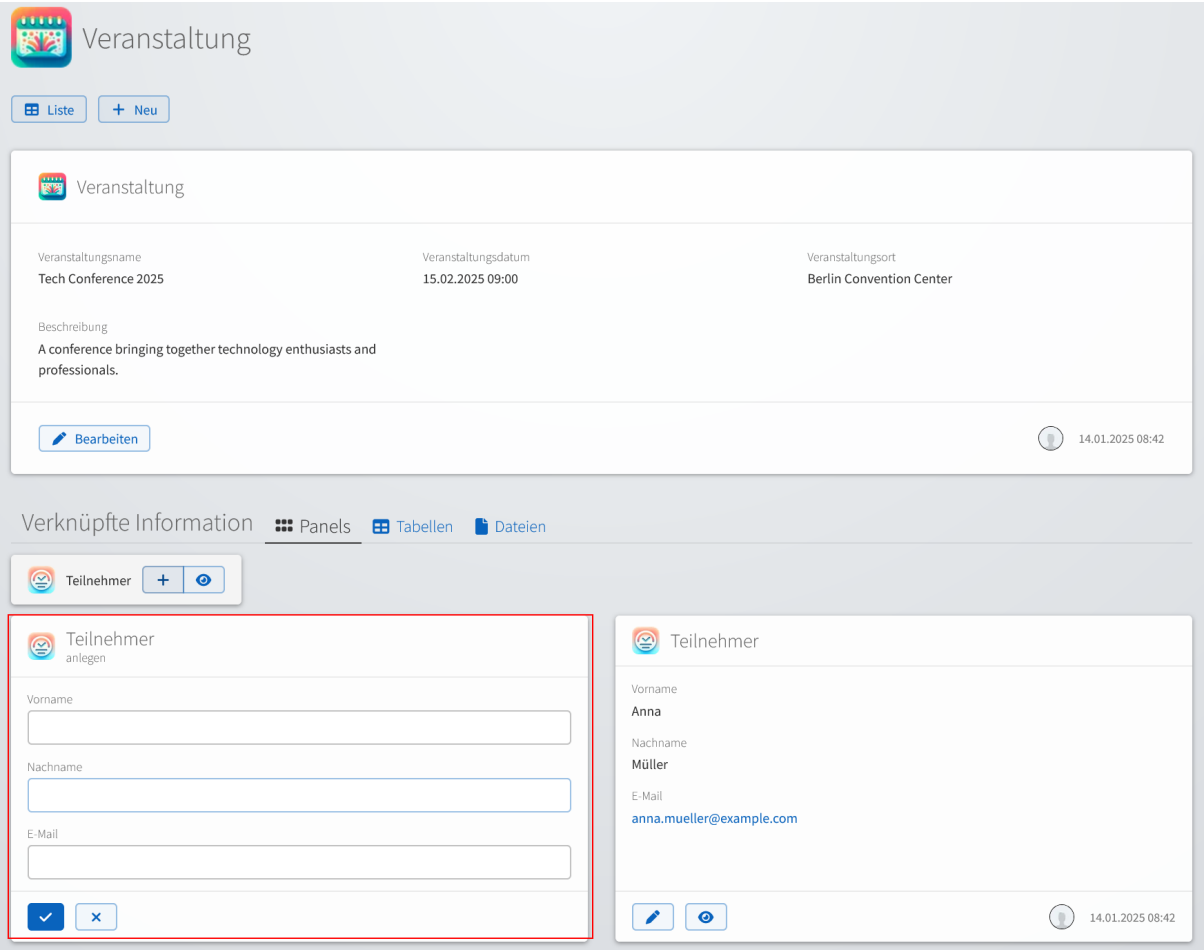

Abb. 35: Datensatz-Anlegen-Kachel - Ergebnis

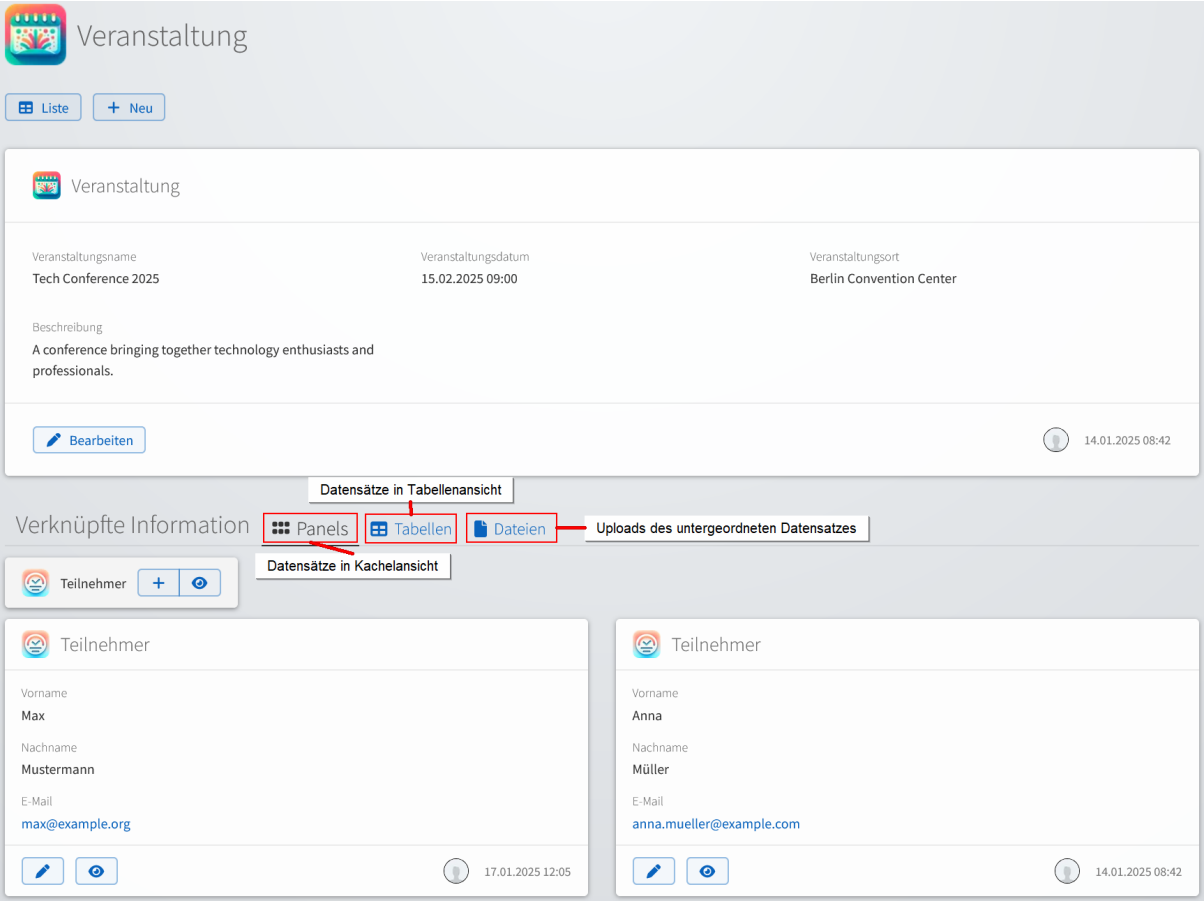

Abb. 36: Navigationsmöglichkeiten zwischen Anzeigevarianten

***Detail-Anlegen-Kachel* für Tabellen-Ansicht - `detail_new_panel_tables`**

Eine Detail-Kachel zum Anlegen neuer untergeordneter Datensätze in Tabellen-Ansicht. Diese wird per Ajax nachgeladen, wenn der Benutzer in den untergeordneten Datensätzen in der Tabellen-Ansicht auf „Neu“ klickt.

Zur Erstellung dieses Anzeige-Templates wählen Sie bei *Kontext Panel als HTML-Fragment (in einer Tabelle von Datensätzen)* und bei *Modus Anlegen* aus und klicken dann auf *Speichern*.

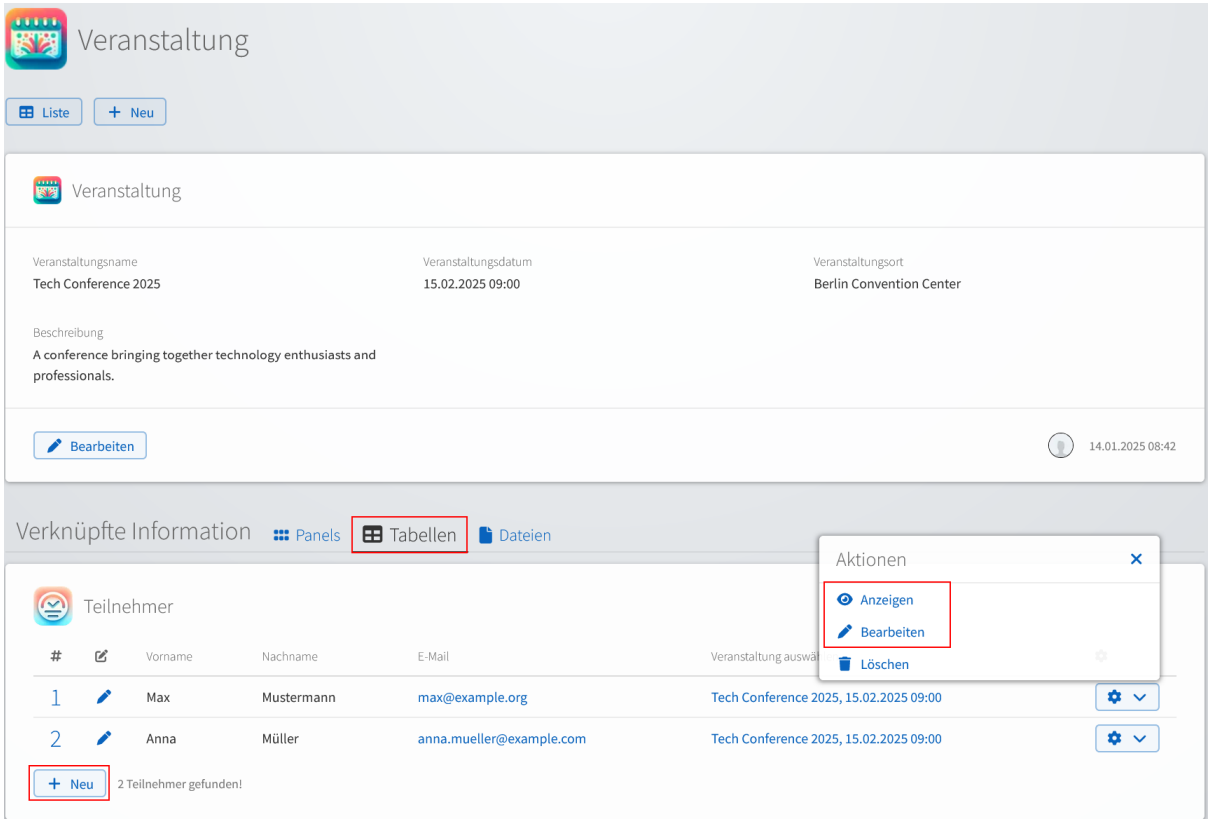

Abb. 37: Untergeordnete Datensätze in Tabellen-Ansicht - anzeigen, bearbeiten, neu anlegen

***Unter-Datensätze als Panels* - `detail_children_panels`**

Unterdatensätze eines Datensatzes werden als Panels angezeigt. Außerdem gibt es eine Navigationsmöglichkeit zwischen den verschiedenen Anzeige-Varianten (Panels, Tabellen, Dateien).

Zur Erstellung dieses Anzeige-Templates klicken Sie in der Liste der Templatevorlagen (in der untergeordneten LivingApp) auf *Unter-Datensätze als Panels* und dann in der Übersicht der Konfigurationen auf *Speichern*.

***Unter-Datensätze als Tabelle* - `detail_children_tables`**

Unterdatensätze eines Datensatzes werden in Tabellen angezeigt. Außerdem gibt es eine Navigationsmöglichkeit zwischen den verschiedenen Anzeige-Varianten (Panels, Tabellen, Dateien).

Zur Erstellung dieses Anzeige-Templates klicken Sie in der Liste der Templatevorlagen (in der untergeordneten LivingApp) auf *Unter-Datensätze als Tabelle* und dann in der Übersicht der Konfigurationen auf *Speichern*.

***Unter-Datensätze als Thumbnails* - `detail_children_thumbnails`**

Upload-Felder von Unter-Datensätzen eines Datensatzes werden als Thumbnails angezeigt. Außerdem gibt es eine Navigationsmöglichkeit zwischen den verschiedenen Anzeige-Varianten (Panels, Tabellen, Dateien).

Zur Erstellung dieses Anzeige-Templates klicken Sie in der Liste der Templatevorlagen (in der untergeordneten LivingApp) auf *Unter-Datensätze als Thumbnails* und dann in der Übersicht der Konfigurationen auf *Speichern*.

Mit Klick auf *Dateien* können Sie alle Dateien, die in den untergeordneten LivingApps hochgeladen wurden,

anzeigen lassen.

In unserem Beispiel werden Rechnungen angezeigt, die zu den Teilnehmern hochgeladen wurden.

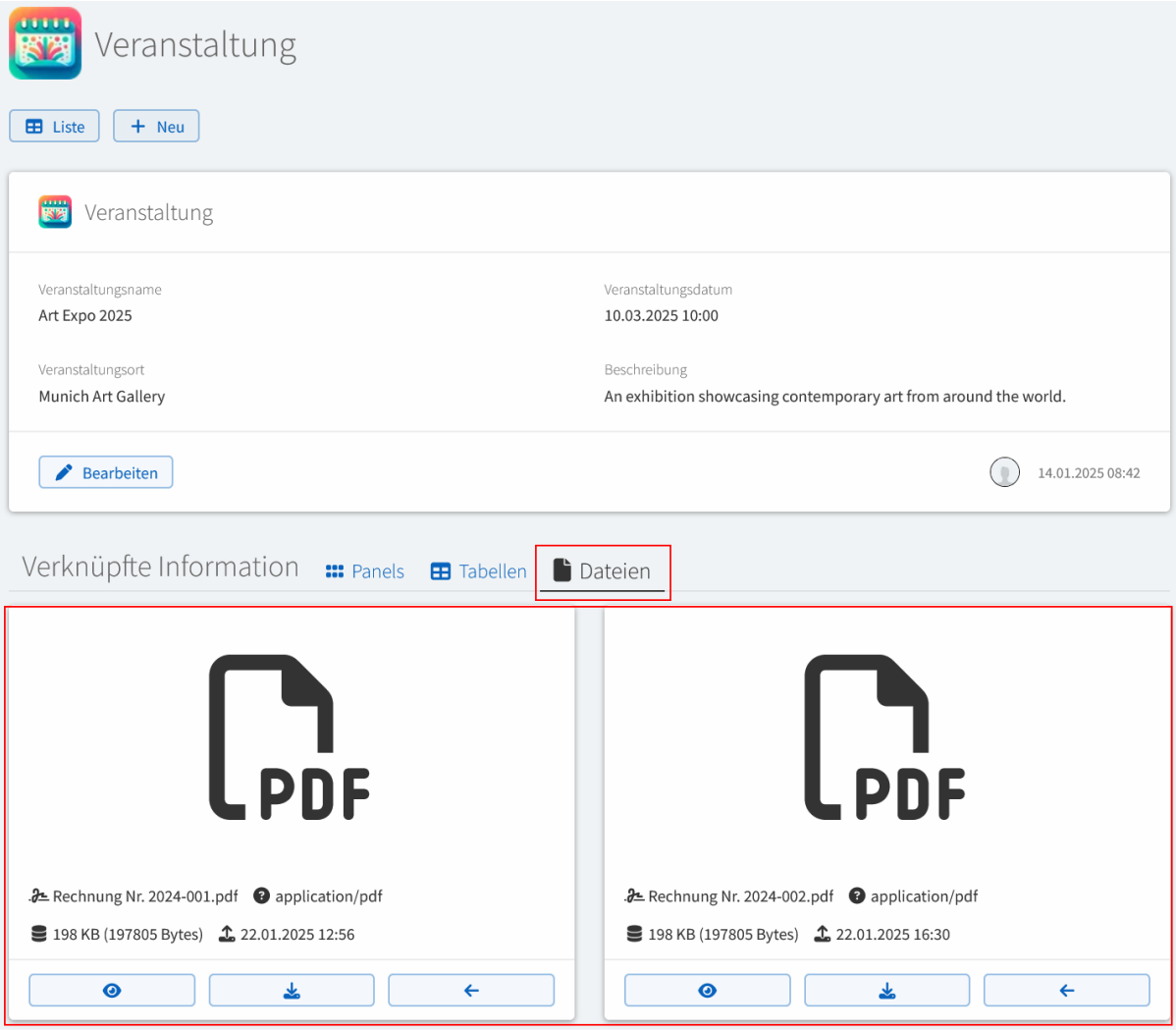

Abb. 38: Unter-Datensätze als Thumbnails - Ergebnis

Ergebnisseite mit Weiterleitung - `result_redirect`

Diese Template ist eine Ergebnisseite, die nach dem Abspeichern eines Datensatzes im Formular aufgerufen wird.

Dies wird dazu verwendet um zur *Listen-Seite* weiterzuleiten.

Um dieses Anzeige-Template zu erstellen, wählen Sie in der Übersicht der Templatevorlagen *Ergebnisseite mit Weiterleitung* aus.

Es öffnet sich folgendes Fenster.

Identifikation

Identifizierer

Die eindeutige Kennung für dieses Template.

Bezeichnung

Icon/Bezeichnung für das Template.

Dieses Template wird normalerweise automatisch von LivingApps aufgerufen, daher taucht es nicht als Link im Datenmanagement oder im Menü auf. Die Bezeichnung wird daher normalerweise nur hier in der Liste der Anzeige-Templates angezeigt.

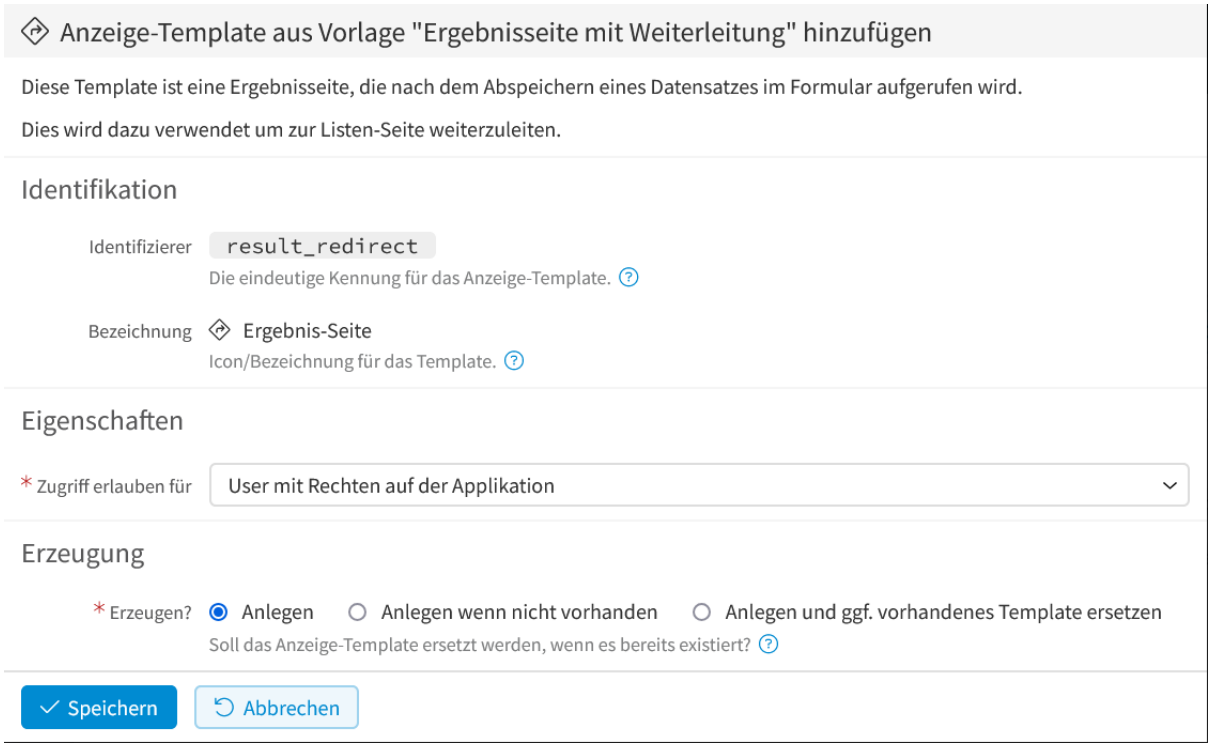

Abb. 39: Ergebnisseite mit Weiterleitung - Konfiguration

Eigenschaften

Zugriff erlauben für
Hier legen Sie fest, welche Berechtigung ein Nutzer haben muss, um auf das Template zugreifen zu können.

Erzeugung

Erzeugen?
Legen Sie hier fest, ob das Anzeige-Template ersetzt werden soll, wenn es bereits existiert.

Durch klicken auf *Speichern* ist die Erstellung des Templates abgeschlossen.

Ergebnisseite mit Datensatz-Anzeige - `result_display`
Diese Template ist eine Ergebnisseite, die nach dem Abspeichern eines Datensatzes im Formular aufgerufen wird.

Diese Seite zeigt den Datensatz an und bietet Links um den Datensatz nochmals zu verändern, oder um einen neuen Datensatz anzulegen.

Um dieses Anzeige-Template zu erstellen, wählen Sie in der Übersicht der Templatevorlagen *Ergebnisseite mit Datensatz-Anzeige* aus.

Es öffnet sich folgendes Fenster.

Identifikation

Identifizierer
Die eindeutige Kennung für dieses Template.

Bezeichnung
Icon/Bezeichnung für das Template.

Dieses Template wird normalerweise automatisch von LivingApps aufgerufen, daher taucht es nicht als Link im Datenmanagement oder im Menü auf. Die Bezeichnung wird daher normalerweise nur hier in der Liste der Anzeige-Templates angezeigt.

Eigenschaften

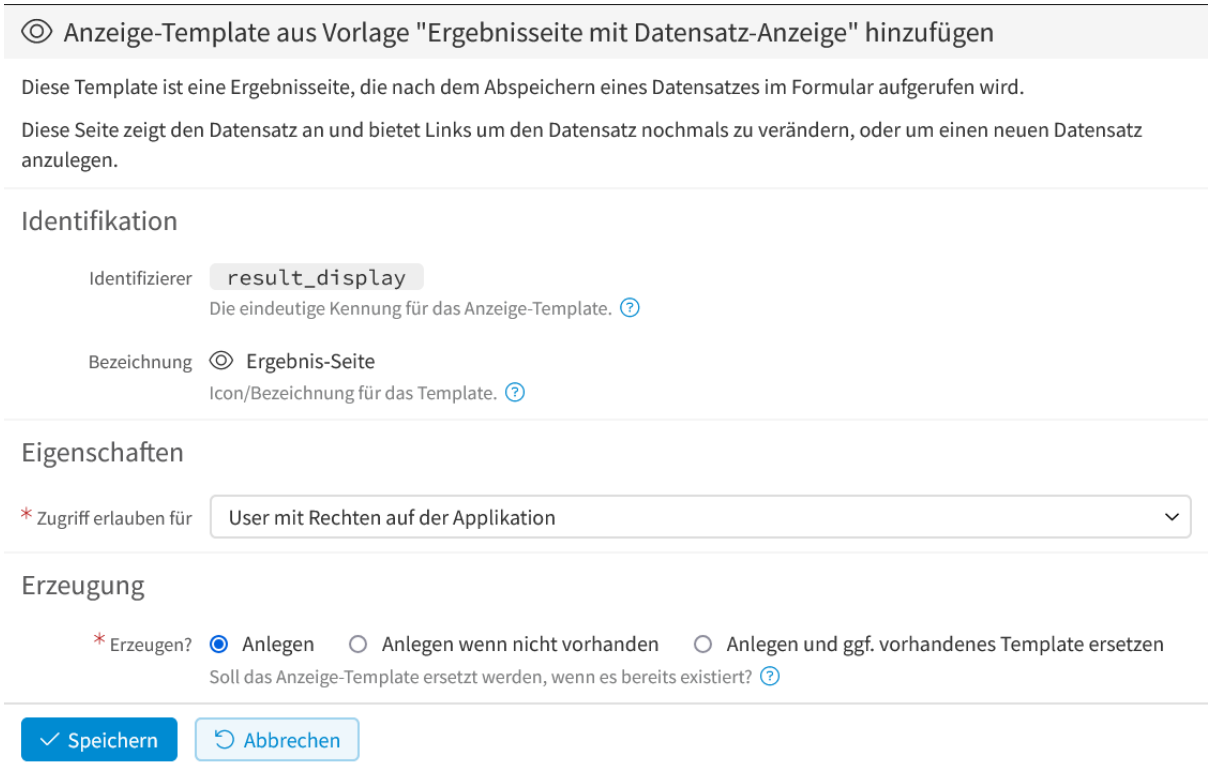

Abb. 40: Ergebnisseite mit Datensatz-Anzeige - Konfiguration

Zugriff erlauben für
Hier legen Sie fest, welche Berechtigung ein Nutzer haben muss, um auf das Template zugreifen zu können.

Erzeugung

Erzeugen?
Legen Sie hier fest, ob das Anzeige-Template ersetzt werden soll, wenn es bereits existiert.

Durch klicken auf *Speichern* ist die Erstellung des Templates abgeschlossen.

Das Ergebnis sehen Sie nach dem Speichern eines Datensatzes im Formular der LivingApp.

***Übersichtsseite* - `overview_page`**
Auf der *Übersichtsseite* haben Sie einen Überblick über alle *App-Panels* und *Benutzer-Panels*, die Sie angelegt und dort das Häkchen bei *Auf eigener Übersichtsseite* gesetzt haben.

Um eine Übersichtsseite anzulegen, wählen Sie in der Auswahl der Templatevorlagen *Übersichts-Seite* aus.

Sie haben folgende Konfigurationsmöglichkeiten:

Identifikation und Eigenschaften

Identifizierer
Die eindeutige Kennung für dieses Anzeige-Template.

Bezeichnung
Über diese Bezeichnung finden Sie das Template in der Menüzeile und in der Fußzeile der Datenliste.

Icon
Falls Sie ein Icon wählen, steht dieses links von der Bezeichnung des Templates. Durch Klick auf das ? erfahren Sie, welche Icons verwendet werden können.

Typ
Wie ein Template, je nach Typ, angesprochen werden kann, können Sie unter *Aufruf eines*

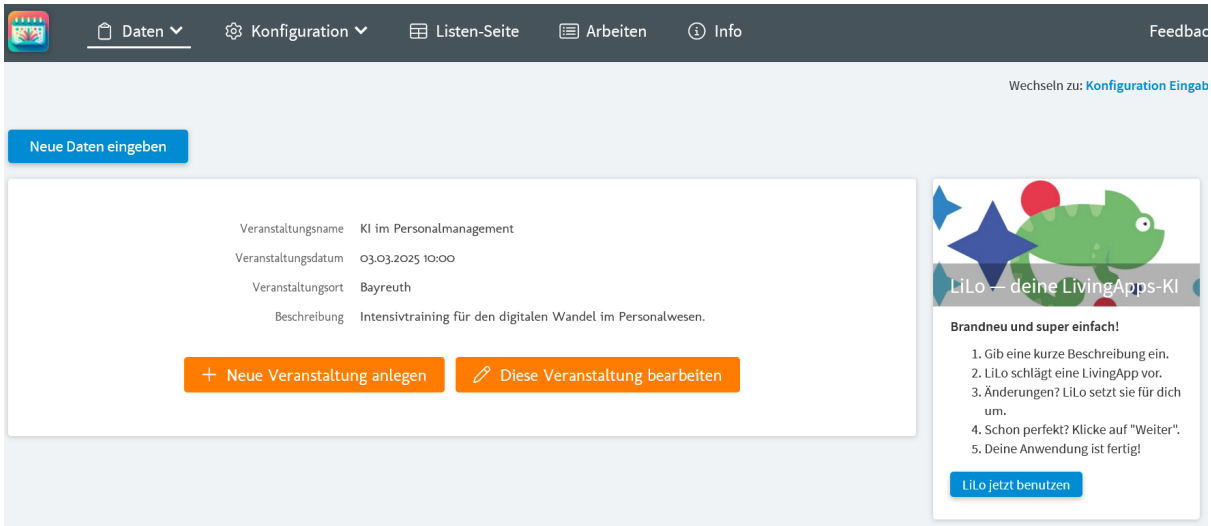

Abb. 41: Ergebnisseite mit Datensatz-Anzeige - Ergebnis

Anzeige-Templates nachlesen.

Beim Typ *Liste (mit Menüeintrag)* steht der Link zum Anzeige-Template in der *Menüzeile und in der Datenliste* (in der Fußzeile der Liste unter Listenansichten) zur Verfügung.

Berechtigung

Zugriff erlauben für
Hier legen Sie fest, welche Berechtigung ein Nutzer haben muss, um auf das Template zugreifen zu können.

Quelltext

Kommentare?
Wird hier das Häkchen gesetzt, werden im Quelltext des Templates Kommentare `<?note Kommentar?>` eingefügt, um die Orientierung zu erleichtern.

Quelltext - Bezeichnung
Die Bezeichnung des Fenstertitels (Tab im Browser)

Erzeugung

Erzeugen?
Legen Sie hier fest, ob das Anzeige-Template ersetzt werden soll, wenn es bereits existiert.

Benötigte Templates erzeugen?
Für die Übersichtsseite werden zusätzlich die Anzeigetemplates `css` und `js` benötigt. Legen Sie hier fest, wann diese erzeugt werden sollen.

Die Übersichtsseite können Sie (mit den oben festgelegten Einstellungen) wie folgt aufrufen:

Das Ergebnis sieht dann so aus:

***Kachel-Ansicht* - `tile_page`**
Mit diesem Template werden die Daten Ihrer LivingApp in *Kachelform* ausgegeben.

In der Kachel-Ansicht werden die Datensätze Ihrer LivingApp nur angezeigt, sie können nicht bearbeiten werden. In den Kacheln angezeigt werden alle Felder Ihrer LivingApp, die unter *Konfiguration* → *Liste* als Listenfelder ausgewählt sind.

Um dieses Anzeige-Template zu erstellen, klicken Sie in der *Auswahl der Template-Vorlagen* auf *Kachel-Ansicht*.

Sie haben folgende Konfigurationsmöglichkeiten:

Identifikation und Eigenschaften

Anzeige-Template aus Vorlage "Übersichts-Seite" hinzufügen

Eine Übersichts-Seite bietet eine Navigation durch alles Panels an, die Sie unter „App-Panels“ angelegt haben und wo „Auf der eigenen Übersichtsseite?“ ausgewählt wurde.

Identifikation und Eigenschaften

* Identifizierer: overview_page
Eine eindeutige Kennung für dieses Anzeige-Template. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Bezeichnung: Übersichts-Seite
Die Bezeichnung für den Link im Datenmanagement und im Menü.

* Icon: house
Das Icon für den Link im Datenmanagement.

* Typ: ○ Liste ○ Liste (Standard) ◉ Liste (mit Menüeintrag)
Wo steht das Anzeige-Template zur Verfügung, und wie kann es angesprochen werden?

Berechtigung

* Zugriff erlauben für: User mit Rechten auf der Applikation

Quelltext

Kommentare? ☑
Sollen Kommentare in den Quelltext eingefügt werden, um die Orientierung zu erleichtern?

* Bezeichnung: Übersichts-Seite
Der Seitentitel (d.h. das `title`-Element im `head`, also der Fenster-Titel)

Erzeugung

* Erzeugen? ◉ Anlegen ○ Anlegen und ggf. vorhandenes Template ersetzen
Soll das Anzeige-Template ersetzt werden, wenn es bereits existiert?

* Benötigte Templates erzeugen? ○ Nicht anlegen ◉ Anlegen wenn nicht vorhanden ○ Anlegen und ggf. vorhandenes Template ersetzen
`overview_page` verlinkt auf die Anzeige-Templates `css` und `js`. Hiermit legen Sie fest, wann diese Anzeige-Templates erzeugt werden sollen.

✓ Speichern Abbrechen

Abb. 42: Übersichtsseite konfigurieren

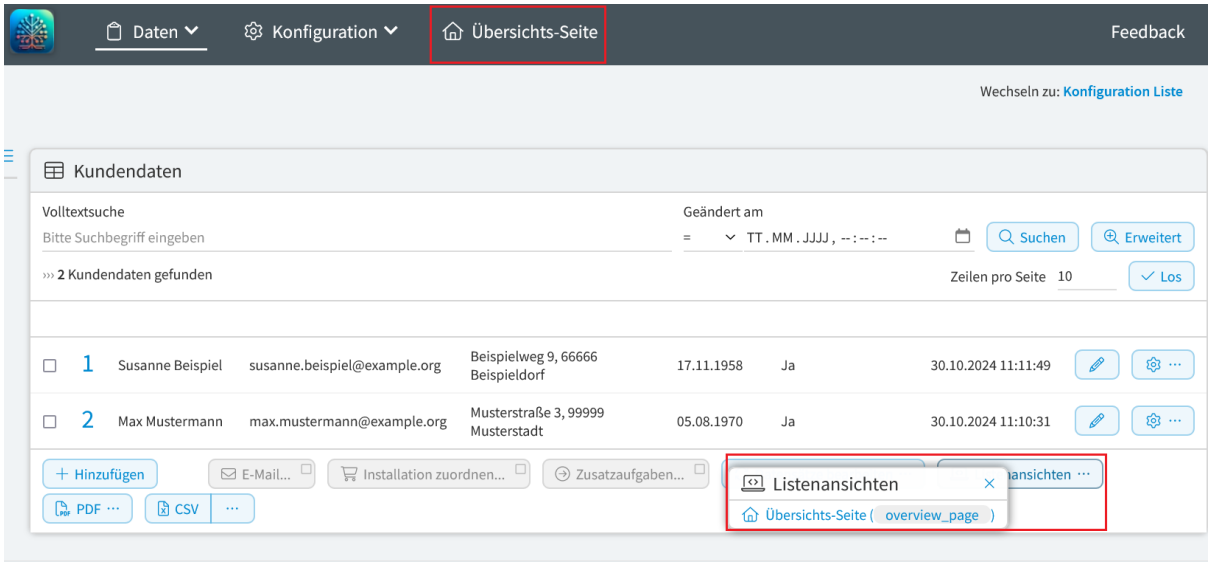

Abb. 43: Übersichtsseite aufrufen

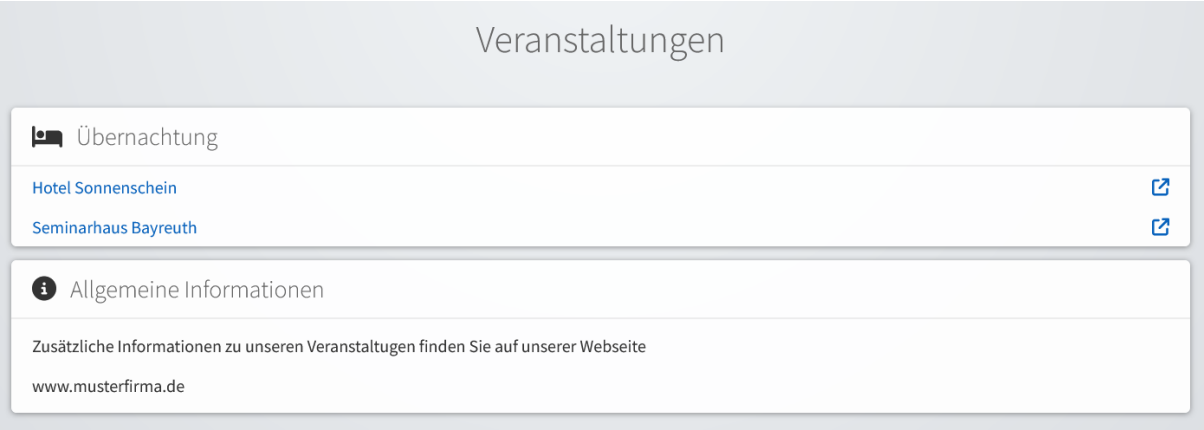

Abb. 44: Übersichtsseite Ergebnis

Anzeige-Template aus Vorlage "Kachel-Ansicht" hinzufügen

Diese Template bietet eine einfache Kachel-Ansicht der Datensätze in dieser LivingApp.

Identifikation und Eigenschaften

* Identifizierer: tile_page
Eine eindeutige Kennung für dieses Anzeige-Template. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Bezeichnung: Kachel-Ansicht
Die Bezeichnung für den Link im Datenmanagement und im Menü.

* Icon: grid
Das Icon für den Link im Datenmanagement.

* Typ: (●) Liste ( ) Liste (Standard) ( ) Liste (mit Menüeintrag)
Wo steht das Anzeige-Template zur Verfügung, und wie kann es angesprochen werden?

* Modus: (●) Standard ( ) Mit Inhalt ( ) Schleife über Felder ( ) Felder einzeln
Wie wird der Inhalt der Tabelle ausgegeben?

Kommentare? [x]
Sollen Kommentare in den Quelltext eingefügt werden, um die Orientierung zu erleichtern?

Berechtigung

* Zugriff erlauben für: User mit Rechten auf der Applikation

Erzeugung

* Erzeugen? (●) Anlegen ( ) Anlegen und ggf. vorhandenes Template ersetzen
Soll das Anzeige-Template ersetzt werden, wenn es bereits existiert?

* Benötigte Templates erzeugen? ( ) Nicht anlegen (●) Anlegen wenn nicht vorhanden ( ) Anlegen und ggf. vorhandenes Template ersetzen
`tile_page` verlinkt auf die Anzeige-Templates `css` und `js`. Hiermit legen Sie fest, wann diese Anzeige-Templates erzeugt werden sollen.

Speichern | Abbrechen

Abb. 45: Kachel-Ansicht konfigurieren

Identifizierer
Die eindeutige Kennung für dieses Anzeige-Template.

Bezeichnung
Über diese Bezeichnung finden Sie das Template in der Menüzeile und in der Fußzeile der Datenliste.

Icon
Falls Sie ein Icon wählen, steht dieses links von der Bezeichnung des Templates. Durch Klick auf das ? erfahren Sie, welche Icons verwendet werden können.

Typ
Wie ein Template, je nach Typ, angesprochen werden kann, können Sie unter *Aufruf eines Anzeige-Templates* nachlesen.

Beim Typ *Liste (mit Menüeintrag)* steht der Link zum Anzeige-Template in der *Menüzeile und in der Datenliste* (in der Fußzeile der Liste unter Listenansichten) zur Verfügung.

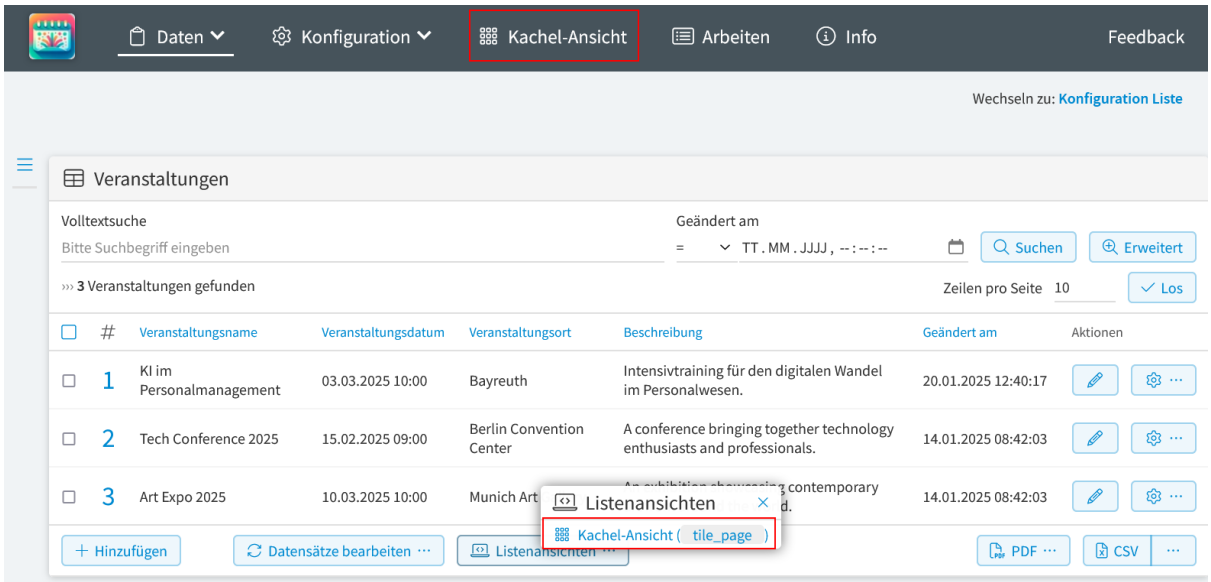

Abb. 46: Kachel-Ansicht Aufruf

Modus

Kommentare?
Wird hier das Häkchen gesetzt, werden im Quelltext des Templates Kommentare `<?note Kommentar?>` eingefügt, um die Orientierung zu erleichtern.

Berechtigung

Zugriff erlauben für
Hier legen Sie fest, welche Berechtigung ein Nutzer haben muss, um auf das Template zugreifen zu können.

Erzeugung

Erzeugen?
Legen Sie hier fest, ob das Anzeige-Template ersetzt werden soll, wenn es bereits existiert.

Benötigte Templates erzeugen?
`tile_page` verlinkt auf die Anzeigetemplates `css` und `js`. Legen Sie hier fest, wann diese erzeugt werden sollen.

Ergebnis Kachel-Ansicht:

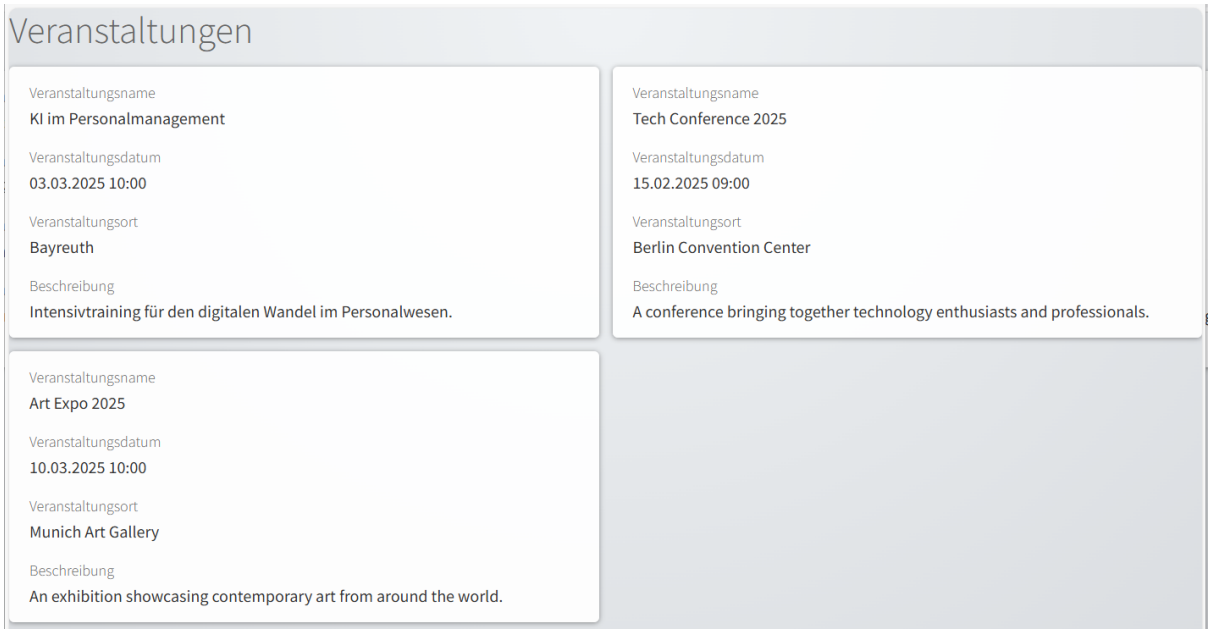

Abb. 47: Kachel-Ansicht Ergebnis

5.5.3 Einen Template-Satz für einen Anwendungsfall übernehmen

Möchten Sie die Templates nicht einzeln erzeugen, haben Sie hier die Möglichkeit, für zwei Anwendungsfälle jeweils einen kompletten Template-Satz zu übernehmen.

Um einen Template-Satz für Ihre LivingApp zu übernehmen wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Anzeige-Templates* und klicken dann auf *Anwendungsfälle* . . . .

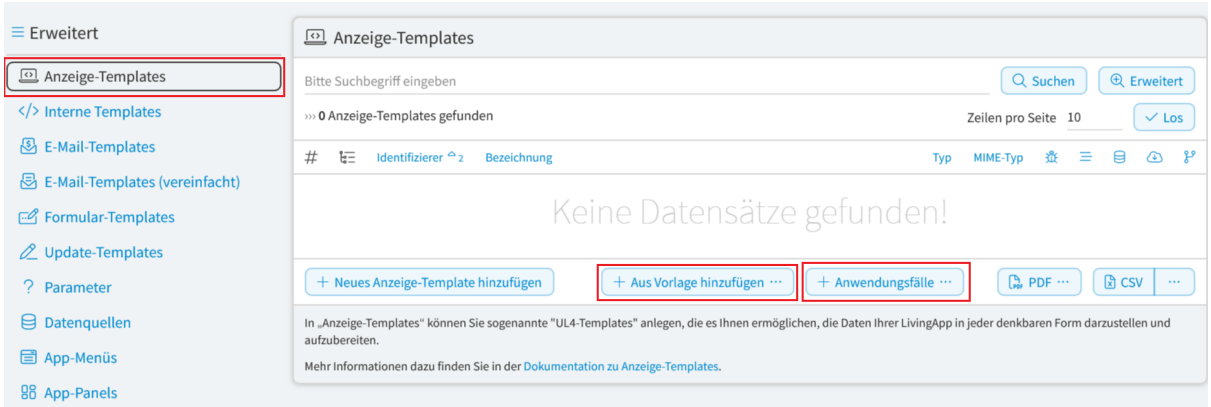

Abb. 48: Anwendungsfall aus Vorlage hinzufügen

Wählen Sie hier den Anwendungsfall aus, der Ihren Vorstellungen entspricht.

1. Anwendungsfall: *Datensatzliste nach Formular absenden* In diesem Anwendungsfall wird nach dem Speichern oder Bearbeiten des Formulars in Ihrer LivingApp die Liste der eingegebenen Datensätze angezeigt. Sie können diese Liste über die „Suchbegriff-Zeile“ durchsuchen.

   Hier wird in der Liste keinerlei Möglichkeit angeboten, die vorhandenen Datensätze zu ändern oder zu löschen. Es können nur Datensätze hinzugefügt werden.

   Um diesen Anwendungsfall zu übernehmen, klicken Sie in der Auswahl der Anwendungsfälle auf *Datensatzliste nach Formular absenden* und dann im geöffneten Fenster auf *Speichern*. Es müssen keine weiteren Konfigurationen vorgenommen werden.

   Es werden zwei Templates angelegt:

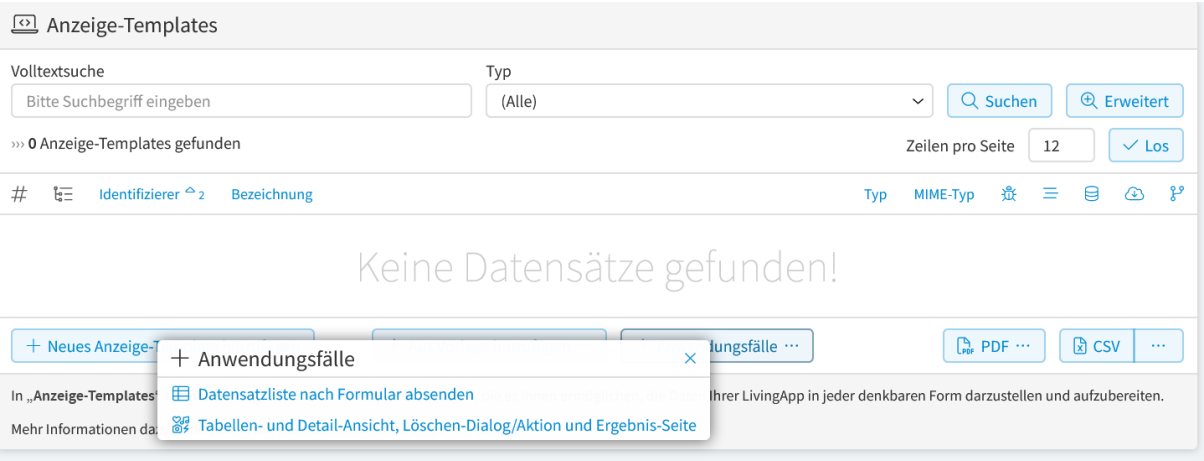

Abb. 49: Anwendungsfall auswählen

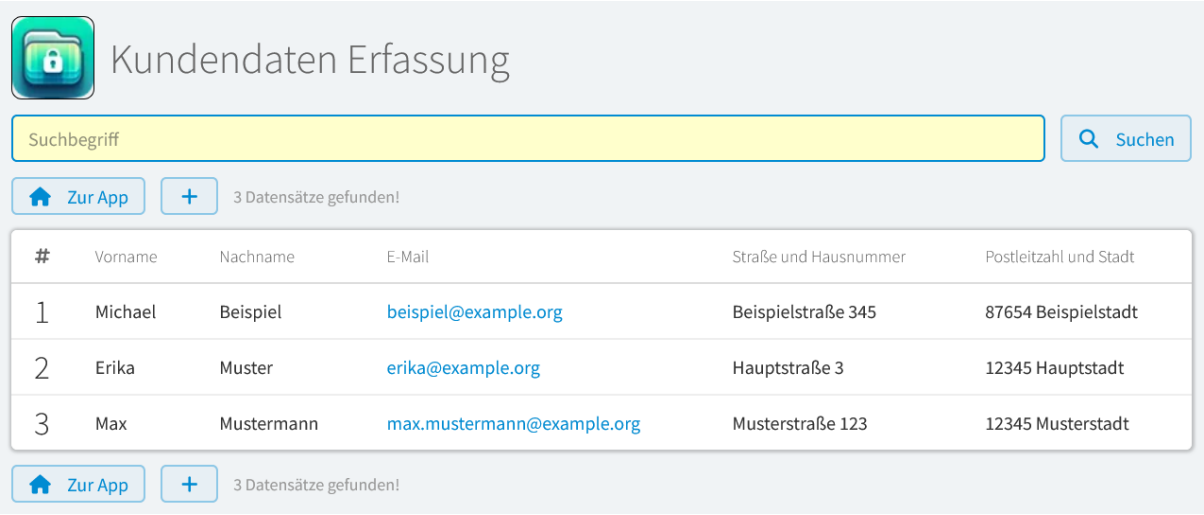

Abb. 50: Anwendungsfall „Datensatzliste“ - Ergebnis

1. Die *Listen-Seite* (`list_page`).
2. Die *Ergebnisseite mit Weiterleitung* (`result_direct`), die nach dem Speichern des Eingabe-Formulars angezeigt wird. Diese Seite leitet zur Listen-Seite weiter.

2. Anwendungsfall: *Tabellen- und Detail-Ansicht, Löschen-Dialog/Aktion* In diesem Anwendungsfall werden verschiedene Anzeige-Templates angelegt, die gemeinsam eine komplette Anwendung darstellen.

   Die Datensätze Ihrer LivingApp werden als Liste ausgegeben. Sie können diese Liste über die „Suchbegriff-Zeile“ durchsuchen. Sie haben hier zusätzlich die Möglichkeit, einzelne Datensätze anzuzeigen, zu bearbeiten und zu löschen.

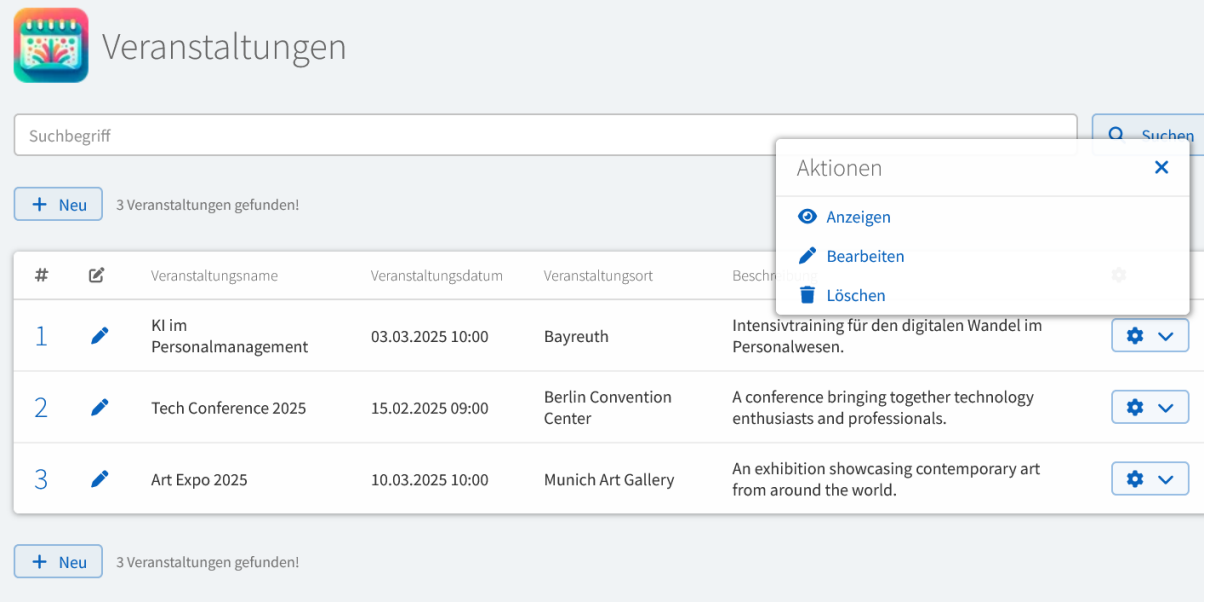

Abb. 51: Anwendungsfall „Liste/Detail/Löschen-Dialog“ - Ergebnis

Außerdem können Sie untergeordnete Datensätze aus allen LivingApps, die per `applookup`-Felder auf diese LivingApp verweisen, anzeigen lassen. Dafür muss bei Erstellung des Anwendungsfalles bei *Detail Panels?* das Häkchen gesetzt werden.

Zur Ansicht der untergeordneten Datensätze gelangen Sie, wenn Sie in der Liste im Datensatz-Aktions-Menü bei einem Datensatz, der untergeordnete Datensätze enthält, auf *Anzeigen* klicken.

Um diesen Anwendungsfall zu übernehmen, klicken Sie in der Auswahl der Anwendungsfälle auf *Tabellen- und Detail-Ansicht, Löschen-Dialog/Aktion*

Es öffnet sich folgendes Fenster.

Es können folgende Konfigurationen vorgenommen werden:

Typ
: Standardmäßig ist hier *Liste (mit Menüeintrag)* vorausgewählt. Mehr zum *Typ eines Templates* können Sie oben in der Dokumentation der einzelnen Template-Vorlagen nachlesen.

Detail-Panels?
: Setzen Sie hier das Häkchen, wenn die untergeordnete Datensätze, aus allen LivingApps die per `applookup`-Felder auf diese LivingApp verweisen, angezeigt werden sollen.

Zugriff erlauben für
: Hier legen Sie fest, welche Berechtigung ein Nutzer haben muss, um auf das Template zugreifen zu können.

Erzeugen?
: Legen Sie hier fest, ob das Anzeige-Template ersetzt werden soll, wenn es bereits existiert.

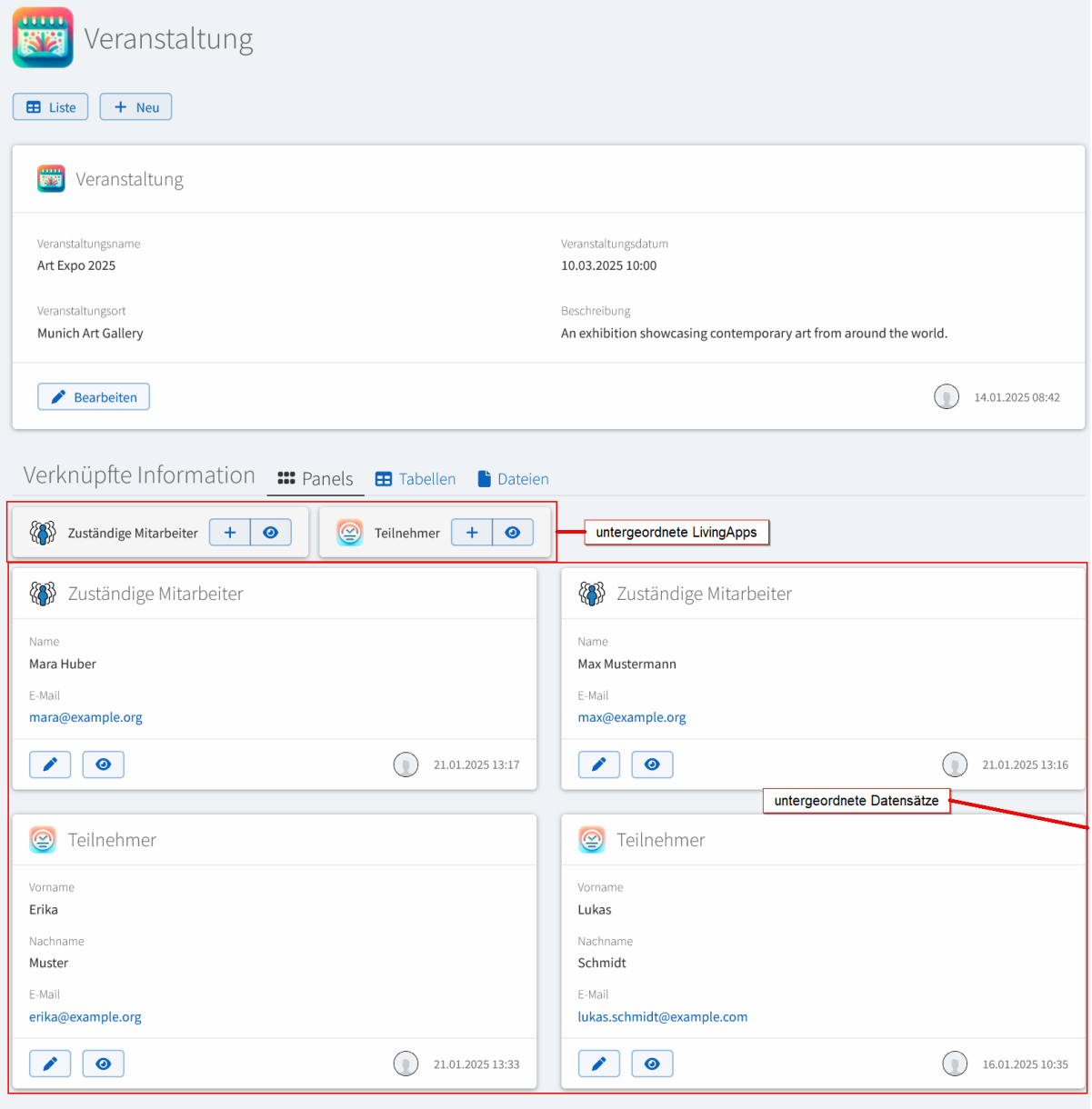

Abb. 52: Anwendungsfall „Liste/Detail/Löschen-Dialog“ - Ergebnis Kachel-Ansicht der untergeordneten Datensätze

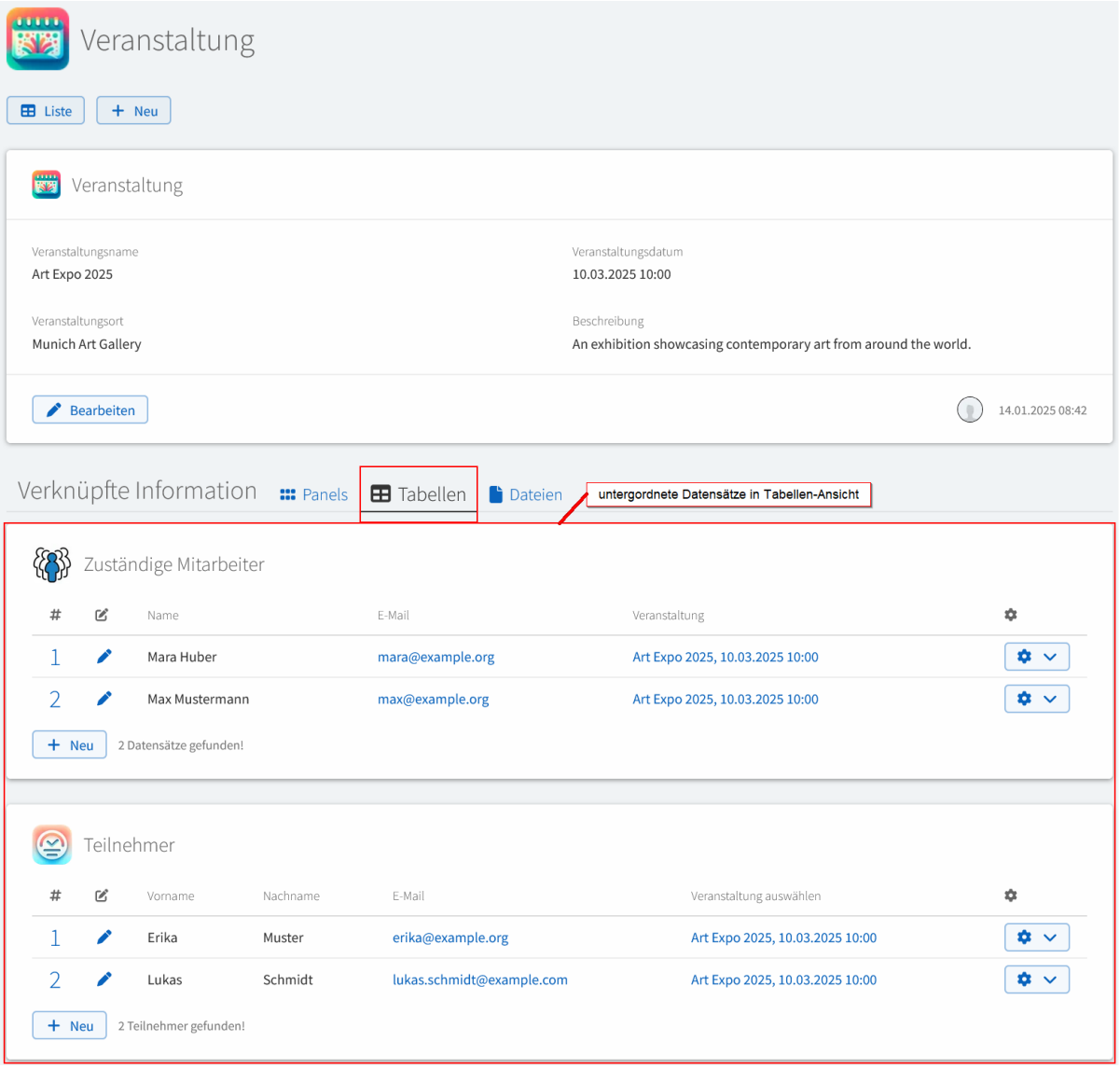

Abb. 53: Anwendungsfall „Liste/Detail/Löschen-Dialog“ - Ergebnis Listen-Ansicht der untergeordneten Datensätze

Satz von Anzeige-Templates für Tabellen/Detail-Ansicht hinzufügen

Es werden folgende Anzeige-Templates angelegt, die gemeinsam eine komplette Anwendung darstellen:

Liste der Anzeige-Templates >

Listen-Seite

* Typ ○ Liste ○ Liste (Standard) ◉ Liste (mit Menüeintrag)
Wo steht das Anzeige-Template zur Verfügung, und wie kann es angesprochen werden? ⓘ

Detail-Ansicht

Detail-Panels? ☐
Sollen untergeordnete Datensätze angezeigt werden aus allen LivingApps die per `applookup`-Felder auf diese LivingApp verweisen?

Berechtigung

* Zugriff erlauben für User mit Rechten auf der Applikation

Erzeugung

* Erzeugen? ◉ Anlegen ○ Anlegen wenn nicht vorhanden ○ Anlegen und ggf. vorhandenes Template ersetzen
Sollen die Anzeige-Templates ersetzt werden, wenn sie bereits existieren?

* Benötigte Templates erzeugen? ○ Nicht anlegen ◉ Anlegen wenn nicht vorhanden ○ Anlegen und ggf. vorhandenes Template ersetzen
Die erzeugten Anzeige-Templates verlinken auf die Anzeige-Templates `css` und `js`. Hiermit legen Sie fest, wann diese Anzeige-Templates erzeugt werden sollen.

✓ Speichern ↺ Abbrechen

Abb. 54: Anwendungsfall „Liste/Detail/Löschen-Dialog“ - Konfiguration

Benötigte Templates erzeugen?
Der erzeugten Anzeige-Templates verlinken auf die Anzeigetemplates `css` und `js`. Legen Sie hier fest, wann diese erzeugt werden sollen.

Mit klicken auf *Speichern* ist die Erstellung des Anwendungsfalles abgeschlossen. Im Hintergrund werden automatisch folgende Templates angelegt:

Satz von Anzeige-Templates für Tabellen/Detail-Ansicht hinzufügen

Es werden folgende Anzeige-Templates angelegt, die gemeinsam eine komplette Anwendung darstellen:

Liste der Anzeige-Templates

Listen-Seite (`list_page`)
Listen-Seite mit allen Datensätzen dieser LivingApp.

Detail-Anzeige-Seite (`detail_page`)
Eine Detail-Anzeige-Seite die einen einzelnen Datensatz sowie untergeordnetet Datensätze anzeigt.

Detail-Bearbeiten-Seite (`detail_edit_page`)
Eine Detail-Bearbeiten-Seite für einen einzelnen Datensatz.

Detail-Anlegen-Seite (`detail_new_page`)
Eine Seite zum Anlegen neuer Datensätze.

Löschen-Dialog/-Aktion (`record_delete`)
Dialog zur Bestätigung der Löschung eines Datensatzes und Lösch-Aktion. Dieses Anzeige-Template ist im Datensatzmenü in der Tabellen-Ansicht verlinkt. Nach dem Löschen wird zur vorherigen Seite weitergeleitet.

Wird unter „**Detail-Ansicht**“ „**Detail-Panels?**“ ausgewählt, werden auch noch folgende Anzeige-Templates erzeugt:

Unter-Datensätze als Panels (`detail_children_panels`)
Unter-Datensätze eines Datensatzes werden als Panels angezeigen. Außerdem gibt es eine Navigationsmöglichkeiten zwischen verschiedenen Anzeige-Varianten.

Unter-Datensätze als Tabelle (`detail_children_tables`)
Unter-Datensätze eines Datensatzes werden in Tabellen angezeigt. Außerdem gibt es eine Navigationsmöglichkeiten zwischen verschiedenen Anzeige-Varianten.

Unter-Datensätze als Thumbnails (`detail_children_thumbnail`)
Upload-Felder von Unter-Datensätzen eines Datensatzes werden als Thumbnails angezeigt. Außerdem gibt es eine Navigationsmöglichkeiten zwischen verschiedenen Anzeige-Varianten.

Desweiteren werden, wenn unter „**Detail-Ansicht**“ „**Detail-Panels?**“ ausgewählt wurde, in den LivingApps der untergeordneten Datensätze folgende Anzeige-Templates erzeugt:

Detail-Anzeige-Kachel für Kachel-Ansicht (`detail_display_panel`)
Eine Detail-Anzeige-Kachel, die für die Anzeige von untergeordneten Datensätzen verwendet wird (und per Ajax nachgeladen wird, wenn das Bearbeiten eines Datensatzes abgebrochen wird).

Detail-Bearbeiten-Kachel für Kachel-Ansicht (`detail_edit_panel`)
Eine Detail-Kachel zum Bearbeiten eines Datensatzes; Diese wird per Ajax nachgeladen, wenn der Benutzer in den untergeordneten Datensätzen auf „**Bearbeiten**“ klickt.

Detail-Anlegen-Kachel für Kachel-Ansicht (`detail_new_panel`)
Eine Detail-Kachel zum Anlegen neuer Datensätze; Diese wird per Ajax nachgeladen, wenn der Benutzer in den untergeordneten Datensätzen auf „**Neu**“ klickt.

Detail-Bearbeiten-Kachel für Tabellen-Ansicht (`detail_edit_panel_tables`)
Eine Detail-Kachel zum Bearbeiten eines Datensatzes; Diese wird per Ajax nachgeladen, wenn der Benutzer in den untergeordneten Datensätzen in der Tabellen-Ansicht auf „**Bearbeiten**“ klickt.

Detail-Anlegen-Kachel in der Tabellen-Ansicht (`detail_new_panel_tables`)
Eine Detail-Kachel zum Anlegen neuer Datensätze; Diese wird per Ajax nachgeladen, wenn der Benutzer in den untergeordneten Datensätzen in der Tabellen-Ansicht auf „**Neu**“ klickt.

Abb. 55: Anwendungsfall „Liste/Detail/Löschen-Dialog“ - Templates

KAPITEL 6

Interne Templates

In *Interne Templates* können Sie sich eine Bibliothek von „UL4-Templates“ anlegen. Diese Templates können Sie dann in Ihren E-Mail- oder Anzeige-Templates aufrufen.

Ein internes Template kann nur von anderen Templates aufgerufen werden, nicht von außen. Z. B. für eine Funktion, die sich um die Anzeige eines Feldes kümmert und mit den entsprechenden Parametern aufgerufen wird.

6.1 Erstellung eines Internen Templates

Um ein Internes Template für Ihre LivingApp zu erstellen, wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Interne Templates* und dann auf *Hinzufügen*.

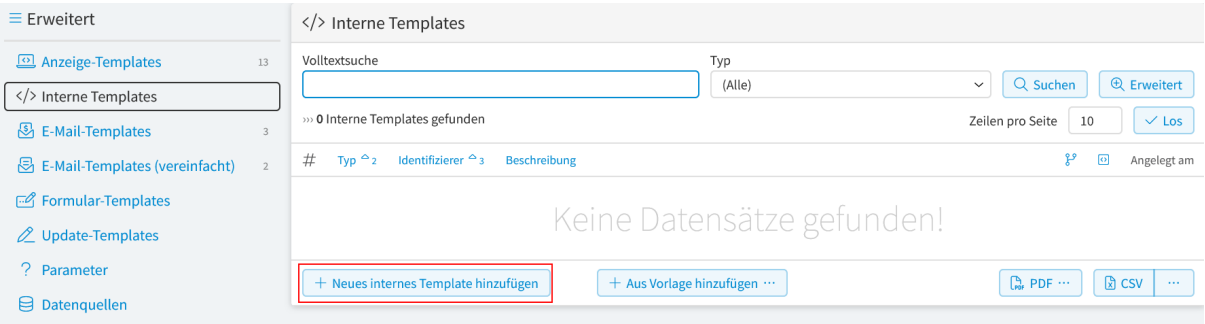

Abb. 1: Internes Template hinzufügen

Im nun geöffneten Fenster sehen Sie die *Eingabeansicht* des Internen Templates.

Typ
: Wählen Sie hier den LivingAPI-Typ aus, dem dieses Template als Methode zugeordnet sein soll.

Identifizierer
: Der Identifizierer ist die eindeutige Kennung für das Template. Er kann verwendet werden, um das Template aufzurufen. Der Identifizierer darf nur Buchstaben, Ziffern und _ enthalten.

Aus Library-Template
: Möchten Sie den Quelltext aus einem Library-Template übernehmen, wählen Sie das entsprechende Template hier aus.

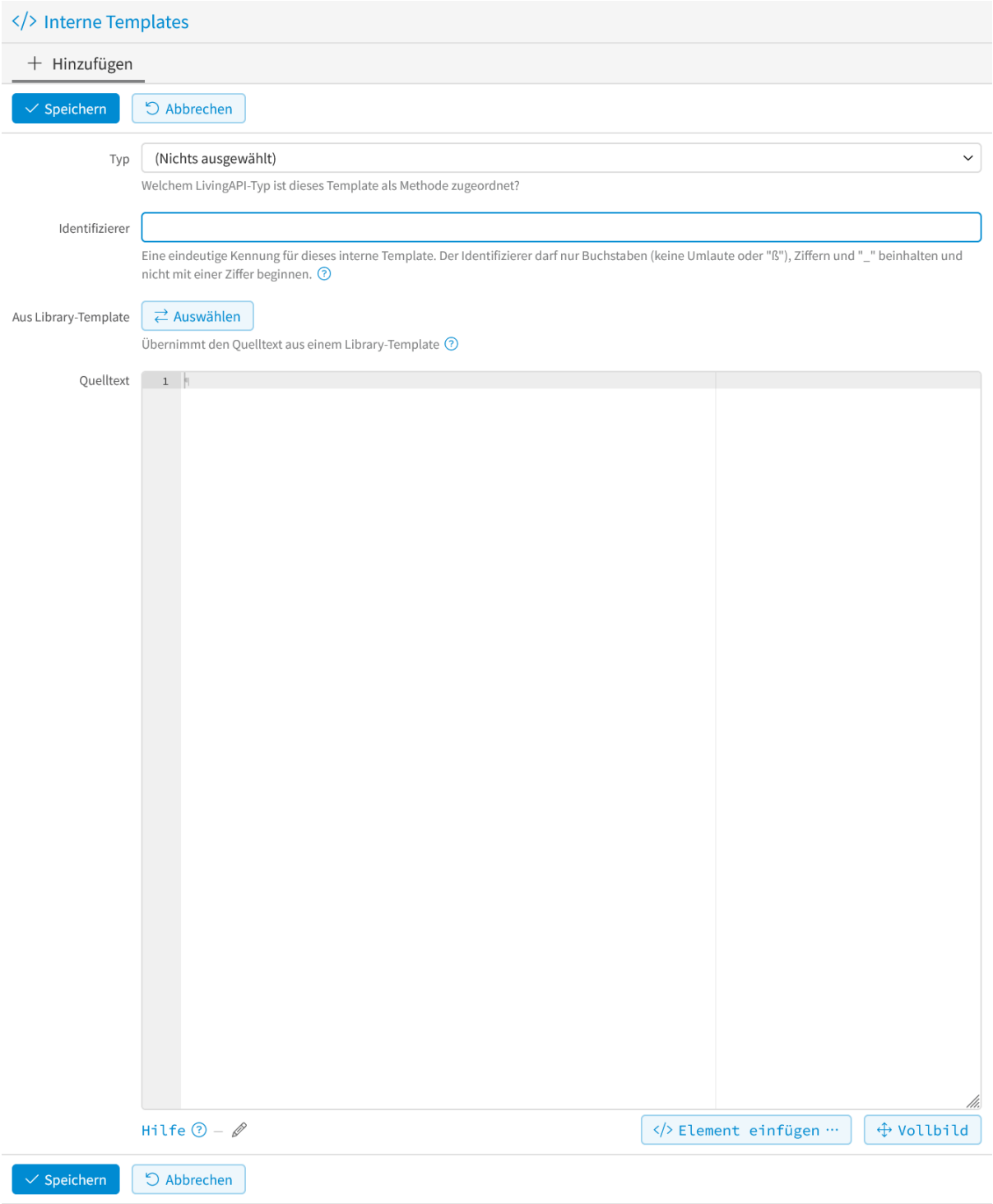

Abb. 2: Internes Template - Eingabeansicht

Wenn Sie den Quelltext aus einem Library-Template übernehmen (und dabei den Original-Identifizierer und -Typ übernehmen), überschreiben Sie damit die Funktionalität des Library-Templates und können ihre eigene Anzeigelogik implementieren.

Quelltext
Hier erfolgt die *Formulierung des Templates*.

Wird im Quelltext ein `<?ul4>`-Tag, ein `<?whitespace?>`-Tag oder ein `<?doc?>`-Tag eingegeben, dann erscheint unter *Templateeigenschaften* folgendes:

Signatur
Die Liste der Parameter, die dieses Template erwartet (wird automatisch aus dem `<?ul4?>`-Tag im Template-Quelltext extrahiert). Für interne Templates sollte eine Signatur angegeben werden, damit das Template aufgerufen werden kann, ohne die Parameter per „Keyword“ übergeben zu müssen. D.h. wenn z.B. eine Signatur `<?ul4 berechnung(wert)?>` für ein Template namens `berechnung` verwendet wird, kann dieses mittels `globals.t_berechnung(42)` aufgerufen werden, statt `globals.t_berechnung(wert=42)` verwenden zu müssen.

Whitespace
Whitespace konfiguriert, wie Einrückungen/Zeilenvorschübe im Template-Quelltext behandelt werden (wird automatisch aus dem `<?whitespace?>`-Tag im Template-Quelltext extrahiert).

Beschreibung
Beschreibung des Templates (wird automatisch aus dem `<?doc?>`-Tag im Template-Quelltext extrahiert). Z. B. `<?doc Teilnehmerliste?>` oder `<?doc?>Teilnehmerliste<?end doc?>`.

Aufgerufen werden kann das Interne Template mit dem Identifizierer `beispiel` mit:

```
<?render globals.templates.beispiel(argumente)?>
```

Shortcut:

```
<?render globals.t_beispiel(argumente)?>
```

Haben Sie z. B. eine *Datenquelle* mit dem Identifizierer `muster` angelegt und in dieser nicht die aktuelle App sondern eine andere ausgewählt, können Sie auf die internen Templates dieser App zugreifen mit:

```
<?render datasources.muster.app.templates.beispiel(argumente)?>
```

Shortcut:

```
<?render datasources.muster.app.t_beispiel(argumente)?>
```

Wobei `beispiel` jeweils für den Identifizierer des internen Templates steht.

6.2 Anwendungsbeispiel

Im Anwendungsbeispiel soll ein Internes Template angelegt werden, das die Font-Awesome-Icons einbindet. Diese Icons können Sie dann in Ihren Anzeige-Templates verwenden.

Um eine neues Internes Template zu erstellen, wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Interne Templates*.

Klicken Sie auf *Neues internes Template hinzufügen* um ein Internes Template anzulegen.

Im nun geöffneten Fenster werden folgende Konfigurationen vorgenommen.

Über den *Identifizierer* kann das Interne Template z. B. in Ihrem Anzeigetemplate aufgerufen werden mit:

```
<?render globals.templates.static_font_awesome()?>
```

Shortcut:

Interne Templates

1/2

+ Hinzufügen

Speichern | Abbrechen | Hilfe

* Identifizierer static_font_awesome

Eine eindeutige Kennung für das Template

Quelltext

```
1  <?ul4 static_font_awesome(version="5.11.2-pro")?>
2  <?whitespace strip?>
3  <?doc
4
5  ——Bindet die Font Awesome Icons ein.
6
7  ?>
8  <?if version not in {"4.7.0", "5.0.9", "5.0.9-pro", "5.0.10", "5.0.10-pro", "5.1.0", "5.1.0-pro",
     "5.7.0", "5.7.0-pro", "5.11.2-pro"}?>
9  ——<?code version = "4.7.0"?>
10 <?end if?>
11 <?if version == "4.7.0"?>
12 ——<link href="/static/font-awesome/<?printx version?>/css/font-awesome.min.css" rel="stylesheet"
       />
13 <?elif version in {"5.0.9", "5.0.9-pro", "5.0.10", "5.0.10-pro"}?>
14 ——<link href="/static/font-awesome/<?printx version?>/web-fonts-with-css/css/fontawesome-all.min
       .css" rel="stylesheet" />
15 <?else?>
16 ——<link href="/static/font-awesome/<?printx version?>/css/all.css" rel="stylesheet" />
17 <?end if?>
```

Hilfe — Vollbild

Historie

Angelegt von	Geändert von
Angelegt am	Geändert am

Speichern | Abbrechen | Hilfe

Abb. 3: Internes Template Konfiguration

```
<?render globals.t_static_font_awesome()?>
```

In *Quelltext* erfolgt die *Formulierung des Templates*.

KAPITEL 7

E-Mail-Templates

Mit den E-Mail-Templates können Vorlagen für E-Mail-Benachrichtigungen erstellt werden, die in bestimmten Situationen verschickt werden. Zum Beispiel kann automatisch eine E-Mail nach dem Erstellen eines Datensatzes (d. h. nach dem Absenden des Formulars) oder nach dem Ändern eines Datensatzes versendet werden. Zudem gibt es die Möglichkeit, E-Mail-Vorlagen zu erstellen, die dann manuell in der Datenliste versendet (Voraussetzung dafür ist ein E-Mail-Feld im Formular der App) oder in der *Aktion E-Mail-Versendung* eingebunden werden können.

Sie können festlegen, an welchen Empfänger die E-Mail verschickt werden soll und wer eine Kopie der E-Mail erhält. So können beispielsweise einzelne Mitarbeiter über einen neuen Datensatz informiert und auf anfallende Aufgaben hingewiesen werden. Die E-Mail-Funktion kann auch genutzt werden, um dem Absender des Formulars eine Empfangsbestätigung zu schicken, in der z. B. auch seine angegebenen Daten nochmals aufgelistet werden.

Innerhalb der E-Mail kann auf die einzelnen Felder des jeweiligen Datensatzes und wenn Sie Datenquellen anlegen, auch auf die Felder anderer Apps zugegriffen werden. So können wichtige Informationen in den Inhalt der Benachrichtigung integriert werden und der Empfänger erhält in der E-Mail einen Überblick über die neuen oder geänderten Daten.

7.1 Erstellung eines E-Mail-Templates

Um ein neues E-Mail-Template zu erstellen, wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken anschließend in der linken Menüleiste auf *E-Mail-Templates* und dann auf *Neues E-Mail-Template hinzufügen*.

Im nun geöffneten Fenster sehen Sie die *Eingabeansicht* des E-Mail-Templates.

Identifizierer
: Der Identifizierer ist eine eindeutige Kennung für das Template und darf nur Buchstaben, Zahlen und _ enthalten.

Bezeichnung
: Hier können Sie eine Beschreibung des E-Mail-Templates eingeben, um eine bessere Übersicht über mehrere Templates zu haben. Unter dieser Bezeichnung finden Sie das E-Mail-Template, wenn Sie es aus der Datenliste manuell verschicken wollen.

Aktiv
: Ist hier der Haken gesetzt, ist das E-Mail-Template sofort nach dem Anlegen aktiv. Wenn Sie noch Änderungen am E-Mail-Template vornehmen wollen, ist es sinnvoll das E-Mail-Template zuerst inaktiv anzulegen, es dann anzupassen, und erst danach zu aktivieren.

Verwendung
: Hier können Sie festlegen, wann eine E-Mail verschickt werden soll.

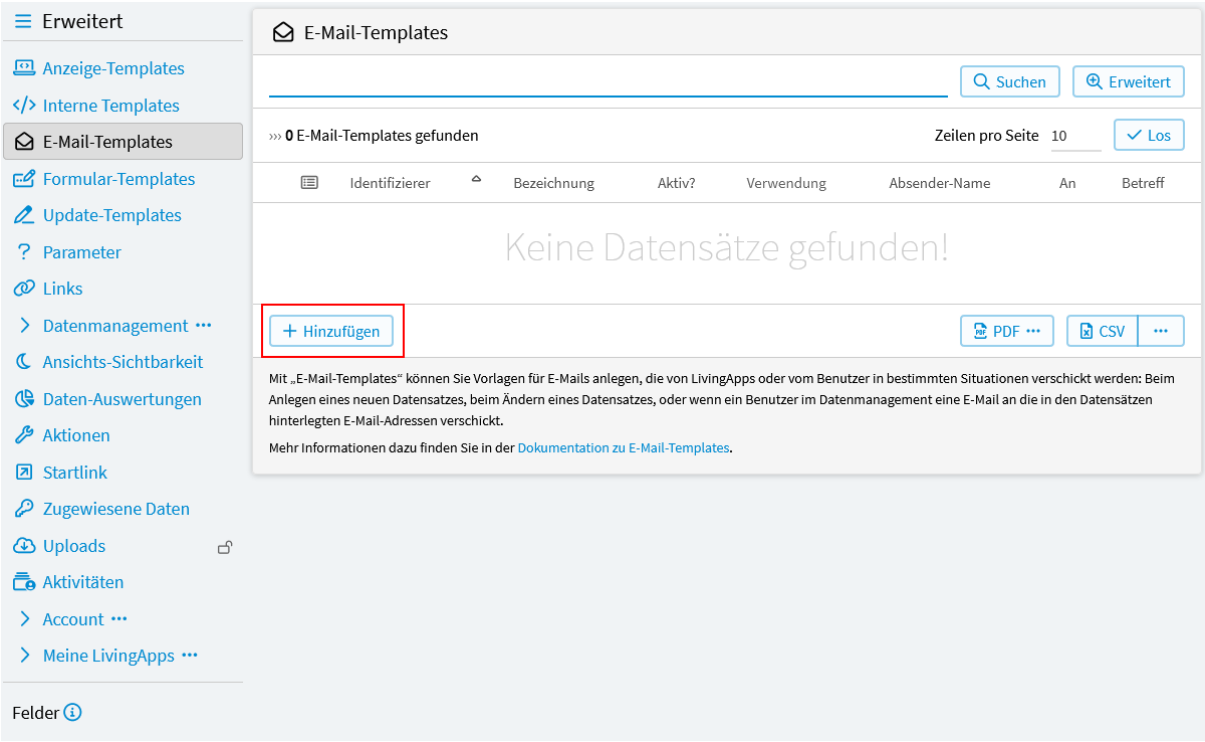

Abb. 1: E-Mail-Template hinzufügen

Bei neuem Datensatz
Versendet eine Benachrichtigung, wenn ein neuer Datensatz angelegt wurde.

Beim Bearbeiten eines Datensatzes
Versendet eine Benachrichtung, wenn ein bestehender Datensatz verändert wurde.

Vorlage für Nachricht an alle Datensätze
Hier können E-Mail-Vorlagen erstellt werden, die dann aus dem Datenmanagement manuell, oder über eine Aktion *(E-Mail-Versendung)* verschickt werden.

Mailserver
Haben Sie einen eigenen SMTP-Server installiert, können Sie diesen hier auswählen. Wenn Sie hier nichts auswählen, erfolgt der Versand über den internen SMTP-Server von LivingApps.

E-Mail-Header:

Absender-Name
Geben Sie hier den Namen ein, der dem Empfänger der E-Mail als Absender angezeigt werden soll.

Bei Versendung über den LivingApps-SMTP-Server wird immer die E-Mail-Adresse `benachrichtigung@living-apps.de` verwendet.

Bei Versendung über einen externen SMTP-Server wird die beim SMTP-Server hinterlegte E-Mail-Adresse verwendet.

An (E-Mail-Feld)
Empfänger der E-Mail. Gibt es im Formular Ihrer App ein oder mehrere Felder vom Typ „E-Mail“, können Sie hier das entsprechende auswählen. Dieses E-Mail-Feld sollte im Formular ein Pflichtfeld sein, sonst kann an die Personen, die keine E-Mail-Adresse angeben, z. B. keine Anmeldebestätigung verschickt werden. Möchten Sie feste Adressaten eingeben, wählen Sie hier *(Nichts ausgewählt)* und geben die E-Mail-Adresse(n) bei *An* ein.

An
Haben Sie bei *An (E-Mail-Feld) (Nichts ausgewählt)* gewählt oder gibt es im Formular Ihrer App kein E-Mail-Feld, können Sie hier feste Adressaten eingeben. Soll die E-Mail an mehrere Empfänger versendet werden, listen Sie hier deren E-Mail-Adressen kommagetrennt auf.

E-Mail-Templates Aktiv? Nein 1/6

+ Hinzufügen

Speichern Abbrechen

* Identifizierer

Eine eindeutige Kennung für dieses E-Mail-Template. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Bezeichnung

Aktiv? ☐

Verwendung ☐ Bei neuem Datensatz
☐ Beim Bearbeiten eines Datensatzes
☐ Vorlage für Nachricht an alle Datensätze

Mailserver

Mailserver (Nichts ausgewählt)

E-Mail-Adressen

* Absender-Name

Der Name des Absenders. Die E-Mail-Adresse wird automatisch von LivingApps ergänzt.

An (E-Mail-Feld) E-Mail-Adresse

Hier wählen Sie, welches E-Mail-Feld ihrer App verwendet werden soll, um den Adressaten der E-Mail festzulegen. Wenn Sie hier nichts auswählen, können Sie einen festen Adressaten bei „An" eingeben.

Kopie

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` sowie mehrere vollständige E-Mail-Adressen (mit Komma getrennt) werden unterstützt.

Blindkopie

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` sowie mehrere vollständige E-Mail-Adressen (mit Komma getrennt) werden unterstützt.

* Antworten an

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` wird unterstützt.

E-Mail-Betreff

* Betreff

Text-E-Mail

Template

Hilfe – Element einfügen ··· Vollbild

Verfügbare Variablen: `app`, `record`, `datasources`. Verfügbare globale Variablen: `globals`, `la`

HTML-E-Mail

Template

Hilfe – Element einfügen ··· Vollbild

Verfügbare Variablen: `app`, `record`, `datasources`. Verfügbare globale Variablen: `globals`, `la`

Datei-Anhang

Datei-Anhang Durchsuchen... Keine Datei ausgewählt.

Speichern Abbrechen

Abb. 2: E-Mail-Template Eingabeansicht

Kopie
Hier können Sie weitere Empfänger angeben. Wenn Sie mehr als eine E-Mail-Adresse eingeben wollen, dann trennen Sie diese durch Kommas.

Blindkopie
An die hier aufgelisteten, kommagetrennten Empfänger wird eine Blindkopie der E-Mail versendet.

Antworten an
Der Empfänger der Benachrichtigung kann auf die E-Mail antworten. Hier legen Sie fest, an welche E-Mail-Adresse die Antwort verschickt werden soll.

Betreff
Schreiben Sie hier den Betreff Ihrer E-Mail.

7.1.1 E-Mail-Header - letzte Änderungsoptionen

Sie haben die Möglichkeit, im Quelltext des E-Mail-Templates den Inhalt der E-Mail-Header noch ein letztes Mal zu verändern. Die hier gemachten Angaben haben Vorrang vor den oben gemachten Einträgen. Durch eigene programmierte Bedingungen können Sie z. B. Einfluss darauf nehmen, wann, ob und an wen eine E-Mail verschickt wird.

Beispiele:

Möchten Sie z. B. nicht, dass als Absender der E-Mail „Max Mustermann <benachrichtigung@living-apps.de>“ steht oder möchten Sie einen eigenen SMTP-Server verwenden, schreiben Sie folgendes in den Quelltext Ihres E-Mail-Templates:

```
<?code globals.response.from = "Max Mustermann <mm@beispiel.de>"?>
```

Soll die E-Mail z. B. an unterschiedliche E-Mail-Adressen eines Empfängers gehen, geben Sie folgendes ein:

```
<?code to = record.v_e_mail?>
<?if record.v_email2?>
   <?code to += "," + record.v_email2?>
<?end if?>
<?if record.v_email3?>
   <?code to += "," + record.v_email3?>
<?end if?>
<?code globals.response.to = to?>
```

Wobei `e_mail`, `email2` und `email3` die jeweiligen Identifizierer der E-Mail-Felder sind. An dieser Stelle hätten Sie auch die Möglichkeit, eine Versendung der E-Mail noch zu stoppen, wenn bestimmte eigene programmierte Bedingungen nicht erfüllt wären.

Sind die E-Mail-Adressen der Empfänger nicht direkt in der App verfügbar, sondern stehen im Verteiler einer verknüpften App, können Sie darauf zugreifen mit:

```
<?code globals.response.to = record.v_verteiler.v_email?>
```

Wobei `verteiler` und `email` die Identifizierer der jeweiligen Felder sind.

Eine Kopie der E-Mail soll nur dann verschickt werden, wenn bei *Wird Übernachtung benötigt? Ja* gewählt wurde:

```
<?if record.v_wird_uebernachtung_benoetigt.key == "ja"?>
   <?code globals.response.cc = "zimmerbuchung@livinglogic.de"?>
<?else?>
   <?code globals.response.cc = None?>
<?end if?>
```

Dasselbe gilt auch für die Blindkopie (bcc).

Auch was in *Antworten an* stehen soll, kann hier an Bedingungen geknüpft werden:

```
<?if record.v_veranstaltung2.v_veranstaltung == "powerpoint"?>
   <?code globals.response.replyto = "beispiel@livinglogic.de"?>
<?else?>
   <?code globals.response.replyto = record.v_zustaendiger_mitarbeiter?>
<?end if?>
```

Wenn Sie keinen festen Betreff eingeben, sondern hier auf die Daten Ihrer App zugreifen möchten, könnten Sie das folgendermaßen tun:

```
<?code globals.response.subject = record.v_betreff?>
```

Wobei `betreff` für den Identifizierer des Feldes steht.

Mithilfe folgender Variablen können Sie z. B. ein E-Mail-Rahmentemplate erstellen, das nur die Anrede und die Signatur enthält. Der Text wird dann über die Varianblen eingefügt.

Mit `<?print globals.request.bodyhtml?>` wird als E-Mail-Text ausgegeben, was in der Daten-Liste unter *E-Mail versenden* bei *Nachricht* eingegeben wird oder bei einer *Aktion* mit dem Anweisungstyp *E-Mail-Versendung* im Feld *Inhalt (HTML)*.

Mit `<?print globals.request.bodytext?>` wird als E-Mail-Text ausgegeben, was in einer *Aktion* mit dem Anweisungstyp *E-Mail-Versendung* im Feld *Inhalt (Text)* eingegeben wird.

Bespiel für ein E-Mail-Rahmentemplate:

```
Sehr geehrte(r) <?printx record.v_vorname?> <?printx record.v_nachname?>,

<?print globals.request.bodyhtml?>

Mit freundlichen Grüßen
Ihr Veranstaltungsteam

Signatur
...
```

E-Mail-Inhalt bereitstellen

Die Daten Ihrer App sind über die *LivingAPI* zugänglich. Im zugehörigen Kapitel sind die Objekte ausführlich beschrieben. Haben Sie zusätzlich zum Template eine *Datenquelle* konfiguriert, können Sie über diese z. B. auf *App-Parameter* zugreifen, die zur Verfügung stehenden Daten im Vorhinein einschränken und Daten mehrerer Apps ausgeben.

Den Inhalt Ihrer E-Mail können sie als einfachen Text bei *Text-E-Mail* und/oder als HTML-Code bei *HTML-E-Mail* eingeben. Außerdem stehen jeweils unter *Daten einfügen…* E-Mail-Vorschläge zur Verfügung. Bei Verwendung dieser Vorschläge werden alle Felder Ihrer App, bei denen Eingaben gemacht wurden, entsprechend ihres Typs, richtig ausgegeben.

Template Text-E-Mail
: Schreiben Sie den Inhalt als reine Text-E-Mail, werden die Absätze so in die E-Mail übernommen, wie sie hier eingegeben wurden. Arbeiten Sie mit UL4-Ausdrücken oder möchten Sie andere Templates einbinden, empfiehlt es sich zur besseren Darstellung, einen *<?whitespace?>-Tag* ins Template einzufügen. In der Text-E-Mail haben Sie keine Formatierungsmöglichkeiten.

Template HTML-E-Mail
: Möchten Sie Ihren E-Mail-Text formatieren und mit CSS gestalten, schreiben Sie hier den E-Mail-Inhalt in HTML-Tags.

Datei-Anhang
: Hier können Sie eine Datei auswählen, die an Ihre automatische E-Mail-Benachrichtigung angehängt wird. Achten Sie hier auf eine möglichst geringe Dateigröße.

7.1.2 Zugriff auf Elemente des Datensatzes und Ausgabe von Feldwerten

Die Datensätze der jeweiligen App sind im *Record*-Dictionary der *App* gespeichert. Um auf einzelne Elemente des Datensatzes zuzugreifen und diese in Ihrer E-Mail auszugeben, benötigen Sie, wie in Kapitel 1 erläutert, deren *Identifizierer*. Diese finden Sie in der linken Menüleiste unter *Felder*.

Ist Ihnen der Identifizierer des jeweiligen Feldes bekannt, so können Sie auf dieses mit Punktnotation zugreifen:

```
<?record.fields.identifier.value?>
```

Die Werte stehen auch über „Shortcut"-Attribute zur Verfügung. `record.fields.identifier` steht auch direkt als `record.f_identifier` zur Verfügung. `record.fields.identifier.value` steht auch direkt als `record.v_identifier` zur Verfügung. Wobei `identifier` jeweils für den Identifizierer des Feldes steht, auf das Sie zugreifen möchten.

Um Elemente des Datensatzes in der Benachrichtigungs-E-Mail auszugeben, benutzen Sie den Befehl `printx`:

```
<?printx record.fields.vorname.value?>
```

Shortcut:

```
<?printx record.v_vorname?>
```

greift auf den Wert des Text-Feldes `Vorname` zu und gibt den eingegebenen Vornamen des betrachteten Datensatzes (z. B. Max) aus. Hierbei gilt es zu beachten, dass `vorname` der Identifizierer des Feldes `Vorname` ist.

```
<?printx record.fields.wird_uebernachtung_benoetigt.value.label?>
```

Shortcut:

```
<?printx record.v_wird_uebernachtung_benoetigt.label?>
```

greift auf den Wert des Lookup-Feldes `Wird Übernachtung benötigt` zu und gibt die Bezeichnung der gewählten Option („Ja" oder „Nein") aus. :

```
<?printx record.fields.veranstaltung2.value.fields.veranstaltung.value?>
```

Shortcut:

```
<?printx record.v_veranstaltung2.v_veranstaltung?>
```

greift auf den Wert des Applookup-Feldes `Veranstaltung` zu und gibt den Wert des Feldes `Veranstaltung` (z. B. Excel Grundkurs) der verknüpften App aus.

Mehr darüber können Sie in der Dokumentation der Anzeigetemplates unter *Formulierung des Templates* nachlesen. In der *Übersicht der Feldtypen* erfahren Sie mehr darüber, wie die einzelnen Feldtypen in Templates ausgegeben werden können.

Bevor Sie den Inhalt eines Formularfeldes in der E-Mail ausgeben, sollten Sie prüfen, ob ein Inhalt eingegeben wurde. Einige Felder Ihres Formulars könnten nicht-verpflichtende Felder sein. Somit ist nicht sichergestellt, dass die Ausgabe des Feldinhaltes möglich ist.

Deshalb ist es sinnvoll, Ausgaben immer mit einer `if`-Bedingung zu umschließen, die prüft, ob eine Angabe im betrachteten Feld vorhanden ist. Für das Formularfeld mit dem Identifizierer `vorname` sieht diese `if`-Bedingung bespielsweise so aus:

```
<?if record.fields.vorname.value?>
   <?printx record.fields.vorname.value?>
<?end if?>
```

Shortcut:

```
<?if record.v_vorname?>
   <?printx record.v_vorname?>
<?end if?>
```

Im Kapitel *Prüfung von Feldwerten auf Inhalt* können Sie mehr darüber nachlesen.

7.1.3 Einbinden eines Links in das E-Mail-Template

Haben Sie unter *Aktionen* einen *Link für E-Mails* angelegt, gibt es zwei Möglichkeiten diesen Link in Ihr E-Mail-Template einzubinden. Zum einen über den Link mit `{actionlink:identifier}` und zum anderen über die URL mit `{actionurl:identifier}`. Wobei `identifier` für den Identifizierer der Aktion steht.

`{actionlink:identifier}`
In der HTML-Email wird daraus: `<a href="http://lapp.io/1234567">Aktions-Beschriftung</a>`

In der Text-Email wird daraus: `(Aktions-Beschriftung)[http://lapp.io/1234567]`

`{actionurl:identifier}`
In beiden Fällen wird daraus: `http://lapp.io/1234567`

Haben Sie bei den *Anzeige-Templates* ein *Detailtemplate* angelegt und möchten in Ihrem E-Mail-Template auf dieses Detailtemplate zugreifen, gibt es auch hier die folgenden zwei Möglichkeiten. Wobei hier `identifier` für den Identifizierer des Detailtemplates steht.

`{detaillink:identifier}`
In der HTML-Email wird daraus: `<a href="http://lapp.io/1234567">Detailtemplate-Identifizierer</a>`

In der Text-Email wird daraus: `(Detailtemplate-Identifizierer)[http://lapp.io/1234567]`

`{detailurl:identifier}`
In beiden Fällen wird daraus: `http://lapp.io/1234567`

7.1.4 Einbinden eines Bildes in der HTML-E-Mail

Sie können in der HTML-Version des E-Mail-Templates Bilder einbinden, dies könnte beisipielsweise folgendermaßen aussehen:

```
<img src="https://<?printx globals.hostname?><?printx file.url?>" />
```

Aus Datenschutzgründen wird jedoch dieses Bild möglicherweise vom E-Mail-Program nicht angezeigt, da ein Abrufen des Bildes von einem Webserver es möglich macht, nachzuvollziehen, daß die E-Mail gelesen wurde.

Daher können Sie das E-Mail-Template dazu veranlassen, daß Bild direkt in die E-Mail einzubetten. Dies geschieht über das HTMl-Attribute `data-embed`. Folgende Werte sind bei diesem Attribut zulässig:

`dataurl`
Das Bild wird per `data`-URL eingebunden. Das empfielt sich nur, wenn das Bild nur an einer Stelle verwendet wird.

In diesm Fall müßten Sie folgendes im E-Mail-Template schreiben:

```
<img data-embed="dataurl" src="https://<?printx globals.hostname?><?printx file.
↪url?>" />
```

`attachment`
Das Bild wird als zusätzliches Attachment an die Mail gehängt, und im HTML wird darauf verwiesen.

In diesm Fall müßten Sie folgendes im E-Mail-Template schreiben:

```
<img data-embed="attachment" src="https://<?printx globals.hostname?><?printx␣
↪file.url?>" />
```

`external` (oder auch jeder andere Wert)
Das Bid wird wie bisher vom externen Server geholt.

Datenquellen

Nachdem Sie das E-Mail-Template abgespeichert haben, können Sie Datenquellen dazu anlegen. Über die Datenquelle können Sie z. B. auf App-Parameter zugreifen, die Daten mehrerer Apps ausgeben und die zur Verfügung stehenden Daten im Vorhinein einschränken. Mehr darüber können Sie in der Dokumentation der Anzeigetemplates unter *Anlegen von Datenquellen* und weiter unten im *Anwendungsbeispiel* nachlesen.

7.2 Anwendungsbeispiel

Beim Eingang einer neuen Anmeldung zu einer Veranstaltung, soll der Teilnehmer / die Teilnehmerin automatisch eine Anmeldebestätigung erhalten. Diese E-Mail soll in der Anrede den entsprechenden Namen der Person und in der Einleitung die gewählte Veranstaltung ausgeben. Außerdem sollen alle vom Teilnehmer gemachten Eingaben aufgelistet werden. Zusätzlich enthält die E-Mail einen *Linkf für E-Mails*, über den sich der Teilnehmer wieder von der Veranstaltung abmelden kann.

Am Ende der Anmeldebestätigung sollen noch die neuen Veranstaltungen 2020 zur Information aufgelistet werden. Weil nicht nur auf die Veranstaltung des Datensatzes zugegriffen werden soll, sondern auf alle Veranstaltungen, muss eine *Datenquelle* dafür angelegt werden.

Die *Anmeldebestätigung* soll folgendermaßen aussehen:

Sehr geehrte(r) Max Mustermann,

vielen Dank für die Anmeldung zu unserer Veranstaltung: **Word für Einsteiger**.

Wir haben folgende Informationen von Ihnen erhalten:

Name:	Max Mustermann
Straße Hsnr.:	Musterstraße 3
PLZ Ort:	88888 Musterstadt
E-Mail:	max@example.org
Anzahl Personen:	2
Übernachtung:	Ja
Einzelzimmer:	2
Euro / Nacht:	76

Wir werden uns um die Reservierung Ihres Zimmers / Ihrer Zimmer kümmern.

Rechtzeitig vor Veranstaltungsbeginn erhalten Sie noch weitere Informationen von uns.

Sollten Sie doch nicht teilnehmen wollen, klicken Sie bitte hier: Von Veranstaltung abmelden

Dies ist eine automatisch generierte E-Mail.

Mit freundlichen Grüßen
Ihr Veranstaltungsteam

Abb. 3: Anmeldebestätigung (HTML-Beispiel)

Um ein neues E-Mail-Template anzulegen, wählen Sie im Menü *Konfiguration* → *Erweitert* und dann *E-Mail-Templates*. Klicken Sie anschließend auf *Hinzufügen*.

Im nun geöffneten Fenster werden folgende *Konfigurationen* vorgenommen werden.

Unter der *Bezeichnung* `Anmeldebestätigung` finden sie das E-Mail-Template, wenn sie es z. B. aus der Datenliste manuell verschicken möchten. Da die Anmeldebestätigung bei jeder Neuanmeldung verschickt werden soll, wird bei *Verwendung Bei neuem Datensatz* ausgewählt.

Bei *An (E-Mail-Feld)* wurde *E-Mail-Adresse* gewählt, weil die Anmeldebestätigung an die Adresse gehen soll, die an dieser Stelle vom Teilnehmer im Anmeldeformular eingegeben wurde.

Folgendes Template wurde unter *HTML-E-Mail* angelegt:

```
<html>
<body>
   <p>Sehr geehrte(r) <?printx record.v_vorname?> <?printx record.v_nachname?>,</p>
   <p>
      vielen Dank für die Anmeldung zu unserer Veranstaltung: <b><?printx record.v_
↪veranstaltung2.v_veranstaltung?></b>.
   </p>
   <p>Wir haben folgende Informationen von Ihnen erhalten:</p>
   <table>
      <tr>
         <td>Name:</td>
         <td><?printx record.v_vorname?> <?printx record.v_nachname?></td>
      </tr>
      <tr>
         <td>Straße Hsnr.:</td>
         <td><?printx record.v_strasse_und_hsnr2?></td>
      </tr>
      <tr>
         <td>PLZ Ort:</td>
         <td><?printx record.v_plz_ort?></td>
      </tr>
      <tr>
         <td>E-Mail:</td>
         <td><?printx record.v_e_mail_adresse?></td>
      </tr>
      <?if record.v_telefon?>
         <tr>
            <td>Telefon:</td>
            <td><?printx record.v_telefon?></td>
         </tr>
      <?end if?>
      <tr>
         <td>Anzahl Personen:</td>
         <td><?printx record.v_anzahl_personen2?></td>
      </tr>
      <?if record.v_wird_uebernachtung_benoetigt.key == "ja"?>
         <tr>
            <td>Übernachtung:</td>
            <td>Ja</td>
         </tr>
         <?if record.v_einzelzimmer?>
            <tr>
               <td>Einzelzimmer:</td>
               <td><?printx record.v_einzelzimmer?></td>
            </tr>
         <?end if?>
```

(Fortsetzung auf der nächsten Seite)

E-Mail-Templates anmeldebestaetigung Bezeichnung: Anmeldebestätigung Aktiv?: Ja
Verwendung: Bei neuem Datensatz, Beim Bearbeiten eines Datensatzes Von: Veranstaltungsteam Betreff: Bestätigung der Anmeldung Angelegt am: 05.09.2019 06:57:53 Geändert am: 29.05.2020 08:29:37

1/7

Datenquellen | Text-E-Mail: Meldungen | HTML-E-Mail: Meldungen
Bearbeiten | Text-E-Mail: Versionen | HTML-E-Mail: Versionen

Speichern | Wiederherstellen | Hilfe

* Identifizierer: anmeldebestaetigung
* Bezeichnung: Anmeldebestätigung
Aktiv? ☑
Verwendung: ☑ Bei neuem Datensatz ☐ Beim Bearbeiten eines Datensatzes ☐ Vorlage für Nachricht an alle Datensätze

E-Mail-Adressen

* Von: Veranstaltungsteam
An (E-Mail-Feld): E-Mail-Adresse
Hier wählen Sie, welches E-Mail-Feld ihrer App verwendet werden soll, um den Adressaten der E-Mail festzulegen. Wenn Sie hier nichts auswählen, können Sie einen festen Adressaten bei „An" eingeben.
Kopie:
Blindkopie:
* Antworten an: info@veranstaltung.de

E-Mail-Betreff

* Betreff: Bestätigung der Anmeldung

Text-E-Mail

Template:

```
1 <?whitespace smart?>
2 Sehr geehrte(r) <?print record.v_vorname?> <?print record.v_nachname?>,
3
4 vielen Dank für die Anmeldung zu unserer Veranstaltung: <?print record.v_veranstaltung2.v_veranstaltung?>.
5
6 Wir haben folgende Informationen von Ihnen erhalten:
7
8
```

Hilfe | Daten einfügen ··· | Vollbild

Text-E-Mail: Template-Eigenschaften > Whitespace

HTML-E-Mail

Template:

```
1 <html>
2   <body>
3     <p>Sehr geehrte(r) <?printx record.v_vorname?> <?printx record.v_nachname?>,</p>
4     <p>
5       vielen Dank für die Anmeldung zu unserer Veranstaltung: <b><?printx
        record.v_veranstaltung2.v_veranstaltung?></b>.
6     </p>
7
```

Hilfe | Daten einfügen ··· | Vollbild

Datei-Anhang

Datei-Anhang: Datei auswählen Keine ausgewählt

Historie

Angelegt von: Sabine Voigt | Geändert von: Sabine Voigt
Angelegt am: 05.09.2019 06:57:53 | Geändert am: 29.05.2020 08:29:37

Speichern | Wiederherstellen | Hilfe

Abb. 4: E-Mail-Template Konfiguration

(Fortsetzung der vorherigen Seite)

```
         <?if record.v_euro_nacht?>
            <tr>
               <td>Euro / Nacht:</td>
               <td><?printx record.v_euro_nacht?></td>
            </tr>
         <?end if?>
         <?if record.v_mehrbettzimmer?>
            <tr>
               <td>Doppelzimmer:</td>
               <td><?printx record.v_mehrbettzimmer?></td>
            </tr>
         <?end if?>
         <?if record.v_euro_nacht2?>
            <tr>
               <td>Euro / Nacht:</td>
               <td><?printx record.v_euro_nacht2?></td>
            </tr>
         <?end if?>
            <tr>
               <td colspan="2">Wir werden uns um die Reservierung Ihres Zimmers /␣
↪Ihrer Zimmer kümmern.</td>
            </tr>
      <?else?>
         <tr>
            <td>Übernachtung:</td>
            <td>Nein</td>
         </tr>
      <?end if?>
   </table>
   <p>Rechtzeitig vor Veranstaltungsbeginn erhalten Sie noch weitere Informationen␣
↪von uns.</p>
   <p>Sollten Sie doch nicht teilnehmen wollen, klicken Sie bitte hier:
↪{actionlink:abmelden}</p>
   <p>Dies ist eine automatisch generierte E-Mail.</p>
   <p>
      Mit freundlichen Grüßen <br />
      Ihr Veranstaltungsteam
   </p>
   </body>
</html>
```

Mit der `if`-Bedingung wird überprüft, ob das jeweilige Feld gefüllt ist. Wenn ja, wird der Wert mit `printx` ausgegeben. Diese Überprüfung kann bei Pflichtfeldern weggelassen werden, da sie ja gefüllt sein müssen. Die Auflistung der vom Teilnehmer gemachten Angaben wird in einer Tabelle dargestellt.

Nur wenn bei *Wird Übernachtung benötigt? Ja* gewählt wurde, sollen die Zimmer, Euro/Nacht und der Hinweis zur Reservierung aufgeführt werden. Mit *{actionlink:abmelden}* wird der Link eingebunden, über den sich der Teilnehmer/die Teilnehmerin wieder von der Veranstaltung abmelden kann. Mehr darüber können Sie in der Dokumentation der Aktionen unter *Link für E-Mails* nachlesen.

Als *Text-E-Mail* sieht das Template folgendermaßen aus:

```
<?whitespace smart?>
Sehr geehrte(r) <?print record.v_vorname?> <?print record.v_nachname?>,

vielen Dank für die Anmeldung zu unserer Veranstaltung: <?print record.v_
↪veranstaltung2.v_veranstaltung?>.
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
Wir haben folgende Informationen von Ihnen erhalten:

Name: <?print record.v_vorname?> <?print record.v_nachname?>
Straße Hsnr.: <?print record.v_strasse_und_hsnr2?>
PLZ Ort: <?print record.v_plz_ort?>
E-Mail: <?print record.v_e_mail_adresse?>
<?if record.v_telefon?>
   Telefon: <?print record.v_telefon?>
<?end if?>
Anzahl Personen: <?print record.v_anzahl_personen2?>
<?if record.v_wird_uebernachtung_benoetigt.key == "ja"?>
   Übernachtung: Ja
   <?if record.v_einzelzimmer?>
      Einzelzimmer: <?print record.v_einzelzimmer?>
   <?end if?>
   <?if record.v_euro_nacht?>
      Euro / Nacht: <?print record.v_euro_nacht?>
   <?end if?>
   <?if record.v_mehrbettzimmer?>
      Doppelzimmer: <?print record.v_mehrbettzimmer?>
   <?end if?>
   <?if record.v_euro_nacht2?>
      Euro / Nacht: <?print record.v_euro_nacht2?>
   <?end if?>
   Wir werden uns um die Reservierung Ihres Zimmers / Ihrer Zimmer kümmern.
<?else?>
   Übernachtung: Nein
<?end if?>

Rechtzeitig vor Veranstaltungsbeginn erhalten Sie noch weitere Informationen von uns.

Sollten Sie doch nicht teilnehmen wollen, klicken Sie bitte hier: {actionurl:abmelden}

Dies ist eine automatisch generierte E-Mail.

Mit freundlichen Grüßen

Ihr Veranstaltungsteam
```

Folgendermaßen sieht die Anmeldebestätigung als reine Text-E-Mail aus:

7.2.1 Datenquelle

In unserem Beispiel soll dem Teilnehmer/der Teilnehmerin in der Anmeldebestätigung eine Übersicht der neuesten Veranstaltungen zur Information aufgelistet werden. Um auf alle Veranstaltungen der App zugreifen zu können, müssen Sie dafür eine Datenquelle anlegen. Nachdem Sie Ihr E-Mail-Template das erste Mal gespeichert haben, erscheint neben dem *Bearbeiten* - Button der Button *Datenquelle*.

Um eine Datenquelle anzulegen, klicken Sie auf *Datenquelle* und dann auf *Hinzufügen*. In unserem Beispiel wurden folgende Konfigurationen vorgenommen.

Über den *Identifizierer* `veranstaltungen` können Sie im E-Mail-Template auf die Datenquelle zugreifen. Bei *App* wählen Sie die App aus, auf deren Daten Sie zugreifen wollen. In unserem Beispiel ist es die App *Veranstaltungen (Doku)*. Da nur die Veranstaltungen ab 2020 aufgelistet werden sollen, muss folgender *Datensatzfilter* eingegeben werden:

```
Sehr geehrte(r) Max Mustermann,

vielen Dank für die Anmeldung zu unserer Veranstaltung: Word für Einsteiger.

Wir haben folgende Informationen von Ihnen erhalten:

Name: Max Mustermann
Straße Hsnr.: Beispielstraße 6
PLZ Ort: 66666 Maxstadt
E-Mail: max@example.org
Anzahl Personen: 2
Übernachtung: Ja
Einzelzimmer: 2
Euro / Nacht: 76
Wir werden uns um die Reservierung Ihres Zimmers / Ihrer Zimmer kümmern.

Rechtzeitig vor Veranstaltungsbeginn erhalten Sie noch weitere Informationen von uns.

Sollten Sie doch nicht teilnehmen wollen, klicken Sie bitte hier: http://lapp.io/N177KlP8d

Dies ist eine automatisch generierte E-Mail.

Mit freundlichen Grüßen

Ihr Veranstaltungsteam
```

Abb. 5: Anmeldebestätigung (Text-Beispiel)

```
r.v_datum.year > 2019
```

Eine ausführliche Beschreibung zu den *Datenquellen* und zur *Formulierung der Bedingung* können Sie in der Dokumentation der Anzeigetemplates unter Datenquelle nachlesen.

Folgendermaßen wird im E-Mail-Template auf die Datenquelle und deren Daten zugegriffen:

```
<?for v in datasources.veranstaltungen.app.records.values()?>
   <?printx format(v.v_datum, "%d.%m.%Y")?> - <?printx v.v_veranstaltung?> in <?
↪printx v.v_ort?>
<?end for?>
```

Die *for*-Schleife durchläuft alle vorhandenen Datensätze der Datenquelle. Mit `printx` wird dann das Datum, die Veranstaltung und der Ort ausgegeben.

Folgendes wird dann in der E-Mail ausgegeben:

```
23.05.2020 - Kommunikation/Gesprächsführung im Personalwesen in Bayreuth
12.03.2020 - Telefontraining in Bayreuth
18.01.2020 - Basiswissen Existenzgründung in Bamberg
25.01.2020 - Basiswissen Existenzgründung in Bayreuth
```

7.3 Verwendung von Template-Vorlagen

Mit den Template-Vorlagen haben Sie die Möglichkeit, für zwei Anwendungsfälle jweils ein vorgefertigtes E-Mail-Template zu übernehmen.

Um ein E-Mail-Template mit einer Template-Vorlage zu erstellen, wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *E-Mail-Templates* aus und klicken dann auf *Aus Vorlage hinzufügen*.

Wählen Sie die gewünschte Vorlage aus.

E-Mail-Templates `anmeldebestaetigung` Bezeichnung: Anmeldebestätigung Aktiv?: Ja
Verwendung: Bei neuem Datensatz, Beim Bearbeiten eines Datensatzes Absender-Name: Veranstaltungsteam
Betreff: Bestätigung der Anmeldung

Datenquellen `veranstaltungen` Quelle & Optionen: App **Veranstaltungen (Doku)**; Alle Felder; Alle Datensätze mit Datensatz-Filter `r.v_datum.year > 2019`

Bearbeiten | Sortierung | Untergeordnete Datensätze | Felder

+ Hinzufügen | Kopieren | − Löschen | Hilfe

* Identifizierer: veranstaltungen

Der eindeutige Name dieser Datenquelle. Der Name darf nur Buchstaben, Ziffern und "_" beinhalten. Die Datenquelle mit dem Namen `beispiel` kann in den E-Mail-Templates mittels des Ausdrucks `datasources.beispiel` angesprochen werden.

Quelle

* App: App Veranstaltungen (Doku) (Beispiel-App für Doku)

Wählen Sie hier die App aus deren Datensätze die Datenquelle dem E-Mail-Template zur Verfügung stellen soll. Wird hier „Alle Apps" ausgewählt, so werden die Datensätze aus **allen** Apps zur Verfügung gestellt.

Kopien einbeziehen? ☐

Ist „Kopien einbeziehen?" ausgewählt , so wird nicht nur die App selbst sondern auch alle ihre Kopien dem Template zur Verfügung gestellt.

Felder und Datensätze

* Felder/Datensätze: ○ Keine Daten ○ Felder ◉ Felder und Datensätze

Legt fest ob Informationen zu den Felder bzw. die Datensätze in die Datenquelle aufgenommen werden sollen.

* Felder: ○ Keine Felder ○ Prioritäts-Felder ◉ Alle Felder ○ Alle Felder und Layout-Felder

Legt fest welche Felder in die Datenquelle aufgenommen werden.

Anzahl Datensätze? ☐

Wird „Anzahl Datensätze?" gesetzt, so enthält das `App`-Attribut `recordcount` die Anzahl der Datensätze (ansonsten ist `recordcount` None).

* Datensätze: ○ Angelegte Datensätze ○ Zugewiesene Datensätze ○ Zugewiesene Datensätze bzw. alle Datensätze für Admins ◉ Alle Datensätze

Legt fest welche Datensätze dem Template in der Liste der Datensätze in die Datenquelle aufgenommen werden (bzw. für das `App`-Attribut `recordcount` gezählt werden, falls „Anzahl Datensätze?" gesetzt ist).

Datensatz-Filter: r.v_datum.year > 2019

Diese Bedingung muß ein Datensatz `r` erfüllen, um in die Liste der Datensätze in der Datenquelle aufgenommen zu werden (bzw. um gezählt zu werden, falls „Anzahl Datensätze?" gesetzt ist).

Kompilierter Datensatz-Filter: `r.v_datum.year > 2019`

Sonstige Datensatz-Informationen

Rechte? ☐

Wird „Rechte?" gesetzt, enthalten die `Record`-Objekte unter dem Attribut `permissions` Informationen zu den Zugriffsrechten auf diesem Datensatz (ansonsten ist `permissions` None) (**wird noch nicht unterstützt**).

Anhänge? ☐

Wird „Anhänge?" gesetzt, enthalten die `Record`-Objekte unter dem Attribut `attachments` Informationen zu den Anhängen an diesem Datensatz (ansonsten ist `attachments` None).

Sonstige App-Informationen

Parameter? ☐

Wird „Parameter?" gesetzt, enthält das `App`-Objekt unter dem Attribut `params` die App-Parameter (ansonsten ist `params` None).

Views? ☐

Wird „Views?" gesetzt, enthält das `App`-Objekt unter dem Attribut `views` die Views (ansonsten ist `views` None).

* Kategorien: ◉ Keine Kategorien ○ Kategorien-Pfade ○ Kategorien-Bäume ○ Kategorien-Bäume mit Apps

Legt fest welche Informationen zu den App-Kategorien in den `App`-Objekten zu Verfügung stehen.

Abb. 6: Beispiel Datenquelle

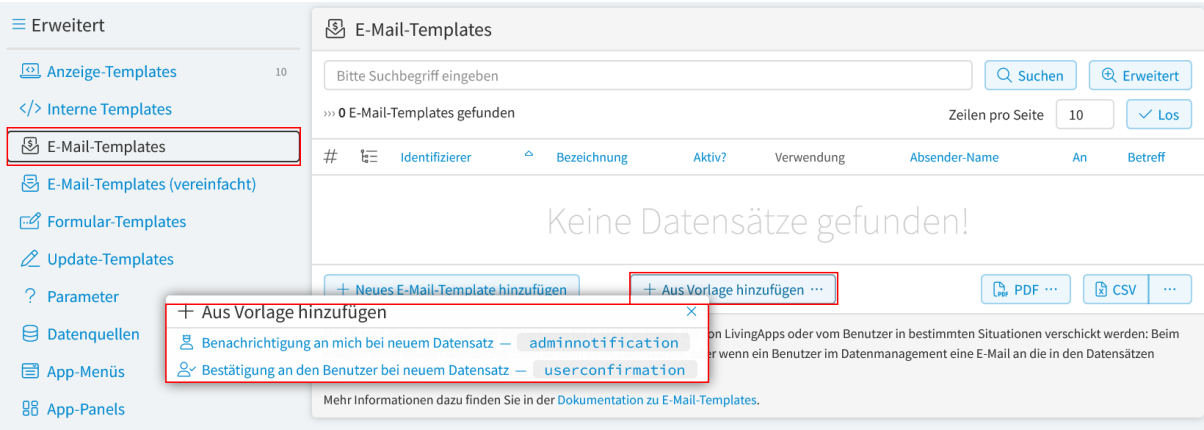

Abb. 7: E-Mail-Template-Vorlage hinzufügen

7.3.1 Benachrichtigung an mich bei neuem Datensatz

Dieses E-Mail-Template versendet automatisch eine E-Mail an Sie, wenn ein Benutzer im Formular einen neuen Datensatz eingibt.

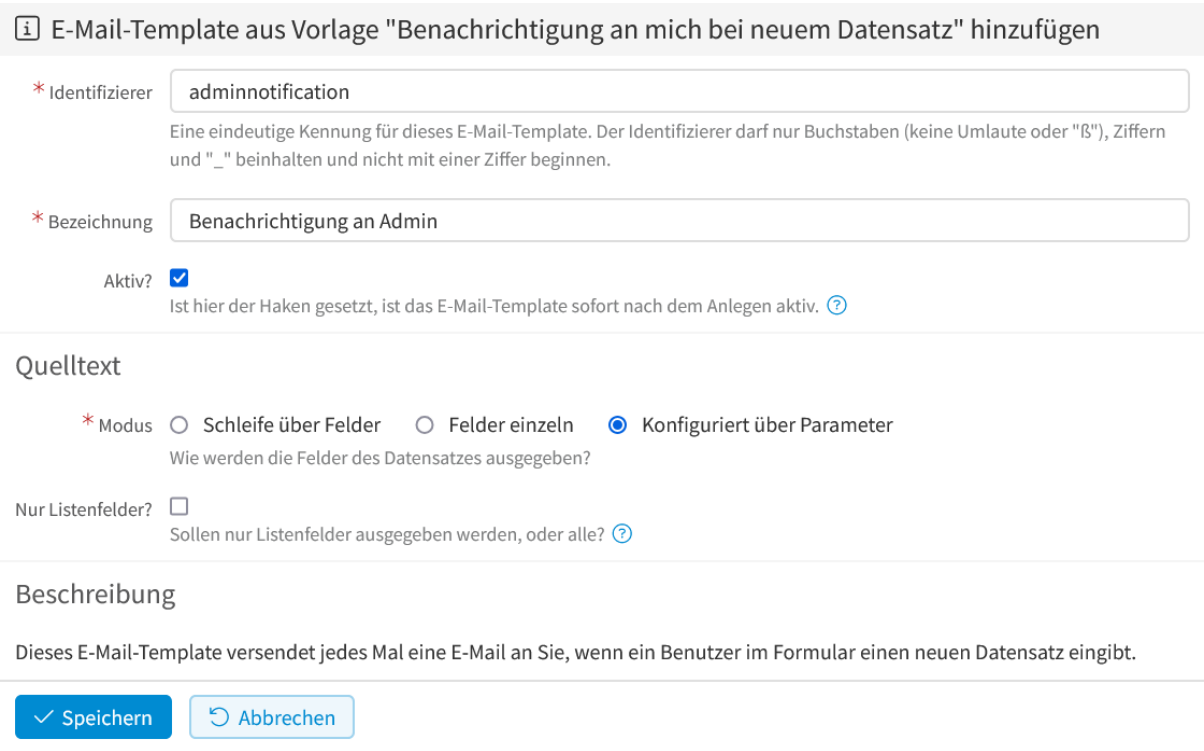

Abb. 8: E-Mail-Template-Vorlage - Benachrichtigung an mich - auswählen

Identifizierer
Wird automatisch gefüllt.

Bezeichnung
Wird automatisch gefüllt.

Aktiv
Ist hier der Haken gesetzt, ist das E-Mail-Template sofort nach dem Anlegen aktiv. Wenn Sie noch Änderungen am E-Mail-Template vornehmen wollen, ist es sinnvoll das E-Mail-Template zuerst inaktiv anzulegen, es dann anzupassen, und erst danach zu aktivieren.

Modus

Schleife über Felder
Die Felder in der Tabellenüberschrift und den Datensatzzeilen werden mit einer Schleife über die Felder ausgegeben, statt dass `la_table_headcols` und `la_table_boldycols` verwendet werden.

Felder einzeln
Hier wird jede Feld-Bezeichnung und jeder Feldwert einzeln ausgegeben. Wenn Sie andere Felder, oder Felder in anderer Reihenfolge oder mit anderem Inhalt ausgeben wollen, ist dies die beste Ausgangsbasis, um den Code nach Ihren Bedürfnissen anzupassen.

Konfiguriert über Parameter
Wenn Sie *Konfiguriert über Parameter* wählen, wird das E-Mail-Template automatisch auch unter *E-Mail-Templates (vereinfacht)* angelegt. Dort können Sie den E-Mail-Inhalt leicht ändern oder ergänzen. Mehr darüber können Sie im *Anwendungsbeispiel* der vereinfachten E-Mail-Templates nachlesen.

Nur Listenfelder
Setzen Sie das Häkchen, wenn nur die Felder ausgegeben werden sollen, die Sie unter *Konfiguration → Liste* ausgewählt haben.

Nach Klicken auf *Speichern* wird das E-Mail-Template angelegt.

Folgende Felder sind dann bereits gefüllt, können aber geändert oder ergänzt werden

Mailserver
Haben Sie einen eigenen SMTP-Server installiert, können Sie diesen hier auswählen. Wenn Sie hier nichts auswählen, erfolgt der Versand über den internen SMTP-Server von LivingApps.

Absendername
Hier kann ein Absendername eingegeben werden. Die E-Mail-Adresse wird automatisch von LivingApps ergänzt. Bei Versendung über den LivingApps-SMTP-Server wird immer die E-Mail-Adresse `benachrichtigung@living-apps.de` verwendet.

Bei Versendung über einen externen SMTP-Server wird die beim SMTP-Server hinterlegte E-Mail-Adresse verwendet.

An
Hier ist der Name und die E-Mail-Adresse des Benutzers, der das E-Mail-Template angelegt hat, bereits eingetragen. Es können noch weitere E-Mail-Adressen, die auch über den neuen Datensatz informiert werden sollen, mit Komma getrennt, eingetragen werden.

Kopie* und *Blindkopie
Hier können Sie vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` sowie mehrere vollständige E-Mail-Adressen (mit Komma getrennt) eintragen.

Antworten an
Ist bereits vorbelegt mit dem Namen und der E-Mail-Adresse des Benutzers, der das E-Mail-Template angelegt hat. Dies kann geändert werden.

Betreff
Ist vorbelegt mit `Neuer Datensatz in der App "Anmeldung"`. Kann individuell geändert werden.

Text-E-Mail
Der E-Mail-Inhalt wird vom Template `la_emailtext_fromparams` erzeugt.

HTML-E-Mail
Der E-Mail-Inhalt wird vom Template `la_emailhtml_fromparams` erzeugt.

Wenn sich jetzt ein Benutzer über das Formular zu einer Veranstaltung anmeldet, wird folgende E-Mail automatisch an Sie verschickt:

* Identifizierer: adminnotification

Eine eindeutige Kennung für dieses E-Mail-Template. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Bezeichnung: Benachrichtigung an Admin

Aktiv? ☑

Verwendung
- ☑ Bei neuem Datensatz
- ☐ Beim Bearbeiten eines Datensatzes
- ☐ Vorlage für Nachricht an alle Datensätze

Mailserver

Mailserver: (Nichts ausgewählt)

E-Mail-Adressen

* Absender-Name: Veranstaltungsmanagement

Der Name des Absenders. Die E-Mail-Adresse wird automatisch von LivingApps ergänzt.

An (E-Mail-Feld): (Nichts ausgewählt)

Hier wählen Sie, welches E-Mail-Feld ihrer App verwendet werden soll, um den Adressaten der E-Mail festzulegen. Wenn Sie hier nichts auswählen, können Sie einen festen Adressaten bei „**An**“ eingeben.

An: Max Mustermann <max.mustermann@livinglogic.de>

Hier können Sie einen festen Adressaten für diese E-Mail festlegen. Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` sowie mehrere vollständige E-Mail-Adressen (mit Komma getrennt) werden unterstützt.

Kopie:

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` sowie mehrere vollständige E-Mail-Adressen (mit Komma getrennt) werden unterstützt.

Blindkopie:

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` sowie mehrere vollständige E-Mail-Adressen (mit Komma getrennt) werden unterstützt.

* Antworten an: Max Mustermann <max.mustermann@livinglogic.de>

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` wird unterstützt.

E-Mail-Betreff

* Betreff: Neuer Datensatz in der App "Anmeldung "

Text-E-Mail

Template

```
1 <?whitespace smart?>
2
3 <?doc?>
4 Text-Version der Benachrichtigungs-E-Mail
5 <?end doc?>
6
7 <?render globals.t_la_emailtext_fromparams()?>
```

Hilfe — Element einfügen ··· | Vollbild

Verfügbare Variablen: `app`, `record`, `datasources`. Verfügbare globale Variablen: `globals`, `la`

Text-E-Mail: Template-Eigenschaften > Whitespace, Beschreibung

HTML-E-Mail

Template

```
1 <?whitespace strip?>
2
3 <?doc?>
4 HTML-Version der Benachrichtigungs-E-Mail
5 <?end doc?>
6
7 <?render globals.t_la_emailhtml_fromparams()?>
```

Hilfe — Element einfügen ··· | Vollbild

Verfügbare Variablen: `app`, `record`, `datasources`. Verfügbare globale Variablen: `globals`, `la`

HTML-E-Mail: Template-Eigenschaften > Whitespace, Beschreibung

Datei-Anhang

Datei-Anhang: Durchsuchen... Keine Datei ausgewählt.

Abb. 9: E-Mail-Template-Vorlage - Benachrichtigung an mich - Konfiguration

Neuer Datensatz in der App "Anmeldung "!

Veranstaltung	Veranstaltung: Kommunikation/Gesprächsführung im Personalwesen, Ort: Bayreuth, Datum: 20.07.2032, von - bis: 10:00 - 18:00
Vorname	Max
Nachname	Mustermann
Straße Hsnr.	Musterstraße 3
PLZ Ort	88888 Musterstadt
E-Mail-Adresse	max@example.org
Anzahl Personen	1,0
Wird Übernachtung benötigt?	Nein
Einzelzimmer	1,0
Euro / Nacht	38,0
Doppelzimmer	0,0
Euro / Nacht	0,0

Abb. 10: E-Mail-Template-Vorlage - Benachrichtigung an mich - Ergebnis

7.3.2 Bestätigung an den Benutzer bei neuem Datensatz

Dieses E-Mail-Template versendet eine Bestätigungs-E-Mail an den Benutzer, der das Formular ausgefüllt hat.

Dazu muß es ein E-Mail-Feld im Formular geben (ansonsten wird Ihre eigene E-Mail-Adresse eingetragen). Gibt es mehr als ein E-Mail-Feld wird das erste verwendet. Sie können aber natürlich das E-Mail-Template nach der Erzeugung entsprechend anpassen.

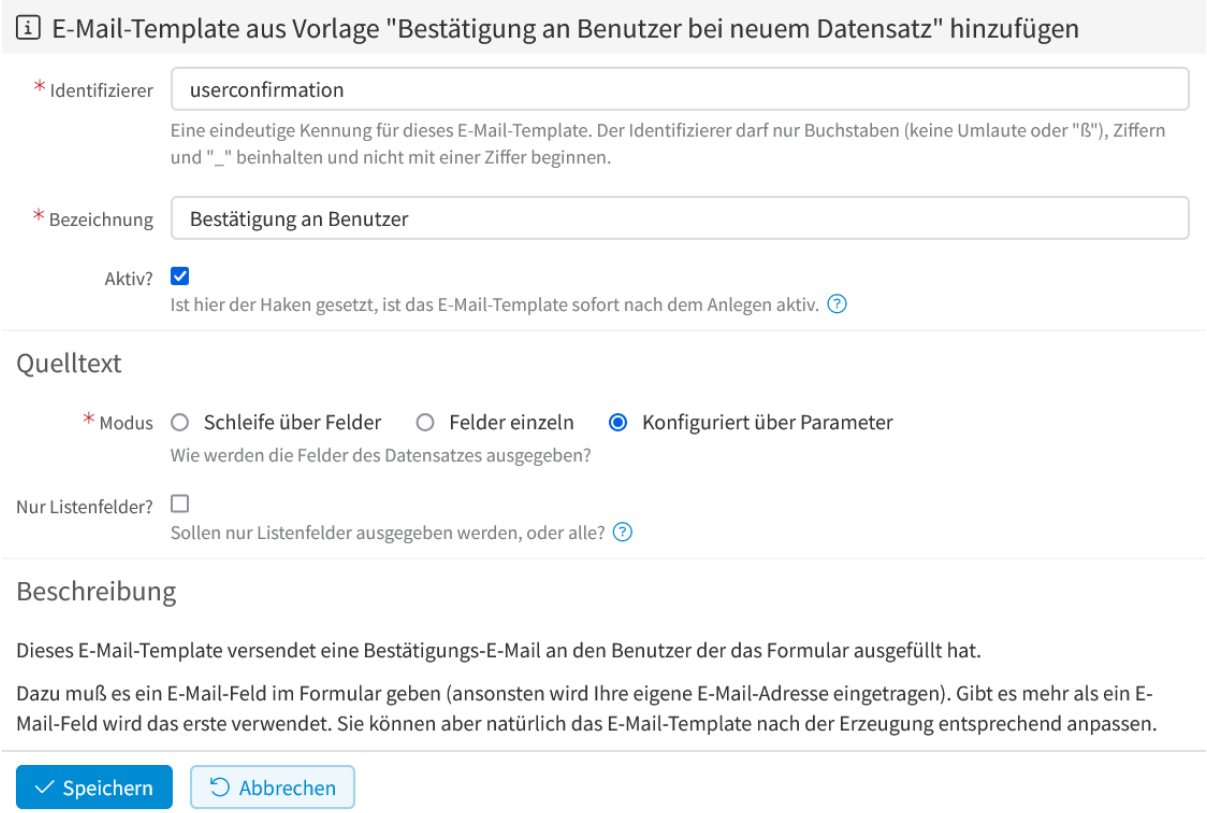

Abb. 11: E-Mail-Template-Vorlage - Bestätigung an Benutzer - auswählen

Identifizierer
Wird automatisch gefüllt.

Bezeichnung
Wird automatisch gefüllt.

Aktiv
Ist hier der Haken gesetzt, ist das E-Mail-Template sofort nach dem Anlegen aktiv. Wenn Sie noch Änderungen am E-Mail-Template vornehmen wollen, ist es sinnvoll das E-Mail-Template zuerst inaktiv anzulegen, es dann anzupassen, und erst danach zu aktivieren.

Modus

Schleife über Felder
Die Felder in der Tabellenüberschrift und den Datensatzzeilen werden mit einer Schleife über die Felder ausgegeben, statt dass `la_table_headcols` und `la_table_boldycols` verwendet werden.

Felder einzeln
Hier wird jede Feld-Bezeichnung und jeder Feldwert einzeln ausgegeben. Wenn Sie andere Felder, oder Felder in anderer Reihenfolge oder mit anderem Inhalt ausgeben wollen, ist dies die beste Ausgangsbasis, um den Code nach Ihren Bedürfnissen anzupassen.

Konfiguriert über Parameter
Wenn Sie *Konfiguriert über Parameter* wählen, wird das E-Mail-Template automatisch auch unter *E-Mail-Templates (vereinfacht)* angelegt. Dort können Sie den E-Mail-Inhalt leicht ändern

oder ergänzen. Mehr darüber können Sie im *Anwendungsbeispiel* der vereinfachten E-Mail-Templates nachlesen.

Nur Listenfelder
Setzen Sie das Häkchen, wenn nur die Felder ausgegeben werden sollen, die Sie unter *Konfiguration* → *Liste* ausgewählt haben.

Nach Klicken auf *Speichern* wird das E-Mail-Template angelegt.

Folgende Felder sind dann bereits gefüllt, können aber geändert oder ergänzt werden

Mailserver
Haben Sie einen eigenen SMTP-Server installiert, können Sie diesen hier auswählen. Wenn Sie hier nichts auswählen, erfolgt der Versand über den internen SMTP-Server von LivingApps.

Absendername
Hier kann ein Absendername eingegeben werden. Die E-Mail-Adresse wird automatisch von LivingApps ergänzt. Bei Versendung über den LivingApps-SMTP-Server wird immer die E-Mail-Adresse `benachrichtigung@living-apps.de` verwendet.

Bei Versendung über einen externen SMTP-Server wird die beim SMTP-Server hinterlegte E-Mail-Adresse verwendet.

An (E-Mail-Feld)
Hier wählen Sie (wenn es mehr als ein E-Mail-Feld in Ihrer LivingApp gibt), welches E-Mail-Feld verwendet werden soll, um den Adressaten der E-Mail festzulegen. Dieses E-Mail-Feld sollte ein Pflichtfeld im Formular der LivingApp sein.

Kopie* und *Blindkopie
Hier können Sie vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` sowie mehrere vollständige E-Mail-Adressen (mit Komma getrennt) eintragen.

Antworten an
Ist bereits vorbelegt mit dem Namen und der E-Mail-Adresse des Benutzers, der das E-Mail-Template angelegt hat. Dies kann geändert werden.

Betreff
Ist vorbelegt mit `Ihre Anmeldung bei "Name der App"`. Kann individuell geändert werden.

Text-E-Mail
Der E-Mail-Inhalt wird vom Template `la_emailtext_fromparams` erzeugt.

HTML-E-Mail
Der E-Mail-Inhalt wird vom Template `la_emailhtml_fromparams` erzeugt.

Wenn sich jetzt ein Benutzer über das Formular zu einer Veranstaltung anmeldet, erhält er automatisch folgende Bestätigung:

7.4 Versendete E-Mails

Klicken Sie im Datenmanagemt unter *Daten* → *Liste* in der linken Navigation auf *E-Mail-Historie*, sehen Sie alle E-Mails, die bisher über die App versendet wurden.

* Identifizierer: userconfirmation

Eine eindeutige Kennung für dieses E-Mail-Template. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Bezeichnung: Bestätigung an Benutzer

Aktiv? ☑

Verwendung
- ☑ Bei neuem Datensatz
- ☐ Beim Bearbeiten eines Datensatzes
- ☐ Vorlage für Nachricht an alle Datensätze

Mailserver

Mailserver: (Nichts ausgewählt)

E-Mail-Adressen

* Absender-Name: Veranstaltungsmanagement

Der Name des Absenders. Die E-Mail-Adresse wird automatisch von LivingApps ergänzt.

An (E-Mail-Feld): E-Mail-Adresse

Hier wählen Sie, welches E-Mail-Feld ihrer App verwendet werden soll, um den Adressaten der E-Mail festzulegen. Wenn Sie hier nichts auswählen, können Sie einen festen Adressaten bei „**An**“ eingeben.

Kopie:

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` sowie mehrere vollständige E-Mail-Adressen (mit Komma getrennt) werden unterstützt.

Blindkopie:

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` sowie mehrere vollständige E-Mail-Adressen (mit Komma getrennt) werden unterstützt.

* Antworten an: Max Mustermann <max.mustermann@livinglogic.de>

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` wird unterstützt.

E-Mail-Betreff

* Betreff: Ihre Anmeldung zur Veranstaltung

Text-E-Mail

Template

```
1 <?whitespace smart?>
2
3 <?doc?>
4 Text-Version der Bestätigungs-E-Mail
5 <?end doc?>
6
7 <?render globals.t_la_emailtext_fromparams()?>
```

Hilfe — Element einfügen ··· | Vollbild

Verfügbare Variablen: `app`, `record`, `datasources`. Verfügbare globale Variablen: `globals`, `la`

Text-E-Mail: Template-Eigenschaften > Whitespace, Beschreibung

HTML-E-Mail

Template

```
1 <?whitespace strip?>
2
3 <?doc?>
4 HTML-Version der Bestätigungs-E-Mail
5 <?end doc?>
6
7 <?render globals.t_la_emailhtml_fromparams()?>
```

Hilfe — Element einfügen ··· | Vollbild

Verfügbare Variablen: `app`, `record`, `datasources`. Verfügbare globale Variablen: `globals`, `la`

HTML-E-Mail: Template-Eigenschaften > Whitespace, Beschreibung

Datei-Anhang

Datei-Anhang: Durchsuchen... Keine Datei ausgewählt.

Abb. 12: E-Mail-Template-Vorlage - Bestätigung an Benutzer - Konfiguration

Vielen Dank für die Übermittlung Ihrer Daten!

Wir haben folgende Informationen von Ihnen erhalten:

Vorname	Erika
Nachname	Muster
Anzahl Personen	1,0
Wird Übernachtung benötigt?	Nein
Einzelzimmer	0,0
Doppelzimmer	0,0

Mit freundlichen Grüßen

Ihr Team

Abb. 13: E-Mail-Template-Vorlage - Bestätigung an Benutzer - Ergebnis

KAPITEL 8

E-Mail-Templates (vereinfacht)

Mit den vereinfachten E-Mail-Templates können Sie, ohne Programmierkenntnisse, einfache Vorlagen für E-Mails anlegen, die von LivingApps oder vom Benutzer in bestimmten Situationen verschickt werden: Beim Anlegen eines neuen Datensatzes, beim Ändern eines Datensatzes, oder wenn ein Benutzer im Datenmanagement eine E-Mail an die in den Datensätzen hinterlegten E-Mail-Adressen verschickt.

Möchten Sie mehr Gestaltungsmöglichkeiten, dann lesen Sie in der Dokumentation der *E-Mail-Templates* nach.

8.1 Erstellung eines vereinfachten E-Mail-Templates

Um ein neues vereinfachtes E-Mail-Template zu erstellen, wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken anschließend in der linken Menüleiste auf *E-Mail-Templates (vereinfacht)* und dann auf *Neues E-Mail-Template hinzufügen*.

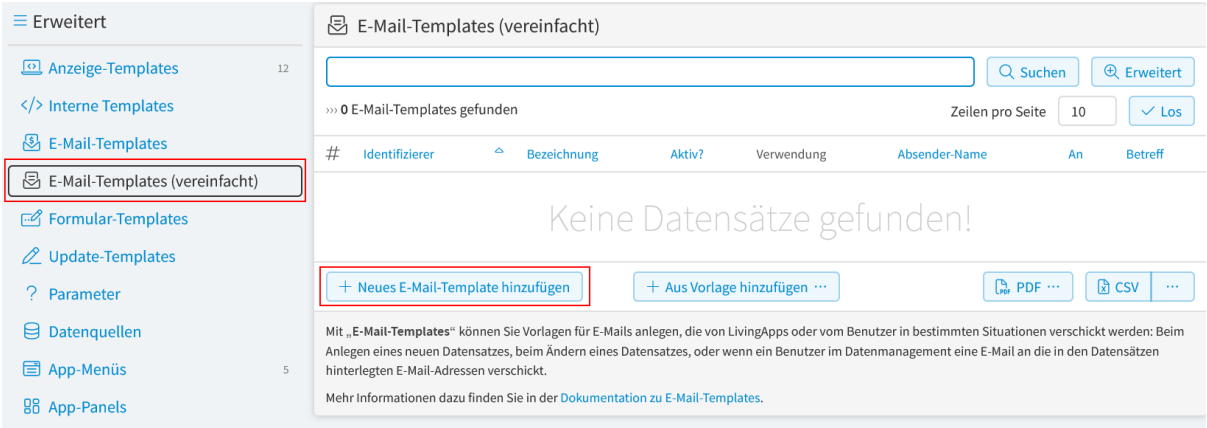

Abb. 1: E-Mail-Template (vereinfacht) hinzufügen

Im nun geöffneten Fenster können folgende Konfigurationen vorgenommen werden.

Identifizierer
: Der Identifizierer ist eine eindeutige Kennung für das Template und darf nur Buchstaben, Zahlen und _ enthalten.

Bezeichnung
: Geben Sie hier die Bezeichnung des E-Mail-Templates an, um eine bessere Übersicht über mehrere Tem-

E-Mail-Templates (vereinfacht) Aktiv? Nein

+ Hinzufügen

Speichern Abbrechen

* Identifizierer

Eine eindeutige Kennung für dieses E-Mail-Template. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Bezeichnung

Aktiv? ☐

Verwendung ☐ Bei neuem Datensatz
☐ Beim Bearbeiten eines Datensatzes
☐ Vorlage für Nachricht an alle Datensätze

Mailserver

Mailserver (Nichts ausgewählt)

E-Mail-Adressen

* Absender-Name

Der Name des Absenders. Die E-Mail-Adresse wird automatisch von LivingApps ergänzt.

An (E-Mail-Feld) E-Mail

Hier wählen Sie, welches E-Mail-Feld ihrer App verwendet werden soll, um den Adressaten der E-Mail festzulegen. Wenn Sie hier nichts auswählen, können Sie einen festen Adressaten bei „An“ eingeben.

Kopie

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` sowie mehrere vollständige E-Mail-Adressen (mit Komma getrennt) werden unterstützt.

Blindkopie

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` sowie mehrere vollständige E-Mail-Adressen (mit Komma getrennt) werden unterstützt.

* Antworten an

Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.org>` wird unterstützt.

E-Mail-Betreff

* Betreff

Datei-Anhang

Datei-Anhang Durchsuchen... Keine Datei ausgewählt.

Speichern Abbrechen

Abb. 2: E-Mail-Template (vereinfacht) konfigurieren

plates zu haben. Unter dieser Bezeichnung finden Sie das E-Mail-Template, wenn Sie es aus der Datenliste manuell verschicken wollen.

Aktiv
Ist hier der Haken gesetzt, ist das E-Mail-Template sofort nach dem Anlegen aktiv. Wenn Sie noch Änderungen am E-Mail-Template vornehmen wollen, ist es sinnvoll das E-Mail-Template zuerst inaktiv anzulegen, es dann anzupassen, und erst danach zu aktivieren.

Verwendung
Hier können Sie festlegen, wann eine E-Mail verschickt werden soll.

Bei neuem Datensatz
Versendet eine Benachrichtigung, wenn ein neuer Datensatz angelegt wurde.

Beim Bearbeiten eines Datensatzes
Versendet eine Benachrichtung, wenn ein bestehender Datensatz verändert wurde.

Vorlage für Nachricht an alle Datensätze
Hier können E-Mail-Vorlagen erstellt werden, die dann aus dem Datenmanagement manuell, oder über eine Aktion *(E-Mail-Versendung)* verschickt werden.

Mailserver
Haben Sie einen eigenen SMTP-Server installiert, können Sie diesen hier auswählen. Wenn Sie hier nichts auswählen, erfolgt der Versand über den internen SMTP-Server von LivingApps.

Absender-Name
Geben Sie hier den Namen ein, der dem Empfänger der E-Mail als Absender angezeigt werden soll.

Bei Versendung über den LivingApps-SMTP-Server wird immer die E-Mail-Adresse `benachrichtigung@living-apps.de` verwendet.

Bei Versendung über einen externen SMTP-Server wird die beim SMTP-Server hinterlegte E-Mail-Adresse verwendet.

An (E-Mail-Feld)
Empfänger der E-Mail. Gibt es im Formular Ihrer App ein oder mehrere Felder vom Typ „E-Mail“, können Sie hier das entsprechende auswählen. Dieses E-Mail-Feld sollte im Formular ein Pflichtfeld sein, sonst kann an die Personen, die keine E-Mail-Adresse angeben, z. B. keine Anmeldebestätigung verschickt werden. Möchten Sie feste Adressaten eingeben, wählen Sie hier *(Nichts ausgewählt)* und geben die E-Mail-Adresse(n) bei *An* ein.

An
Haben Sie bei *An (E-Mail-Feld) (Nichts ausgewählt)* gewählt oder gibt es im Formular Ihrer App kein E-Mail-Feld, können Sie hier feste Adressaten eingeben. Soll die E-Mail an mehrere Empfänger versendet werden, listen Sie hier deren E-Mail-Adressen kommagetrennt auf.

Kopie
Hier können Sie weitere Empfänger angeben. Wenn Sie mehr als eine E-Mail-Adresse eingeben wollen, dann trennen Sie diese durch Kommas.

Blindkopie
An die hier aufgelisteten, kommagetrennten Empfänger wird eine Blindkopie der E-Mail versendet.

Antworten an
Der Empfänger der Benachrichtigung kann auf die E-Mail antworten. Hier legen Sie fest, an welche E-Mail-Adresse die Antwort verschickt werden soll.

Betreff
Schreiben Sie hier den Betreff Ihrer E-Mail.

Datei-Anhang
Hier können Sie eine Datei auswählen, die an Ihre automatische E-Mail-Benachrichtigung angehängt wird. Achten Sie hier auf eine möglichst geringe Dateigröße.

Nachdem Sie auf *Speichern* geklickt haben, können Sie den Nachrichtentext konfigurieren. Klicken Sie dafür oben in der Ansicht auf *E-Mail-Inhalt* und dann auf *Neues Inhalts-Element hinzufügen*.

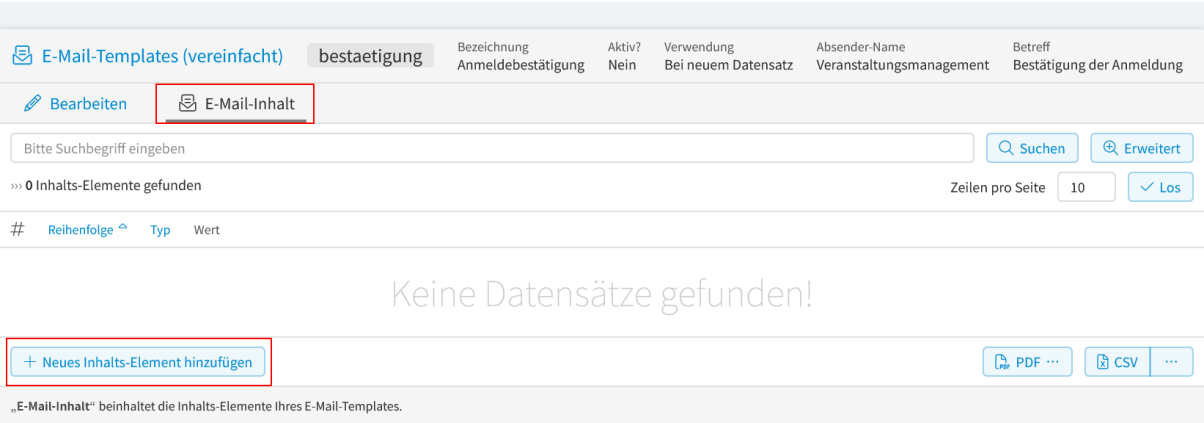

Abb. 3: E-Mail-Template (vereinfacht) E-Mail-Inhalt hinzufügen

Es öffnet sich folgendes Fenster.

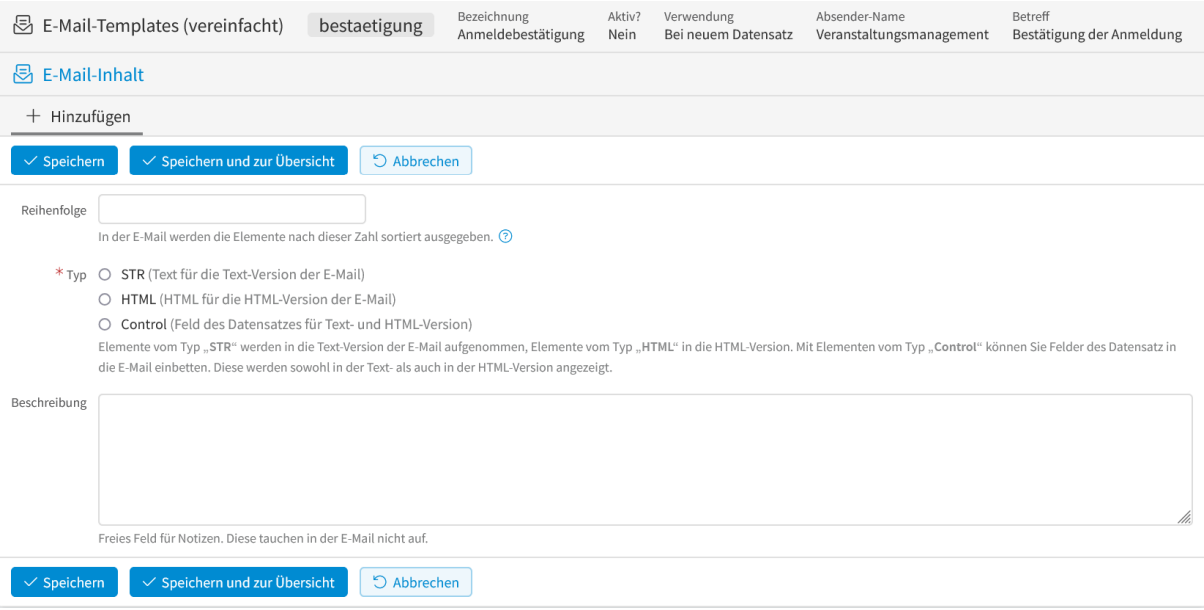

Abb. 4: E-Mail-Template (vereinfacht) E-Mail-Inhalt konfigurieren

Reihenfolge

In der E-Mail werden die Elemente, nach dieser Zahl sortiert, ausgegeben.

Es empfiehlt sich die Elemente in 10er-Schritten durchzunummerieren (10, 20, 30, usw.), um im Bedarfsfall leicht zusätzliche Texte oder andere Elemente „dazwischenschieben“ zu können.

Typ

STR

Text für die Text-Version der E-Mail.

HTML

Text für die HTML-Version der E-Mail.

CONTROL

Feld des Datensatzes (für Text- und HTML-Version).

Beschreibung

Freies Feld für Notizen. Diese tauchen in der E-Mail nicht auf.

8.2 Anwendungsbeispiel

Die Anwendung der vereinfachten E-Mail-Templates wird anhand der automatischen Erstellung durch E-Mail-Template-Vorlagen erklärt.

Zum Beispiel wird, wenn Sie beim Anlegen der E-Mail-Template-Vorlage *Bestätigung an Benutzer* bei *Modus Konfiguriert über Parameter* wählen, automatisch auch unter *E-Mail-Templates (vereinfacht)* das E-Mail-Template angelegt.

Um den Inhalt des E-Mail-Templates einzusehen, wählen Sie im Menü *Konfiguration* → *Erweitert*, klicken dann in der linken Menüleiste auf *E-Mail-Templates (vereinfacht)* und dann in der Auswahl der Templates auf *Bestätigung an Benutzer*.

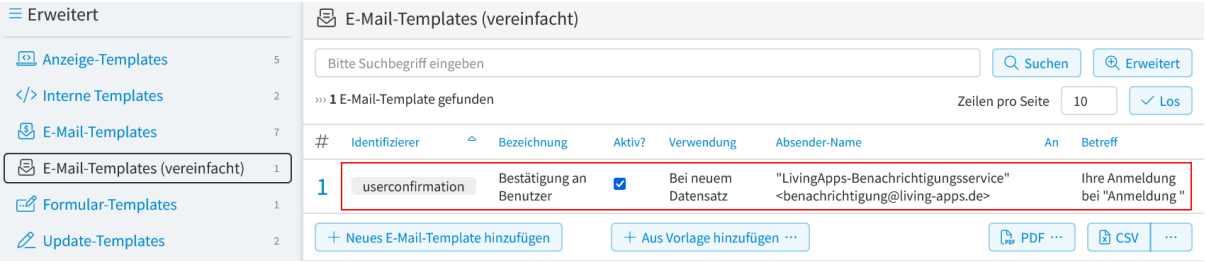

Abb. 5: E-Mail-Template (vereinfacht) Anwendungsbeispiel - E-Mail-Template auswählen

Klicken Sie anschließend auf *E-Mail-Inhalt*.

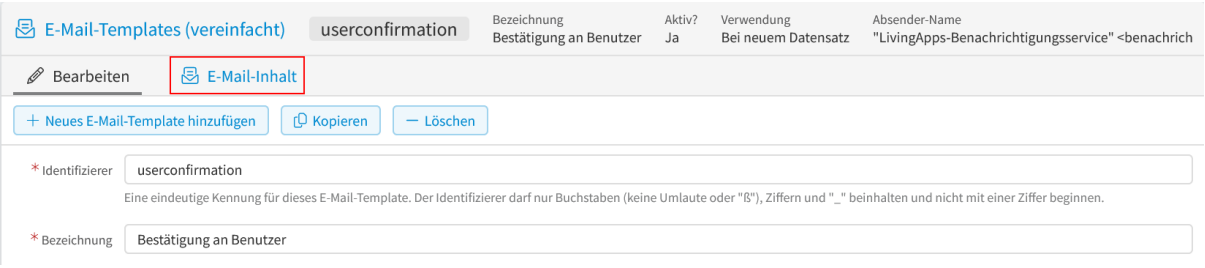

Abb. 6: E-Mail-Template (vereinfacht) Anwendungsbeispiel - E-Mail-Inhalt anzeigen

Es öffnet sich folgendes Fenster:

Hier sehen Sie in Zeile 1 und 2 den Einleitungstext der E-Mail in Textform und in HTML-Form. In Zeile 3 bis 15 sind die einzelnen Felder der LivingApp aufgeführt. In der E-Mail-Bestätigung werden nur die Felder angezeigt, die vom Benutzer bei der Dateneingabe gefüllt wurden.

Durch Anklicken der Zeilen können Sie Änderungen am E-Mail-Inhalt vornehmen.

Möchten Sie zum Beispiel einen anderen Text als Einleitung, dann klicken Sie auf die Zeile 1 mit dem Typ `STR`.

Nehmen sie Ihre Änderungen vor und klicken dann auf *Speichern*. Passen Sie den Text für die `HTML`-Version in Zeile 2 genauso an.

Möchten Sie zum Beispiel, dass die Felder *Einzelzimmer*, *Doppelzimmer* und *Euro/Nacht* nicht in der Bestätigungs-E-Mail angezeigt werden, dann können Sie diese hier aus dem E-Mail-Inhalt löschen. Klicken Sie dafür auf die entsprechende Zeile und dann im geöffneten Fenster auf *Löschen*.

Sie können dem E-Mail-Inhalt auch neue Zeilen hinzufügen. Möchten Sie zum Beispiel zwischen Zeile 9 und 10 noch Text einfügen, klicken Sie in der Übersicht des E-Mail-Inhaltes auf *Neues Inhalts-Element hinzufügen*.

Es öffnet sich folgendes Fenster:

Hier wurden folgende Konfigurationen vorgenommen:

Reihenfolge
: Geben Sie eine Zahl ein, die zwischen den beiden Zeilen liegt, an der Sie den Text einfügen wollen. Im Beispiel muss es eine Zahl zwischen 90 und 100 sein.

E-Mail-Templates (vereinfacht) | userconfirmation | Bezeichnung: Bestätigung an Benutzer | Aktiv?: Ja | Verwendung: Bei neuem Datensatz | Absender-Name: "LivingApps-Benachrichtigungsservice" <benachrich

Bearbeiten | E-Mail-Inhalt

Bitte Suchbegriff eingeben | Suchen | Erweitert

›› 17 Inhalts-Elemente gefunden | Zeilen pro Seite 24 | Los

#	Reihenfolge	Typ	Wert
1	10	STR	'Vielen Dank für die Übermittlung Ihrer Daten!\n\nWir haben folgende Informationen von Ihnen erhalten:'
2	20	HTML	' Vielen Dank für die Übermittlung Ihrer Daten! \n\n Wir haben folgende Informationen von Ihnen erhalten: '
3	30	Control	Veranstaltung applookup/select
4	40	Control	Vorname string/text
5	50	Control	Nachname string/text
6	60	Control	Straße Hsnr. string/text
7	70	Control	PLZ Ort string/text
8	80	Control	E-Mail-Adresse string/email
9	90	Control	Telefon string/tel
10	100	Control	Anzahl Personen number
11	110	Control	Wird Übernachtung benötigt? lookup/radio
12	120	Control	Einzelzimmer number
13	130	Control	Euro / Nacht number
14	140	Control	Doppelzimmer number
15	150	Control	Euro / Nacht number
16	160	STR	'Mit freundlichen Grüßen\nIhr Team'
17	170	HTML	' Mit freundlichen Grüßen \n\n Ihr Team '

+ Neues Inhalts-Element hinzufügen | PDF ··· | CSV | ···

Abb. 7: E-Mail-Template (vereinfacht) Anwendungsbeispiel - E-Mail-Inhalt

E-Mail-Templates (vereinfacht) userconfirmation Bezeichnung Bestätigung an Benutzer Aktiv? Ja Verwendung Bei neuem Datensatz Absende "Living/

E-Mail-Inhalt Typ STR (Text für die Text-Version der E-Mail) 1/17

Bearbeiten

Speichern | Speichern und zur Übersicht | Wiederherstellen

Reihenfolge 10

In der E-Mail werden die Elemente nach dieser Zahl sortiert ausgegeben.

* Typ
- STR (Text für die Text-Version der E-Mail)
- HTML (HTML für die HTML-Version der E-Mail)
- Control (Feld des Datensatzes für Text- und HTML-Version)

Elemente vom Typ „STR“ werden in die Text-Version der E-Mail aufgenommen, Elemente vom Typ „HTML“ in die HTML-Version. Mit Elementen vom Typ „Control“ können Sie Felder des Datensatz in die E-Mail einbetten. Diese werden sowohl in der Text- als auch in der HTML-Version angezeigt.

Text

Vielen Dank für die Übermittlung Ihrer Daten!

Wir haben folgende Informationen von Ihnen erhalten:

Beschreibung

Freies Feld für Notizen. Diese tauchen in der E-Mail nicht auf.

Abb. 8: E-Mail-Template (vereinfacht) Anwendungsbeispiel - E-Mail-Inhalt - Text ändern

E-Mail-Templates (vereinfacht) userconfirmation Bezeichnung Bestätigung an Benutzer Aktiv? Ja Verwendung Bei neuem Datensatz Absende "Living/

E-Mail-Inhalt Typ Control (Feld des Datensatzes für Text- und HTML-Version) 12/17

Bearbeiten

Neues Inhalts-Element hinzufügen | Kopieren | Löschen

Reihenfolge 120

In der E-Mail werden die Elemente nach dieser Zahl sortiert ausgegeben.

* Typ
- STR (Text für die Text-Version der E-Mail)
- HTML (HTML für die HTML-Version der E-Mail)
- Control (Feld des Datensatzes für Text- und HTML-Version)

Elemente vom Typ „STR“ werden in die Text-Version der E-Mail aufgenommen, Elemente vom Typ „HTML“ in die HTML-Version. Mit Elementen vom Typ „Control“ können Sie Felder des Datensatz in die E-Mail einbetten. Diese werden sowohl in der Text- als auch in der HTML-Version angezeigt.

Abb. 9: E-Mail-Template (vereinfacht) Anwendungsbeispiel - E-Mail-Inhalt - Feld löschen

E-Mail-Templates (vereinfacht) | userconfirmation | Bezeichnung: Bestätigung an Benutzer | Aktiv?: Ja | Verwendung: Bei neuem Datensatz | Absender-Name: "LivingApps-Benachrichtigungs | 2/2

Bearbeiten | E-Mail-Inhalt

Bitte Suchbegriff eingeben | Suchen | Erweitert

›› 13 Inhalts-Elemente gefunden | Zeilen pro Seite 24 | Los

#	Reihenfolge	Typ	Wert
1	10	STR	'Sehr geerhte(r) Teilnehmer(in),\r\n\r\nvielen Dank für die Übermittlung Ihrer Daten! \r\n\r\nWir haben folgende Informationen von Ihnen erhalten:'
2	20	HTML	' Sehr geehrte(r) Teilnehmer(in), \r\n vielen Dank für die Übermittlung Ihrer Daten! \r\n Wir haben folgende Informationen von Ihnen erhalten: '
3	30	Control	Veranstaltung applookup/select
4	40	Control	Vorname string/text
5	50	Control	Nachname string/text
6	60	Control	Straße Hsnr. string/text
7	70	Control	PLZ Ort string/text
8	80	Control	E-Mail-Adresse string/email
9	90	Control	Telefon string/tel
10	100	Control	Anzahl Personen number
11	110	Control	Wird Übernachtung benötigt? lookup/radio
12	160	STR	'Mit freundlichen Grüßen\nIhr Team'
13	170	HTML	' Mit freundlichen Grüßen \n\n Ihr Team '

+ Neues Inhalts-Element hinzufügen | PDF ··· | CSV | ···

Abb. 10: E-Mail-Template (vereinfacht) Anwendungsbeispiel - E-Mail-Inhalt - Element hinzufügen

Typ

Wählen Sie STR für die Text-Version der E-Mail.

Text

Geben Sie hier den Text ein, der zusätzlich in der E-Mail angezeigt werden soll.

Klicken Sie auf *Speichern* und wiederholen Sie den Vorgang für die HTML-Version der E-Mail. Wählen Sie dafür bei *Typ HTML* aus und geben denselben Text ein.

Das Ergebnis in der automatisch verschickten E-Mail sieht dann so aus:

E-Mail-Templates (vereinfacht) | userconfirmation | Bezeichnung: Bestätigung an Benutzer | Aktiv?: Ja | Verwendung: Bei neuem Datensatz | Absender-I: "LivingAp

E-Mail-Inhalt 1/13

+ Hinzufügen

Speichern | Speichern und zur Übersicht | Abbrechen

Reihenfolge: 95

In der E-Mail werden die Elemente nach dieser Zahl sortiert ausgegeben.

* Typ
- STR (Text für die Text-Version der E-Mail)
- HTML (HTML für die HTML-Version der E-Mail)
- Control (Feld des Datensatzes für Text- und HTML-Version)

Elemente vom Typ „STR“ werden in die Text-Version der E-Mail aufgenommen, Elemente vom Typ „HTML“ in die HTML-Version. Mit Elementen vom Typ „Control“ können Sie Felder des Datensatz in die E-Mail einbetten. Diese werden sowohl in der Text- als auch in der HTML-Version angezeigt.

Text: Hier steht Ihr zusätzlicher Text!

Beschreibung:

Freies Feld für Notizen. Diese tauchen in der E-Mail nicht auf.

Speichern | Speichern und zur Übersicht | Abbrechen

Abb. 11: E-Mail-Template (vereinfacht) Anwendungsbeispiel - E-Mail-Inhalt - Textelement hinzufügen

Sehr geehrte(r) Teilnehmer(in),

vielen Dank für die Übermittlung Ihrer Daten!

Wir haben folgende Informationen von Ihnen erhalten:

Veranstaltung	Veranstaltung: Kommunikation/Gesprächsführung im Personalwesen, Ort: Bayreuth, Datum: 20.07.2032, von - bis: 10:00 - 18:00
Vorname	Max
Nachname	Mustermann
Straße Hsnr.	Musterstraße 3
PLZ Ort	45612 Musterstadt
E-Mail-Adresse	max.mustermann@example.org

Hier steht Ihr zusätzlicher Text.

Anzahl Personen	1,0
Wird Übernachtung benötigt?	Nein

Mit freundlichen Grüßen

Ihr Team

Abb. 12: E-Mail-Template (vereinfacht) Anwendungsbeispiel - Ergebnis

KAPITEL 9

Formular-Templates

Formular-Templates werden vor und nach dem Ausfüllen eines Eingabeformulars ausgeführt. Damit ist es möglich:

- die Vorbelegung der Felder zu beeinflussen;
- bei Applookup-Feldern eine eigene Auswahl festzulegen, d.h. aus welchen Datensätzen ausgewählt werden kann, wie und in welcher Reihenfolge diese Datensätze angezeigt werden;
- den Datensatz bevor er gespeichert wird nochmals zu verändern;
- ein eigene Gültigkeitsprüfung der eingegebenen Werte zu implementieren;
- nachdem der Datensatz gespeichert wurde noch zusätzliche Datensätze anzulegen, die auf diesen neuen Datensatz verweisen.

Wenn ein Formular-Template definiert ist, so wird dieses Formular-Template während der Phasen der Formular-Verarbeitung ein- oder mehrmals aufgerufen. In welcher Phase man sich befindet, erfährt man mittels *globals.mode*.

9.1 Ablaufplan bei „Neu“-Formularen

Bei einen „Neu“-Formular erfolgt die Verarbeitung in folgenden Phasen:

1. Neues Datensatz-Objekt anlegen;

   Bemerkung

   Dies ist ein temporäres Objekt, das noch nicht in der Datenbank gespeichert ist.

2. Felder des Objektes auf im Formbuilder konfigurierte Feld-Standard-Werte setzen (Vor-/Nachname des eingeloggten Users etc.);
3. Felder des Objektes mit den in der URL übergebenen Parametern überschreiben;
4. Formular-Template im Modus `form/new/init` ausführen;

   Bemerkung

   Zu diesem Zeitpunkt können eigene Feld-Vorbelegungen vorgenommen werden.

Damit wird dann das „Neu“-Formular angezeigt. Wenn das Formular gespeichert wird, passieren folgende Schritte:

5. Daten nach Schritt 4 (siehe oben) laden, wenn nicht vorhanden, Schritte 1. - 4. wiederholen;
6. Felder des Objektes mit den im Formular eingegebenen Werten überschreiben;

Das weitere Vorgehen hängt davon ab, ob bis dahin ein Fehler aufgetreten ist (z.B. wegen nicht ausgefüllter Pflichtfelder).

Bei einem Fehler im Formular passiert folgendes:

7. Formular-Template im Modus `form/new/fail` ausführen.

Danach wird dann wieder das Formular angezeigt.

Wenn der Datensatz gespeichert werden kann passiert folgendes:

8. Formular-Template im Modus `form/new/presave` ausführen;

> **Bemerkung**
>
> Zu diesem Zeitpunkt ist der Datensatz noch nicht in der Datenbank gespeichert und es können noch Änderungen vorgenommen werden, die dann mitgespeichert werden.
>
> Außerdem können hier eigene Fehlermeldungen an das Datensatz-Objekt gehängt werden, die dann das Speichern verhindern (und wie nicht ausgefüllte Pflichtfelder behandelt werden).

9. Datensatz abspeichern;

> **Bemerkung**
>
> Ab diesem Zeitpunkt ist der Datensatz in der Datenbank vorhanden.

10. Datenaktion *Nach Anlegen des Datensatzes ausführen (auch bei Import)* ausführen;
11. Formular-Template im Modus `form/new/postsave` ausführen.

> **Bemerkung**
>
> Zu diesem Zeitpunkt ist der Datensatz gespeichert, daher ist es hier möglich andere Datensätze anzulegen, die sich auf diesen beziehen.

Danach wird die Ergebnisseite angezeigt.

9.2 Ablaufplan bei „Bearbeiten“-Formularen

Bei einem „Bearbeiten“-Formular sieht die Ausführungsreihenfolge ähnlich aus:

1. Datensatz als Objekt laden;
2. Datenaktion *Vor dem Anzeigen des „Bearbeiten“-Formulars ausführen“* ausführen;
3. Felder des Objektes mit den in der URL übergebenen Parametern überschreiben;
4. Formulartemplate im Modus `form/edit/init` ausführen.

> **Bemerkung**
>
> Zu diesem Zeitpunkt können die Feld-Werte nochmal geändert werden, bevor das Formular angezeigt wird).

Damit wird dann das „Bearbeiten“-Formular angezeigt. Wenn das Formular gespeichert wird, passieren folgende Schritte:

5. Daten nach Schritt 4 (siehe oben) laden, wenn nicht vorhanden, Schritte 1. - 4. wiederholen;
6. Felder das Datensatzes mit den im Formular eingegebenen Werten überschreiben;

Auch hier hängt das weitere Vorgehen davon ab, ob bis dahin Fehler aufgetreten sind (z.B. wegen nicht ausgefüllter Pflichtfelder).

Bei einem Fehler im Formular passiert folgendes:

7. Formulartemplate im Modus `form/edit/fail` ausführen.

Danach wird dann wieder das Formular angezeigt.

Wenn der Datensatz gespeichert werden kann passiert folgendes:

8. Formular-Template im Modus `form/edit/presave` ausführen;

> **Bemerkung**
>
> Zu diesem Zeitpunkt ist der geänderte Datensatz noch nicht in der Datenbank gespeichert und es können noch Änderungen vorgenommen werden, die dann mitgespeichert werden. Außerdem können hier eigene Fehlermeldungen an das Datensatz-Objekt gehängt werden, die dann das Speichern verhindern);

9. Datensatz abspeichern;
10. Datenaktion *Nach Änderung des Datensatzes ausführen (auch bei Import)* ausführen;
11. Formular-Template im Modus `form/edit/postsave` ausführen.

> **Bemerkung**
>
> Zzu diesem Zeitpunkt ist der Datensatz gespeichert, daher ist es hier möglich andere Datensätze anzulegen, die sich auf diesen beziehen).

Danach wird die Ergebnisseite angezeigt.

9.3 Erstellung eines Formular-Templates

Um ein Formular-Template für Ihre LivingApp zu erstellen, wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Formular-Templates* und wählen dann die Formular-Ansicht für die Sie ein Formular-Template erstellen wollen.

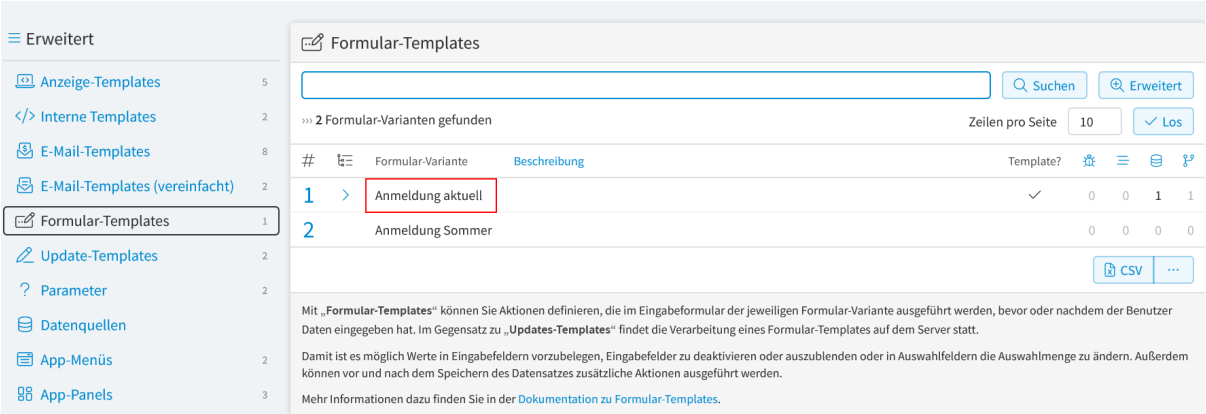

Abb. 1: Formular-Template hinzufügen

Um auf die Daten Ihrer App zugreifen zu können, benötigten Sie die Templatesprache UL4. Die UL4-Dokumentation finden Sie unter https://python.livinglogic.de/UL4.html. Die Ausgabe der Daten erfolgt über die *LivingAPI*. Im zugehörigen Kapitel sind die Objekte ausführlich beschrieben.

Die einzelnen Felder der App werden über den jeweiligen Identifizierer angesprochen. Darüber können Sie unter *Felder und deren Identifizierer* mehr nachlesen.

Möchten Sie in Ihrem Formular-Template z. B. auf App-Parameter zugreifen, die Daten mehrerer Apps ausgeben oder die zur Verfügung stehenden Daten im Vorhinein einschränken, dann können Sie dafür Datenquellen anlegen. Mit `globals.datasources` können Sie auf die Datenquelle zugreifen. Mehr darüber können Sie unter *Anlegen von Datenquellen* nachlesen. Auf die Datenquellen, die Sie im Formular-Template anlegen, können Sie auch von den Update-Templates aus zugreifen.

9.4 Bearbeiten eines Formular-Templates

Gehen Sie vor wie oben beschrieben und wählen die Ansicht aus, deren Formulartemplate sie bearbeiten wollen.

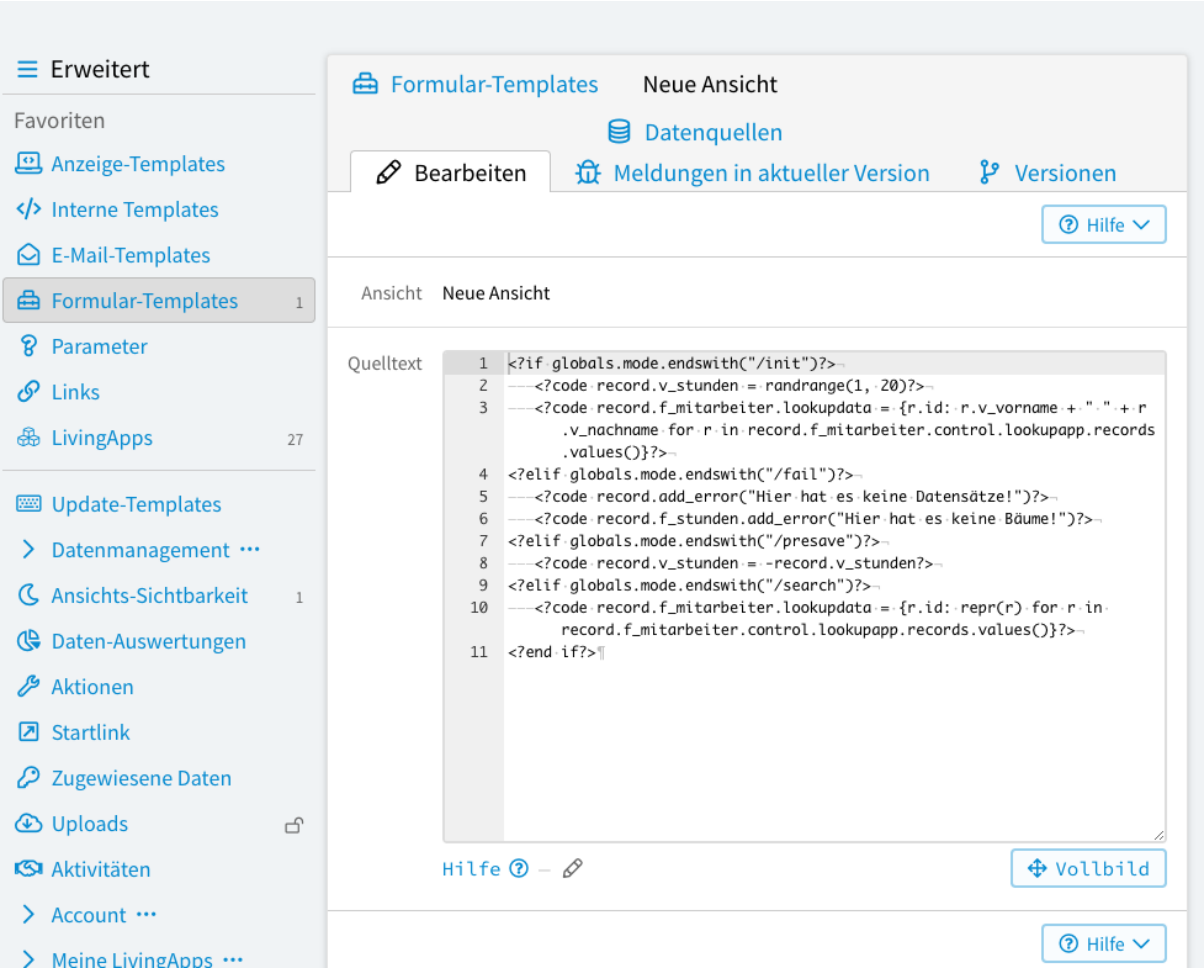

Abb. 2: Formular-Template bearbeiten

9.5 Anwendungsbeispiele

Im Folgenden wird ein Formular-Template in der App „Anmeldung“ aus Kapitel 1 angelegt.

Im Anwendungsbeispiel wird gezeigt, wie Sie Felder mit einer Auswahl vorbelegen können, wie Sie die Auswahl im Applookup einschränken und sortieren und Feldbezeichnungen ändern können.

Beim Feld *Wird Übernachtung benötigt?* soll *Nein* vorausgewählt sein. Die Anzahl der Personen, die mit einer

Anmeldung eingegeben werden können, wird auf 10 begrenzt und es sollen nur ganze Zahlen eingegeben werden können.

Als nächstes wird die Auswahl der Veranstaltungen im Applookup *Veranstaltung* eingeschränkt und zwar sollen nur die Veranstaltungen angezeigt werden, die in der Zukunft liegen und sie sollen in der Auswahl aufsteigend sortiert, also immer die aktuellste oben angezeigt werden. Außerdem soll die Feldbezeichnung in der Auswahl der Dateneingabe nicht zu sehen sein, sondern nur der Inhalt. Also z. B. `Word für Fortgeschrittene` anstatt `Veranstaltung: Word für Fortgeschrittene`.

Dafür muss eine Datenquelle angelegt werden, weil auf die App „Veranstaltungen“ zugegriffen werden soll und die Einschränkung und Sortierung auch über die Datenquelle eingestellt werden. Wie Sie eine Datenquelle anlegen, können Sie unter *Anlegen von Datenquellen* nachlesen.

Folgende Konfigurationen wurden im Anwendungsbeispiel in der Datenquelle vorgenommen:

Datenquelle

Sortierung

Untergeordnete Datensätze (wurde angelegt für Update-Template)

Für das Anwendungsbeispiel wurde die *Ansicht aktuell* gewählt und folgender Code eingegeben:

```
<?code record.f_veranstaltung2.lookupdata = {
   r.id: format(r.v_datum, "%d.%m.%Y") + " " + r.v_veranstaltung
   for r in record.app.c_veranstaltung2.lookupapp.records.values()
}?>
<?note Ändert die Anzeige der Veranstaltungen von Datum:
   01.01.2022, Veranstaltung: Word für Fortgeschrittene
   in 01.01.2022 Word für Fortgeschrittene.?>
<?code record.f_anzahl_personen2.control.precision = 0?>
<?note Für Zahlfelder (Typ = number) können die Anzahl der
   Nachkommastellen angegeben werden.?>
<?code record.f_anzahl_personen2.control.minimum = 0?>
<?note Außerdem kann ein Minimalwert...?>
<?code record.f_anzahl_personen2.control.maximum = 10?>
<?note ... und ein Maximalwert angegeben werden. Bei Integer-Zahlen
   (ganze Zahlen) kann man das im Formbuilder einstellen.?>

<?code record.v_wird_uebernachtung_benoetigt = "nein"?>
<?note Vorbelegung des Lookup-Feldes "Wird Übernachtung benötigt" mit Nein?>
```

Kurze Erklärungen zum Code finden Sie dort als `<?note ...?>`.

Anmeldung

Veranstaltung* 06.09.22 Word für Fortgeschrittene

Anzahl Teilnehmer 0

Ihre persönlichen Daten:

Vorname* Nachname*

Straße Hsnr.* PLZ Ort*

E-Mail-Adresse* Telefon

Mit mir melde ich folgende Anzahl an Personen zu oben genannter Veranstaltung an. Person

Eingabe max. 10
nur ganze Zahlen möglich

Wird Übernachtung benötigt?

Ja Nein

"Nein" ist vorausgewählt

Verbindlich anmelden

Abb. 3: Formular-Template - Felder vorbelegen

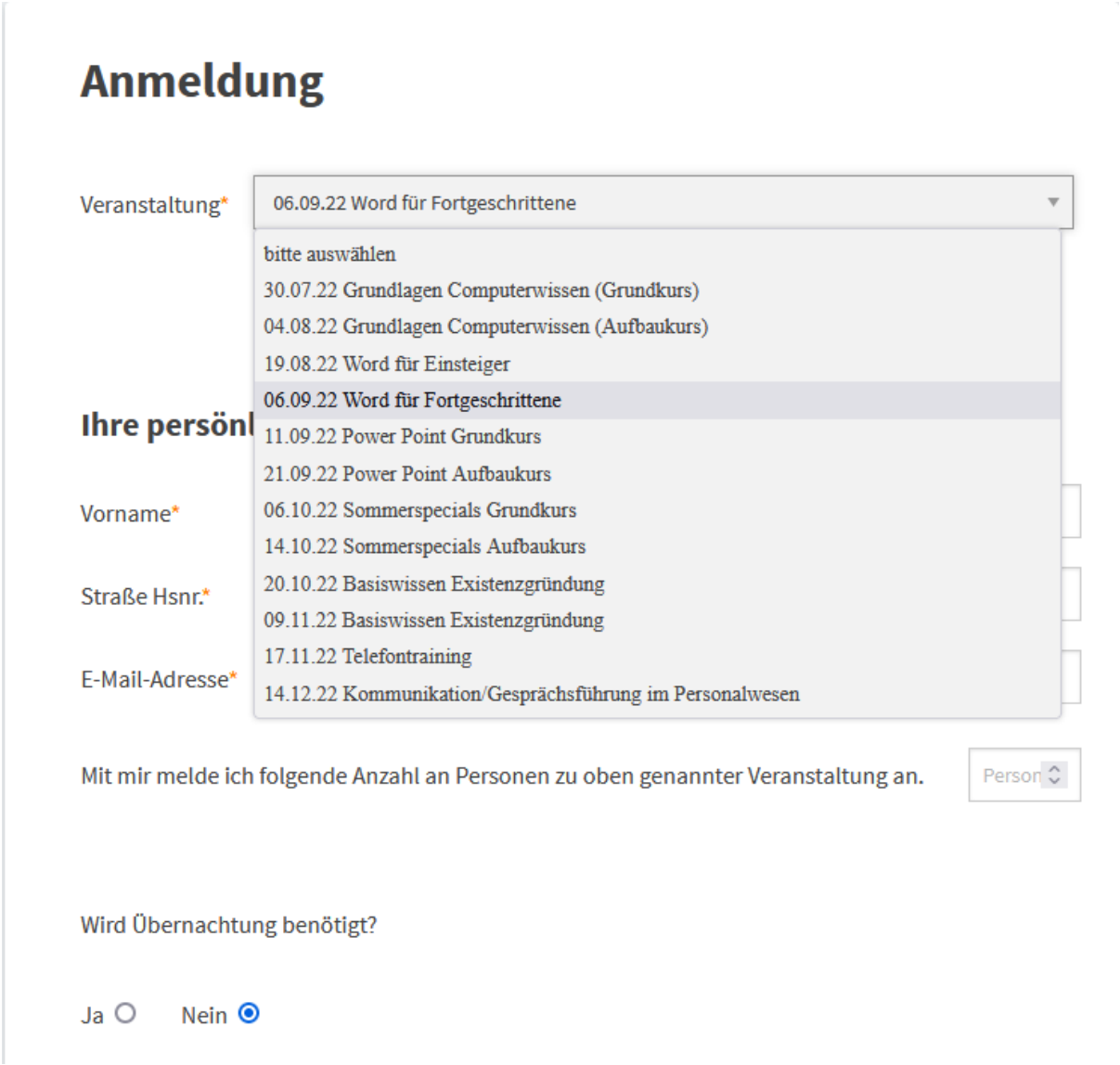

Abb. 4: Formular-Template - Auswahl einschränken, Sortierung

Formular-Templates Anmeldung aktuell

Datenquellen `veranstaltungen` Quelle & Optionen: App **Veranstaltungen (Doku)**; Alle Felder; Alle Datensätze mit Datensatz-Filter `r.v_datum >= today()`; Datensätze sortiert nach `r.v_datum` (A–Z, ?)

Bearbeiten | Sortierung | Untergeordnete Datensätze | Felder

+ Hinzufügen | Kopieren | − Löschen | Hilfe

* Identifizierer: veranstaltungen

Der eindeutige Name dieser Datenquelle. Der Name darf nur Buchstaben, Ziffern und "_" beinhalten. Die Datenquelle mit dem Namen `beispiel` kann in den Anzeige-Templates mittels des Ausdrucks `datasources.beispiel` angesprochen werden.

Quelle

* App: App Veranstaltungen (Doku) (Beispiel-App für Doku)

Wählen Sie hier die App aus deren Datensätze die Datenquelle dem Anzeige-Template zur Verfügung stellen soll. Wird hier „Alle Apps" ausgewählt, so werden die Datensätze aus **allen** Apps zur Verfügung gestellt.

Kopien einbeziehen? ☐

Ist „Kopien einbeziehen?" ausgewählt , so wird nicht nur die App selbst sondern auch alle ihre Kopien dem Template zur Verfügung gestellt.

Felder und Datensätze

* Felder/Datensätze: ○ Keine Daten ○ Felder ◉ Felder und Datensätze

Legt fest ob Informationen zu den Felder bzw. die Datensätze in die Datenquelle aufgenommen werden sollen.

* Felder: ○ Keine Felder ○ Prioritäts-Felder ◉ Alle Felder ○ Alle Felder und Layout-Felder

Legt fest welche Felder in die Datenquelle aufgenommen werden.

Anzahl Datensätze? ☐

Wird „Anzahl Datensätze?" gesetzt, so enthält das App-Attribut `recordcount` die Anzahl der Datensätze (ansonsten ist `recordcount` None).

* Datensätze: ○ Angelegte Datensätze ○ Zugewiesene Datensätze ○ Zugewiesene Datensätze bzw. alle Datensätze für Admins ◉ Alle Datensätze

Legt fest welche Datensätze dem Template in der Liste der Datensätze in die Datenquelle aufgenommen werden (bzw. für das App-Attribut `recordcount` gezählt werden, falls „Anzahl Datensätze?" gesetzt ist).

Datensatz-Filter: r.v_datum >= today()

Diese Bedingung muß ein Datensatz `r` erfüllen, um in die Liste der Datensätze in der Datenquelle aufgenommen zu werden (bzw. um gezählt zu werden, falls „Anzahl Datensätze?" gesetzt ist).

Kompilierter Datensatz-Filter: `r.v_datum >= today()`

Sonstige Datensatz-Informationen

Rechte? ☐

Wird „Rechte?" gesetzt, enthalten die `Record`-Objekte unter dem Attribut `permissions` Informationen zu den Zugriffsrechten auf diesem Datensatz (ansonsten ist `permissions` None) (**wird noch nicht unterstützt**).

Anhänge? ☐

Wird „Anhänge?" gesetzt, enthalten die `Record`-Objekte unter dem Attribut `attachments` Informationen zu den Anhängen an diesem Datensatz (ansonsten ist `attachments` None).

Sonstige App-Informationen

Parameter? ☐

Wird „Parameter?" gesetzt, enthält das App-Objekt unter dem Attribut `params` die App-Parameter (ansonsten ist `params` None).

Views? ☐

Wird „Views?" gesetzt, enthält das App-Objekt unter dem Attribut `views` die Views (ansonsten ist `views` None).

* Kategorien: ◉ Keine Kategorien ○ Kategorien-Pfade ○ Kategorien-Bäume ○ Kategorien-Bäume mit Apps

Legt fest welche Informationen zu den App-Kategorien in den App-Objekten zu Verfügung stehen.

Abb. 5: Formular-Template - Datenquelle

Formular-Templates Anmeldung aktuell

Datenquellen veranstaltungen Quelle & Optionen: App **Veranstaltungen (Doku)**; Alle Felder; Alle Datensätze mit Datensatz-Filter (r.v_datum >= today()); Datensätze sortiert nach (r.v_datum) (A–Z, ?)

Sortierung (r.v_datum) Reihenfolge: 1

Bearbeiten

+ Hinzufügen | Kopieren | − Löschen | Hilfe

* Reihenfolge 1

* Ausdruck r.v_datum

Nach dem Wert dieses Ausdrucks wird der Datensatz r einsortiert.

Kompilierter Ausdruck (r.v_datum)

* Sortier-Richtung ◉ Aufsteigend ○ Absteigend

* Null-Werte ○ Zuerst ◉ Zuletzt

Abb. 6: Formular-Template - Datenquelle Sortierung

Formular-Templates Anmeldung aktuell

Datenquellen veranstaltungen Quelle & Optionen: App **Veranstaltungen (Doku)**; Alle Felder; Alle Datensätze mit Datensatz-Filter (r.v_datum >= today()); Datensätze sortiert nach (r.v_datum) (A–Z, ?)

Untergeordnete Datensätze anmeldungen Ziel: App **Anmeldung** /Feld **Veranstaltung**

Bearbeiten Sortierung

+ Hinzufügen Kopieren — Löschen Hilfe

* Identifizierer anmeldungen

Der eindeutige Name für diese Konfiguration. Der Name darf nur Buchstaben, Ziffern und "_" beinhalten. Die Konfiguration mit dem Namen beispiel kann in den Anzeige-Templates für einen Datensatz record mittels des Ausdrucks record.children.beispiel angesprochen werden.

Ziel
- (Nichts ausgewählt)
- App **Test-App E-Mail-Templates**/Feld **AppLookup**
- App **Test-App E-Mail-Templates**/Feld **MultipleAppLookup**
- App **Doku der Feldtypen**/Feld **Veranstaltung**
- App **Doku der Feldtypen**/Feld **Veranstaltungen**
- App **Anmeldung** /Feld **Veranstaltung**
- App **Doku der Feldtypen**/Feld **Auswahlbox (einfach)**
- System-App **Aktivitäten**/Feld **Datensatz**

Hier können Sie die App auswählen, deren Datensätze den Datensätzen dieser Datenquelle zugeordnet sind.

Filter

Filter-Bedingung

Diese Bedingung muß ein Datensatz r erfüllen, um in die Liste der untergeordneten Datensätze aufgenommen zu werden (zusätzlich zu der Bedingung, daß er dem Hauptdatensatz zugeordnet ist).

Abb. 7: Formular-Template - Datenquelle untergeordnete Datensätze

KAPITEL 10

Update-Templates

Die *Update-Templates* ermöglichen es, im Formular einer LivingApp die Werte von bestimmten Feldern aus anderen Feldern zu berechnen, sowie Felder zu aktivieren, deaktivieren, oder auszublenden. Wenn z. B. einzelne Eingabefelder unter bestimmten Voraussetzungen nicht sichtbar sein sollen, können Sie das Update-Template nutzen, um diese Eingabefelder auszublenden. Sie können für jede Formular-Ansicht ein Update-Template konfigurieren.

Das Update-Template wird ausgeführt bei der Eingabe oder der Bearbeitung eines Feldes, also wenn sich der Inhalt des Feldes ändert.

Möchten Sie in Ihrem Update-Template z. B. auf App-Parameter zugreifen, die Daten mehrerer Apps ausgeben oder die zur Verfügung stehenden Daten im Vorhinein einschränken, dann können Sie auf die Datenquellen zugreifen, die unter Formular-Templates angelegt wurden. Mit `globals.datasources` können Sie auf die Datenquelle zugreifen. Mehr darüber können Sie unter *Anlegen von Datenquellen* nachlesen.

10.1 Erstellung eines Update-Templates

Wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken dann in der linken Menüleiste auf *Update-Template*. Hier sehen Sie nun die verschiedenen Formular-Ansichten Ihrer LivingApp.

Um für eine der Ansichten ein *Update-Template zu erstellen*, klicken Sie auf diese und füllen das Feld *Template*.

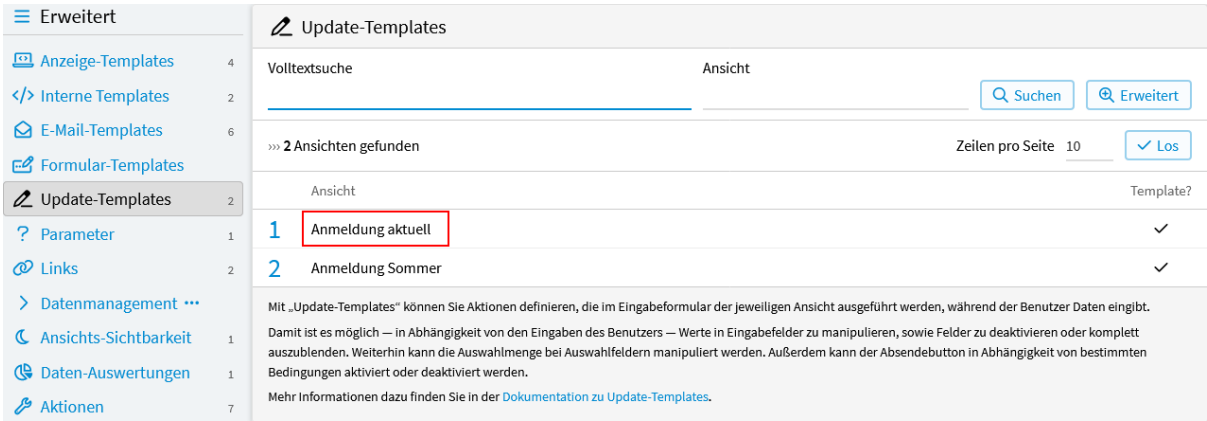

Abb. 1: Update-Template hinzufügen

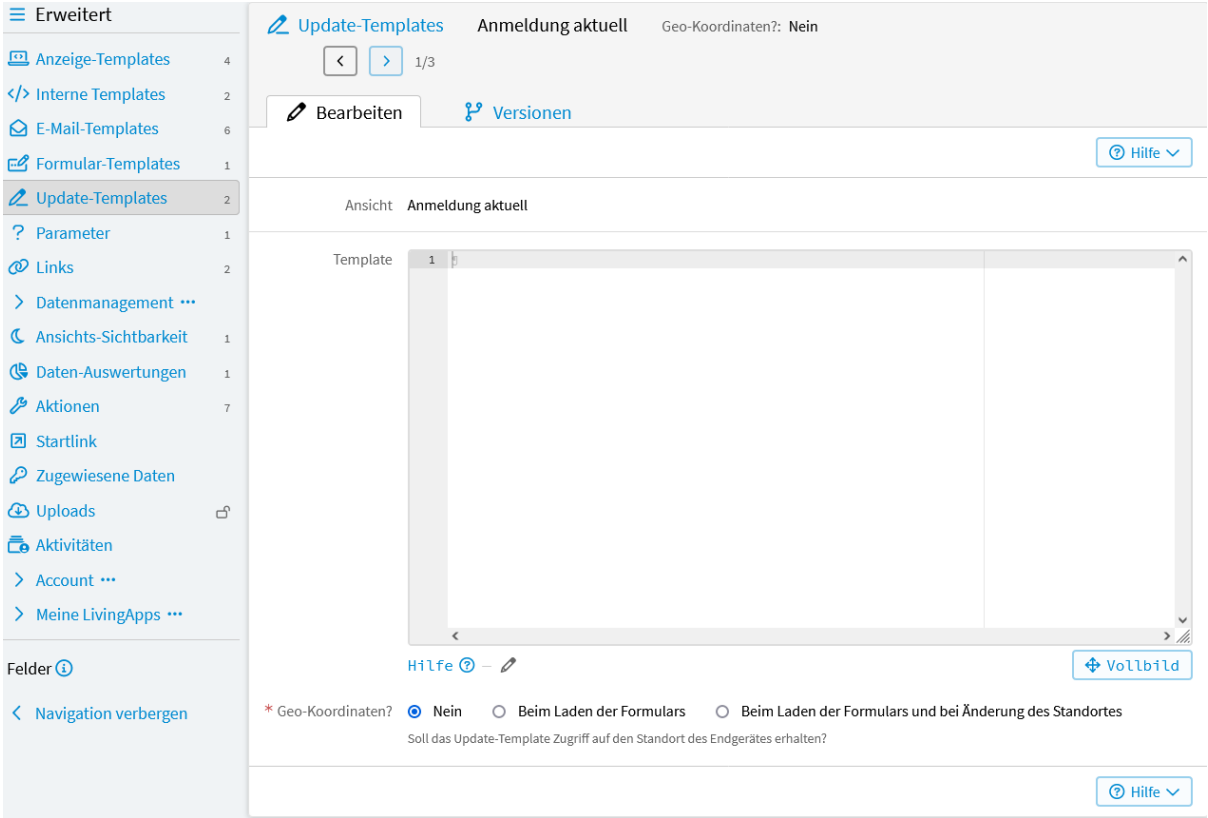

Abb. 2: Update-Template erstellen

Geo-Koordinaten?

Hier können Sie angeben, ob das Update-Template Zugriff auf den Standort des Endgerätes erhalten soll.

Um auf die Daten Ihrer App zugreifen zu können, benötigten Sie die Templatesprache UL4. Die UL4-Dokumentation finden Sie unter https://python.livinglogic.de/UL4.html. Die Ausgabe der Daten erfolgt über die *LivingAPI*. Im zugehörigen Kapitel sind die Objekte ausführlich beschrieben.

Die einzelnen Felder der App werden über den jeweiligen Identifizierer angesprochen. Darüber können Sie unter *Felder und deren Identifizierer* mehr nachlesen.

Dem Update-Template werden drei Parameter übergeben:

`record`

`record` enthält alle aktuellen Daten des in Bearbeitung befindlichen Datensatzes. Details dazu finden Sie in der Dokumentation der LivingAPI unter *Record*.

Außerdem gibt es in Update-Templates unter bei den *Field*-Objekten folgende Attribute:

`writable`

Mit diesem Attribut kann das Feld beschreibar bzw. unbeschreibar gemacht werden. D. h. `record.f_name_writeable = True` entfernt das `readonly`-Attribut vom Eingabe-Feld mit dem Identifizierer `name`, `record.f_name.writable = False` fügt es hinzu. Die Werte aus Feldern, die auf diese Weise als „readonly“ konfiguiert sind, werden ins Datenmanagement übertragen.

`visible`

Mit diesem Attribut kann das Feld ausgeblendet werden. `record.f_name.visible = False` entfernt das Feld mit dem Identifizierer `name` vom Formular. Mit `record.f_name.visible = True` kann man es wieder sichtbar machen. Somit können im Formular-Template und im Update-Template Bedingungen programmiert werden, mit denen man Felder ein- und ausschalten kann. Auch unsichtbare Werte werden ins Datenmanagement übertragen.

`enabled`

Damit kann das Feld aktiviert und deaktiviert werden. D.h. `record.f_name.enabled = True`

entfernt das `disabled`-Attribut vom Eingabe-Feld mit dem Identifizierer `name`, `record.f_name.enabled = False` fügt es hinzu, sodass keine Eingabe mehr möglich ist. Hierbei gilt es zu beachten, dass Werte aus Feldern, die disabled sind, nicht ins Datenmanagement von LivingApps übertragen werden. Sie sind somit nur für den Nutzer des Formulars zu sehen.

`identifier`

Das Template wird einmal nach dem Laden der Seite aufgerufen. Dies ermöglicht es beispielsweise, im Formular dynamische Default-Werte zu setzen oder ähnliches. In diesem Fall hat `identifier` den Wert `None`.

Des Weiteren wird das Template jedes Mal aufgerufen, wenn sich eines der Felder ändert (das heißt bei jedem `change`- oder `keyup`-Event). In diesem Fall wird in `identifier` der Identifizierer des Feldes übergeben, das sich gerade geändert hat.

`globals`

Globals enthält wichtige Grundinformationen über z. B. den angemeldeten Benutzer, die Datenquelle, die im Formular-Template-init geöffnet wurden, die App usw. Details dazu finden Sie in der Dokumentation der LivingAPI unter *Globals*.

Eine Besonderheit in Bezug auf Update-Templates ist die Verwendung der Log-Funktionen wie z.B. `globals.log_info`. Diese Debuginformationen landen in der Javascriptkonsole.

10.2 Anwendungsbeispiel

Im Folgenden wird das Update-Template der App „Anmeldung“ aus Kapitel 1 konfiguriert.

Abhängig von der Auswahl bei *Wird Übernachtung benötigt?* sollen die Felder *Einzelzimmer*, *Doppelzimmer* sowie die zwei Felder *Euro / Nacht* für die Übernachtungskosten *ein- bzw. ausgeblendet* werden.

Ist keine Option oder die Option *Nein* ausgewählt, sollen die restlichen Felder ausgeblendet sein. Ist die Option *Ja* ausgewählt, sollen die Felder eingeblendet werden. Der Nutzer soll dann die Anzahl der Einzel- oder Doppelzimmer eintragen und in den Kosten-Feldern sollen mit dem Update- Template die *Kosten berechnet* werden.

Außerdem soll, abhängig von der Auswahl der Veranstaltung, das entsprechende *Icon* dazu eingeblendet werden und die *Anzahl* der bereits angemeldeten Teilnehmer zu dieser Veranstaltung angezeigt werden.

Dafür muss im Formular der App „Anmeldung“ jeweils ein „Platzhalter“ für das Bild und für die Anzahl der Teilnehmer in Form eines „Formatierten Textfeldes“ hinzugefügt werden. Wählen Sie dafür im Menü *Konfiguration → Eingabe* und ziehen dann das *Formularelement Formatierter Text* zweimal ins Formular.

Als Text wurde im Anwendungsbeispiel „ohne Icon“ für das Bild und „Anzahl Teilnehmer 0“ eingetragen. Um diese „Platzhalter“ im Update-Template anzusprechen, brauchen Sie deren Identifizierer.

Dafür gehen Sie folgendermaßen vor:

Wählen Sie in der Dateneingabe eine Veranstaltung aus, für die ein Icon hochgeladen wurde und klicken dann mit der rechten Maustaste auf das Icon. Im nun geöffneten Fenster wählen Sie *„Untersuchen“*.

Es öffnet sich der Inspektor im Browser. Suchen Sie dort nach dem `div` in dem die *formatierten Textfelder* stehen.

Im Anwendungsbeispiel lautet der Identifizierer für den Platzhalter des Icons `formatierter_text` und für den Platzhalter der Teilnehmeranzahl `formatierter_text2`.

Zum Erstellen des Update-Templates wählen Sie im Menü *Konfiguration → Erweitert* und klicken dann in der linken Menüleiste auf *Update-Template*. Wählen Sie die entsprechende Ansicht aus, für die Sie das Update-Template konfigurieren möchten.

Im Anwendungsbeispiel wurde die *Ansicht aktuell* ausgewählt und dafür folgender Code eingegeben:

```
<?code fields.euro_nacht.writable = False ?>
<?code fields.euro_nacht2.writable = False ?>
<?note Damit kann man in dem Feld "euro_nacht" und "euro_nacht2"
   nichts mehr ändern.?>
```

(Fortsetzung auf der nächsten Seite)

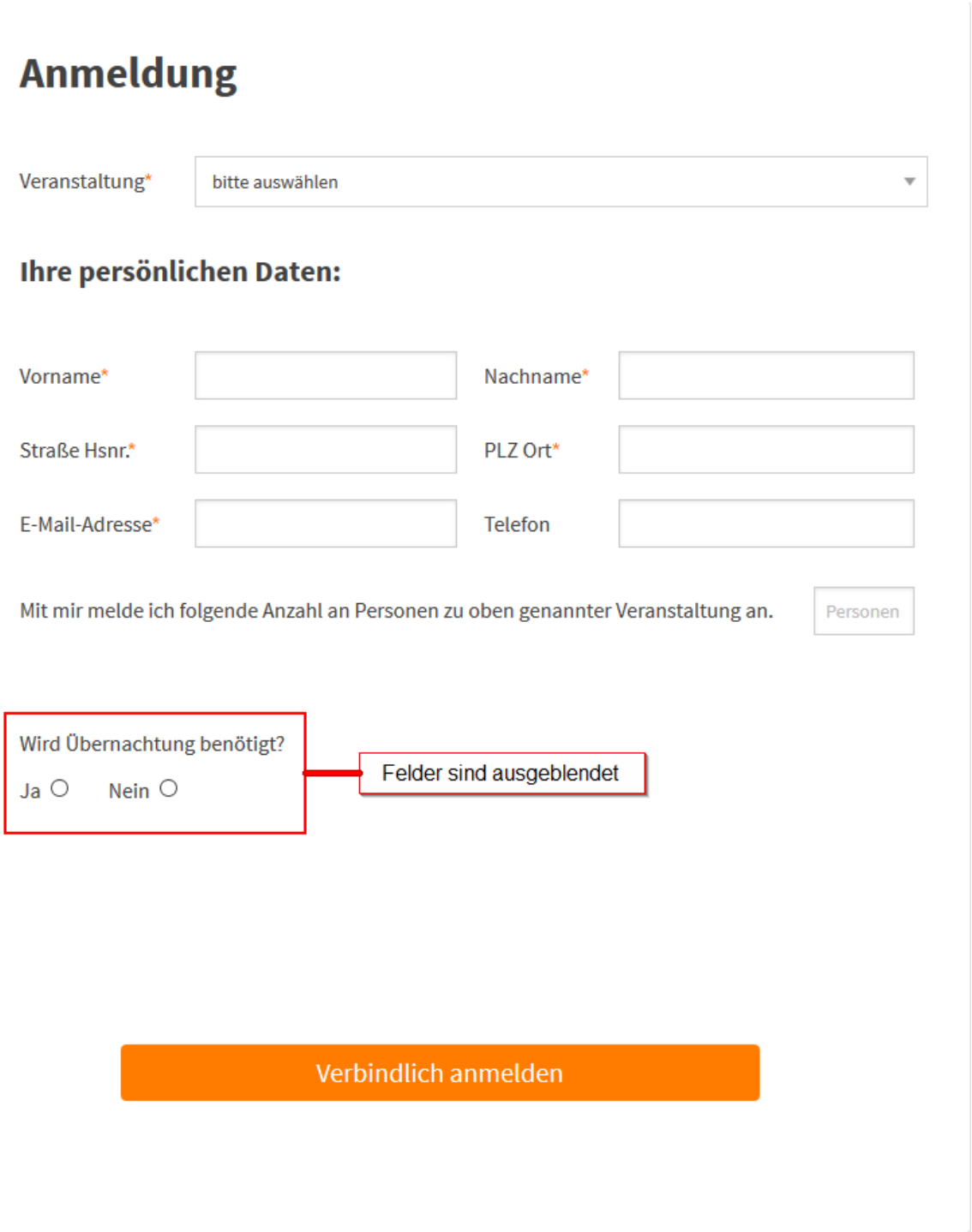

Abb. 3: Update-Template - Felder ein- und ausblenden

Anmeldung

Veranstaltung * bitte auswählen

Ihre persönlichen Daten:

Vorname * Nachname *

Straße Hsnr. * PLZ Ort *

E-Mail-Adresse * Telefon

Mit mir melde ich folgende Anzahl an Personen zu oben genannter Veranstaltung an. Personen

Wird Übernachtung benötigt?

Ja Nein

Einzelzimmer	1	38	Euro / Nacht
Doppelzimmer	2	120	Euro / Nacht

Verbindlich anmelden

Abb. 4: Update-Template - Kosten berechnen

Anmeldung

Veranstaltung* 06.05.17 Power Point Grundkurs

Anzahl Teilnehmer 3

Ihre persönlichen Daten:

Vorname* Nachname*

Straße Hsnr.* PLZ Ort*

E-Mail-Adresse* Telefon

Mit mir melde ich folgende Anzahl an Personen zu oben genannter Veranstaltung an. Person

Wird Übernachtung benötigt?

Ja Nein

Verbindlich anmelden

Abb. 5: Update-Template - Icon einblenden und Teilnehmer anzeigen

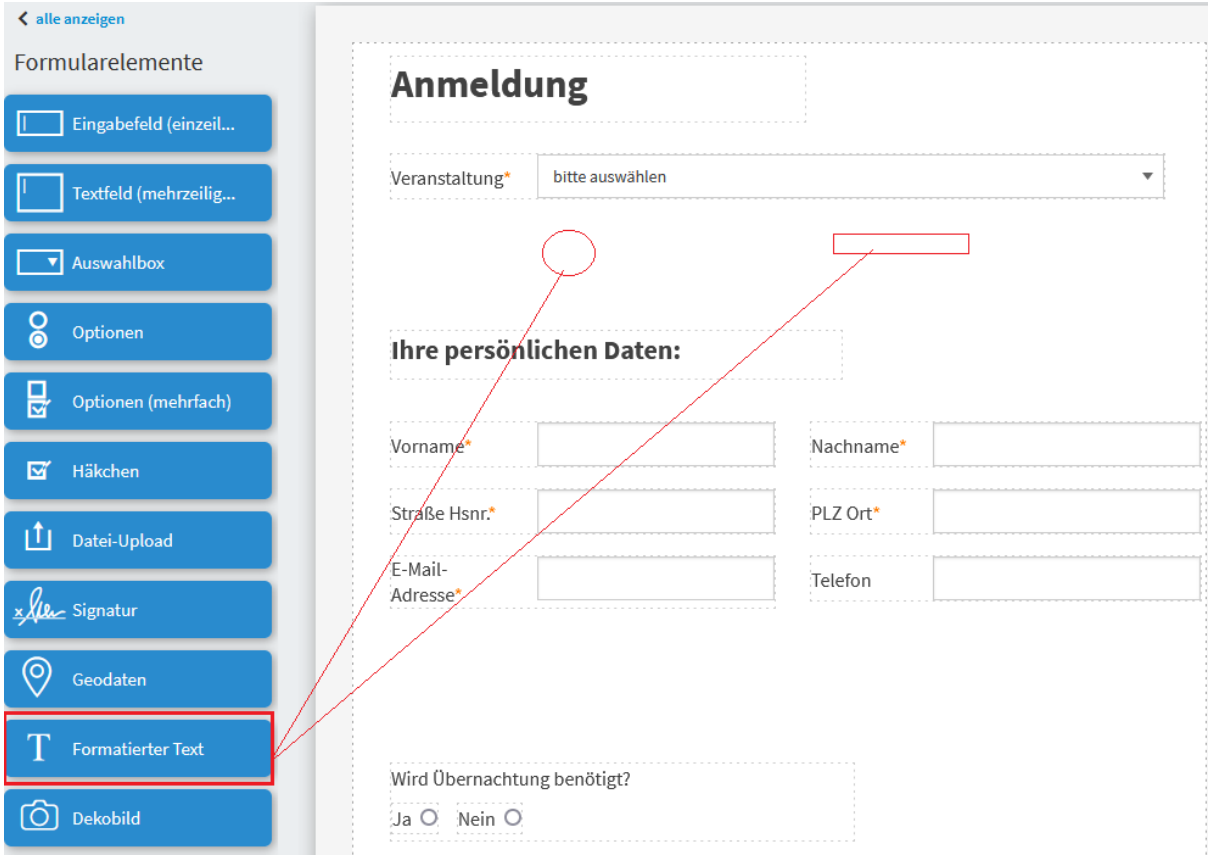

Abb. 6: Update-Template - Platzhalter im Formular erstellen

(Fortsetzung der vorherigen Seite)

```
<?if identifier is None or identifier == "wird_uebernachtung_benoetigt" ?>
   <?code fields.einzelzimmer.visible = fields.wird_uebernachtung_benoetigt.value.key␣
↪== "ja" ?>
   <?code fields.einzelzimmer.value = 0?>
   <?code fields.euro_nacht.visible = fields.wird_uebernachtung_benoetigt.value.key␣
↪== "ja" ?>
   <?code fields.euro_nacht.value = 0?>
   <?code fields.mehrbettzimmer.visible = fields.wird_uebernachtung_benoetigt.value.
↪key == "ja" ?>
   <?code fields.mehrbettzimmer.value = 0?>
   <?code fields.euro_nacht2.visible = fields.wird_uebernachtung_benoetigt.value.key␣
↪== "ja" ?>
   <?code fields.euro_nacht2.value = 0?>
<?end if?>

<?if identifier is None or fields.einzelzimmer.visible ?>
   <?code fields.euro_nacht.value = fields.einzelzimmer.value * 38 ?>
   <?code fields.euro_nacht2.value = fields.mehrbettzimmer.value * 60 ?>
<?end if?>

<?if identifier is None or identifier == "veranstaltung2"?>
   <?note zunächst das Icon der Veranstaltung anzeigen?>
   <?if record.v_veranstaltung2?>
      <?note Berechnungen zur Veranstaltung machen nur Sinn, wenn eine
         Veranstaltung ausgewählt wurde.?>
      <?if record.v_veranstaltung2.v_icon?>
```

(Fortsetzung auf der nächsten Seite)

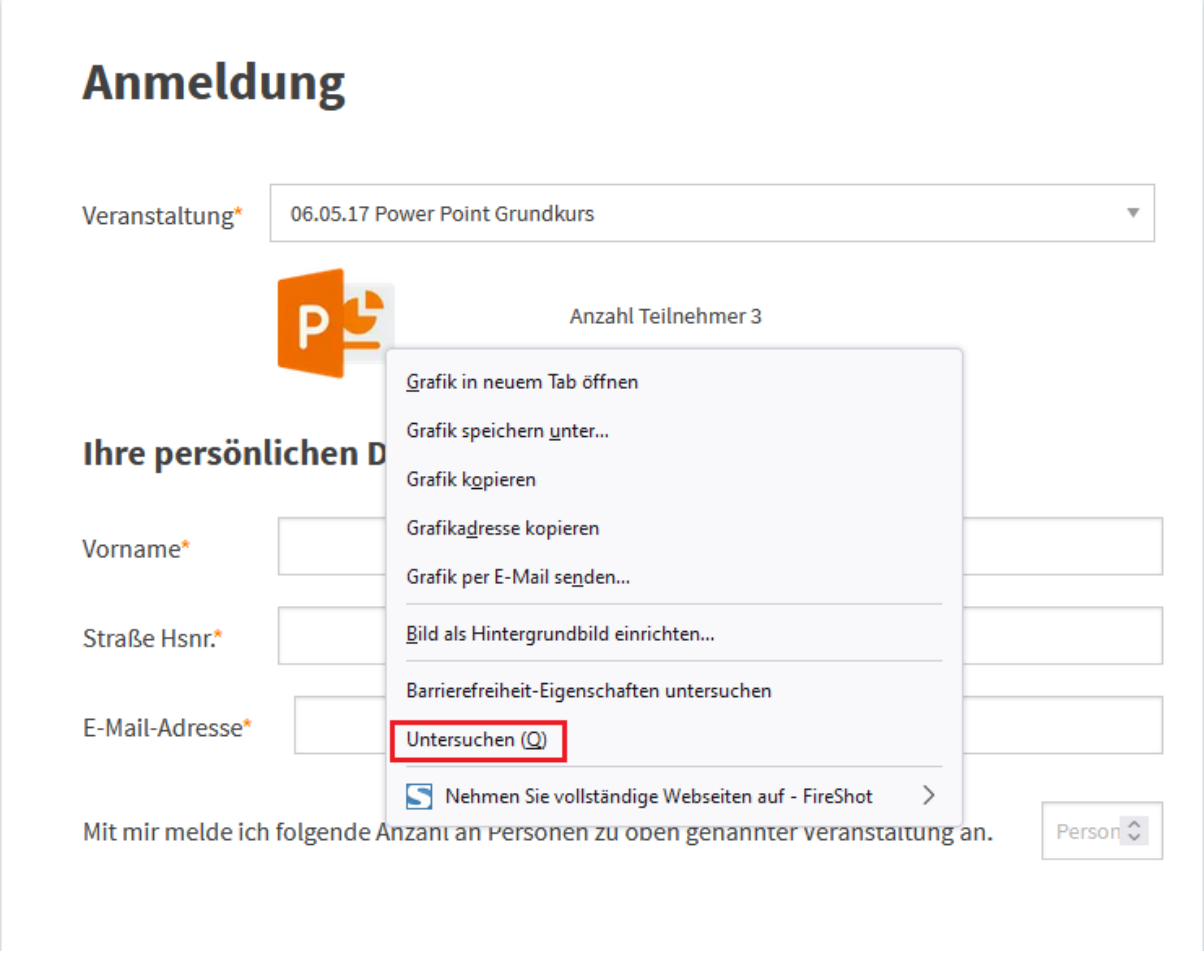

Abb. 7: Update-Template - Identifizierer für Platzhalter finden (Teil 1)

Abb. 8: Update-Template - Identifizierer für Platzhalter finden (Teil 2)

(Fortsetzung der vorherigen Seite)

```
         <?code texticon = '<p><img src="' +
            globals.scaled_url(record.v_veranstaltung2.v_icon,
            30, 30)+'" /></p>'?>
         <?note Wir bauen den HTML-Textschnipsel für die Anzeige des
            Icons. "scaled_url" ist hierbei eine Hilfsfunktion unter
            globals mit der man die URL eines Bildes berechnen
            lassen kann, inklusive der Abmessungen des Bildes
            (hier 30px, 30px).?>
         <?code record.app.lc_formatierter_text.value = texticon?>
         <?note Hier wird der HTML-Schnipsel dem Layoutfeld
            "formatierter_text" zugeordnet. Wie man an den Identifizierer
            des Formatierten Textfeldes rankommt, wird weiter oben in der
            Dokumentation beschrieben.?>
      <?else?>
         <?code record.app.lc_formatierter_text.value = ""?>
      <?end if?>
      <?code record.app.lc_formatierter_text2.value = "Anzahl Teilnehmer " +
         str(sum(1+(a.v_anzahl_personen2 or 0) for a in
         record.v_veranstaltung2.c_anmeldungen.values()))?>
      <?note Das Layoutfeld "formatierter_text2" wird mit der Anzahl der
         Teilnehmer als Zusatzinformation bestückt. Dazu haben wir in
         der Datenquelle "veranstaltung" noch die "Untergeordneten Daten"
         "anmeldungen" konfiguriert. Damit stehen beim Veranstaltungsdatensatz
         unter "c_anmeldungen" die Anmeldedatensätze zu dieser Veranstaltung
         zur Verfügung. Über "sum" können wir also die Anzahl der Teilnehmer
         ermitteln.?>
   <?end if?>
<?end if?>
```

Kurze Erklärungen zum Code finden Sie dort als `<?note ...?>`.

KAPITEL 11

Parameter

App-Parameter sind Konfigurationswerte, auf die in den Anzeige-Templates zugegriffen werden kann. Durch deren Nutzung können Werte und Dateien ohne Anpassung des Template-Codes geändert werden.

11.1 Erstellung von App-Parametern

Wählen Sie im Menu *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Parameter* und dann auf *Hinzufügen*, um einen App-Parameter anzulegen.

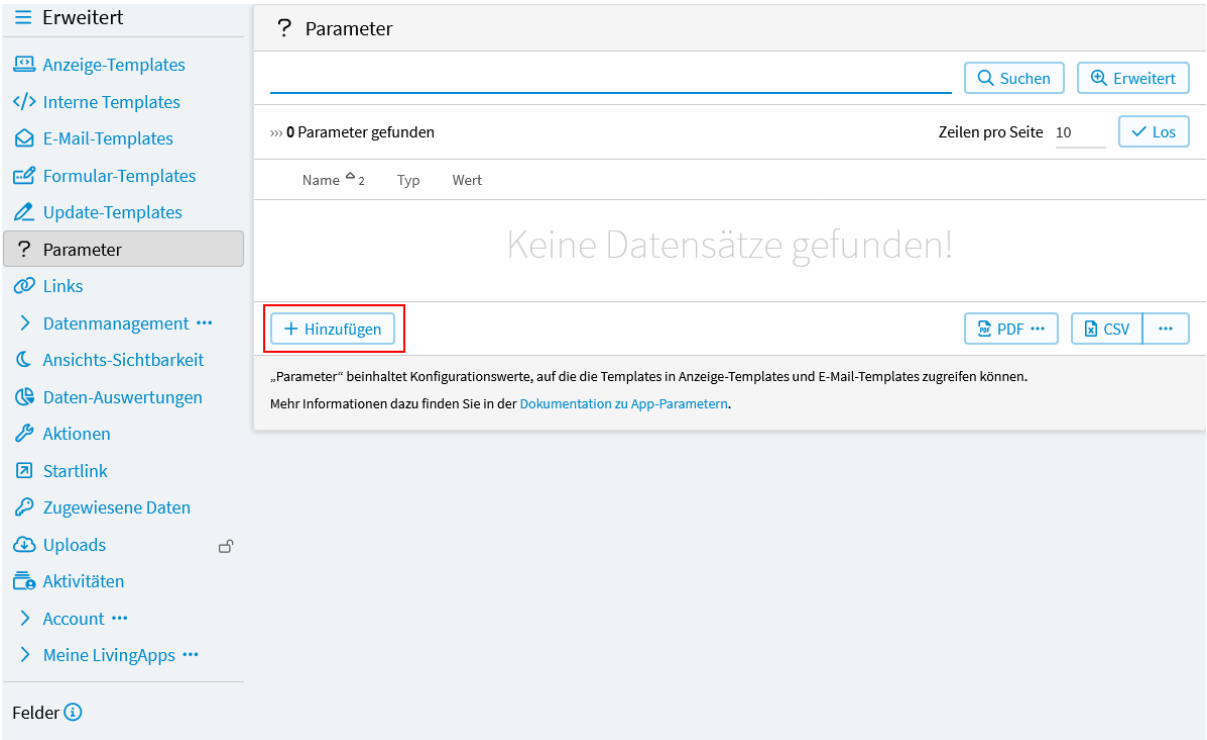

Abb. 1: App-Parameter hinzufügen

Im nun geöffneten Fenster sehen sie die *Eingabeansicht* Ihrer App-Parameter.

Parameter

1/2

+ Hinzufügen

Speichern | Speichern und zur Übersicht | Abbrechen | Hilfe

* Name

Der eindeutige Name des Parameters. Der Name darf nur Buchstaben, Ziffern und "_" beinhalten. Der Parameter mit dem Namen `beispiel` kann in den Anzeige-Templates mittels des Ausdrucks `globals.params.beispiel` angesprochen werden.

* Typ
- BOOL (Wahrheitswert)
- INT (Ganze Zahl)
- NUMBER (Zahl)
- STR (Text)
- COLOR (Farbe)
- DATE (Datum)
- DATETIME (Datum/Uhrzeit)
- DATEDELTA (Zeitraum in Tagen)
- DATETIMEDELTA (Zeitraum in Tagen/Stunden/Minuten/Sekunden)
- MONTHDELTA (Zeitraum in Monaten)
- Upload
- App

Wert

Beschreibung

Historie

Angelegt von	Geändert von
Angelegt am	Geändert am

Speichern | Speichern und zur Übersicht | Abbrechen | Hilfe

Abb. 2: App-Parameter Eingabeansicht

Name

Hier muss ein eindeutiger Name für den App-Parameter festgelegt werden. Der Name darf nur Buchstaben, Ziffern und „_“ beinhalten. Über diesen Namen können sie in Ihren Templates auf den App-Parameter zugreifen.

Typ

Wählen Sie hier den Typ Ihres Parameters.

BOOL Wahrheitswert (Ja/Nein)

INT Ganze Zahl (z.B. `2`)

NUMBER Zahl (z.B. `2.394`)

STR Text (z.B. `gurk`)

COLOR Farbe (z.B. `#ff0000`)

DATE Datum (z.B. `29.02.2000`)

DATETIME Datum/Uhrzeit (z.B. `29.02.2000 16:05:54`)

DATEDELTA Zeitraum im Tagen (z.B. `10`)

DATETIMEDELTA Zeitraum in Tagen/Stunden/Minuten/Sekunden (z.B. 5 `days, 12:33:15`)

MONTHDELTA Zeitraum in Monaten (z.B. `6`)

Upload Dateien (z.B. ein Bild oder eine PDF-Datei)

App die entsprechende App kann hier ausgewählt werden.

Wert

Je nach Typ des Parameters wird hier der Wert festgelegt. Beispiele siehe oben in der Klammer.

Beschreibung

Die Beschreibung der Parameter ist nur intern sichtbar.

11.2 Zugriff in E-Mail- und Anzeige-Templates

Um auf die App-Parameter zugreifen zu können, muss im Template eine *Datenquelle* konfiguriert und in dieser bei *Parameter?* das Häkchen gesetzt werden.

In den E-Mail- und Anzeige-Templates kann auf die App-Parameter mittels des Ausdrucks `app.params.beispiel` oder `app.p_beispiel` (Shortcut-Attribut) zugegriffen werden. Wobei `beispiel` jeweils der Name des Parameters ist.

Auf den Wert kann direkt mit dem Shortcut-Attribut `app.pv_beispiel` zugegriffen werden.

Der Wert des App-Parameters kann mit dem `print`-Statement:

```
<?printx app.pv_beispiel?>
```

ausgegeben werden.

Ist der App-Parameter vom Typ *DATE* kann das Datum in deutscher Schreibweise ausgegeben werden mit:

```
<?printx format(app.pv_beispiel, "%d.%m.%Y", "de")?>
```

Bei App-Parametern vom Typ *Upload* erhält man die URL, unter welcher die Datei zu finden ist mit:

```
<?printx app.pv_beispiel.url?>
```

Ist der App-Parameter vom Typ *App*, kann die App-ID ausgegeben werden mit:

```
<?printx app.pv_beispiel.id?>
```

Mehr Informationen zum `App`- und `AppParameter`-Objekt finden Sie unter *App* und *AppParameter*.

11.3 Anwendungsbeispiel

Um die Anwendung der App-Parameter zu demonstrieren, wird wieder auf das Anwendungsbeispiel aus Kapitel 1 zurückgegriffen. Es soll ein App-Parameter angelegt werden, der als Überschrift des Anzeigetemplates eingesetzt werden soll.

Wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Parameter* und anschließend auf *Hinzufügen*.

Im nun geöffneten Fenster werden folgende *Konfigurationen* vorgenommen.

Der *Name* soll `ueberschrift` lauten. Damit können Sie im E-Mail- oder Anzeigetemplate auf den App-Parameter zugreifen. Als *Typ* wird *STR (Text)* ausgewählt. Bei *Wert* wird die Bezeichnung der Überschrift - `Teilnehmerliste` eingegeben.

Wie Sie den App-Parameter in Ihren Templates ausgeben, können Sie im Abschnitt *Zugriff in E-Mail- und Anzeige-Templates* nachlesen.

Parameter

1/2

+ Hinzufügen

Speichern | Speichern und zur Übersicht | Abbrechen | Hilfe

* Name ueberschrift

Der eindeutige Name des Parameters. Der Name darf nur Buchstaben, Ziffern und "_" beinhalten. Der Parameter mit dem Namen `beispiel` kann in den Anzeige-Templates mittels des Ausdrucks `globals.params.beispiel` angesprochen werden.

* Typ
- BOOL (Wahrheitswert)
- INT (Ganze Zahl)
- NUMBER (Zahl)
- STR (Text)
- COLOR (Farbe)
- DATE (Datum)
- DATETIME (Datum/Uhrzeit)
- DATEDELTA (Zeitraum in Tagen)
- DATETIMEDELTA (Zeitraum in Tagen/Stunden/Minuten/Sekunden)
- MONTHDELTA (Zeitraum in Monaten)
- Upload
- App

Wert Teilnehmerliste

Beschreibung Überschrift des Anzeigetemplates.

Historie

Angelegt von | Geändert von

Angelegt am | Geändert am

Speichern | Speichern und zur Übersicht | Abbrechen | Hilfe

Abb. 3: App-Parameter erstellen

KAPITEL 12

Datenquellen

Unter *Datenquellen* sehen Sie alle *Datenquellen*, die Sie für ein Anzeige-, E-Mail- oder Formular-Template dieser LivingApp angelegt haben oder die diese LivingApp als Ziel-App haben.

Hierüber können die Datenquellen gelöscht werden.

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und klicken dann in der linken Menüspalte auf *Datenquellen*.

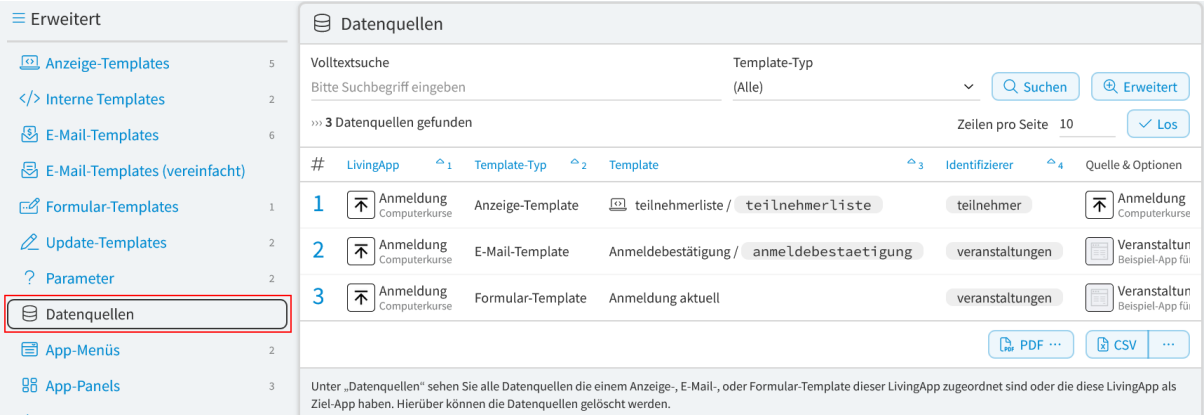

Abb. 1: Datenquellen auswählen

Klicken Sie in der Auswahl auf die Datenquelle, die Sie anschauen wollen und klicken Sie auf *Löschen*, wenn Sie die Datenquelle löschen wollen.

Datenquellen | Anmeldung Computerkurse | teilnehmer | Template-Typ: Anzeige-Template | Template: | 1/3

Anzeigen

— Löschen | Hilfe

LivingApp: Anmeldung Computerkurse

Template-Typ: Anzeige-Template

Template: teilnehmerliste / `teilnehmerliste`

Identifizierer: teilnehmer

Die eindeutige Kennung dieser Datenquelle. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen. Die Datenquelle mit dem Namen `beispiel` kann in den Anzeige-Templates mittels des Ausdrucks `globals.datasources.beispiel` angesprochen werden.

Quelle

App: App Anmeldung (Computerkurse)

Wählen Sie hier die App aus deren Datensätze die Datenquelle dem Anzeige-Template zur Verfügung stellen soll. Wird hier „Alle Apps“ ausgewählt, so werden die Datensätze aus **allen** Apps zur Verfügung gestellt.

Kopien einbeziehen?: Nein

Ist „Kopien einbeziehen?“ ausgewählt , so wird nicht nur die App selbst sondern auch alle ihre Kopien dem Template zur Verfügung gestellt.

Felder und Datensätze

Felder/Datensätze: Felder und Datensätze

Legt fest ob Informationen zu den Felder bzw. die Datensätze in die Datenquelle aufgenommen werden sollen.

Felder: Alle Felder

Legt fest welche Felder in die Datenquelle aufgenommen werden.

Anzahl Datensätze?: Nein

Wird „Anzahl Datensätze?“ gesetzt, so enthält das `App`-Attribut `recordcount` die Anzahl der Datensätze (ansonsten ist `recordcount` `None`).

Datensätze: Alle Datensätze

Legt fest welche Datensätze dem Template in der Liste der Datensätze in die Datenquelle aufgenommen werden (bzw. für das `App`-Attribut `recordcount` gezählt werden, falls „Anzahl Datensätze?“ gesetzt ist).

Datensatz-Filter:

Diese Bedingung muß ein Datensatz `r` erfüllen, um in die Liste der Datensätze in der Datenquelle aufgenommen zu werden (bzw. um gezählt zu werden, falls „Anzahl Datensätze?“ gesetzt ist).

Sonstige Datensatz-Informationen

Rechte?: Nein

Wird „Rechte?“ gesetzt, enthalten die `Record`-Objekte unter dem Attribut `permissions` Informationen zu den Zugriffsrechten auf diesem Datensatz (ansonsten ist `permissions` `None`) (**wird noch nicht unterstützt**).

Anhänge?: Nein

Wird „Anhänge?“ gesetzt, enthalten die `Record`-Objekte unter dem Attribut `attachments` Informationen zu den Anhängen an diesem Datensatz (ansonsten ist `attachments` `None`).

Sonstige App-Informationen

Parameter?: Ja

Wird „Parameter?“ gesetzt, enthält das `App`-Objekt unter dem Attribut `params` die App-Parameter (ansonsten ist `params` `None`).

Views?: Nein

Wird „Views?“ gesetzt, enthält das `App`-Objekt unter dem Attribut `views` die Views (ansonsten ist `views` `None`).

Kategorien: Keine Kategorien

Legt fest welche Informationen zu den App-Kategorien in den `App`-Objekten zu Verfügung stehen.

Abb. 2: Datenquellen anzeigen und löschen

KAPITEL 13

App-Menüs

Die erweiterte Funktion *App-Menüs* ermöglicht es Ihnen im Menü auf beliebige Webseiten zu verlinken, um so das Navigationsmenü auf der Detailseite Ihrer LivingApp zu erweitern. Sollten Sie Links auf der App-Übersichtsseite anlegen wollen, lesen Sie bitte bei *Benutzer-Menüs* nach.

Haben Sie beispielsweise Auswertungen mittels der Anzeige-Templates erstellt, so können Sie auf diesem Wege einfach und schnell auf diese zugreifen (siehe *Link zu einem Anzeige-Template*).

13.1 Erstellung von App-Menüs

Wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *App-Menüs* und dann auf *Neuen Link oder neues Menü zur Menüleiste hinzufügen*.

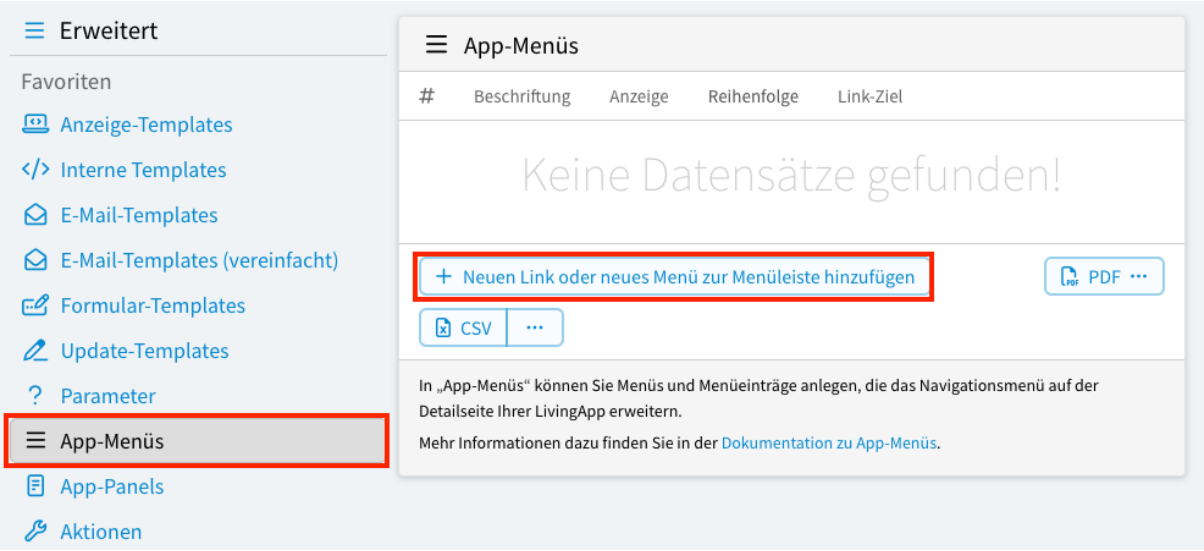

Abb. 1: Hinzufügen eines Menü-Links

Im nun geöffneten Fenster können folgende Konfigurationen vorgenommen werden.

Identifizierer
: Eindeutige Kennung für diesen Menüeintrag. Der Identifizierer darf nur Buchstaben, (keine Umlaute oder „ß“) Ziffern und „_“ beinhalten und nicht mit einer Ziffer beginnen.

App-Menüs

+ Hinzufügen

Speichern | Abbrechen

* Identifizierer

Eine eindeutige Kennung für diesen Menüeintrag. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Beschriftung

Icon

Das Icon für den Link

Anzeigen auf

App-Startseite?

Eingabeformular?

iframe-Seite?

Anzeige-Templates?

eigener Übersichtsseite?

Handy-App?

Position in der Menüleiste

Übergeordnetes Menü (Nichts ausgewählt)

Reihenfolge

Nach dieser Zahl sortiert werden Menüs in der Menü-Leiste bzw. Menü-Einträge in Untermenüs angezeigt.

Link-Ziel

* Typ (Nichts ausgewählt)

Link-Attribute

Title

Das HTML-Attribut `title`. Der Text den Sie hier eingeben, wird beim Link als Tooltip angezeigt (d.h. er wird angezeigt, wenn sich die Maus über dem Link befindet).

Target

Das HTML-Attribut `target`. Wenn Sie hier etwas eingeben wird die Zielseite in einen anderen Fenster angezeigt.

Class

Das HTML-Attribut `class`.

Zeitsteuerung

Aktivierung TT . MM . JJJJ , -- : --

Der Link wird erst nach diesem Zeitpunkt aktiviert. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert.

Deaktivierung TT . MM . JJJJ , -- : --

Nach diesem Zeitpunkt ist der Link nicht mehr aktiv. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Notizen > Notizen

Speichern | Abbrechen

Abb. 2: Erstellung eines Menüs

Beschriftung
Der Name des Menüeintrages.

Icon
Das Icon für den Link. Verwendet werden können alle Namen aus dem FontAwesome-Icon-Set.

Anzeigen auf
Gibt an, wo das Menü erscheint. Der Link kann sowohl auf der *App-Startseite?*, *Eingabeformular?*, *iframe-Seite? Anzeige-Templates*, *eigener Übersichtsseite?* (nur App-Panels und User-Panels) und *Handy-App* erscheinen.

Position in der Menüleiste
Hier kann, wenn ein Untermenü angelegt werden soll, mit *Übergeordnetes Menü* ein Menü ausgewählt werden. Mit *Reihenfolge* kann die Reihenfolge bestimmt werden, in der Menüs der gleichen Ebene sortiert werden. Je grösser die Zahl, desto weiter hinten erscheint der Menüpunkt.

Link-Ziel
Hier kann aus verschiedenen Zielen ausgewählt werden, auf die ein Link zeigen kann. Bei den Zielen

App-Startseite

Eingabeformular (öffentlich)

Eingabeformular (einebettet)

Daten-Management

Auswertung

Import/Export

Arbeitsaufgaben

Konfiguration: Eingabe

Konfiguration: Arbeitsaufgaben

Konfiguration: Daten

Konfiguration: Berechtigungen

Konfiguration: Erweitert

Anzeige-Template

Daten-Auswertung

können Sie die LivingApp auswählen, deren Unterseite Sie verlinken wollen. Wählen Sie keine LivingApp aus, wird zur entsprechenden Unterseite der aktuellen LivingApp verlinkt.

Eigener Link
Hier kann eine beliebige URL angegeben werden. Diese URL kann absolut (z.B. `https://www.example.de/...`) oder relativ (z.B. `/feedback.htm`) sein.

Kein Link
Muss angewählt werden, wenn ein Menüeintrag erstellt werden soll, an dem ein Untermenü hängt.

Link-Attribute

Title
Der Text den Sie hier eingeben, wird angezeigt, wenn sich die Maus über dem Link befindet.

Target
Wenn Sie hier etwas eingeben wird die Zielseite in einem anderen Fenster angezeigt.

Sie können entweder einen selbstgewählten Namen verwenden, dann werden alle Links, die diesen Namen als `target` verwenden, im selben Fenster/Tab mit diesem Namen geöffnet.

Wenn Sie hingegen `_blank` verwenden, wird der Link immer in einem neuen Fenster bzw. Tab geöffnet.

Wenn Sie bei *Link-Ziel/Typ Kein Link* ausgewählt haben wird dieser Wert ignoriert.

Class

Der eingegebene Wert wird als zusätzlicher Inhalt für das `class`-Attribut des Links verwendet, um z.B. das Aussehen des Links zu verändern. Im Normalfall müssen Sie hier nichts eingeben.

Ein möglicher Anwendungsfall wäre, wenn Sie die Links in einem Anzeige-Template verwenden wollen, und für bestimmte Links das Aussehen anpassen wollen.

Zeitsteuerung

Hier kann festgelegt werden in welchem Zeitraum das Menü angezeigt werden soll. Mit *Aktivierung* legt man den Zeitpunkt fest, ab dem der Link angezeigt werden soll. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert. Mit *Deaktivierung* kann angegeben werden ab wann der Link nicht mehr angezeigt werden soll. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

13.2 Anwendungsbeispiel

Um die Funktionsweise der Links zu veranschaulichen wird im Folgenden wieder auf das Anwendungsbeispiel aus Kapitel 1 zurückgegriffen.

13.2.1 Link zu einem Anzeige-Template

Es soll ein Link als Menüpunkt in der Detailmaske erstellt werden, der zur Teilnehmerliste führt, die mittels eines *Anzeige-Templates* angelegt wurde.

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und klicken anschließend in der linken Menüleiste auf *App-Menüs* und dann auf *Neuen Link oder neues Menü zur Menüleiste hinzufügen*.

Im nun geöffneten Fenster werden folgende Konfigurationen vorgenommen. Zuerst wird das übergeordnete Menü `Links` angelegt.

Dabei wird insbesondere bei *Link-Ziel* der Typ *Kein Link* ausgewählt. Klicken sie auf *Speichern* und gehen anschließend wieder in die Übersicht Ihrer App-Menüs.

Hier ist nun der neue Datensatz *Links* zu sehen, bei dem der Button *Neuer Menüpunkt* angeboten wird.

Nach dem Klick auf *Neuer Menüpunkt* sehen sie, dass im Abschnitt *Position in der Menüleiste* das Feld *Übergeordnetes Menü* bereits mit dem Wert *Links* vorbelegt ist.

Füllen Sie nun die Felder *Identifizierer* und *Beschriftung*. In unserem Beispiel soll der Link „Teilnehmerliste“ heißen.

Wählen Sie danach im Abschnitt *Link-Ziel* bei *Typ* den Wert `Anzeige-Template` aus und klicken danach bei *Anzeige-Template* auf *Auswählen*. Wählen Sie nun das entsprechende Anzeige-Template aus und klicken auf *Übernehmen*. Weitere Konfigurationen werden hierfür nicht benötigt. Klicken Sie auf *Speichern* um die Konfiguration abzuschließen.

Der Link als Menüpunkt erscheint, sobald Sie die App-Startseite anklicken.

Möchten Sie noch weitere Links unter dem Menüpunkt *Links* erfassen, klicken Sie wieder auf *Neuer Menüpunkt* und fahren fort, wie oben beschrieben. Bei *Reihenfolge* können Sie einstellen, in welcher Reihenfolge die Einträge im Untermenü angezeigt werden.

App-Menüs 1/4

+ Hinzufügen

Speichern Abbrechen

* Identifizierer: links
Eine eindeutige Kennung für diesen Menüeintrag. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Beschriftung: Links

Icon:
Das Icon für den Link

Anzeigen auf

App-Startseite? ☑
Eingabeformular? ☐
iframe-Seite? ☐
Anzeige-Templates? ☐
eigener Übersichtsseite? ☐
Handy-App? ☐

Position in der Menüleiste

Übergeordnetes Menü: (Nichts ausgewählt)

Reihenfolge:
Nach dieser Zahl sortiert werden Menüs in der Menü-Leiste bzw. Menü-Einträge in Untermenüs angezeigt.

Link-Ziel

* Typ: Kein Link

Link-Attribute

Title:
Das HTML-Attribut `title`. Der Text den Sie hier eingeben, wird beim Link als Tooltip angezeigt (d.h. er wird angezeigt, wenn sich die Maus über dem Link befindet).

Target:
Das HTML-Attribut `target`. Wenn Sie hier etwas eingeben wird die Zielseite in einen anderen Fenster angezeigt.

Class:
Das HTML-Attribut `class`.

Zeitsteuerung

Aktivierung: TT . MM . JJJJ , -- : --
Der Link wird erst nach diesem Zeitpunkt aktiviert. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert.

Deaktivierung: TT . MM . JJJJ , -- : --
Nach diesem Zeitpunkt ist der Link nicht mehr aktiv. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Notizen > Notizen

Speichern Abbrechen

Abb. 3: Übergeordnetes Menü ‚Links‘ erstellen

App-Menüs

#	Beschriftung		Anzeige	Reihenfolge	Link-Ziel
1	+ Neuer Menüpunkt	Links	App-Startseite	9	∅

+ Neuen Link oder neues Menü zur Menüleiste hinzufügen PDF ··· CSV ···

In „App-Menüs“ können Sie Menüs und Menüeinträge anlegen, die das Navigationsmenü auf der Detailseite Ihrer LivingApp erweitern.

Mehr Informationen dazu finden Sie in der Dokumentation zu App-Menüs.

Abb. 4: Button ‚Neuer Menüpunkt‘

App-Menüs Teilnehmerliste Reihenfolge: 10

+ Hinzufügen

Speichern Abbrechen Hilfe

* Identifizierer teilnehmerliste

Eine eindeutige Kennung für diesen Menüeintrag. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Beschriftung Teilnehmerliste

Icon

Das Icon für den Link

Anzeigen auf

App-Startseite? ☑

Eingabeformular? ☐

iframe-Seite? ☐

eigener Übersichtsseite? ☐

Position in der Menüleiste

Übergeordnetes Menü Links

Reihenfolge 10

Nach dieser Zahl sortiert werden Menüs in der Menü-Leiste bzw. Menü-Einträge in Untermenüs angezeigt.

Link-Ziel

* Typ Anzeige-Template

Anzeige-Template Anmeldung Computerkurse → teilnehmerliste / teilnehmerliste Entfernen Auswählen

Das Anzeige-Auswertung, das das Ziel der Link ist.

Request-Parameter

Zusätzliche Parameter für den HTTP-Request auf das Anzeige-Template.

Link-Attribute

Title

Das HTML-Attribut title. Der Text den Sie hier eingeben, wird beim Link als Tooltip angezeigt (d.h. er wird angezeigt, wenn sich die Maus über dem Link befindet).

Target

Das HTML-Attribut target. Wenn Sie hier etwas eingeben wird die Zielseite in einen anderen Fenster angezeigt.

Class

Das HTML-Attribut class.

Zeitsteuerung

Aktivierung TT . MM . JJJJ , -- : --

Der Link wird erst nach diesem Zeitpunkt aktiviert. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert.

Deaktivierung TT . MM . JJJJ , -- : --

Nach diesem Zeitpunkt ist der Link nicht mehr aktiv. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Notizen > Notizen

Speichern Abbrechen Hilfe

Abb. 5: Link-Ziel auswählen

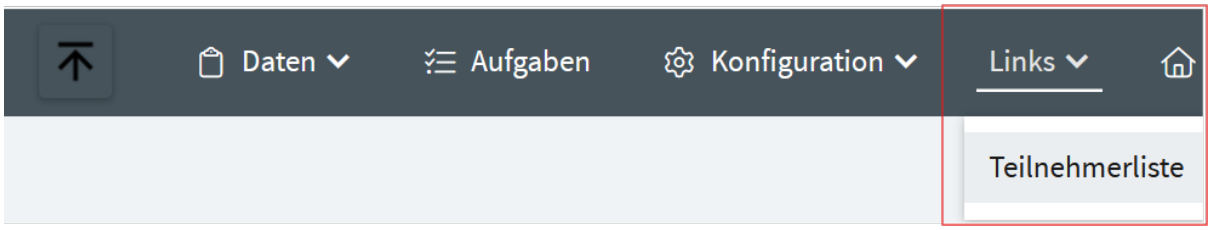

Abb. 6: Link als Menüpunkt in der Detailmaske

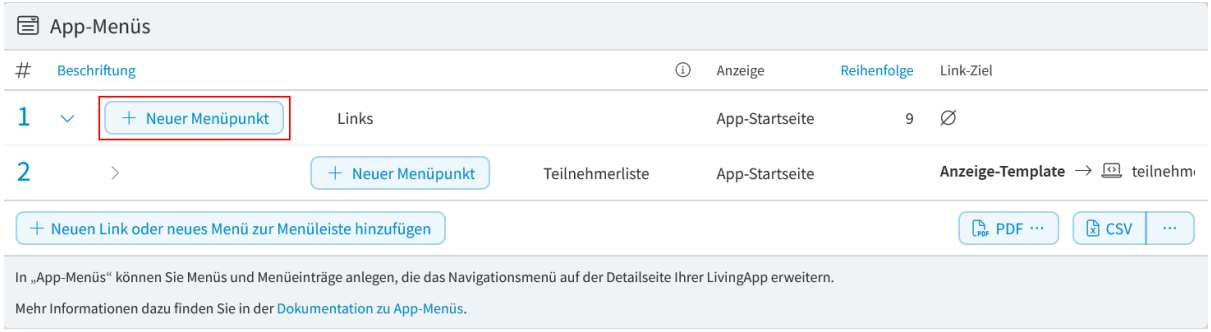

Abb. 7: Weiteren Link unter Menüpunkt ‚Links‘ erfassen

KAPITEL 14

App-Panels

Die erweiterte Funktion *App-Panels* ermöglicht es Ihnen auf verschiedenen Seiten Kacheln mit z.B. Hinweistexten oder Links zu platzieren. Haben Sie unter *Anzeige-Templates* → *Template-Vorlagen* eine eigene *Übersichtsseite* erstellt, werden Ihre *App-Panels* auch da angezeigt.

Sollten Sie Links auf der App-Übersichtsseite anlegen wollen, lesen Sie bitten bei *Benutzer-Panels* nach.

14.1 Erstellung von App-Panels

Wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *App-Panels* und dann auf *Neues Panel hinzufügen*.

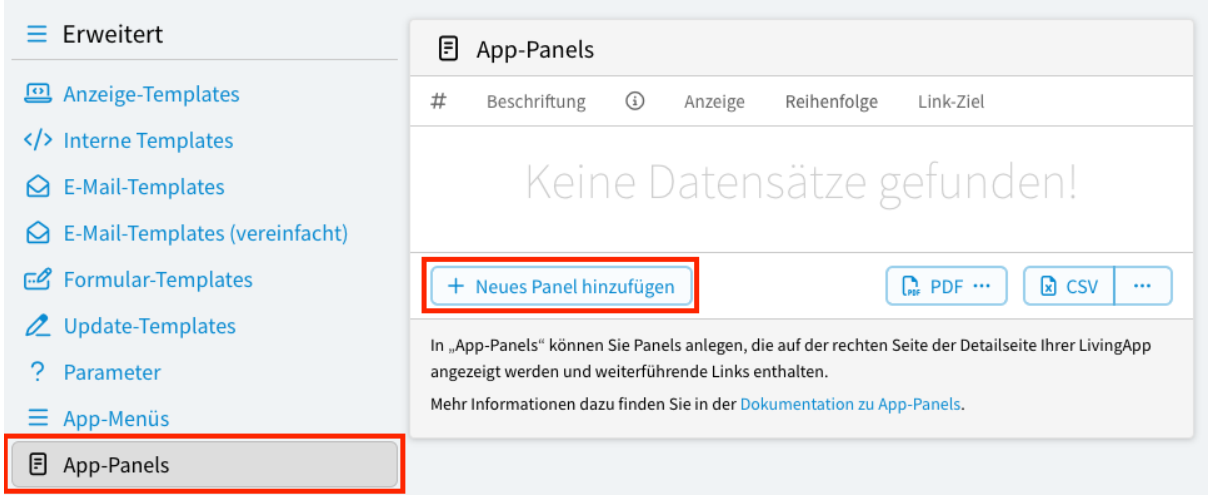

Abb. 1: Hinzufügen eines App-Panels

Im nun geöffneten Fenster können folgende Konfigurationen vorgenommen werden:

Identifizierer
: Der eindeutige Name des Links. Der Name darf nur Buchstaben, Ziffern und „_“ beinhalten.

Beschriftung
: Der Text im Kopfbereich des Panels.

App-Panels

+ Hinzufügen

✓ Speichern ↺ Abbrechen

* Identifizierer

Eine eindeutige Kennung für dieses Panel. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Beschriftung

Icon

Das Icon für den Link

Darstellung & Hintergrund

* Darstellung ◉ Normal ○ Karte

„Karte" benötigt mehr vertikalen Platz als „Normal", zeigt aber den Hintergrund größer an.

Hintergrund ◉ (Nichts ausgewählt) ○ Einzelne Farbe ○ Linearer Farbverlauf ○ Kreisförmiger Farbverlauf ○ Bild

Anzeigen auf

App-Startseite? ☐

Eingabeformular? ☐

iframe-Seite? ☐

Anzeige-Templates? ☐

eigener Übersichtsseite? ☐

Positionierung im Panel

Übergeordnetes Panel (Nichts ausgewählt)

Reihenfolge

Nach dieser Zahl sortiert werden die Panels angezeigt.

Link-Ziel

* Typ (Nichts ausgewählt)

Link-Attribute

Title

Das HTML-Attribut `title`. Der Text den Sie hier eingeben, wird beim Link als Tooltip angezeigt (d.h. er wird angezeigt, wenn sich die Maus über dem Link befindet).

Target

Das HTML-Attribut `target`. Wenn Sie hier etwas eingeben wird die Zielseite in einen anderen Fenster angezeigt.

Class

Das HTML-Attribut `class`.

Zeitsteuerung

Aktivierung TT . MM . JJJJ , -- : --

Der Link wird erst nach diesem Zeitpunkt aktiviert. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert.

Deaktivierung TT . MM . JJJJ , -- : --

Nach diesem Zeitpunkt ist der Link nicht mehr aktiv. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Beschreibung

Beschreibung

Datei Bearbeiten Ansicht Einfügen Format Werkzeuge Tabelle

Absatz **B** *I*

p 0 Wörter tiny

Anzeige-Template ⇄ Auswählen

Wenn Sie hier ein Anzeige-Template auswählen wird die Beschreibung von diesem Anzeige-Template geholt.

Notizen > Notizen

✓ Speichern ↺ Abbrechen

Abb. 2: App-Panel erstellen

Icon
Das Icon für den Link. Verwendet werden können alle Namen aus dem FontAwesome-Icon-Set.

Darstellung
Normal bzw. *Karte* kann hier ausgewählt werden. *Karte* führt zu einer etwas höheren Darstellung mit mehr Hintergrund.

Hintergrund
Hier besteht die Möglichkeit zwischen *Einzelner Farbe*, *Linearer Farbverlauf*, *Kreisförmiger Farbverlauf* und *Bild* zu wählen. Je nach Auswahl werden anschliessend Felder für *Textfarbe*, *Hintergrundfarbe oben*, *Hintergrundfarbe unten* oder ein *Bild* angeboten.

Anzeigen auf
Gibt an, wo das Panel erscheint. Der Link kann sowohl auf der *App-Startseite?*, dem *Eingabeformular?*, der *iframe-Seite?* bei *Anzeige-Templates`und auf :guilabel:`eigener Übersichtsseite?* erscheinen.

Positionierung im Panel
Hier kann, wenn Panels ineinander verschachtelt werden sollen, mit *Übergeordnetes Panel* ein Panel ausgewählt werden. Mit *Reihenfolge* kann die Reihenfolge bestimmt werden, in der Panels der gleichen Ebene sortiert werden. Je grösser die Zahl, desto weiter hinten erscheint die Kachel.

Position im Gitter auf der Übersichts-Seite
Erscheint nur, wenn bei *eigener Übersichtsseite?* das Häkchen gesetzt wurde.

Hier können Sie die *Zeile* und *Spalte*, wo das Panel auf der Übersichts-Seite angezeigt werden soll und die *Breite* und *Höhe* des Panels festlegen.

Link-Ziel
Hier kann aus verschiedenen Zielen ausgewählt werden, auf die ein Link zeigen kann. Bei den Zielen

App-Startseite,

Eingabeformular (öffentlich),

Eingabeformular (einebettet),

Daten-Management,

Auswertung,

Import/Export,

Arbeitsaufgaben,

Konfiguration: Eingabe,

Konfiguration: Arbeitsaufgaben,

Konfiguration: Daten,

Konfiguration: Berechtigungen,

Konfiguration: Erweitert,

Anzeige-Template und

Daten-Auswertung

können Sie die LivingApp auswählen, deren Unterseite Sie verlinken wollen. Wählen Sie keine LivingApp aus, wird zur entsprechenden Unterseite der aktuellen LivingApp verlinkt.

Eigener Link
Hier kann eine beliebige URL angegeben werden. Diese URL kann absolut (z.B. `https://www.example.de/...`) oder relativ (z.B. `/feedback.htm`) sein.

Kein Link
Muss angewählt werden, wenn ein Menüeintrag erstellt werden soll, an dem ein Untermenü hängt.

Link-Attribute

Title
Der Text den Sie hier eingeben, wird angezeigt, wenn sich die Maus über dem Link befindet.

Target
Wenn Sie hier etwas eingeben wird die Zielseite in einem anderen Fenster angezeigt.

Sie können entweder einen selbstgewählten Namen verwenden, dann werden alle Links, die diesen Namen als `target` verwenden, im selben Fenster/Tab mit diesem Namen geöffnet.

Wenn Sie hingegen `_blank` verwenden, wird der Link immer in einem neuen Fenster bzw. Tab geöffnet.

Wenn Sie bei *Link-Ziel/Typ Kein Link* ausgewählt haben wird dieser Wert ignoriert.

Class
Der eingegebene Wert wird als zusätzlicher Inhalt für das `class`-Attribut des Links verwendet, um z.B. das Aussehen des Links zu verändern. Im Normalfall müssen Sie hier nichts eingeben.

Ein möglicher Anwendungsfall wäre, wenn Sie die Links in einem Anzeige-Template verwenden wollen, und für bestimmte Links das Aussehen anpassen wollen.

Zeitsteuerung
Hier kann festgelegt werden in welchem Zeitraum das Menü angezeigt werden soll. Mit *Aktivierung* legt man den Zeitpunkt fest, ab dem der Link angezeigt werden soll. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert. Mit *Deaktivierung* kann angegeben werden ab wann der Link nicht mehr angezeigt werden soll. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Beschreibung
Hier kann ein Beschreibungstext angegeben werden, der in dem Panel angezeigt wird. Alternativ kann auch ein *Anzeige-Template* ausgewählt werden, das den Beschreibungstext ausgibt. Das vom Anzeige-Template geladene HTML-Fragment ersetzt die Beschreibung, die am Anfang angezeigt wird.

14.2 Anwendungsbeispiel

Um die Funktionsweise der App-Panels zu veranschaulichen wird im Folgenden wieder auf das Anwendungsbeispiel aus Kapitel 1 zurückgegriffen.

Es soll ein App-Panel erstellt werden, das zum Anlegen eines Textelementes verwendet wird. Dieses Textelement soll Hinweise des Veranstalters enthalten und bei der Eingabe einer Anmeldung rechts vom Eingabeformular angezeigt werden.

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und klicken anschließend in der linken Menüleiste auf *App-Panels* und dann auf *Neues Panel hinzufügen* um ein neues App-Panel zu erstellen.

Im nun geöffneten Fenster werden folgende Konfigurationen vorgenommen.

Klicken Sie auf *Speichern*. Das Ergebnis sehen Sie dann unter *Daten* → *Eingabe* rechts vom Eingabe-Formular.

App-Panels

+ Hinzufügen

✓ Speichern ↺ Abbrechen

* Identifizierer: kursinhalte
Eine eindeutige Kennung für dieses Panel. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Beschriftung: Kursinhalte

Icon
Das Icon für den Link

Darstellung & Hintergrund

* Darstellung: ◉ Normal ○ Karte
„Karte" benötigt mehr vertikalen Platz als „Normal", zeigt aber den Hintergrund größer an.

Hintergrund: ◉ (Nichts ausgewählt) ○ Einzelne Farbe ○ Linearer Farbverlauf ○ Kreisförmiger Farbverlauf ○ Bild

Anzeigen auf

App-Startseite? ☐
Eingabeformular? ☑
iframe-Seite? ☐
Anzeige-Templates? ☐
eigener Übersichtsseite? ☐

Positionierung im Panel

Übergeordnetes Panel: (Nichts ausgewählt)

Reihenfolge
Nach dieser Zahl sortiert werden die Panels angezeigt.

Link-Ziel

* Typ: Kein Link

Link-Attribute

Title
Das HTML-Attribut `title`. Der Text den Sie hier eingeben, wird beim Link als Tooltip angezeigt (d.h. er wird angezeigt, wenn sich die Maus über dem Link befindet).

Target
Das HTML-Attribut `target`. Wenn Sie hier etwas eingeben wird die Zielseite in einen anderen Fenster angezeigt.

Class
Das HTML-Attribut `class`.

Zeitsteuerung

Aktivierung: TT . MM . JJJJ , -- : --
Der Link wird erst nach diesem Zeitpunkt aktiviert. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert.

Deaktivierung: TT . MM . JJJJ , -- : --
Nach diesem Zeitpunkt ist der Link nicht mehr aktiv. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Beschreibung

Beschreibung
Datei Bearbeiten Ansicht Einfügen Format Werkzeuge Tabelle
Absatz

Ausführliche Informationen finden Sie auf unserer Webseite https://www.mustermann.de.

p 8 Wörter tiny

Anzeige-Template: ⇄ Auswählen
Wenn Sie hier ein Anzeige-Template auswählen wird die Beschreibung von diesem Anzeige-Template geholt.

Notizen > Notizen

✓ Speichern ↺ Abbrechen

Abb. 3: Erstellen eines App-Panels als Textelement

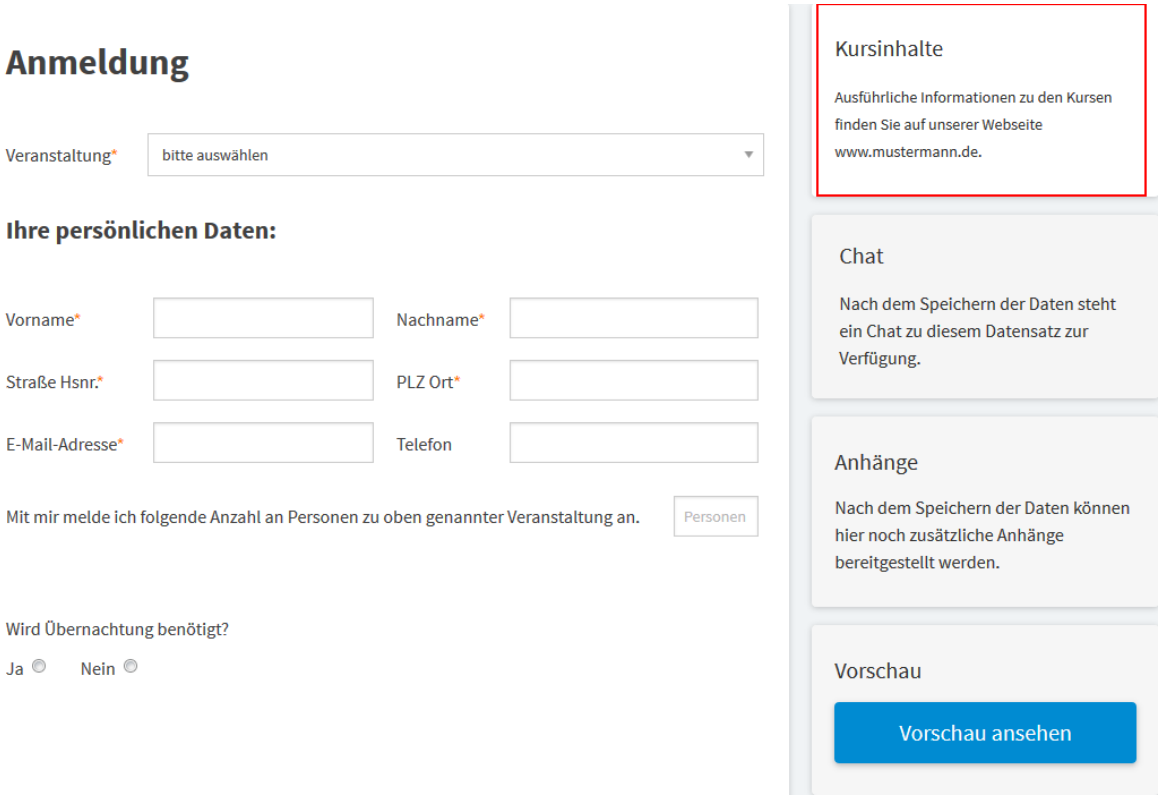

Abb. 4: Links als Textelement in der Eingabemaske

KAPITEL 15

Datenmanagement

Unter *Konfiguration* → *Erweitert - Datenmanagement* können verschiedene Einstellungen für die Datenliste vorgenommen werden.

15.1 Farben

Mit *Farben* kann konfiguriert werden, dass Datensätze in der Liste unter bestimmten Bedingungen eingefärbt werden. So können z. B. Datensätze, die bestimmte Werte besitzen, optisch hervorgehoben werden. Im Beispiel sollen die Datensätze von Teilnehmern, die noch nicht bezahlt haben, rot eingefärbt werden.

15.1.1 Erstellung

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Datenmanagement* → *Farben* aus. Klicken Sie auf *Hinzufügen* um eine neue Einfärbung einzustellen. (Siehe *Einfärbung hinzufügen* und *Einfärbung erstellen*)

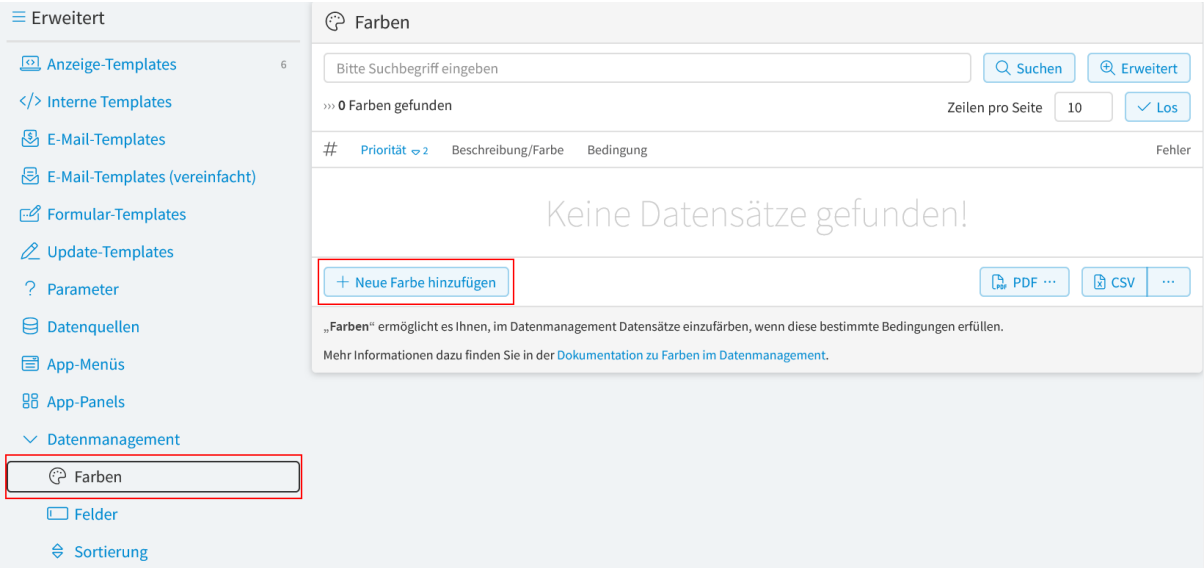

Abb. 1: Einfärbung hinzufügen

Farben | Diese Teilnehmer haben noch nicht bezahlt. | Priorität: 1

Bearbeiten

Speichern | Wiederherstellen | Hilfe

* Priorität 1

Die Bedingungen werden in absteigender Priorität getestet. Die erste Bedingung, die erfüllt ist, bestimmt die Farbe.

darkred4	darkred3	darkred2	darkred	red	lightred
darkgreen4	darkgreen3	darkgreen2	darkgreen	green	lightgreen
darkblue4	darkblue3	darkblue2	darkblue	blue	lightblue
darkyellow4	darkyellow3	darkyellow2	darkyellow	yellow	lightyellow
darkpurple4	darkpurple3	darkpurple2	darkpurple	purple	lightpurple
darkgray4	darkgray3	darkgray2	darkgray	gray	lightgray

* Farbe Wählen Sie hier die Farbe, mit der die Zeile im Datenmanagement eingefärbt werden soll, wenn die angegebene Bedingung erfüllt ist.

* Beschreibung Diese Teilnehmer haben noch nicht bezahlt.

Diese Beschreibung wird im Datenmanagement in der Legende unter der Liste angezeigt.

* Bedingung `r.v_bezahlt is None or not r.v_bezahlt`

Diese Bedingung muß ein Datensatz erfüllen, um die zugehörige Farbe zugewiesen zu bekommen.

Kompilierte Bedingung `r.v_bezahlt is None or not r.v_bezahlt`

Historie

Angelegt von Carina Oetter | Geändert von Sabine Voigt

Angelegt am 23.03.2017 15:09:04 | Geändert am 21.09.2017 11:37:26

Speichern | Wiederherstellen | Hilfe

Abb. 2: Einfärbung erstellen

Priorität
Die Bedingungen werden in absteigender Priorität getestet. Die erste Bedingung, die erfüllt ist, bestimmt die Farbe.

Farbe
Wählen Sie hier die Farbe, mit der die Zeile im Datenmanagement eingefärbt werden soll, wenn die angegebene Bedingung erfüllt ist.

Beschreibung
Diese Beschreibung wird im Datenmanagement in der Legende unter der Liste angezeigt.

Bedingung
Diese Bedingung muss ein Datensatz erfüllen, um die zugehörige Farbe zugewiesen zu bekommen.

15.1.2 Formulieren der Bedingung

Für die Formulierung wird *vSQL* verwendet.

Um Werte von Feldern Ihres Formulars zu prüfen, werden die Identifizierer der Felder benötigt. Diese finden Sie in der linken Menüleiste unter *Felder*. Auf die Werte kann dann zugegriffen werden mit `r.v_identifier` .

Sie haben unterschiedliche Prüfmöglichkeiten. Die Datentypen, Operationen, Attribute und Methoden, die Ihnen hier zur Verfügung stehen, finden Sie im Kapitel *vSQL* bzw. wenn Sie das ? unterhalb des Feldes anklicken.

Um beispielsweise ein Text-Feld mit dem Identifizierer `name` auf einen bestimmten Wert zu prüfen, benötigen Sie folgende Bedingung: `r.v_name == "LivingApps"`.

Ist die Bedingung nicht vom Typ `BOOL` (d.h. liefert `True` oder `False`), wird sie mittels der vSQL-Funktion `bool` konvertiert.

Sobald Sie auf *Speichern* gedrückt haben, erscheint unterhalb des Bedingungsfelds die kompilierte Bedingung und falls beim Kompilieren ein Fehler aufgetreten ist, auch dieser Fehler.

15.2 Sortierung

Bei *Sortierung* kann festgelegt werden, wonach die Datensätze in der Datenliste sortiert sein sollen. Wird nichts angegeben, so werden die Datensätze absteigend nach dem Erzeugungsdatum sortiert.

15.2.1 Erstellung

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Datenmanagement* → *Sortierung* aus. Klicken Sie auf *Hinzufügen* um eine *neue Sortierung* anzulegen.

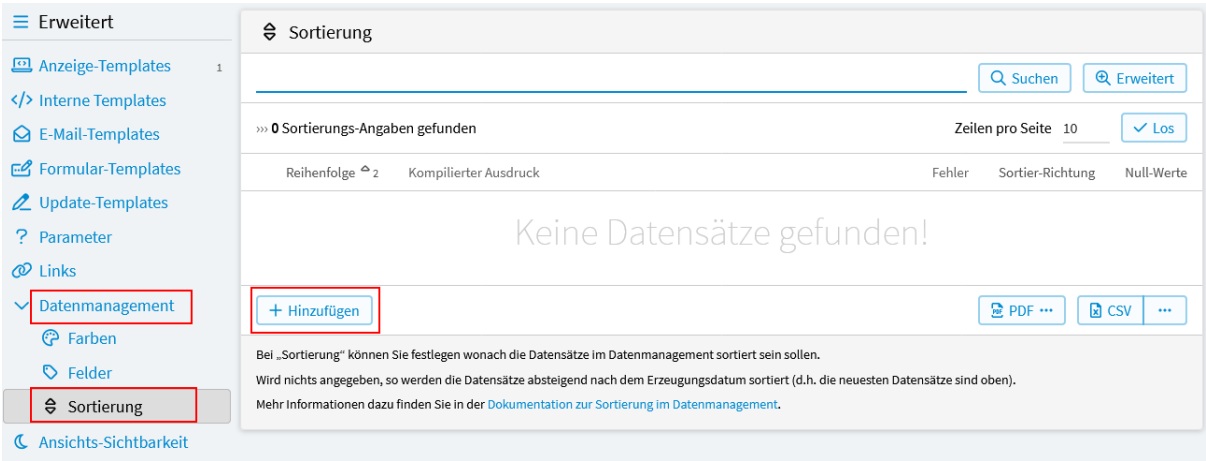

Abb. 3: Sortierung hinzufügen

Im nun geöffneten Fenster können folgende *Einstellungen* vorgenommen werden.

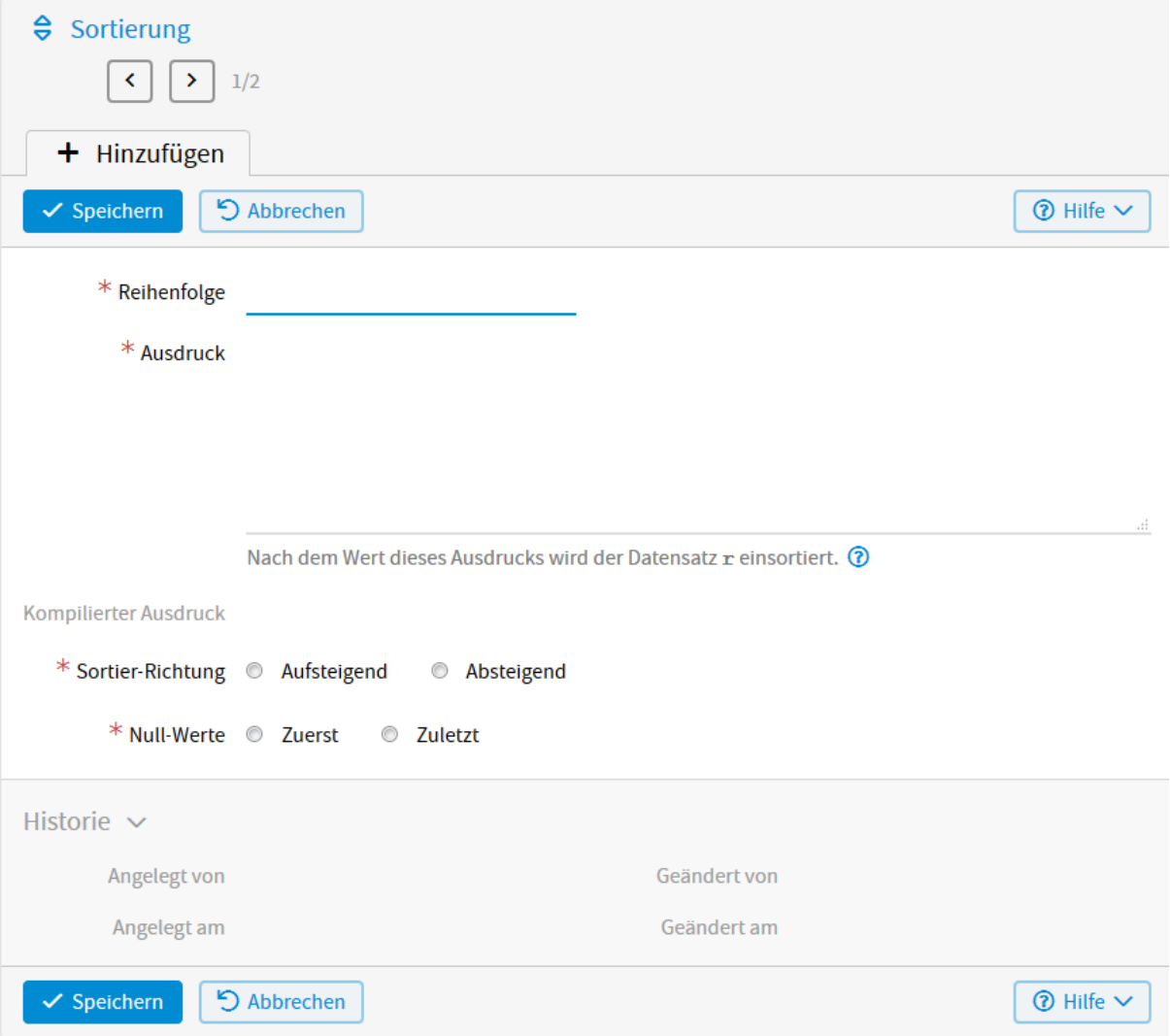

Abb. 4: Sortierung erstellen

Reihenfolge
Legt fest, in welcher Reihenfolge die Sortierungen (wenn mehrere konfiguriert sind) abgehandelt werden.

Ausdruck
Nach dem Wert dieses Ausdrucks werden die Datensätze sortiert. Für die Formulierung wird *vSQL* verwendet. Um die Werte der Felder auszugeben, werden die eindeutigen *Identifizierer der Felder* benötigt. Diese finden Sie in der linken Menüleiste unter *Felder*. Auf die Werte kann dann zugegriffen werden mit: `r.v_identifier`.

Sortier-Richtung
Aufsteigend (A-Z oder 1, 2, 3 oder altes Datum zuerst) *Absteigend* (Z-A oder 3, 2, 1 oder neues Datum zuerst)

Null-Werte
Können zuerst oder zuletzt angezeigt werden.

15.2.2 Anwendungsbeispiel

Um die Funktionsweise der Sortierung zu veranschaulichen, wird im Folgenden auf das Anwendungsbeispiel aus Kapitel 1 zurückgegriffen.

Die eingegangenen Anmeldungen sollen in der Datenliste sortiert nach Veranstaltung und nach dem Nachnamen der Teilnehmer angezeigt werden. Dafür müssen zwei Sortierungen angelegt werden.

Zuerst soll *nach Veranstaltung* sortiert werden.

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Datenmanagement* → *Sortierung* aus und klicken auf *Hinzufügen*.

Für die *1. Sortierung* werden folgende Konfigurationen vorgenommen.

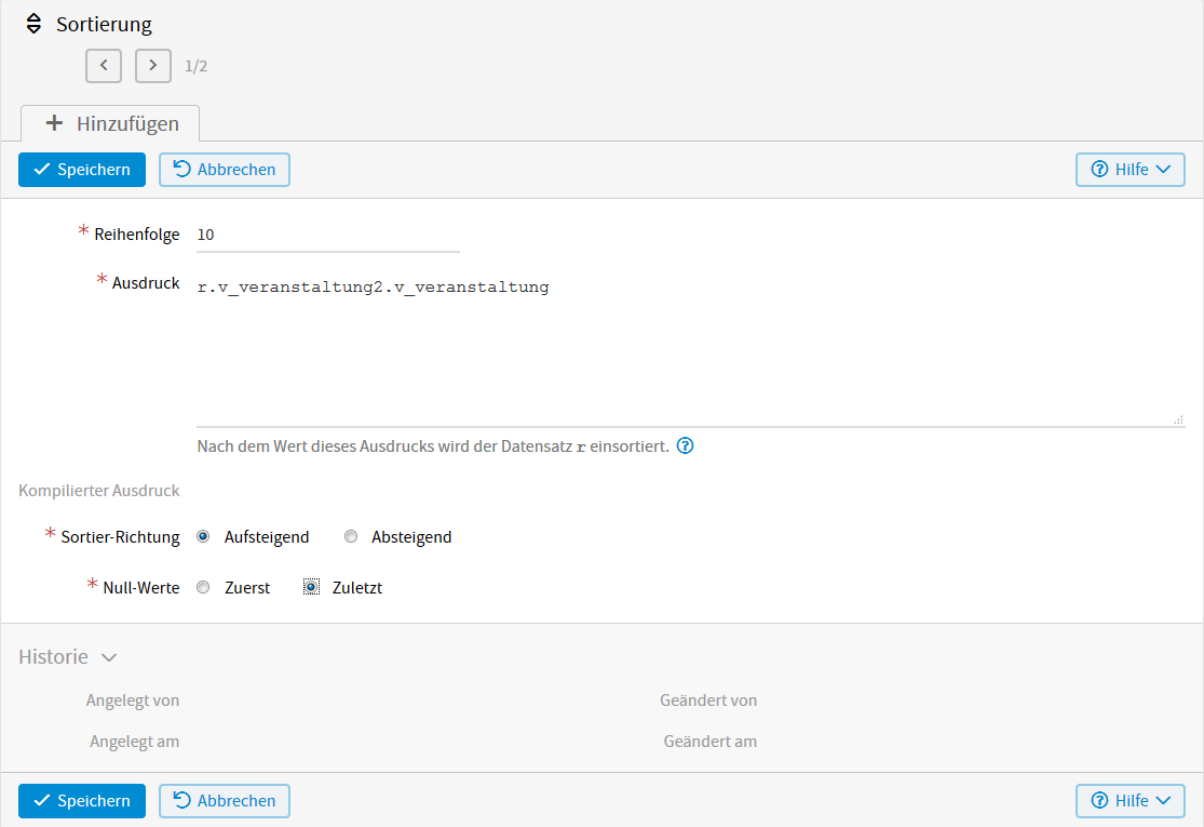

Abb. 5: Sortierung 1 erstellen

Im Anwendungsbeispiel lautet der Ausdruck `r.v_veranstaltung2.v_veranstaltung`. Dabei wird über das Feld `veranstaltung2` (applookup/select) der App „Anmeldung“ auf das Feld `veranstaltung` der verknüpften App „Veranstaltungen“ zugegriffen.

Bei der zweiten Sortierung soll zusätzlich nach dem *Nachnamen* sortiert werden.

Klicken Sie auf *Hinzufügen* um die *2. Sortierung* anzulegen.

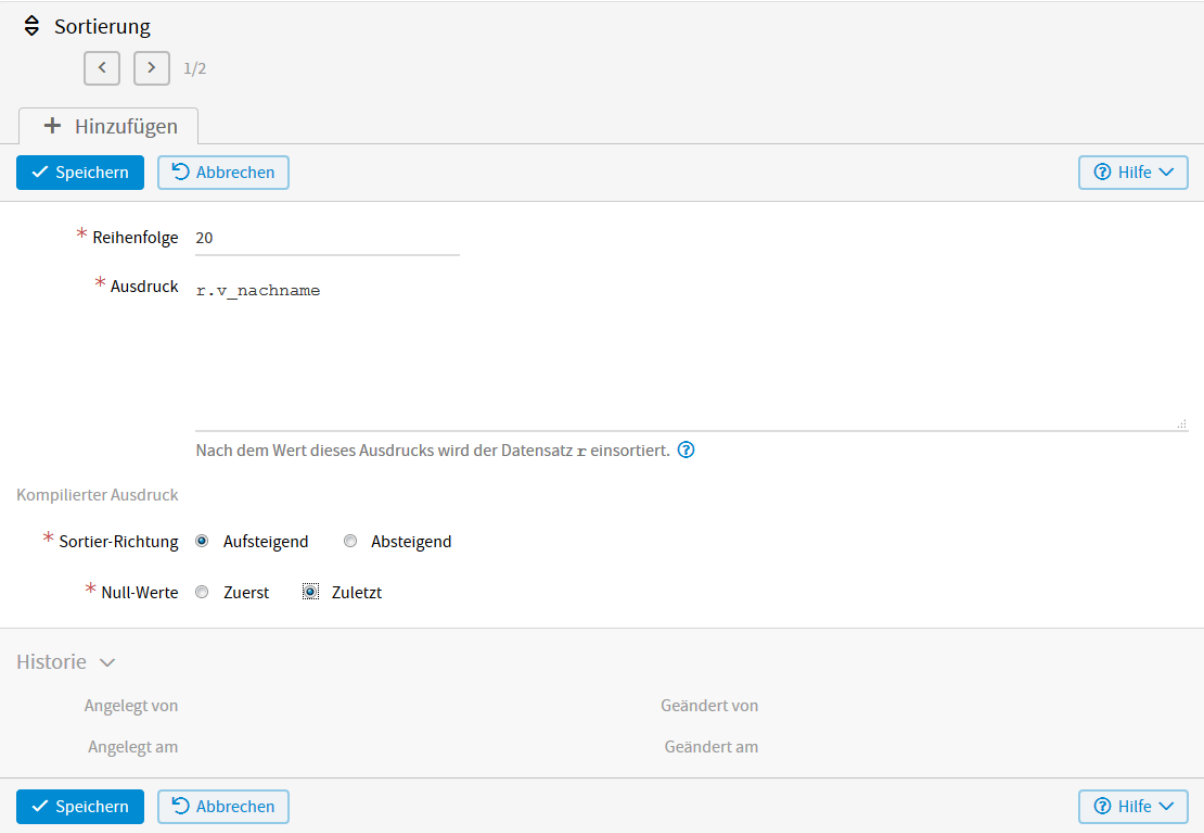

Abb. 6: Sortierung 2 erstellen

So sieht dann das *Ergebnis der Sortierung* aus. Ohne Sortierung würden die Datensätze absteigend nach Erzeugungsdatum angezeigt werden.

Anmeldung

Suchen | Erweitert

››› 7 Anmeldung gefunden

Zeilen pro Seite 7 | Los

		Veranstaltung	Nachname	Vorname	E-Mail-Adresse	Anzahl Personen	Aktionen
☐	1	Excel Aufbaukurs	Mustermann	Max	max.mustermann@livinglogic.de	3	
☐	2	Excel Aufbaukurs	Sonnenschein	Klara	klara.sonnenschein@livinglogic.de	3	
☐	3	Excel Grundkurs	Beispiel	Anne	anne.beispiel@livinglogic.de	4	
☐	4	Excel Grundkurs	Mustermann	Max	max.mustermann@livinglogic.de	1	
☐	5	Word Aufbaukurs	Muster	Hans	hans.muster@livinglogic.de	2	
☐	6	Word Aufbaukurs	Mustermann	Max	max.mustermann@livinglogic.de	1	
☐	7	Word Grundkurs	Mustermann	Max	max.mustermann@livinglogic.de	1	

Hinzufügen | E-Mail... | Zusatzaufgaben... | Aktion durchführen... | Daten ändern... | CSV | PDF

Abb. 7: Ergebnis der Sortierung

KAPITEL 16

Formularvarianten

Unter *Formularvarianten* können Sie einen Gültigkeits-Zeitraum für Formularvarianten festlegen, sowie konfigurieren, was vor und nach diesem Zeitraum statt des Formulars angezeigt werden soll. Unterschiedliche Formularvarianten können erstellt werden unter *Konfiguration* → *Eingabe*.

16.1 Erstellung

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Formularvarianten*. Klicken Sie auf die *jeweilige Formularvariante*, um dafür *Sichtbarkeits-Konfigurationen* festzulegen.

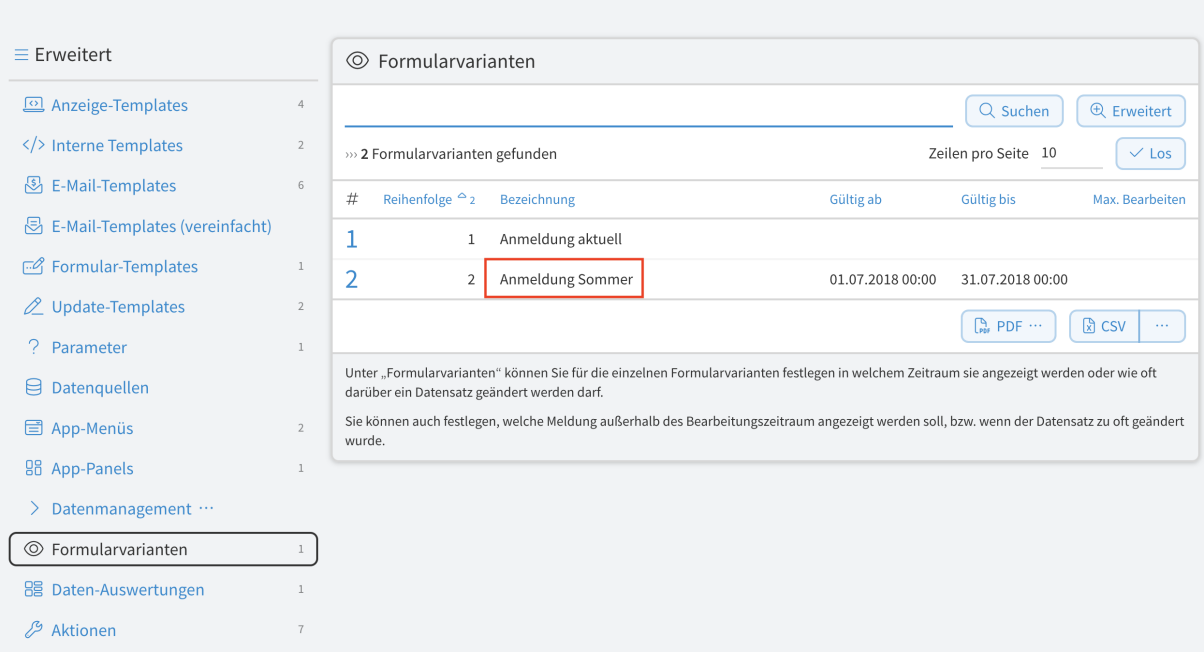

Abb. 1: Formularvariante auswählen

Gültig ab
Wählen Sie hier ein Datum, ab wann diese Formularvariante sichtbar sein soll.

Formularvarianten 2/2

Bearbeiten

Hilfe

Reihenfolge 2

Bezeichnung Anmeldung Sommer

Gültig ab 01.07.2018, 00:00

Ist hier ein Zeitpunkt eingetragen, so wird vor diesem Zeitpunkt nicht das Formular angezeigt sondern das was unter „Anzeigetemplate vorher“ eingetragen wurde.

Gültig bis 31.07.2018, 00:00

Ist hier ein Zeitpunkt eingetragen, so wird vor nach Zeitpunkt nicht das Formular angezeigt sondern das was unter „Anzeigetemplate nachher“ eingetragen wurde.

Anzeigetemplate vorher

```
<p>Sehr geehrte(r) Teilnehmer(in),</p>
<p>die Anmeldung zu unseren Sommerspecials ist erst ab 01. Juli möglich.</p>
<p>Vielen Dank für Ihr Verständis.</p>
<p>Mit freundlichen Grüßen</p>
<p>Ihr Veranstaltungsteam</p>
```

Ist unter „Gültig ab“ ein Zeitpunkt eingetragen, so wird vor diesem Zeitpunkt dieses Template ausgegeben.

Anzeigetemplate nachher

```
<p>Sehr geehrte(r) Teilnehmer(in),</p>
<p>die Anmeldung zu unseren Sommerspecials war nur bis zum 31. Juli möglich.</p>
<p>Vielen Dank für Ihr Verständis.</p>
<p>Mit freundlichen Grüßen</p>
<p>Ihr Veranstaltungsteam</p>
```

Ist unter „Gültig bis“ ein Zeitpunkt eingetragen, so wird nach diesem Zeitpunkt dieses Template ausgegeben.

Max. Anzahl Bearbeiten

Anzeigetemplate max. Anzahl überschritten

Abb. 2: Sichtbarkeits-Konfiguration erstellen

Gültig bis
Wählen Sie hier ein Datum, bis wann diese Formularvarianten sichtbar sein soll.

Anzeigetemplate vorher
Für den Zeitraum davor wird der hier eingegebene Text als Inhalt der Formularseite angezeigt. Zur Gestaltung können Sie HTML-Elemente nutzen. Informationen dazu finden Sie unter https://wiki.selfhtml.org/.

Anzeigetemplate nachher
Für den Zeitraum danach wird der hier eingegebene Text als Inhalt der Formularseite angezeigt. Zur Gestaltung können Sie HTML-Elemente nutzen. Informationen dazu finden Sie unter https://wiki.selfhtml.org/.

Max. Anzahl bearbeiten
Hier kann eingestellt werden, wie oft ein Datensatz in dieser Formularvariante bearbeitet werden kann. Das ist sinnvoll z. B. bei Umfragen. Nach der Teilnahme an der Umfrage werden die Daten sofort weiterverarbeitet und sollen nicht mehr geändert werden dürfen.

Anzeigetemplate max. Anzahl überschritten
Geben Sie hier den Text ein der angezeigt werden soll, wenn die max. Anzahl überschritten wurde. Zur Gestaltung können Sie HTML-Elemente nutzen. Informationen dazu finden Sie unter https://wiki.selfhtml.org/.

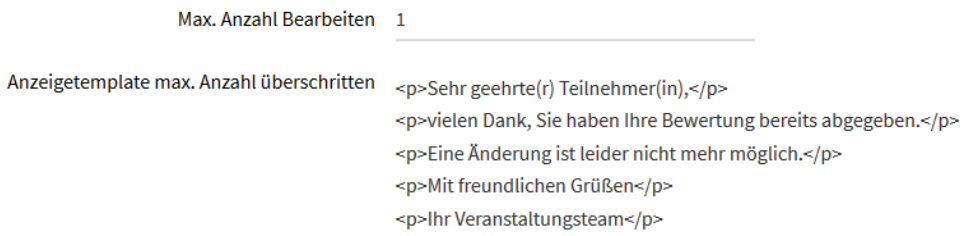

Abb. 3: Beispiel max. Anzahl

KAPITEL 17

Daten-Auswertungen

Daten-Auswertungen ermöglicht es Ihnen, im Datenmanagement zusätzliche Auswertungen für Ihre Daten einzurichten.

Welche Felder in diesen Auswertungen angezeigt werden, lässt sich flexibel konfigurieren. Weiterhin ist es möglich, die Daten zu aggregieren (d.h. die Datensätze anhand bestimmter Kriterien zu gruppieren).

Eine Daten-Auswertung bei Applookup-Feldern mit Mehrfachauswahl ist nicht möglich.

17.1 Erstellung von Daten-Auswertungen

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Daten-Auswertungen*. Klicken Sie auf *Hinzufügen* um eine neue Auswertung zu erstellen.

Sie gelangen dann in die erste Einstellungsmaske. (Siehe *Daten-Auswertung erstellen Schritt 1*)

Beschriftung
: Unter dieser erscheint die Auswertung in der linken Navigation in der Datenliste.

Icon
: Hier können Sie ein Icon für die Auswertung auswählen. Verwendet werden können alle Namen aus dem FontAwesome-Icon-Set (http://fontawesome.io/)

Reihenfolge
: Geben Sie hier an, an welcher Stelle die Auswertung in der linken Navigation der Datenliste erscheint.

Anzahl in Navigation
: Durch das Setzen des Hakens wird neben dem Icon die Anzahl der Datensätze in dieser Auswertung angezeigt.

Wenn Sie diese Felder gefüllt haben, klicken Sie auf *Speichern*. Anschließend sehen Sie neben *Bearbeiten* die Tabs *Haupt-Felder* und *Filterbedingungen*. (Siehe *Daten-Auswertung erstellen Schritt 2*)

17.1.1 Haupt-Felder

Klicken Sie zunächst auf *Haupt-Felder* und anschließend auf *Hinzufügen*, um festzulegen, welche Felder in die Auswertung aufgenommen werden sollen. (Siehe *Hauptfelder hinzufügen* und *Hauptfelder erstellen*) Die in dieser Auswertung konfigurierten Felder erscheinen in der neuen Auswertung als Spalten.

Feld
: Wählen Sie hier das Feld aus, wonach ausgewertet werden soll.

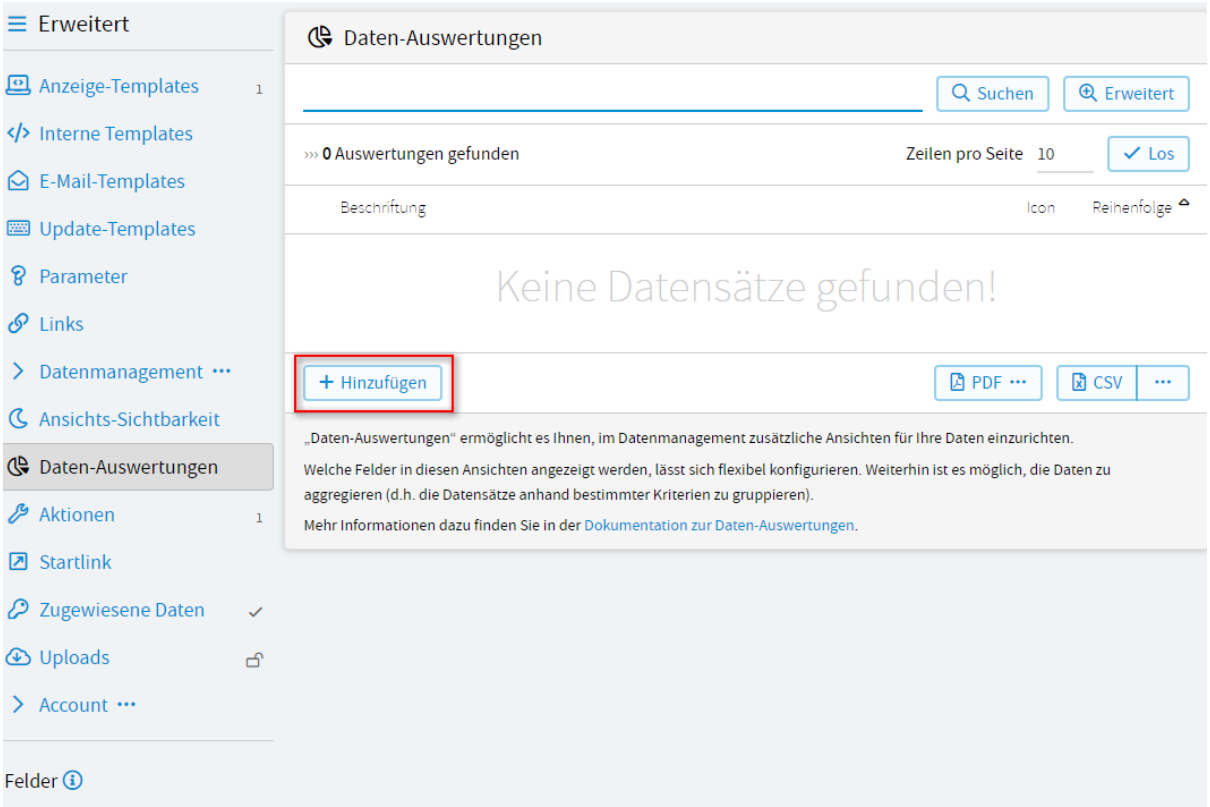

Abb. 1: Daten-Auswertung hinzufügen

Eigene Beschriftung
Diese ersetzt die Beschriftung des Feldes in der Spalte der Auswertung.

Reihenfolge
Legt bei mehreren Hauptfeldern die Reihenfolge in der Übersicht fest.

Ansicht
Hier können alle Eigenschaften der Standard-Datenliste ausgewählt werden.

Listenansicht
Das Feld soll in der Listenansicht der Auswertung mit angezeigt werden.

Detailansicht
Das Feld soll in der Detailansicht der Auswertung mit angezeigt werden.

Suche
Das Feld bekommt ein eigenes Suchfeld in der Listenansicht der Auswertung.

Volltextsuche
Das Feld wird in die Volltextsuche der Listenansicht der Auswertung aufgenommen.

Expertensuche
Das Feld wird in die erweiterte Suche der Listenansicht der Auswertung aufgenommen.

Pfad
Das Feld wird im Pfad immer angezeigt, wenn aus der Listenansicht heraus Untermasken ausgewählt werden. Zum Beispiel: *Anzeige* oder *Einzeldatensätze*

Panel
Das Feld wird im Pfad immer angezeigt, wenn aus der Listenansicht heraus Untermasken ausgewählt werden. Zum Beispiel: *Anzeige* oder *Einzeldatensätze*. Das Feld steht dann rechts vom Pfad.

Soll es in Ihrer Auswertung keine Suche geben, oder kein Pfad angezeigt werden, können die entsprechenden Häkchen weggelassen werden.

Daten-Auswertungen

+ Hinzufügen

Speichern | Abbrechen | Hilfe

* Beschriftung

Die Beschriftung unter der Ihre Ansicht im Daten-Management erscheint.

Icon

Das Icon für diese Ansicht in der Navigation im Datenmanagement.

* Reihenfolge

Ansichten werden nach diesem Feld sortiert im Daten-Management angezeigt

Icon-Vorschau

Anzahl in Navigation?

Soll die Anzahl der Datensätze für diese Ansicht in der Navigation angezeigt werden?

Historie

Angelegt von | Geändert von

Angelegt am | Geändert am

Speichern | Abbrechen | Hilfe

Abb. 2: Daten-Auswertung erstellen Schritt 1

Daten-Auswertungen | Gruppiert nach Veranstaltung | Icon: arrow-circle-up

Bearbeiten | Haupt-Felder | Filterbedingungen

+ Hinzufügen | Kopieren | – Löschen | Ansicht erzeugen | Hilfe

* Beschriftung: Gruppiert nach Veranstaltung

Die Beschriftung unter der Ihre Ansicht im Daten-Management erscheint.

Icon: arrow-circle-up

Das Icon für diese Ansicht in der Navigation im Datenmanagement.

* Reihenfolge: 10

Ansichten werden nach diesem Feld sortiert im Daten-Management angezeigt

Icon-Vorschau

Anzahl in Navigation?

Soll die Anzahl der Datensätze für diese Ansicht in der Navigation angezeigt werden?

Abb. 3: Daten-Auswertung erstellen Schritt 2

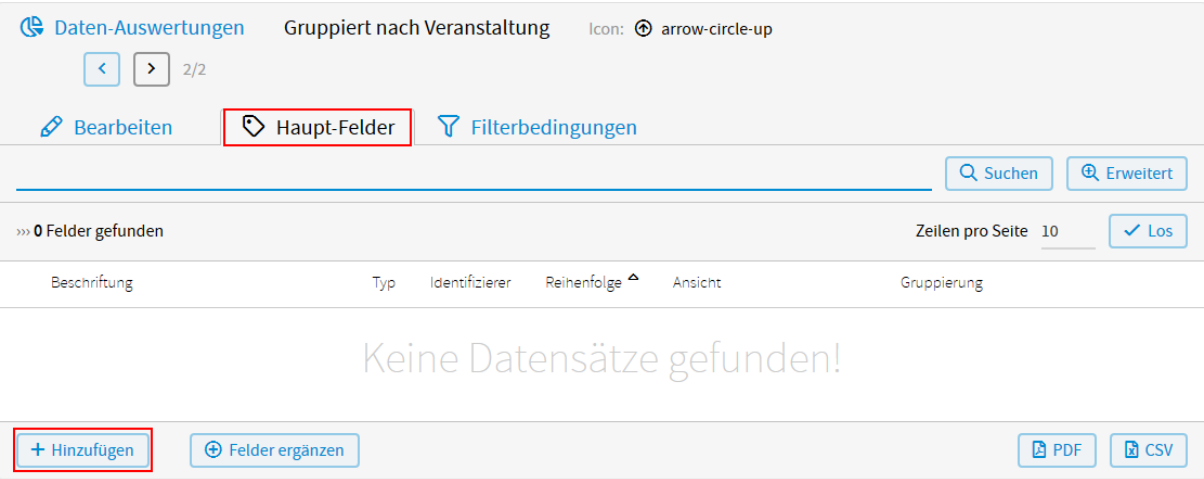

Abb. 4: Hauptfelder hinzufügen

Gruppierung

Sobald ein Feld ausgewählt wurde, erscheint noch die Auswahl *Gruppierung*. Hier kann eingestellt werden, ob und wie das Feld gruppiert werden soll. Die Auswahl ist abhängig vom Feldtyp.

Gruppiert

Einfaches Gruppieren; mehrfach vorkommende Einträge werden zusammengefasst.

Gruppiert nach Klang

Einträge, die gleich klingen, werden zusammengefasst (z. B. Mayer und Meier). Sinnvoll für Adressdoubletten-Suche.

Gruppiert nach Beginn

Einträge, bei denen die ersten x Zeichen übereinstimmen, werden zusammengefasst. Z. B. Postleitzahlengebiet bei einer PLZ.

Gruppiert nach Ende

Einträge, bei denen die letzten x Zeichen übereinstimmen, werden zusammengefasst. Z. B. die letzten Zeichen einer E-Mail-Adresse.

Gruppiert nach Tag

Einträge, die vom gleichen Tag sind, werden zusammengefasst. Betrifft nur Datumsfelder.

Gruppiert nach Monat

Einträge, die vom gleichen Monat sind, werden zusammengefasst. Betrifft nur Datumsfelder.

Gruppiert nach Jahr

Einträge, die vom gleichen Jahr sind, werden zusammengefasst. Betrifft nur Datumsfelder.

Minimum

Zeigt den kleinsten Wert der betroffenen Datensätze an. Betrifft nur Zahlenfelder.

Maximum

Zeigt den größten Wert der betroffenen Datensätze an. Betrifft nur Zahlenfelder.

Durchschnitt

Zeigt den Durchschnitt der betroffenen Datensätze an. Betrifft nur Zahlenfelder.

Summe

Zeigt die Summe der betroffenen Datensätze an. Betrifft nur Zahlenfelder.

Nach Anklicken von *Gruppiert nach Beginn* oder *Gruppiert nach Ende* kann darunter noch ein entsprechender Wert eingegeben werden. Nach Auswahl der Felder klicken Sie auf *Speichern* und anschließend wieder auf *Haupt-Felder*. Sie können nun zusätzliche Haupt-Felder oder, wenn nötig, *Filterbedingungen hinzufügen*. Für letzteres klicken Sie auf *Filterbedingungen* und anschließend auf *Hinzufügen*.

Daten-Auswertungen Gruppiert nach Veranstaltung Icon: arrow-circle-up

Haupt-Felder

1/2

+ Hinzufügen

Speichern Abbrechen

Feld

* Feld
- Veranstaltung applookup/select veranstaltung2
- Vorname string/text vorname
- Nachname string/text nachname
- Straße Hsnr. string/text strasse_und_hsnr2
- PLZ Ort string/text plz_ort
- E-Mail-Adresse string/email e_mail_adresse
- Telefon string/tel telefon
- Anzahl Personen number anzahl_personen2
- Wird Übernachtung benötigt? lookup/radio wird_uebernachtung_benoetigt
- Einzelzimmer number einzelzimmer
- Euro / Nacht number euro_nacht
- Doppelzimmer number mehrbettzimmer
- Euro / Nacht number euro_nacht2
- Anrede lookup/radio anrede
- Zuständiger Mitarbeiter string/email zustaendiger_mitarbeiter
- abgemeldet bool abgemeldet
- Zimmer sind gebucht bool zimmer_sind_gebucht
- Teilnahmegebühr number teilnahmegebuehr
- bezahlt bool bezahlt
- bezahlt am date/date bezahlt_am
- Anzahl in Gruppe int
- Erstellungsdatum date
- Änderungsdatum date

Anzeige

Eigene Beschriftung

* Reihenfolge

Ansicht
- Listenansicht
- Detailansicht
- Suche
- Volltextsuche
- Expertensuche
- Pfad
- Panel

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen

Abb. 5: Hauptfelder erstellen

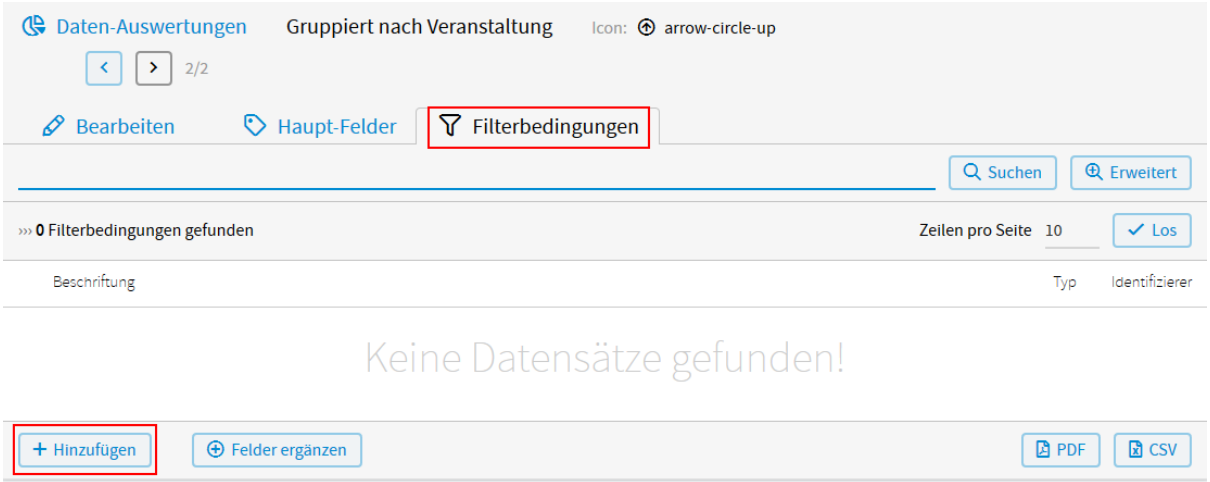

Abb. 6: Filterbedingungen hinzufügen

17.1.2 Filterbedingungen

Mit den *Filterbedingungen* kann eingeschränkt werden, welche Datensätze in der Auswertung angezeigt werden.

Feld
: Geben Sie an, nach welchem Feld gefiltert werden soll.

Filter
: Geben Sie an, welchen Filter Sie nutzen wollen. Die Auswahl ist abhängig vom Feldtyp.

Wert
: Wird zusammen mit dem Filter als Filterbedingung genutzt.

Sind alle gewünschten Einstellungen getroffen und gespeichert, muss im Tab *Bearbeiten* der Button *Ansicht erzeugen* gedrückt werden.

WICHTIG: Nach jeder Änderung muss die Ansicht immer wieder neu erzeugt werden.

Wenn beim Aufrufen der Auswertung ein „Unvorhersehbarer Fehler“ erscheint, ist eine Einstellung ungültig.

17.2 Anwendungsbeispiel

Um die Funktionsweise der Sortierung zu veranschaulichen, wird im Folgenden auf das Anwendungsbeispiel aus Kapitel 1 zurückgegriffen.

Es soll eine Daten-Auswertung erstellt werden, in der nach Veranstaltung gruppiert, die jeweilige Anzahl der Anmeldungen angezeigt wird.

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Daten-Auswertungen*. Klicken Sie auf *Hinzufügen* um eine neue Auswertung zu erstellen.

Im *Schritt 1* werden hier folgende Konfigurationen vorgenommen.

Nach dem Speichern klicken Sie auf *Hauptfelder* und *Hinzufügen* um das *1. Hauptfeld* anzulegen.

Da nach Veranstaltung gruppiert werden soll, wird zunächst bei *Feld Veranstaltung* ausgewählt. Bei *Eigene Beschriftung* wird nichts eingegeben, da der Feldname `Veranstaltung` passend ist. Die *Reihenfolge* soll `10` lauten und bei *Ansicht* werden alle Häkchen gesetzt, bis auf *Panel*. Bei *Gruppierung* wird *gruppiert* ausgewählt.

Nach dem Speichern kann durch Klicken auf *Hinzufügen* das *2. Hauptfeld erstellt* werden.

Hier soll die Anzahl der Anmeldungen zu den jeweiligen Veranstaltungen angezeigt werden. Dafür muss bei *Feld Anzahl Personen* ausgewählt werden. Bei *Eigene Beschriftung* wird `Anmeldungen` eingegeben. Das ist dann der Name der Spalte in der Auswertung. Die *Reihenfolge* soll `20` sein und bei *Ansicht* werden alle Häkchen gesetzt, bis auf *Pfad*. Bei *Gruppierung* wird *Summe* gewählt. Zum Schluss müssen alle Eingaben gespeichert werden. *Filterbedingungen* werden in diesem Beispiel nicht benötigt.

Daten-Auswertungen Gruppiert nach Veranstaltung Icon: arrow-circle-up

Filterbedingungen

+ Hinzufügen

Speichern Abbrechen

Feld				Filterbedingung	
* Feld	bezahlt	bool	bezahlt	* Filter	Ist leer
	bezahlt am	date/date	bezahlt_am		Ist nicht leer
	Teilnahmegebühr	number	teilnahmegebuehr		
	abgemeldet	bool	abgemeldet		
	Zimmer sind gebucht	bool	zimmer_sind_gebucht		
	Veranstaltung	applookup/select	veranstaltung		
	Vorname	string/text	vorname		
	Nachname	string/text	nachname		
	Str./Hs.Nr.	string/text	str_hs_nr2		
	PLZ/Ort	string/text	plz_ort		
	E-Mail-Adresse	string/email	e_mail_adresse		
	Telefon	string/tel	telefon		
	Anzahl Personen	number	anzahl		
	Wird Übernachtung benötigt?	lookup/radio	wird_uebernachtung_benoetigt		
	Einzelzimmer	number	einzelzimmer		
	Euro/Nacht	number	euro_nacht		
	Doppelzimmer	number	doppelzimmer		
	Euro/Nacht	string/text	euro_nacht2		
	Anzahl in Gruppe	int			
	Erstellungsdatum	date			
	Änderungsdatum	date			

Historie

Angelegt von	Geändert von
Angelegt am	Geändert am

Speichern Abbrechen

Abb. 7: Filterbedingungen erstellen

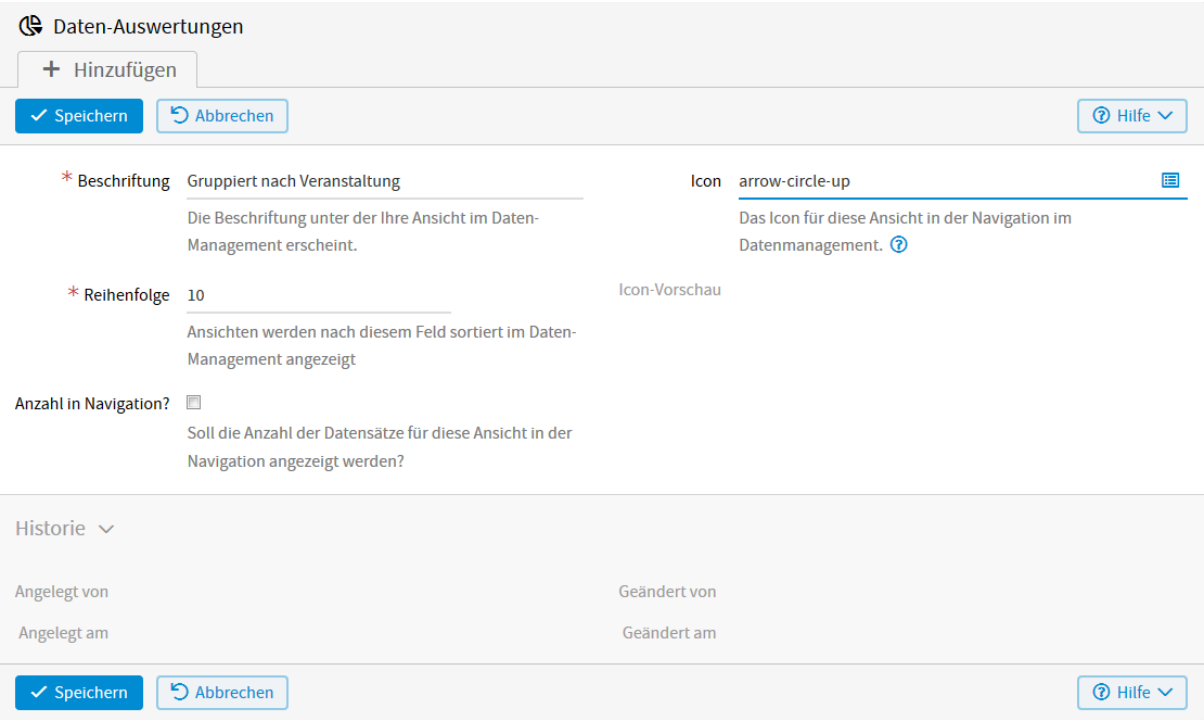

Abb. 8: Anwendungsbeispiel - Schritt 1

Sind alle Hauptfelder erstellt und gespeichert, muss im Tab *Bearbeiten* der Button *Ansicht erzeugen* angeklickt werden.

WICHTIG: Nach jeder Änderung der Daten-Auswertung muss die Ansicht immer wieder neu erzeugt werden.

Das *Ergebnis der Daten-Auswertung* kann jetzt unter *Daten* → *Liste* im Datenmanagement aufgerufen werden.

Durch Klicken auf einen Datensatz der Auswertung, erscheinen die einzelnen Datensätze, die hinter dieser Gruppierung stehen.

Daten-Auswertungen Gruppiert nach Veranstaltung Icon: arrow-circle-up

Haupt-Felder

1/2

+ Hinzufügen

Speichern Abbrechen

Feld

* Feld
- Veranstaltung applookup/select veranstaltung2
- Vorname string/text vorname
- Nachname string/text nachname
- Straße Hsnr. string/text strasse_und_hsnr2
- PLZ Ort string/text plz_ort
- E-Mail-Adresse string/email e_mail_adresse
- Telefon string/tel telefon
- Anzahl Personen number anzahl_personen2
- Wird Übernachtung benötigt? lookup/radio wird_uebernachtung_benoetigt
- Einzelzimmer number einzelzimmer
- Euro / Nacht number euro_nacht
- Doppelzimmer number mehrbettzimmer
- Euro / Nacht number euro_nacht2
- Anrede lookup/radio anrede
- Zuständiger Mitarbeiter string/email zustaendiger_mitarbeiter
- abgemeldet bool abgemeldet
- Zimmer sind gebucht bool zimmer_sind_gebucht
- Teilnahmegebühr number teilnahmegebuehr
- bezahlt bool bezahlt
- bezahlt am date/date bezahlt_am
- Anzahl in Gruppe int
- Erstellungsdatum date
- Änderungsdatum date

Anzeige

Eigene Beschriftung

* Reihenfolge 10

Ansicht
- [x] Listenansicht
- [x] Detailansicht
- [x] Suche
- [x] Volltextsuche
- [x] Expertensuche
- [x] Pfad
- [ ] Panel

Gruppierung

Gruppierung
- (Nichts ausgewählt)
- Gruppiert

Historie

Angelegt von

Angelegt am

Geändert von

Geändert am

Speichern Abbrechen

Abb. 9: Hauptfeld 1 erstellen

Daten-Auswertungen Gruppiert nach Veranstaltung Icon: arrow-circle-up

Haupt-Felder

1/2

+ Hinzufügen

Speichern Abbrechen

Feld

* Feld
- Veranstaltung applookup/select veranstaltung2
- Vorname string/text vorname
- Nachname string/text nachname
- Straße Hsnr. string/text strasse_und_hsnr2
- PLZ Ort string/text plz_ort
- E-Mail-Adresse string/email e_mail_adresse
- Telefon string/tel telefon
- Anzahl Personen number anzahl_personen2
- Wird Übernachtung benötigt? lookup/radio wird_uebernachtung_benoetigt
- Einzelzimmer number einzelzimmer
- Euro / Nacht number euro_nacht
- Doppelzimmer number mehrbettzimmer
- Euro / Nacht number euro_nacht2
- Anrede lookup/radio anrede
- Zuständiger Mitarbeiter string/email zustaendiger_mitarbeiter
- abgemeldet bool abgemeldet
- Zimmer sind gebucht bool zimmer_sind_gebucht
- Teilnahmegebühr number teilnahmegebuehr
- bezahlt bool bezahlt
- bezahlt am date/date bezahlt_am
- Anzahl in Gruppe int
- Erstellungsdatum date
- Änderungsdatum date

Anzeige

Eigene Beschriftung Anmeldungen

* Reihenfolge 20

Ansicht
- [x] Listenansicht
- [x] Detailansicht
- [x] Suche
- [x] Volltextsuche
- [x] Expertensuche
- [ ] Pfad
- [x] Panel

Gruppierung

Gruppierung
- (Nichts ausgewählt)
- Gruppiert
- Minimum
- Maximum
- Durchschnitt
- Summe

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen

Abb. 10: Hauptfeld 2 erstellen

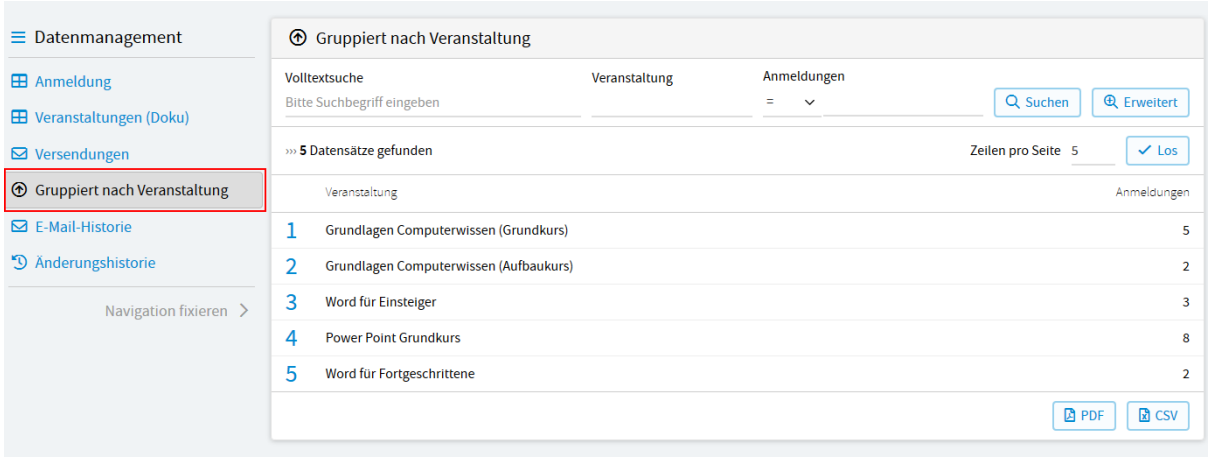

Abb. 11: Ergebnis der Daten-Auswertung

KAPITEL 18

Daten-Aktionen

Aktionen ermöglicht es Ihnen, Aktionen zu definieren, die bestimmte Datensätze ändern, neue Datensätze anlegen, Arbeitsaufgaben anlegen, oder E-Mails verschicken.

Die Aktionen können entweder automatisch (z.B. wenn ein neuer Datensatz vom Benutzer angelegt oder geändert wird) oder vom Benutzer (z. B. durch Anklicken einer URL in einer E-Mail oder eines Buttons im Datenmanagement) ausgelöst werden.

18.1 Erstellung von Aktionen

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Aktionen*. Klicken Sie auf *Hinzufügen* um eine *neue Aktion* zu erstellen.

Im nun geöffneten Fenster können folgende *Konfigurationen* vorgenommen werden.

Identifizierer
: Die eindeutige Kennung für die jeweilige Aktion. (Mit diesem wird die Aktion z. B. bei der Verwendung als Link für E-Mails angesprochen.)

Beschriftung
: Der Name der Aktion, der z. B. in der E-Mail oder in der Datenliste sichtbar ist.

Aktiv?
: Solange hier kein Häkchen gesetzt wird, kann die Aktion nicht ausgeführt werden.

Reihenfolge
: Die Aktionsbuttons werden im Datenmanagement nach dieser Zahl sortiert angezeigt.

Berechtigung für
: Hier muss ausgewählt werden, wer diese Aktion ausführen darf.

Beschreibung
: Eine Beschreibung der Funktion, oder eine Beschreibung, um mehrere Aktionen voneinander zu unterscheiden.

Icon
: Hier können Sie ein Icon für die Aktion auswählen. Verwendet werden können alle Namen aus dem FontAwesome-Icon-Set (http://fontawesome.io/).

Verwendung
: Legt fest, wo die Aktion zur Verfügung steht.

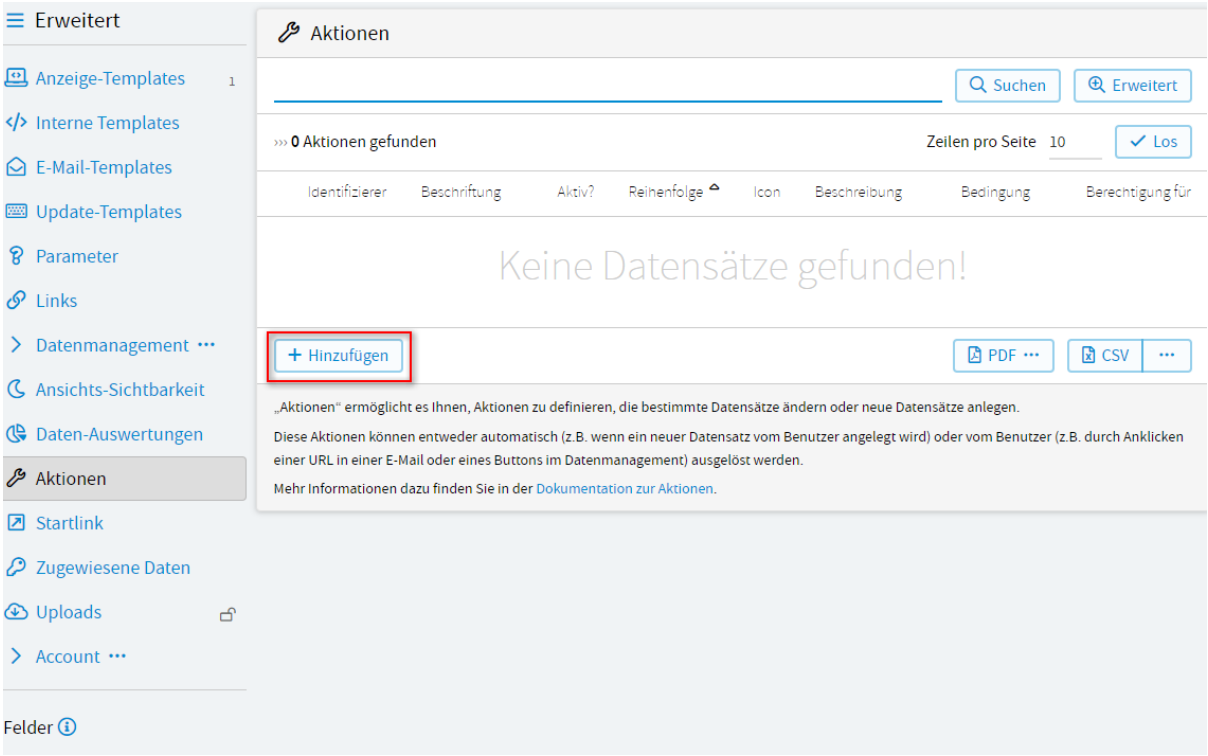

Abb. 1: Aktionen hinzufügen

Aktions-Button im Datenmanagement für mehrere Datensätze
Die Aktion kann in der Datenliste für mehrere ausgewählte Datensätze ausgeführt werden.

Aktions-Button im Datenmanagement für einzelnen Datensatz
Die Aktion kann in der Datenliste für einen einzelnen ausgewählten Datensatz ausgeführt werden.

Link für E-Mails
Die Aktion wird durch Anklicken des Links in einer E-Mail ausgeführt. (*Siehe Anwendungsbeispiel*). Wie Sie einen Link einbinden, können Sie in der Dokumentation der *E-Mail-Templates* nachlesen. Außerdem können Sie auf den Link zugreifen, wenn Sie von der Datenliste aus E-Mails verschicken (sofern ein E-Mail-Feld vorhanden ist). Klicken Sie dafür in der geöffneten E-Mail-Maske auf *Daten einfügen* und wählen den entsprechenden Aktionslink aus.

Vor dem Anzeigen des Bearbeiten-Formulars ausführen
Die Aktion wird beim Aufruf des edit-Formulars ausgeführt. Auf das jeweilige edit-Formular gelangt man über den Bearbeiten-Button des des jeweiligen Datensatzes in der Datenliste oder indem man nach dem Absenden des Formulars `/edit` vor die View-ID `?view=58b3f8b35699ce805c43dc6d` in der URL einfügt. Somit lässt sich z. B. eine Aufrufbestätigung einrichten.

Nach Änderung des Datensatzes ausführen (auch bei Import)
Die Aktion wird beim Speichern, nach dem Ändern eines bestehenden Datensatzes ausgeführt.

Nach Anlegen des Datensatzes ausführen (auch bei Import)
Die Aktion wird beim Speichern eines neuen Datensatzes ausgeführt.

Meldung
Diese Meldung erscheint nachdem die Aktion ausgeführt wurde.

Bedingung
Durch eine Filter-Bedingung kann eingeschränkt werden, auf welche Datensätze die Aktion ausgeführt wird.

Aktionen Aktiv?: Nein

+ Hinzufügen

Speichern Abbrechen Hilfe

* Identifizierer
Eindeutige Kennung für diese Aktion

* Beschriftung

Aktiv? ☐
Eine inaktive Aktion wird nicht angezeigt und kann nicht ausgeführt werden.

Reihenfolge
Die Aktions-Buttons werden im Datenmanagement nach dieser Zahl sortiert angezeigt.

* Berechtigung für alle Benutzer
Wer darf diese Aktion ausführen?

Beschreibung

Icon
Das Icon für den Aktions-Button im Datenmanagement. Verwendet werden können alle Namen aus dem FontAwesome-Icon-Set (ohne den Präfix fa-, also beispielsweise paw für 🐾).

Icon-Vorschau

Verwendung
☐ Aktions-Button im Datenmanagement für mehrere Datensätze
☐ Aktions-Button im Datenmanagement für einen einzelnen Datensatz
☐ Link für E-Mails
☐ Vor dem Anzeigen des Bearbeiten-Formulars ausführen
☐ Nach Änderung des Datensatzes ausführen (auch bei Import)
☐ Nach Anlegen des Datensatzes ausführen (auch bei Import)
Legt fest wo diese Aktion zur Verfügung stehen soll

Meldung
Meldung im Datenmanagement oder bei der Link-Ausführung wenn die Aktion erfolgreich ausgeführt wurde

Filter

Bedingung
Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit die Aktion auf dem Datensatz ausgeführt wird

Kompilierte Bedingung

Historie

Angelegt von | Geändert von
Angelegt am | Geändert am

Speichern Abbrechen Hilfe

Abb. 2: Aktionen erstellen

18.1.1 Formulieren der Bedingung

Für die Formulierung wird *vSQL* verwendet.

Um Werte von Feldern Ihres Formulars zu prüfen, werden die *Identifizierer der Felder* benötigt. Diese finden Sie in der linken Menüleiste unter *Felder*. Auf die Werte kann dann zugegriffen werden mit: `r.v_identifier`.

Sie haben unterschiedliche Prüfmöglichkeiten. Die Datentypen, Operationen, Attribute und Methoden, die Ihnen hier zur Verfügung stehen, finden Sie im Kapitel *vSQL* bzw. wenn Sie das ? unterhalb des Feldes anklicken.

Um beispielsweise ein Text-Feld mit dem Identifizierer `name` auf einen bestimmten Wert zu prüfen, benötigen Sie folgende Bedingung: `r.v_name == "LivingApps"`.

Ist die Bedingung nicht vom Typ `BOOL` (d.h. liefert `True` oder `False`), wird sie mittels der vSQL-Funktion `bool` konvertiert.

Sobald Sie auf *Speichern* gedrückt haben, erscheint unterhalb des Bedingungsfelds die kompilierte Bedingung und, wenn beim Kompilieren ein Fehler aufgetreten ist, auch dieser Fehler.

18.1.2 Anweisungen

Nach dem Speichern der ersten Maske erscheint neben *Bearbeiten* der Tab *Anweisungen*. Fügen Sie eine neue Anweisung hinzu, um einzustellen, was durch die Aktion ausgeführt werden soll. Siehe *Anweisung hinzufügen* und *Anweisung erstellen*.

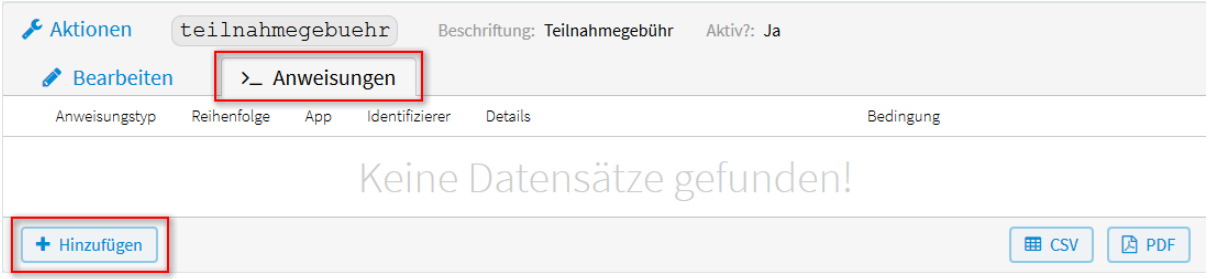

Abb. 3: Anweisungen hinzufügen

Anweisungstyp
Legt die Art der Aktion fest.

Datensatz anlegen
Ein neuer Datensatz wird angelegt, auf den die Konfigurationen angewendet werden.

Datensatz verändern
Der Datensatz, auf dem die Aktion ausgeführt wird, wird entsprechend der Konfigurationen verändert (*siehe Anwendungsbeispiel*).

Arbeitsaufgabe automatisch anlegen
Diese Funktionalität sollte nicht ohne gesonderte Einweisung verwendet werden.

Datensatz löschen
Der Datensatz, auf dem die Aktion ausgeführt wird, wird gelöscht (*siehe Anwendungsbeispiel*).

Datensatz über Formular anlegen
Auf einem bestehenden Datensatz können Sie hiermit ein Eingabeformular derselben oder einer anderen App öffnen. Die Werte des Eingabeformulars können Sie vorbelegen mit festen Werten, oder Ausdrücken, die sich auf den bestehenden Datensatz beziehen. Wird das Eingabeformular dann abgesendet, wird der neue Datensatz angelegt. (*siehe Anwendungsbeispiel*)

Datenmenge
Diese Funktionalität sollte nicht ohne gesonderte Einweisung verwendet werden.

Datensatz über Formular (Statisch)
Diese Funktionalität wird bei bestimmten Installationen automatisch angelegt. Man sollte sie nicht manuell anlegen.

Aktionen abmelden Beschriftung: Von Veranstaltung abmelden
Aktiv?: Ja Icon: minus-square-o

Anweisungen

+ Hinzufügen

Speichern Abbrechen Hilfe

* Anweisungstyp
- Datensatz anlegen
- Datensatz verändern
- Arbeitsaufgabe automatisch anlegen
- Datensatz löschen
- Datensatz über Formular anlegen
- Datenmenge
- Datensatz über Formular (Statisch)
- Onboarding mit Installation
- Arbeitsaufgaben Datensatz E-Mails
- Email-Versendung
- Einladung zu einer App
- Keyview anlegen

* Reihenfolge

Die Sortierung nach dieser Zahl bestimmt in welcher Reihenfolge die Anweisungen ausgeführt werden.

Details

Details

Filter

Bedingung

Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit diese Anweisung auf dem Datensatz ausgeführt wird.

Kompilierte Bedingung

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen Hilfe

Abb. 4: Anweisungen erstellen

Onboarding mit Installation
Diese Funktionalität sollte nicht ohne gesonderte Einweisung verwendet werden.

Arbeitsaufgaben Datensatz E-Mails
Diese Aktion legt eine Arbeitsaufgabe für eine oder mehrere Personen an, die auch als Link per E-Mail an die eingegebenen Adressen verschickt werden kann (*siehe Anwendungsbeispiel*).

Email-Versendung
Mit dieser Aktion kann man E-Mails verschicken (*siehe Anwendungsbeispiel*).

Einladung zu einer App
Mit dieser Aktion können Sie Personen zur Mitarbeit an einer oder mehreren Ihrer Apps einladen. (*siehe Anwendungsbeispiel*).

Keyview anlegen
Diese Funktionalität sollte nicht ohne gesonderte Einweisung verwendet werden.

Reihenfolge
Bei mehreren Anweisungen entscheidet die hier festgelegte Reihenfolge über die Abfolge der Ausführung.

Bedingung
Hier kann eine Filter-Bedingung eingegeben werden, die erfüllt sein muss, damit diese Anweisung auf dem Datensatz ausgeführt wird. Hierzu können Sie im Kapitel *Formulieren der Bedingung* mehr lesen.

Bei Auswahl der Felder *Datensatz anlegen* oder *Datensatz über Formular anlegen* erscheinen zwei weitere Konfigurationsfelder in dieser Maske:

App
Hier können Sie die App auswählen, in der der neue Datensatz erstellt werden soll. Wenn nichts ausgewählt wird, ist es die aktuelle App.

Identifizierer
Hier können Sie festlegen, unter welchem Identifizierer der durch diese Aktion gelieferte Datensatz für die untergeordneten Anweisungen zugänglich sein soll.

18.1.3 Felderbelegung

Nach dem Speichern der Anweisungen erscheint neben *Bearbeiten* der Tab *Felderbelegung*. Durch Klicken darauf sehen Sie alle Felder, die durch die Aktion verändert werden können. Ist es die aktuelle App, dann sehen Sie deren *Felder*. Wenn z. B. ein Datensatz in einer anderenApp angelegt werden soll, sehen Sie die Felder der anderen App.

Durch Klicken auf das entsprechende Feld, kann der *Inhalt konfiguriert* werden.

Je nachdem, um welche Aktion es sich handelt, können verschiedene Einstellungen vorgenommen werden.

Anzeige
Diese Konfiguration erscheint nur wenn *Datensatz über Formular anlegen* gewählt wurde.

Bearbeiten
Der Benutzer kann den unter Aktion generierten Wert noch verändern.

Anzeigen
Der Benutzer kann diesen Wert nicht verändern. Er wird unverändert übernommen.

Verstecken
Der Wert wird unverändert übernommen, ohne angezeigt zu werden.

Aktion

Nichts ausgewählt
Wird hier nichts ausgewählt, so wird mit dem Feld keine Aktion durchgeführt.

Feld leeren
Der Feldinhalt wird gelöscht. Das ist nur beim Ändern von Datensätzen sinnvoll.

Feld auf aktuelles Datum setzen
Diese Auswahl erscheint nur, wenn das ausgewählte Feld vom Typ `date`, `datetimeminute` oder `datetimesecond` ist.

Aktionen teilnahmegebuehr Beschriftung: Teilnahmegebühr Aktiv?: Ja

Anweisungen Datensatz verändern Reihenfolge: 10

Bearbeiten Felderbelegung

Volltextsuche Anzeige (Alle) Suchen Erweitert

››› 17 Felder gefunden Zeilen pro Seite 17 Los

	Beschriftung	Typ	Identifizierer	Aktion
1	bezahlt	bool	bezahlt	
2	bezahlt am	date/date	bezahlt_am	
3	Teilnahmegebühr	number	teilnahmegebuehr	
4	abgemeldet	bool	abgemeldet	
5	Veranstaltung	lookup/select	veranstaltung2	
6	Vorname	string/text	vorname	
7	Nachname	string/text	nachname	
8	Straße Hsnr.	string/text	strasse_und_hsnr2	
9	PLZ Ort	string/text	plz_ort	
10	E-Mail-Adresse	string/email	e_mail_adresse	
11	Telefon	string/tel	telefon	
12	Anzahl Personen	number	anzahl_personen2	
13	Wird Übernachtung benötigt?	lookup/radio	wird_uebernachtung_benoetigt	
14	Einzelzimmer	number	einzelzimmer	
15	Euro / Nacht	number	euro_nacht	
16	Doppelzimmer	number	mehrbettzimmer	
17	Euro / Nacht	number	euro_nacht2	

CSV PDF

Abb. 5: Felderbelegung

Aktionen teilnahmegebuehr Beschriftung: Teilnahmegebühr Aktiv?: Ja

Anweisungen Datensatz über Formular anlegen, Reihenfolge: 10

Felderbelegung teilnahmegebuehr Beschriftung: Teilnahmegebühr Typ: number

3/17

Bearbeiten

Hilfe

Feld

Beschriftung Teilnahmegebühr

Typ number

Identifizierer teilnahmegebuehr

Formular-Modus

Anzeige (Nichts ausgewählt) Bearbeiten Anzeigen Verstecken

Bei **Bearbeiten** kann der Benutzer den unter **Aktion** generierten Wert nochmals verändern. Bei **Anzeigen** kann der Benutzer diesen Wert nicht verändern, er wird unverändert übernommen. Bei **Verstecken** wird der Wert unverändert übernommen ohne angezeigt zu werden.

Aktion

Aktion (Nichts ausgewählt) Feld leeren Feld auf Wert setzen Wert zu Feld addieren Ausdruck

Wird hier nichts ausgewählt, so wird mit dem Feld keine Aktion durchgeführt.

Historie Angelegt von, Angelegt am, Geändert von, Geändert am

Hilfe

Abb. 6: Konfiguration Felder

Feld auf Wert setzen
Das Feld übernimmt, nachdem die Aktion ausgeführt wurde, den hier eingegebenen Wert.

Wert zu Feld addieren
Diese Auswahl erscheint nur, wenn das ausgewählte Feld vom Typ `number` ist. Der hier eingegebene Wert wird zum Wert des Feldes addiert.

Ausdruck
Sollen die Werte des Feldes nicht gleich, sondern individuell sein, kann hier ein Ausdruck eingegeben werden. Auf die Werte kann dann zugegriffen werden mit: `r.v_identifier`. Für die Formulierung des Ausdrucks wird *vSQL* verwendet. Dafür werden die *Identifizierer der Felder* benötigt. Diese finden Sie in der linken Menüleiste unter *Felder*.

Nach dem Speichern erscheint unterhalb des Feldes *Ausdruck* der kompilierte Ausdruck. Ist ein Fehler in der Formulierung, sehen Sie diesen hier.

18.2 Anwendungsbeispiele

Um die Funktionsweise der Aktionen zu veranschaulichen, wird im Folgenden auf das Anwendungsbeispiel aus Kapitel 1 zurückgegriffen. Es werden sieben verschiedene Aktionen vorgestellt.

18.2.1 Datensatz verändern

Hierbei soll das Feld `Zuständiger Mitarbeiter`, nach dem Speichern einer neuen Anmeldung, automatisch mit der E-Mail-Adresse des zuständigen Mitarbeiters aus der App „Veranstaltungen“ gefüllt werden.

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Aktionen*. Klicken Sie auf *Hinzufügen* um eine neue Aktion zu erstellen.

Im nun geöffneten Fenster werden folgende *Konfigurationen* vorgenommen.

Nach dem Speichern erscheint neben *Bearbeiten* der Tab *Anweisungen*. Nach *Hinzufügen* einer neuen Anweisung kann im nun geöffneten Fenster eingestellt werden, was durch die Aktion ausgeführt werden soll. Siehe *Anweisungen*.

Nach dem Speichern der Anweisungen erscheint neben *Bearbeiten* der Tab *Felderbelegung*. Durch Klicken darauf sehen Sie alle *Felder der App*, deren Datensätze durch die Aktion verändert werden können. Durch Klicken auf das zu ändernde Feld *Zuständiger Mitarbeiter*, kann dessen *Inhalt konfiguriert* werden.

Im nun geöffneten Fenster wird bei *Aktion Ausdruck* ausgewählt, da kein fester Wert eingetragen sondern auf die Datensätze der App zugegriffen werden soll.

Mit dem *Ausdruck* `r.v_veranstaltung2.v_e_mail_verteiler` wird über das Feld mit Identifizierer `veranstaltung2` auf den Wert des Feldes mit dem Identifizierer `e_mail_verteiler` der verknüpften App `Veranstaltungen` zugegriffen. Die *Feldinformationen und Identifizierer der Felder* können Sie durch klicken auf *Felder* in der Menüleiste unten links, einsehen. Mit *Speichern* ist die Erstellung der Aktion abgeschlossen. Aufgerufen und ausgeführt wird diese Aktion, sobald sich ein Teilnehmer zu einer Veranstaltung angemeldet hat.

18.2.2 Link für E-Mails

Diese Aktion soll automatisch im Feld *abgemeldet* des Anmeldeformulars ein Häkchen setzen, sobald sich ein Teilnehmer über die Anmeldebestätigung per E-Mail-Link wieder abmeldet.

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Aktionen*. Klicken Sie auf *Hinzufügen* um eine neue Aktion zu erstellen.

Im nun geöffneten Fenster werden folgende *Konfigurationen* vorgenommen.

Über den *Identifizierer* `abmelden` wird die Aktion in die E-Mail eingebunden mit: `{actionlink:abmelden}` (siehe *Code HTML-E-Mail*). Die *Beschriftung* der Aktion `Von Veranstaltung abmelden` steht als *anklickbarer Link* in der HTML-E-Mail des Empfängers. Bei erfolgreicher Ausführung der Aktion wird dem Benutzer die Meldung *Sie haben sich erfolgreich abgemeldet* angezeigt.

Aktionen zustaendiger_mitarbeiter Beschriftung: E-Mail-Adresse des zuständigen Mitarbeiters. Aktiv?: Ja

7/7

Bearbeiten Anweisungen

Speichern Wiederherstellen Hilfe

* Identifizierer zustaendiger_mitarbeiter
Eindeutige Kennung für diese Aktion

* Beschriftung E-Mail-Adresse des zuständigen Mitarbeite

Aktiv? ☑
Eine inaktive Aktion wird nicht angezeigt und kann nicht ausgeführt werden.

Reihenfolge 10
Die Aktions-Buttons werden im Datenmanagement nach dieser Zahl sortiert angezeigt.

* Berechtigung für alle Benutzer
Wer darf diese Aktion ausführen?

Beschreibung Übernimmt die E-Mail-Adresse des zuständigen Mitarbeiters aus der App "Veranstaltungen".

Icon
Das Icon für den Aktions-Button im Datenmanagement.

Icon-Vorschau

Verwendung
- ☐ Aktions-Button im Datenmanagement für mehrere Datensätze
- ☐ Aktions-Button im Datenmanagement für einen einzelnen Datensatz
- ☐ Link für E-Mails
- ☐ Vor dem Anzeigen des Bearbeiten-Formulars ausführen
- ☐ Nach Änderung des Datensatzes ausführen (auch bei Import)
- ☑ Nach Anlegen des Datensatzes ausführen (auch bei Import)

Legt fest wo diese Aktion zur Verfügung stehen soll

Meldung
Meldung im Datenmanagement oder bei der Link-Ausführung wenn die Aktion erfolgreich ausgeführt wurde

Filter

Bedingung
Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit die Aktion auf dem Datensatz ausgeführt wird

Kompilierte Bedingung

Historie

Angelegt von Sabine Voigt
Angelegt am 17.05.2018 09:59:27

Geändert von Sabine Voigt
Geändert am 17.05.2018 10:27:50

Speichern Wiederherstellen Hilfe

Abb. 7: Datensatz verändern

Aktionen zustaendiger_mitarbeiter Beschriftung: E-Mail-Adresse des zuständigen Mitarbeiters. Aktiv?: Ja

Anweisungen

+ Hinzufügen

Speichern Abbrechen Hilfe

* Anweisungstyp
- ○ Datensatz anlegen
- ◉ Datensatz verändern
- ○ Arbeitsaufgabe automatisch anlegen
- ○ Datensatz löschen
- ○ Datensatz über Formular anlegen
- ○ Datenmenge
- ○ Datensatz über Formular (Statisch)
- ○ Onboarding mit Installation
- ○ Arbeitsaufgaben Datensatz E-Mails
- ○ Email-Versendung
- ○ Einladung zu einer App
- ○ Keyview anlegen

* Reihenfolge 1

Die Sortierung nach dieser Zahl bestimmt in welcher Reihenfolge die Anweisungen ausgeführt werden.

Details

Details

Filter

Bedingung

Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit diese Anweisung auf dem Datensatz ausgeführt wird.

Kompilierte Bedingung

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen Hilfe

Abb. 8: Anweisungen - Datensatz verändern

Aktionen zustaendiger_mitarbeiter Beschriftung: E-Mail-Adresse des zuständigen Mitarbeiters. Aktiv?: Ja

Anweisungen Anweisungstyp: Datensatz verändern Reihenfolge: 10

Bearbeiten Felderbelegung

Volltextsuche Anzeige (Alle) Suchen Erweitert

20 Felder gefunden « < 1 2 > » Zeilen pro Seite 10 Los

	Beschriftung	Typ	Identifizierer	Aktion
1	bezahlt	bool	bezahlt	
2	bezahlt am	date/date	bezahlt_am	
3	Teilnahmegebühr	number	teilnahmegebuehr	
4	Zimmer sind gebucht	bool	zimmer_sind_gebucht	
5	abgemeldet	bool	abgemeldet	
6	Anrede	lookup/radio	anrede	
7	Zuständiger Mitarbeiter	string/email	zustaendiger_mitarbeiter	
8	Veranstaltung	applookup/select	veranstaltung2	
9	Vorname	string/text	vorname	
10	Nachname	string/text	nachname	

CSV PDF

Abb. 9: Felderbelegung - Datensatz verändern

Aktionen zustaendiger_mitarbeiter Beschriftung: E-Mail-Adresse des zuständigen Mitarbeiters. Aktiv?: Ja

Anweisungen Anweisungstyp: Datensatz verändern Reihenfolge: 10

Felderbelegung zustaendiger_mitarbeiter Beschriftung: Zuständiger Mitarbeiter Typ: string/email

7/20

Bearbeiten

Speichern | Speichern und zur Übersicht | Wiederherstellen | Hilfe

Feld

Beschriftung Zuständiger Mitarbeiter

Typ string/email

Identifizierer zustaendiger_mitarbeiter

Aktion

Aktion (Nichts ausgewählt) Feld leeren Feld auf Wert setzen Ausdruck

Wird hier nichts ausgewählt, so wird mit dem Feld keine Aktion durchgeführt.

Ausdruck r.v_veranstaltung2.v_e_mail_verteiler

Hier können Sie einen Ausdruck eingeben, der verwendet wird um das Feld zu setzen.

Kompilierter Ausdruck r.v_veranstaltung2.v_e_mail_verteiler

Historie

Angelegt von Sabine Voigt Geändert von Sabine Voigt

Angelegt am 17.05.2018 10:00:38 Geändert am 17.05.2018 10:54:17

Speichern | Speichern und zur Übersicht | Wiederherstellen | Hilfe

Abb. 10: Felderkonfiguration - Datensatz verändern

Felder

Feld-Information

Anmeldung

Bezeichnung	Identifizierer	Typ
Veranstaltung	veranstaltung2	applookup/select

Veranstaltungen (Doku)

Bezeichnung	Identifizierer	Typ	Ziel
Veranstaltung	veranstaltung	string/text	
Ort	ort	string/text	
Datum	datum	date/date	
von - bis	von_bis	string/text	
E-Mail-Verteiler	e_mail_verteiler	string/email	

Abb. 11: Feldinformationen - Datensatz verändern

Aktionen abmelden Aktiv?: Ja Beschriftung: Von Veranstaltung abmelden. Icon: minus-square-o

2/5

Bearbeiten | Anweisungen

Speichern | Wiederherstellen | Hilfe

* Identifizierer: abmelden
Eindeutige Kennung für diese Aktion

* Beschriftung: Von Veranstaltung abmelden.

Aktiv? ☑
Eine inaktive Aktion wird nicht angezeigt und kann nicht ausgeführt werden.

Reihenfolge: 20
Die Aktions-Buttons werden im Datenmanagement nach dieser Zahl sortiert angezeigt.

* Berechtigung für: alle Benutzer
Wer darf diese Aktion ausführen?

Beschreibung: Teilnehmer können sich über einen Link in der Anmeldebestätigung wieder von der Veranstaltung abmelden.

Icon: minus-square-o
Das Icon für den Aktions-Button im Datenmanagement. Verwendet werden können alle Namen aus dem FontAwesome-Icon-Set (ohne den Präfix `fa-`, also beispielsweise `paw` für 🐾).

Icon-Vorschau

Verwendung
- ☐ Aktions-Button im Datenmanagement für mehrere Datensätze
- ☐ Aktions-Button im Datenmanagement für einen einzelnen Datensatz
- ☑ Link für E-Mails
- ☐ Vor dem Anzeigen des Bearbeiten-Formulars ausführen
- ☐ Nach Änderung des Datensatzes ausführen (auch bei Import)
- ☐ Nach Anlegen des Datensatzes ausführen (auch bei Import)

Legt fest wo diese Aktion zur Verfügung stehen soll

Meldung: Sie haben sich erfolgreich abgemeldet.
Meldung im Datenmanagement oder bei der Link-Ausführung wenn die Aktion erfolgreich ausgeführt wurde

Filter

Bedingung
Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit die Aktion auf dem Datensatz ausgeführt wird

Kompilierte Bedingung

Historie

Angelegt von: Sabine Voigt — Geändert von: Sabine Voigt

Angelegt am: 27.04.2017 10:42:54 — Geändert am: 31.05.2017 11:08:04

Speichern | Wiederherstellen | Hilfe

Abb. 12: Link für E-Mails

Nach dem Speichern erscheint neben *Bearbeiten* der Tab *Anweisungen*. Nach *Hinzufügen* einer neuen Anweisung kann im nun geöffneten Fenster eingestellt werden, was durch die Aktion ausgeführt werden soll. Siehe *Anweisungen*.

Als *Anweisungstyp* wird *Datensatz verändern* gewählt und bei *Reihenfolge* `10` eingetragen.

Nach dem Speichern der Anweisungen erscheint neben *Bearbeiten* der Tab *Felderbelegung*. Durch Klicken darauf erscheinen alle *Felder der App*, deren Datensätze durch die Aktion verändert werden können. Durch klicken auf das zu ändernde Feld *abgemeldet*, kann dessen *Inhalt konfiguriert* werden.

Im nun geöffneten Fenster wird bei *Aktion Feld auf Wert setzen* und bei *Wert Ja* ausgewählt. Durch Klicken auf *Speichern* ist die Erstellung der Aktion abgeschlossen. Der Link kann mit `{actionurl:abmelden}` oder `{actionlink:abmelden}` in Ihr E-Mail-Template eingebunden werden. Der Unterschied wird in der Dokumentation der *E-Mail-Templates* anhand eines Beispiels erklärt.

Aufgerufen und ausgeführt wird diese Aktion, sobald ein Teilnehmer in der Anmeldebestätigungs-E-Mail den Link zum Abmelden anklickt.

18.2.3 Datensatz löschen

Mit dieser Aktion soll ein einzelner, oder mehrere Datensätze im Datenmanagement gelöscht werden. Mit der Bedingung, dass der (die) Teilnehmer sich abgemeldet hat (haben).

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Aktionen*. Klicken Sie auf *Hinzufügen* um eine neue Aktion zu erstellen.

Im nun geöffneten Fenster werden folgende *Konfigurationen* vorgenommen.

Über das *Formulieren der Bedingung* können Sie im entsprechenden Kapitel mehr lesen.

Nach dem Speichern erscheint neben *Bearbeiten* der Tab *Anweisungen*. Nach *Hinzufügen* einer neuen Anweisung kann im nun geöffneten Fenster eingestellt werden, was durch die Aktion ausgeführt werden soll. Siehe *Anweisungen*.

Als *Anweisungstyp* wird *Datensatz löschen* gewählt und bei *Reihenfolge* `10` eingetragen. Nach dem Speichern der Anweisungen ist die Erstellung der Aktion abgeschlossen.

Ausgeführt werden kann diese Aktion in der Liste des Datenmanagements unter *Daten* → *Liste*.

18.2.4 Datensatz über Fomular anlegen

Bei dieser Aktion wird mit einer zusätzlichen App gearbeitet. In dieser App sind verschiedene Veranstaltungen erfasst. Durch Auswahl eines dieser Datensätze soll die Aktion „Anmelden“ ausgelöst werden. Dadurch öffnet sich das Eingabeformular der App „Anmeldung“. Hier sollen dann bereits Felder mit festen Werten, oder Ausdrucken, die sich auf den Datensatz beziehen, vorbelegt werden. Wird das Eingabeformuar dann gespeichert, wird der neue Datensatz angelegt. Das heißt, es kann aus der App „Veranstaltungen“ heraus, ein Datensatz in der App „Anmeldung“ angelegt werden.

Wählen Sie die App „Veranstaltungen“ und dort im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Aktionen*. Klicken Sie auf *Hinzufügen* um eine neue Aktion zu erstellen.

Im nun geöffneten Fenster werden folgende *Konfigurationen* vorgenommen.

Nach dem Speichern erscheint neben *Bearbeiten* der Tab *Anweisungen*. Nach *Hinzufügen* einer neuen Anweisung kann im nun geöffneten Fenster eingestellt werden, was durch die Aktion ausgeführt werden soll. Siehe *Anweisungen* hinzuzufügen.

Weil die Datensätze in der App *Anmeldung* angelegt werden sollen, wird diese bei *App* ausgewählt. Nach dem Speichern erscheint neben *Bearbeiten* der Button *Felderbelegung*. Durch Klicken darauf erscheinen in der Übersicht alle *Felder der ausgewählten App Anmeldung*.

Diese können hier mit Ausdrucken der App *Veranstaltungen* belegt werden. In unserem Beispiel betrifft das die Felder *Veranstaltung*, *Ort*, *Datum* und *von-bis*. Durch Klicken auf die jeweilige Zeile, können diese Felder konfiguriert werden. Dazu können Sie in der allgemeinen Dokumentation über *Felderbelegung* mehr lesen.

Aktionen abmelden Beschriftung: Von Veranstaltung abmelden. Aktiv?: Ja Icon: minus-square-o

Anweisungen

+ Hinzufügen

Speichern Abbrechen Hilfe

* Anweisungstyp
- Datensatz anlegen
- Datensatz verändern
- Arbeitsaufgabe automatisch anlegen
- Datensatz löschen
- Datensatz über Formular anlegen
- Datenmenge
- Datensatz über Formular (Statisch)
- Onboarding mit Installation
- Arbeitsaufgaben Datensatz E-Mails
- Email-Versendung
- Einladung zu einer App

* Reihenfolge 10

Details

Details

Filter

Bedingung

Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit diese Anweisung auf dem Datensatz ausgeführt wird.

Kompilierte Bedingung

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen Hilfe

Abb. 13: Anweisungen - Link für E-Mails

Aktionen abmelden Beschriftung: Von Veranstaltung abmelden. Aktiv?: Ja Icon: minus-square-o

Anweisungen Datensatz verändern Reihenfolge: 10

Bearbeiten Felderbelegung

Volltextsuche Anzeige (Alle) Suchen Erweitert

›› 17 Felder gefunden Zeilen pro Seite 17 Los

	Beschriftung	Typ	Identifizierer	Aktion
1	bezahlt	bool	bezahlt	
2	bezahlt am	date/date	bezahlt_am	
3	Teilnahmegebühr	number	teilnahmegebuehr	
4	abgemeldet	bool	abgemeldet	
5	Veranstaltung	lookup/select	veranstaltung2	
6	Vorname	string/text	vorname	
7	Nachname	string/text	nachname	
8	Straße Hsnr.	string/text	strasse_und_hsnr2	
9	PLZ Ort	string/text	plz_ort	
10	E-Mail-Adresse	string/email	e_mail_adresse	
11	Telefon	string/tel	telefon	
12	Anzahl Personen	number	anzahl_personen2	
13	Wird Übernachtung benötigt?	lookup/radio	wird_uebernachtung_benoetigt	
14	Einzelzimmer	number	einzelzimmer	
15	Euro / Nacht	number	euro_nacht	
16	Doppelzimmer	number	mehrbettzimmer	
17	Euro / Nacht	number	euro_nacht2	

CSV PDF

Abb. 14: Felderbelegung - Link für E-Mails

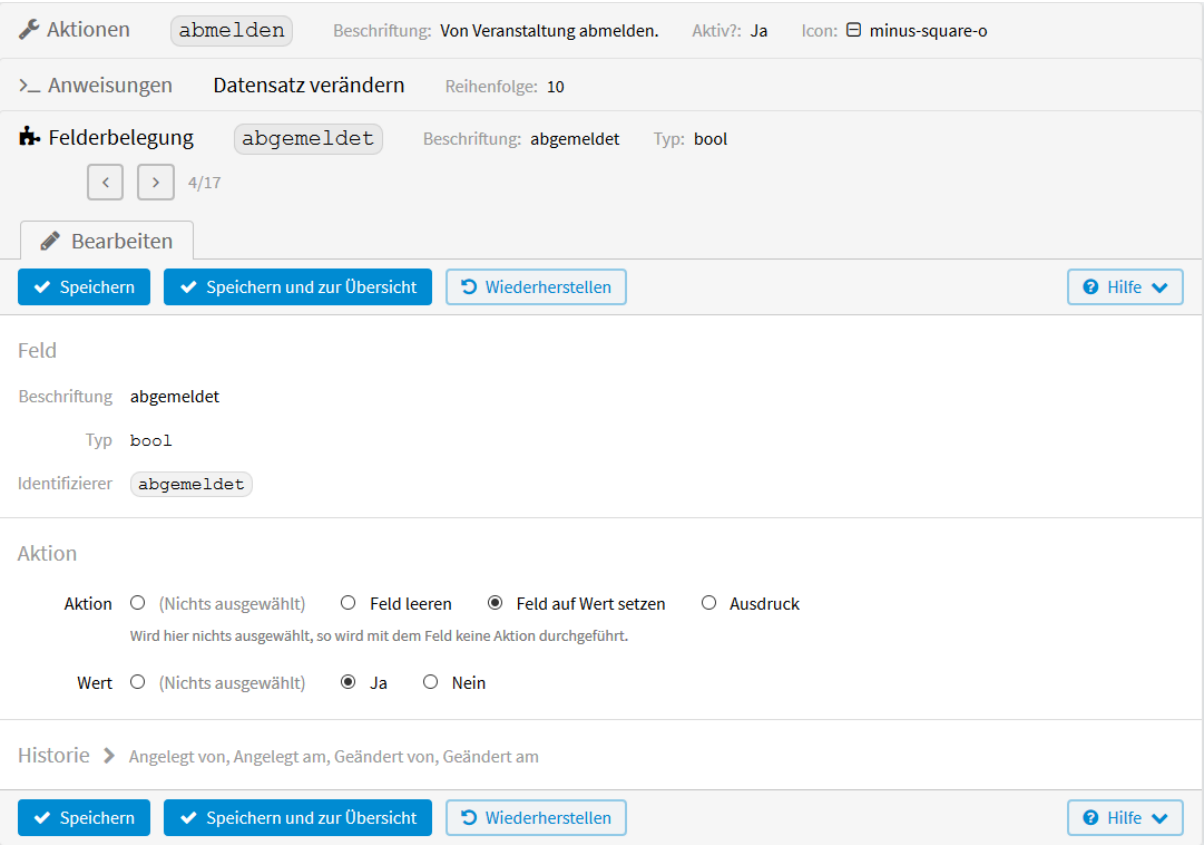

Abb. 15: Felderkonfiguration - Link für E-Mails

Für die Formulierung des Ausdrucks wird *vSQL* verwendet. Dafür werden die Identifizierer der Felder der App *Veranstaltungen* benötigt. Diese finden Sie in der linken Menüleiste unter *Felder*. Auf die Werte kann dann zugegriffen werden mit: `r.v_identifier`. In unserem Beispiel lauten die Ausdrucke `r.v_veranstaltung`, `r.v_ort`, `r.v_datum` und `r.v_von_bis`. Siehe *Ergebnis Felderkonfiguration*.

Nach der Eingabe und dem Speichern aller benötigten Ausdrucke ist die Erstellung der Aktion abgeschlossen. *Aufgerufen* werden kann die Aktion in der Liste des Datenmanagements der App *Veranstaltungen* unter *Daten* → *Liste*. Klicken Sie dafür ganz rechts auf den Button in der Zeile der Veranstaltung, für die Sie eine Anmeldung anlegen wollen und wählen in dem Menü die Aktion *Anmeldung*. . . .

Nach dem Klicken auf *Anmeldung* öffnet sich das Eingabefenster der App *Anmeldung*, in dem dann bereits die vier Felder mit den Daten aus der App *Veranstaltung* vorbelegt sind. Siehe *Ergebnis*.

Jetzt müssen nur noch die persönlichen Daten ergänzt und der Datensatz gespeichert werden.

18.2.5 Arbeitsaufgaben Datensatz E-Mails

Im folgenden Anwendungsbeispiel soll bei einer neuen Anmeldung zur Veranstaltung eine Arbeitsaufgabe angelegt werden, mit der Bedingung, dass im Anmeldeformular bei *Übernachtung benötigt ja* gewählt wurde.

Die Arbeitsaufgabe fordert auf, die entsprechende Anzahl an Zimmern zu buchen. Nach Erledigung der Aufgabe soll durch eine weitere Aktion das Häkchen bei *Zimmer gebucht* im Anmeldeformular gesetzt werden.

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Aktionen*. Klicken Sie auf *Hinzufügen* um eine neue Aktion zu erstellen.

Im nun geöffneten Fenster werden folgende *Konfigurationen* vorgenommen.

Da die Arbeitsaufgabe nur angelegt werden sollen, wenn im Anmeldeformular bei *Übernachtung benötigt Ja* gewählt wurde, muss folgende *Bedingung* eingegeben werden:

Aktionen loeschen Beschriftung: Datensatz löschen Aktiv?: Ja Icon: trash

3/6

Bearbeiten Anweisungen

Speichern Wiederherstellen Hilfe

* Identifizierer loeschen
Eindeutige Kennung für diese Aktion

* Beschriftung Datensatz löschen

Aktiv? ☑
Eine inaktive Aktion wird nicht angezeigt und kann nicht ausgeführt werden.

Reihenfolge 30
Die Aktions-Buttons werden im Datenmanagement nach dieser Zahl sortiert angezeigt.

* Berechtigung für alle Benutzer
Wer darf diese Aktion ausführen?

Beschreibung Löscht einen oder mehrere Datensätze.

Icon trash
Das Icon für den Aktions-Button im Datenmanagement.

Icon-Vorschau

Verwendung
- ☑ Aktions-Button im Datenmanagement für mehrere Datensätze
- ☑ Aktions-Button im Datenmanagement für einen einzelnen Datensatz
- ☐ Link für E-Mails
- ☐ Vor dem Anzeigen des Bearbeiten-Formulars ausführen
- ☐ Nach Änderung des Datensatzes ausführen (auch bei Import)
- ☐ Nach Anlegen des Datensatzes ausführen (auch bei Import)

Legt fest wo diese Aktion zur Verfügung stehen soll

Meldung
Meldung im Datenmanagement oder bei der Link-Ausführung wenn die Aktion erfolgreich ausgeführt wurde

Filter

Bedingung `r.v_abgemeldet is not None and r.v_abgemeldet`
Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit die Aktion auf dem Datensatz ausgeführt wird

Kompilierte Bedingung `r.v_abgemeldet is not None and r.v_abgemeldet`

Historie

Angelegt von Sabine Voigt Geändert von Ulrich Eidt

Angelegt am 07.04.2017 13:57:48 Geändert am 15.03.2018 09:07:56

Speichern Wiederherstellen Hilfe

Abb. 16: Datensatz löschen

Aktionen loeschen Beschriftung: Datensatz löschen Aktiv?: Ja Icon: trash

Anweisungen

+ Hinzufügen

Speichern Abbrechen Hilfe

* Anweisungstyp
- Datensatz anlegen
- Datensatz verändern
- Arbeitsaufgabe automatisch anlegen
- Datensatz löschen
- Datensatz über Formular anlegen
- Datenmenge
- Datensatz über Formular (Statisch)
- Onboarding mit Installation
- Arbeitsaufgaben Datensatz E-Mails
- Email-Versendung
- Einladung zu einer App

* Reihenfolge 10

Details

Details

Filter

Bedingung

Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit diese Anweisung auf dem Datensatz ausgeführt wird.

Kompilierte Bedingung

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen Hilfe

Abb. 17: Anweisungen - Datensatz löschen

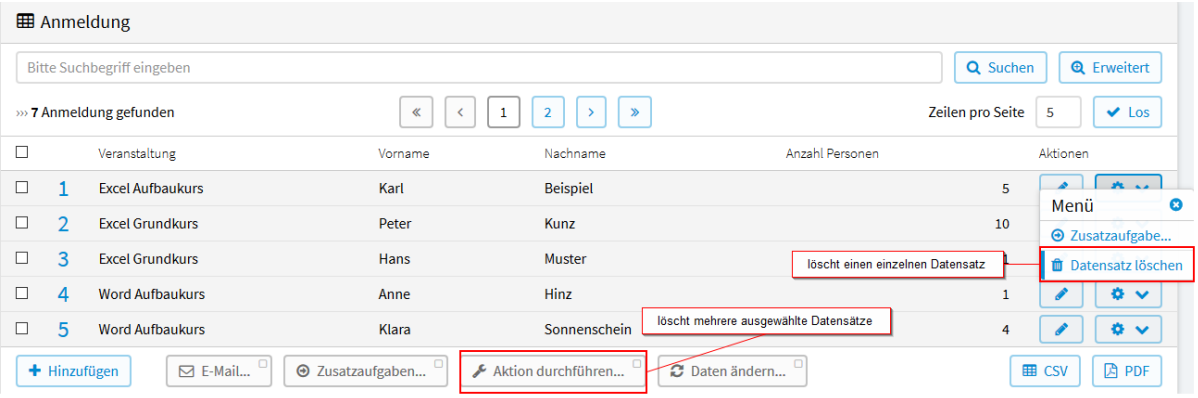

Abb. 18: Aktion ausführen - Datensatz löschen

```
r.v_wird_uebernachtung_benoetigt == "ja"
```

Hierzu können sie im Kapitel *Formulieren der Bedingung* mehr lesen.

Nach dem Speichern erscheint neben *Bearbeiten* der Tab *Anweisungen*. Nach *Hinzufügen* einer neuen Anweisung kann im nun geöffneten Fenster eingestellt werden, was durch die Aktion ausgeführt werden soll. Siehe *Anweisungen*.

Als *Anweisungstyp* wird *Arbeitsaufgaben Datensatz E-Mails* gewählt und bei *Reihenfolge* `10` eingegeben.

Nach dem Speichern der Anweisungen erscheint neben *Bearbeiten* der Tab *Felderbelegung*. Durch Klicken darauf sehen Sie folgende *Felder*. Diese sind bei jeder App gleich.

User - string/text - c_user
: Angemeldeter Benutzer - kann nicht geändert werden.

Datensatz - string/text - dat_id
: Datensatz, an den die Aufgabe gehängt wird (dies ist normalerweise `r.id`, d.h. der Datensatz für den die Aktion aufgerufen wurde).

E-Mails als Komma-Liste - string/text - emails
: E-Mail-Adresse(n), an die die Arbeitsaufgabe als Link verschickt wird (werden). Z. B. können Sie `r.v_identifier` (Identifizierer des E-Mail-Feldes) bei *Ausdruck* oder die E-Mail-Adressen kommagetrennt bei *Feld auf Wert setzen* eintragen.

  Bitte beachten: Bei E-Mail-Adressen, die im System sind, werden die Aufgaben im Account erzeugt. Ansonsten kommt die Aufgabe immer als E-Mail-Aufgabe. Wird bei *Per Email immer Ja* gesetzt ist, kommt die Aufgabe immer (auch) als E-Mail-Aufgabe.

Aufgabe - string/text - tsk_name
: Name der Aufgabe

Beschreibung - string/text - tsk_description
: Beschreibung der Aufgabe

Ergebnis 1 - string/text - res_name1
: Text des Ergebnis-Buttons (z. B. Erledigt, Abgelehnt, ...)

Datenaktion bei 1 - string/text - da_id_1
: Datenaktion, die bei Wahl von Ergebnis 1 ausgeführt werden soll. Hier muss die id oder der identifier der Aktion bei *Feld auf Wert setzen* eingetragen werden.

Ergebnis 2 - string/text - res_name2
: Text des Ergebnis-Buttons (z. B. Erledigt, Abgelehnt, ...)

Datenaktion bei 2 - string/text - da_id_2
: Datenaktion, die bei Wahl von Ergebnis 2 ausgeführt werden soll. Hier muss die id oder der identifier der Aktion bei *Feld auf Wert setzen* eingetragen werden.

Aktionen anmeldung Aktiv?: Ja Beschriftung: Anmeldung Icon: plus-square

1/2

Bearbeiten | Anweisungen

Speichern | Wiederherstellen | Hilfe

* Identifizierer: anmeldung
Eindeutige Kennung für diese Aktion

* Beschriftung: Anmeldung

Aktiv? ☑
Eine inaktive Aktion wird nicht angezeigt und kann nicht ausgeführt werden.

Reihenfolge: 10
Die Aktions-Buttons werden im Datenmanagement nach dieser Zahl sortiert angezeigt.

* Berechtigung für: alle Benutzer
Wer darf diese Aktion ausführen?

Beschreibung: Legt neuen Datensatz in der App "Anmeldung" an.

Icon: plus-square
Das Icon für den Aktions-Button im Datenmanagement. Verwendet werden können alle Namen aus dem FontAwesome-Icon-Set (ohne den Präfix `fa-`, also beispielsweise `paw` für 🐾).

Icon-Vorschau

Verwendung
- ☐ Aktions-Button im Datenmanagement für mehrere Datensätze
- ☑ Aktions-Button im Datenmanagement für einen einzelnen Datensatz
- ☐ Link für E-Mails
- ☐ Vor dem Anzeigen des Bearbeiten-Formulars ausführen
- ☐ Nach Änderung des Datensatzes ausführen (auch bei Import)
- ☐ Nach Anlegen des Datensatzes ausführen (auch bei Import)

Legt fest wo diese Aktion zur Verfügung stehen soll

Meldung
Meldung im Datenmanagement oder bei der Link-Ausführung wenn die Aktion erfolgreich ausgeführt wurde

Filter

Bedingung
Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit die Aktion auf dem Datensatz ausgeführt wird

Kompilierte Bedingung

Historie

Angelegt von: Sabine Voigt | Geändert von

Angelegt am: 22.06.2017 09:31:53 | Geändert am

Speichern | Wiederherstellen | Hilfe

Abb. 19: Datensatz über Formular anlegen

Aktionen anmeldung Beschriftung: Anmeldung Aktiv?: Ja Icon: plus-square

Anweisungen

+ Hinzufügen

Speichern Abbrechen Hilfe

* Anweisungstyp
- Datensatz anlegen
- Datensatz verändern
- Arbeitsaufgabe automatisch anlegen
- Datensatz löschen
- Datensatz über Formular anlegen
- Datenmenge
- Datensatz über Formular (Statisch)
- Onboarding mit Installation
- Arbeitsaufgaben Datensatz E-Mails
- Email-Versendung
- Einladung zu einer App

* Reihenfolge 10

App Anmeldung

Wird hier nichts ausgewählt, betrifft die Aktion die aktuelle App.

Identifizierer

Hier können Sie festlegen, unter welchem Identifizierer der durch diese Aktion gelieferte Datensatz für die untergeordneten Anweisungen zugänglich sein soll.

Details

Details

Filter

Bedingung

Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit diese Anweisung auf dem Datensatz ausgeführt wird.

Kompilierte Bedingung

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen Hilfe

Abb. 20: Anweisungen - Datensatz über Formular anlegen

Aktionen anmeldung Beschriftung: Anmeldung Aktiv?: Ja Icon: plus-square

Anweisungen Anweisungstyp: Datensatz über Formular anlegen Reihenfolge: 2

App: Anmeldung

1/2

Bearbeiten Felderbelegung

Volltextsuche Anzeige (Alle) Suchen Erweitert

›› 11 Felder gefunden Zeilen pro Seite 13 Los

	Beschriftung	Typ	Identifizierer	Aktion
1	Veranstaltung	string/text	veranstaltung	
2	Ort	string/text	ort	
3	Datum	date/date	datum	
4	von - bis	string/text	von_bis	
5	Vorname	string/text	vorname	
6	Nachname	string/text	nachname	
7	Straße Hsnr.	string/text	strasse_hsnr2	
8	PLZ Ort	string/text	plz_ort	
9	E-Mail-Adresse	string/email	e_mail_adresse	
10	Telefonnummer	string/text	telefonnummer	
11	Anzahl Personen	number	anzahl_personen	

CSV PDF

Abb. 21: Felderbelegung - Datensatz über Formular anlegen

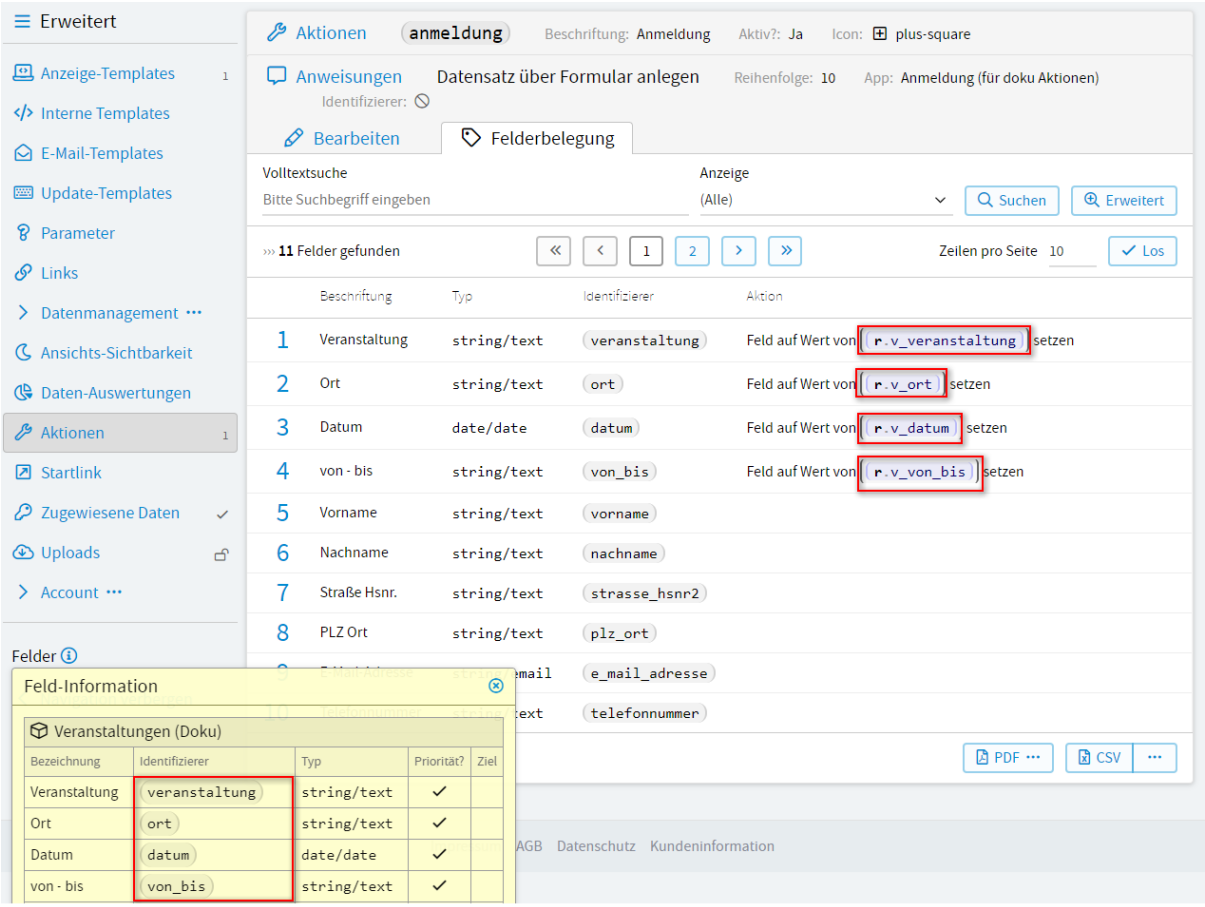

Abb. 22: Ergebnis Felderkonfiguration - Datensatz über Formular anlegen

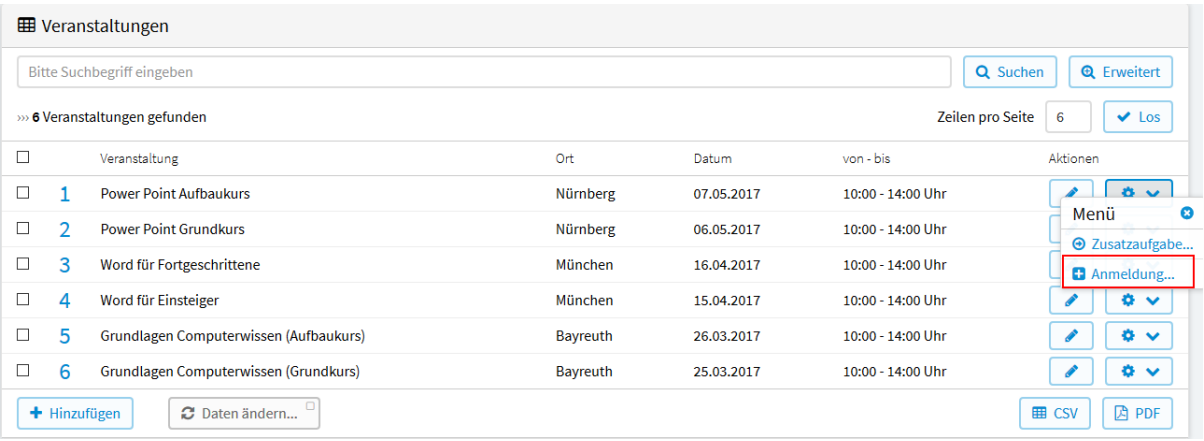

Abb. 23: Aktion aufrufen - Datensatz über Formular anlegen

Anmeldung

Veranstaltung: Power Point Aufbaukurs

Ort: Nürnberg

Datum: 07.05.2017

übernommen von App "Veranstaltung"

von - bis: 10:00 - 14:00 Uhr

Vorname

Nachname

Straße Hsnr.

PLZ Ort

E-Mail-Adresse

Telefonnummer

Anzahl Personen

Speichern | Abbrechen

Abb. 24: Ergebnis - Datensatz über Formular anlegen

Aktionen arbeitsaufgabe_email Beschriftung: Legt Arbeitsaufgabe an. Aktiv?: Ja Icon: » angle-double-right

5/6

Bearbeiten Anweisungen

+ Hinzufügen Kopieren − Löschen Hilfe

* Identifizierer arbeitsaufgabe_email
Eindeutige Kennung für diese Aktion

* Beschriftung Legt Arbeitsaufgabe an.

Aktiv? ☑
Eine inaktive Aktion wird nicht angezeigt und kann nicht ausgeführt werden.

Reihenfolge 40
Die Aktions-Buttons werden im Datenmanagement nach dieser Zahl sortiert angezeigt.

* Berechtigung für alle Benutzer
Wer darf diese Aktion ausführen?

Beschreibung Legt bei neuem Datensatz eine Arbeitsaufgabe an und verschickt zur Bearbeitung einen Link per E-Mail. Bedingung: Übernachtung benötigt: "JA".

Icon angle-double-right
Das Icon für den Aktions-Button im Datenmanagement.

Icon-Vorschau »

Verwendung
- ☐ Aktions-Button im Datenmanagement für mehrere Datensätze
- ☐ Aktions-Button im Datenmanagement für einen einzelnen Datensatz
- ☐ Link für E-Mails
- ☐ Vor dem Anzeigen des Bearbeiten-Formulars ausführen
- ☐ Nach Änderung des Datensatzes ausführen (auch bei Import)
- ☑ Nach Anlegen des Datensatzes ausführen (auch bei Import)

Legt fest wo diese Aktion zur Verfügung stehen soll

Meldung
Meldung im Datenmanagement oder bei der Link-Ausführung wenn die Aktion erfolgreich ausgeführt wurde

Filter

Bedingung `r.v_wird_uebernachtung_benoetigt == "ja"`
Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit die Aktion auf dem Datensatz ausgeführt wird

Kompilierte Bedingung `r.v_wird_uebernachtung_benoetigt == "ja"`

Historie

Angelegt von Sabine Voigt Geändert von Ulrich Eidt

Angelegt am 16.11.2017 13:06:51 Geändert am 15.03.2018 09:07:37

+ Hinzufügen Kopieren − Löschen Hilfe

Abb. 25: Arbeitsaufgaben Datensatz E-Mails

Aktionen arbeitsaufgabe_email Beschriftung: Legt Arbeitsaufgabe an. Aktiv?: Ja Icon: »
angle-double-right

Anweisungen

+ Hinzufügen

Speichern Abbrechen Hilfe

* Anweisungstyp
- Datensatz anlegen
- Datensatz verändern
- Arbeitsaufgabe automatisch anlegen
- Datensatz löschen
- Datensatz über Formular anlegen
- Datenmenge
- Datensatz über Formular (Statisch)
- Onboarding mit Installation
- Arbeitsaufgaben Datensatz E-Mails
- Email-Versendung
- Einladung zu einer App

* Reihenfolge 10

Details

Details

Filter

Bedingung

Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit diese Anweisung auf dem Datensatz ausgeführt wird.

Kompilierte Bedingung

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen Hilfe

Abb. 26: Anweisungen - Arbeitsaufgaben Datensatz E-Mails

Aktionen `arbeitsaufgabe_email` Beschriftung: Legt Arbeitsaufgabe an.
Aktiv?: Ja

Anweisungen Arbeitsaufgaben Datensatz E-Mails Reihenfolge: 1

Bearbeiten | Felderbelegung

Volltextsuche | Anzeige (Alle) | Suchen | Erweitert

›› 25 Felder gefunden | Zeilen pro Seite 25 | Los

	Beschriftung	Typ	Identifizierer	Aktion
1	User	string/text	c_user	
2	Datensatz	string/text	dat_id	
3	E-Mails als Komma-Liste	string/text	emails	
4	Aufgabe	string/text	tsk_name	
5	Beschreibung	string/text	tsk_description	
6	Ergebnis 1	string/text	res_name1	
7	Datenaktion bei 1	string/text	da_id_1	
8	Ergebnis 2	string/text	res_name2	
9	Datenaktion bei 2	string/text	da_id_2	
10	Ergebnis 3	string/text	res_name3	
11	Datenaktion bei 3	string/text	da_id_3	
12	Ergebnis 4	string/text	res_name4	
13	Datenaktion bei 4	string/text	da_id_4	
14	Ergebnis 5	string/text	res_name5	
15	Datenaktion bei 5	string/text	da_id_5	
16	Zu erledigen bis (Datum)	date	wfl_date	
17	Warn-Email an Bearbeiter am	date	escalate_date	
18	Aufgabe deaktivieren am	date	timeout_date	
19	Benachrichtigung (adhoc_data_note)	string/text	adhoc_type	
20	Aufgabe anzeigen ab	date	wfl_datefrom	
21	Aufgabe anzeigen bis	date	wfl_dateto	
22	Kalenderexport	bool	wfl_calendarexport	
23	Kalender-Alarm	bool	wfl_calendaralarm	
24	Per Email immer	bool	email_forced	
25	Felder verstecken (Komma getrennt)	string/text	hidecontrols	

PDF | CSV

Abb. 27: Felderbelegung - Arbeitsaufgaben Datensatz E-Mails

Ergebnis 3 - string/text - res_name3
Text des Ergebnis-Buttons (z. B. Erledigt, Abgelehnt, …)

Datenaktion bei 3 - string/text - da_id_3
Datenaktion, die bei Wahl von Ergebnis 3 ausgeführt werden soll. Hier muss die id oder der identifier der Aktion bei *Feld auf Wert setzen* eingetragen werden.

Ergebnis 4 - string/text - res_name4
Text des Ergebnis-Buttons (z. B. Erledigt, Abgelehnt, …)

Datenaktion bei 4 - string/text - da_id_4
Datenaktion, die bei Wahl von Ergebnis 4 ausgeführt werden soll. Hier muss die id oder der identifier der Aktion bei *Feld auf Wert setzen* eingetragen werden.

Ergebnis 5 - string/text - res_name5
Text des Ergebnis-Buttons (z. B. Erledigt, Abgelehnt, …)

Datenaktion bei 5 - string/text - da_id_5
Datenaktion, die bei Wahl von Ergebnis 5 ausgeführt werden soll. Hier muss die id oder der identifier der Aktion bei *Feld auf Wert setzen* eingetragen werden.

Zu erledigen bis - date - wfl_date
Bis wann soll die Aufgabe erledigt sein.

Warn-Email an Bearbeiter am - date - escalate_date
Erinnerung

Aufgabe deaktivieren am - date - timeout_date
Wenn die Aufgabe bis zu diesem Datum nicht erledigt ist, wird sie deaktiviert.

Benachrichtigung (adhoc_data_note) - string/text - adhoc_type
`adhoc_data_note` gibt dem Ersteller (= c_user) eine Benachrichtigungs-Aufgabe, mit „Aufgabe wurde erledigt“ - einziges Ergebnis = „Zur Kenntnis genommen“.

Bei Eingabe von `adhoc_no_action`, oder wenn nichts eingegeben wird, passiert nichts.

Aufgabe anzeigen ab - date - wfl_datefrom
Ab wann soll die Aufgabe in der Aufgabenübersicht angezeigt werden (ist gut für „Wiedervorlage“). Funktioniert mit E-Mail-Aufgaben noch nicht.

Aufgabe anzeigen bis - date - wfl_dateto
Bis wann soll die Aufgabe angezeigt werden?

Kalenderexport - bool - wfl_calendarexport
Für Aufgaben-Kalender

Kalender-Alarm - bool - wfl_calendaralarm
Alarm - Stunden oder Tage vorher

Per Email immer - bool - email_forced
Versendet die Aufgabe immer (auch) per E-Mail, auch wenn die E-Mail-Adresse im System ist.

Felder verstecken (kommagetrennt) - string/text - hidecontrols
Felder, die nicht in den allgemeinen Informationen der Arbeitsaufgabe mit angezeigt werden sollen (z. B. private Felder), können hier versteckt werden. Geben Sie dafür die *Identifizierer* dieser Felder bei *Feld auf Wert setzen* kommagetrennt ein.

Durch Klicken auf das zu ändernde Feld, sind folgende Konfigurationen möglich.

Aktion

Nichts ausgewählt
Wird hier nichts ausgewählt, so wird mit dem Feld keine Aktion durchgeführt.

Feld leeren
Der Feldinhalt wird gelöscht. Das ist nur beim Ändern von Datensätzen sinnvoll.

Feld auf aktuelles Datum setzen
Diese Auswahl erscheint nur, wenn das ausgewählte Feld vom Typ `date`, `datetimeminute` oder `datetimesecond` ist.

Feld auf Wert setzen
Das Feld übernimmt, nachdem die Aktion ausgeführt wurde, den hier eingegebenen Wert.

Ausdruck
Für die Formulierung des Ausdrucks wird *vSQL* verwendet. Dafür werden die Identifizierer der Felder benötigt. Diese finden Sie in der linken Menüleiste unter *Felder*. Auf die Werte kann dann zugegriffen werden mit: `r.v_identifier`.

Nach dem Speichern erscheint unterhalb des Feldes *Ausdruck* der kompilierte Ausdruck. Ist ein Fehler in der Formulierung, sehen Sie diesen hier.

In unserem Beispiel wurden folgende *Konfigurationen* vorgenommen.

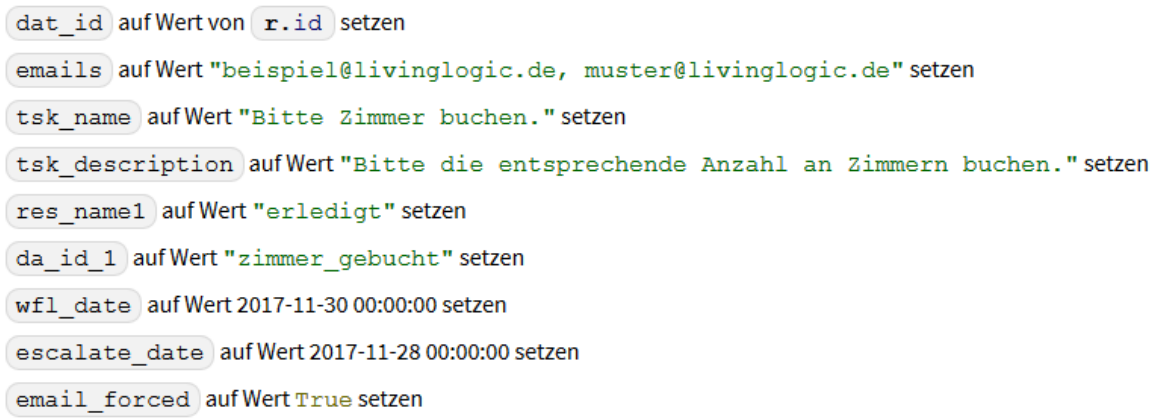

Abb. 28: Felderkonfiguration - Arbeitsaufgaben Datensatz E-Mails

***dat_id* auf Wert von *r.id* setzen**
Die Arbeitsaufgabe wird an den aktuellen Datensatz gehängt.

***emails* auf Wert `"beispiel@livinglogic.de, muster@livinglogic.de"` setzen**
An diese E-Mail-Adressen wird die Arbeitsaufgabe geschickt. Da die E-Mail- Adressen im System sind, wird die *Arbeitsaufgabe im Account* erzeugt.

***tsk_name* auf Wert `"Bitte Zimmer buchen"` setzen**
Name der Aufgabe

***tsk_description* auf Wert `"Bitte die entsprechende Anzahl an Zimmern buchen."` setzen**
Beschreibung der Aufgabe.

***res_name1* auf Wert `"erledigt"` setzen**
Name des Ergebnisbuttons

***da_id_1* auf Wert `"zimmer_gebucht"` setzen**
Die Aktion mit dem Identifizierer `zimmer_gebucht` wird ausgeführt, sobald die Arbeitsaufgabe `erledigt` ist. Diese Aktion setzt das Häkchen bei „Zimmer sind gebucht“ im Formular der App „Anmeldung“. Wie sie eine Aktion anlegen die einen *Datensatz verändert*, können Sie im entsprechenden Kapitel nachlesen.

***wfl_date* auf Wert `"2017-11-30 00:00:00"` setzen**
Die Aufgabe soll bis zum 30.11.2017 erledigt sein.

***escalate_date* auf Wert `"2017-11-28 00:00:00"`**
Erinnerung am 28.11.2017

***email_forced* auf Wert `True` setzen**
Die Arbeitsaufgabe wird immer auch *per E-Mail* versendet.

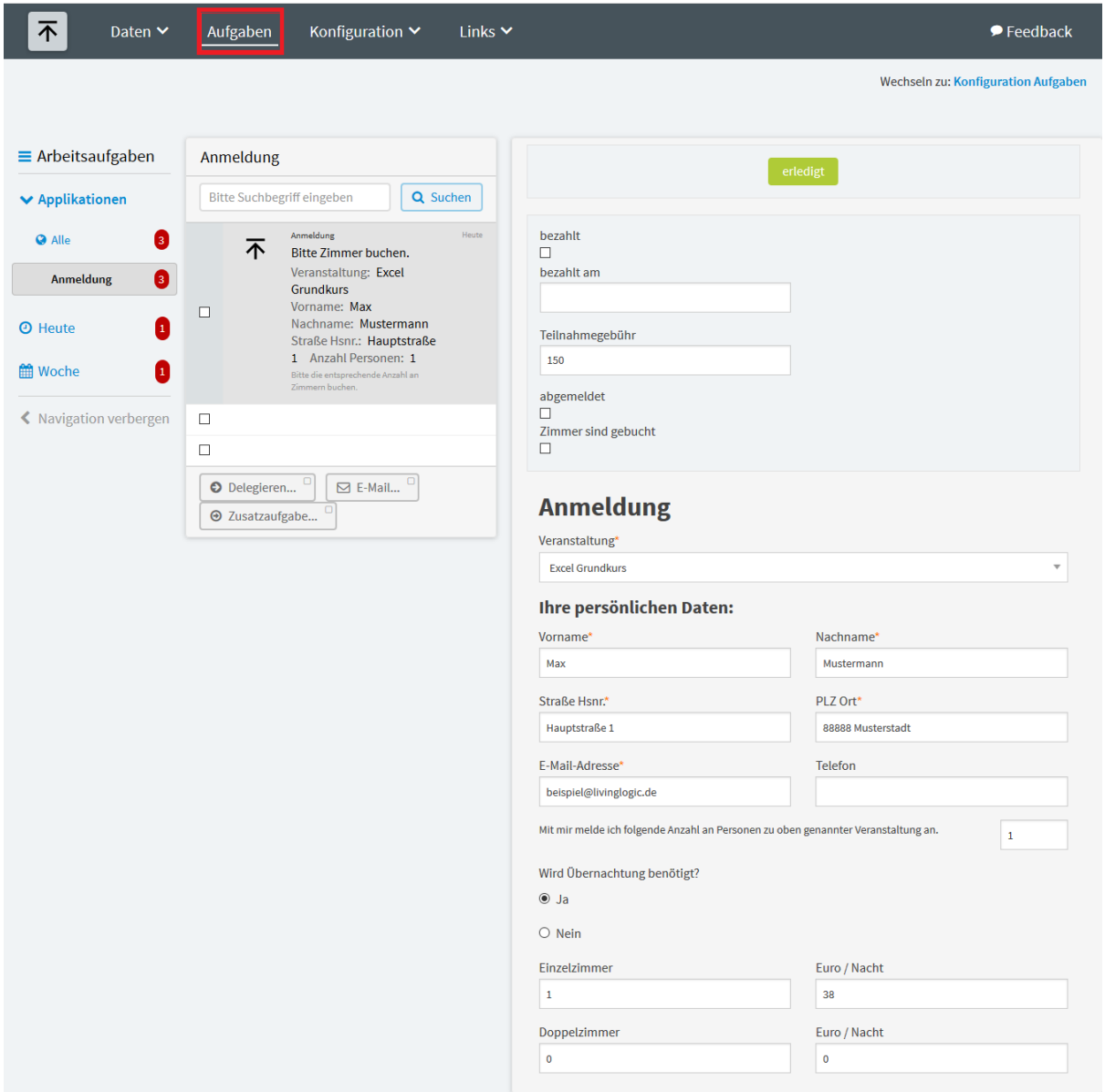

Abb. 29: Arbeitsaufgabe im Account

Für Sie wurde eben eine Aufgabe angelegt.
Bitte die entsprechende Anzahl an Zimmern buchen.

Zur Bearbeitung der Aufgabe benutzen Sie bitte folgenden Link:

http://lapp.io/EyyscsgZV

Vielen Dank für Ihre Mitwirkung.

Diese Nachricht wurde automatisch erzeugt.

Abb. 30: Arbeitsaufgabe per E-Mail

18.2.6 E-Mail-Versendung

Im folgenden Anwendungsbeispiel sollen Teilnehmer vor einer Veranstaltung per E-Mail erinnert werden. Diese Mails können im Datenmanagement durch eine Aktion verschickt werden. Der E-Mail-Inhalt wird über ein E-Mail-Template eingebunden. Die Erinnerung soll nur an die Teilnehmer verschickt werden, die sich nicht wieder abgemeldet haben.

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Aktionen*. Klicken Sie auf *Hinzufügen* um eine neue Aktion zu erstellen.

Im nun geöffneten Fenster werden folgende *Konfigurationen* vorgenommen.

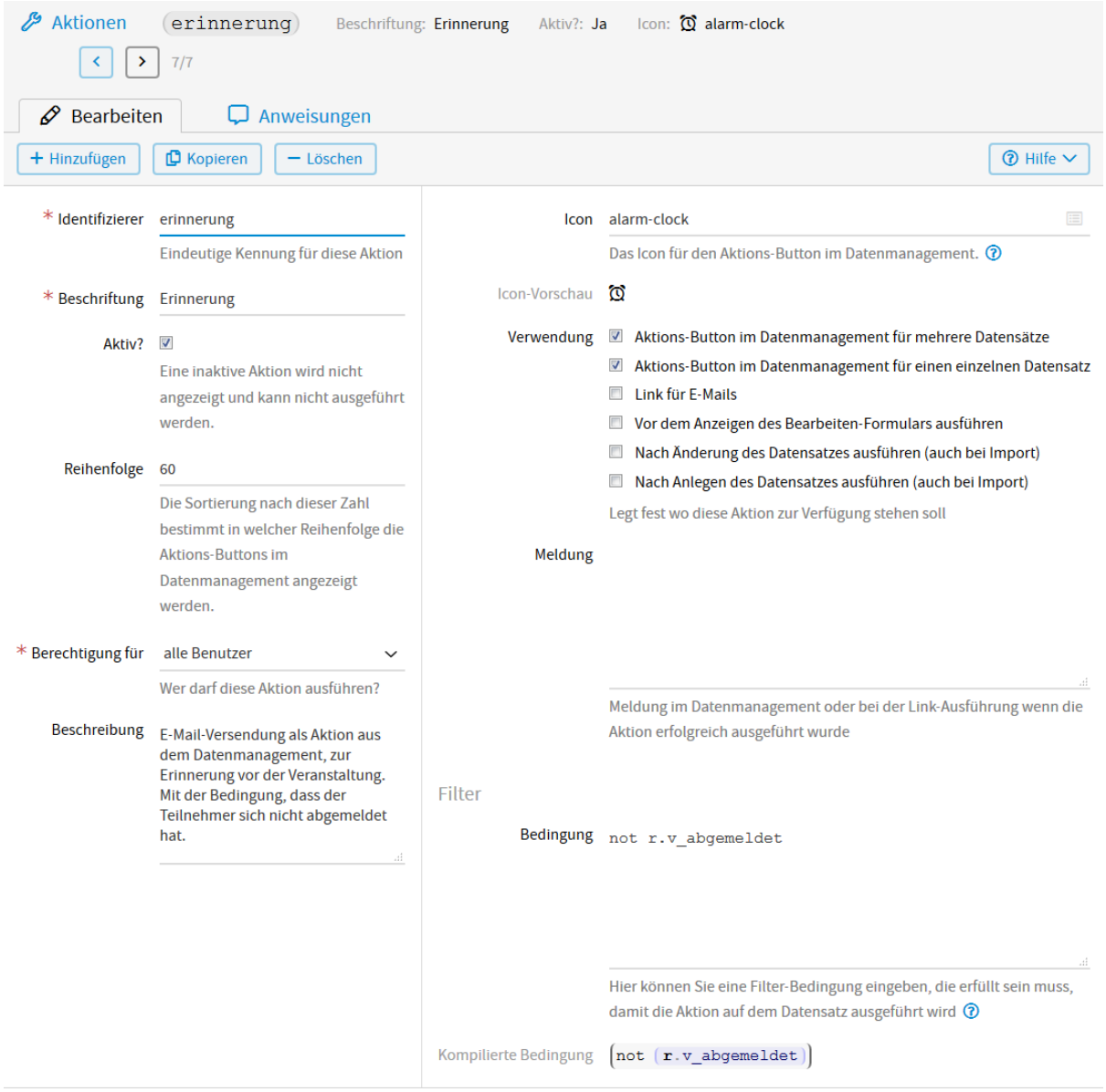

Abb. 31: E-Mail-Versendung

Da nur die Teilnehmer erinnert werden sollen, die sich nicht wieder abgemeldet haben, muss folgende *Bedingung* eingegeben werden:

```
not r.v_abgemeldet
```

Hierzu können sie im Kapitel *Formulieren der Bedingung* mehr lesen.

Nach dem Speichern erscheint neben *Bearbeiten* der Tab *Anweisungen*. Nach *Hinzufügen* einer neuen Anweisung

kann im nun geöffneten Fenster eingestellt werden, was durch die Aktion ausgeführt werden soll. Siehe *Anweisungen*.

Als *Anweisungstyp* wird *E-Mail-Versendung* gewählt und bei *Reihenfolge* `10` eingegeben.

Nach dem Speichern der Anweisungen erscheint neben *Bearbeiten* der Tab *Felderbelegung*. Durch Klicken darauf sehen Sie folgende *Felder*. Diese sind bei jeder App gleich.

Sprache (=de)
Wert kann gesetzt werden auf „de“ für Deutsch, oder „en“ für Englisch.

Typ (=dataaction)
Wert muss immer auf `dataaction` gesetzt werden.

An
Geben Sie hier den Empfänger der E-Mail an. Die E-Mail kann auch an mehrere Empfänger versendet werden. Dazu müssen Sie deren E-Mail-Adressen unter *Feld auf Wert setzen* kommagetrennt auflisten. Gibt es in Ihrer App ein E-Mail-Feld, können sie, wenn sie *Ausdruck* wählen, auf die dort gespeicherten E-Mail-Adressen zugreifen mit `r.v_identifier`. Wobei identifier für den Identifizierer des E-Mail-Feldes steht.

cc
Hier können Sie weitere Empfänger angeben. Wenn Sie mehr als eine E-Mail-Adresse eingeben wollen, dann trennen Sie diese durch Kommas.

bcc
An die hier aufgelisteten, kommagetrennten Empfänger wird eine Blindkopie der E-Mail versendet.

Betreff
Betreff der E-Mail. Einen gleichbleibenden Wert geben Sie bei *Feld auf Wert setzen* ein. Wählen Sie *Ausdruck*, können Sie auch hier auf die Felder Ihrer App zugreifen mit `r.v_identifier`. Wobei identifier für den Identifizierer des Feldes steht, auf das Sie zugreifen möchten.

Inhalt (Text)
Hier können sie den Inhalt der E-Mail als Text eingeben.

Inhalt (HTML)
Hier können Sie den Inhalt Ihrer E-Mail mit HTML gestalten.

tpl_id (=r.app.id)
Die App, von der aus die E-Mail verschickt wird. Hier muss als *Ausdruck* `r.app.id` eingegeben werden.

dat_id (=r.id)
Datensatz für den die E-Mail verschickt wird. Hier muss bei *Ausdruck* `r.id` eingegeben werden.

User (automatisch)
Ist immer der Benutzer der die Aktion durchführt (angemeldeter Benutzer) - kann nicht geändert werden.

Account E-Mail
Der bei LivingApps registrierte Account von dem „von“ (Name) und „Antwort an“ (E-Mail-Adresse) gefüllt werden soll. Geben Sie hier nichts ein, wird das Feld automatisch mit dem Namen und der E-Mail-Adresse von dem gefüllt, der die Aktion auslöst. Möchten Sie z. B. dass hier immer der Name und die E-Mail-Adresse des zuständige Mitarbeiters steht, können Sie bei *Feld auf Wert setzen* dessen E-Mail-Adresse eingeben, oder bei *Ausdruck* darauf zugreifen mit `r.v_identifier`. Wobei identifier für den Identifizierer des E-Mail-Feldes steht.

Email-Template-ID
Möchten Sie den E-Mail-Inhalt über ein E-Mail-Template einfügen, geben Sie hier die Email-Template-ID an. Diese erhalten Sie, wenn Sie in der Übersicht Ihrer E-Mail-Templates das entsprechende Template anklicken. Die ID besteht aus dem letzen Ziffern- und Zahlenblock, der hinter `entry=` der angezeigten *URL* steht. Bei Verwendung eines E-Mail-Templates werden die Felder *An Betreff* und *Account E-Mail* in der E-Mail von der Aktion übernommen. Der Inhalt der E-Mail wird vom E-Mail-Template übernommen. Über die Erstellung eines E-Mail-Templates können sie in der Dokumentation der erweiterten Funktionen von LivingApps unter *E-Mail-Templates* mehr lesen. Als *Typ* des E-Mail-Templates muss hierfür *Vorlage für Nachricht an alle Datensätze* gewählt werden.

Durch Klicken auf das zu ändernde Feld, sind folgende Konfigurationen möglich.

Aktionen erinnerung Beschriftung: Erinnerung Aktiv?: Ja Icon: alarm-clock

Anweisungen

+ Hinzufügen

Speichern Abbrechen Hilfe

* Anweisungstyp
- Datensatz anlegen
- Datensatz verändern
- Arbeitsaufgabe automatisch anlegen
- Datensatz löschen
- Datensatz über Formular anlegen
- Datenmenge
- Datensatz über Formular (Statisch)
- Onboarding mit Installation
- Arbeitsaufgaben Datensatz E-Mails
- Email-Versendung
- Einladung zu einer App

* Reihenfolge 10

Details

Details

Filter

Bedingung

Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit diese Anweisung auf dem Datensatz ausgeführt wird.

Kompilierte Bedingung

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen Hilfe

Abb. 32: Anweisungen - E-Mail-Versendung

Aktionen erinnerung Beschriftung: Erinnerung Aktiv?: Ja Icon: alarm-clock

Anweisungen Anweisungstyp: Email-Versendung Reihenfolge: 10

Bearbeiten Felderbelegung

Volltextsuche Anzeige

Bitte Suchbegriff eingeben (Alle) Suchen Erweitert

›› 13 Felder gefunden Zeilen pro Seite 13 Los

	Beschriftung	Typ	Identifizierer	Aktion
1	Sprache (=de)	string/text	language	
2	Typ (=dataaction)	string/text	type	
3	An	string/text	to	
4	cc	string/text	cc	
5	bcc	string/text	bcc	
6	Betreff	string/text	subject	
7	Inhalt (Text)	string/text	body_text	
8	Inhalt (HTML)	string/text	body_html	
9	tpl_id (=r.app.id)	string/text	tpl_id	
10	dat_id (=r.id)	string/text	dat_id	
11	User (automatisch)	string/text	c_user	
12	Account E-Mail	string/text	account_from	
13	Email-Template-ID	string/text	eml_id_template	

CSV PDF

Abb. 33: Felderbelegung - E-Mail-Versendung

https://my.living-apps.de/waf_web/web/selectEntry.do?xist4c_eagw_coId=1153&template=2040&entry=5aaa35f6943dadca3ce02859

Abb. 34: Felderbelegung - E-Mail-Versendung - E-Mail-Template-ID

Aktion

Nichts ausgewählt
Wird hier nichts ausgewählt, so wird mit dem Feld keine Aktion durchgeführt.

Feld leeren
Der Feldinhalt wird gelöscht. Das ist nur beim Ändern von Datensätzen sinnvoll.

Feld auf Wert setzen
Das Feld übernimmt, nachdem die Aktion ausgeführt wurde, den hier eingegebenen Wert.

Ausdruck
Für die Formulierung des Ausdrucks wird *vSQL* verwendet. Dafür werden die Identifizierer der Felder benötigt. Diese finden Sie in der linken Menüleiste unter *Felder*. Auf die Werte kann dann zugegriffen werden mit: `r.v_identifier`.

Nach dem Speichern erscheint unterhalb des Feldes *Ausdruck* der kompilierte Ausdruck. Ist ein Fehler in der Formulierung, sehen Sie diesen hier.

In unserem Beispiel wurden folgende *Konfigurationen* vorgenommen.

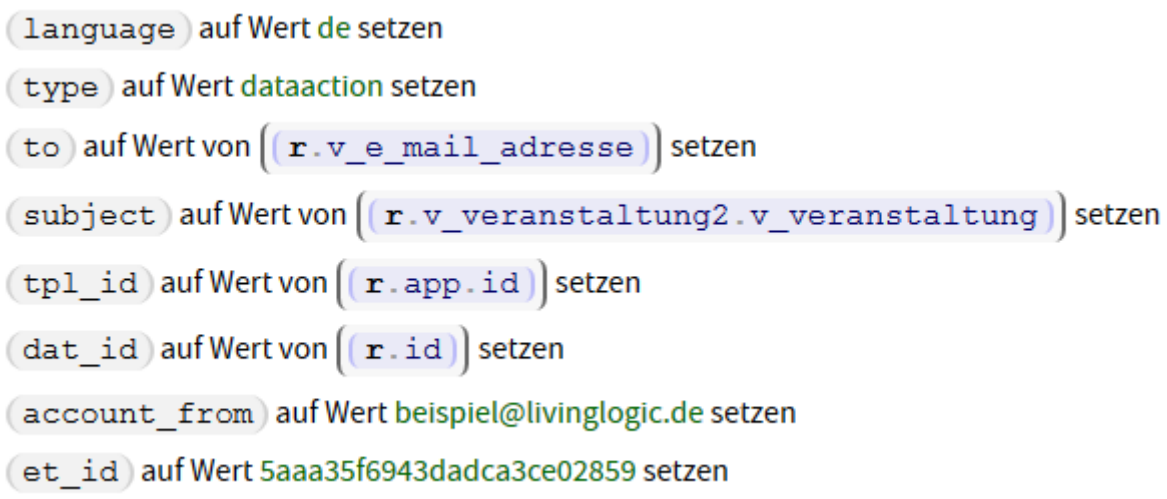

Abb. 35: Felderkonfiguration - E-Mail-Versendung

***language* auf Wert von "de" setzen**
Sprache ist Deutsch.

***type* auf Wert "dataaction" setzen**
Vorgegebener Wert

***to* auf Wert von `r.v_e_mail_adresse` setzen**
Empfänger der E-Mail. Greift auf die jeweilige E-Mail-Adresse der Teilnehmer zu.

***subject* auf Wert von "Erinnerung an " + r.v_veranstaltung2.v_veranstaltung setzen**
Schreibt in die Betreffzeile der E-Mail `Erinnerung an` und greift dann auf die jeweilige Veranstaltung zu, an die erinnert werden soll.

***tpl_id* auf Wert von `r.app.id` setzen**
Die App, von der aus die E-Mail verschickt wird. Hier muss als *Ausdruck* `r.app.id` eingegeben werden.

***dat_id* auf Wert von `r.id` setzen**
Datensatz für den die E-Mail verschickt wird. Hier muss bei *Ausdruck* `r.id` eingegeben werden.

***eml_id_template* auf Wert "5aaa35f6943dadca3ce02859" setzen**
Der E-Mail-Inhalt wird über ein E-Mail-Template eingefügt. Dafür muss hier die E-Mail-Template-ID als Wert eingegeben werden. Mehr dazu können Sie oben unter *E-Mail-Template-ID* nachlesen.

Ausgelöst werden kann die Aktion in der Liste des Datenmanagements der App *Anmeldung* unter *Daten* → *Liste*. Wobei die Aktion *Erinnerung* nur bei den Teilnehmern aktiv ist, die sich nicht wieder abgemeldet haben.

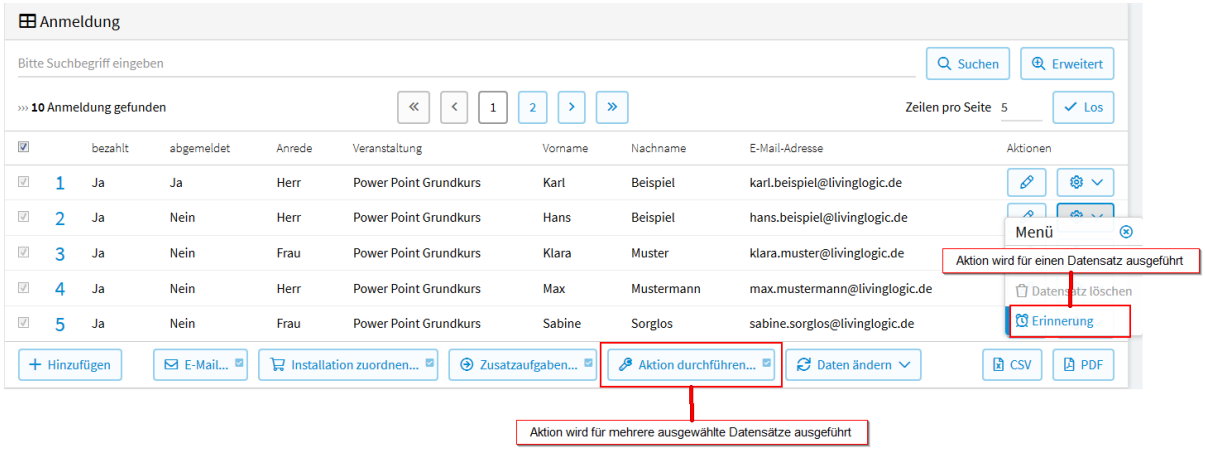

Abb. 36: Aktion „E-Mail-Versendung“ ausführen

Folgende E-Mail erhalten die Teilnehmer in unserem Beispiel nach Auslösen der Aktion:

```
Sehr geehrte Frau Sand,

wir möchten Sie erinnern, dass Sie sich zur Veranstaltung Word für Einsteiger
am 17.03.18 von 10:00 bis 16:00 Uhr,
in Bamberg, Kulturzentrum, Raum 105,
angemeldet haben.

Bei Rückfragen erreichen Sie uns unter 0951/123456789.

Mit freundlichen Grüßen
Ihr Veranstaltungsteam
```

18.2.7 Einladung zu einer App

Im folgenden Anwendungsbeispiel sollen die Personen in der App „Mitarbeiter“ zur Mitarbeit an den Apps „Anmeldung“ und „Veranstaltungen“ eingeladen werden. Der Vorteil dies über eine Aktion zu tun ist der, dass man nicht jeden einzeln einladen muss, sondern alle auf einmal, auch zu mehreren Apps, einladen kann.

Wählen Sie im Menü *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Aktionen*. Klicken Sie auf *Hinzufügen* um eine neue Aktion zu erstellen.

Im nun geöffneten Fenster werden folgende *Konfigurationen* vorgenommen.

Nach dem Speichern erscheint neben *Bearbeiten* der Tab *Anweisungen*. Nach *Hinzufügen* einer neuen Anweisung kann im nun geöffneten Fenster eingestellt werden, was durch die Aktion ausgeführt werden soll. Siehe *Anweisungen*.

Als *Anweisungstyp* wird *Einladung zu einer App* gewählt und bei *Reihenfolge* `10` eingegeben.

Nach dem Speichern der Anweisungen erscheint neben *Bearbeiten* der Tab *Felderbelegung*. Durch Klicken darauf sehen Sie folgende *Felder*. Diese sind bei jeder App gleich.

User-ID
: Dieses Feld wird automatisch mit dem angemeldete Benutzer gefüllt.

App-ID
: Wenn Sie eine App anklicken, sehen Sie in der URL hinter `apps/` eine Kombination aus Buchstaben und Zahlen:

Aktionen einladung Beschriftung: Einladung zu einer App Aktiv?: Ja

Bearbeiten Anweisungen

+ Hinzufügen Kopieren − Löschen Hilfe

* Identifizierer einladung
Eindeutige Kennung für diese Aktion

* Beschriftung Einladung zu einer App

Aktiv? ☑
Eine inaktive Aktion wird nicht angezeigt und kann nicht ausgeführt werden.

Reihenfolge 10
Die Aktions-Buttons werden im Datenmanagement nach dieser Zahl sortiert angezeigt.

* Berechtigung für alle Benutzer
Wer darf diese Aktion ausführen?

Beschreibung Personen zur Mitarbeit an einer oder mehreren Apps einladen.

Icon
Das Icon für den Aktions-Button im Datenmanagement.

Icon-Vorschau

Verwendung
- ☑ Aktions-Button im Datenmanagement für mehrere Datensätze
- ☑ Aktions-Button im Datenmanagement für einen einzelnen Datensatz
- ☐ Link für E-Mails
- ☐ Vor dem Anzeigen des Bearbeiten-Formulars ausführen
- ☐ Nach Änderung des Datensatzes ausführen (auch bei Import)
- ☐ Nach Anlegen des Datensatzes ausführen (auch bei Import)

Legt fest wo diese Aktion zur Verfügung stehen soll

Meldung
Meldung im Datenmanagement oder bei der Link-Ausführung wenn die Aktion erfolgreich ausgeführt wurde

Filter

Bedingung
Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit die Aktion auf dem Datensatz ausgeführt wird

Kompilierte Bedingung

Historie

Angelegt von Sabine Voigt
Angelegt am 13.07.2018 14:12:55

Geändert von Sabine Voigt
Geändert am 18.07.2018 11:44:55

+ Hinzufügen Kopieren − Löschen Hilfe

Abb. 37: Einladung zu einer App

Aktionen einladung Beschriftung: Einladung Aktiv?: Ja

Anweisungen

+ Hinzufügen

Speichern Abbrechen Hilfe

* Anweisungstyp
- Datensatz anlegen
- Datensatz verändern
- Arbeitsaufgabe automatisch anlegen
- Datensatz löschen
- Datensatz über Formular anlegen
- Datenmenge
- Datensatz über Formular (Statisch)
- Onboarding mit Installation
- Arbeitsaufgaben Datensatz E-Mails
- Email-Versendung
- Einladung zu einer App

* Reihenfolge 10

Details

Details

Filter

Bedingung

Hier können Sie eine Filter-Bedingung eingeben, die erfüllt sein muss, damit diese Anweisung auf dem Datensatz ausgeführt wird.

Kompilierte Bedingung

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen Hilfe

Abb. 38: Anweisungen - „Einladung zu einer App“

Aktionen einladung Beschriftung: Einladung Aktiv?: Ja

Anweisungen Einladung zu einer App Reihenfolge: 10

Bearbeiten Felderbelegung

Volltextsuche Anzeige (Alle) Suchen Erweitert

››› 7 Felder gefunden Zeilen pro Seite 13 Los

	Beschriftung	Typ	Identifizierer	Aktion
1	User - ID	string/text	c_user	
2	App -ID	string/text	tpl_uuid	
3	Vorname	string/text	firstname	
4	Nachname	string/text	lastname	
5	E-Mail/Account	string/text	email	
6	Berechtigungslevel (2-5)	string/text	rol_id	
7	Einzelberechtigungen als String	string/text	permissions	

PDF CSV

Abb. 39: Felderbelegung - „Einladung zu einer App“

https://my.living-apps.de/apps/da11219bb494aff28062f50f.htm

Abb. 40: App-ID

Dies ist die *App-UUID* (Universally Unique Identifier), der eindeutige Identifizierer Ihrer App. Die App-ID der App, zu der Sie zur Mitarbeit einladen wollen, muss hier angegeben werden. Bei der Einladung zu mehreren Apps geben Sie die App-IDs kommagetrennt ein.

Vorname
Vorname der Person, die Sie einladen wollen. Wählen Sie *Ausdruck* und greifen mit `r.v_vorname` auf die Werte des Feldes `Vorname` zu. Wobei `vorname` für den Identifizierer des Feldes `Vorname` steht.

Nachname
Nachname der Person, die Sie einladen wollen. Einen gleichbleibenden Wert (z. B. Mustermann) geben Sie bei *Feld auf Wert setzen* ein. Wählen Sie *Ausdruck*, können Sie auch hier auf die Felder Ihrer App zugreifen mit `r.v_nachname`. Wobei `nachname` für den Identifizierer des Feldes `Nachname` steht.

E-Mail/Account
E-Mail-Adresse der Person, die Sie einladen wollen. Wählen Sie *Ausdruck* und greifen mit `r.v_email` auf die Werte des E-Mail-Feldes Ihrer App zu. Wobei `email` für den Identifizierer des E-Mail-Feldes steht.

Berechtigungslevel (2-5)
Geben Sie hier an, welche Berechtigung die Mitarbeiter an der App haben sollen. Wobei `2` für *Minimal*, `3` für *Standard*, `4` für *Datenverarbeitung* und `5` für *Administrator* steht.

Einzelberechtigungen als String
Möchten Sie die App-Berechtigungen nicht als Rolle, sondern benutzerdefiniert vergeben, können Sie dies hier tun. Dafür wird jede einzelne Berechtigung jeweils mit `0` „nicht gewählt“ und mit `1` „gewählt“.

Applikationsrechte

#	Name	Berechtigung	Erlaubt
1	`app_edit`	Dateneingabe anpassen	den Formularwizard aufzurufen
2	`app_konfig`	Datenmanagement konfigurieren	Interne Felder definieren, Anzeige im Datenmanagement zu konfigurieren
3	`app_delete`	Applikation löschen	
4	`app_taskmanage`	Arbeitsaufgaben konfigurieren	Aufgabenwizard aufrufen, zu Aufgaben einladen
5	`app_data_edit`	Datenmanagement anzeigen	Daten sichten und editieren im Datenmanagement
6	`app_data_view`	Daten sichten	Daten sichten im Datenmanagement
7	`app_taskview`	Arbeitsaufgaben anzeigen	sichten/bearbeiten, der meinen Aufgaben zugeordneten Datensätze
8	`app_frontend`	Formular aufrufen	
9	`app_dataconnectex`	nicht mehr relevant!	
10	`app_performevalua`	Auswertung anzeigen	
11	`app_tocatalog`	Im Katalog veröffentlichen	
12	`app_dataimexport`	Daten importieren	
13	`app_mydata_edit`	meine Daten editieren	nach Login, die von mir erfassten Daten editieren
14	`app_mydata_view`	meine Daten sichten	nach Login, die von mir erfassten Daten ansehen
15	`app_user_admin`	Berechtigungen zuweisen	

Um z. B. die Einzelberechtigungen nur für `app_mydata_edit` und `app_mydata_view` zuzuweisen, müssten Sie bei *Feld auf Wert setzen* folgende Angaben machen:

```
000000000000110
```

Durch Klicken auf das zu ändernde Feld, sind folgende Konfigurationen möglich.

Aktion

Nichts ausgewählt
Wird hier nichts ausgewählt, so wird mit dem Feld keine Aktion durchgeführt.

Feld leeren
Der Feldinhalt wird gelöscht. Das ist nur beim Ändern von Datensätzen sinnvoll.

Feld auf Wert setzen
Das Feld übernimmt, nachdem die Aktion ausgeführt wurde, den hier eingegebenen Wert.

Ausdruck
Für die Formulierung des Ausdrucks wird *vSQL* verwendet. Dafür werden die Identifizierer der Felder benötigt. Diese finden Sie in der linken Menüleiste unter *Felder*. Auf die Werte kann dann zugegriffen werden mit: `r.v_identifier`.

Nach dem Speichern erscheint unterhalb des Feldes *Ausdruck* der kompilierte Ausdruck. Ist ein Fehler in der Formulierung, sehen Sie diesen hier.

In unserem Beispiel wurden folgende *Konfigurationen* vorgenommen.

```
tpl_uuid auf Wert "da11219bb494aff28062f50f, a6df949681835742ab27a967" setzen
firstname auf Wert von r.v_vorname setzen
lastname auf Wert von r.v_nachname2 setzen
email auf Wert von r.v_email_adresse setzen
rol_id auf Wert "5" setzen
```

Abb. 41: Felderkonfiguration - „Einladung zu einer App“

***tpl_uuid* auf Wert „da11219bb494aff28062f50f, a6df949681835742ab27a967“ setzen**
App-IDs der Apps „Anmeldung und Veranstaltung“

***firstname* auf Wert von *r.v_vorname* setzen**
Greift auf den Vornamen des eingeladenen Mitarbeiters zu.

***lastname* auf Wert von *r.v_nachname2* setzen**
Greift auf den Nachnamen des eingeladenen Mitarbeiters zu.

***email* auf Wert von *r.v_email_adresse* setzen**
Greift auf die E-Mail-Adresse des eingeladenen Mitarbeiters zu.

***rol_id* auf Wert „5“ setzen**
Weist den eingeladenen Mitarbeitern die Rolle als Administrator zu.

Ausgelöst werden kann die Aktion in der Liste des Datenmanagements der App *Mitarbeiter* unter *Daten* → *Liste*.

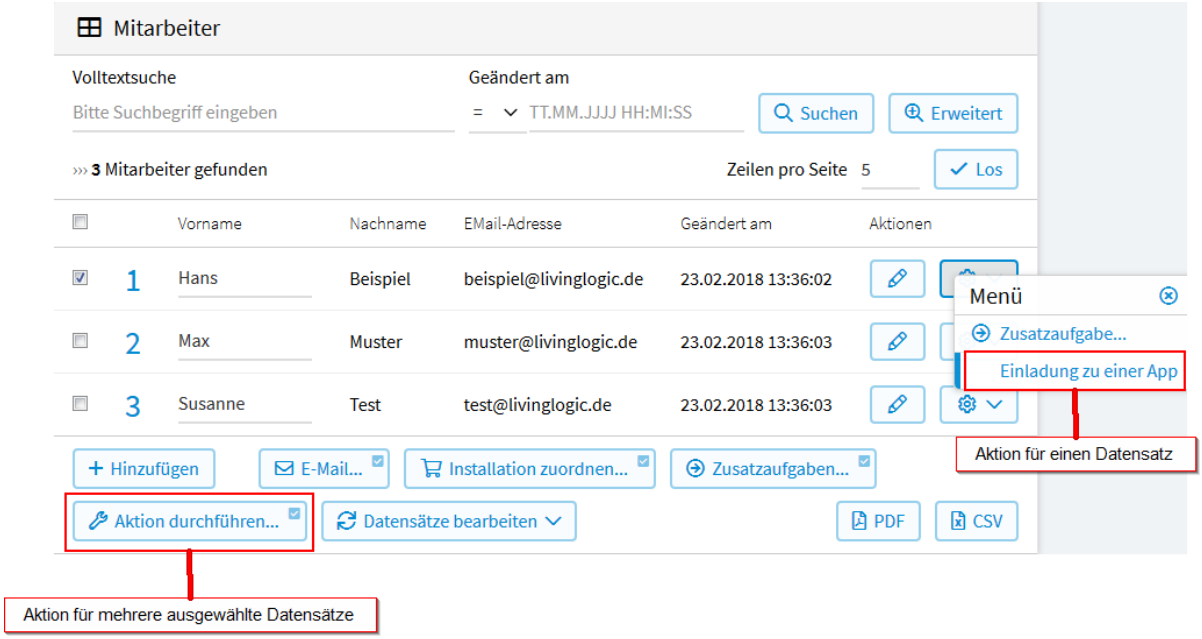

Abb. 42: Aktion „Einladung zu einer App“ ausführen

KAPITEL 19

Meldungen

In *Meldungen* werden Fehler- oder Log-Meldungen Ihrer LivingApps gespeichert.

Diese können aus den Anzeige-, Formular- oder E-Mail-Templates stammen, oder von anderen Teilen des System ausgelöst werden. Mögliche Auslöser so einer Meldung sind unter anderem folgende Fälle:

- Wenn Sie sich in einer Daten-Aktion in *vSQL-Ausdrücken* auf ein Feld beziehen, dieses Feld dann aber löschen, werden Sie beim Ausführen dieser Daten-Aktion eine Fehler-Meldung erhalten.
- Wenn jemand im Formulareditor ein geändertes Formular speichert, und dieses Formular AppLookup-Felder hat, auf deren Ziel-LivingApp aber dieser Benutzer keine Berechtigungen besitzt, wird ebenfalls eine Fehler-Meldung erzeugt.

Um die Fehler-Meldungen anzuschauen, wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken dann in der linken Menüspalte auf *Meldungen*.

KAPITEL 20

Konfiguration

Hier können allgemeine Konfigurationen in der aktuellen LivingApp vorgenommen werden.

20.1 Basisdaten

In *Basis-Daten* können Sie den Namen, die Beschreibung und das Icon der LivingApp ändern.

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Konfiguration* und dann auf *Basisdaten*. Im nun geöffneten Fenster können Sie die Änderung der Basisdaten vornehmen.

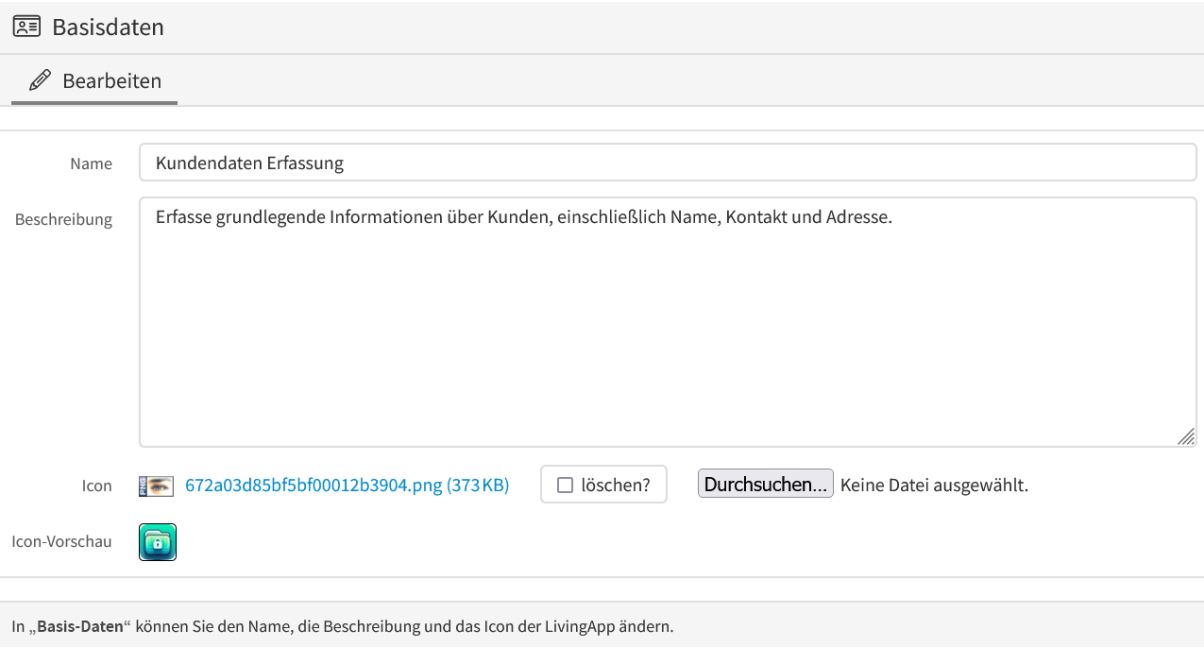

Abb. 1: Basis-Daten ändern

20.2 Bezeichnungen

In *Bezeichnungen* können Sie festlegen, wie die Dinge in Ihrer LivingApp bezeichnet werden (also z.B. „Kunden“ oder „Anmeldungen“ usw. statt nur „Datensätze“).

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Konfiguration* und dann auf *Bezeichnungen*. Im nun geöffneten Fenster können Sie die Änderung der Bezeichnungen vornehmen.

Wählen Sie zunächst das Geschlecht der Bezeichnung, die Sie verwenden möchten. Füllen Sie anschließend jedes einzelne Feld mit dem deklinierten Nomen Ihrer Bezeichnung.

Im Beispiel wurde anstatt „Datensatz“, die Bezeichnung „Kunde“ verwendet.

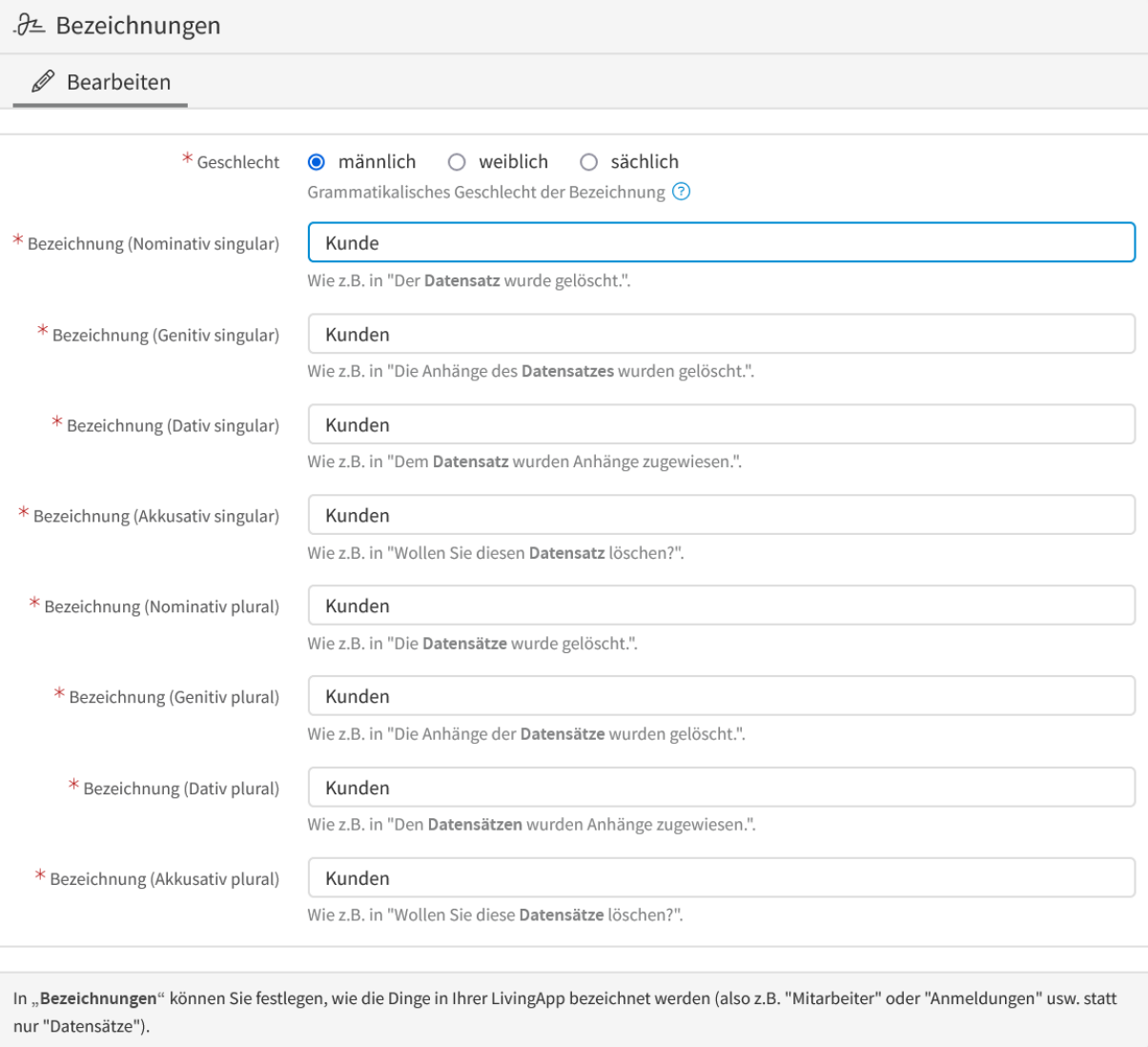

Abb. 2: Bezeichnung ändern

Beispiel vor der Änderung:

Beispiele nach der Änderung:

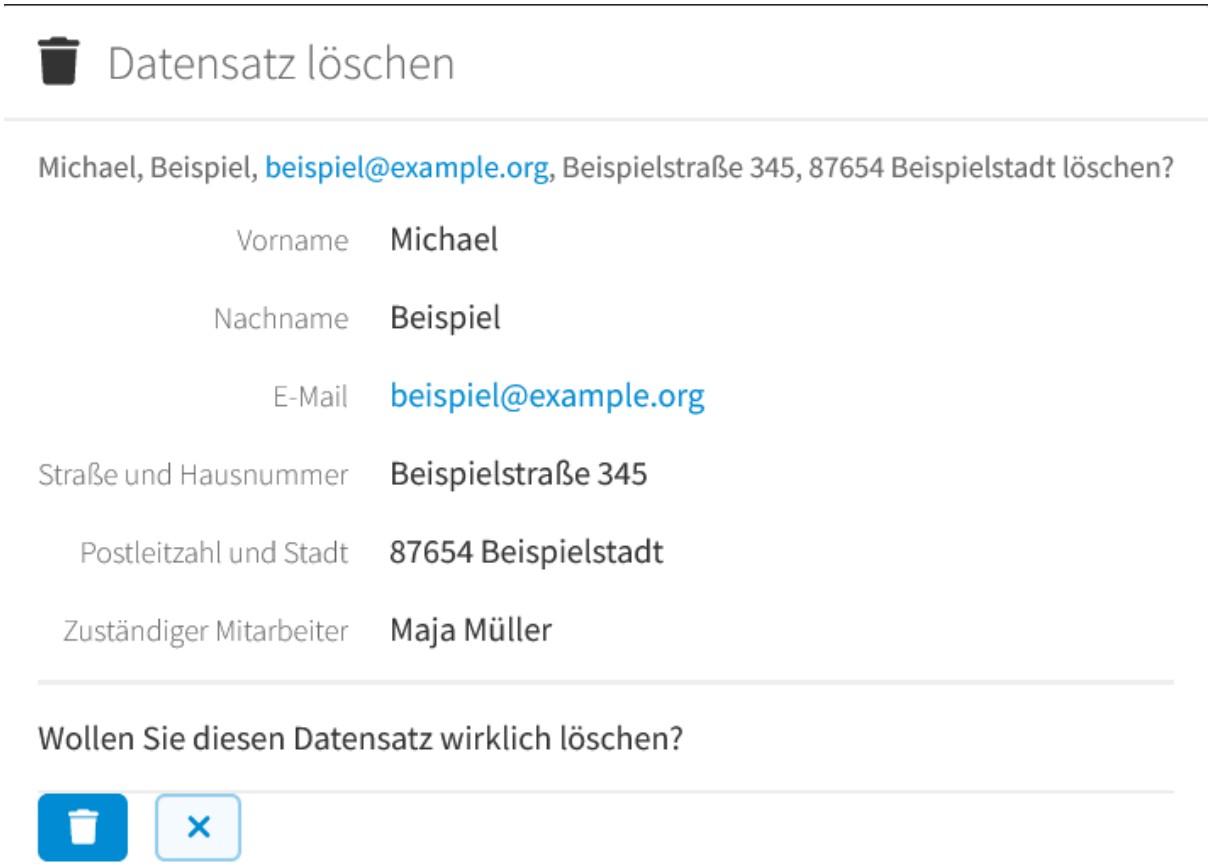

Abb. 3: Beispiel vorher

Kunden löschen

Michael, Beispiel, beispiel@example.org, Beispielstraße 345, 87654 Beispielstadt löschen?

Vorname	Michael
Nachname	Beispiel
E-Mail	beispiel@example.org
Straße und Hausnummer	Beispielstraße 345
Postleitzahl und Stadt	87654 Beispielstadt
Zuständiger Mitarbeiter	Maja Müller

Wollen Sie diesen Kunden wirklich löschen?

Abb. 4: Beispiel nach Änderung

Kunde gelöscht
Der Kunde **Michael, Beispiel, beispiel@example.org, Beispielstraße 345, 87654 Beispielstadt** wurde gelöscht!

Abb. 5: Beispiel

20.3 Felder

Unter *Felder* können Sie festlegen, ob, wie und an welcher Stelle die Felder im Datenmanagement unter *Daten* → *Liste* angezeigt werden.

Für Zahlenfelder können Sie die Summe in der Fußzeile der Datenliste anzeigen lassen. Bei `applookup`-Feldern können Sie konfigurieren, was passieren soll, wenn der referenzierte Datensatz gelöscht wird.

20.3.1 Erstellung

Wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken anschließend in der linken Menüleiste auf *Konfiguration* → *Felder*.

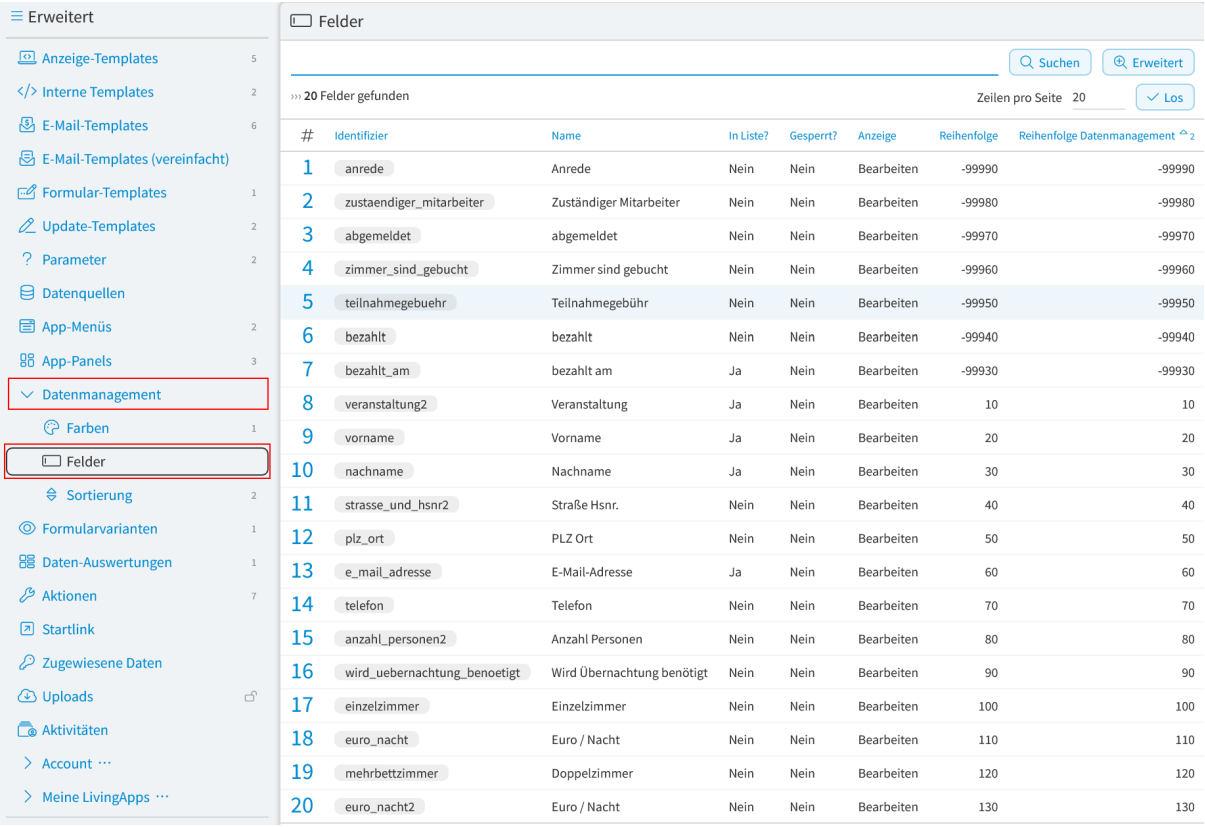

Abb. 6: Datenmanagement - Felder auswählen

Klicken Sie in der Übersicht auf das Feld, für das Sie Konfigurationen vornehmen möchten.

Es öffnet sich folgendes Fenster.

Allgemeine Daten

Identifizierer
Die eindeutige Kennung für dieses Feld (wird bei Erstellung der LivingApp automatisch angelegt).

Name
Name des Feldes (wird bei Erstellung der LivingApp vom Benutzer festgelegt). Der Name kann unter *Konfiguration* → *Eingabe* geändert werden und es kann eine abweichende Bezeichnung für das Datenmanagement eingegeben werden.

Typ
Der Feldtyp. Darüber können Sie mehr in der *Dokumentation der Feldtypen* nachlesen.

Reihenfolge
Reihenfolge der Felder, in der sie bei Erstellung der LivingApp angelegt wurden. Diese Rei-

Felder vorname Name Vorname Typ string/text Reihenfolge 10 1/8

Bearbeiten Interne Templates

Allgemeine Daten

Identifizier vorname
Eine eindeutige Kennung für dieses Feld. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

Name Vorname

Typ string/text

Reihenfolge 10
Die ursrpüngliche Reihenfolge, in der die Felder angelegt wurden.

Konfiguration für Datenmanagement und Anzeige-Templates

In Liste? ☑
Soll das Feld im Datenmanagement und in Anzeige-Templates in der Liste angezeigt werden?

Reihenfolge 10
Wenn Sie hier einen von „**Reihenfolge**“ abweichenden Wert eingeben, behält LivingApps im Datenmanagement und den Anzeige-Templates die dadurch festgelegte Reihenfolge bei.

Beschreibung
Sollte kurz die Bedeutung und Verwendung dieses Feldes erklären.

Konfiguration für Datenmanagement

* Anzeige ◉ Bearbeiten ○ Anzeigen ○ Verstecken
Soll das Feld im Datenmanagement angezeigt werden, bearbeitet werden können, oder komplett versteckt werden? Dies betrifft sowohl die Listen- als auch die Detailansicht.

Konfiguration für Anzeige-Templates

In Mobil-Liste? ☑
Soll das Feld in Anzeige-Templates in der Liste auf schmalen Bildschirmen angezeigt werden?

In Text-Anzeige? ☑
Soll das Feld in Anzeige-Templates anzeigt werden, wenn der Datensatz als einzeiliger Text angezeigt wird?

Pflichtfeld? ◉ (Nichts ausgewählt) ○ Ja ○ Nein
Muß in diesem Feld ein Wert eingetragen/ausgewählt werden?

Allgemeine Konfiguration

Gesperrt? ☐
Ein gesperrtes Feld kann im Formular-Editor nicht gelöscht werden und dessen Typ kann nicht geändert werden.

Abb. 7: Felder konfigurieren

henfolge kann nicht geändert werden. Die Felder, deren Reihenfolge mit - beginnt, sind private Felder, d. h. sie sind nicht im Eingabe-Formular sichtbar.

Konfiguration für Datenmanagement und Anzeige-Templates

In Liste?
Setzen Sie das Häkchen, wenn dieses Feld in der Datenliste im Datenmanagement und in Anzeige-Templates angezeigt werden soll. (Funktioniert nicht, wenn bei *Anzeige Verstecken* gewählt wurde.)

***Reihenfolge* (Datenmanagement)**
Möchten Sie, dass Ihre Felder in der Datenliste in einer anderen Reihenfolge angezeigt werden, können Sie dies hier festlegen. Geben Sie dafür für jedes Feld entsprechende Zahlen in aufsteigender oder absteigender Reihenfolge ein.

Um dies zurückzusetzen, sodaß die Reihenfolge im Datenmangement wieder der Reihenfolge im Eingabeformular folgt, deaktivieren Sie dies im Erweitert-Menü unter *Konfiguration → Feldreihenfolge beibehalten?*.

Beschreibung
Hier können Sie kurz die Bedeutung und Verwendung dieses Feldes erklären.

Konfiguration für Datenmanagement

Anzeige
Diese Konfiguration betrifft sowohl die Datenliste als auch die Detailansicht.

Bearbeiten
Das Feld kann bearbeitet (geändert) werden.

Anzeigen
Das Feld wird angezeigt (kann nicht geändert werden).

Verstecken
Das Feld ist nicht sichtbar.

***Fußzeile* (nur bei Zahlenfeldern)**
Soll für Zahlenfelder in der Fußzeile etwas angezeigt werden?

Nichts ausgewählt
Für dieses Feld wird nichts in der Fußzeile der Datenliste angezeigt.

Summe
Für dieses Feld wird die Summe in der Datenliste in der Fußzeile angezeigt.

Konfiguration für Anzeige-Templates

In Mobil-Liste?
Legen Sie hier fest, ob dieses Feld in der Liste auf schmalen Bildschirmen (Handy) angezeigt werden soll.

In Text-Anzeige?
Legen Sie hier fest, ob das Feld in Anzeige-Templates anzeigt werden soll, wenn der Datensatz als einzeiliger Text angezeigt wird?

Pflichtfeld?
Soll dieses Feld ein Pflichtfeld sein?

Diese Konfiguration ist nur relevant wenn das Feld außerhalb eines normalen Eingabeformulars bearbeitet werden soll (wie z.B. in einem Anzeige-Template). In einem Fomular wird dagegen die Konfiguration in der entsprechenden Formular-Variante verwenden.

Wenn Sie hier nichts auswählen, so wird hierfür die Konfiguration aus der ersten Formularvariante verwendet.

Allgemeine Konfiguration

Gesperrt?
Ein gesperrtes Feld kann im Formulareditor unter *Konfiguration → Eingabe* nicht gelöscht werden und dessen Typ kann nicht geändert werden.

***Löschen* (nur bei AppLookup-Feldern)**
Was soll mit den untergeordneten Datensätzen passieren, wenn ein Haupt-Datensatz gelöscht wird?

Nicht erlauben
Das Löschen des referenzierten Datensatzes führt zu einer Fehlermeldung.

Diesen Datensatz löschen
Beim Löschen des referenzierten Datensatzes wird dieser Datensatz ebenfalls gelöscht.

Referenz löschen
Beim Löschen des referenzierten Datensatzes wird diese Referenz gelöscht (d.h. dieses Feld referenziert dann keinen Ziel-Datensatz mehr).

Die oben genannten Konfigurationen können Sie auch direkt in der Übersicht der Felder vornehmen. Das hat den Vorteil, dass sie z. B. bei Änderung der Reihenfolge, einen besseren Überblick behalten.

Klicken Sie dafür direkt in der Übersicht der Felder auf die entsprechende Spaltenüberschrift (*In Liste (DM/Templates)?*, *Reihenfolge (DM/Templates)*, *Beschreibung*, *Anzeige (DM)*, *Fußzeile (DM)*, *In Mobil-Liste (Templates)?*, *In Text-Anzeige (Templates) Pflichtfeld (Templates)?*, *Gesperrt?*, *Löschen*) und wählen *Bearbeiten ein* aus. Nehmen Sie nun die Änderungen direkt in der Übersicht vor und klicken zum Schluss auf *Alle speichern*.

20.3.2 Anwendungsbeispiele

In den folgenden drei Anwendungsbeispielen wird wieder auf den Anwendungsfall aus Kapitel 1 zurückgegriffen.

- Sie möchten in der Datenliste in der 1. Spalte anzeigen lassen, ob die Teilnehmer schon bezahlt haben.

  Wählen Sie dafür im Menü *Konfiguration → Erweitert* und klicken anschließend in der linken Menüleiste auf *Konfiguration → Felder*.

  Klicken Sie nun in der *Übersicht der Felder* auf das Feld `bezahlt` und nehmen dann folgende Konfigurationen vor.

  Setzen Sie das Häkchen bei *In Liste?* und geben Sie bei *Reihenfolge Datenmanagement* eine Zahl ein, die kleiner ist als die Zahl des Feldes, das momentan an 1. Stelle in der Liste steht. Klicken Sie zum Schluss auf *Alle Speichern*.

  Das Ergebnis sehen Sie, wenn Sie im Menü *Daten → Liste* wählen.

- Sie möchten sich die Anzahl der gebuchten Einzel- und Doppelzimmer als Summe in der Fußzeile der Datenliste anzeigen lassen. (Nur möglich, wenn die Felder Einzelzimmer und Doppelzimmer als Zahlenfelder angelegt sind.)

  Wählen Sie dafür im Menü *Konfiguration → Erweitert* und klicken anschließend in der linken Menüleiste auf *Konfiguration → Felder*.

  Klicken Sie nun in der *Übersicht der Felder* auf die Spaltenüberschrift *Fußzeile* und klicken auf *Bearbeiten ein*. Wählen Sie dann jeweils für die Felder „Einzelzimmer“ und „Doppelzimmer“ *Summe* aus und klicken auf *Alle speichern*.

  Das Ergebnis sieht dann so aus:

- Sie möchten, dass wenn ein Hauptdatensatz gelöscht wird, nur diese Referenz gelöscht wird, nicht alle untergeordneten Datensätze. (Betrifft nur AppLookup-Felder).

  In unserem Beispiel ist die LivingApp „Veranstaltungen“ über ein AppLookup-Feld mit der LivingApp „Mitarbeiter“ verbunden. Wenn ein zuständiger Mitarbeiter gelöscht wird, soll nur diese Referenz gelöscht werden und nicht der ganze Datensatz. Das heißt, das Feld Mitarbeiter ist danach in den entsprechenden Veranstaltungs-Datensätzen nicht mehr gefüllt.

  Wählen Sie dafür im Menü *Konfiguration → Erweitert* und klicken anschließend in der linken Menüleiste auf *Konfiguration → Felder*.

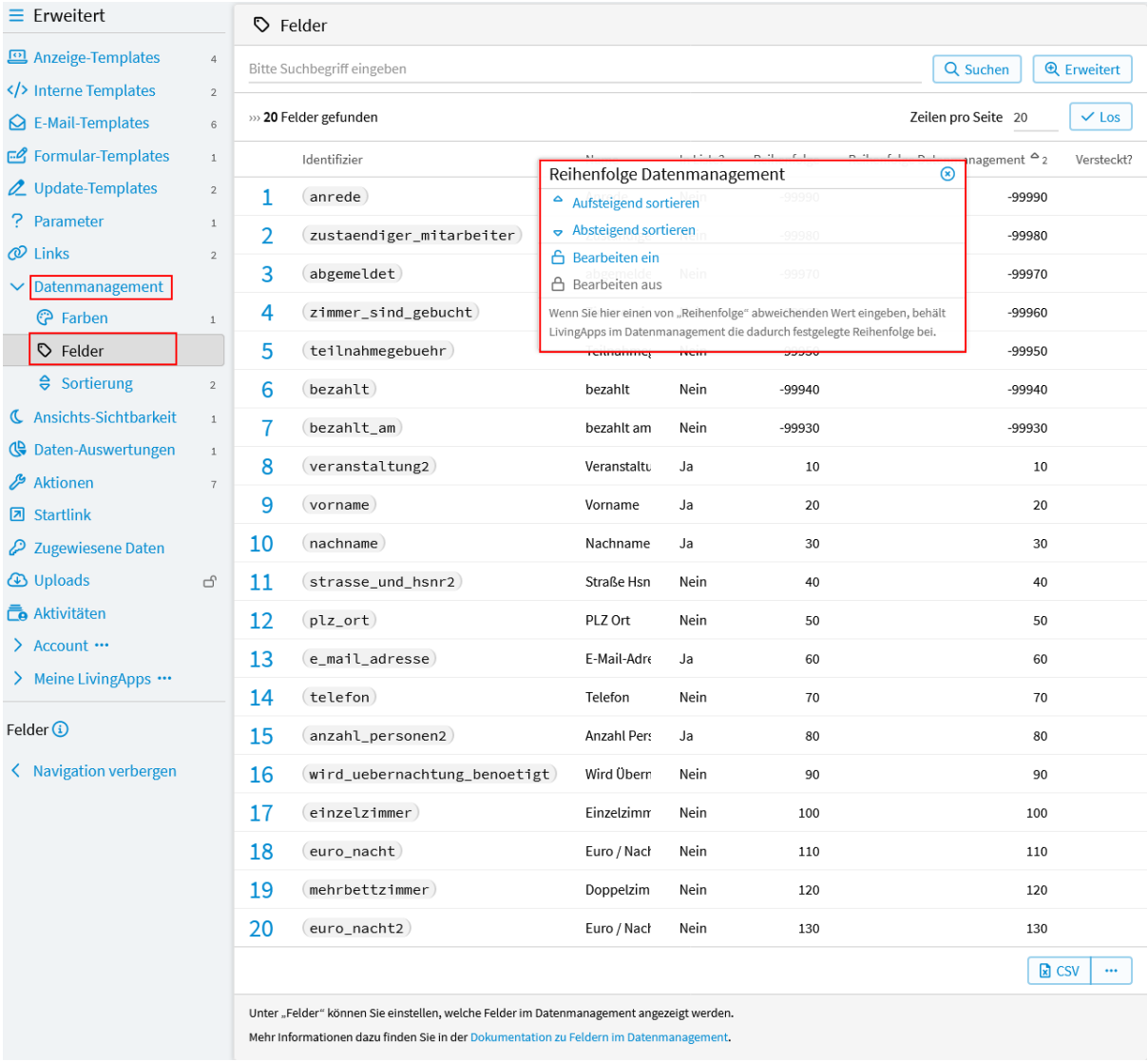

Abb. 8: Datenmanagement - Felder in Übersicht konfigurieren

Felder bezahlt Name: bezahlt In Liste?: Nein Gesperrt?: Nein Anzeige: Bearbeiten Reihenfolge: -99940 R 6/20

Bearbeiten

Speichern Wiederherstellen Hilfe

Identifizier bezahlt
Eine eindeutige Kennung für dieses Feld. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

Name bezahlt

In Liste? ☑
Soll das Feld im Datenmanagement in der Liste angezeigt werden (wenn bei „Anzeige“ nicht „Versteckt“ gewählt wurde)?

Gesperrt? ☐
Ein gesperrtes Feld kann im Formbuilder nicht gelöscht werden und dessen Typ kann nicht geändert werden.

* Anzeige ◉ Bearbeiten ○ Anzeigen ○ Verstecken
Soll das Feld im Datenmanagement angezeigt werden, bearbeitet werden können, oder komplett versteckt werden? Dies betrifft sowohl die Listen- als auch die Detailansicht.

Reihenfolge -99940
Die ursrpüngliche Reihenfolge, in der die Felder angelegt wurden.

Reihenfolge Datenmanagement 5
Wenn Sie hier einen von „Reihenfolge“ abweichenden Wert eingeben, behält LivingApps im Datenmanagement die dadurch festgelegte Reihenfolge bei.

Abb. 9: Felder - In Liste und Reihenfolge

Anmeldung

Volltextsuche
Bitte Suchbegriff eingeben

Geändert am
= TT . MM . JJJJ , -- : -- : -- Suchen Erweitert

›› 11 Anmeldung gefunden « ‹ 1 2 3 › » Zeilen pro Seite 5 Los

#	bezahlt	Veranstaltung	Vorname	Nachname	E-Mail-Adresse	Geändert am	Aktionen
1	Ja	Datum: 26.03.2017, Veranstaltung: Grundlagen Computerwissen (Aufbaukurs)	Karl	Beispiel	karl.beispiel@example.org	02.06.2020 10:41:44	
2	Ja	Datum: 25.03.2017, Veranstaltung: Grundlagen Computerwissen (Grundkurs)	Hans	Beispiel	hans.beispiel@example.org	02.06.2020 10:41:44	
3	Ja	Datum: 20.07.2032, Veranstaltung: Kommunikation/ Gesprächsführung im Personalwesen	Sabine	Voigt	voigt.sabine@gmail.com	08.11.2024 11:39:27	
4	Ja	Datum: 06.05.2017, Veranstaltung: Power Point Grundkurs	Klara	Muster	klara.muster@example.org	02.06.2020 10:41:44	
5	Ja	Datum: 17.08.2022, Veranstaltung: Power Point Grundkurs	Erika	Muster	erika.muster@gmail.com	08.11.2024 10:04:45	

Abb. 10: Felder - In Liste und Reihenfolge - Ergebnis

Anmeldung

Volltextsuche
Bitte Suchbegriff eingeben

Geändert am
= TT.MM.JJJJ, --:--:-- Suchen Erweitert

›› 11 Anmeldung gefunden

Zeilen pro Seite 11 Los

#	bezahlt	Vorname	Nachname	Anzahl Personen	Wird Übernachtung benötigt?	Einzelzimmer	Doppelzimmer	Geändert am	Aktionen
1	Ja	Karl	Beispiel	2	Ja	0	1	12.12.2024 09:00:39	
2	Ja	Hans	Beispiel	4	Nein	0	0	12.12.2024 09:00:39	
3	Ja	Sabine	Voigt	1	Ja	1	0	12.12.2024 08:44:58	
4	Ja	Klara	Muster	4	Nein	0	0	02.06.2020 10:41:44	
5	Ja	Erika	Muster	10	Ja	0	5	12.12.2024 09:00:39	
6	Ja	Max	Mustermann	10	Nein	0	0	08.11.2024 10:04:45	
7	Ja	Sabine	Sorglos	1	Ja	1	0	02.06.2020 10:41:44	
8	Ja	Peter	Test	2	Ja	2	0	12.12.2024 09:00:39	
9	Ja	Anne	Hinz	1	Nein	0	0	08.11.2024 10:04:45	
10	Ja	Silvia	Sand	2	Ja	0	1	27.03.2020 15:29:00	
11	Ja	Peter	Lustig	10	Nein	0	0	08.11.2024 10:04:45	
						4	7		

Summe der Felder Einzelzimmer und Doppelzimmer in der Fußzeile

+ Hinzufügen | E-Mail... | Installation zuordnen... | Zusatzaufgaben... | Aktion durchführen... | Datensätze bearbeiten ···

Abb. 11: Felder - Fußzeile, Summe - Ergebnis

Klicken Sie nun in der *Übersicht der Felder* auf die Spaltenüberschrift *Löschen* und klicken auf *Bearbeiten ein*. Wählen Sie dann für das Applookup-Feld „Zuständiger Mitarbeiter“ *Referenz löschen* aus und klicken auf *Alle speichern*. Damit ist ist Konfiguration abgeschlossen.

20.4 Startlink

In *Startlink* können Sie festlegen zu welcher Seite der Benutzer kommt, wenn er diese Applikation in der App-Übersichtsseite anklickt.

Wählen Sie dafür im Menü *Konfiguration → Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Konfiguration –> Startlink*. Im nun geöffneten Bereich kann ein Startlink neu angelegt, geändert oder zurückgenommen werden.

Startlink

Bearbeiten

Startlink Detailsicht
- (Nichts ausgewählt)
- Menü: Daten ⟶ Eingabe
- Menü: Daten ⟶ Liste
- Menü: Daten ⟶ Auswertung
- Menü: Daten ⟶ Import
- Menü: Aufgaben
- Menü: Konfiguration ⟶ Eingabe
- Menü: Konfiguration ⟶ Liste
- Menü: Konfiguration ⟶ Aufgaben
- Menü: Konfiguration ⟶ Berechtigungen
- Menü: Konfiguration ⟶ Erweitert
- Menü: Links ⟶ Teilnehmerliste

Abb. 12: Startlink erstellen

Menü: ***Daten*** → ***Eingabe***
Setzt die eingebettete Dateneingabe als Startlink.

Menü: ***Daten*** → ***Liste***
Setzt die Datenliste als Startlink.

Menü: ***Daten*** → ***Auswertung***
Setzt die automatisch bereitgestellten Auswertungen der Daten als Startlink.

Menü: ***Daten*** → ***Import***
Setzt die Import-Seite als Startlink.

Menü: ***Aufgaben***
Setzt die Aufgaben-Seite als Startlink. Nur möglich, wenn Aufgaben für diese App angelegt sind.

Menü: ***Konfiguration*** → ***Eingabe***
Setzt die Konfiguration der Eingabe (den Formbuilder) als Startlink.

Menü: ***Konfiguration*** → ***Aufgaben***
Setzt die Konfiguration der Arbeitsaufgaben als Startlink.

Menü: ***Konfiguration*** → ***Liste***
Setzt die Konfiguration der Datenliste als Startlink.

Menü: ***Konfiguration*** → ***Berechtigungen***
Setzt die Konfiguration der Berechtigungen als Startlink.

MenÜ: ***Konfiguration*** → ***Erweitert***
Setzt die Erweitert-Konfiguration als Startlink.

Menü: ***Links***
Haben Sie in Ihrer LivingApp unter *App-Menüs* Links zu z. B. Anzeigetemplates erstellt, können diese ebenfalls als Startlink verwendet werden.

Wählen Sie einen beliebigen *Startlink* aus und klicken Sie auf *Speichern*. Testen Sie den Startlink, indem Sie über die obere Navigation zurück zu *Meine Apps* gehen und die eben konfigurierte App durch einen Klick auf das Icon öffnen.

20.5 Zugewiesene Daten

In *Zugewiesene Daten* können Sie festlegen, welche Bedingung erfüllt sein muß, damit ein Datensatz einem Benutzer zugewiesen ist. Wenn Sie hier keine Bedingung eingeben, wird standardmäßig `r.createdby.id == user.id` verwendet, d.h. der Datensatz ist dem Benutzer zugewiesen, wenn er ihn angelegt hat.

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Konfiguration* → *Zugewiesene Daten*. Im nun geöffneten Bereich können Sie *die Bedingung eingeben*, die erfüllt sein muss, dass ein Datensatz einem Benutzer zugewiesen wird, auch wenn er ihn nicht angelegt hat. Für das *Formulieren der Bedingung* wird *vSQL* verwendet. Es stehen die *Variablen* `r` (zu testender Datensatz), `app` (die App zu der diese Konfiguration gehört) und `user` (eingeloggter Benutzer) zur Verfügung.

So können Sie z. B. in einer App „Veranstaltungen“, die zuständigen Mitarbeiter zu den Veranstaltungen im Feld „E-Mail-Verteiler“ erfassen. Um zu erreichen, dass die Mitarbeiter nur die Datensätze zu den Veranstaltungen sehen und bearbeiten können, denen sie zugewiesen sind, muss folgende Bedingung eingegeben werden:

```
r.v_e_mail_verteiler == user.email
```

Wobei `e_mail_verteiler` für den Identifizierer des E-Mail-Feldes steht. `user.email` ist die E-Mail-Adresse des eingeloggten Benutzers.

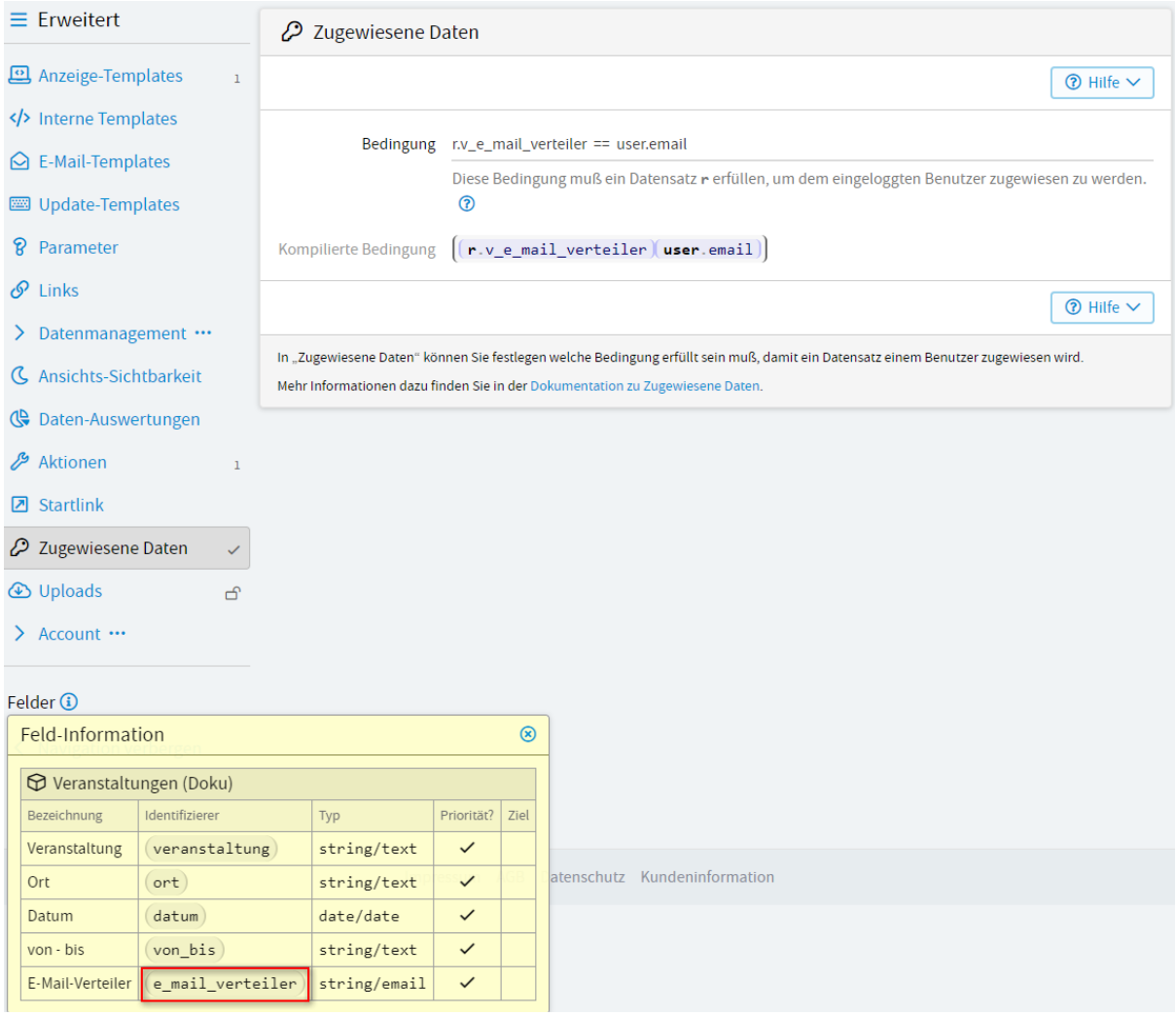

Abb. 13: Zugewiesene Daten - Bedingung

20.6 Uploads schützen?

In *Uploads schützen* können Sie festlegen, ob der Benutzer eingeloggt sein und Zugriffsrechte auf die App besitzen muss, um Dateien, die in dieser App hochgeladen wurden, downloaden zu können, oder ob die Dateien frei zugänglich sein sollen.

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Konfiguration* → *Uploads schützen?*.

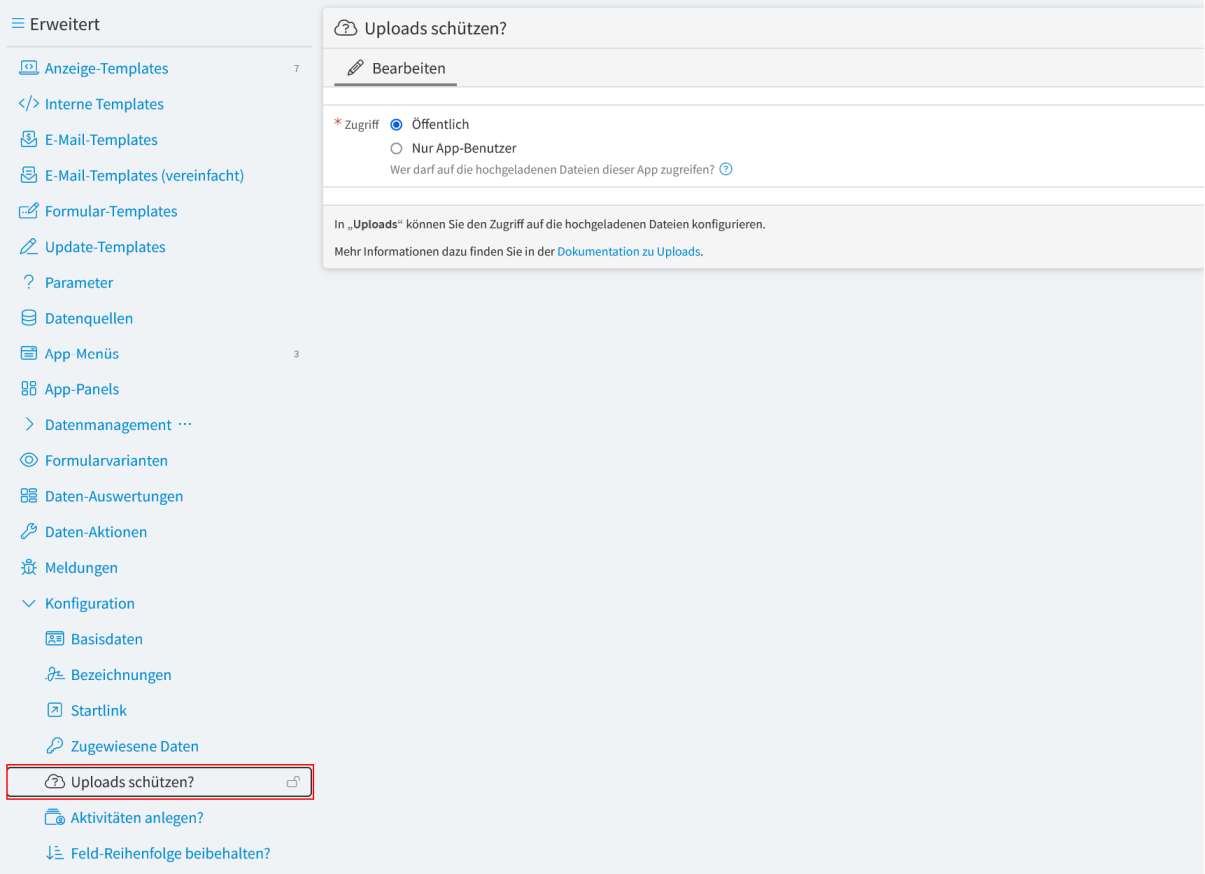

Abb. 14: Uploads schützen?

Öffentlich

Wählen Sie *Öffentlich*, wenn Ihre Dateien frei zugänglich sein sollen. Das heißt, Ihre Dateien können von jedem jederzeit heruntergeladen werden. Dies ist z.B. dann sinnvoll, wenn Sie ein Bild in einer Email (oder einem öffentlichen Anzeige-Template) verwenden wollen.

Nur App-Benutzer

Hier können die Dateien von einem eingeloggten Benutzer heruntergeladen werden, wenn er Zugriffsrechte auf der App besitzt. Dies ist dann sinnvoll, wenn alle Benutzer dieser App gemeinsam die Datensätze der App bearbeiten und hochgeladene Dokumente einsehen können sollen, aber diese vor Zugriff von Dritten geschützt sein sollen (z. B. Bewerbungsunterlagen).

Desweiteren kann eine Datei über die LivingAPI-Funktion *globals.response.send_file()* ausgeliefert werden. Dies ermöglicht es, jeden beliebigen Zugriffsschutz für nicht-eingeloggte oder andere Benutzer im Anzeige-Template selbst zu implementieren.

20.7 Aktivitäten anlegen?

Aktivitäten stellen Ereignisse dar, die im Living-Apps-System stattgefunden haben, von denen der eingeloggte Benutzer verständigt werden soll.

Diese Aktivitäten werden aktuell intern gespeichert und sind nur über eine System-App zugänglich (siehe *System-App „Aktivitäten“*), d.h. Aktivitäten können aktuell nur über Anzeige-Templates angezeigt werden.

Damit Informationen zu den Aktivitäten angelegt werden, müssen Sie dies entsprechend konfigurieren. Wählen Sie dazu im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Konfiguration –> Aktivitäten anlegen*.

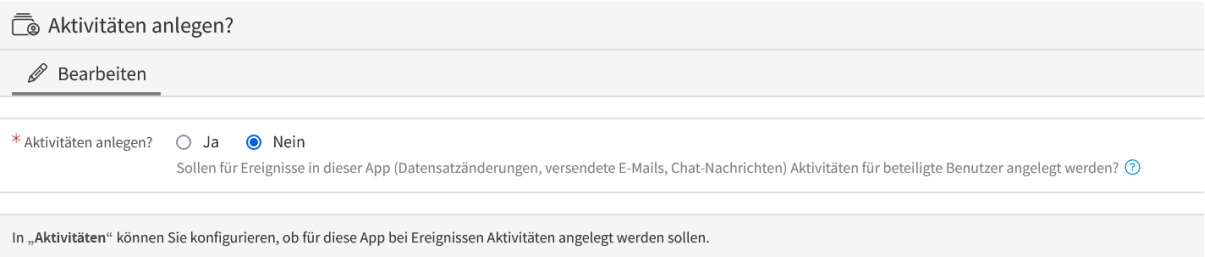

Abb. 15: Erzeugen von Aktivitäten konfigurieren

Ja
: Für Ereignisse in dieser LivingApp werden Aktivitäts-Informationen angelegt.

Nein
: Er werden keine Aktivitäts-Informationen angelegt.

20.8 Feldreihenfolge beibehalten?

In *Feld-Reihenfolge beibehalten?* können Sie festlegen, ob eine geänderte Feld-Reihenfolge, die Sie unter *Konfiguration - Erweitert - Datenmanagement - Felder* festgelegt haben, beibehalten werden soll, auch wenn Sie die LivingApp ändern.

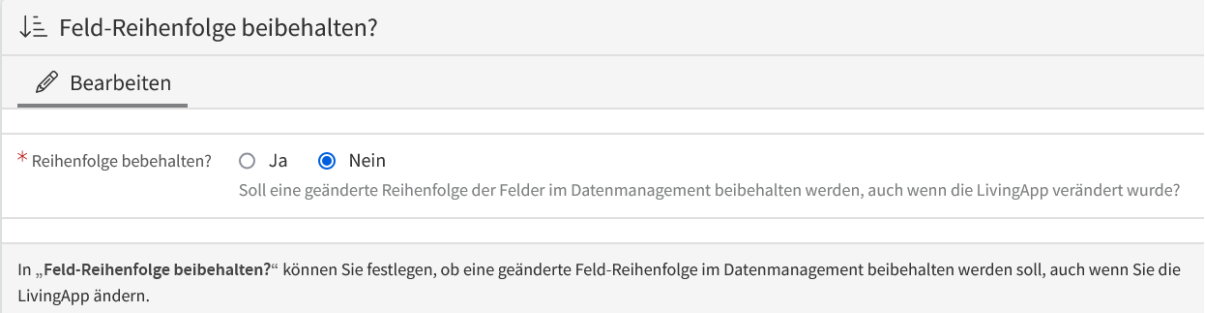

Abb. 16: Feldreihenfolge beibehalten

KAPITEL 21

Account

Unter *Account* können Konfigurationen vorgenommen werden, die nicht an die jeweilige App gebunden sind, sondern für Ihr gesamtes LivingApps-Konto gelten.

Um *Einstellungen des Accounts* vorzunehmen, wählen Sie im Menü einer beliebigen App *Konfiguration* → *Erweitert* und anschließend in der linken Menüleiste *Account*. Wählen Sie dann den jeweiligen Menüpunkt aus.

21.1 Benutzer-Menüs

In *Benutzer-Menüs* können Sie Menüs und Menüeinträge anlegen, die das Navigationsmenü auf der App-Übersichtsseite oder der Detailseite aller Ihrer LivingApps erweitern.

Im Gegensatz zu *App-Menüs* werden *Benutzer-Menüs* in allen Ihren Apps angezeigt (es sei denn, Sie ordnen das Menü einer bestimmten App zu).

Menüs, die sie hier anlegen, werden nur für Sie angezeigt, nicht für andere Benutzer.

21.1.1 Erstellung von Benutzer-Menüs

Wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Account* und *Benutzer-Menüs* und dann auf *Neuen Link oder neues Menü zur Menüleiste hinzufügen*.

Im nun geöffneten Fenster können folgende Konfigurationen vorgenommen werden.

LivingApp
: Die App in deren Kontext dieses Menü angezeigt werden soll. Wenn Sie hier keine App auswählen, wird dieses Menü bei all Ihren Apps dargestellt.

Identifizierer
: Der eindeutige Name des Links. Der Name darf nur Buchstaben, Ziffern und „_“ beinhalten.

Beschriftung
: Der Name des Menü-Eintrages.

Icon
: Das Icon für den Menü-Eintrag. Verwendet werden können alle Namen aus dem FontAwesome-Icon-Set.

Anzeigen auf
: Gibt an, wo das Menü erscheint. Der Link kann sowohl auf der

  App-Übersichtsseite?

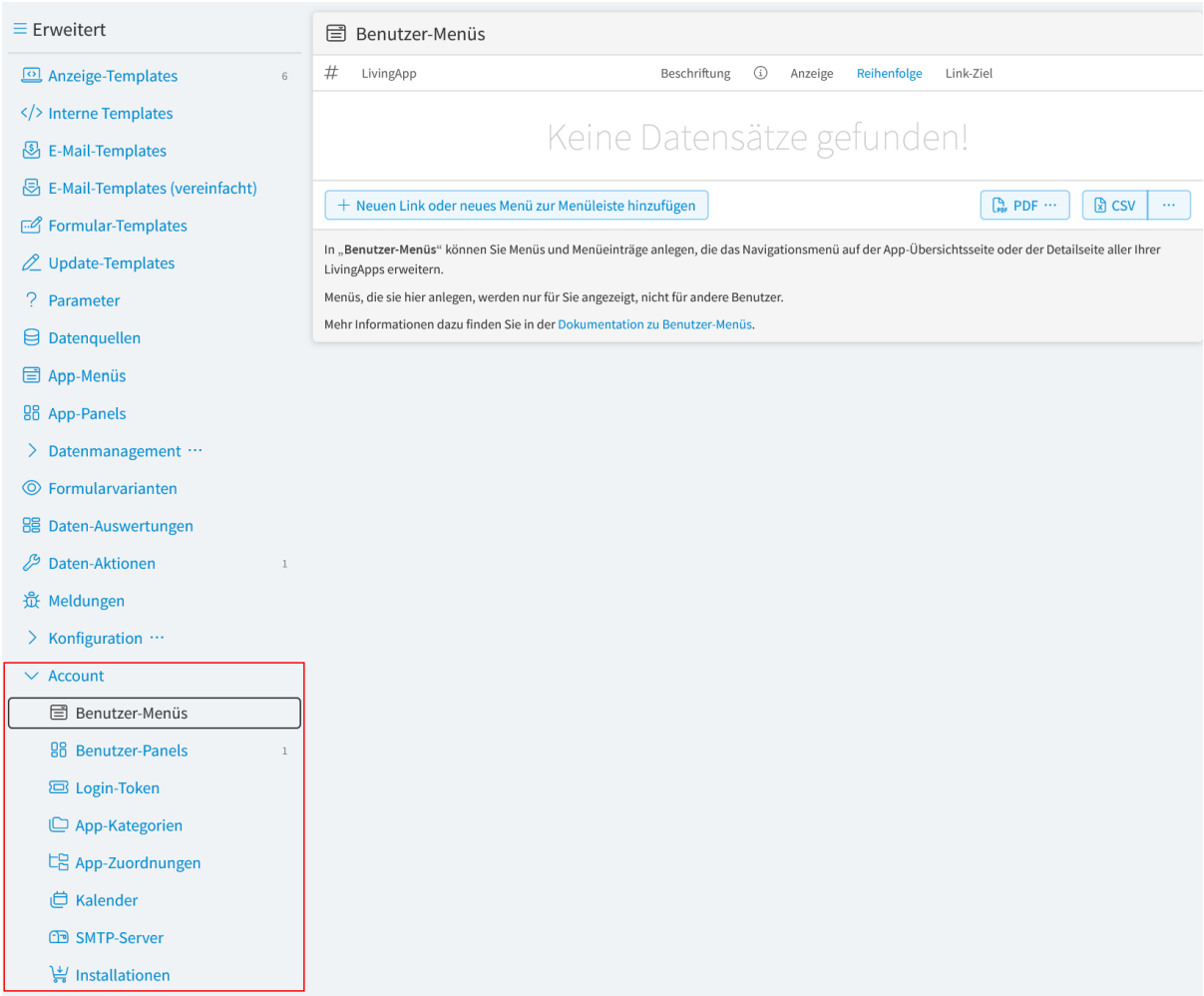

Abb. 1: Account

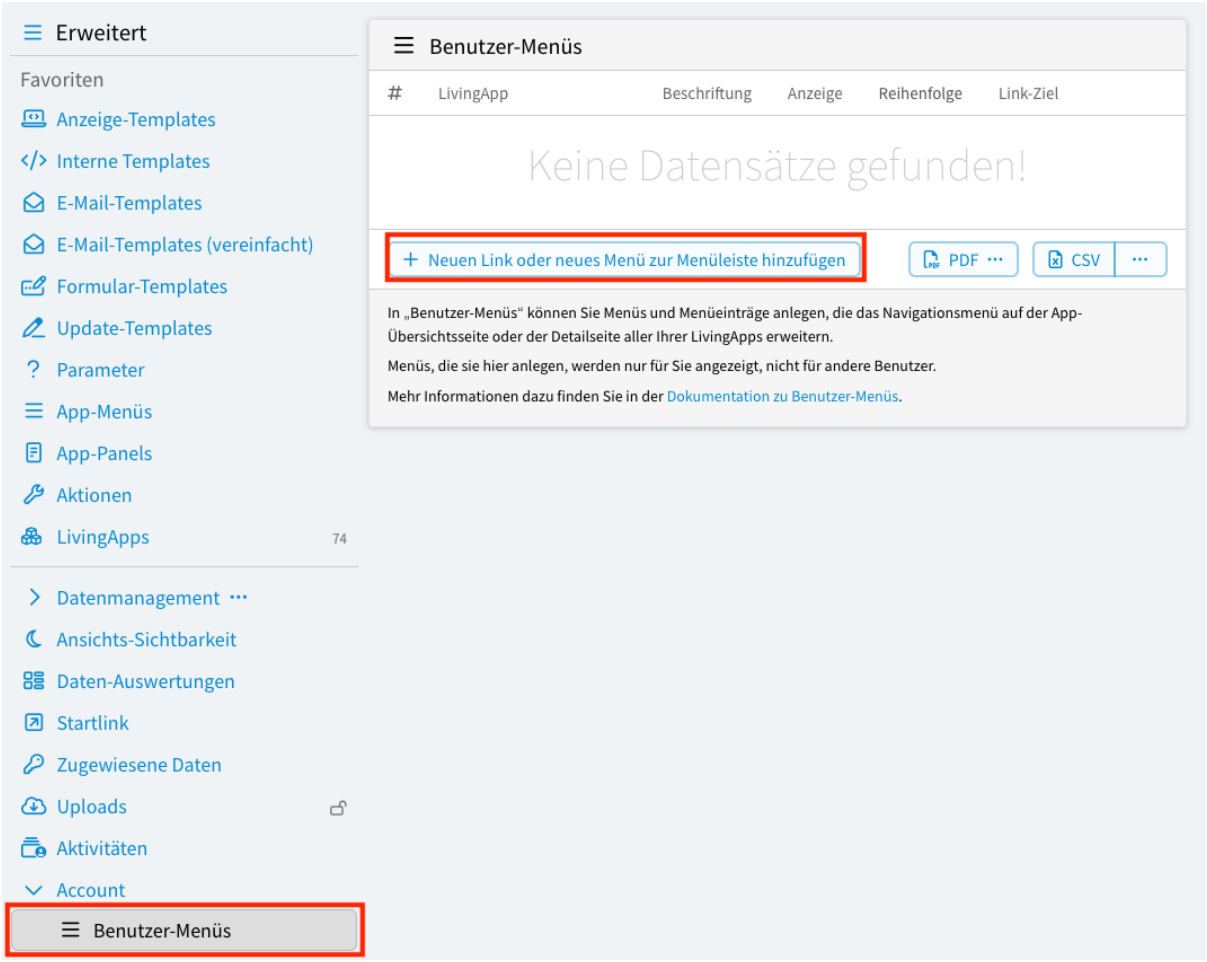

Abb. 2: Benutzer-Menü hinzufügen

Benutzer-Menüs

+ Hinzufügen

Speichern | Abbrechen

LivingApp: Auswählen

* Identifizierer

Eine eindeutige Kennung für diesen Menüeintrag. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Beschriftung

Icon

Das Icon für den Link

Anzeigen auf

App-Übersichtsseite?

App-Startseite?

Eingabeformular?

iframe-Seite?

Anzeige-Templates?

eigener Übersichtsseite?

Handy-App?

Position in der Menüleiste

Übergeordnetes Menü: (Nichts ausgewählt)

Reihenfolge

Nach dieser Zahl sortiert werden Menüs in der Menü-Leiste bzw. Menü-Einträge in Untermenüs angezeigt.

Link-Ziel

* Typ: (Nichts ausgewählt)

Link-Attribute

Title

Das HTML-Attribut `title`. Der Text den Sie hier eingeben, wird beim Link als Tooltip angezeigt (d.h. er wird angezeigt, wenn sich die Maus über dem Link befindet).

Target

Das HTML-Attribut `target`. Wenn Sie hier etwas eingeben wird die Zielseite in einen anderen Fenster angezeigt.

Class

Das HTML-Attribut `class`.

Zeitsteuerung

Aktivierung: TT.MM.JJJJ, --:--

Der Link wird erst nach diesem Zeitpunkt aktiviert. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert.

Deaktivierung: TT.MM.JJJJ, --:--

Nach diesem Zeitpunkt ist der Link nicht mehr aktiv. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Notizen > Notizen

Speichern | Abbrechen

Abb. 3: Erstellung eines Benutzer-Menüeintrages

App-Startseite?

Eingabeformular?

iframe-Seite?

Anzeige-Templates

eigener Übersichtsseite? (nur für App-Panels und Benutzer-Panels)

Handy-App?

erscheinen.

Position im Panel

Hier kann, wenn ein Untermenü angelegt werden soll, mit *Übergeordnetes Menü* ein Menü ausgewählt werden. (Wenn Sie in der Menü-Liste den Button *Neues Menü* bei einem vorhandenen Menü anklicken, so wird dieses Menü bei *Übergeordnetes Menü* vorausgewählt).

Mit *Reihenfolge* kann die Reihenfolge bestimmt werden, in der Menüs der gleichen Ebene sortiert werden. Je grösser die Zahl, desto weiter hinten erscheint der Menüpunkt.

Position im Gitter auf der Übersichts-Seite

Erscheint nur, wenn bei *eigener Übersichtsseite?* das Häkchen gesetzt wurde.

Hier können Sie die *Zeile* und *Spalte*, wo das Panel auf der Übersichts-Seite angezeigt werden soll und die *Breite* und *Höhe* des Panels festlegen.

Link-Ziel

Wählen Sie hier das Ziel aus, auf das der Link zeigen soll.

Bei den Zielen

App-Startseite

Eingabeformular (öffentlich)

Eingabeformular (einebettet)

Daten-Management

Auswertung

Import/Export

Arbeitsaufgaben

Konfiguration: Eingabe

Konfiguration: Arbeitsaufgaben

Konfiguration: Daten

Konfiguration: Berechtigungen

Konfiguration: Erweitert

Anzeige-Template

Daten-Auswertung

können Sie die LivingApp auswählen, deren Unterseite Sie verlinken wollen. Wählen Sie keine LivingApp aus, wird zur entsprechenden Unterseite der aktuellen LivingApp verlinkt.

Eigener Link

Hier kann eine beliebige URL angegeben werden. Diese URL kann absolut (z.B. `https://www.example.de/...`) oder relativ (z.B. `/feedback.htm`) sein.

Kein Link

Muss angewählt werden, wenn ein Menüeintrag erstellt werden soll, an dem ein Untermenü hängt.

Link-Attribute

Title
Der Text den Sie hier eingeben, wird angezeigt, wenn sich die Maus über dem Link befindet.

Target
Wenn Sie hier etwas eingeben wird die Zielseite in einem anderen Fenster angezeigt.

Sie können entweder einen selbstgewählten Namen verwenden, dann werden alle Links, die diesen Namen als `target` verwenden, im selben Fenster/Tab mit diesem Namen geöffnet.

Wenn Sie hingegen `_blank` verwenden, wird der Link immer in einem neuen Fenster bzw. Tab geöffnet.

Wenn Sie bei *Link-Ziel/Typ Kein Link* ausgewählt haben wird dieser Wert ignoriert.

Class
Der eingegebene Wert wird als zusätzlicher Inhalt für das `class`-Attribut des Links verwendet, um z.B. das Aussehen des Links zu verändern. Im Normalfall müssen Sie hier nichts eingeben.

Ein möglicher Anwendungsfall wäre, wenn Sie die Links in einem Anzeige-Template verwenden wollen, und für bestimmte Links das Aussehen anpassen wollen.

Zeitsteuerung
Hier kann festgelegt werden in welchem Zeitraum das Menü angezeigt werden soll. Mit *Aktivierung* legt man den Zeitpunkt fest, ab dem der Link angezeigt werden soll. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert. Mit *Deaktivierung* kann angegeben werden ab wann der Link nicht mehr angezeigt werden soll. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

21.1.2 Anwendungsbeispiel

Im Anwendungsbeispiel wird ein Menüeintrag erstellt, der zum aktuellen Wetterbericht verlinkt.

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und klicken anschließend in der linken Menüleiste auf *Account* → *Benutzer-Menüs* und dann auf *Neuen Link oder neues Menü zur Menüleiste hinzufügen* um einen neuen Menüeintrag zu erstellen.

Im nun geöffneten Fenster werden folgende Konfigurationen vorgenommen.

Der Menüeintrag soll auf der *App-Übersichtsseite*, auf der *App-Startseite* und beim *Eingabeformular* angezeigt werden.

Beim *Typ* des *Link-Zieles* muss *Eigener Link* gewählt werden. Bei *Eigener Link* wird die URL eingetragen. Klicken Sie dann auf *Speichern* und gehen danach zu einer der Seiten, die Sie bei *Anzeigen auf* ausgewählt haben.

Der Link steht dann im Menüeintrag unter „Wetter“ zur Verfügung.

21.2 Benutzer-Panels

Die erweiterte Funktion *Benutzer-Panels* ermöglicht es Ihnen auf verschiedenen Seiten Kacheln mit z.B. Hinweistexten oder Links zu platzieren. Im Gegensatz zu *App-Panels* werden *Benutzer-Panels* in allen Ihren Apps angezeigt (es sei denn, Sie ordnen das Panel einer bestimmten App zu). Außerdem können diese Panels auch auf der *App-Übersichtsseite?* platziert werden.

21.2.1 Erstellung von Benutzer-Panels

Wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Account* → *Benutzer-Panels* und dann auf *Neues Panel hinzufügen*.

Im nun geöffneten Fenster können folgende Konfigurationen vorgenommen werden.

LivingApp
Die App in deren Kontext dieses Panel angezeigt werden soll. Wenn Sie hier keine App auswählen, wird dieses Panel bei allen Ihrer Apps dargestellt.

Identifizierer
Der eindeutige Name des Links. Der Name darf nur Buchstaben, Ziffern und „_“ beinhalten.

Benutzer-Menüs

+ Hinzufügen

Speichern | Abbrechen

LivingApp: Auswählen

* Identifizierer: wetter

Eine eindeutige Kennung für diesen Menüeintrag. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Beschriftung: Wetter

Icon: poo-storm

Das Icon für den Link

Anzeigen auf

- App-Übersichtsseite? ☑
- App-Startseite? ☑
- Eingabeformular? ☑
- iframe-Seite? ☐
- Anzeige-Templates? ☐
- eigener Übersichtsseite? ☐
- Handy-App? ☐

Position in der Menüleiste

Übergeordnetes Menü: (Nichts ausgewählt)

Reihenfolge:

Nach dieser Zahl sortiert werden Menüs in der Menü-Leiste bzw. Menü-Einträge in Untermenüs angezeigt.

Link-Ziel

* Typ: Eigener Link

Eigener Link: https://www.wetter.de

Diese URL kann absolut (z.B. `https://www.example.de/...`) oder relativ (z.B. `/feedback.htm`) sein.

Link-Attribute

Title:

Das HTML-Attribut `title`. Der Text den Sie hier eingeben, wird beim Link als Tooltip angezeigt (d.h. er wird angezeigt, wenn sich die Maus über dem Link befindet).

Target:

Das HTML-Attribut `target`. Wenn Sie hier etwas eingeben wird die Zielseite in einen anderen Fenster angezeigt.

Class:

Das HTML-Attribut `class`.

Zeitsteuerung

Aktivierung: TT . MM . JJJJ , -- : --

Der Link wird erst nach diesem Zeitpunkt aktiviert. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert.

Deaktivierung: TT . MM . JJJJ , -- : --

Nach diesem Zeitpunkt ist der Link nicht mehr aktiv. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Notizen > Notizen

Speichern | Abbrechen

Abb. 4: Benutzer-Menü - Eigener Link erstellen

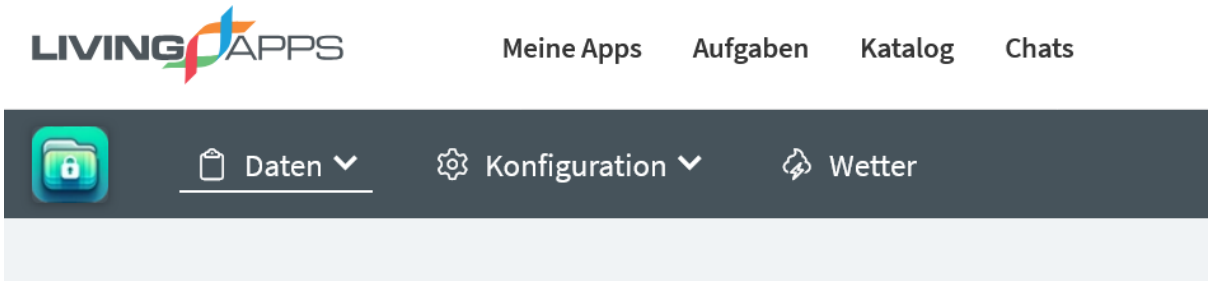

Abb. 5: Benutzer-Menü - Eigener Link Ergebnis

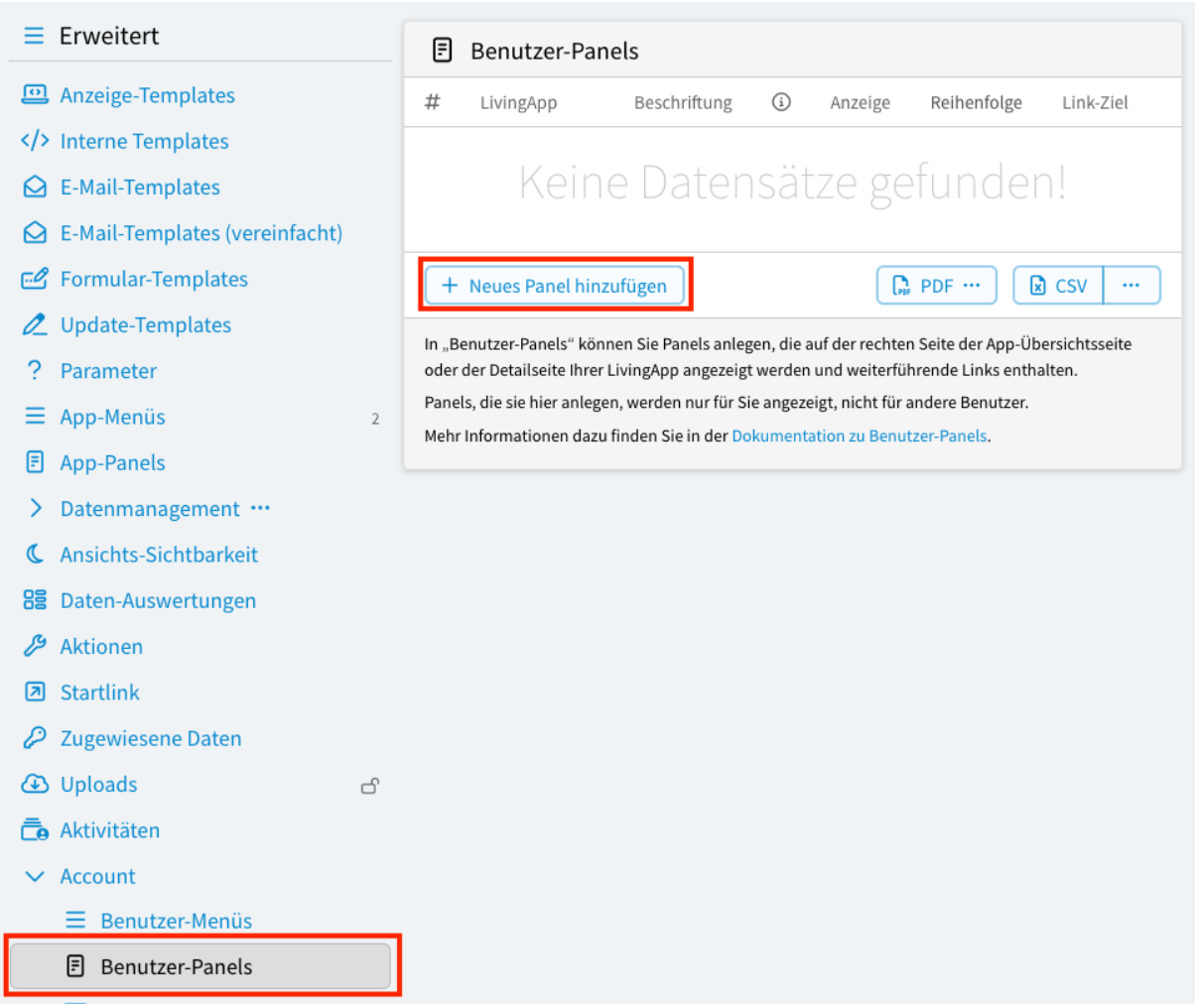

Abb. 6: Benutzer-Panel hinzufügen

Benutzer-Panels

+ Hinzufügen

✓ Speichern ↺ Abbrechen

LivingApp: ⇄ Auswählen

* Identifizierer:
Eine eindeutige Kennung für dieses Panel. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Beschriftung:

Icon:
Das Icon für den Link

Darstellung & Hintergrund

* Darstellung: ◉ Normal ○ Karte
„**Karte**" benötigt mehr vertikalen Platz als „**Normal**", zeigt aber den Hintergrund größer an.

Hintergrund: ◉ (Nichts ausgewählt) ○ Einzelne Farbe ○ Linearer Farbverlauf ○ Kreisförmiger Farbverlauf ○ Bild

Anzeigen auf

App-Übersichtsseite? ☐
App-Startseite? ☐
Eingabeformular? ☐
iframe-Seite? ☐
Anzeige-Templates? ☐
eigener Übersichtsseite? ☐

Position im Panel

Übergeordnetes Panel: (Nichts ausgewählt)

Reihenfolge:
Nach dieser Zahl sortiert werden die Panels angezeigt.

Link-Ziel

* Typ: (Nichts ausgewählt)

Link-Attribute

Title:
Das HTML-Attribut `title`. Der Text den Sie hier eingeben, wird beim Link als Tooltip angezeigt (d.h. er wird angezeigt, wenn sich die Maus über dem Link befindet).

Target:
Das HTML-Attribut `target`. Wenn Sie hier etwas eingeben wird die Zielseite in einen anderen Fenster angezeigt.

Class:
Das HTML-Attribut `class`.

Zeitsteuerung

Aktivierung: TT . MM . JJJJ , -- : --
Der Link wird erst nach diesem Zeitpunkt aktiviert. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert.

Deaktivierung: TT . MM . JJJJ , -- : --
Nach diesem Zeitpunkt ist der Link nicht mehr aktiv. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Beschreibung

Beschreibung: Datei Bearbeiten Ansicht Einfügen Format Werkzeuge Tabelle
Absatz **B** *I*
p 0 Wörter tiny

Anzeige-Template: ⇄ Auswählen
Wenn Sie hier ein Anzeige-Template auswählen wird die Beschreibung von diesem Anzeige-Template geholt.

Notizen > Notizen

✓ Speichern ↺ Abbrechen

Abb. 7: Benutzer-Panel erstellen

Beschriftung
Der Link-Text im Kopfbereich des Panels.

Icon
Das Icon für den Link. Verwendet werden können alle Namen aus dem FontAwesome-Icon-Set.

Darstellung
Normal bzw. *Karte* kann hier ausgewählt werden. *Karte* führt zu einer etwas höheren Darstellung mit mehr Hintergrund.

Hintergrund
Hier besteht die Möglichkeit zwischen *Einzelner Farbe*, *Linearer Farbverlauf*, *Kreisförmiger Farbverlauf* und *Bild* zu wählen. Je nach Auswahl werden anschliessend Felder für *Textfarbe*, *Hintergrundfarbe oben*, *Hintergrundfarbe unten* oder ein *Bild* angeboten.

Anzeigen auf
Gibt an, wo das Panel erscheint. Der Link kann sowohl auf der

App-Übersichtsseite?

App-Startseite?

Eingabeformular?

iframe-Seite?

Anzeige-Templates

eigener Übersichtsseite?

erscheinen.

Positionierung im Panel
Hier kann, wenn Panels ineinander verschachtelt werden sollen, mit *Übergeordnetes Panel* ein Panel ausgewählt werden. (Wenn Sie in der Panel-Liste den Button *Neues Panel* bei einem vorhandenen Panel anklicken, so wird dieses Panel bei *Übergeordnetes Panel* vorausgewählt)

Mit *Reihenfolge* kann die Reihenfolge bestimmt werden, in der Panels der gleichen Ebene sortiert werden. Je grösser die Zahl, desto weiter hinten erscheint die Kachel.

Link-Ziel
Hier kann aus verschiedenen Zielen ausgewählt werden, auf die ein Link zeigen kann. Bei den Zielen

App-Startseite

Eingabeformular (öffentlich)

Eingabeformular (einebettet)

Daten-Management

Auswertung

Import/Export

Arbeitsaufgaben

Konfiguration: Eingabe

Konfiguration: Arbeitsaufgaben

Konfiguration: Daten

Konfiguration: Berechtigungen

Konfiguration: Erweitert

Anzeige-Template

Daten-Auswertung

können Sie die LivingApp auswählen, deren Unterseite Sie verlinken wollen. Wählen Sie keine LivingApp aus, wird zur entsprechenden Unterseite der aktuellen LivingApp verlinkt.

Eigener Link
Hier kann eine beliebige URL angegeben werden. Diese URL kann absolut (z.B. `https://www.example.de/...`) oder relativ (z.B. `/feedback.htm`) sein.

Kein Link
Muss angewählt werden, wenn ein Panel erstellt werden soll, in dem sich andere Panels befinden.

Link-Attribute

Title
Der Text den Sie hier eingeben, wird angezeigt, wenn sich die Maus über dem Link befindet.

Target
Wenn Sie hier etwas eingeben wird die Zielseite in einem anderen Fenster angezeigt.

Sie können entweder einen selbstgewählten Namen verwenden, dann werden alle Links, die diesen Namen als `target` verwenden, im selben Fenster/Tab mit diesem Namen geöffnet.

Wenn Sie hingegen `_blank` verwenden, wird der Link immer in einem neuen Fenster bzw. Tab geöffnet.

Wenn Sie bei *Link-Ziel/Typ Kein Link* ausgewählt haben wird dieser Wert ignoriert.

Class
Der eingegebene Wert wird als zusätzlicher Inhalt für das `class`-Attribut des Links verwendet, um z.B. das Aussehen des Links zu verändern. Im Normalfall müssen Sie hier nichts eingeben.

Ein möglicher Anwendungsfall wäre, wenn Sie die Links in einem Anzeige-Template verwenden wollen, und für bestimmte Links das Aussehen anpassen wollen.

Zeitsteuerung
Hier kann festgelegt werden in welchem Zeitraum das Panel angezeigt werden soll. Mit *Aktivierung* legt man den Zeitpunkt fest, ab dem der Link angezeigt werden soll. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert. Mit *Deaktivierung* kann angegeben werden ab wann der Link nicht mehr angezeigt werden soll. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Beschreibung
Hier kann ein Beschreibungstext angegeben werden, der in dem Panel angezeigt wird. Alternativ kann auch ein *Anzeige-Template* ausgewählt werden, das den Beschreibungstet ausgibt.

21.2.2 Anwendungsbeispiel

Es soll ein Benutzer-Panel erstellt werden, das häufig benötigte „Schnellaufrufe“ anzeigt. So soll zum Beispiel ein Link zum Anzeige-Template „Veranstaltungen“ und ein weiterer zur Bahnauskunft verlinkt werden.

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und klicken anschließend in der linken Menüleiste auf *Account* → *Benutzer-Panels* und dann auf *Neues Panel hinzufügen* um ein neues Panel zu erstellen.

Im nun geöffneten Fenster werden folgende Konfigurationen vorgenommen.

Im 1. Schritt wird die Überschrift „Schnellaufrufe“ des Panels angelegt.

Im 2. Schritt wird das Anzeigetemplate „Veranstaltungen“ verlinkt.

Klicken Sie dafür in der Übersicht Ihrer Benutzer-Panels auf *Neues Panel*.

Das übergeordnete Panel „Schnellaufrufe“ ist bereits ausgewählt. Es werden folgende weitere Konfigurationen vorgenommen.

Im 3. Schritt wird die URL zur Bahnauskunft verlinkt.

Klicken Sie dafür wieder in der Übersicht Ihrer Benutzer-Panels auf *Neues Panel*.

Das übergeordnete Panel „Schnellaufrufe“ ist wieder bereits ausgewählt. Es werden folgende weitere Konfigurationen vorgenommen.

Das Ergebnis sieht dann so aus und steht überall dort zur Verfügung, wo Sie bei *Anzeigen auf* das Häkchen gesetzt haben.

Benutzer-Panels Schnellaufrufe Identifizierer ueberschrift Info

Bearbeiten

Kopieren — Löschen

LivingApp: Auswählen

* Identifizierer: ueberschrift
Eine eindeutige Kennung für dieses Panel. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Beschriftung: Schnellaufrufe

Icon:
Das Icon für den Link

Darstellung & Hintergrund

* Darstellung: Normal / Karte
„Karte" benötigt mehr vertikalen Platz als „Normal", zeigt aber den Hintergrund größer an.

Hintergrund: (Nichts ausgewählt) / Einzelne Farbe / Linearer Farbverlauf / Kreisförmiger Farbverlauf / Bild

Anzeigen auf

App-Übersichtsseite? ☑
App-Startseite? ☑
Eingabeformular? ☑
iframe-Seite? ☐
Anzeige-Templates? ☐
eigener Übersichtsseite? ☑

Position im Panel

Übergeordnetes Panel: (Nichts ausgewählt)

Reihenfolge:
Nach dieser Zahl sortiert werden die Panels angezeigt.

Positionierung im Gitter auf der Übersichtsseite

Zeile:
Spalte:
Breite:
Höhe:

Link-Ziel

* Typ: Kein Link

Link-Attribute

Title:
Das HTML-Attribut `title`. Der Text den Sie hier eingeben, wird beim Link als Tooltip angezeigt (d.h. er wird angezeigt, wenn sich die Maus über dem Link befindet).

Target:
Das HTML-Attribut `target`. Wenn Sie hier etwas eingeben wird die Zielseite in einen anderen Fenster angezeigt.

Class:
Das HTML-Attribut `class`.

Zeitsteuerung

Aktivierung: TT.MM.JJJJ, --:--
Der Link wird erst nach diesem Zeitpunkt aktiviert. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert.

Deaktivierung: TT.MM.JJJJ, --:--
Nach diesem Zeitpunkt ist der Link nicht mehr aktiv. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Beschreibung

Beschreibung: Datei Bearbeiten Ansicht Einfügen Format Werkzeuge Tabelle
Absatz
p — 0 Wörter — tiny

Anzeige-Template: Auswählen
Wenn Sie hier ein Anzeige-Template auswählen wird die Beschreibung von diesem Anzeige-Template geholt.

Notizen > Notizen

Abb. 8: Benutzer-Panel Überschrift

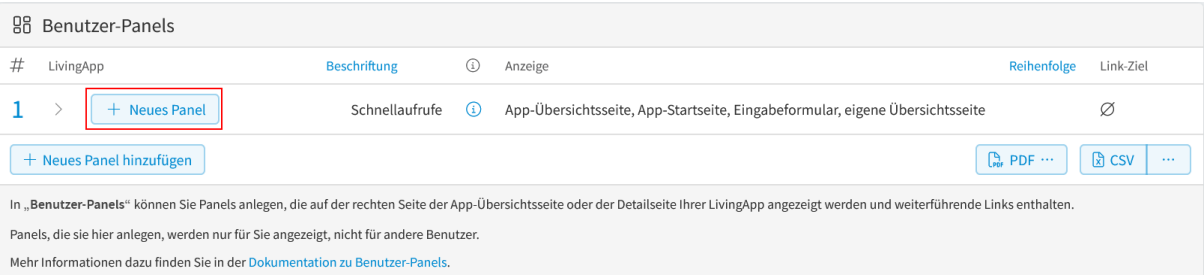

Abb. 9: Benutzer-Panel - 2. Schritt

21.3 Login-Token

Hier können Sie einen alternativen Login-Mechanismus aktivieren.

Dieser kann beispielsweise in externen Entwicklertools zur Authentifizierung ggü. LivingApps verwendet werden, und ist unabhängig von einer etwaigen Änderung des Passworts.

21.4 App-Kategorien

Hier können Apps gegliedert werden. Die Kategorien erscheinen dann auf der App-Übersichtsseite *Meine Apps* auf der linken Seite als Navigation. Eine App kann verschiedenen Kategorien zugewiesen werden.

Das Auswahlmenü der App-Kategorien kann auch mehrstufig sein. Zu beachten ist, dass ein Klick auf einen Menüpunkt alle Apps, auch die der darunterliegenden Menüpunkte, anzeigt. Ein Untermenüpunkt filtert immer feiner. Zum Beispiel können hier die Apps in „Privat“ und „Beruflich“ aufgeteilt werden.

Nachdem Sie im Menü *Konfiguration* → *Erweitert* → *Account* → *App-Kategorien* ausgewählt und auf *Hinzufügen* geklickt haben erscheint folgende *Eingabemaske*.

Identifier
: Identifizierer der Kategorie.

Übergeordnete Kategorie
: Um eine Kategorie einer anderen als Unterkategorie zuzuordnen, wählen Sie hier die übergeordnete aus.

Reihenfolge
: Legt die Anzeige-Reihenfolge fest.

#Apps
: Sobald der Kategorie Apps zugeordnet sind, wird hier deren Anzahl angezeigt.

Nachdem Sie auf *Speichern* geklickt haben, erscheinen neben *Bearbeiten* die *Tabs Übersetzungen* und *Zugeordnete Apps*.

Klicken Sie zunächst auf *Übersetzungen*. Hier muss der *Name* der Kategorie in der entsprechenden Sprache (Deutsch oder Englisch) angegeben werden. Dieser Name steht dann auch in der Navigationsleiste in der App-Übersicht.

Um *Unterkategorien* anzulegen klicken Sie in der Kategorienübersicht auf das + bei der Kategorie, zu der Sie Unterkategorien anlegen wollen.

Es erscheint folgende *Eingabemaske*

Identifier
: Identifizierer der Unterkategorie

Übergeordnete Kategorie
: Die übergeordnete Kategorie ist dann bereits ausgewählt.

Reihenfolge
: Legt die Anzeige-Reihenfolge fest.

Benutzer-Panels Teilnehmerliste Reihenfolge 10

+ Hinzufügen

Speichern Abbrechen

LivingApp: Auswählen

* Identifizierer: teilnehmerliste
Eine eindeutige Kennung für dieses Panel. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Beschriftung: Teilnehmerliste

Icon
Das Icon für den Link

Darstellung & Hintergrund

* Darstellung: Normal / Karte
„Karte" benötigt mehr vertikalen Platz als „Normal", zeigt aber den Hintergrund größer an.

Hintergrund: (Nichts ausgewählt) / Einzelne Farbe / Linearer Farbverlauf / Kreisförmiger Farbverlauf / Bild

Anzeigen auf

App-Übersichtsseite? ☑
App-Startseite? ☑
Eingabeformular? ☑
iframe-Seite? ☐
Anzeige-Templates? ☐
eigener Übersichtsseite? ☑

Position im Panel

Übergeordnetes Panel: Schnellaufrufe

Reihenfolge: 10
Nach dieser Zahl sortiert werden die Panels angezeigt.

Positionierung im Gitter auf der Übersichtsseite

Zeile
Spalte
Breite
Höhe

Link-Ziel

* Typ: Anzeige-Template

Anzeige-Template: Anmeldung Computerkurse → Teilnehmerliste / teilnehmerliste Entfernen Auswählen
Das Anzeige-Auswertung, das das Ziel der Link ist.

Request-Parameter
Zusätzliche Parameter für den HTTP-Request auf das Anzeige-Template.

Link-Attribute

Title
Das HTML-Attribut title. Der Text den Sie hier eingeben, wird beim Link als Tooltip angezeigt (d.h. er wird angezeigt, wenn sich die Maus über dem Link befindet).

Target
Das HTML-Attribut target. Wenn Sie hier etwas eingeben wird die Zielseite in einen anderen Fenster angezeigt.

Class
Das HTML-Attribut class.

Zeitsteuerung

Aktivierung: TT . MM . JJJJ , -- : --
Der Link wird erst nach diesem Zeitpunkt aktiviert. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert.

Deaktivierung: TT . MM . JJJJ , -- : --
Nach diesem Zeitpunkt ist der Link nicht mehr aktiv. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Beschreibung

Beschreibung
Datei Bearbeiten Ansicht Einfügen Format Werkzeuge Tabelle
Absatz B I
p 0 Wörter tiny

Anzeige-Template: Auswählen
Wenn Sie hier ein Anzeige-Template auswählen wird die Beschreibung von diesem Anzeige-Template geholt.

Notizen > Notizen

Speichern Abbrechen

Abb. 10: Benutzer-Panel - Anzeige-Template auswählen

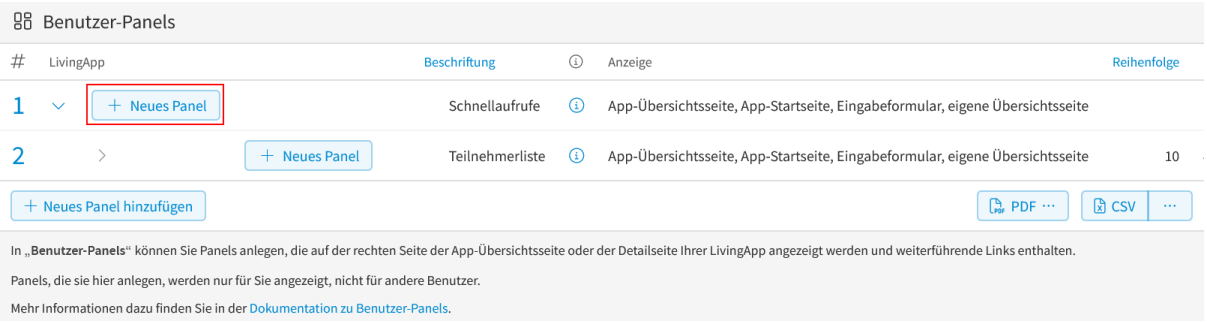

Abb. 11: Benutzer-Panel - 3. Schritt

#Apps
: Sobald der Kategorie Apps zugeordnet sind, wird hier deren Anzahl angezeigt.

Nach dem *Speichern* der Unterkategorie muss auch hier bei *Übersetzungen* der Name der Kategorie in der entsprechenden Sprache eingetragen werden.

Nachdem die Kategorien angelegt wurden, können die entsprechenden Apps zugewiesen werden. Das können Sie entweder über *Account* → *App-Zuordnungen* tun, oder über den Tab *Zugeordnete Apps*. Für letzteres klicken Sie in der Kategorien-Übersicht auf die Kategorie, der Sie Apps zuordnen wollen und anschließend auf den Tab *Zugeordnete Apps*. Durch *Hinzufügen* können Sie der Kategorie beliebig viele *Apps zuordnen*.

App
: Wählen Sie die App aus, die Sie dieser Kategorie hinzufügen wollen.

Reihenfolge
: Nach dieser Zahl werden die Apps in der App-Übersichtsseite sortiert. (Wird noch nicht unterstützt)

Das *Ergebnis* der Kategorisierung sehen Sie auf Ihrer App-Übersichtsseite unter *Meine Apps*.

21.5 App-Zuordnungen

Sofern App-Kategorien erstellt sind, kann auch hier die Zuordnung der Apps vorgenommen werden. Jede Ihrer Apps kann einer Kategorie zugewiesen werden, sowie die Reihenfolge, nach denen die Apps sortiert werden, festgelegt werden.

Nachdem Sie im Menü *Konfiguration* → *Erweitert* und dann *Account* → *App-Zuordnungen* ausgewählt und auf *Hinzufügen* geklickt haben, erscheint folgende *Eingabemaske*.

App
: Wählen Sie hier die App aus, die einer Kategorie zugeordnet werden soll.

App-Kategorie
: Wählen Sie hier die Kategorie (evtl. auch Unterkategorie) aus, der die oben gewählte App zugeordnet werden soll.

Reihenfolge
: Nach dieser Zahl werden die Apps in der App-Übersichtsseite sortiert. (Wird noch nicht unterstützt)

In unserem *Beispiel* wurden 4 Apps den Kategorien *Privat-Familie* und *Privat-Verein* zugeordnet.

21.6 Kalender

Hier kann konfiguriert werden, wie Termine der Arbeitsaufgaben in den Apps publiziert werden. Dies ermöglicht es Ihnen, diese Termine mit Ihrem Kalender-Programm zu synchronisieren (z. B. „Outlook Express“ (auf Windows) oder „Kalender“ (auf Mac OS X)).

Nachdem Sie im Menü *Konfiguration* → *Erweitert* und dann *Account* → *Kalender* ausgewählt und auf *Hinzufügen* geklickt haben erscheinen folgende Eingabepunkte.

Benutzer-Panels | Bahnauskunft | Identifizierer: bahn | Info | Reihenfolge: 20 | 3/3

Bearbeiten

Kopieren | Löschen

LivingApp: Auswählen

* Identifizierer: bahn
Eine eindeutige Kennung für dieses Panel. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Beschriftung: Bahnauskunft

Icon:
Das Icon für den Link

Darstellung & Hintergrund

* Darstellung: (•) Normal ( ) Karte
„Karte" benötigt mehr vertikalen Platz als „Normal", zeigt aber den Hintergrund größer an.

Hintergrund: (•) (Nichts ausgewählt) ( ) Einzelne Farbe ( ) Linearer Farbverlauf ( ) Kreisförmiger Farbverlauf ( ) Bild

Anzeigen auf

- [x] App-Übersichtsseite?
- [x] App-Startseite?
- [x] Eingabeformular?
- [ ] iframe-Seite?
- [ ] Anzeige-Templates?
- [x] eigener Übersichtsseite?

Position im Panel

Übergeordnetes Panel: Schnellaufrufe

Reihenfolge: 20
Nach dieser Zahl sortiert werden die Panels angezeigt.

Positionierung im Gitter auf der Übersichtsseite

Zeile:

Spalte:

Breite:

Höhe:

Link-Ziel

* Typ: Eigener Link

Eigener Link: https:/www.bahn.de
Diese URL kann absolut (z.B. `https://www.example.de/...`) oder relativ (z.B. `/feedback.htm`) sein.

Link-Attribute

Title:
Das HTML-Attribut `title`. Der Text den Sie hier eingeben, wird beim Link als Tooltip angezeigt (d.h. er wird angezeigt, wenn sich die Maus über dem Link befindet).

Target:
Das HTML-Attribut `target`. Wenn Sie hier etwas eingeben wird die Zielseite in einen anderen Fenster angezeigt.

Class:
Das HTML-Attribut `class`.

Zeitsteuerung

Aktivierung: TT . MM . JJJJ , -- : --
Der Link wird erst nach diesem Zeitpunkt aktiviert. Wenn hier nichts eingegeben wird, wird der Link sofort aktiviert.

Deaktivierung: TT . MM . JJJJ , -- : --
Nach diesem Zeitpunkt ist der Link nicht mehr aktiv. Wenn hier nichts eingegeben wird, wird der Link nicht automatisch deaktiviert.

Beschreibung

Beschreibung: Datei Bearbeiten Ansicht Einfügen Format Werkzeuge Tabelle
Absatz | B I
p | 0 Wörter | tiny

Anzeige-Template: Auswählen
Wenn Sie hier ein Anzeige-Template auswählen wird die Beschreibung von diesem Anzeige-Template geholt.

Abb. 12: Benutzer-Panel - Eigener Link

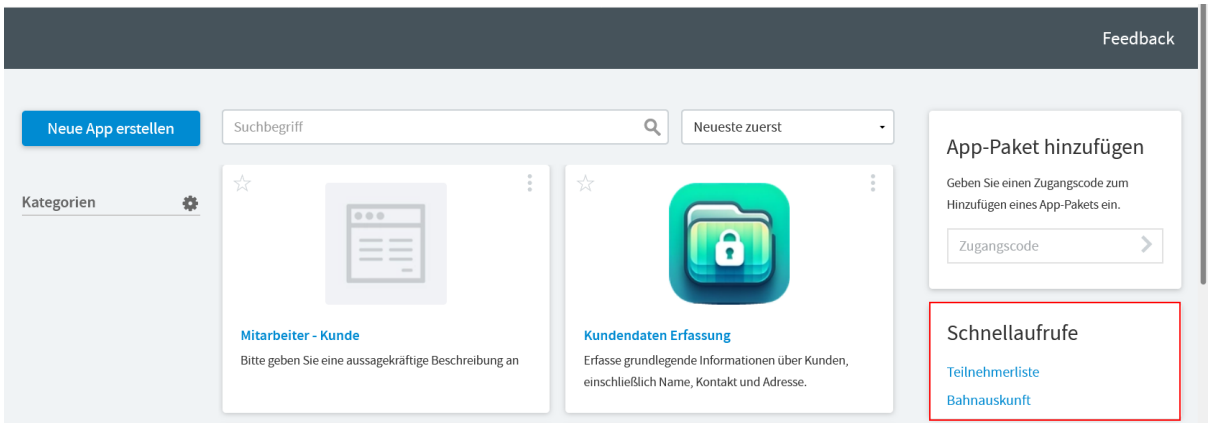

Abb. 13: Benutzer-Panel - Ergebnis

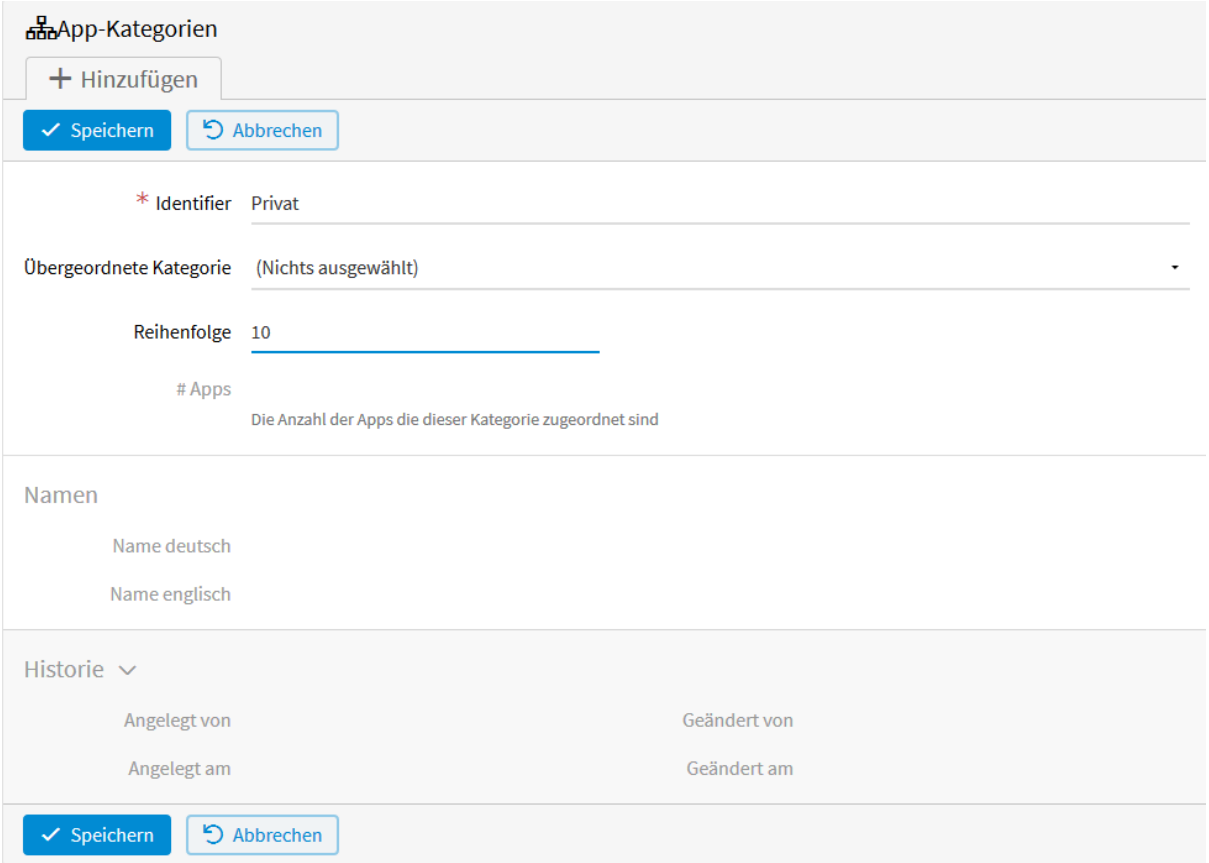

Abb. 14: App-Kategorie anlegen

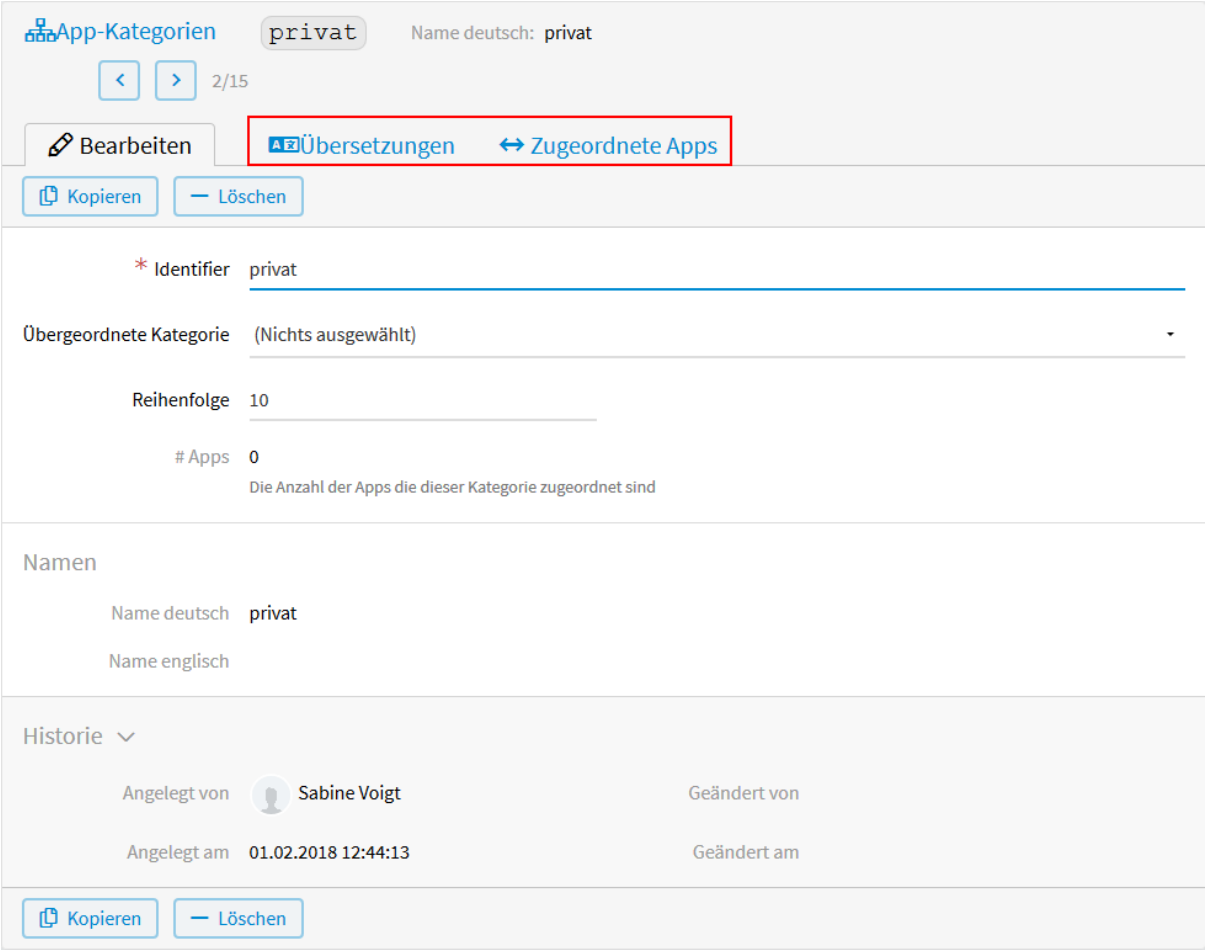

Abb. 15: App-Kategorie Tabs

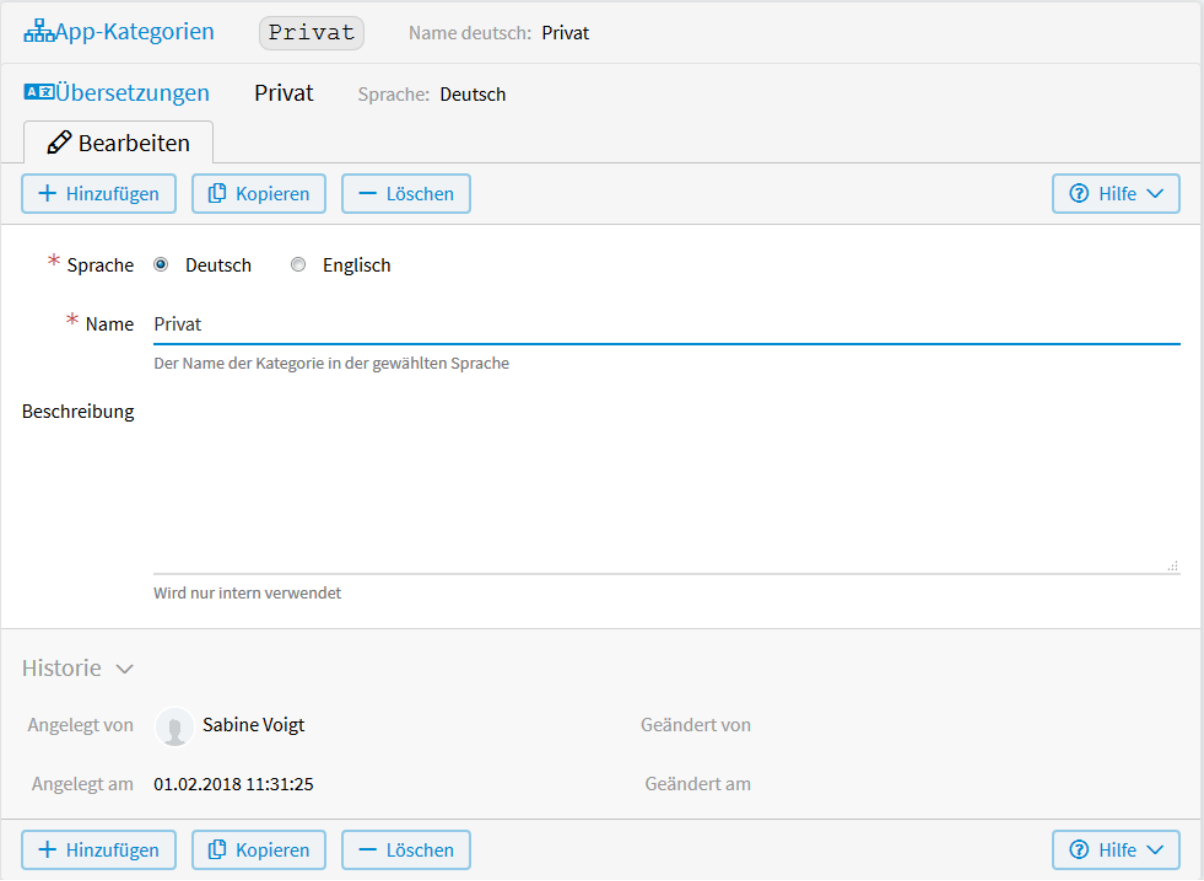

Abb. 16: App-Kategorie Name

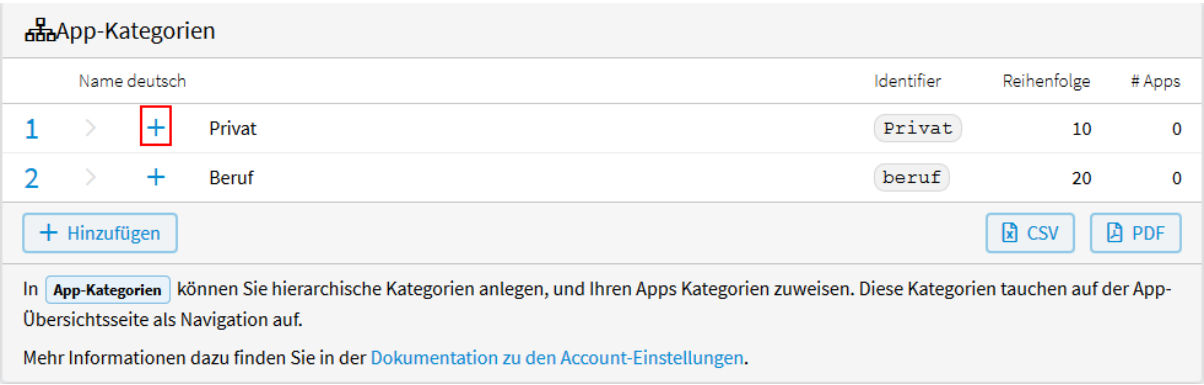

Abb. 17: Unterkategorie hinzufügen

App-Kategorien

1/2

+ Hinzufügen

Speichern Abbrechen

* Identifier familie

Übergeordnete Kategorie Privat

Reihenfolge 10

Apps

Die Anzahl der Apps die dieser Kategorie zugeordnet sind

Namen

Name deutsch

Name englisch

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen

Abb. 18: Unterkategorie anlegen

App-Kategorien familie Name deutsch: Familie

↔ Zugeordnete Apps

+ Hinzufügen

Speichern Abbrechen Hilfe

App Haushaltsbuch

Reihenfolge 10

Nach dieser Zahl werden die Apps in der App-Übersichtsseite sortiert.

Historie

Angelegt von Geändert von

Angelegt am Geändert am

Speichern Abbrechen Hilfe

Abb. 19: App-Zuordnung

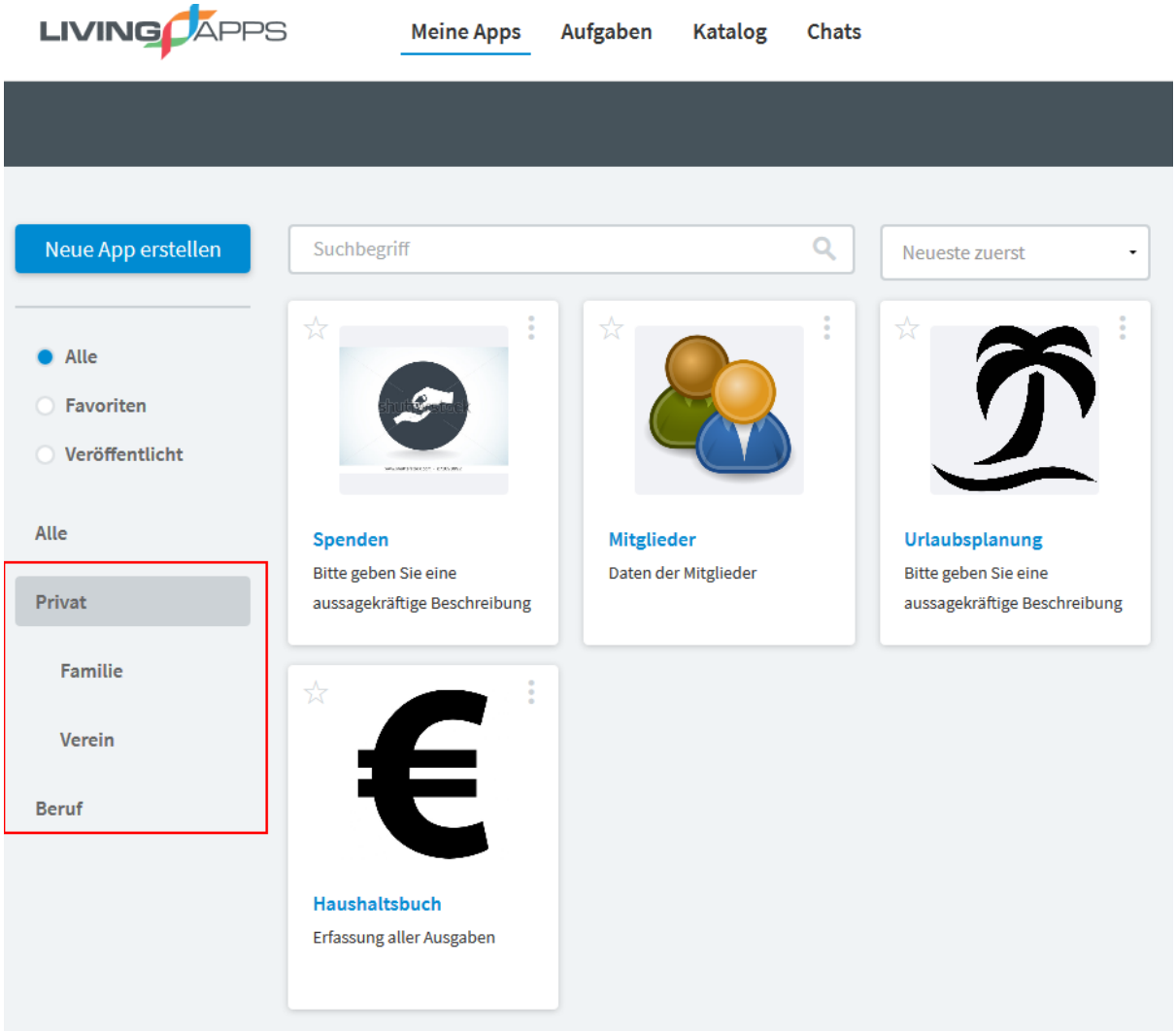

Abb. 20: Ergebnis der Kategorisierung

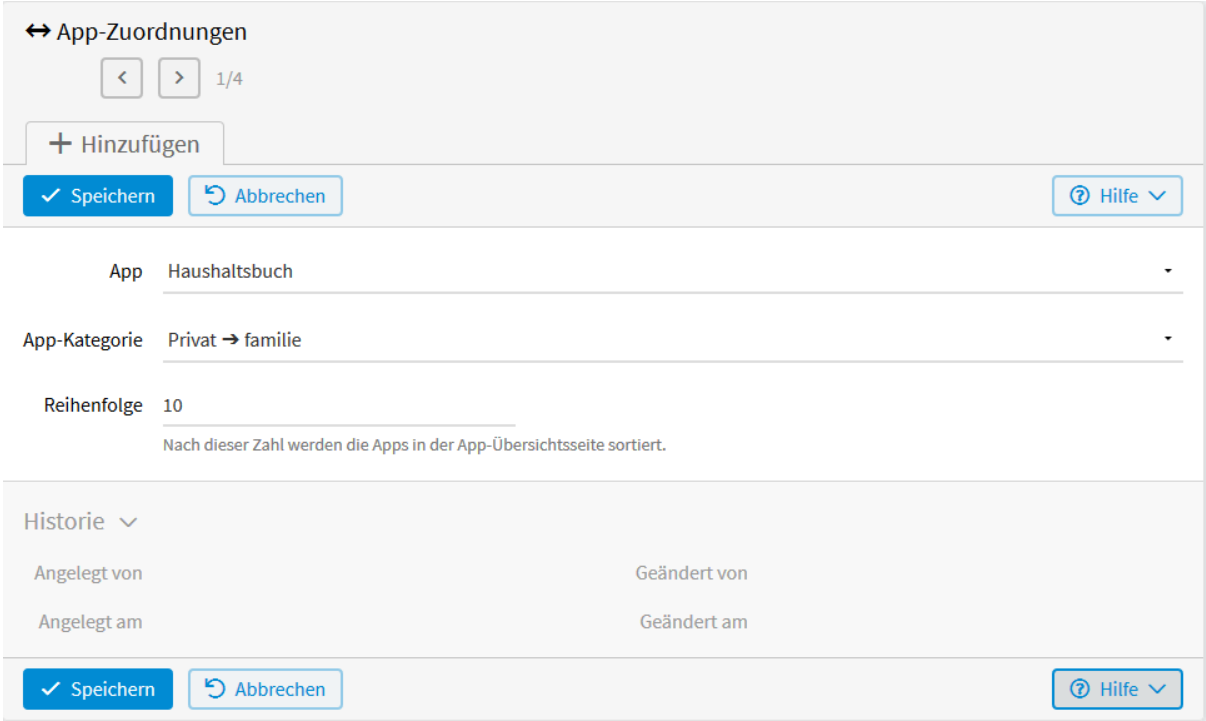

Abb. 21: App-Zuordnungen anlegen

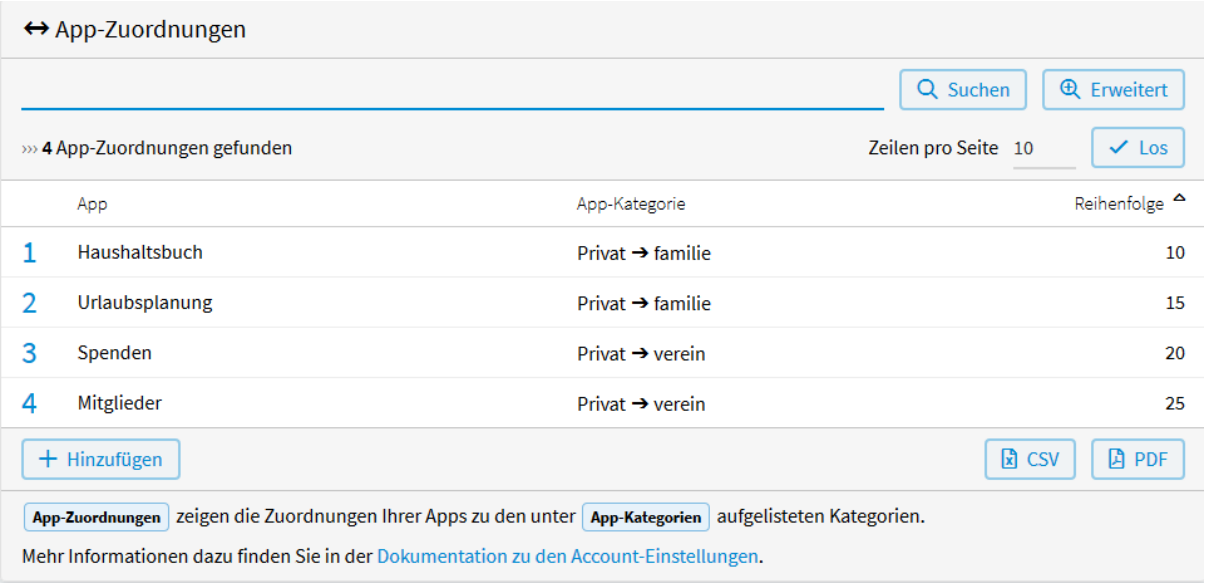

Abb. 22: App-Zuordnungen Ergebnis

Name
Der Name, unter dem dieser Kalender beim Abonnieren in Ihrem Kalender-Programm erscheint.

Dateiname
Unter diesem Namen wird der Kalender abgespeichert, wenn Sie ihn downloaden statt abonnieren.

App
Wählen Sie hier die App aus, deren Termine aus den Arbeitsaufgaben Sie im Kalender sehen wollen. Wird hier keine App gewählt, enthält der Kalender alle Ihre Termine aus den Arbeitsaufgaben. Möchten Sie Termine mehrerer Apps sehen, müssen Sie für jede App einen neuen Kalender hinzufügen.

21.7 SMTP-Server

Hier kann ein eigener E-Mail-Server für E-Mails, die über LivingApps versendet werden (z. B. bei Versendungen und E-Mail-Templates), eingerichtet werden.

Nachdem Sie im Menü *Konfiguration* → *Erweitert* und dann *Account/SMTP-Server* ausgewählt und auf *Hinzufügen* geklickt haben, erscheint folgende *Eingabemaske*.

SMTP-Host
Geben Sie hier den Namen Ihres Mail-Servers an (z.B. `mail.example.com`).

SMTP-Benutzername
Tragen Sie hier Ihren Benutzernamen des Mail-Servers ein.

SMTP-Passwort
Geben Sie hier Ihr Passwort des Mail-Servers ein.

SMTP-E-Mail-Adresse
Geben Sie hier Ihre E-Mail-Adresse an. Diese E-Mail-Adresse wird bei allen über diesen SMTP-Server versendeten E-Mails als Absender verwendet. Eine vollständige E-Mail-Adresse (inklusive Anzeigename) in der Form `Anzeige Name <name@example.com>` wird unterstützt.

Port
Die Port-Nummer unter der der SMTP-Server auf dem *SMTP-Host* angesprochen werden kann. Gängige Port-Nummern sind 25, 587, 465 und 2525.

TLS?
Setzen Sie hier das Häkchen, wenn Sie Ihre Daten sicher und verschlüsselt übertragen wollen.

Authentifizierung
Wählen Sie hier, wie Ihre Authentifizierung erfolgen soll.

Status des letzten Tests
Das Ergebnis des zuletzt durchgeführten Zugang-Tests. Wenn Sie die Konfiguration neu abspeichern wird dieser Status zurückgesetzt. Um den Zugangstest erneut durchzuführen, klicken Sie den Button *SMTP-Zugang testen*.

Datenschutz
Um Ihre E-Mail bei Ihrem SMTP-Server abliefern zu können, muß sich LivingApps bei Ihrem SMTP-Server einloggen. Dazu muß LivingApps Ihre E-Mail-Zugangsdaten speichern. Die E-Mail-Zugangsdaten werden verschlüsselt gespeichert und ausschliesslich zum Zwecke und zum Zeitpunkt des E-Mail-Versands verwendet.

Sie müssen hier bestätigen, daß Sie mit dieser Speicherung einverstanden sind.

21.8 Installationen

Hier werden alle unter diesem Account durchgeführten Installtionen aufgeführt.

SMTP-Server

+ Hinzufügen

Speichern | Abbrechen | Hilfe

* SMTP-Host

* SMTP-Benutzername

* SMTP-Passwort

* SMTP-E-Mail-Adresse

Port 587

TLS?

Authentifizierung (Nichts ausgewählt)

Status

Datenschutz Ich stimme der Speicherung meiner E-Mail-Zugangsdaten zu.

LivingApps erlaubt es, einen Mailserver Ihrer Wahl für den Versand von E-Mails zu verwenden. Die E-Mail-Zugangsdaten werden verschlüsselt gespeichert und ausschliesslich zum Zwecke und zum Zeitpunkt des E-Mail-Versands verwendet.

Diese dürfen nur zum Versand von E-Mails in meinem Auftrag benutzt werden. Diese Zustimmung ist jederzeit widerrufbar. Hierfür müssen nur — durch Klick auf "Zugangsdaten löschen" — die Zugangsdaten aus LivingApps entfernt werden.

Ab dem Zeitpunkt des Widerrufs findet die E-Mail-Versendung über den LivingApps-Mailserver statt.

Historie

Angelegt von | Geändert von

Angelegt am | Geändert am

Speichern | Abbrechen | Hilfe

Abb. 23: SMTP-Server anlegen

KAPITEL 22

Meine LivingApps

Unter *Meine Living-Apps* finden Sie die Übersicht und alle Konfigurationsmöglichkeiten der LivingApps, bei denen Sie Admin-Berechtigung haben. Das macht es einfacher, vorhandene Konfigurationen zu suchen und von einer LivingApp in eine andere zu kopieren.

22.1 Living-Apps

Living-Apps ist eine Liste aller LivingApps, auf die der Benutzer Zugriff hat.

Über Maskenlinks sind alle Masken, die es sonst auf oberster Ebene für die per CMS-Navigation ausgewählte LivingApp gibt, für die in der WAF-Liste ausgewählte LivingApp zugänglich:

Unter *Konfiguration*
: *Basis-Daten*, *Bezeichnungen*, *Startlink*, *Zugewiesene Daten*, *Uploads schützen?*, *Aktivitäten anlegen?*, *Feldreihenfolge beibehalten?*

Unter *Templates*:
: *Anzeige-Templates*, *Interne Templates*, *E-Mail-Templates :guilabel:`Formular-Templates*, *Update-Templates*, *Parameter*, *Datenquellen zu dieser LivingApp*, *Datenquellen dieser LivingApp*.

Unter *Sonstiges*:
: *Daten-Aktionen*, *Formular-Varianten*, *Auswertungen*, *Menüs*, *Panels*, *Meldungen*.

Unter *Datenmanagement*:
: *Farben*, *Felder*, *Sortierung*

22.2 Alle Meldungen

Alle Fehler- oder Log-Meldungen aller LivingApps, die Sie administrieren.

22.3 Alle Anzeige-Templates

Die Anzeigetemplates aller LivingApps, die Sie administrieren.

22.4 Alle Internen Templates

Die Internen Templates aller LivingApps, die Sie administrieren.

22.5 Alle E-Mail-Templates

Die E-Mail-Teplates aller LivingApps, die Sie administrieren.

22.6 Alle E-Mail-Templates (vereinfacht)

Die vereinfachten E-Mail-Templates aller LivingApps, die Sie administrieren.

22.7 Alle Formular-Templates

Die Formular-Templates aller LivingApps, die Sie administrieren.

22.8 Alle Update-Templates

Die Update-Templates aller LivingApps, die Sie administrieren.

22.9 Alle Parameter

Die konfigurierten Parameter aller LivingApps, die Sie administrieren.

22.10 Alle Datenquellen

Alle Datenquellen die einem Anzeige-, E-Mail-, oder Formular-Template einer Ihrer LivingApps zugeordnet sind oder die eine Ihrer LivingApps als Ziel-App haben.

22.11 Alle App-Menüs

Die erstellten Menüs oder Menüeinträge aller LivingApps, die Sie administrieren.

22.12 Alle App-Panels

Alle App-Panels aller LivingApps, die Sie administrieren.

22.13 Farben (Datenmanagement)

Die Farbkonfigurationen aller LivingApps, die Sie administrieren.

22.14 Alle Felder (Datenmanagement)

Die Felderkonfigurationen aller LivingApps, die Sie administrieren.

22.15 Sortierung (Datenmanagement)

Die Sortierkonfigurationen aller LivingApps, die Sie administrieren.

22.16 Alle Formularvarianten

Die Formularvarianten aller LivingApps, die Sie administrieren.

22.17 Alle Daten-Aktionen

Die Aktionen aller LivingApps, die Sie administrieren.

22.18 Alle Daten-Auswertungen

Die Daten-Auswertungen aller LivingApps, die Sie administrieren.

KAPITEL 23

Werkzeuge

23.1 LivingApps kopieren

Mit diesem Werkzeug können Sie mehrere zusammengehörende LivingApps kopieren.

Querverweise zwischen diesen LivingApps (z.B. Applookup-Felder, oder Datenquellen in Anzeige-Templates, etc.) werden dabei beim Kopieren so angepasst, dass sie auf die jeweiligen kopierten LivingApps verweisen.

Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und klicken dann in der linken Menüleiste auf *Werkzeuge* → *LivingApps kopieren*. Es öffnet sich folgendes Fenster:

LivingApps

Zu kopierende LivingApps

Klicken Sie auf *Hinzufügen* um die zu kopierenden LivingApps auszuwählen.

Zur Auswahl stehen alle LivingApps, die Sie besitzen (angelegt haben) oder an denen Sie Administrationsrechte haben.

Setzen Sie das Häkchen bei den LivingApps, die Sie kopieren wollen und klicken Sie auf *Übernehmen*.

Benutzer

Besitzer der kopierten LivingApps

Legen Sie hier fest, welcher Account Besitzer der kopierten LivingApps sein soll und welche Berechtigungen es für diese LivingApps geben soll.

Klicken Sie dafür auf *Auswählen*, setzen das Häkchen bei dem Account, der der Besitzer sein soll und klicken dann auf *Übernehmen*.

Wenn Sie hier keinen Account auswählen, werden die Berechtigungen der zu kopierenden Apps übernommen, Sie werden jedoch zum Besitzer der kopierten LivingApps.

Wenn Sie hier einen Account auswählen, so wird dieser Account Besitzer der kopierten LivingApps, es werden jedoch keine weiteren Berechtigungen von den Original-LivingApps übernommen. Es können jedoch mit den folgenden zwei Optionen zusätzliche Berechtigungen hinzugefügt werden.

Zusätzliche Benutzer mit Administratorrechten

Legen Sie hier fest, welche zusätzlichen Benutzer Administrationsrechte auf den kopierten LivingApps erhalten sollen.

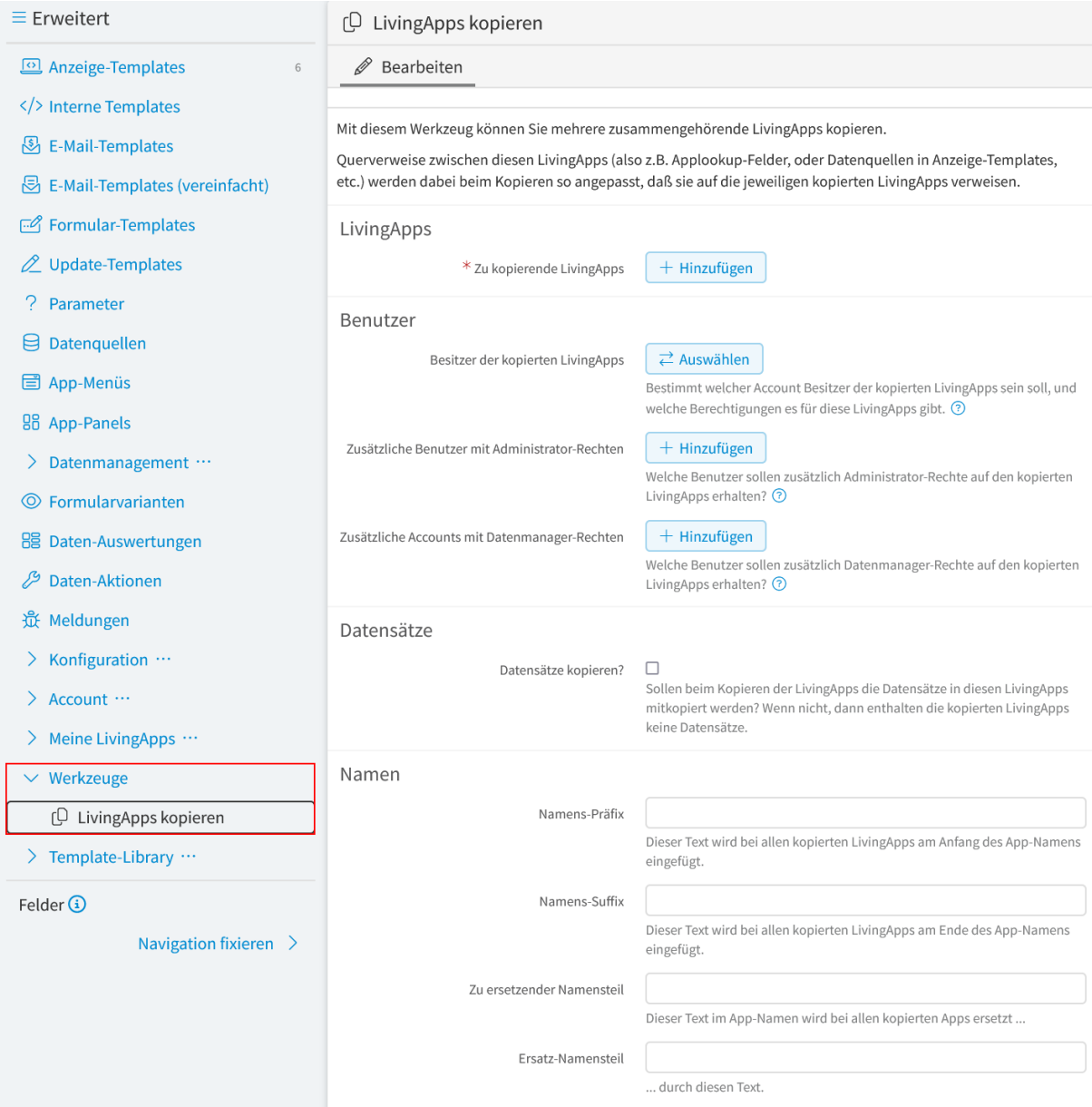

Abb. 1: LivingApps kopieren

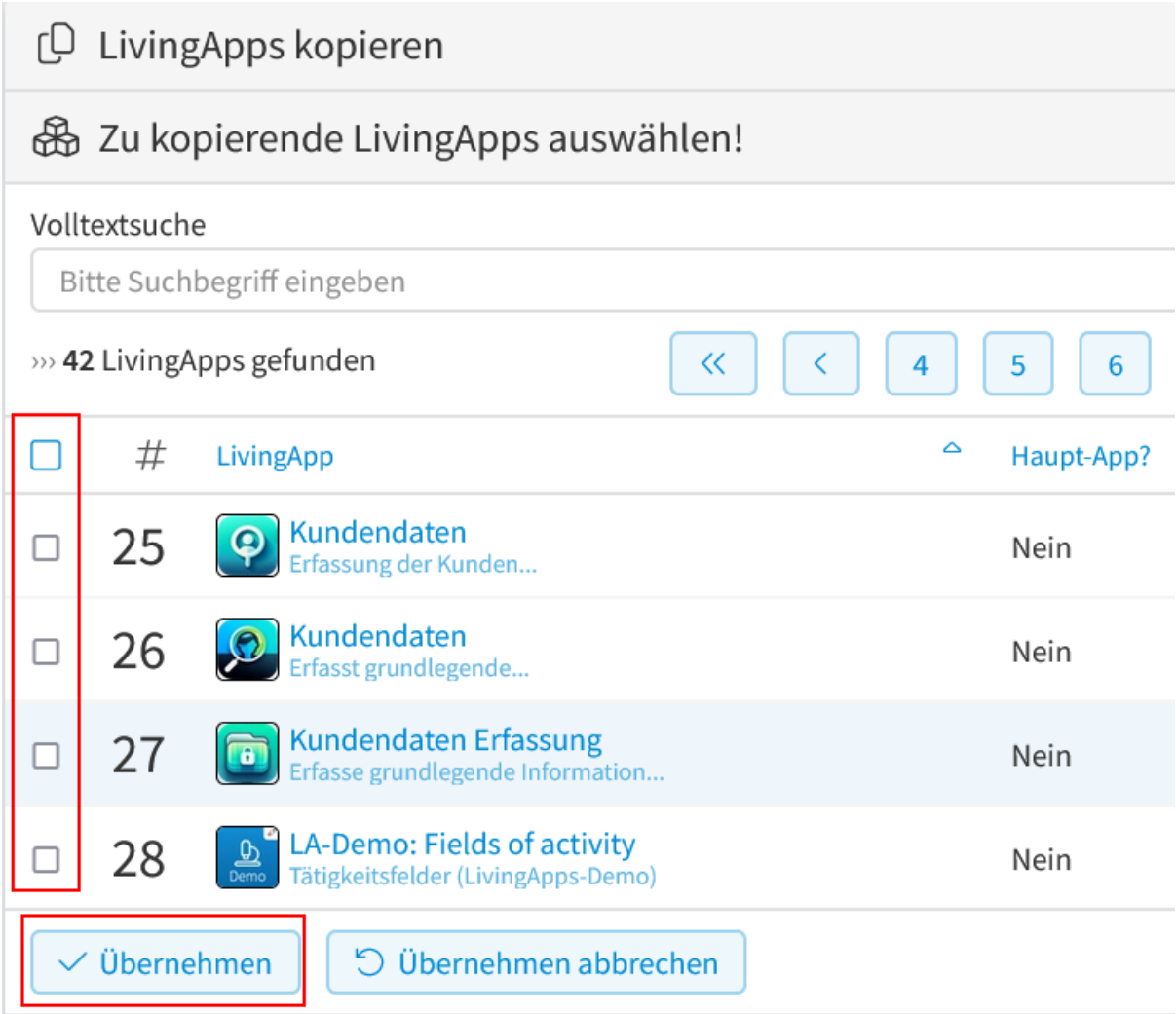

Abb. 2: LivingApps auswählen und übernehmen

Klicken Sie dafür auf *Hinzufügen*, setzen das Häkchen bei den entsprechenden Benutzern und klicken dann auf *Übernehmen*.

Dies ist unabhängig davon, ob Sie bei *Besitzer der kopierten LivingApps* einen neuen Besitzer ausgewählt haben oder nicht.

Wenn jedoch bestehende Berechtigungen einer anderen Berechtigungsstufe mitkopiert werden, werden diese nicht angepasst.

Zusätzliche Accounts mit Datenmanager-Rechten
Legen Sie hier fest, welche zusätzlichen Benutzer Datenverarbeitungsrechte auf den kopierten LivingApps erhalten sollen.

Klicken Sie dafür auf *Hinzufügen*, sezten das Häkchen bei den entsprechenden Benutzern und klicken dann auf *Übernehmen*.

Dies ist unabhängig davon, ob Sie bei *Besitzer der kopierten LivingApps* einen neuen Besitzer der LivingApps ausgewählt haben oder nicht.

Wenn jedoch bestehende Berechtigungen einer anderen Berechtigungsstufe mitkopiert werden, werden diese nicht angepasst.

Datensätze

Datensätze kopieren
Legen Sie hier fest, ob beim Kopieren der LivingApps die Datensätze dieser LivingApps mitkopiert werden sollen. Falls nicht, dann enthalten die kopierten LivingApps keine Datensätze.

Namen
Es wird empfohlen, den Namen der kopierten LivingApps in irgendeiner Art zu ändern, dass die Kopien von den Ausgangs-LivingApps zu unterscheiden sind. Zum Beispiel könnte als *Namens-Suffix* `(Kopie)` verwendet werden.

Namens-Präfix
Dieser Text wird bei allen kopierten LivingApps am Anfang des App-Namens eingefügt.

Namens-Suffix
Dieser Text wird bei allen kopierten LivingApps am Ende des App-Namens eingefügt.

Zu ersetzender Namensteil
Dieser Text im App-Namen wird bei allen kopierten Apps ersetzt …

Ersatz Namensteil
…durch diesen Text.

Durch klicken auf *Kopieren* ist die Konfiguration abgeschlossen und die LivingApps stehen den ausgewählten Benutzern, mit den jeweiligen Berechtigungen, als Kopie zur Verfügung.

KAPITEL 24

Library-Templates

LivingApps bietet Ihnen einige vorgefertigte Templates, die sie in Ihren Anzeige-, Formular- und E-Mail-Templates verwenden können.

24.1 Übersicht über die Templates

Um zur Liste der Library-Templates zu gelangen, wählen Sie im Menü einer beliebigen App *Konfiguration* → *Erweitert* und klicken anschließend in der linken Menüleiste auf *Account* und dann auf *Library-Templates*:

Ein Klick auf eines der Templates in der Liste führt Sie zum Quelltext und zur Dokumentation des Templates:

24.2 Einbinden eines Templates

Sämtliche Library-Templates sind auf die selbe Art und Weise aufrufbar, wie wenn sie interne Templates der jeweiligen App wären. Um z.B. das Library-Template `la_static_ul4` in Ihrem Anzeige-Template auszuführen, benutzen Sie folgenden Code:

```
<?render globals.app.templates.la_static_ul4()?>
```

Genauso wie bei den internen Templates sind auch hier folgende alternative Aufrufe möglich:

```
<?render globals.app.t_la_static_ul4()?>
<?render globals.templates.la_static_ul4()?>
<?render globals.t_la_static_ul4()?>
```

24.3 Überschreiben eines Templates

Wollen Sie die Funktionalität eines Library-Templates ergänzen oder ersetzen, können Sie das tun indem Sie ein Template mit dem gleichen Namen in den internen Templates Ihrer App ablegen.

Sie können dies auch tun, indem Sie ihre eigenen Templates in einer Ihrer Apps gesammelt in deren internen Templates ablegen.

Wenn Sie beispielsweise ein eigenes internes Template `la_static_ul4` anlegen, wird beim Aufruf des Templates mittels folgendem Code:

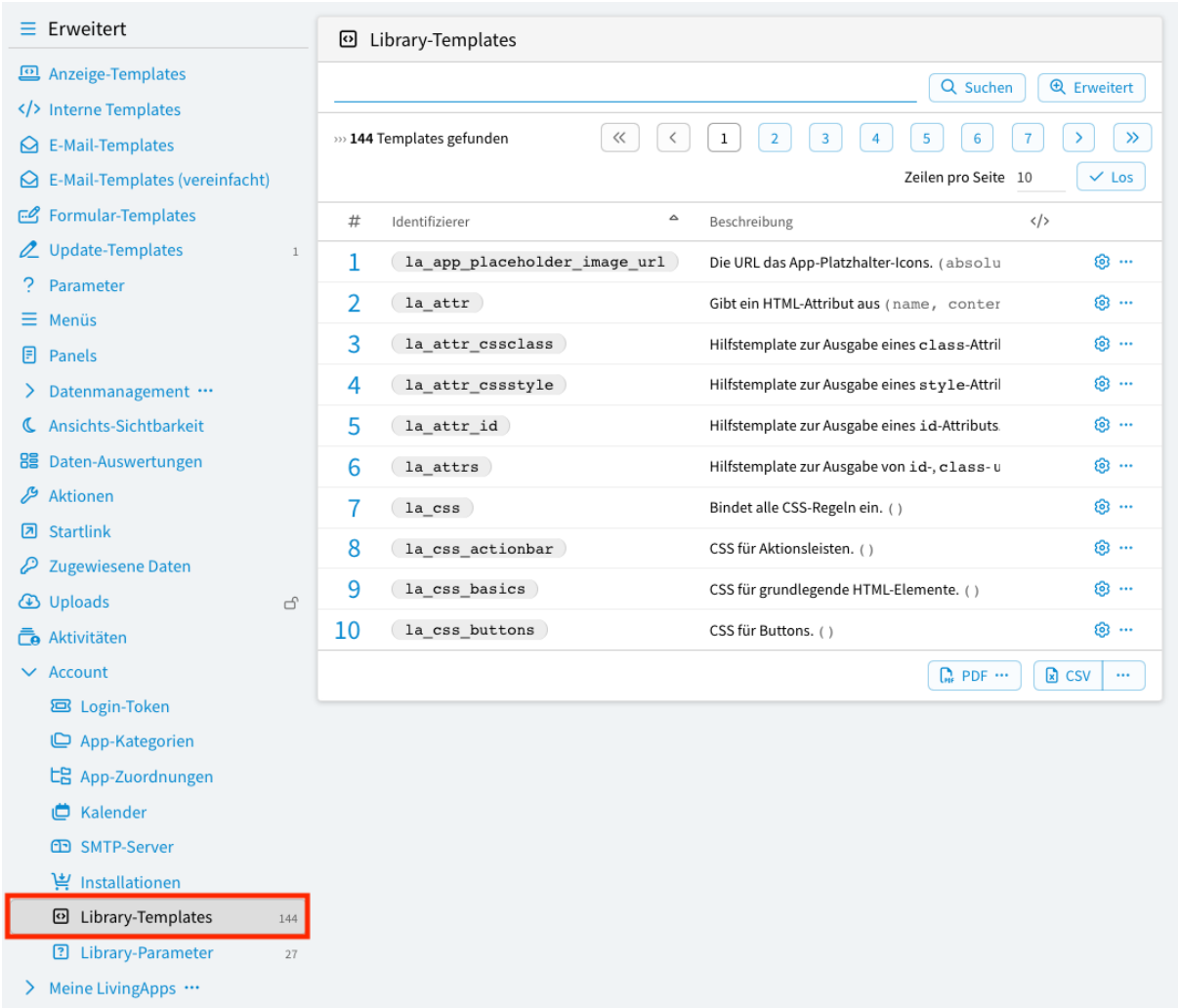

Abb. 1: Library-Templates

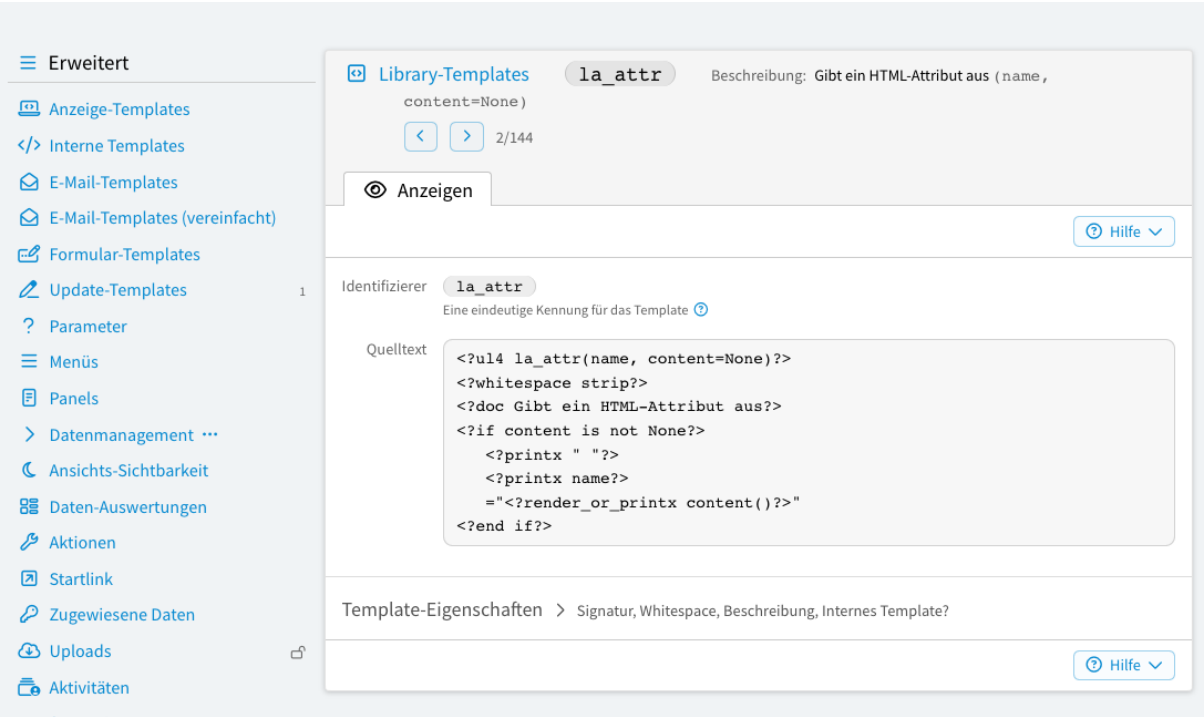

Abb. 2: Library-Template

```
<?render globals.t_la_static_ul4()?>
```

der Reihe nach die Existenz der folgenden Templates überprüft:

- `globals.app.templates.la_static_ul4`
- `globals.app.params.la.value.templates.la_static_ul4`
- `globals.app.params.la.value.params.la.value.templates.la_static_ul4`
- ...

Das erste gefundene Template wird ausgeführt. Wird kein Template gefunden, so wird `la_static_ul4` aus der Template-Library verwendet.

Das heißt:

- zuerst wird überprüft ob `globals.app` ein Template `la_static_ul4` besitzt.
- Ist dies nicht der Fall, wird getestet, ob diese App einen App-Parameter `la` hat, der vom Typ `App` ist. In diesem Fall wird das Template dann in dieser App gesucht.
- Ist es auch dort nicht vorhanden, so wird wiederum der App-Parameter `la` **dieser** App getestet usw.
- Diese Suche bricht ab, sobald eine der Apps in der Kette keinen App-Parameter `la` hat oder dieser Parameter nicht vom Typ `App` ist. In diesem Fall wird dann final das Template `la_static_ul4` aus der Template-Library verwendet.

24.4 Library-Templates als Methoden eines Typs

In der Liste der Library-Templates fällt auf, dass es Templates gibt, die einen Typ haben. Dieser Typ kann *App*, *Control*, *Field*, *File*, *Geo*, *LayoutControl*, *Record MenuItem* oder *Panel* sein. Das bedeutet, dass die jeweiligen Templates diesem Objekt-Typ zugeordnet sind. Der erste Parameter hat immer den Namen *self* und muss beim Aufruf nicht angegeben werden. Stattdessen kann das Template wie eine Methode des Objekts referenziert werden (mit *t_* als Präfix):

..sourcecode:: html+ul4

<?code geo = globals.geo(lat=49.9552129, long=11.5901843)?> <?render geo.t_la_displayhtml()?>

24.5 Namens-Konvention

Der Namen aller Templates in der LivingApps-Template-Library beginnt mit `la_`.

Sie sollten es daher vermeiden, Ihren eigenen internen Templates einen Namen zu geben, der mit `la_` beginnt. Eine Ausnahme bilden selbstverständlich jene Templates, die Sie anlegen, um eines der Templates aus einer LivingApps-Template-Library zu überschreiben.

KAPITEL 25

Library-Parameter

Die Template-Library verwendet einige Parameter mit denen das Aussehen Ihrer Anzeige- und E-Mail-Templates angepasst werden können, wenn Sie die Template-Library verwenden.

25.1 Übersicht über die Parameter

Um zur Liste der Library-Parameter zu gelangen, wählen Sie im Menü einer beliebigen App *Konfiguration* → *Erweitert* und klicken anschließend in der linken Menüleiste auf *Account* und dann auf *Library-Parameter*:

Ein Klick auf einen der Parameter in der Liste führt Sie zur Detailansicht des Parameters:

25.2 Verwenden eines Parameters

Die Parameter in der Template-Library werden von den Templates in der Template-Library verwendet, daher taucht ein solcher Parameter normalerweise nicht in Ihren eigenen Templates auf. Sie könnten aber natürlich trotzdem auf einen Library-Parameter zugreifen und zwar genauso wie Sie auf App-Parameter zugreifen. Um z.B. den Wert das Parameters `la_css_active_color` auszugeben, können Sie folgenden Code benutzen:

```
<?print globals.app.params.la_css_active_color.value?>
```

Genauso wie bei den App-Parametern sind auch hier folgende alternative Aufrufe möglich:

```
<?print globals.app.params.la_static_ul4.value?>
<?print globals.params.la_static_ul4.value?>
<?print globals.app.p_la_static_ul4.value?>
<?print globals.app.pv_la_static_ul4?>
<?print globals.p_la_static_ul4.value?>
<?print globals.pv_la_static_ul4?>
```

25.3 Überschreiben eines Parameters

Wenn Sie innerhalb Ihrer App für einen der Library-Parameter einen anderen Wert verwenden wollen, können Sie das tun indem sie einen App-Parameter mit dem gleichen Namen in den Parametern Ihrer App anlegen.

Sie können dies auch tun, indem Sie ihre Parameter-Wert in einer Ihrer Apps gesammelt in deren App-Parametern ablegen.

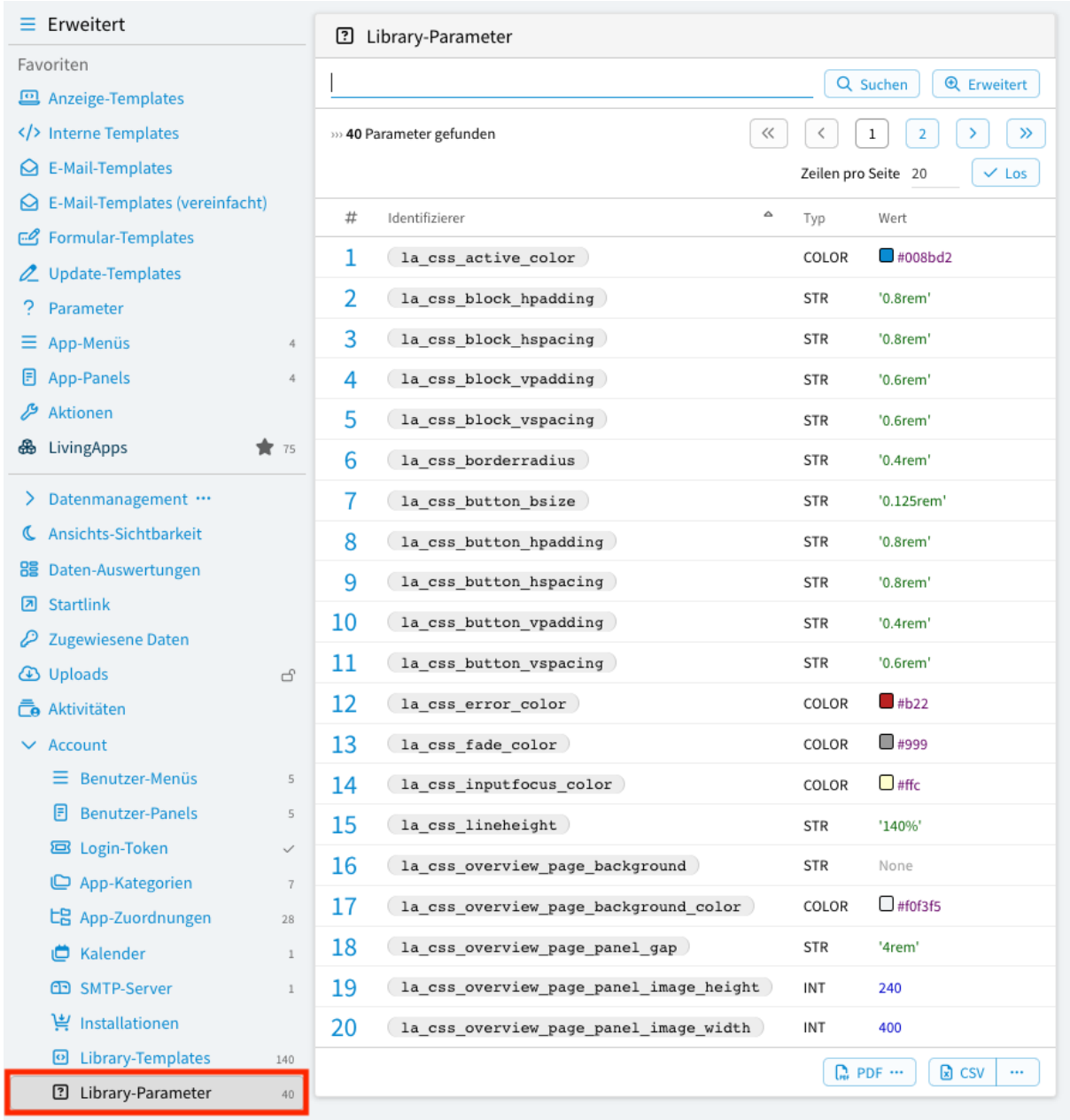

Abb. 1: Library-Parameter

Abb. 2: Library-Parameter

Wenn Sie beispielsweise einen eigenen Wert für `la_css_active_color` anlegen, wird beim Abruf des Parameters mittels folgendem Code:

```
<?print globals.pv_la_css_active_color()?>
```

der Reihe nach die Existenz der folgenden Parameter überprüft:

- `globals.app.params.la_css_active_color`
- `globals.app.params.la.value.params.la_css_active_color`
- `globals.app.params.la.value.params.la.value.params.la_css_active_color`
- ...

Der erste gefundene Parameter wird verwndet. Wird kein Parameter gefunden, so wird `la_css_active_color` aus der Template-Library verwendet.

Das heißt:

- zuerst wird überprüft ob `globals.app` einen Parameter `la_css_active_color` besitzt.
- Ist dies nicht der Fall, wird getestet, ob diese App einen App-Parameter `la` hat, der vom Typ `App` ist. In diesem Fall wird der Parameter dann in dieser App gesucht.
- Ist es auch dort nicht vorhanden, so wird wiederum der App-Parameter `la` **dieser** App getestet usw.
- Diese Suche bricht ab, sobald eine der Apps in der Kette keinen App-Parameter `la` hat oder dieser Parameter nicht vom Typ `App` ist. In diesem Fall wird dann final der Parameter `la_css_active_color` aus der Template-Library verwendet.

25.4 Namens-Konvention

Der Namen aller Parameter in der LivingApps-Template-Library beginnt mit `la_`.

Sie sollten es daher vermeiden, Ihren eigenen Parametern einen Namen zu geben, der mit `la_` beginnt. Eine Ausnahme bilden selbstverständlich jene Parameter, die Sie anlegen, um eines der Parameter-Werte aus der LivingApps-Template-Library zu überschreiben.

KAPITEL 26

vSQL

vSQL wird verwendet um Filterbedingungen und ähnliche Ausdrücke angeben zu können, die für Datenbankabfragen verwendet werden.

vSQL ist eine auf Ausdrücke reduzierte und eingeschränkte Teilmenge von UL4.

26.1 Verwendung

vSQL kommt an folgenden Stellen zum Einsatz:

- als Filter-Bedingung bei *Datenquellen* und *untergeordneten Datensätzen*;
- zur Konfiguration der Sortierreihenfolge im *Datenmanagement*, bei den *Datenquellen* und bei den *untergeordneten Datensätzen*;
- zur Konfiguration der *Zeilenfarben im Datenmanagement*;
- um festzulegen, *welche Datensätze einem Benutzer zugewiesen sind*;
- um festzulegen, *welche Datenaktionen für einen Datensatz zur Verfügung stehen*;
- um festzulegen, *welche Anweisungen einer Datenaktionen ausgeführt werden sollen*;
- um bei einer Datenaktion festzulegen, *auf welchen Wert ein Feld gesetzt werden soll*.

26.2 Beispiele

Die folgende Filterbedingung liefert nur Datensätze, die innerhalb der letzten 30 Tage angelegt wurden:

```
r.createdat >= now() - days(30)
```

Die folgende Filterbedingung liefert nur Datensätze, die von jemanden angelegt oder geändert wurden, dessen Accountname auf `@example.org` endet:

```
r.createdby.email.endswith("@example.org") or r.updatedby.email.endswith("@example.org
↪")
```

26.3 Variablen

Welche Variablen zur Verfügung stehen, hängt davon ab, an welcher Stelle vSQL eingesetzt wird. Insgesamt gibt es folgende Variablen:

user
: [*Benutzer*]
Der eingeloggte Benutzer ist immer über die vSQL-Variable `user` zugänglich, d.h. dass z.B. `user.firstname` der Vorname des eingeloggten Benutzers ist.

r
: [*Datensätze*]
Der Datensatz der für den jeweiligen Zweck getestet werden soll, also der Datensatz der für die Filterbedingung bei Datenquellen gefiltert werden soll, oder bei den Farben im Datenmanagement eingefärbt werden soll, oder bei der Konfiguration für untergeordnete Datensätze (bei der Filterbedingung und der Sortierung) der untergeordnete Datensatz etc.

record
: [*Datensätze*]
Wenn ein Anzeige-Template als Detail-Template aufgerufen wird und wenn ein E-Mail-Template aufgerufen wird, so ist der Detaildatensatz unter der Variable `record` in der Filterbedingung für Datenquellen zugänglich. Das einzige Attribut das aber tatsächlich vorhanden ist, ist `id` (d.h. `record.id` ist die eindeutige 30-stellige Kennung des Detail-Datensatzes.)

lang
: [String]
Die aktuelle Sprache (`globals.lang` in UL4).

mode
: [String]
Der Template-Modus (`globals.mode` in UL4).

search
: [String]
Der aktuelle Suchausdruck für die Ajax-Suche für Applookups vom Untertyp `"choice"` im Eingabeformular.

request.id
: In der Filterbedingung für Datenquellen kann mittels `request.id` auf die eindeutige ID des Requests zugegriffen werden.

request.method
: In der Filterbedingung für Datenquellen kann mittels `request.method` auf die HTTP-Request-Methode zugegriffen werden.

request.url
: In der Filterbedingung für Datenquellen kann mittels `request.url` auf die vollständige URL (mit Query-String) zugegriffen werden.

request.params
: In der Filterbedingung für Datenquellen kann mittels `request.params` auf die Parameter des HTTP-Requests zugegriffen werden (nur bei Anzeige-Templates, nicht bei E-Mail-Templates). `request.params` besitzt zehn Unter-Attribute.

  request.params.str.*
  : Enthält die Parameter als Einzelwerte und interpretiert sie als String. Wenn ein Parametername trotzdem doppelt angegeben wurde, ist der Wert der des ersten Auftretens, d.h. in der URL:

```
http://my.living-apps.de/gateway/apps/1234567890abcdef12345678?template=gurk&
↪x[]=1&y=2&x[]=3
```

  ergibt `request.params.str.x` den String `"1"`. Wurde der Parametername gar nicht angegeben, so ist der Wert `None`.

request.params.strlist.*
Enthält die Parameter als Listen und interpretiert sie als String, d.h. wenn ein Parametername mehrmals angegeben wurde, so ist der Wert eine Liste mit mehreren Strings, wurde er nur einmal angegeben, so enthält die Liste nur einen String, wurde der Parametername gar nicht angegeben, so ist die Liste leer. Bei obiger URL ergibt also `request.params.strlist.x` die Liste `["1", "3"]`, `request.params.strlist.y` ergibt `["2"]` und `request.params.strlist.z` ergibt `[]`.

request.params.int.*
Enthält die Parameter als Einzelwerte und interpretiert sie als ganze Zahl. Wird ein Parametername trotzdem doppelt angegeben, ist der Wert der des ersten Auftretens, d.h. in der URL:

```
http://my.living-apps.de/gateway/apps/1234567890abcdef12345678?template=gurk&
↪x[]=1&y=2&x[]=3
```

ergibt `request.params.int.x` die ganze Zahl `1`. Wurde der Parametername gar nicht angegeben, oder entspricht er nicht dem Format für `INT`-Parameter, so ist der Wert `None`.

request.params.intlist.*
Enthält die Parameter als Listen und interpretiert sie als ganze Zahl, d.h. wenn ein Parametername mehrmals angegeben wurde, so ist der Wert eine Liste mit mehreren ganzen Zahlen, wurde er nur einmal angegeben, so enthält die Liste nur eine ganze Zahl. Wurde der Parametername gar nicht angegeben, oder entspricht er nicht dem Format für `INT`-Parameter, so ist die Liste leer. Bei obiger URL ergibt also `request.params.intlist.x` die Liste `[1, 3]`, `request.params.intlist.y` ergibt `[2]` und `request.params.intlist.z` ergibt `[]`.

request.params.float.*
Enthält die Parameter als Einzelwerte und interpretiert sie als Zahl mit Nachkommastellen. Wird ein Parametername trotzdem doppelt angegeben, ist der Wert der des ersten Auftretens, d.h. in der URL:

```
http://my.living-apps.de/gateway/apps/1234567890abcdef12345678?template=gurk&
↪x[]=1.1&y=2.2&x[]=3.3
```

ergibt `request.params.float.x` die Zahl `1.1`. Wurde der Parametername gar nicht angegeben, oder entspricht er nicht dem Format für `NUMBER`-Parameter, so ist der Wert `None`.

request.params.floatlist.*
Enthält die Parameter als Listen und interpretiert sie als Zahl mit Nachkommastellen, d.h. wenn ein Parametername mehrmals angegeben wurde, so ist der Wert eine Liste mit mehreren Zahlen, wurde er nur einmal angegeben, so enthält die Liste nur eine Zahl. Wurde der Parametername gar nicht angegeben, oder entspricht er nicht dem Format für `NUMBER`-Parameter, so ist die Liste leer. Bei obiger URL ergibt also `request.params.floatlist.x` die Liste `[1.1, 3.3]`, `request.params.floatlist.y` ergibt `[2.2]` und `request.params.floatlist.z` ergibt `[]`.

request.params.date.*
Enthält die Parameter als Einzelwerte und interpretiert sie als Datum. Wird ein Parametername trotzdem doppelt angegeben, ist der Wert der des ersten Auftretens, d.h. in der URL:

```
http://my.living-apps.de/gateway/apps/1234567890abcdef12345678?template=gurk&
↪x[]=2018-05-01&y=2018-06-01&x[]=2018-07-01
```

ergibt `request.params.date.x` das Datum `01.05.2018`. Das Format für `DATE`-Parameter ist `YYYY-MM-DD`. Zum Beispiel kann mit `request.params.date.datum` in der Filterbedingung der Datenquelle auf `datum=2000-02-29` in der URL zugegriffen werden. Wurde der Parametername gar nicht angegeben, oder entspricht er nicht dem Format für `DATE`-Parameter, so ist der Wert `None`.

request.params.datelist.*
Enthält die Parameter als Listen und interpretiert sie als Datum, d.h. wenn ein Parametername mehrmals angegeben wurde, so ist der Wert eine Liste mit mehreren Daten, wurde er nur einmal angegeben, so enthält die Liste nur ein Datum. Wurde der Parametername gar nicht angegeben, oder entspricht er nicht dem Format für `DATE`-Parameter, so ist die Liste leer. Bei obiger URL ergibt also `request.params.datelist.x` die Liste `[01.05.2018, 01.07.2018]`, `request.params.datelist.y` ergibt `[01.06.2018]` und `request.params.datelist.z` ergibt `[]`.

request.params.datetime.*
Enthält die Parameter als Einzelwerte und interpretiert sie als Datum mit Uhrzeit. Wird ein Parametername trotzdem doppelt angegeben, ist der Wert der des ersten Auftretens, d.h. in der URL:

```
http://my.living-apps.de/gateway/apps/1234567890abcdef12345678?template=gurk&
↪x[]=2018-05-01T14:30:55&y=2018-06-01T09:50:22&x[]=2018-07-01T12:30:05
```

ergibt `request.params.datetime.x` das Datum mit Uhrzeit `01.05.2018 14:30:55`. Das Format für `DATETIME`-Parameter ist `YYYY-MM-DDTHH:MM:SS`. Zum Beispiel kann mit `request.params.datetime.datum` in der Filterbedingung der Datenquelle auf `datum=2000-02-29T14:30:29` in der URL zugegriffen werden. Wurde der Parametername gar nicht angegeben, oder entspricht er nicht dem Format für `DATETIME`-Parameter, so ist der Wert `None`.

request.params.datetimelist.*
Enthält die Parameter als Listen und interpretiert sie als Datum mit Uhrzeit, d.h. wenn ein Parametername mehrmals angegeben wurde, so ist der Wert eine Liste mit mehreren Daten und Uhrzeiten, wurde er nur einmal angegeben, so enthält die Liste nur ein Datum mit Uhrzeit. Wurde der Parametername gar nicht angegeben, oder entspricht er nicht dem Format für `DATETIME`-Parameter, so ist die Liste leer. Bei obiger URL ergibt also `request.params.datetimelist.x` die Liste `[01.05.2018 14:30:55, 01.07.2018 12:30:05]`, `request.params.datetimelist.y` ergibt `[01.06.2018 09:50:22]` und `request.params.datetimelist.z` ergibt `[]`.

request.headers
In der Filterbedingung für Datenquellen kann mittels `request.headers` auf die Header des HTTP-Requests zugegriffen werden (nur bei Anzeige-Templates, nicht bei E-Mail-Templates). `request.headers` besitzt zwei Unter-Attribute.

request.headers.str.*
Enthält die Header als Einzelwerte und interpretiert sie als String. Wenn ein Header mehrfach angegeben wurde, ist es der Wert der des ersten Auftretens.

request.headers.strlist.*
Enthält die Header als Listen und interpretiert sie als String, d.h. wenn ein Header mehrmals angegeben wurde, so ist der Wert eine Liste mit mehreren Strings, wurde er nur einmal angegeben, so enthält die Liste nur einen String, wurde der Header gar nicht angegeben, so ist die Liste leer.

request.cookies.*
In der Filterbedingung für Datenquellen kann mittels `request.cookies` auf die Cookies des HTTP-Requests zugegriffen werden (nur bei Anzeige-Templates, nicht bei E-Mail-Templates).

26.4 Objekte

In vSQL stehen einige Objekte mit Attributen zu Verfügungen, die in Ausdrücken verwendet werden können.

Im folgenden werden diese Objekte beschrieben.

26.4.1 Benutzer

Benutzer-Objekte sind z.B. Informationen über den eingeloggten Benutzer oder über den Benutzer, der eine App oder einen Datensatz angelegt oder geändert hat. Benutzer-Objekte besitzen folgende Attribute:

id
[*STR*]
Die eindeutige Datenbank-Identifizierer des Benutzers;

email
[*STR*]
Die Email-Adresse des Benutzers (gleichzeitig auch der Accountname);

firstname
[*STR*]
Der Vorname des Benutzers;

surname
[*STR*]
Der Nachname des Benutzers;

initials
[*STR*]
Die Initialen des Benutzers.

26.4.2 App

App-Objekte enthalten Informationen über die App. Dies kann einerseits die App sein, innerhalb der sich der Benutzer befindet, andererseits auch die App, zu der ein Datensatz gehört. Dies kann evtl. eine andere App sein, da sich z.B. in den Anzeige-, Formular-, Update- oder E-Mail-Templates über Datenquellen konfigurieren läßt, daß dem Template auch Datensätze aus anderen Apps zur Verfügung stehen. Außerdem sind über verknüpfte Apps andere Apps als die Ausgangs-App zugänglich. App-Objekte besitzen folgende Attribute:

id
[*STR*]
Die eindeutige Datenbank-Identifizierer der App.

name
[*STR*]
Der Name der App.

description
[*STR*]
Die Beschreibung der App.

createdat
[*DATETIME*]
Der Zeitpunkt, zu dem die App erstellt wurde.

createdby
[*Benutzer*]
Der Benutzer, der die App erstellt hat.

updatedat
[*DATETIME*]
Der Zeitpunkt, zu dem die App das letzte Mal geändert wurde. (Wurde die App noch nicht geändert, so ist `updatedat` `None`.)

updatedby
[*Benutzer*]
Der Benutzer, der die App zuletzt geändert hat. (Wurde die App noch nicht geändert, so sind alle Attribute des `updatedby`-Benutzers `None`.)

installation
Wenn die Applikation durch einen Installationsvorgang erzeugt wurde, ist `installation` ein `Installation`-Objekt. Ansonsten ist `installation` `None`. Ein `Installation`-Objekt hat folgendes Attribut:

name
[*STR*]
Der Name der Installation.

p_<identifier>
[*App-Parameter*]
Alle vom Benutzer angelegten App-Parameter stehen über Shortcut-Attribute zur Verfügung. D.h., dass beispielsweise der Parameter `beispiel` unter `app.p_beispiel.value` zur Verfügung steht.

26.4.3 App-Parameter

Hier können sie auf die Parameter der Applikation, die unter *Konfiguration* → *Erweitert* in der *Parameter*-Maske angelegt wurden, zugreifen. Dies funktioniert nicht bei zu filternden Apps in der Datenquelle. App-Parameter-Objekte besitzen folgende Attribute:

id
[*STR*]
Der eindeutige Datenbank-Identifizierer des Parameters.

identifier
[*STR*]
Die eindeutige Bezeichnung des Parameters.

description
[*STR*]
Die Beschreibung des Parameters.

value
[Objekt]
Der Wert des Objekts. Dabei hängt der Typ des Wertes vom Typ des Parameters ab.

createdat
[*DATETIME*]
Der Zeitpunkt, zu dem der Parameter erstellt wurde.

createdby
[*Benutzer*]
Der Benutzer, der den Parameter erstellt hat.

updatedat
[*DATETIME*]
Der Zeitpunkt, zu dem der Parameter das letzte Mal geändert wurde. (Wurde der Parameter noch nicht geändert, so ist `updatedat` `None`.)

updatedby
[*Benutzer*]
Der Benutzer, der den Parameter zuletzt geändert hat. (Wurde der Parameter noch nicht geändert, so sind alle Attribute des `updatedby`-Benutzers `None`.)

26.4.4 Datensätze

Datensatz-Objekte haben folgende Attribute:

id
[*STR*]
Die eindeutige Datenbank-Identifizierer des Datensatzes.

url
[*STR*]
Die URL für die Bearbeiten-Ansicht.

createdat
[*DATETIME*]
Der Zeitpunkt, zu dem der Datensatz erzeugt wurde.

createdby
[*Benutzer*]
Der Benutzer, der den Datensatz erstellt hat.

updatedat
[*DATETIME*]
Der Zeitpunkt, zu dem der Datensatz das letzte Mal geändert wurde. (Wurde der Datensatz noch nicht geändert, so ist `updatedat` `None`.)

updatedby
[*Benutzer*]
Der Benutzer, der den Datensatz zuletzt geändert hat. (Wurde der Datensatz noch nicht geändert, so sind alle Attribute des `updatedat`-Benutzers `None`.)

app
[*App*]
Die App, zu der dieser Datensatz gehört.

v_<identifier>
[Objekt]
Alle vom Benutzer angelegten Felder sind über mit Präfixen versehene Attribute verfügbar. D.h. wenn `r` das Datensatz-Objekt ist, ist beispielsweise das Feld mit den Identifier `titel` als `r.v_titel` verfügbar.

26.5 Datentypen, Attribute und Methoden

vSQL unterstützt eine Teilmenge der Datentypen, die von UL4 unterstützt werden. Diese und die damit in Verbindung stehenden Methoden werden im folgenden erklärt.

26.5.1 None (NONE)

`None` ist die Null-Konstante in vSQL (und UL4).

26.5.2 Boolsche Werte (BOOL)

`True` oder `False`

26.5.3 Ganze Zahlen (INT)

Beispielsweise `123`

26.5.4 Zahlen mit Nachkommastellen (NUMBER)

Beispielsweise `123.45`

26.5.5 Strings (STR)

String-Konstanten werden in vSQL (wie in UL4) folgendermaßen geschrieben: `"foo"` oder `'foo'`.

Methoden

Strings besitzen in vSQL folgende Methoden:

str.lower()

Konvertiert den String-Wert `str` in Kleinbuchstaben.

Beispiele:

Ausdruck	Wert
`"GURK".lower()`	`"gurk"`

str.upper()

Konvertiert den String-Wert `str` in Großbuchstaben.

Beispiele:

Ausdruck	Wert
`"Gurk".upper()`	`"GURK"`

`str.startswith(präfix)`

Testet, ob der String-Wert `str` mit dem übergebenen Präfix `präfix` beginnt. `präfix` kann auch eine Liste von Strings sein, dann gibt `x.startswith(y)` `True` zurück, wenn `x` mit einem der Strings in `y` beginnt.

Beispiele:

Ausdruck	Wert
`"Gurk".startswith("Gu")`	`True`
`"Gurk".startswith("gu")`	`False`
`"Gurk".startswith("")`	`True`
`"Gurk".startswith(["Gu", "Hu", "Ku"])`	`True`
`"Gurk".startswith(["Hu", "Ku"])`	`False`

`str.endswith(suffix)`

Testet, ob der String-Wert `str` mit dem übergebenen Suffix `suffix` endet. `suffix` kann auch eine Liste von Strings sein, dann gibt `x.endswith(y)` `True` zurück, wenn `x` mit einem der Strings in `y` endet.

Beispiele:

Ausdruck	Wert
`"Gurk".endswith("rk")`	`True`
`"Gurk".endswith("rz")`	`False`
`"Gurk".endswith("")`	`True`
`"Gurk".endswith(["rk", "rz", "nz"])`	`True`
`"Gurk".endswith(["rz", "nz"])`	`False`

`str.strip()` oder `str.strip(None)`

Entfernt Leerzeichen und anderen Whitespace von beiden Enden des String-Wertes `str`.

Beispiele:

Ausdruck	Wert
`"gurk".strip()`	`"gurk"`
`" gurk ".strip()`	`"gurk"`
`" \t\r\n gurk \t\r\n ".strip()`	`"gurk"`
`" \t\r\n gurk \t\r\n hurz \t\r\n ".strip()`	`"gurk \t\r\n hurz"`
`" gurk ".strip(None)`	`"gurk"`

`str.strip(zeichen)`

Entfernt alle Zeichen die in dem String-Wert `zeichen` vorkommen von beiden Enden des String-Wertes `str`.

Beispiele:

Ausdruck	Wert
`"abrakadabra".strip("a")`	`"brakadabr"`
`"abrakadabra".strip("rab")`	`"kad"`

`str.lstrip()`, `str.lstrip(None)` oder `str.lstrip(zeichen)`

Ähnlich wie `strip()`, jedoch werden nur Zeichen am Anfang des Strings entfernt.

Beispiele:

Ausdruck	Wert
`"gurk".lstrip()`	`"gurk"`
`" gurk ".lstrip()`	`"gurk "`
`" \t\r\n gurk \t\r\n ".lstrip()`	`"gurk \t\r\n "`
`" \t\r\n gurk \t\r\n hurz \t\r\n ".lstrip()`	`"gurk \t\r\n hurz \t\r\n "`
`" gurk ".lstrip(None)`	`"gurk "`
`"abrakadabra".lstrip("a")`	`"brakadabra"`
`"abrakadabra".lstrip("rab")`	`"kadabra"`

`str.rstrip()`, `str.rstrip(None)` oder `str.rstrip(zeichen)`

Ähnlich wie `strip()`, jedoch werden nur Zeichen am Ende des Strings entfernt.

Beispiele:

Ausdruck	Wert
`"gurk".rstrip()`	`"gurk"`
`" gurk ".rstrip()`	`" gurk"`
`" \t\r\n gurk \t\r\n ".rstrip()`	`" \t\r\n gurk"`
`" \t\r\n gurk \t\r\n hurz \t\r\n ".rstrip()`	`" \t\r\n gurk \t\r\n hurz"`
`" gurk ".rstrip(None)`	`" gurk"`
`"abrakadabra".rstrip("a")`	`"abrakadabr"`
`"abrakadabra".rstrip("rab")`	`"abrakad"`

`str.find(suchstring)`

Gibt den Offset zurück an dem `suchstring` das erste Mal in `str` auftaucht. Wird `suchstring` nicht gefunden, wird `-1` zurückgegeben.

Beispiele:

Ausdruck	Wert
`"abrakadabra".find("a")`	`0`
`"abrakadabra".find("ra")`	`2`
`"abrakadabra".find("rab")`	`-1`

`str.find(suchstring, start, stop)`

Ähnlich wie der Aufruf mit einem Argument, jedoch wird die Suche auf den Unterstring `str[start:stop]` beschränkt.

Beispiele:

Ausdruck	Wert
`"abrakadabra".find("a", 0)`	`0`
`"abrakadabra".find("a", 1)`	`3`
`"abrakadabra".find("a", 11)`	`-1`
`"abrakadabra".find("a", -1)`	`10`
`"abrakadabra".find("a", -4)`	`7`
`"abrakadabra".find("ra", 3)`	`9`
`"abrakadabra".find("ra", 3, 11)`	`9`
`"abrakadabra".find("ra", 3, 10)`	`-1`
`"abrakadabra".find("ra", 3, -7)`	`9`
`"abrakadabra".find("ra", 3, -8)`	`-1`

`str.rfind(suchstring)` oder `str.rfind(suchstring, start, stop)`

Ähnlich wie `find()`, die Suche beginnt jedoch am Ende des Strings.

Beispiele:

Ausdruck	Wert
`"abrakadabra".rfind("a", 0)`	`10`
`"abrakadabra".rfind("a", 11)`	`10`
`"abrakadabra".rfind("a", 10)`	`7`
`"abrakadabra".rfind("a", 4, -2)`	`7`

`str.split(sep)`

Trennt den String `str` an den Stellen, an denen `sep` vorkommt und liefert die resultierende Liste von Strings zurück.

Beispiele:

Ausdruck	Wert
`"A+B+C".split("+")`	`["A", "B", "C"]`
`"++A++B++C++".split("+")`	`["", "", "A", "", "B", "", "C", "", ""]`

`str.split(sep, maxsplit)`

Ähnlich `str.split(sep)` jedoch wird `str` höchstens `maxsplit` mal getrennt.

Beispiele:

Ausdruck	Wert
`"A+B+C".split("+", 2)`	`["A", "B", "C"]`
`"A+B+C".split("+", 1)`	`["A", "B+C"]`

`str.join(liste)`

Verbindet alle Strings in `liste` mit dem Trennstring `str` zu einen einzigen String.

Beispiele:

Ausdruck	Wert
`"+".join(["A", "B", "C"])`	`"A+B+C"`

26.5.6 Datum (DATE) und Datum/Uhrzeit (DATETIME)

@(2000-02-29)
Datums-Konstante

@(2000-02-29T12:34:56)
Datums/Zeit-Konstante

Außerdem können DATE-Werte mit der Funktion date und DATETIME-Werte mit der Funktion datetime erzeugt werden.

Attribute

DATE- und DATETIME-Werte besitzen folgender Attribute:

date.year

Das Jahr des Datumswertes date.

Beispiele:

Ausdruck	Wert
@(2000-02-29).year	2000

date.month

Das Monat des Datumswertes date.

Beispiele:

Ausdruck	Wert	Bemerkung
@(2000-02-29).month	2	Februar

date.day

Der Tag des Datumswertes date

Beispiele:

Ausdruck	Wert
@(2000-02-29).day	29

date.weekday

Der Wochentag des Datumswertes date. 0 ist Montag, 1 ist Dienstag, ..., 6 ist Sonntag.

Beispiele:

Ausdruck	Wert	Bemerkung
@(2000-02-29).weekday	1	Dienstag

date.yearday

Die Anzahl der Tage seit Beginn das Jahres.

Beispiele:

Ausdruck	Wert
@(2000-01-01).yearday	1
@(2000-02-29).yearday	60
@(2000-12-31).yearday	366
@(2001-12-31).yearday	365

Zusätzlich besitzen DATETIME-Werte noch die Attribute:

datetime.hour

Die Stundenangabe des Datumswertes datetime.

Beispiele:

Ausdruck	Wert
@(2000-02-29T12:34:56).hour	12

datetime.minute

Die Minutenangabe des Datumswertes datetime.

Beispiele:

Ausdruck	Wert
@(2000-02-29T12:34:56).minute	34

datetime.second

Die Sekundenangabe des Datumswertes datetime.

Beispiele:

Ausdruck	Wert
@(2000-02-29T12:34:56).second	56

26.5.7 Zeiträume (DATEDELTA und DATETIMEDELTA)

DATEDELTA-Werte stellen die Differenz zwischen zwei DATE-Werten dar. DATEDELTA-Werte können mit den Funktionen *timedelta* und *days* erzeugt werden.

DATETIMEDELTA-Werte stellen die Differenz zwischen zwei DATETIME-Werten dar und können mit den Funktionen *timedelta*, *hours*, *minutes* und *seconds* erzeugt werden.

26.5.8 Monatszeiträume (MONTHDELTA)

MONTHDELTA-Werte können mit den Funktionen *monthdelta*, *years* und *months* erzeugt werden.

26.5.9 Farben (COLOR)

Farb-Werte (mit oder ohne Alpha-Wert) werden in vSQL (und UL4) ähnlich geschrieben wie in CSS: `#rgba`, `#rrggbb`, `#rgba` oder `#rrggbbaa` (wobei `r`, `g`, `b` und `a` jeweils Hexadezimal-Ziffern sind).

Beispiele:

Wert	Bemerkung
`#000`	Schwarz
`#fff`	Weiß
`#f00`	Rot
`#0f0`	Grün
`#00f`	Blau
`#7f7f7f`	Mittelgrau
`#00ff007f`	Halbtransparentes Grün

Mehr zum RGB-Farbmodel in der Wikipedia: https://de.wikipedia.org/wiki/RGB-Farbraum.

Attribute

Farben haben folgende Attribute:

`color.r`

Rot-Anteil der Farbe `color` (Zahl zwischen 0 und 255);

`color.g`

Grün-Anteil der Farbe `color` (Zahl zwischen 0 und 255);

`color.b`

Blau-Anteil der Farbe `color` (Zahl zwischen 0 und 255);

`color.a`

Transparenz der Farbe `color` (0 = komplett transparent, 255 = komplett undurchsichtig).

Methoden

Farben haben folgende Methode:

`color.lum()`

Gibt die Helligkeit der Farbe `color` zurück. Das Ergebnis liegt zwischen 0 (dunkel) und 1 (hell).

Beispiele:

Ausdruck	Wert
`#000.lum()`	`0.0`
`#808080.lum()`	`0.5019607843137255`
`#fff.lum()`	`1.0`

Funktionen

Desweiteren gibt es folgende Funktion um eine Farben zu erzeugen:

`rgb(r, g, b, a)`

Dabei ist `r` der Rot-Anteil, `g` der Grün-Anteil, `b` der Blau-Anteil und `a` die Transparenz.

Diese Werte müssen jeweils zwischen 0 und 1 liegen. Der Defaultwert für a ist 1.

Beispiele:

Ausdruck	Wert	Bemerkung
`rgb(0, 0, 0)`	`#000`	Schwarz
`rgb(1, 1, 1)`	`#fff`	Weiß
`rgb(1, 0, 0)`	`#f00`	Rot
`rgb(0.5, 0.5, 0.5)`	`#7f7f7f`	Mittelgrau
`rgb(0, 1, 0, 0.5)`	`#00ff007f`	Halbtransparentes Grün

Für Farben gibt es zusätzlich den Farbmisch-Operator `%`. Dabei ist `a % b` die Farbe, die sich ergibt wenn die Farbe `a` auf einem Hintergrund der Farbe `b` aufgetragen würde.

Beispiele:

Ausdruck	Wert	Bemerkung
`#fff % #000`	`#fff`	intransparerentes Weiß auf Schwarz also Weiß
`#fff8 % #000`	`#888`	halbtransparentes Weiß auf Schwarz also Grau
`#f00a % #0f0`	`#a50`	halbtransparentes Rot auf Grün also Gelb

26.5.10 Geo-Koordinaten (`GEO`)

Geo-Koordinaten können in vSQL mit der Funktion `geo()` erzeugt werden. Es gibt eine Version mit zwei Argumenten (Breiten- und Längengrad) sowie drei Argumenten (Breitengrad, Längengrad und Beschreibung).

Beispiele:

Ausdruck	Beschreibung
`geo(49.955267, 11.591212)`	Breiten- und Längengrad
`geo(49.955267, 11.591212, "LivingLogic AG, Bayreuth")`	Breitengrad, Längengrad und Info

Attribute

Geo-Koordinaten haben folgende Attribute:

`geo.lat`

Der Breitengrad.

Beispiel:

Ausdruck	Wert
`geo(49.955267, 11.591212, "LivingLogic AG, Bayreuth").lat`	`49.955267`

geo.long

Der Längengrad.

Beispiel:

Ausdruck	Wert
`geo(49.955267, 11.591212, "LivingLogic AG, Bayreuth").long`	`11.591212`

geo.info

Die Beschreibung des Ortes.

Beispiel:

Ausdruck	Wert
`geo(49.955267, 11.591212, "LivingLogic AG, Bayreuth").info`	`"LivingLogic AG, Bayreuth"`

Funktionen

Desweiteren gibt es eine Funktion zur Verarbeitung von Geo-Koordinaten.

dist(geo1, geo2)

Berechnet die Entfernung der Geo-Koordinaten `geo1` und `geo2` in Kilomettern.

Beispiel:

Ausdruck	Wert
`dist(geo(49.955267, 11.591212, "LivingLogic AG, Bayreuth"), geo(51.5319695, 9.9416986, "Albanikirchhof, 37085 Göttingen, Deutschland"))`	210,5

26.5.11 Listen

vSQL unterstützt Listen, wenn die Einträge einen der folgende Typen haben: `INT`, `NUMBER`, `STR`, `DATE` oder `DATETIME`. Außerdem müssen alle Einträge denselben Typ haben. Solange nicht alle Werte `None` sind, sind auch `None`-Einträge in der Liste erlaubt.

Beispiele

Ausdruck	Bemerkung
`[1, 2, 3, None, 5]`	`INT`-Liste
`[1.0, 2.0, 3.0, 17.23]`	`NUMBER`-Liste
`["gurk", "hurz", None, "hinz", "kunz"]`	`STR`-Liste
`[@(2000-02-29), @(2019-02-28), today()]`	`DATE`-Liste
`[@(2000-02-29T12:34), @(2019-02-28T12:34), now()]`	`DATETIME`-Liste

26.6 Operatoren

vSQL unterstützt folgende Teilmenge der UL4-Operatoren:

26.6.1 `x == y` (Vergleich auf Gleichheit)

Gibt zurück ob `x` und `y` den gleichen Wert haben.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

26.6.2 `x != y` (Vergleich auf Ungleichheit)

Gibt zurück ob `x` und `y` einen unterschiedlichen Wert haben.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

26.6.3 `x < y` (Vergleich auf kleiner)

Gibt zurück ob `x` kleiner als `y` ist.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

Dabei müssen die Datentypen von `x` und `y` kompatibel sein. (`"gurk" < 42` wird beispielsweise nicht unterstützt.)

26.6.4 `x <= y` (Vergleich auf kleiner/gleich)

Gibt zurück ob `x` kleiner oder gleich `y` ist.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

Dabei müssen die Datentypen von `x` und `y` kompatibel sein. (`"gurk" <= 42` wird beispielsweise nicht unterstützt.)

26.6.5 `x > y` (Vergleich auf größer)

Gibt zurück ob `x` größer als `y` ist.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

Dabei müssen die Datentypen von `x` und `y` kompatibel sein. (`"gurk" > 42` wird beispielsweise nicht unterstützt.)

26.6.6 `x >= y` (Vergleich auf größer/gleich)

Gibt zurück ob `x` kleiner oder gleich `y` ist.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

Dabei müssen die Datentypen von `x` und `y` kompatibel sein. (`"gurk" >= 42` wird beispielsweise nicht unterstützt.)

26.6.7 `x + y` (Addition)

Addition von Zahlen, außerdem String- und Listen-Verkettung sowie Datumsarithmetik.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

Beispiele:

Ausdruck	Wert
`False + True`	`1`
`2 + True`	`3`
`17 + 23`	`40`
`"gurk" + "hurz"`	`"gurkhurz"`
`["gurk", "hurz"] + ["hinz", "kunz"]`	`["gurk", "hurz", "hinz", "kunz"]`
`@(2000-02-29) + days(1)`	`@(2000-03-01)`
`@(2000-02-29T12:34:56) + days(1)`	`@(2000-03-01T12:34:56)`
`@(2000-02-29T12:34:56) + minutes(1)`	`@(2000-02-29T12:35:56)`
`@(2000-02-29) + months(1)`	`@(2000-03-29)`
`@(2000-02-29T12:34:56) + months(1)`	`@(2000-03-29T12:34:56)`
`days(7) + days(7)`	`timedelta(14)`
`days(1) + hours(1)`	`timedelta(1, 3600)`
`hours(1) + minutes(30)`	`timedelta(0, 5400)`
`months(6) + months(6)`	`monthdelta(12)`

26.6.8 `x - y` (Subtraktion)

Subtraktion von Zahlen sowie Datumsarithmetik.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

Beispiele:

Ausdruck	Wert
`0 - True`	`-1`
`7 - 3`	`4`
`@(2000-03-01) - @(2000-02-29)`	`timedelta(1)`
`@(2000-03-01) - days(1)`	`@(2000-02-29)`
`days(1) - hours(23)`	`timedelta(0, 3600)`

26.6.9 `x * y` (Multiplikation)

Multiplikation von Zahlen, sowie String- und Listen-Wiederholung sowie Datumsarithmetik.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

Beispiele:

Ausdruck	Wert
`False * 2`	`0`
`7 * True`	`7`
`3 * 7`	`21`
`2 * "gurk"`	`"gurkgurk"`
`2 * ["gurk", "hurz"]`	`"["gurk", "hurz", "gurk", "hurz"]`
`False * "hurz"`	`""`
`days(7) * 2`	`timedelta(14)`
`hours(6) * 4`	`timedelta(1)`
`monthdelta(6) * 2`	`monthdelta(12)`

26.6.10 `x / y` (Division)

Divison von Zahlen sowie Datumsarithmetik.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

Beispiele:

Ausdruck	Wert	Bemerkung
`4 / True`	`4.0`	
`False / 2`	`0.0`	
`7 / 2`	`3.5`	
`7 / 0`	`None`	Fehler
`days(1) / 24`	`timedelta(0, 3600)`	
`seconds(2) / True`	`timedelta(0, 2)`	

26.6.11 `x // y` (Ganzzahl-Division)

Division von Zahlen mit Rundung auf ganze Zahlen.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

Beispiele:

Ausdruck	Wert	Bemerkung
`4 // True`	`4`	
`False // 2`	`0`	
`7 // 2`	`3`	
`7 // 0`	`None`	Fehler
`days(7) // 3`	`timedelta(2)`	
`monthdelta(12) // 3`	`monthdelta(4)`	

26.6.12 `x % y` (Modulo-Operator)

Modulo-Operator bzw. Farbmischung.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

Beispiele:

Ausdruck	Wert
`13 % 10`	`3`
`True % 2`	`1`
`6.5 % 2.5`	`1.5`
`#fff % #000`	`#fff`
`#fff8 % #000`	`#888`
`#f00a % #0f0`	`#a50`

26.6.13 `x and y` (Boolsche „Und“-Verknüpfung)

Das Ergebnis ist `y` wenn `bool(x)` `True` ist, ansonsten `x`.

Dabei müssen die Datentypen von `x` und `y` kompatibel sein. (`"gurk" and 42` wird beispielsweise nicht unterstützt.)

Siehe auch

bool().

26.6.14 `x or y` (Boolsche „Oder“-Verknüpfung)

Das Ergebnis ist `y` wenn `bool(x)` `False` ist, ansonsten `x`.

Dabei müssen die Datentypen von `x` und `y` kompatibel sein. (`"gurk" or 42` wird beispielsweise nicht unterstützt.)

Siehe auch

bool().

26.6.15 `x in y` (Enthalten-Test)

Testet ob `x` in `y` enthalten ist.

Beispiele:

Ausdruck	Wert
`"u" in "gurk"`	`True`
`"ur" in "gurk"`	`True`
`"hu" in "gurk"`	`False`
`"u" in ["gurk", "hurz", "hinz", "kunz"]`	`False`
`"hinz" in ["gurk", "hurz", "hinz", "kunz"]`	`True`

26.6.16 `x not in y` (Nicht-Enthalten-Test)

Testet ob `x` nicht in `y` enthalten ist.

Beispiele:

Ausdruck	Wert
`"u" not in "gurk"`	`False`
`"ur" not in "gurk"`	`False`
`"hu" not in "gurk"`	`True`
`"u" not in ["gurk", "hurz", "hinz", "kunz"]`	`True`
`"hinz" not in ["gurk", "hurz", "hinz", "kunz"]`	`False`

26.6.17 `x is y` (Identitäts-Test)

Der Wert ist derselbe wie bei `x == y`, jedoch muß `x` oder `y` ein literales `None` sein.

26.6.18 `x is not y` (Negativer Identitäts-Test)

Der Wert ist derselbe wie bei `x != y`, jedoch muß `x` oder `y` ein literales `None` sein.

26.6.19 `x[index]` (Index-Zugriff)

Holt das `index`-te Zeichen aus dem String `x` oder den `index`-ten Eintrag aus der Liste `x`. Negative Werte für `index` werden als relativ zum Ende des Strings bzw. der Liste interpretiert.

Beispiele:

Ausdruck	Wert
`"gurk"[0]`	`"g"`
`"gurk"[2]`	`"r"`
`"gurk"[5]`	`None`
`"gurk"[-1]`	`"k"`
`"gurk"[-3]`	`"u"`
`"gurk"[-5]`	`None`
`["gurk", "hurz", "hinz", "kunz"][0]`	`"gurk"`
`["gurk", "hurz", "hinz", "kunz"][-2]`	`"hinz"`

26.6.20 `x[start:stop]` (Slice-Zugriff)

Gibt einen Unterstring des Strings `x` von Index `start` bis Index `stop` zurück, bzw. gibt eine Unterliste der Liste `x` von Index `start` bis Index `stop` zurück. Das Zeichen/der Eintrag an Index `start` ist dabei Bestandteil des Ergebnisses, das Zeichen/der Eintrag an Index `stop` nicht. Negative Werte für `start` und `stop` werden als dabei als relativ zum Ende des Strings bzw. der Liste interpretiert. Wird `start` weggelassen, beginnt der Abschnitt am Anfang, wird `stop` weggelassen, endet der Abschnitt am Ende.

Beispiele:

Ausdruck	Wert
`"gurk"[1:3]`	`"ur"`
`"gurk"[-3:3]`	`"ur"`
`"gurk"[:2]`	`"gu"`
`"gurk"[-2:]`	`"rk"`
`"gurk"[3:1]`	`""`
`"gurk"[10:]`	`""`
`"gurk"[:-10]`	`""`

26.6.21 `not x` (Boolsche Negation)

Gibt `False` zurück wenn `bool(x)` `True` ist, ansonsten `True`.

Beispiele:

Ausdruck	Wert
`not None`	`True`
`not False`	`True`
`not True`	`False`
`not 0`	`True`
`not 42`	`False`
`not ""`	`True`
`not "gurk"`	`False`
`not "0"`	`False`
`not days(0)`	`True`
`not days(7)`	`False`
`not months(0)`	`True`
`not months(12)`	`False`
`not [1, 2, 3]`	`False`

Siehe auch

bool().

26.6.22 `-x` (Unäres Minus)

Kehrt das Vorzeichen der Zahl (oder des Zeitraums) `x` um.

`BOOL`-Werte werden dabei wie die entsprechenden Ganzzahl-Werte 0 und 1 behandelt.

26.6.23 `x if cond else y` (Wenn/Dann-Ausdruck)

Gibt `x` zurück wenn `bool(cond)` `True` ist, ansonsten `y`.

Dabei müssen die Datentypen von `x` und `y` kompatibel sein. (`"gurk" if True else 42` wird beispielsweise nicht unterstützt.)

Siehe auch

bool().

26.6.24 `x.attrname` (Attribut-Zugriff)

Die Attribute der jeweiligen Datentypen werden in *Datentypen, Attribute und Methoden* beschrieben.

26.6.25 `funktion(...)` (Funktionsaufruf)

Die zur Verfügung stehenden Funktionen werden in *Funktionen* beschrieben.

26.6.26 `x.methode(...)` (Methodenaufruf)

Die methoden der jeweiligen Datentypen werden in *Datentypen, Attribute und Methoden* beschrieben.

26.7 Funktionen

26.7.1 `today()`

Gibt das aktuellen Datum als *DATE*-Wert zurück.

26.7.2 `now()`

Gibt den aktuellen Zeitpunkt als *DATETIME*-Wert zurück.

26.7.3 `bool()`

Gibt `False` zurück.

26.7.4 `bool(obj)`

Konvertiert `obj` in einen Boolschen Wert. Das Ergebnis ist `False` für `None`, `False`, die Zahl 0, leere Strings und Listen sowie für leere Zeiträume und `True` für alle anderen Werte.

Beispiele:

Ausdruck	Wert
`bool(None)`	`False`
`bool(False)`	`False`
`bool(True)`	`True`
`bool(0)`	`False`
`bool(42)`	`True`
`bool("")`	`False`
`bool("gurk")`	`True`
`bool("0")`	`True`
`bool(days(0))`	`False`
`bool(days(7))`	`True`
`bool(months(0))`	`False`
`bool(months(12))`	`True`
`bool([1, 2, 3])`	`True`

26.7.5 `int()`

Gibt `0` zurück.

26.7.6 `int(obj)`

Konvertiert `obj` in einen `INT`-Wert.

Beispiele:

Ausdruck	Wert	Bemerkung
`int(False)`	`0`	
`int(True)`	`1`	
`int(42)`	`42`	
`int(42.5)`	`42`	
`int(-42.5)`	`-42`	
`int("42")`	`42`	
`int("42.5")`	`None`	„Fehler“
`int("gurk")`	`None`	„Fehler“

26.7.7 `float()`

Gibt `0.0` zurück.

26.7.8 `float(obj)`

Konvertiert `obj` in einen `NUMBER`-Wert.

Beispiele:

Ausdruck	Wert	Bemerkung
`float(False)`	`0.0`	
`float(True)`	`1.0`	
`float(42)`	`42.0`	
`float(42.5)`	`42.5`	
`float(-42.5)`	`-42.5`	
`float("42")`	`42.0`	
`float("42.5")`	`42.5`	
`float("gurk")`	`None`	„Fehler“

26.7.9 str()

Gibt den leeren String ("") zurück.

26.7.10 str(obj)

Konvertiert `obj` in einen *STR*-Wert.

Beispiele:

Ausdruck	Wert
`str(None)`	`"None"`
`str(False)`	`"False"`
`str(True)`	`"True"`
`str(42)`	`"42"`
`str(42.5)`	`"42.5"`
`str("gurk")`	`None`
`str([1, 2, 3])`	`"[1, 2, 3]"`

26.7.11 date(jahr, monat, tag)

Erzeugt einen `DATE`-Wert aus dem übergebenen Jahr-, Monats- und Tagangaben.

Beispiele:

Ausdruck	Wert	Bemerkung
`date(2000, 2, 29)`	`@(2000-02-29)`	29. Feb. 2000

26.7.12 datetime(ja, mo, ta, st, mi, se)

Erzeugt einen `DATETIME`-Wert aus dem übergebenen Jahr-, Monats-, Tag-, Stunden-, Minuten- und Sekundenangaben.

Beispiele:

Ausdruck	Wert	Bemerkung
`datetime(2000, 2, 29)`	`@(2000-02-29T)`	29. Feb. 2000 00:00
`datetime(2000, 2, 29, 12)`	`@(2000-02-29T12:00)`	29. Feb. 2000 12:00
`datetime(2000, 2, 29, 12, 34)`	`@(2000-02-29T12:34)`	29. Feb. 2000 12:34
`datetime(2000, 2, 29, 12, 34, 56)`	`@(2000-02-29T12:34:56)`	29. Feb. 2000 12:34:56

26.7.13 len(str)

Gibt die Länge des Strings `str` zurück.

Beispiele:

Ausdruck	Wert
`len("")`	`0`
`len("gurk")`	`4`

26.7.14 len(list)

Gibt die Länge der Liste `list` zurück.

Beispiele:

Ausdruck	Wert
`len([1, 2, 3])`	`3`

26.7.15 timedelta()

Gibt einen „leeren“ Zeitraum zurück.

Beispiele:

Ausdruck	Wert
`@(2000-02-29) + timedelta()`	`@(2000-02-29)`

26.7.16 timedelta(tage)

Gibt einen `DATE`-Wert von `tage` Tagen zurück.

Beispiele:

Ausdruck	Wert
`@(2000-02-29) + timedelta(1)`	`@(2000-03-01)`
`@(2000-02-29) - timedelta(1)`	`@(2000-02-28)`

26.7.17 timedelta(tage, sekunden)

Gibt einen `DATETIME`-Wert von `tage` Tagen und `sekunden` Sekunden zurück.

Beispiele:

Ausdruck	Wert
`@(2000-02-29T12:34:56) + timedelta(1, 100)`	`@(2000-03-01T12:36:36)`

26.7.18 monthdelta()

Gibt einen *MONTHDELTA*-Wert für 0 Monate zurück.

Beispiele:

Ausdruck	Wert
`@(2000-02-29) + monthdelta()`	`@(2000-02-29)`

26.7.19 monthdelta(n)

Gibt einen *MONTHDELTA*-Wert von `n` Monaten zurück.

Beispiele:

Ausdruck	Wert
`@(2000-01-01) + monthdelta(1)`	`@(2000-02-01)`
`@(2000-01-31) + monthdelta(1)`	`@(2000-02-29)`

26.7.20 `years(n)`

Gibt einen *MONTHDELTA*-Wert von `n` Jahren zurück.

Beispiele:

Ausdruck	Wert
`@(2000-01-01) + years(1)`	`@(2001-01-01)`
`@(2000-02-29) + years(1)`	`@(2001-02-28)`

26.7.21 `months(n)`

Gibt einen *MONTHDELTA*-Wert von `n` Monaten zurück.

Beispiele:

Ausdruck	Wert
`@(2000-01-01) + monthdelta(1)`	`@(2000-02-01)`
`@(2000-01-31) + monthdelta(1)`	`@(2000-02-29)`

26.7.22 `days(n)`

Gibt einen *DATEDELTA*-Wert von `n` Tagen zurück.

Beispiele:

Ausdruck	Wert
`@(2000-02-29) + days(1)`	`@(2000-03-01)`
`@(2000-02-29) - days(1)`	`@(2000-02-28)`
`@(2000-01-31) + days(30)`	`@(2000-03-01)`
`@(2000-02-29) + days(89)`	`@(2000-05-28)`

26.7.23 `hours(n)`

Gibt einen *DATETIMEDELTA*-Wert von `n` Stunden zurück.

Beispiele:

Ausdruck	Wert
`@(2000-02-29T12:34:56) + hours(1)`	`@(2000-02-29T13:34:56)`

26.7.24 `minutes(n)`

Gibt einen *DATETIMEDELTA*-Wert von `n` Minuten zurück.

Beispiele:

Ausdruck	Wert
`@(2000-02-29T12:34:56) + minutes(1)`	`@(2000-02-29T13:35:56)`

26.7.25 seconds(n)

Gibt einen *DATETIMEDELTA*-Wert von `n` Sekunden zurück.

Beispiele:

Ausdruck	Wert
`@(2000-02-29T12:34:56) + seconds(1)`	`@(2000-02-29T13:34:57)`

26.7.26 md5(s)

Gibt die MD5-Prüfsumme für den String `s` zurück.

Beispiele:

Ausdruck	Wert
`md5("")`	`"d41d8cd98f00b204e9800998ecf8427e"`
`md5("gurk")`	`"4b5b6a3fa4af2541daa569277c7ff4c5"`

26.7.27 random()

Gibt eine zufällige Zahl zwischen `0.0` (einschließlich) und `1.0` (ausschließlich) zurück.

26.7.28 randrange(start, stop)

Gibt eine zufällige ganze Zahl zwischen `start` (einschließlich) und `stop` (ausschließlich) zurück.

26.7.29 cos(x)

Gibt den Cosinus von `x` (im Bogenmaß) zurück.

26.7.30 sin(x)

Gibt den Sinus von `x` (im Bogenmaß) zurück.

26.7.31 tan(x)

Gibt den Tangens von `x` (im Bogenmaß) zurück.

26.7.32 sqrt(x)

Gibt den Quadratwurzel von `x` zurück.

26.7.33 abs(x)

Gibt der Absolutbetrag von `x` zurück.

26.7.34 `geo(lat, long)`

Gibt einen Geo-Wert mit dem Längengrad `lat` und dem Breitengrad `long` zurück. Das `info`-Atribut ist `None`.

26.7.35 `geo(lat, long, info)`

Gibt einen Geo-Wert mit dem Längengrad `lat`, dem Breitengrad `long` und der Beschreibung `info` zurück.

26.7.36 `dist(x, y)`

Gibt den Abstand zwischen den beiden Geo-Koordinaten `x` und `y` zurück.

KAPITEL 27

LivingAPI

LivingAPI bezeichnet die Gesamtmenge an Objekten die einem Anzeige-Template, einem E-Mail-Template oder anderen Templates übergeben werden.

(Im Gegensatz dazu wird *vSQL* verwendet, um Datensätze zu filtern oder auszuwählen, *bevor* sie an Templates übergeben werden.)

Von welchen Apps Daten über die LivingAPI an ein Anzeige-, Formular-, Update- oder E-Mail-Template übergeben werden, kann unter *Konfiguration* → *Erweitert* in der *Anzeige-Templates*-Maske beim jeweiligen Anzeige-Template unter der *Datenquellen*-Maske konfiguriert werden (sowie über bestimmte Abfrage-Parameter im HTTP-Request beeinflußt werden). Für E-Mail-Templates passiert dies in der *E-Mail-Templates*-Maske beim jeweiligen E-Mail-Template unter der *Datenquellen*-Maske.

27.1 Objekt-Typen

Die folgende Objekt-Typen stellen die zentralen Objekte in der LivingAPI dar: Apps (*App*) mit ihren Feldern (*Control*) oder Dekorationsfeldern (*LayoutControl*) sowie Datensätze (*Record*) mit den Feld-Werten (*Field*), Anhängen zu den Datensätzen (*Attachment*) und untergeordnetet Datensätze (*RecordChildren*) sowie Auswahl-Objekte (*LookupItem*), Objekte für hochgeladene Dateien (*File*) und für geographische Koordinaten (*Geo*).

Weiterhin werden einem Anzeige-Template im *Globals*-Objekt globale Informationen übergeben. Darüber hat der Benutzer die Möglichkeit auf Informationen über die zu verarbeitende HTTP-Anfrage zuzugreifen (*HTTPRequest*), die zu versendende HTTP-Anwort zu beeinflußen (*HTTPResponse*), und einige Dinge mehr.

Bei einem E-Mail-Template werden ebenfalls im *Globals*-Objekt globale Informationen übergeben. Das *EmailRequest*-Objekt enhält Informationen über den Kontext in dem die E-Mail-Versendung stattfindet. Das *EmailResponse*-Objekt kann dazu verwendet werden, Betreff und Adressaten der Email zu ändern.

Dem *App*-Objekt sind weiterhin einige Informationen zugeordnet: Die App-Kategorien (*Category*), die App-Parameter (*AppParameter*), die Menüs (*MenuItem*), die Panels/Kacheln (*Panel*), die Installation, die diese App erzeugt hat (*Installation*) sowie die Ansichten (*View*).

Die einzelnen Objekt-Typen werden in den folgenden Unter-Kapiteln beschrieben:

27.1.1 `Globals`

Ein `Globals`-Objekt wird einem Anzeige-, Formular-, Update- oder E-Mail-Template in der Variablen `globals` übergeben.

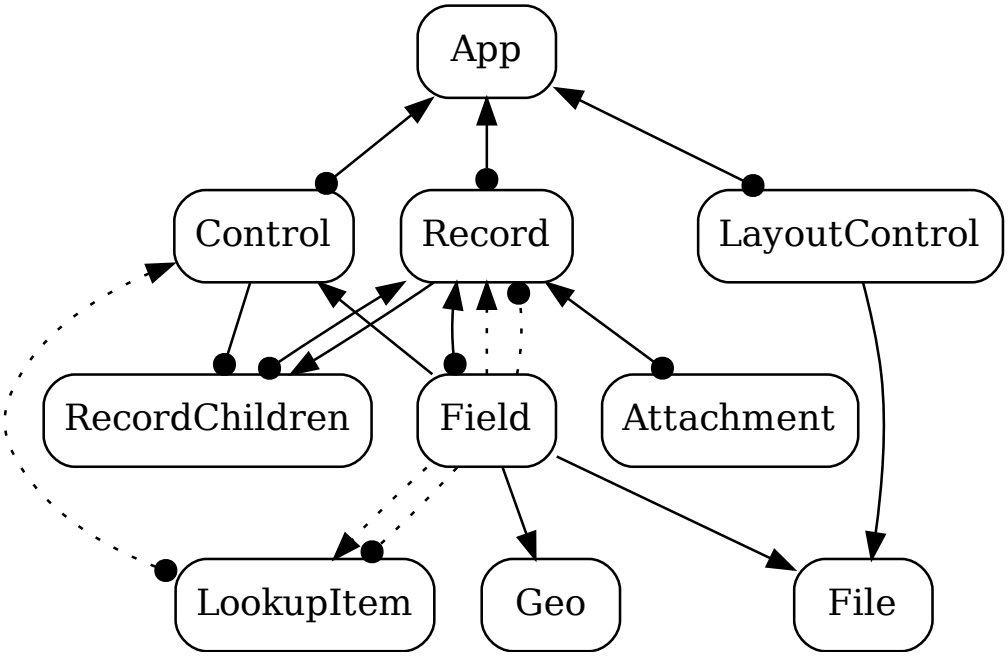

Abb. 1: Die zentralen Objekt-Typen der LivingAPI

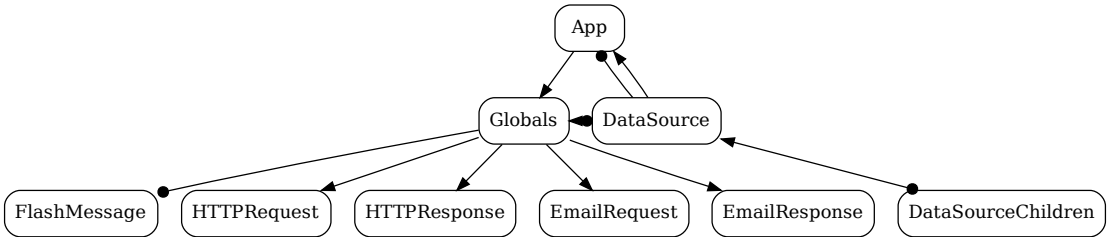

Abb. 2: Globale Informationen

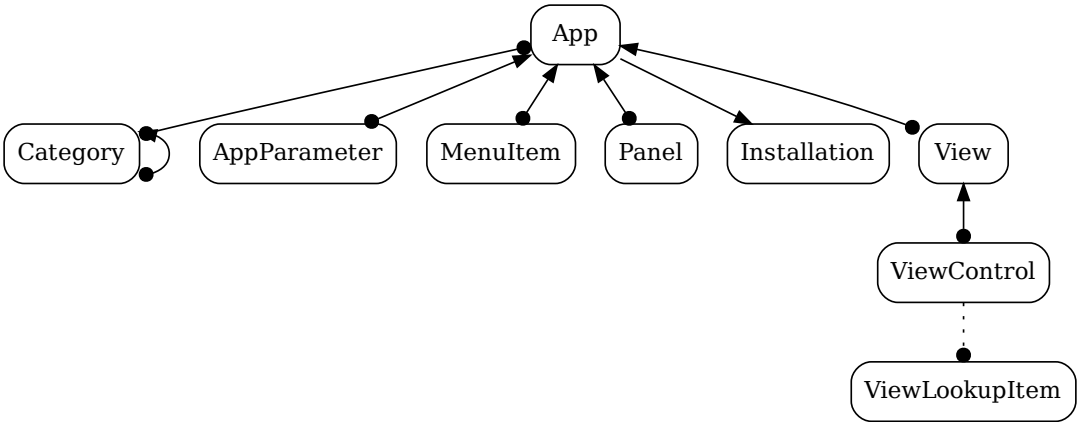

Abb. 3: Zusätzliche Information zu `Apps`

Diese Variable ist global, d.h. sie ist auch in aufgerufenen Unter-Templates bekannt, ohne daß `globals` beim Aufruf an diese übergeben werden muß.

Das `Globals`-Objekt enthält globale Informationen und besitzt folgende Attribute:

app
[*App*]
Die App des aufgerufenen Anzeigetemplates.

record
[*Record* oder `None`]
Der über die URL mitgegebene Datensatz.

version
[String]
Die Version der LivingAPI die zur Erzeugung sämtlicher Objekte verwendet wurde.

platform
[String]
Der Name für die installierte Version von LivingApps. Auf http://my.living-apps.de ist dies z.B. `"LivingApps"`, auf http://my.leadair.de `"LeadAir"`.

hostname
[String]
Der Hostname des LivingApps-Server (also beispielsweise `my.living-apps.de` oder `my.leadair.de`).

user
[*User* oder `None`]
Der aktuell eingeloggte Benutzer. `user` ist `None`, falls ein nicht authentifizierter Nutzer ein öffentliches Template nutzt.

lang
[String]
Die zu verwendende Sprache (d.h. in dieser Sprache sollten die Ausgaben in einem Anzeige-Template erzeugt werden). Wenn ein Benutzer eingeloggt ist, ist dies die Sprache des Benutzers (ansonsten Standard-Sprache der LivingApp-Platform: `"de"` für Deutsch). Dieses Attribut kann geändert werden (z.B. wenn wegen eines `Accept-Language`-Headers eine andere Sprache verwendet werden soll).

In Formular-Templates ist `globals.lang` die Sprache der Formularvariante (nicht die bevorzugte Sprache des eingeloggten Benuzters).

p_<identifier>
[*AppParameter*]
„Shortcut"-Attribute für die `AppParameter`-Objekte der App. Beispielsweise ist `globals.p_beispiel` äquivalent zu `globals.app.p_beispiel` (und dies wiederum zu `globals.app.params.beispiel`).

pv_<identifier>
[Objekt]
„Shortcut"-Attribute für die Werte von `AppParameter`-Objekten der App. `globals.pv_beispiel` äquivalent zu `globals.app.pv_beispiel` (und im Endeffekt wiederum zu `globals.app.params.beispiel.value`).

templates
[Dictionary(String → UL4-Template)]
Alle internen Templates, die in der aktuellen App definiert sind. Diese Templates können unter *Konfiguration → Erweitert* in der *Interne Templates*-Maske angelegt werden. Die Schlüssel des Dictionarys sind dabei der Identifizierer des Templates und die Werte die UL4-Templates selbst.

Damit ist es möglich, z.B. mehrfach benötigte Funktionalität in einem Template zu implementieren und diese Funktionalität von anderen Anzeige-Templates aus aufzurufen.

t_<identifier>
[UL4-Template]
Die internen Templates stehen auch über diese „Shortcut"-Attribute zur Verfügung. `globals.t_gurk` is beispielsweise äquivalent zu `globals.templates.gurk`.

datasources
[Dictionary(String → *DataSource*)]
Die Datenquelle des aufrufenden Templates

custom
[Objekt]
Dieses Attribut kann vom Benutzer für beliebige zusätzliche Informationen gesetzt werden.

x_<identifier>
[Objekt]
Es werden beliebige zusätzliche Attribute unterstützt deren Namen mit `x_` beginnt.

mode
[String]
Gibt an in welchem Typ von Template man sich gerade befindet bzw. in welcher Ablaufphase dieses Templates. Mögliche Werte sind:

form/new/init
[Formular- und Update-Template]
Der erste Aufruf eines „Neu anlegen"-Formulars.

form/new/search
[Formular-Template]
Ajax-Suche für `applookup/choice`- bzw. `multipleapplookup/choice`-Felder.

form/new/failed
[Formular-Template]
Das Ausfüllen des „Neu anlegen"-Formular ist fehlgeschlagen (wegen nicht ausgefüllter Pflichtfelder etc.).

form/new/presave
[Formular-Template]
Das „Neu anlegen"-Formular wurde erfolgreich ausgefüllt. Der Datensatz wurde noch nicht gespeichert.

form/new/postsave
[Formular-Template]
Das „Neu anlegen"-Formular wurde erfolgreich ausgefüllt. Der Datensatz wurde bereits gespeichert.

`form/new/input`
[Update-Template]
Der Benutzer hat im „Neu“-Formular ein Eingabefeld geändert. Die Variable `identifier` enthält den Identifier des geänderten Feldes.

`form/new/geo`
[Update-Template]
In einen „Neu“-Formular liegt Geo-Information vor über den Standort des Benutzers vor.

`form/edit/init`
[Formular- und Update-Template]
Der erste Aufruf eines „Bearbeiten“-Formulars.

`form/edit/search`
[Formular-Template]
Ajax-Suche für `applookup/choice`- bzw. `multipleapplookup/choice`-Felder.

`form/edit/failed`
[Formular-Template]
Das Ausfüllen des „Bearbeiten“-Formular ist fehlgeschlagen (wegen nicht ausgefüllten Pflichtfeldern etc.).

`form/edit/presave`
[Formular-Template]
Das „Bearbeiten“-Formular wurde erfolgreich ausgefüllt. Der Datensatz wurde noch nicht gespeichert.

`form/edit/postsave`
[Formular-Template]
Das „Bearbeiten“-Formular wurde erfolgreich ausgefüllt. Der Datensatz wurde bereits gespeichert.

`form/edit/input`
[Update-Template]
Der Benutzer hat im „Bearbeiten“-Formular ein Eingabefeld geändert. Die Variable `identifier` enthält den Identifier des geänderten Feldes.

`form/new/geo`
[Update-Template]
In einen „Bearbeiten“-Formular liegt Geo-Information vor über den Standort des Benutzers vor.

`view/list`
[Anzeige-Template]
In einem Anzeige-Template vom Typ „Liste“.

`view/detail`
[Anzeige-Template]
In einem Anzeige-Template vom Typ „Detail“.

`view/support`
[Anzeige-Template]
In einem Anzeige-Template vom Typ „Support“.

`email/text`
[E-Mail-Template]
Text-Version der E-Mail.

`email/html`
[E-Mail-Template]
HTML-Version der E-Mail.

`form`
[String]
Gibt an, ob das Formular in die LivingApps-Plattform eingebettet ist oder nicht. Kann die Werte `embedded` oder `standalone` annehmen.

geo(...)

[Methode(...) → *Geo*]
Dieser Methode gibt *Geo*-Objekte zurück. Sie besitzt zwei verschiedene Signaturen:

geo(lat, long)

[Methode(Zahl, Zahl) → *Geo*]
Übergeben wird der Breitengrad `lat` und der Längengrad `long`. Die fehlende Beschreibung des Ortes (z.B. eine Adresse) wird daraus ermittelt. Beispielsweise liefert:

```
<?printx globals.geo(lat=49.9552129, long=11.5901843)?>
```

das Geo-Objekt:

```
<com.livinglogic.appdd.Geo
   lat=49.9552129
   long=11.5901843
   info='Markgrafenallee 44, 95448 Bayreuth, Deutschland'
>
```

geo(info)

[Methode(String) → *Geo*]
Übergeben wird eine Beschreibung des Ortes z.B. als Adresse. Der fehlende Längen- und Breitengrad wird daraus ermittelt. Beispielsweise liefert:

```
<?code info = "LivingLogic AG, Markgrafenallee 44, 95448 Bayreuth"?>
<?printx globals.geo(info=info)?>
```

das Geo-Objekt:

```
<com.livinglogic.appdd.Geo
   lat=49.95559
   long=11.59011
   info='LivingLogic AG, Markgrafenallee 44, 95448 Bayreuth'
>
```

Bemerkung

Wenn *Geo*-Objekte dargestellt werden, muss Open Streetmap als Datenquelle genannt werden. Mehr Informationen zur Nennung[1] und zur Lizenz[2].

current_geo()

[Methode() → *Geo*]
Gibt den aktuellen Standort des Benutzers zurück (nur in Update-Templates).

seq()

[Methode() → Integer]
Jeder Aufruf dieser Methode gibt eine eindeutige, fortlaufende Nummer zurück. `seq` benötigt keine Argumente.

dist(geo1, geo2)

[Methode(*Geo*, *Geo*) → Zahl]
Gibt die Entfernung zwischen den zwei Geo-Koordinaten `geo1` und `geo2` in Kilometern zurück.

scaled_url(...)

[Methode(...) → String]
Diese Methode liefert eine URL unter der die skalierte Version eines Bildes ausgeliefert wird. Zum Skalieren der Bilder wird imgproxy[3] verwendet.

[1] https://www.openstreetmap.de/faq.html#lizenz
[2] https://opendatacommons.org/licenses/odbl/summary/
[3] https://imgproxy.net/

Die Methode unterstützt folgende Argumente:

image
[*File* oder String]
Entweder die absolute URL eines Bildes oder ein *File*-Objekt das ein Bild anthält. Dieses Bild wird skaliert.

type
[String oder `None`]
Erlaubte Werte sind: `"fit"`, `"fill"`, `"fill-down"`, `"force"` und `"auto"`. Der Standardwert ist `"fill"`.

Für mehr Information siehe die imgproxy-Dokumentation[4].

width
[Integer oder `None`]
Die Zielbreite des skalierten Bilder. Der Standardwert ist `None`.

Für mehr Information siehe die imgproxy-Dokumentation[5].

height
[Integer oder `None`]
Die Zielhöhe des skalierten Bildes. Der Standardwert ist `None`. `width` und `height` dürfen nicht beide `None` sein.

Für mehr Information siehe die imgproxy-Dokumentation[6].

enlarge
[Bool]
Soll das Bild vergrößert werden, wenn es kleiner als die angegebene Zielgröße ist? Der Standardwert ist `True`.

Für mehr Information siehe die imgproxy-Dokumentation[7].

gravity
[String oder `None`]
Erlabute Werte sich `"no"`, `"so"`, `"ea"`, `"we"`, `"noea"`, `"nowe"`, `"soea"`, `"sowe"`, `"ce"` und `"sm"`. Der Standardwert ist `"sm"`.

Für mehr Information siehe die imgproxy-Dokumentation[8].

quality
[Integer oder `None`]
Wird dieses Argument angegeben, muß der Wert zwischen 0 und 100 liegen. Der Standardwert ist `None`.

Für mehr Information siehe die imgproxy-Dokumentation[9].

rotate
[Integer oder `None`]
Rotiert das Bild. Wird dieses Argument angegeben, muß der Wert ein ganzzahliges Vielfaches von 90 sein.

Für mehr Information siehe die imgproxy-Dokumentation[10].

blur
[Zahl oder `None`]
Macht das Bild unschwarf.

[4] https://docs.imgproxy.net/generating_the_url?id=resizing-type
[5] https://docs.imgproxy.net/generating_the_url?id=width
[6] https://docs.imgproxy.net/generating_the_url?id=height
[7] https://docs.imgproxy.net/generating_the_url?id=enlarge
[8] https://docs.imgproxy.net/generating_the_url?id=gravity
[9] https://docs.imgproxy.net/generating_the_url?id=quality
[10] https://docs.imgproxy.net/generating_the_url?id=rotate

Für mehr Information siehe die imgproxy-Dokumentation[11].

sharpen
[Zahl oder `None`]
Schärft das Bild.

Für mehr Information siehe die imgproxy-Dokumentation[12].

format
[String oder `None`]
Das Bildformat (PNG, GIF, JPEG usw.) des skalierten Bildes.

Für mehr Information siehe die imgproxy-Dokumentation[13].

cache
[Bool]
Wird `True` gegeben, so wird eine URL zurückgeliefert, die das skalierte Bild zwischenspeichert, sodaß es bei einen erneuten Aufruf nicht nochmal skaliert werden muß. Bei `False` wird das Bild bei jeden Aufruf skaliert.

„Flash"-Meldungen werden dazu verwendet um dem Benutzer über das Ergebnis von Aktionen zu informieren, die er auf der vorhergehenden Seite durchgeführt hat. Z.B. kann es in einer Liste von Datensätzen zu jeden Datensatz einen Button geben, der über ein Formular/einen POST-Request diesen Datensatz ändert. Das UL4-Template das diese Änderung durchführt setzt dann eine Flash-Meldung und leitet den Benutzer wieder zur Listenansicht zurück. Die Listenansicht zeigt dann diese Flash-Meldung an. D.h. die Flash-Meldungen bleiben auch über Requests hinweg erhaltet.

flashes()
[Methode() → Liste(*FlashMessage*)]
`flashes` gibt eine Liste von „Flash"-Meldungen zurück. Beim Aufruf von `flashes()` wird diese Liste automatisch geleert (d.h. der nächste Aufruf gibt eine leere Liste zurück).

flash_info(title, message)
[Methode(String, String oder `None`) → `None`]
Erzeugt ein neues `FlashMessage`-Objekt von Typ `info` und fügt dieses zur Liste der „Flash"-Meldungen hinzu. Der Parameter `title` muß ein String sein, `message` ein String oder `None`.

flash_notice(title, message)
[Methode(String, String oder `None`) → `None`]
Erzeugt ein neues `FlashMessage`-Objekt von Typ `notice` und fügt dieses zur Liste der „Flash"-Meldungen hinzu. Der Parameter `title` muß ein String sein, `message` ein String oder `None`.

flash_warning(title, message)
[Methode(String, String oder `None`) → `None`]
Erzeugt ein neues `FlashMessage`-Objekt von Typ `warning` und fügt dieses zur Liste der „Flash"-Meldungen hinzu. Der Parameter `title` muß ein String sein, `message` ein String oder `None`.

flash_error(title, message)
[Methode(String, String oder `None`) → `None`]
Erzeugt ein neues `FlashMessage`-Objekt von Typ `error` und fügt dieses zur Liste der „Flash"-Meldungen hinzu. Der Parameter `title` muß ein String sein, `message` ein String oder `None`.

Die folgenden Methoden können dazu verwendet werden die Ausführung eines Templates zu protokollieren. Die erzeugten Meldungen sind beim jeweiligen Anzeige-Template unter *Meldungen in aktueller Version* zu finden.

Alle Methoden unterstützen eine beliebige Anzahl von Argumenten. Ein Argument, das nicht vom Typ String ist wird mittels der UL4-Funktion `repr` in einen String konvertiert.

log_debug(*args)
[Methode(**Objekt) → `None`]
Erzeugt eine neue Meldung vom Typ `debug`.

[11] https://docs.imgproxy.net/generating_the_url?id=blur
[12] https://docs.imgproxy.net/generating_the_url?id=sharpen
[13] https://docs.imgproxy.net/generating_the_url?id=format

log_info(*args)
[Methode(**Objekt) → None]
Erzeugt eine neue Meldung vom Typ `info`.

log_notice(*args)
[Methode(**Objekt) → None]
Erzeugt eine neue Meldung vom Typ `notice`.

log_warning(*args)
[Methode(**Objekt) → None]
Erzeugt eine neue Meldung vom Typ `warning`.

log_error(*args)
[Methode(**Objekt) → None]
Erzeugt eine neue Meldung vom Typ `error`.

In Anzeige-Templates haben die beiden Attribute `request` und `response` folgende Bedeutung:

request
[*HTTPRequest*]
Informationen über die HTTP-Anfrage über die das Anzeige-Template aufgerufen wurde.

response
[*HTTPResponse*]
Informationen über die HTTP-Antwort über die das Anzeige-Template ausgeliefert werden wird.

Innerhalb von Email-Templates exisistieren die beiden Attribute `request` und `response` ebenfalls, haben jedoch eine andere Bedeutung:

request
[None]
Wird für Email-Templates nicht benötigt und ist immer `None`

response
[*EmailResponse*]
Enthält den Betreff der zu versendenden E-Mail sowie die E-Mail-Adressaten. Diese Informationen können vom E-Mail-Template geändert werden.

Die folgenden Methoden geben die relativen URLs der aus der Web-Oberfläche bekannten Navigations-Links zurück.

my_apps_url()
[Methode() → String]
Gibt die relative URL zur „Meine Apps“-Seite zurück.

Beispiel:

```
<?print app.my_apps_url()?>
```

erzeugt:

```
https://my.living-apps.de/apps.htm
```

my_tasks_url()
[Methode() → String]
Gibt die relative URL zur „Aufgaben“-Seite zurück.

Beispiel:

```
<?print app.my_tasks_url()?>
```

erzeugt:

```
https://my.living-apps.de/xist4c/web/aufgaben_id_393_.htm
```

catalog_url()
[Methode() → String]
Gibt die relative URL zur „Katalog“-Seite zurück.

Beispiel:

```
<?print app.catalog_url()?>
```

erzeugt:

```
https://my.living-apps.de/katalog/home.htm
```

chats_url()
[Methode() → String]
Gibt die relative URL zur „Chats“-Seite zurück.

Beispiel:

```
<?print app.chats_url()?>
```

erzeugt:

```
https://my.living-apps.de/chats.htm
```

profile_url()
[Methode() → String]
Gibt die relative URL zur „Profil“-Seite zurück.

Beispiel:

```
<?print app.profile_url()?>
```

erzeugt:

```
https://my.living-apps.de/profil/index.htm
```

account_url()
[Methode() → String]
Gibt die relative URL zur „Account“-Seite zurück.

Beispiel:

```
<?print app.account_url()?>
```

erzeugt:

```
https://my.living-apps.de/account.htm
```

logout_url()
[Methode() → String]
Gibt die relative URL des Logout-Links zurück.

Beispiel:

```
<?print app.logout_url()?>
```

erzeugt:

```
https://my.living-apps.de/login.htm?logout=standardCug
```

27.1.2 HTTPRequest

Ein `HTTPRequest`-Objekt beinhaltet Informationen über die HTTP-Anfrage über die das Anzeige-Template aufgerufen wurde und ist zugänglich als `globals.request`. Es besitzt folgende Attribute:

id
[String]
Eine eindeutige ID zur Kennzeichnung des Requests.

method
[String]
Die HTTP-Methode über die das Anzeige-Template aufgerufen wurde. Dies kann entweder `"get"` oder `"post"` sein.

url
[String]
Die vollständige Request-URL (mit Query-String).

headers
[Dictionary(String → String)]
Die HTTP-Header, die bei der HTTP-Anfrage übergeben wurden. Die Dictionary-Schlüssel sind die Header-Namen (Groß-/Kleinschreibung wird dabei ignoriert) und die Werte sind die Werte der Header. Beispielsweise gibt

```
<?print globals.request.headers["Accept"]?>
```

folgendes aus:

```
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
```

> **Bemerkung**
> Cookies werden speziell behandelt. Der Mechanismus, für das Setzen von Cookies wird unter *HTTPResponse::headers* beschrieben. Falls per JavaScript Cookies gesetzt wurden, so sind nur diejenigen vorhanden, deren Namen mit `custom` beginnt. Alle anderen Cookies werden herausgefiltert.

cookies
[Dictionary(String → String)]
Die Cookies, die bei der HTTP-Anfrage übergeben wurden. Es sind lediglich diejenigen Cookies enthalten, die auch von dieser App gesetzt wurden. Beispielsweise gibt

```
<?print globals.request.cookies["my-cookie"]?>
```

den Wert des im letzten Response gesetzten Cookies `my-cookie` aus.

params
[Dictionary(String → (String oder *File*))]
Die Parameter, die bei der HTTP-Anfrage übergeben wurden. Beispielsweise gibt bei folgender URL

```
http://my.living-apps.de/gateway/apps/1234567890abcdef1234567890abcd/
↪fedcba0987654321fedcba09876543?template=gurk&modus=normal
```

der Ausdruck `<?print globals.request.params.modus?>` den Wert `normal` aus.

Wenn über ein Formular mit dem `enctype` `multipart/form-data` eine Datei hochgeladen wurde, so ist der Wert des entsprechenden Parameters ein *File*-Objekt. Diese Objekt kann dann z.B. direkt benutzt werden um einen Datensatz anzulegen:

```
<?code personapp.insert(
   firstname=globals.request.params.firstname,
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
   lastname=globals.request.params.lastname,
   portrait=globals.request.params.portrait,
)?>
```

dabei wurde dann z.B. folgendes Formular verwendet:

```
<form enctype="multipart/form-data" method="post" action="/gateway/apps/
↪1234567890abcdef1234567890abcd">
   <input type="hidden" name="template" value="gurk">
   <label>First name <input type="text" name="firstname" /></label>
   <label>Last name <input type="text" name="lastname" /></label>
   <label>Portrait <input type="file" name="portrait" /></label>
</form>
```

seq

[Methode() → Number]
Gibt einen fortlaufenden Integer-Wert zurück, der innerhalb des Requests eindeutig ist.

Dies kann z.B. dazu verwendet werden um für HTML-Seiten-Elemente eindeutige IDs zu erzeugen.

27.1.3 HTTPResponse

Ein `HTTPResponse`-Objekt enhält Informationen über die HTTP-Antwort über die das Anzeige-Template ausgeliefert werden wird und ist zugänglich als `globals.response`. Das Anzeige-Template kann die Attribute diese Objekts ändern, um die Antwort zu beinflussen. Die Attribute sind:

status

[Integer]
Der HTTP-Statuscode

headers

[Dictionary(String → String)]
Die HTTP-Header, die bei der Antwort geschickt werden (Groß-/Kleinschreibung wird dabei bei den Schlüsseln ignoriert). `Set-Cookie` Header werden nur in das `HTTPResponse` Objekt geschrieben, wenn eine gültige App-ID vorliegt. In diesem Fall wird der Cookie-Name mit dem Präfix `la-{appId}-` ergänzt (`{appId}` stellt dabei die ID der aktuellen App dar. Beispiel: `la-661519498060b8918021d884-`). Dieser Präfix wird beim nächsten Request wieder entfernt, bevor das Cookie an die `LivingAPI` weitergereicht wird. Mit diesem Mechanismus wird sichergestellt, dass Cookies von einer App nur an diese durchgereicht werden und Inhalte nicht von anderen Apps ausgelesen werden können.

send_file(file)

[Methode(*File*) → `None`]
Mit dieser Methode können Sie die LivingAPI anweisen, statt der Ausgabe, die Ihr Anzeige-Template erzeugt, ein *File*-Objekt auszuliefern.

Wenn Sie unter *Uploads schützen?* konfiguriert haben, daß nur Benutzer mit Rechten an der App Dateien herunterladen können, können Sie in einem Anzeige-Template Ihren eigenen Zugriffsschutz implementieren und dann mittels `send_file()` die gewünschte Datei ausliefern.

`send_file()` kann mehrmal aufgerufen werden. Das Argument muß ein *File*-Objekt sein. Alle übergebenen *File*-Objekte werden in einer Liste gesammelt. Ist am Ende des Anzeige-Templates diese Liste leer, so wird die Ausgabe des Anzeige-Templates ausgeliefert. Enthält die Liste ein *File*-Objekt, so wird dieses File ausgeliefert. Enthält die Liste mehrere Files, so werden diese Files in ein ZIP-Archiv verpackt, und dieses ZIP-Archiv wird ausgeliefert.

Werden mittels `send_file()` Dateien ausgeliefert, so werden trotzdem die in `headers` gesetzten Header berücksichtigt, mit Ausnahme des `"Content-Type"`-Headers, der automatisch auf den richtigen Datei-Typ gesetzt wird und auch nicht verändert werden kann. Außerdem wird der `"Content-Disposition"`-Header gesetzt um dem Browser den Original-Dateinamen bekannt zu machen.

clear_files()

[Methode() → None]
Leert die Liste der zu auszuliefernden Files, d.h. danach wird wieder die Ausgabe des Anzeige-Templates ausgeliefert.

add_cookie(name, value, expires, max_age, secure, http_only, path, same_site)

[Methode(String, String, Datum, Integer, Bool, Bool, String, String) → None]
Fügt einen `Set-Cookie` Header in die Response hinzu. `name` und `value` müssen angegeben werden, für die restlichen Parameter gibt es Default-Werte. `expires` (default `None`) ist der Zeitstempel wann das Cookie abläuft. `max_age` (default `None`) gibt die Lebensdauer des Cookies in Sekunden an. `secure` (default `True`) sorgt dafür, dass das Cookie nur in verschlüsselten https-Requests vom Browser an den Server geschickt wird. `http_only` (default `False`) steuert, ob ein Cookie per JavaScript abgerufen werden kann. `path` (default `None`) gibt einen URL-Pfad an, der in der angeforderten URL existieren muss, um das Cookie zu senden. `same_site` (default `Strict`) steuert, ob/wann Cookies mit Cross-Site-Anfragen gesendet werden. Mögliche Werte sind `Strict`, `Lax` und `None`.

Beispiele

Ein Redirect läßt sich mit `status` und `headers` beispielsweise folgendermaßen implementieren:

```
<?code globals.response.status = 303?>
<?code globals.response.headers["Location"] = "http://www.example.org/"?>
```

Sie können mit folgendem Code Ihr eigenes Account-Icon ausliefern:

```
<?code globals.response.send_file(globals.user.image)?>
```

27.1.4 EmailRequest

In einem E-Mail-Template enhält das `EmailRequest`-Objekt (zugänglich als `globals.request`) Informationen über den Kontext in dem die E-Mail-Versendung stattfindet.

bodytext

[String oder None]
Wenn das E-Mail-Template in einer Datenaktion verwendet wird, so ist `globals.request.bodytext` der Wert der durch das Feld *Inhalt (Text)* in der Email-Versendungs-Anweisung definiert wurde.

bodyhtml

[String oder None]
Wenn das E-Mail-Template in einer Datenaktion verwendet wird, so ist `globals.request.bodyhtml` der Wert der durch das Feld *Inhalt (HTML)* in der Email-Versendungs-Anweisung definiert wurde.

Wenn das E-Mail-Template verwendet wird, um im Datenmanagement eine E-Mail zu versenden, so ist `globals.request.bodyhtml` die im *E-Mail versenden*-Dialog eingegebene E-Mail-*Nachricht*.

27.1.5 EmailResponse

In einem E-Mail-Template enhält das `EmailResponse`-Objekt (zugänglich als `globals.response`) Informationen über die zu versendende E-Mail. Diese Informationen können vom E-Mail-Template geändert werden, um die zu versendende E-Mail zu beinflussen. Die Attribute sind:

from

[String]
Der Name der in der Absender-Adresse auftaucht (die E-Mail-Adresse wird aber immer ein LivingApps-E-Mail-Adresse sein).

to

[String oder None]
Der Empfänger der Email. Dies kann eine einfache E-Mail-Adresse sein (z.B. `person@example.org`) oder eine Kombination aus Name der Person und E-Mail-Adresse (in der Form `Vorname Nachame <person@example.org>`). Außerdem können mehrere E-Mail-Adressen mit Komma getrennt verwendet werden.

Es ist weiterhin möglich das Attribut `to` auf `None` zu setzen. Damit wird die Versendung der E-Mail komplett unterdrückt.

`cc`
[String oder `None`]
E-Mail-Adressen die die E-Mail als „Kopie“ (d.h. zur Kenntnisnahme) zugesendet bekommen sollen. Es sind die selben Werte wie beim Attribut `to` erlaubt. `None` bedeutet jedoch lediglich, daß niemand diese E-Mail als Kopie erhält, nicht daß sie nicht versendet wird.

`bcc`
[String oder `None`]
E-Mail-Adressen die die E-Mail als „Blindkopie“ zugesendet bekommen sollen (d.h. zur Kenntnisnahme, jedoch ohne daß andere Empfänger dies sehen können). Es sind die selben Werte wie beim Attribut `cc` erlaubt.

`replyto`
[String oder `None`]
Eine E-Mail-Adresse an die eine Antwort-E-Mail adressiert werden soll, wenn der Empfänger der E-Mail in seinem E-Mail-Programm die E-Mail beantwortet. Der Standardwert ist hier `None`, was bedeuten würde, daß die Antwort-E-Mail an den automatischen LivingApps-E-Mail-Account gesendet würde. Daher empfielt es sich immer das Attribut `replyto` zu setzen.

`subject`
[String]
Der Betreff der E-Mail.

`attachments`
[Liste(*File*)]
Die Attachments, zu dieser E-Mail. Wenn in der Konfiguration ein Attachment konfiguriert ist, so enthält diese Liste dieses Attachment, ansonsten ist die Liste leer. Das E-Mail-Template kann diese Liste auch ändern. Mehrere Attachments in der Liste werden ebenso unterstützt.

Alle diese Attribute sind durch die Konfiguration des E-Mail-Templates vorinitialisiert können jedoch durch das E-Mail-Template geändert werden.

27.1.6 `FlashMessage`

Ein `FlashMessage`-Objekt stellt eine Meldung dar, die dem Benutzer „bei nächster Gelegenheit“ angezeigt werden soll. Damit kann z.B. ein Anzeige-Template das einen Datensatz ändert eine Meldung hinterlassen und den Benutzer zu einer Übersichtsseite weiterleiten. Die Übersichtsseite kann dann die `FlashMessage` anzeigen, um den Benutzer über das Ergebnis seiner letzten Aktion zu informieren.

Ein `FlashMessage`-Objekt hat folgende Attribute:

`timestamp`
[Datum]
Der Zeitpunkt zu dem das `FlashMessage`-Objekt angelegt wurde (und damit der Zeitpunkt zu dem die durch diese Meldung beschriebene Aktion stattgefunden hat).

`type`
[String]
Der Meldungstyp. Mögliche Werte sind:

`info`
Diese Meldung beschreibt eine erfolgreich durchgeführte Aktion.

`notice`
Diese Meldung beschreibt eine erfolgreich durchgeführte Aktion, bei der aber Seiteneffekte aufgetreten sind, die dem Benutzer zur Kenntnis gebracht werden sollen. Es sind aber keine weiteren Aktionen des Benutzer erforderlich.

`warning`
Diese Meldung beschreibt eine erfolgreich durchgeführte Aktion, bei der unvorhergesehene Umstände auftreten sind, die möglicherweise das Eingreifen des Benutzers erfordern.

`error`
Diese Meldung weist den Benutzer darauf hin, das die von ihm durchgeführte Aktion fehlgeschlagen ist.

`title`
[String]
Eine Überschrift für die Meldung. Dieser String wird als HTML interpretiert, d.h. er darf HTML-Elemente enthalten (wie z.B. <b> oder etc.) Es sollten nur Inline-Element verwendet werden.

`message`
[String oder `None`]
Der Meldungstext. Auch dieser String wird als HTML interpretiert, und sollte nur Inline-Element beinhalten.

27.1.7 DataSource

Ein `DataSource`-Objekt bildet eine vom Benutzer konfigurierte Datenquelle ab. Eine solche Datenquelle kann unter *Konfiguration* → *Erweitert* in der *Anzeige-Templates*-Maske für ein Anzeige-Template angelegt werden. Darüber können Sie in der *Dokumentation zur Datenquelle* mehr lesen. Ein `DataSource`-Objekt besitzt folgende Attribute:

`id`
[String]
Der interne Datenbank-Identifizierer für diese Datenquelle.

`identifier`
[String]
Der vom Benutzer vergebene Identifizierer für diese Datenquelle.

`app`
[*App* oder `None`]
Die App die für diese Datenquelle konfiguriert wurde, oder `None` wenn diese Datenquelle für alle Apps konfiguriert wurde (d.h. wenn in der Konfiguration der Datenquelle bei *App* keine App ausgewählt wurde).

`apps`
[Dictionary(String → *App*)]
Alle Apps die zu dieser Datenquelle gehören. Schlüssel sind die Datenbank-Identifizierer der Apps und Werte die *App*-Objekte.

Ist in der Konfiguration der Datenquelle bei *App* eine App ausgewählt (und *Kopien einbeziehen?* nicht gesetzt), so beinhaltet `apps` nur diese eine App (dies kann natürlich auch die App sein, zu der die Datenquelle selber gehört). Ist *Kopien einbeziehen?* gesetzt, so beinhaltet `apps` dieses Basis-App und alle ihre Kopien und wenn keine App bei *App* ausgewählt wurde, beinhaltet `apps` *alle* Apps auf die der Benutzer Zugriff hat.

`children`
[Dictionary(String → *DataSourceChildren*)]
Informationen über die Konfigurationen von untergeordneten Datensätzen. Der Dictionary-Schlüssel ist dabei der *Identifizierer*, der dieser Zuordnung in der Konfiguration gegeben wurde. Die Werte sind *DataSourceChildren*-Objekte.

27.1.8 DataSourceChildren

Ein `DataSourceChildren`-Objekt enthält die Konfiguration für untergeordnete Datensätze aus einen *DataSource*-Objeke und hat folgende Attribute:

`id`
[String]
Der interne Datenbank-Identifizierer für dieses `DataSourceChildren`-Objekt.

`datasource`
[*DataSource*]
Das *DataSource*-Objekt zu der dieses `DataSourceChildren`-Objekt gehört.

identifier
[String]
Der Identifizierer, der für diese `DataSourceChildren`-Objekt in der Konfiguration vergeben wurde.

control
[*Control*]
Das `Control`-Objekt in der untergeordneten LivingApp über das die Zuordnung zur übergeordneten LivingApp hergestellt wird.

27.1.9 App

Ein `App`-Objekt stellt eine Applikation dar und hat folgende Attribute:

id
[String]
Der eindeutige Datenbank-Identifizierer der App.

globals
[*Globals*]
Verweist auf das `Globals`-Objekt, das globale Informationen enthält.

name
[String]
Der Name der App.

Dieses Attribut kann geändert werden.

description
[String oder `None`]
Die Beschreibung der App.

Dieses Attribut kann geändert werden.

lang
[String]
Die Sprache in der die App angezeigt wird (beispielsweise `"de"` für Deutsch und `"en"` für Englisch).

typename_nom_sin
[String]
Die Bezeichnung für Datensätze in dieser LivingApp (im Nominativ Singular).

Sind in Ihrer LivingApps z.B. Kunden gespeichert, so wäre dies sinnvollerweise „Kunde“.

Dieses Attribut kann geändert werden.

typename_gen_sin
[String]
Die Bezeichnung für Datensätze in dieser LivingApp (im Genitiv Singular).

Dieses Attribut kann geändert werden.

typename_dat_sin
[String]
Die Bezeichnung für Datensätze in dieser LivingApp (im Dativ Singular).

Dieses Attribut kann geändert werden.

typename_acc_sin
[String]
Die Bezeichnung für Datensätze in dieser LivingApp (im Akkusativ Singular).

Dieses Attribut kann geändert werden.

typename_nom_plu
[String]
Die Bezeichnung für Datensätze in dieser LivingApp (im Nominativ Plural).

Sind in Ihrer LivingApps z.B. Kunden gespeichert, so wäre dies sinnvollerweise „Kunden“.

Dieses Attribut kann geändert werden.

typename_gen_plu
[String]
Die Bezeichnung für Datensätze in dieser LivingApp (im Genitiv Plural).

Dieses Attribut kann geändert werden.

typename_dat_plu
[String]
Die Bezeichnung für Datensätze in dieser LivingApp (im Dativ Plural).

Dieses Attribut kann geändert werden.

typename_acc_plu
[String]
Die Bezeichnung für Datensätze in dieser LivingApp (im Akkusativ Plural).

Dieses Attribut kann geändert werden.

gramgen
[String]
Das grammatikalische Gechlecht der Bezeichnung für Datensätze in dieser LivingApp. `"m"` für männlich, `"f"` für weiblich, `"n"` für sählich.

Dies (und die obigen Attribute `typename_nom_sin` usw.) können dazu verwendet werden um korrekte Meldungen für Ihre Datensätze in den Anzeige-Templates zu erzeugen.

Dieses Attribut kann geändert werden.

group
[*AppGroup* oder `None`]
Die *AppGroup* zu der diese App gehört.

main
[Bool]
Ein Flag, das kennzeichnet, ob dies die Haupt-App innerhalb ihrer *AppGroup* ist.

image
[*File* oder `None`]
Das App-Icon in Original-Größe.

Wird das Icon in einer anderen Größe benötigt, kann *scaled_url(...)* benutzt werden.

Dieses Attribut kann geändert werden.

private_uploads
[Bool]
Gibt an, ob auf hochgeladene Dateien dieser App nur von autorisierten Benutzern zugegriffen werden darf.

createdby
[*User*]
Der Besitzer der App.

createdat
[Datum]
Der Zeitpunkt zu dem die App erstellt wurde.

updatedby
[*User* oder `None`]
Der Benutzer, der die App zuletzt geändert hat. (Wurde die App noch nicht geändert, so ist `updatedby` `None`.)

updatedat
[Datum oder `None`]
Der Zeitpunkt, zu dem die App das letzte Mal geändert wurde. (Wurde die App noch nicht geändert, so ist `updatedat` `None`.)

controls

[Dictionary(String → *Control*)]
Die Definition der Felder der App. Dieses Dictionary ist sortiert, d.h. beim Durchlauf werden die Felder in der Reihenfolge durchlaufen in der sie angelegt wurden. Die Schlüssel in diesem Dictionary sind die Feld-Identifizierer und die Werte sind `Control`-Objekte.

Normalerweise beinhaltet `controls` alle Felder des Datensatzes, außer wenn in der Datenquellen-Konfiguration *Nur Listenfelder?* gesetzt ist, dann werden nur diejenigen Felder übernommen, die unter *Konfiguration → Liste* ausgewählt wurden.

c_<identifier>

[*Control*]
`Control`-Objekte stehen auch über sogenannte „Shortcut"-Attribute zur Verfügung. D.h. dass beispielsweise das Feld `vorname` das normalerweise unter `app.controls.vorname` zu finden ist, auch direkt als `app.c_vorname` zur Verfügung steht.

records

[Dictionary(String → *Record*) oder `None`]
Die Datensätze der App. Dies sind evtl. nicht alle Datensätze, wenn der Benutzer nur eingeschränkte Zugriffrechte besitzt. Dieses Dictionary ist sortiert, die Reihenfolge kann in der Konfiguration der Datenquellen konfiguriert werden. Die Schlüssel des Dictionarys sind die Datensatz-Identifizierer und die Werte sind `Record`-Objekte.

Wenn in der Datenquellen-Konfiguration *Nur Anzahl?* gesetzt ist, ist `records` `None`.

recordcount

[Integer oder `None`]
Wenn in der Datenquellen-Konfiguration *Nur Anzahl?* gesetzt ist, beinhaltet `recordcount` die Anzahl der Datensätze (ansonsten ist `recordcount` `None`).

installation

[*Installation* oder `None`]
Wenn die Aplikation durch einen Installationsvorgang erzeugt wurde, ist `installation` ein `Installation`-Objekt. Ansonsten ist `installation` `None`.

templates

[Dictionary(String → UL4-Template)]
Alle internen Templates, die in der App definiert sind. Diese Templates können unter *Konfiguration → Erweitert* in der *Interne Templates*-Maske angelegt werden. Die Schlüssel des Dictionarys sind dabei der Identifizierer des Templates und die Werte die UL4-Templates selbst.

Haben Sie als *Datenquelle* nicht die aktuelle, sondern eine andere App ausgewählt, haben Sie hiermit die Möglichkeit, auf die internen Templates dieser in der Datenquelle konfigurierten App mit zuzugreifen.

t_<identifier>

[UL4-Template]
`Template`-Objekte stehen auch über sogenannte „Shortcut"-Attribute zur Verfügung. D. h., dass beispielsweise das Template `gurk`, das normalerweise unter `app.templates.gurk` zu finden ist, auch direkt als `app.t_gurk` zur Verfügung steht. Oder als `datasources.beispiel.app.t_gurk` wenn Sie als Datenquelle eine andere App ausgewählt haben.

viewtemplates

[Dictionary(String → *ViewTemplateInfo*) oder `None`]
Die Anzeige-Templates dieser App. Die Schlüssel des Dictionarys sind die `identifier` der Anzeige-Templates.

categories

[Dictionary(String → *Category*) oder `None`]
Wenn in der Datenquellen-Konfiguration *Kategorien* etwas ausgewählt ist, enthält `categories` die Kategorien, denen diese App zugeordnet ist. Die Schlüssel des Dictionarys sind die internen Datenbank-Identifizierer der Kategorie und die Werte sind `Category`-Objekte.

Ist bei *Kategorien* nichts ausgewählt, ist `categories` `None`.

params

[Dictionary(String → *AppParameter*)]
Die Parameter der Applikation (die unter *Konfiguration* → *Erweitert* in der *Parameter*-Maske angelegt werden können). Die Schlüssel des Dictionarys sind die Identifizierer der Parameter. Die Werte sind `AppParameter`-Objekte.

p_<identifier>

[*AppParameter*]
`AppParameter`-Objekte stehen auch über sogenannte „Shortcut"-Attribute zur Verfügung. D. h., dass beispielsweise der App-Parameter `beispiel`, der normalerweise unter `app.params.beispiel` zu finden ist, auch direkt als `app.p_beispiel` zur Verfügung steht.

pv_<identifier>

[Objekt]
Damit kann direkt auf den Wert eines``AppParameter``-Objekts zugegriffen werden. D. h. `app.pv_beispiel` ist äquivalent zu `app.p_beispiel.value` (was wiederum zu `app.params.beispiel.value` äquivalent ist).

views

[Dictionary(String → *View*)]
Die verschiedenen Formularvarianten der Applikation (die unter *Konfiguration* → *Eingabe* bei *Formularvarianten* angelegt werden können). Die Schlüssel des Dictionarys sind die Identifizierer der Views. Die Werte sind `View`-Objekte.

active_view

[*View*]
Die aktive Formularvariante. Ist `active_view` gesetzt (was durch Ändern dieses Attributs im Template getan werden kann, bzw. bei Formular-Templates automatisch passiert), werden beim Setzen von Feld-Werten die in dieser Variante definierten Feld-Restriktionen berücksichtigt (`minlength` und `maxlength` für String-Felder, `required` für alle).

Wird das Attribut `active_view` gesetzt kann als Wert sowohl ein *View*-Objekt verwendet werden, als auch die `id` eines *View*-Objekts. Dieses *View*-Objekt muß zu dieser App gehören.

datasource

[*DataSource*]
Das *DataSource*-Objekt das diese App beinhaltet, oder `None`, wenn es für diese App kein *DataSource*-Objekt gibt. D.h. es gibt für jede App `a` immer `a.datasource is None or `a.datasource.app is a`.

layout_controls

[Dictionary(String → *LayoutControl*) oder `None`]
Die Layout-Controls des aktiven Views. Dazu muß es einen aktiven *View* geben und in der Datenquelle muß bei *Felder Alle Felder und Layout-Felder* ausgewählt sein. Ansonsten ist `layout_controls` `None`.

lc_<identifier>

[*LayoutControl*]
„Shortcut"-Attribut zum Zugriff auf die Layout-Controls des aktiven Views. `app.lc_beispiel` ist äquivalent zu `app.layout_controls.beispiel`.

favorite

[Bool]
Gibt an, ob der aktuelle eingeloggte Benutzer diese App als Favorit festgelegt hat.

Dieses Attribut kann geändert werden.

menus

[Dictionary(String → *MenuItem*) oder `None`]
Die zu dieser App konfigurierten Menüs. Die Schlüssel in diesem Dictionary sind die Identifizierer und die Werte sind `MenuItem`-Objekte.

panels

[Dictionary(String → *Panel*) oder `None`]

Die zu dieser App konfigurierten Panels. Die Schlüssel in diesem Dictionary sind die Identifizierer und die Werte sind `Panel`-Objekte.

`save()`

[Methode() → Bool]
Speichert die Werte der veränderbaren Attribute in der Datenbank.

Veränderbare Attribute sind``name``, `description`, `image`, `favorite`, `typename_nom_sin`, `typename_gen_sin`, `typename_dat_sin`, `typename_acc_sin`, `typename_nom_plu`, `typename_gen_plu`, `typename_dat_plu` und `typename_acc_plu`.

`insert(**values)`

[Methode(**Objekt) → *Record*]
Mithilfe von `insert()` kann in der App ein neuer Datensatz angelegt werden. Die Parameter müssen per Schlüsselwort übergeben werden. Der Parametername ist dabei jeweils der Feld-Identifizierer des zu setzenden Feldes. Beispielsweise kann in eine Personen-App mit den Feldern `vorname`, `nachname` und `geburtstag` mittels folgendem Aufruf ein Datensatz eingefügt werden:

```
<?code record = app.insert(
   vorname="Max",
   nachname="Mustermann",
   geburtsdatum=@(2000-02-29),
)?>
```

Der zurückgegebene Datensatz wird mit den übergebenen Feldwerten initialisiert und das Attribut `id` (der eindeutige Datensatz-Identifizierer) ist gesetzt. Jedoch spiegelt der Datensatz keinerlei Änderungen wieder, die evtl. vom System über Datenaktionen durchgeführt wurden.

`new_embedded_url(**params)`

[Methode(**Objekt) → String]
Gibt die absolute URL zurück für das Eingabe-Formular zum Anlegen neuer Datensätze dieser App. Bei dieser URL ist das Formular in den üblichen LivingApps-Rahmen eingebettet und der Benutzer muß eingeloggt sein um es benutzen zu können.

Mittels `params` können der URL zusätzliche Parameter übergeben werden. Als Werte werden sowohl Strings unterstützt als auch Listen von Strings (in diesem Fall wird der Parameter mehrmals an die URL angefügt). Außerdem wird beim Wert `None` der Parameter ignoriert.

Beispiel 1:

```
<?print app.new_embedded_url()?>
```

erzeugt:

```
https://my.living-apps.de/dateneingabe/1234567890abcdef12345678/new
```

Beispiel 2:

```
<?print app.new_embedded_url(view="abcdef12345678901234")?>
```

erzeugt:

```
https://my.living-apps.de/dateneingabe/1234567890abcdef12345678/new?
↪view=abcdef12345678901234
```

Beispiel 3:

```
<?print app.new_embedded_url(a=["17", 23], b=None, c=[today(), None])?>
```

erzeugt:

```
https://my.living-apps.de/dateneingabe/1234567890abcdef12345678/new?a=17&a=23&
↪c=2022-11-10
```

new_standalone_url(params)**

[Methode(**Objekt) → String]
Gibt die absolute URL zurück für das Eingabe-Formular zum Anlegen neuer Datensätze dieser App. Bei dieser URL ist das Formular nicht in den üblichen LivingApps-Rahmen eingebettet sondern unabhängig und kann auch von nicht eingeloggten Benutzern ausgefüllt werden.

Mittels `params` können der URL zusätzliche Parameter übergeben werden.

Beispiel 1:

```
<?print app.new_standalone_url()?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/new
```

Beispiel 2:

```
<?print app.new_standalone_url(view="abcdef12345678901234")?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/new?
↪view=abcdef12345678901234
```

Beispiel 3:

```
<?print app.new_standalone_url(a={"17", 23}, b=None, c=[today(), None])?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/new?a=17&a=23&
↪c=2022-11-10
```

template_url(identifier, record=None, /, **params)

[Methode(String, *Record* oder `None`, **Objekt) → String]
Gibt die absolute URL zurück für ein Anzeige-Template mit dem Identifizierer `identifier`. Ist `record` `None` so wird ein Link zu einem Listen-Template generiert, wird für `record` ein Datensatz übergeben, so wird ein Link zu einem Detail-Template generiert (Die verlinkte App ist dabei die App für die diese Methode aufgerufen wird, nicht die App zu der der Datensatz gehört).

Mittels `params` können der URL zusätzliche Parameter übergeben werden.

Beispiel 1:

```
<?print app.template_url("beispiel")?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678?template=beispiel
```

Beispiel 2:

```
<?print app.template_url("beispiel", r)?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/
↪0987654321fedcba1234?template=beispiel
```

Beispiel 3:

```
<?print app.template_url("beispiel", x=17, y=23)?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678?
↪template=beispiel&x=17&y=23
```

home_url()

[Methode() → String]
Gibt die absolute URL zur Detail-Seite einer App zurück.

Beispiel:

```
<?print app.home_url()?>
```

erzeugt:

```
https://my.living-apps.de/apps/1234567890abcdef12345678.htm
```

datamanagement_url()

[Methode() → String]
Gibt die absolute URL zur Datenmanagement-Seite einer App zurück.

Beispiel:

```
<?print app.datamanagement_url()?>
```

erzeugt:

```
https://my.living-apps.de/_id_36_.htm?uuid=1234567890abcdef12345678&
↪dId=1234567890abcdef12345678&resetInfo=true&templateIdentifier=created_
↪1234567890abcdef12345678
```

import_url()

[Methode() → String]
Gibt die absolute URL zur Import-Seite einer App zurück.

Beispiel:

```
<?print app.import_url()?>
```

erzeugt:

```
https://my.living-apps.de/import-export/1234567890abcdef12345678.htm
```

tasks_url()

[Methode() → String]
Gibt die absolute URL zur Aufgaben-Seite einer App zurück.

Beispiel:

```
<?print app.tasks_url()?>
```

erzeugt:

```
https://my.living-apps.de/_id_1073_.htm?uuid=1234567890abcdef12345678&
↪dId=1234567890abcdef12345678&p_tpl_uuid=1234567890abcdef12345678&
↪resetInfo=true&templateIdentifier=created_task_1234567890abcdef12345678
```

datamanagement_config_url()

[Methode() → String]
Gibt die absolute URL zur Datenmanagement-Konfigurations-Seite einer App zurück.

Beispiel:

```
<?print app.datamanagement_config_url()?>
```

erzeugt:

```
https://my.living-apps.de/datenmanagement-konfigurieren/1234567890abcdef12345678.
↪htm
```

permissions_url()

[Methode() → String]
Gibt die absolute URL zur Berechtigungs-Seite einer App zurück.

Beispiel:

```
<?print app.permissions_url()?>
```

erzeugt:

```
https://my.living-apps.de/_id_833_.htm?uuid=1234567890abcdef12345678&
↪dId=1234567890abcdef12345678&resetInfo=true
```

datamanageview_url(identifier)

[Methode(String) → String]
Gibt die absolute URL zu einer Datenauswertungs-Seite im Datenmanagement einer App zurück.

Beispiel:

```
<?print app.datamanageview_url("beispiel")?>
```

erzeugt:

```
https://my.living-apps.de/_id_36_.htm?uuid=1234567890abcdef12345678&
↪dId=1234567890abcdef12345678&resetInfo=true&templateIdentifier=created_
↪1234567890abcdef12345678_datamanage_master_beispiel
```

seq()

[Methode() → Number]
Gibt einen fortlaufenden Integer-Wert zurück, der innerhalb dieser App eindeutig ist.

send_mail()

[Methode(`from`: String oder `None`, `reply_to`: String oder `None`, `to`: String oder `None`, `cc`: String oder `None`, `bcc`: String oder `None`, `subject`: String oder `None`, `body_text`: String oder `None`, `body_html`: String oder `None`, attachments: *File* oder `None`) → `None`]
Verschickt eine E-Mail.

Nähere Informationen finden Sie unter *E-Mails über die LivingAPI versenden*.

custom

[Objekt]
Dieses Attribut kann vom Benutzer für beliebige zusätzliche Informationen gesetzt werden.

x_<identifier>

[Objekt]
Es werden beliebige zusätzliche Attribute unterstützt deren Namen mit `x_` beginnt.

Neue Datensatzobjekte können angelegt werden, indem das App-Objekt aufgerufen wird. Feld-Werte können als Keyword-Argumente übergeben werden. So erzeugt z. B. `datasources.beispiel.app()` ein leeres Datensatzobjekt zur App „Beispiel“. Besitzt diese App die Felder `vorname`, `nachname` und `geburtstag` kann mittels folgendem Aufruf ein Datensatz-Objekt mit den entsprechenden Feld-Werten angelegt werden:

```
<?code record = app(
   vorname="Liv",
   nachname="Logic",
   geburtsdatum=@(2000-02-29),
)?>
```

Im Gegensatz zu *insert()* wurde der Datensatz nach diesem Aufruf noch nicht gespeichert, dies erfolgt erst durch den Aufruf von *record.save()*. Daher sind vorher die Attribute `id`, `createdby`, `createdat`, `updatedby` und `updatedat` auch `None` und werden erst durch den Aufruf von *record.save()* gesetzt.

Ist eine App nicht explizit in einer Datenquelle konfiguriert, so kann das zugehörige `App`-Objekt trotzdem innerhalb der LivingAPI auftauchen, wenn ein Datensatz über ein Auswahlfeld auf diese App verweist. In einem solchen Fall ist sowohl `records` als auch `recordcount` `None`.

Wird durch Aufruf des App-Objektes eine neues Datensatz-Object angelegt, und die App hat einen aktiven View so werden die in diesem View konfigurierten Standardwerte für die Felder berücksichtigt.

Außerdem werden bei aktivem View zwar alle Felder des Datensatzes gespeichert, aber Fehler in Feldern, die nicht im aktiven Vies sind werden ignoriert.

27.1.10 AppGroup

Ein `AppGroup`-Objekt stellt eine Gruppe von zusammengehörigen Applikationen dar und hat folgende Attribute:

id
: [String]
Der eindeutige Datenbank-Identifizierer der `AppGroup`.

globals
: [*Globals*]
Verweist auf das `Globals`-Objekt, das globale Informationen enthält.

name
: [String]
Der Name der `AppGroup`.

apps
: [Dictionary(String → *App*) oder `None`]
Dictionary mit den Apps, die zu dieser Gruppe gehören. Die Schlüssel des Dictionarys sind die internen Datenbank-Identifizierer der `App` und die Werte sind `App`-Objekte.

main_app
: [*App* oder `None`]
Die Haupt-App dieser Gruppe.

27.1.11 Control

`Control`-Objekte beinhalten die Metadaten eines Feldes einer Applikation. Je nach Feld-Typ ist dies eine spezieller Objekt-Typ (also z.B. `TextControl` für ein normales einzeiliges Text-Feld, `DatetimeMinuteControl` für ein Datumsfeld mit Stunden- und Minuten-Angaben, `LookupSelectControl` für ein Auswahlfeld etc.).

Allgemeine Attribute

Allen Objekte-Typen gemeinsam sind folgende Attribute:

id
: [String]
Der eindeutige Datenbank-Identifizierer des Feldes.

identifier
[String]
Der menschenlesbare Identifizierer des Feldes (dies ist eine Version der Feld-Beschriftung, die in einen gültigen UL4-Variablennamen ungewandelt wurde und eindeutig ist).

type
[String]
Der Typ des Feldes. Mögliche Werte sind `"string"`, `"int"`, `"number"`, `"date"`, `"lookup"`, `"applookup"`, `"multiplelookup"`, `"multipleapplookup"`, `"bool"`, `"file"` und `"geo"`. Hierzu können Sie im Kapitel *Übersicht der Feldtypen* mehr lesen.

subtype
[String oder `None`]
Der Feld-Untertyp.

Typ	Mögliche Untertypen
`"string"`	`"text"`, `"textarea"`, `"email"`, `"url"`, `"password"` und `"tel"`.
`"date"`	`"date"`, `"datetimeminute"` und `"datetimesecond"`.
`"lookup"`	`"select"`, `"choice"` und `"radio"`.
`"applookup"`	`"select"` und `"choice"`.
`"multiplelookup"`	`"select"`, `"choice"` und `"checkbox"`.
`"multipleapplookup"`	`"select"` und `"choice"`.

Bei allen anderen ist `subtype` immer `None`.

Hierzu können Sie im Kapitel *Übersicht der Feldtypen* mehr lesen.

fulltype
[String]
Feld-Typ und Feld-Untertyp (falls vorhanden) getrennt mit `/`, also z.B. `number` oder `lookup/select` usw..

app
[*App*]
Die App zu der dieses Feld gehört.

label
[String]
Die Feld-Beschriftung.

Dieses Attribut kann geändert werden.

description
[String]
Eine Beschreibung des Feldes.

Dieses Attribut kann geändert werden.

priority
[Bool]
Gibt an, ob das Feld hohe Priorität besitzt, d.h. in der Listenansicht angezeigt werden soll.

Dieses Attribut kann geändert werden.

in_list
[Bool]
`in_list` ist ein anderer Name für `priority`.

Dieses Attribut kann geändert werden.

in_mobile_list
[Bool]
Gibt an, ob das Feld in der Listenansicht auf Handys angezeigt werden soll. (Dies wird nur berücksichtigigt, wenn `in_list` `True` ist).

Dieses Attribut kann geändert werden.

in_text

[Bool]
Gibt an, ob das Feld in einer Text-Ansicht des Datensatzes angezeigt werden soll. Dies ist z.B. der Fall, wenn der Datensatz als Wert eines Applookup-Feldes angezeigt werden soll, oder in einer Uberschrift oder sonstigen Meldung.

Dieses Attribut kann geändert werden.

order

[Integer]
Die Feld-Reihenfolge, d.h. nach dieser Zahl sind die Felder im `App`-Attribut `controls` (bzw. im `Record`-Attribut `fields`) sortiert.

top

[Integer oder `None`]
Vertikale Position des Eingabefeldes im Eingabeformular in der aktiven Formularvariante. Gibt es keine aktive Formularvariante, ist `top` `None`.

left

[Integer oder `None`]
Horizontale Position des Eingabefeldes im Eingabeformular in der aktiven Formularvariante. Gibt es keine aktive Formularvariante, ist `left` `None`.

width

[Integer oder `None`]
Breite des Eingabefeldes im Eingabeformular in der aktiven Formularvariante. Gibt es keine aktive Formularvariante, ist `width` `None`.

height

[Integer oder `None`]
Höhe des Eingabefeldes im Eingabeformular in der aktiven Formularvariante. Gibt es keine aktive Formularvariante, ist `height` `None`.

z_index

[Integer]
Überlagerungs-Reihenfolge des Eingabefeldes in der aktiven Formularvariante. Gibt es keine aktive Formularvariante, ist `z_index` `None`.

tabindex

[Integer]
Tab-Reihenfolge des Eingabefeldes im Eingabeformular in der aktiven Formularvariante (zum Durchschalten der Eingabefelder mit Hilfe der Tab-Taste). Gibt es keine aktive Formularvariante, ist `tabindex` `None`.

required

[Bool]
`True` wenn dieses Feld in der aktiven Formularvariante als Pflichfeld konfiguriert wurde. Gibt es keine aktive Formularvariante, ist `required` `False`.

mode

[`"display"` oder `"edit"`]
Kann der Wert dieses Feldes im Eingabeformular in der aktiven Formularvariante bearbeitet werden (`"edit"`) oder wird der Wert nur angezeigt (`"display"`)? (`"display"` wird verwendet, wenn das Feld im FormBuilder mit *Inhalt zur zum Lesen anzeigen* konfiguriert wurde.) Der Wert wird beim Abschicken nicht berücksichtigt.

In Anzeige-Templates wird `mode` als Default-Wert für *mode* verwendet.

Der Standardwert ist `"edit"`. Gibt es keine aktive Formularvariante, ist der Standardwert `"display"`.

labelpos

[`"top"`, `"left"`, `"right"`, `"bottom"` oder `None`]

Position der Feldbeschriftung relativ zum Eingabefeld in der aktiven Formularvariante. Bei `None` wird die Beschriftung gar nicht angezeigt. Gibt es keine aktive Formularvariante, ist `mode "left"`.

`labelwidth`
[Integer oder `None`.]
Breite des Labels in der aktiven Formularvariante. Gibt es keine aktive Formularvariante, ist `labelwidth "LEFT"`.

`autoalign`
[Bool]
`True` wenn die Breite des Labels automatisch vom FormBuilder berechnet wird. `False` wenn sie der Benutzer selbst angepasst hat. Gibt es keine aktive Formularvariante, ist `autoalign True`.

`in_active_view()`
[Methode() → Bool]
Gibt an, ob es dieses `Control` in der aktiven Formularvariante gibt. Gibt es keine aktive Formularvariante, ist der Return-Wert von `in_active_view() False`.

`save()`
[Methode() → Bool]
Speichert die Werte der veränderbaren Attribute in der Datenbank.

Veränderbare Attribute sind `label`, `description`, `priority`, `in_mobile_list`, `in_text` und `required`.

`is_focused()`
[Methode() → Bool]
Gibt an, ob es dieses `Control` den Eingabefokus hat.

`custom`
[Objekt]
Dieses Attribut kann vom Benutzer für beliebige zusätzliche Informationen gesetzt werden.

`x_<identifier>`
[Objekt]
Es werden beliebige zusätzliche Attribute unterstützt deren Namen mit `x_` beginnt.

Attribute für String-Felder

Zusätzlich zu obigen Attributen besitzt ein `StringControl` noch folgende Attribute:

`placeholder`
[String]
Der Platzhalter für das HTML-Eingabefeld.

`minlength`
[Integer oder `None`]
Die minimale String-Länge (`None` bedeutet ohne Einschränkung).

`maxlength`
[Integer oder `None`]
Die maximale String-Länge (`None` bedeutet ohne Einschränkung).

`StringControl`s des Untertyps `textarea` besitzen weiterhin noch folgendes Attribut:

`encrypted`
[String oder `None`]
Konfiguriert die Verschlüsselung von Feldinhalten. Die möglichen Werte sind:

`None`
Das Feld wird nicht verschlüsselt.

`"force"`
Die Feldverschlüsselung wird erzwungen. In einem Formular das eines oder mehrere dieser Felder enthält, werden alle Felder mit dieser Einstellung beim Abschicken verschlüsselt.

`"optional"`
Der Benutzer kann selbst entscheiden, ob er Verschlüsseln möchte oder nicht.

Attribute für Zahlen-Felder

Zusätzlich zu obigen Attributen besitzt ein `NumberControl` noch folgende Attribute:

`precision`
[Integer oder `None`]
Die Anzahl der Nachkommastellen. Wird `precision` im Formular-Template auf eine Zahl gesetzt, so wird außerdem für die Eingabe ein HTML-Element `<input type="number">` statt `<input type="text">` verwendet.

`precision` im Update-Template zu setzen hat keinerlei Einfluß auf die Anzahl der Nachkommastellen und das verwendete HTML-Element.

`minimum`
[Zahl oder `None`]
Der minimal erlaubte Wert. Kleinere Werte werde automatisch durch den Wert von `minimum` ersetzt. Ist `minimum` `None` gibt es keine Untergrenze.

`maximum`
[Zahl oder `None`]
Der maximal erlaubte Wert. Größere Werte werde automatisch durch den Wert von `maximum` ersetzt. Ist `maximum` `None` gibt es keine Obergrenze.

Attribute für Lookup-Felder

Zusätzlich zu obigen Attributen besitzen `LookupControl`- und `MultipleLookupControl`-Objekte noch folgende Attribute:

`lookupdata`
[Dictionary(String → *LookupItem*)]
`lookupdata` beinhaltet die Auswahlmöglichkeiten als Dictionary (in der Reihenfolge wie die Optionen vom Benutzer angelegt wurde). Die Schlüssel sind die Identifierer der Option und die Werte sind *LookupItem*-Objekt.

`autoexpandable`
[Bool]
`True` wenn in FormBuilder für das Feld *Auto-Hinzufügen aktivieren* konfiguriert wurde. In diesem Fall ist es möglich dem zugehörigen *Field*-Objekt einen beliebigen Wert zu geben. Ist dieser Wert weder als *LookupItem* `key`-Attribut noch als `label`-Attribut bekannt, wird ein neues *LookupItem* hinzugefügt.

Attribute für AppLookup-Felder

Zusätzlich zu obigen Attributen besitzen `AppLookupControl`- und `MultipleAppLookupControl`-Objekte noch folgende Attribute:

`lookup_app`
[*App*]
Die Zielapp aus der Datensätze als Wert für dieses Feld ausgewählt werden können.

`lookup_controls`
[Dictionary(String → *Control*)]
`lookup_controls` ist ein Dictionary, das die Felder beinhaltet, die der Benutzer zur Anzeige für die Auswahl konfiguriert hat. Die Schlüssel in diesem Dictionary sind die Feld-Identifizierer und die Werte sind *Control*-Objekte.

Attribute für Lookup-und AppLookup-Felder

`LookupControl`-, `MultipleLookupControl`-, `AppLookupControl`- und `MultipleAppLookupControl`-Objekte besitzen zusätzlich noch folgende Attribute:

`none_key`
[String oder `None`]
Ist `none_key` nicht `None` (was nur bei einem aktiven *View* der Fall sein kann) wird im Eingabeformular eine „Nichts ausgewählt"-Option angeboten. Der Wert von `none_key` wird als Wert dieser Auswahl-Option verwendet. Ist `none_key` `None` wird keine „Nichts ausgewählt"-Option angezeigt.

`none_label`
[String oder `None`]
Wird als Beschriftung der „Nichts ausgewählt"-Option verwendet, wenn diese „Nichts ausgewählt"-Option angeboten wird. Ist `none_label` `None` sollte eine generische Beschriftung verwendet werden.

`search_url`
[String]
Nur bei `AppLookupChoiceControl` verfügbar. Enthält die Basis-URL, um per Ajax-Request in einer für grosse Datenmengen optimierten Auswahlbox suchen zu können. Default ist `field_<IDENTIFIER>_search`.

`search_param_name`
[String]
Nur bei `AppLookupChoiceControl` verfügbar. Der Name des Request-Parameters, der den zu suchenden Wert enthält. Default ist `q`.

`target_param_name`
[String]
Nur bei `AppLookupChoiceControl` verfügbar. Der Name des Request-Parameters, in dem der CSS-Selektor übergeben wird, den die Suchseite benutzt, um die neuen Optionen an die richte Stelle der HTML-Seite zu setzen. Default ist `target`.

Attribute für Datums-Felder

Zusätzlich zu obigen Attibuten haben `DateControl`, `DatetimeMinuteControl` und `DatetimeSecondControl` noch das folgende Attribut:

`format`
[String]
`format` ist ein für den Datums-Typ passender UL4-Formatstring, der die in `globals.lang` konfigurierte Sprache berücksichtigt.

D.h. das z.B. für ein `DatetimeMinuteControl`-Objekt `format` der Wert `"%d.%m.%Y %H:%M"` hat, wenn `globals.lang` `"de"` ist.

27.1.12 Record

Ein Record-Objekt stellt einen Datensatz einer App dar und hat folgende Attribute:

`id`
[String oder `None`]
Der eindeutige Datensatz-Identifizierer. Wurde das Datensatz-Objekt neu angelegt (durch Aufruf des App-Objektes) so ist `id` noch `None` und wird erst beim Abspeichern (mittels *save()*) gesetzt.

`app`
[*App*]
Die App zu der dieser Datensatz gehört.

`createdat`
[Datum]
Der Zeitpunkt zu dem dieser Datensatz angelegt wurde.

`createdby`
[*User*]
Der Benutzer der diesen Datensatz angelegt hat.

`updatedat`
[Datum oder `None`]
Der Zeitpunkt zu dem dieser Datensatz zuletzt geändert wurde. Wurde der Datensatz noch nie geändert, so ist `updatedat` `None`.

`updatedby`
[*User* oder `None`]
Der Benutzer, der den Datensatz zuletzt geändert hat. Wurde der Datensatz noch nie geändert, so ist `updatedby` `None`.

`fields`
[Dictionary(String → *Field*)]
`fields` beinhaltet die Feld-Werte des Datensatzes. Einträge in `fields` haben dieselbe Reihenfolge, wie die Einträge im `controls`-Attribut eines `App`-Objekts. Die Schlüssel sind wieder die Feld-Identifizierer und die Werte sind `Field`-Objekte.

`f_<identifier>`
[*Field*]
Die `Field`-Objekte stehen auch über sogenannte „Shortcut"-Attribute zur Verfügung. `record.fields.vorname` steht auch z.B. direkt als `record.f_vorname` zur Verfügung.

`values`
[Dictionary(String → Objekt)]
`values` beinhaltet statt *Field*-Objekten direkt die Werte der Felder. Für alle Felder `x` gilt:

```
record.fields.x.value is record.values.x
```

`v_<identifier>`
[Objekt]
Auch Feld-Werte stehen über „Shortcut"-Attribute zur Verfügung. `record.values.vorname` also z.B. als `record.v_vorname`.

`attachments`
[Dictionary(String → *Attachment*)]
Die Anhänge zu diesem Datensatz. Die Dictionary-Schlüssel sind die Datenbank-Identifizierer des Anhangs, die Werte sind *Attachment*-Objekte.

`children`
[Dictionary(String → Dictionary(String → *Record*))]
Die diesem Datensatz zugeordneten Datensätze. Der Dictionary-Schlüssel auf oberster Ebene ist dabei der *Identifizierer*, der dieser Zuordnung in der Konfiguration gegeben wurde. Das Dictionary auf zweiter Ebene beinhaltet die zugeordneten Datensätze. Die Schlüssel sind die Datensatz-Identifizierer und die Werte sind *Record*-Objekte. Als Sortierung wird die in der Konfiguration festgelegte Sortierung verwendet.

`c_<identifier>`
[Dictionary(String → *Record*)]
Auch für zugeordnete Datensätze gibt es Shortcut-Attribute. `record.children.beispiel`, steht auch direkt als `record.c_beispiel` zur Verfügung.

`details`
[Dictionary(String → *RecordChildren*)]
Weitergehende Informationen zu den diesem Datensatz zugeordneten Datensätzen. Der Dictionary-Schlüssel ist dabei der *Identifizierer*, der dieser Zuordnung in der Konfiguration gegeben wurde. Die Werte sind *RecordChildren*-Objekte.

`d_<identifier>`
[*RecordChildren*]
Auch für weitergehende Zuordnungsinformationen gibt es Shortcut-Attribute. `record.details.beispiel`, steht auch direkt als `record.d_beispiel` zur Verfügung.

errors

[Liste(String)]
Fehlermeldungen die den gesamten Datensatz betreffen als Liste von Strings. Diese werden im Augenblick nicht von System gesetzt, können aber vom Benutzer beliebig geändert werden.

add_error(error)

[Methode(String) → None]
Fügt die übergebene Fehlermeldung zur Liste der Fehlermeldungen (errors) hinzu. Diese Methode erwartet die Fehlermeldung als Argument. Das heißt, der Aufruf sieht beispielsweise so aus: record.add_error("Die Bearbeitungszeit ist abgelaufen!").

has_errors()

[Methode() → Bool]
Gibt zurück, ob dieser Datensatz oder eines seiner Felder eine Fehlermeldung hat. Diese Methode erwartet keine Argumente. Das heißt, der Aufruf sieht beispielsweise so aus: record.has_errors().

clear_errors()

[Methode() → None]
Leert die Fehlermeldungs-Liste zu diesem Datensatz aber nicht die der Felder. Diese Methode erwartet keine Argumente. Das heißt der Aufruf sieht beispielsweise so aus: record.clear_errors().

clear_all_errors()

[Methode() → None]
Leert die Fehlermeldungs-Listen zu diesem Datensatz und zu den dazugehörigen Feldern. Diese Methode erwartet keine Argumente. Das heißt der Aufruf sieht beispielsweise so aus: record.clear_all_errors().

is_dirty()

[Methode() → Bool]
Gibt an, ob Feld-Werte geändert wurden. Ist der Datensatz noch gar nicht gespeichert, gibt is_dirty() immer True zurück.

update(values)**

[Methode(**Objekt) → None]
Mithilfe von update() kann der Datensatz verändert werden. Die Parameter müssen per Schlüsselwort übergeben werden. Der Parametername ist dabei jeweils der Feld-Identifizierer des zu setzenden Feldes.

D.h. ein Aufruf sieht z.B. so aus:

```
record.update(vorname="Max", nachname="Mustermann")
```

delete()

[Methode() → None]
Mithilfe von delete kann der Datensatz gelöscht werden. Diese Methode benötigt keine Parameter.

Diese Methode kann auch aufgerufen, wenn der Datensatz noch gar nicht gespeichert wurde. In diesen Fall werden keine Datenbankänderungen vorgenommen.

Das Attribut id wird nach dem Aufruf auf None gesetzt und der Datensatz wird als gelöscht markiert und nachfolgende Aufrufe von *save* bleiben wirkungslos.

save(force, sync)

[Methode(Bool, Bool) → Bool]
Mit der Methode save kann ein Datensatzobjekt abgespeichert werden. Ist dieses Datensatzobjekt neu angelegt (d.h. id ist None) wird beim Aufruf von save ein neuer Datensatz angelegt und beim Datensatzobjekt der eindeutige Datensatz-Identifizierer (id) vergeben, sowie die Attribute createdby auf globals.user (eingeloggter Benutzer) und createdat auf das aktuelle Datum gesetzt. Ist das Datensatzobjekt nicht neu, werden beim Aufruf von save nur die geänderten Felder gespeichert, sowie das Attribut updatedby auf globals.user und updatedat auf das aktuelle Datum gesetzt.

Ist force False, so wird der Datensatz nicht gespeichert, wenn er oder einer seiner Felder eine Fehlermeldung hat (d.h. wenn has_errors True zurückgibt), oder wenn Felder von Typ applookup, multipleapplookup oder file auf noch nicht gespeicherte Datensätze oder Dateien verweisen. Statt dessen wird ein Ausnahme-Fehler erzeugt, der die Ausführung des Anzeige-Templates abbricht. Dieser

Ausnahme-Fehler kann dann unter *Konfiguration → Erweitert → Anzeige-Templates → (Template) → Meldungen in aktueller Version* oder *Konfiguration → Erweitert → Meldungen* angesehen werden.

Ist `force True` so werden Fehlermeldungen am Datensatz oder seinen Feldern ignoriert, für noch nicht gespeicherte Datensätze oder Dateien wird stattdessen `None` als Wert verwendet und der Datensatz wird trotzdem gespeichert.

Wenn für `sync False` übergeben wird (der Default) ist das Verhalten wie oben beschrieben, wird `True` übergeben, so wird nach dem Speichern das `Record`-Objekt mit dem Datenbank-Inhalt aktualisiert. Das heißt insbesondere, daß Änderungen die durch Datenaktionen an dem Datensatz durchgeführt wurden, im `Record`-Objekt auftauchen.

Der Rückgabewert von `save` gibt an, ob der Datensatz tatsächlich in der Datenbank gespeichert wurde, oder ob dabei ein Fehler aufgetreten ist. Die entsprechende Fehlermeldung wird dabei an den Datensatz gehängt.

Bemerkung

Für noch nicht gespeicherte Datensätze oder Dateien wird beim Speichern mit `force=True` zusätzlich eine Fehlermeldung an das Feld-Objekt gehängt (dies kann erst zum Speicherzeitpunkt gemacht werden, nicht zum Zeitpunkt der Zuweisung des Feld-Wertes, da es ja möglich ist, daß ein referenziertes Objekt erst gespeichert wird, nachdem es dem Feld zugewiesen wurde).

Noch nicht gespeicherte Dateien können dabei bei der Verwendung innerhalb von Anzeige-Templates nicht auftreten, jedoch bei der Verwendung über das *Python-SDK*.

Bemerkung

Wurde vor dem Aufruf von `save` *delete* aufgerufen, bleibt `save` wirkungslos, der Rückgabewert ist dann `None`.

`executeaction(identifier)`

[Methode(String) → `None`]
Mithilfe von `executeaction` kann eine Daten-Aktion auf dem Datensatz ausgeführt werden. Diese Daten-Aktionen können unter *Konfiguration → Erweitert* in der *Aktionen*-Maske definiert werden. Als Parameter muß der Identifizierer der Daten-Aktion übergeben werden.

Gibt es z.B. eine Datenaktion mit dem Identifizierer `freigeben`, so kann für den Datensatz `record` diese Datenaktion folgendermaßen aufgerufen werden:

```
<?code record.executeaction("freigeben")?>
```

`edit_embedded_url(**params)`

[Methode(**Objekt) → String]
Gibt die absolute URL zurück für das Eingabe-Formular zum Bearbeiten dieses Datensatzes. Bei dieser URL ist das Formular in den üblichen LivingApps-Rahmen eingebettet und der Benutzer muß eingeloggt sein um es benutzen zu können.

Mittels `params` können der URL zusätzliche Parameter übergeben werden. Als Werte werden sowohl Strings unterstützt als auch Listen von Strings (in diesem Fall wird der Parameter mehrmals an die URL angefügt). Außerdem wird beim Wert `None` der Parameter ignoriert.

Beispiel 1:

```
<?print record.edit_embedded_url()?>
```

erzeugt:

```
https://my.living-apps.de/dateneingabe/1234567890abcdef12345678/
↪10293848576afbecd12345678/edit
```

Beispiel 2:

```
<?print record.edit_embedded_url(view="abcdef12345678901234")?>
```

erzeugt:

```
https://my.living-apps.de/dateneingabe/1234567890abcdef12345678/
↪10293848576afbecd12345678/edit?view=abcdef12345678901234
```

Beispiel 3:

```
<?print record.edit_embedded_url(a=["17", 23], b=None, c=[today(), None])?>
```

erzeugt:

```
https://my.living-apps.de/dateneingabe/1234567890abcdef12345678/
↪10293848576afbecd12345678/edit?a=17&a=23&c=2022-11-10
```

edit_standalone_url(params)**

[Methode(**Objekt) → String]
Gibt die absolute URL zurück für das Eingabe-Formular zum Bearbeiten dieses Datensatzes. Bei dieser URL ist das Formular nicht in den üblichen LivingApps-Rahmen eingebettet sondern unabhängig und kann auch von nicht eingeloggten Benutzern ausgefüllt werden.

Mittels `params` können der URL zusätzliche Parameter übergeben werden.

Beispiel 1:

```
<?print record.edit_standalone_url()?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/
↪10293848576afbecd12345678/edit
```

Beispiel 2:

```
<?print record.edit_standalone_url(view="abcdef12345678901234")?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/
↪10293848576afbecd12345678/edit?view=abcdef12345678901234
```

Beispiel 3:

```
<?print record.edit_standalone_url(a=["17", 23], b=None, c=[today(), None])?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/
↪10293848576afbecd12345678/edit?a=17&a=23&c=2022-11-10
```

display_embedded_url(params)**

[Methode(**Objekt) → String]
Gibt die absolute URL für die Antwortseite zurück. Bei dieser URL ist die Seite in den üblichen LivingApps-Rahmen eingebettet und der Benutzer muß eingeloggt sein, um diese Seite angezeigt zu bekommen.

Mittels `params` können der URL zusätzliche Parameter übergeben werden. Als Werte werden sowohl Strings unterstützt als auch Listen von Strings (in diesem Fall wird der Parameter mehrmals an die URL angefügt). Außerdem wird beim Wert `None` der Parameter ignoriert.

Beispiel 1:

```
<?print record.display_embedded_url()?>
```

erzeugt:

```
https://my.living-apps.de/dateneingabe/1234567890abcdef12345678/
↪10293848576afbecd12345678/display
```

Beispiel 2:

```
<?print record.display_embedded_url(view="abcdef12345678901234")?>
```

erzeugt:

```
https://my.living-apps.de/dateneingabe/1234567890abcdef12345678/
↪10293848576afbecd12345678/display?view=abcdef12345678901234
```

Beispiel 3:

```
<?print record.display_embedded_url(a=["17", 23], b=None, c=[today(), None])?>
```

erzeugt:

```
https://my.living-apps.de/dateneingabe/1234567890abcdef12345678/
↪10293848576afbecd12345678/display?a=17&a=23&c=2022-11-10
```

display_standalone_url(params)**

[Methode(**Objekt) → String]

Gibt die absolute URL für die Antwortseite zurück. Bei dieser URL ist die Seite nicht in den üblichen LivingApps-Rahmen eingebettet sondern unabhängig und kann auch von nicht eingeloggten Benutzern gesehen werden.

Mittels `params` können der URL zusätzliche Parameter übergeben werden.

Beispiel 1:

```
<?print record.display_standalone_url()?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/
↪10293848576afbecd12345678/display
```

Beispiel 2:

```
<?print record.display_standalone_url(view="abcdef12345678901234")?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/
↪10293848576afbecd12345678/display?view=abcdef12345678901234
```

Beispiel 3:

```
<?print record.display_standalone_url(a=["17", 23], b=None, c=[today(), None])?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/
↪10293848576afbecd12345678/display?a=17&a=23&c=2022-11-10
```

display_url(params)**

[Methode(**Objekt) → String]
Gibt die absolute URL zurück für die Antwortseite. Mit dem System-Parameter `la_default_form_variant` kann gesteuert werden, ob die `embedded` oder die `standalone` Variante erzeugt wird. Hat dieser den Wert `standalone` wird diese Variante ausgespielt, sonst `embedded` (auch wenn der Parameter nicht angegeben wird).

Beispiel 1:

```
<?print record.display_url()?>
```

erzeugt:

```
https://my.living-apps.de/dateneingabe/1234567890abcdef12345678/
↪10293848576afbecd12345678/display
```

Beispiel 2:

```
<?print record.display_url(la_default_form_variant="standalone")?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/
↪10293848576afbecd12345678/display
```

Mit weiteren Parametern kann die Liste der Request-Parameter in der URL ergänzt werden. Beispiele hierfür finden sich in den Abschnitten *display_embedded_url(**params)* und *display_standalone_url(**params)*.

template_url(identifier, /, **params)

[Methode(String, **Objekt) → String]
Gibt die absolute URL zurück für ein Detail-Anzeige-Template mit dem Identifizierer `identifier`. Die verlinkte App ist dabei die App zu der Datensatz gehört).

Mittels `params` können der URL zusätzliche Parameter übergeben werden.

Beispiel 1:

```
<?print record.template_url("beispiel")?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/
↪10293848576afbecd12345678?template=beispiel
```

Beispiel 2:

```
<?print record.template_url("beispiel", x=17, y=23)?>
```

erzeugt:

```
https://my.living-apps.de/gateway/apps/1234567890abcdef12345678/
↪10293848576afbecd12345678?template=beispiel&x=17&y=23
```

send_mail()

[Methode(`from`: String oder `None`, `reply_to`: String oder `None`, `to`: String oder `None`, `cc`: String oder `None`, `bcc`: String oder `None`, `subject`: String oder `None`, `body_text`: String oder `None`, `body_html`: String oder `None`, attachments: *File* oder `None`) → `None`]
Verschickt eine E-Mail.

Nähere Informationen finden Sie unter *E-Mails über die LivingAPI versenden*.

custom
[Objekt]
Dieses Attribut kann vom Benutzer für beliebige zusätzliche Informationen gesetzt werden.

x_<identifier>
[Objekt]
Es werden beliebige zusätzliche Attribute unterstützt deren Namen mit `x_` beginnt.

27.1.13 RecordChildren

Ein `RecordChildren`-Objekt enthält die untergeordneten Datensätze zu einem übergeordneten Datensatz und hat folgende Attribute:

id
[String]
Der interne Datenbank-Identifizierer für dieses `RecordChildren`-Objekt.

record
[*Record*]
Der übergeordnete Datensatz zu dem dieses `RecordChildren`-Objekt gehört.

datasourcechildren
[*DataSourceChildren*]
Das `DataSourceChildren`-Objekt, das die Zuordnung definiert.

records
[Dictionary(String → *Record*)]
Die untergeordneten Datensätze. Dieses Dictionary ist sortiert, die Reihenfolge kann in der Konfiguration der untergeordneten Datensätze konfiguriert werden. Die Schlüssel des Dictionarys sind die Datensatz-Identifizierer und die Werte sind *Record*-Objekte. Als Sortierung wird die in der Konfiguration festgelegte Sortierung verwendet.

27.1.14 Field

Ein `Field`-Objekt beinhaltet den Wert eines Feldes für einen Datensatz und besitzt folgende Attribute:

control
[*Control*]
Das `Control`-Objekt für das dieses `Field`-Objekt den Wert beinhaltet.

record
[*Record*]
Das Datensatz-Objekt zu dem dieser Feld-Wert gehört (siehe *Record*).

label
[String]
Die Feld-Beschriftung. Diese erbt das `Field` von seinem `Control`. `label` kann jedoch auch gesetzt werden um für dieses `Field` eine andere Beschriftung zu verwenden. Wird als Wert `None` gesetzt, so wird wieder die ursprüngliche Beschriftung des `Control`s verwendet.

value
[Objekt]
Der Wert des Feldes. Der Typ des Wertes hängt dabei natürlich vom Typ des Feldes ab.

Für String-Felder ist dies ein String, für Häckchen-Felder ein Bool, für Zahlen-Felder eine Zahl und für Datumsfelder ein Datum. Für Datei-Uploads ist es ein *File*-Objekt.

Ist das Feld ein Lookup-Feld (d.h. eine Auswahlbox oder ein Optionenfeld) so gibt es zwei Möglichkeiten: Werden für die Auswahl die Datensätze einer anderen App benutzt, so ist der Feld-Wert das `Record`-Objekt, das diesen anderen Datensatz beschreibt. Im anderen Fall ist der Wert ein *LookupItem*-Objekt.

Ist das Feld ein MultipleLookup-Feld (d.h. eine Mehrfach-Optionenfeld), so ist der Wert eine Liste von *LookupItem*- bzw. *Record*-Objekten.

Ist das Feld ein Upload-Feld, so ist der Wert ein *File*-Objekt.

Bei einem Geo-Feld ist der Wert ein *Geo*-Objekt.

Ist das Feld kein Pflichtfeld, so ist als Wert bei allen Feld-Typen natürlich immer `None` möglich (außer bei Mehrfachoptionen, wo der „leere“ Wert eine leere Liste ist).

Das `Field`-Attribut `value` kann immer gesetzt werden (ohne daß bei nicht passenden Werten eine Fehlermeldung ausgegeben wird), jedoch kann eine Fehler-Meldung im `errors`-Attribut gesetzt werden, wenn der Wert nicht sinnvoll ist (in diesem Fall wird auch der Wert durch `None` (bzw. `[]`) ersetzt).

Die folgende Tabelle zeigt die unterschiedlichen möglichen Typen für Feld-Werte für jeden Feld-Typ nochmal in der Übersicht:

Typ	Wert
`bool`	`bool` (`True` oder `False`) oder `None`
`string`	`str` oder `None`
`int`	`int` oder `None`
`number`	`int`, `float` oder `None`
`date`	`date` (für Untertyp `date`), `datetime` (für Untertyp `datetimeminute` und `datetimesecond`) oder `None`
`geo`	*Geo* oder `None`
`file`	*File* oder `None`
`lookup`	*LookupItem* oder `None`
`multiplelookup`	Liste von *LookupItem*s (leere Liste wenn nichts ausgewählt)
`applookup`	*Record* oder `None`
`multipleapplook`	Liste von *Record*s (leere Liste wenn nichts ausgewählt)

Um das Handling von Formularen zu vereinfachen, ist es bei vielen Feldtypen möglich, das `value`-Attribut auf einen String zu setzen, solange dieser String das richtige Format besitzt. Der Wert wird dann automatisch in den richtigen Typ konvertiert.

Die unterstützten String-Formate sind je nach Feldtyp folgende:

`bool`

Der Leerstring wird zu `None` konvertiert, `"false"`, `"off"`, `"no"` und `"0"` zu `False`, alles andere wird zu `True`.

`int`

Eine Ziffernfolge mit optionalem führenden Minuszeichen wird in die Zahl konvertiert.

Der Leerstring wird zu `None`.

`number`

Es wird versucht zu erraten, ob Punkte oder Kommas als Dezimaltrennzeichen verwendet wurden.

Der Leerstring wird zu `None`.

`date`

Für alle Untertypen werden Strings der Form

- `"yyyy-MM-dd"`
- `"yyyy-MM-dd HH:mm"`
- `"yyyy-MM-dd HH:mm:ss"`
- `"yyyy-MM-dd HH:mm:ss.nnnnnn"`
- `"yyyy-MM-dd HH:mm:ss.nnnnnnxxx"`
- `"yyyy-MM-ddTHH:mm"`
- `"yyyy-MM-ddTHH:mm:ss"`
- `"yyyy-MM-ddTHH:mm:ss.nnnnnn"`
- `"yyyy-MM-ddTHH:mm:ss.nnnnnnxxx"`

unterstützt, sowie die jeweils sprachspezifischen Versionen davon. Der finale Datumswert wird aber auf die dem Feldtyp entprechende Genauigkeit gerundet.

Der Leerstring wird zu `None`.

geo
Es wird ein String der Form `Längengrad,Breitengrad,Info` unterstützt. Fehlt der Längengrad/Breitengrad- oder der Info-Teil, so wird mittels einer Geo-Suche versucht den fehlenden Teil zu bestimmen.

Der Leerstring wird zu `None`.

file
Es werden Strings in der Form unterstützt, wie sie im `id`-Attribut von *File* zu finden ist.

Der Leerstring wird zu `None`.

lookup
Es werden die Identifizierer der Auswahl-Optionen unterstützt.

Der Leerstring wird zu `None`.

multiplelookup
Es werden die Identifizierer der Auswahl-Optionen unterstützt. Es können ebenso Listen von Auswahl-Optionen-Identifizierern übergeben werden.

Der Leerstring wird zu `None`.

applookup
Es werden die `id`-Werte von *Record*-Objekten unterstützt.

Der Leerstring wird zu `None`.

multipleapplookup
Es werden die `id`-Werte von *Record*-Objekten unterstützt. Es können ebenso Listen von `id`-Werten übergeben werden.

Der Leerstring wird zu `None`.

errors
[Liste(String)]
Fehlermeldungen die dieses Feld betreffen als Liste von Strings.

add_error(error)
[Methode(String) → `None`]
Fügt die übergebene Fehlermeldung zur Liste der Fehlermeldungen (`errors`) hinzu. Diese Methode erwartet die Fehlermeldung als Argument. Das heißt, der Aufruf sieht beispielsweise so aus: `field.add_error("Bitte geben Sie eine gültige E-Mail-Adresse ein!")`.

set_error(error)
[Methode(String oder `None`) → `None`]
Ist `error` `None` so wird die Liste der Fehlermeldungen geleert. Ist `error` ein String, so besteht die Liste der Fehlermeldungen aus dieser einen Fehlermeldung.

has_errors()
[Methode() → Bool]
Gibt zurück, ob dieses Feld Fehlermeldungen hat (d.h. ob `errors` nicht leer ist). Diese Methode erwartet keine Argumente. Das heißt, der Aufruf sieht beispielsweise so aus: `field.has_errors()`.

clear_errors()
[Methode() → `None`]
Leert die Fehlermeldungs-Liste zu diesem Feld. Diese Methode erwartet keine Argumente. Das heißt der Aufruf sieht beispielsweise so aus: `field.clear_errors()`.

mode
[`"display"`, `"edit"`, `"hidden"`, `"readonly"` oder `"absent"`]
Gibt an wie das Feld in Anzeige-Templates angezeigt wird. Der Wert wird von *mode* geerbt, kann aber überschrieben werden und unterstützt dann folgende Werte:

"display"
Das Feld wird angezeigt, kann aber nicht bearbeitet werden, der Feldwert wird beim Absenden des Formulars nicht mitgesendet.

"edit"
Das Feld kann bearbeitet werden. Der Feldwert wird mitgesendet.

"readonly"
Bedeutet, dass der Feldwert wie bei `"display"` angezeigt, der Feldwert beim Absenden des Formulars aber mitgesendet wird.

"hidden"
Hierbei wird der Feldwerd nicht angezeigt aber (über ein `"hidden"`-Tag) als Parameter beim Formularabsenden mitgesendet.

"absent"
Bedeutet, dass der Feldwert weder angezeigt noch mitgesendet wird.

lookupdata
[Dictionary(String → *LookupItem*), Dictionary(String → *Record*) oder Dictionary(String → String)]
Dieses Attribut ist nur bei `lookup/multiplelookup`- bzw. `applookup/multipleapplookup`-Feldern vorhanden. Es stellt die Auswahlmenge im Eingabeformular dar, und kann daher verwendet werde um diese Auswahlmenge zu ändern. D.h. es kann damit sowohl festgelegt werden, welche *LookupItem*s bzw. *Record*s zur Auswahl stehen, als auch welche Bezeichnungen dafür angezeigt werden.

Bei `lookup/multiplelookup`-Feldern überschreibt es das `lookupdata`-Attribut des *Control*-Objektes. Allerdings dürfen die Werte auch Strings sein. Dieser werden dann im Formular direkt als Bezeichnungen der Optionen verwendet.

Bei `applookup/multipleapplookup`-Feldern müssen die Keys im Dictionary `id`-Werte von Datensätzen aus der Zielapp sein. Die Werte sind entweder die zugehörigen *Record*-Objekte oder Strings, die dann im Formular direkt als Bezeichnungen der Datensätze verwendet werden.

Wird `lookupdata` auf `None` gesetzt wird wieder die ursprüngliche Auswahlmenge verwendet.

is_empty()
[Methode() → Bool]
Gibt an, ob der Feld-Wert leer ist. Dies ist bei `multiplelookup`- und `multipleapplookup`-Feldern die leere Liste, bei allen anderen Feld-Typen `None`. Diese Methode erwartet keine Argumente. Das heißt, der Aufruf sieht beispielsweise so aus: `field.is_empty()`.

is_dirty()
[Methode() → Bool]
Gibt an, ob der Feld-Wert geändert wurde. Diese Methode erwartet keine Argumente. Das heißt, der Aufruf sieht beispielsweise so aus: `field.is_dirty()`.

enabled
[Bool]
Für Anzeige-Templates ist dieses Attribut irrelevant. Es gibt an, ob ein Eingabefeld für diesen Wert aktiviert werden soll oder nicht und wird nur in Update-Templates verwendet (d.h. wenn `enabled False` ist, erhält das Eingabefeld das HTML-Attribut `disabled="disabled"`; Ein solches Feld kann vom Benutzer nicht mehr geändert werden und der dort eingetragene Wert wird beim Absenden des Formulars nicht mitgesendet).

`enabled` kann auch per Zuweisung gesetzt werden.

writable
[Bool]
Für Anzeige-Templates ist dieses Attribut irrelevant. Es gibt an, ob ein Eingabefeld für diesen Wert beschreibbar angezeigt werden soll oder nicht und wird nur in Update-Templates verwendet (d.h. wenn `writable False` ist, erhält das Eingabefeld das HTML-Attribut `readonly="readonly"`; Ein solches Feld kann vom Benutzer nicht mehr geändert werden, der dort eingetragene Wert wird aber beim Absenden des Formulars mitgesendet).

`writable` kann auch per Zuweisung gesetzt werden.

visible

[Bool]
Für Anzeige-Templates ist dieses Attribut irrelevant. Es gibt an, ob ein Eingabefeld für diesen Wert eingeblendet werden soll oder nicht und wird nur in Update-Templates verwendet (d.h. wenn `visible` `False` ist, wird das Eingabefeld komplett ausgeblendet).

`visible` kann auch per Zuweisung gesetzt werden.

required

[Bool]
Gibt an, ob ein Eingabefeld verpflichtend ist oder nicht. Falls der Wert `None` ist, wird der Wert von `required` im *Control* herangezogen.

`required` kann auch per Zuweisung gesetzt werden.

top

[Integer oder `None`]
Vertikale Position des Eingabefeldes im Eingabeformular in der aktiven Formularvariante. Diese erbt das `Field` von seinem `Control`. `top` kann jedoch auch gesetzt werden um für dieses spezielle Feld eine eigene Position zu verwenden. Wird als Wert `None` gesetzt, so wird wieder die ursprüngliche Position des `Control`s verwendet. Wenn Sie `top` eines Feldes im aktiven Formular ändern wird dieses Feld an die enstprechende Stelle geschoben.

left

[Integer oder `None`]
Horizontale Position des Eingabefeldes im Eingabeformular in der aktiven Formularvariante. Diese erbt das `Field` von seinem `Control`. `left` kann jedoch auch gesetzt werden um für dieses spezielle Feld eine eigene Position zu verwenden. Wird als Wert `None` gesetzt, so wird wieder die ursprüngliche Position des `Control`s verwendet. Wenn Sie `left` eines Feldes im aktiven Formular ändern wird dieses Feld an die enstprechende Stelle geschoben.

width

[Integer oder `None`]
Breite des Eingabefeldes im Eingabeformular in der aktiven Formularvariante. Diese erbt das `Field` von seinem `Control`.

height

[Integer oder `None`]
Höhe des Eingabefeldes im Eingabeformular in der aktiven Formularvariante. Diese erbt das `Field` von seinem `Control`.

custom

[Objekt]
Dieses Attribut kann vom Benutzer für beliebige zusätzliche Informationen gesetzt werden.

x_<identifier>

[Objekt]
Es werden beliebige zusätzliche Attribute unterstützt deren Namen mit `x_` beginnt.

Attribute für Lookup- und Applookup-Felder

`Field`-Object zu Controls vom Typ `LookupControl`, `MultipleLookupControl`, `AppLookupControl` oder `MultipleAppLookupControl` besitzen zusätzlich noch folgende Attribute:

lookupdata

[Dictionary(String → *LookupItem* oder]
Record oder String) `lookupdata` kann gesetzt werden um die Auswahlmenge für dieses Feld zu ersetzen.

Für Felder zu Controls vom Typ `LookupControl` oder `MultipleLookupControl` müssen die Keys `key`-Attibute von *LookupItem*s sein, die Werte entweder *LookupItem*s oder Strings (bei String wird der Wert 1:1 für die Anzeige der Auswahlmöglichkeiten verwendet).

Für Felder zu Controls vom Typ `AppLookupControl` oder `MultipleAppLookupControl` müssen die Keys `id`-Attibute von *Record*-Objekten sein, die Werte entweder *Record*s oder Strings (bei String wird der Wert 1:1 für die Anzeige der Auswahlmöglichkeiten verwendet).

Wird der Wert von `lookupdata` auf `None` gesetzt, so wird wieder die Auswahl, die durch das *Control* definitiert ist, verwendet (Bei `LookupControl` oder `MultipleLookupControl` `control.lookupdata` und bei `AppLookupControl` oder `MultipleAppLookupControl` `control.lookup_app.records`).

`has_custom_lookupdata()`
[Methode() → Bool]
Liefert `True` zurück wenn in `lookupdata` die Auswahlmenge überschrieben wurde.

`search_url`
[String]
Nur bei `AppLookupChoiceField` verfügbar. Enthält die Basis-URL, um per Ajax-Request in einer für grosse Datenmengen optimierten Auswahlbox suchen zu können. Default ist `field_<IDENTIFIER>_search`. Wird aus dem zugehörigen `AppLookupChoiceControl` bezogen, falls nicht im `AppLookupChoiceField` überschrieben.

`search_param_name`
[String]
Nur bei `AppLookupChoiceField` verfügbar. Der Name des Request-Parameters, der den zu suchenden Wert enthält. Default ist `q`. Wird aus dem zugehörigen `AppLookupChoiceControl` bezogen, falls nicht im `AppLookupChoiceField` überschrieben.

`target_param_name`
[String]
Nur bei `AppLookupChoiceField` verfügbar. Der Name des Request-Parameters, in dem der CSS-Selektor übergeben wird, den die Suchseite benutzt, um die neuen Optionen an die richte Stelle der HTML-Seite zu setzen. Default ist `target`. Wird aus dem zugehörigen `AppLookupChoiceControl` bezogen, falls nicht im `AppLookupChoiceField` überschrieben.

27.1.15 `LayoutControl`

Ein `LayoutControl`-Objekt stellt eine „Dekoration“ in einem Eingabeformular dar, die mit keinem Feld von Datensätzen in Verbindung steht. Es gibt drei Feld-Typen:

- `HTMLLayoutControl` für zusätzliches HTML das im Eingabeformular angezeigt wird (*Formatierter Text* im FormBuilder);
- `ImageLayoutControl` für ein Dekobild;
- `ButtonLayoutControl` für einen „Absenden“-Button;

Ein `LayoutControl`-Objekt kann sowohl einer *App* zugeordnet sein (wenn es einen aktiven *View* gibt) als auch direkt einem *View*.

Allgemeine Attribute

Allen Objekte-Typen gemeinsam sind folgende Attribute:

`id`
[String]
Der eindeutige Datenbank-Identifizierer des `LayoutControl`s.

`view`
[*View*]
Das `View`-Objekt zu diesem `LayoutControl`.

`label`
[String]
Die Beschriftung.

`identifier`
[String]

Der menschenlesbare Identifizierer des Feldes (dies ist eine Version der Beschriftung, die in einen gültigen UL4-Variablennamen ungewandelt wurde und eindeutig ist).

type
[String]
Der Typ des Layoutfelder: `"string"`, `"image"` oder `"button"`.

subtype
[String oder `None`]
Der Untertyp des Layoutfelder: `"html"` beim Typ `"string"` und `None` beim Typ `"image"` und `"button"`.

top
[Integer]
Vertikale Position des Feldes im Eingabeformular in dieser bzw. der aktiven Formularvariante.

left
[Integer]
Horizontale Position des Feldes im Eingabeformular in dieser bzw. der aktiven Formularvariante.

width
[Integer]
Breite des Feldes im Eingabeformular in dieser bzw. der aktiven Formularvariante.

height
[Integer]
Höhe des Feldes im Eingabeformular in dieser bzw. der aktiven Formularvariante.

z_index
[Integer]
Überlagerungs-Reihenfolge des Feldes in dieser bzw. der aktiven Formularvariante.

visible
[Bool]
Soll dieses Layoutfeld in dieser bzw. der aktiven Formularvariante angezeigt werden?

x_<identifier>
[Objekt]
Es werden beliebige zusätzliche Attribute unterstützt deren Namen mit `x_` beginnt.

Attribute für `HTMLLayoutControl`-Felder

`HTMLLayoutControl`-Felder haben zusätzlich das folgende Attribut:

value
[string]
Der HTML-Quelltext.

Attribute für `ImageLayoutControl`-Felder

`ImageLayoutControl`-Felder haben zusätzlich das folgenden Attribut:

image
[*File*]
Das Original-Bild, das vom Benutzer hochgeladen wurde.

Zur Anzeige sollte es mittels *scaled_url(...)* auf die durch die Attribute `width` und `height` festgelegte Größe skaliert werden.

27.1.16 `AppParameter`

Ein `AppParameter`-Objekt beinhaltet Informationen über einen Parameter einer App. Es besitzt folgende Attribute:

id
[String]
Der eindeutige Datenbank-Identifizierer der App.

app

[*App* oder `None`]
Die App zu der dieser Parameter gehört (oder `None` wenn dieser Parameter zur Template-Library gehört).

parent

[AppParameter oder `None`]
Der übergeordnete Parameter vom Typ `list` oder `dict` (oder `None` falls dieser Parameter kein Element eines übergeordneten Parameters ist).

type

[String]
Der Parameter-Typ. Mögliche Werte sind:

Wert	Beschreibung
`"bool"`	Ein Wahrheitswert: `True` oder `False`
`"int"`	Eine ganze Zahl
`"number"`	Eine Zahl mit Nachkommastellen
`"str"`	Text
`"html"`	HTML-Text
`"color"`	Eine Farbe
`"date"`	Ein Datum
`"datetime"`	Datum + Uhrzeit
`"datedelta"`	Ein Zeitraum von Tagen
`"datetimedelta"`	Ein Zeitraum von Tagen, Stunden, Minuten und Sekunden
`"monthdelta"`	Ein Zeitraum von Monaten
`"upload"`	Eine hochgeladene Datei
`"app"`	Ein Verweis auf eine App
`"control"`	Ein Verweis auf ein Feld der App zu der dieser Parameter gehört
`"list"`	Eine Liste (bestehend aus Unterparametern)
`"dict"`	Eine Menge an Unterparametern mit Namen (Identifizierer)

Bei allen Typen außer `"list"` und `"dict"` kann der Wert auch `None` sein.

Dieses Attribut kann auch gesetzt werden. Wenn der alte Wert nicht zum neu gesetzten Typ passt, wird der Wert auf `None` gesetzt.

order

[Integer oder `None`]
Die Reihenfolge in der dieser Parameter als Kind eines übergeordneten Parameters aufgeführt ist, wenn der übergeordnete Parameter vom Typ `list` ist (oder `None` sonst). Dieses Attribut kann auch gesetzt werden.

identifier

[String oder `None`]
Der Name des Parameters (oder `None` wenn dieser Parameter einen übergeordneten Parameter vom Typ `list` hat). Dieses Attribut kann auch gesetzt werden.

description

[String]
Die Beschreibung des Parameters. Dieses Attribut kann auch gesetzt werden.

value

[Objekt]
Der Wert des Objekts. Dabei hängt der Typ des Wertes vom Typ des Parameters ab. Dieser Attribut kann auch gesetzt werden. Je nach dem Typ des gesetzten Wertes ändert sich dadurch auch der Parameter-Typ.

createdat

[Datum]
Zeitpunkt, zu dem der Parameter angelegt wurde.

createdby

[*User*]

Benutzer, der den Parameter angelegt hat.

updatedat
[Datum oder None]
Der Zeitpunkt, zu dem der Parameter das letzte Mal geändert wurde. (Wurde der Parameter noch nicht geändert, so ist updatedat None.)

updatedby
[*User* oder None]
Der Benutzer, der den Parameter zuletzt geändert hat. (Wurde der Parameter noch nicht geändert, so ist updatedby None.)

append_param(type, description, value)
[Methode(String, String oder None, Objekt) → AppParameter]
Erzeugt einen neuen Parameter vom Typ type mit der Beschreibung description und dem Wert value und fügt ihn an diesen Parameter vom Typ list an. Der neu angelegte Parameter wird zurückgegeben. Diese Methode ist nur bei Parametern vom Typ list vorhanden.

add_param(type, identifier, description, value)
[Methode(String, String, String oder None, Objekt) → AppParameter]
Erzeugt einen neuen Parameter vom Typ type, mit dem Identifizierer identifier, der Beschreibung description und dem Wert value und fügt ihn an diesen Parameter vom Typ dict an. Der neu angelegte Parameter wird zurückgegeben. Diese Methode ist nur bei Parametern vom Typ dict vorhanden.

save(sync)
[Methode(Bool) → None]
Speichert einen neuen oder geänderten Parameter in der Datenbank ab.

Wenn für sync True übergeben, so wird nach dem Speichern das AppParameter-Objekt mit dem Datenbank-Inhalt aktualisiert.

delete()
[Methode() → None]
Löscht den Parameter aus der Datenbank.

state
[String]
Der Zustand des Parameters. Mögliche Werte sind:

Wert	Beschreibung
"new"	Das Parameter-Objekt wurde neu angelegt, ist aber noch nicht gespeichert.
"saved"	Das Parameter-Objekt wurde aus der Datenbank geladen und noch nicht verändert.
"changed"	Das Parameter-Objekt wurde verändert.
"deleted"	Der Parameter wurde bereits in der Datenbank gelöscht.

is_dirty()
[Methode() → Bool]
Git True zurück, wenn das Parameter-Objekt neu ist oder geändert wurde (d.h. wenn state "new" oder "changed" ist).

is_deleted()
[Methode() → Bool]
Git True zurück, wenn das Parameter-Objekt in der Datenbank gelöscht wude (d.h. wenn state "deleted" ist).

27.1.17 MenuItem

MenuItem-Objekte stellen zusätzliche von Ihnen konfigurierte Menüeinträge dar, die Sie unter *Konfiguration* → *Erweitert* und dort bei linken Menüleiste unter *App-Menüs* konfigurieren können.

Ein MenuItem-Objekt hat folgende Attribute:

id
[String]
Der interne Datenbank-Identifizierer für diesen Link.

app
[*App*]
Die App zu der dieser Link gehört.

identifier
[String]
Der Link-Identifier.

label
[String]
Der Link-Text.

parent
[*MenuItem* oder `None`]
Das übergeordnete `MenuItem`.

target_type
[String]
Der Typ auf den der Link verweist. Kann die Werte `newform_standalone`, `newform_embedded`, `datamanagement`, `customoverview`, `evaluation`, `import_export`, `tasks`, `formbuilder`, `workflow_manager`, `data_config`, `permissions`, `expert`, `viewtemplate`, `datamanageview` oder `custom` haben.

icon
[String]
Der Name eines Font-Awesome Icons für den Link. Der Name kein einen der Suffixe `-brand`, `-sharp-solid`, `-solid`, `-regular`, `-light`, `-thin` oder `-duotone` haben, um den entsprechenden Font Awesome Stil statt des Standards zu erzwingen.

title
[String]
Der Titel des Links, d.h. der Text für das HTML-Attribut `title`.

target
[String]
Der Wert für das HTML `target` Attribut des Link s.

cssclass
[String]
Der Wert für das HTML `class` Attribut des Links.

url
[String]
Die URL auf die der Link zeigt.

order
[Integer]
Legt die Reihenfolge fest, in der Links in einem Menü oder als Liste angezeigt werden.

start_time
[String]
Der Zeitpunkt ab dem der Link angezeigt werden soll. Wird ignoriert, wenn der Wert `None` ist.

end_time
[String]
Der Zeitpunkt bis zu dem der Link angezeigt werden soll. Wird ignoriert, wenn der Wert `None` ist.

on_app_overview_page
[Bool]
Soll der Link auf der Übersichtsseite aller Apps angezeigt werden?

on_app_detail_page
[Bool]
Soll der Link auf der Detailseite einer App angezeigt werden?

on_form_page
[Bool]
Soll der Link auf der Formularseite einer App angezeigt werden (sowohl beim Anlegen als auch beim Editieren von Datensätzen)?

on_iframe_page
[Bool]
Soll der Link auf der IFrame-Seite angezeigt werden (momentan ungenutzt)?

on_custom_overview_page
[Bool]
Soll der Link auf der eigenen Übersichtsseite angezeigt werden?

accessible
[Bool]
Ist der Link für den aktuell eingeloggten Benutzer zugänglich?

children
[Dictionary(String → MenuItem)]
Die untergeordneten Kacheln. Die Schlüssel in diesem Dictionary sind die Link-Identifizierer und die Werte sind `MenuItem`-Objekte.

c_<identifier>
[MenuItem]
Untergeordnete `MenuItem`-Objekte stehen auch über sogenannte „Shortcut"-Attribute zur Verfügung.

createdat
[Datum]
Zeitpunkt wann der Link angelegt wurde.

createby
[*User*]
Der Benutzer der diesen Link erstellt hat.

updatedat
[Datum oder `None`]
Zeitpunkt wann der Link zuletzt verändert wurde.

updateby
[*User*]
Der Benutzer der diesen Link zuletzt verändert hat.

x_<identifier>
[Objekt]
Es werden beliebige zusätzliche Attribute unterstützt deren Namen mit `x_` beginnt.

27.1.18 Panel

`Panel`-Objekte stellen zusätzliche von Ihnen konfigurierte Navigations- oder Informations-Panel dar, die Sie unter *Konfiguration* → *Erweitert* und dort bei linken Menüleiste unter *App-Panels* konfigurieren können.

`Panel`-Objekte werden verwendet, um Verweise auf einer Übersichtsseite zu bauen. Ein `Panel`-Objekt hat folgende Attribute:

id
[String]
Der interne Datenbank-Identifizierer für diesen Link.

app
[*App*]
Die App zu der dieser Link gehört.

identifier
[String]
Der Link-Identifier.

label
[String]
Der Link-Text.

parent
[*Panel* oder `None`]
Das übergeordnete Panel.

target_type
[String]
Der Typ auf den der Link verweist. Kann die Werte `newform_standalone`, `newform_embedded`, `datamanagement`, `customoverview`, `evaluation`, `import_export`, `tasks`, `formbuilder`, `workflow_manager`, `data_config`, `permissions`, `expert`, `viewtemplate`, `datamanageview` oder `custom` haben.

description
[String]
Die Beschreibung der Kachel als HTML-Text.

description_url
[String]
Die URL unter der die Panel-Beschreibung abgerufen werden kann.

icon
[String]
Der Name eines Font-Awesome Icons für den Link. Der Name kein einen der Suffixe `-brand`, `-sharp-solid`, `-solid`, `-regular`, `-light`, `-thin` oder `-duotone` haben, um den entsprechenden Font Awesome Stil statt des Standards zu erzwingen.

header_type
[String]
Gibt an wie der Panel-Kopf angezeigt werden soll. Kann die Werte `title` und `card` haben. Bei `card` ist der Kopfbereich höher.

header_background
[String oder `None`]
Gibt an welcher Hintergrund im Panel-Kopf angezeigt werden soll. Kann die Werte `uniformcolor` (d.h. eine Hintergrundfarbe), `lineargradient`, `radialgradient` oder `image` haben.

text_color
[Color oder `None`]
Die Textfarbe.

background_color1
[Color oder `None`]
Die erste der beiden Hintergrundfarben.

background_color2
[Color oder `None`]
Die zweite der beiden Hintergrundfarben.

image
[*File* oder `None`]
Ein Bild, das zu dem Link angezeigt wird.

title
[String]
Der Titel des Links, d.h. der Text für das HTML `title` Attribut.

target
[String]

Der Wert für das HTML `target` Attribut des Link s.

`cssclass`
[String]
Der Wert für das HTML `class` Attribut des Links.

`url`
[String]
Die URL auf die der Link zeigt.

`order`
[Integer]
Legt die Reihenfolge fest, in die Kacheln angezeigt werden. Wird nicht für Panels auf einer eigner Übersichtseite genutzt. Dort werden `row`, `column`, `width` und `height` stattdessen verwendet.

`row`
[String]
Die Zeile auf einer eignen Übersichtsseite in der die Kachel platziert werden soll.

`column`
[String]
Die Spalte auf einer eignen Übersichtsseite in der die Kachel platziert werden soll.

`width`
[String]
Die Breite der Kachel (d.h. die Anzahl der Gitterspalten).

`height`
[String]
Die Höhe der Kachel (d.h. die Anzahl der Gitterzeilen).

`start_time`
[String]
Der Zeitpunkt ab dem der Link angezeigt werden soll. Wird ignoriert, wenn der Wert `None` ist.

`end_time`
[String]
Der Zeitpunkt bis zu dem der Link angezeigt werden soll. Wird ignoriert, wenn der Wert `None` ist.

`on_app_overview_page`
[Bool]
Soll der Link auf der Übersichtsseite aller Apps angezeigt werden?

`on_app_detail_page`
[Bool]
Soll der Link auf der Detailseite einer App angezeigt werden?

`on_form_page`
[Bool]
Soll der Link auf der Formularseite einer App angezeigt werden (sowohl beim Anlegen als auch beim Editieren von Datensätzen)?

`on_iframe_page`
[Bool]
Soll der Link auf der IFrame-Seite angezeigt werden (momentan ungenutzt)?

`on_custom_overview_page`
[Bool]
Soll der Link auf der eigenen Übersichtsseite angezeigt werden?

`accessible`
[Bool]
Ist der Link für den aktuell eingeloggten Benutzer zugänglich?

`children`
[Dictionary(String → Panel)]

Die untergeordneten Kacheln. Die Schlüssel in diesem Dictionary sind die Link-Identifizierer und die Werte sind `Panel`-Objekte.

`c_<identifier>`
[Panel]
Untergeordnete `Panel`-Objekte stehen auch über sogenannte „Shortcut"-Attribute zur Verfügung.

`createdat`
[Datum]
Zeitpunkt wann der Link angelegt wurde.

`createby`
[*User*]
Der Benutzer der diesen Link erstellt hat.

`updatedat`
[Datum oder `None`]
Zeitpunkt wann der Link zuletzt verändert wurde.

`updateby`
[*User*]
Der Benutzer der diesen Link zuletzt verändert hat.

`x_<identifier>`
[Objekt]
Es werden beliebige zusätzliche Attribute unterstützt deren Namen mit `x_` beginnt.

27.1.19 `LookupItem`

Ein `LookupItem` stellt den Wert eines `LookupControls` dar (d.h. einer Auswahlbox oder eines Optionsfeldes) wenn nicht die Datensätze einer anderen App als Auswahlmöglichkeiten verwendet werden. Ein `LookupItem`-Objekt hat folgende Attribute:

`control`
[*Control*]
Das *Control*-Objekt zu dem dieses `LookupItem` gehört.

`key`
[String]
Die eindeutige Kennung der Auswahl-Option.

`label`
[String]
Die Beschriftung der Auswahl-Option.

`visible`
[Bool]
`False` wenn diese Option in der aktiven Formularvariante nicht angezeigt werden soll. `True` wenn sie angezeigt werden soll, oder es keine aktive Formularvariante gibt.

27.1.20 `View`

`View`-Objekte stellen eine spezielle Ansicht auf die Felder einer App dar. Im FormBuilder unter *Konfiguration* → *Eingabe* werden sie bei *Formularvarianten* angelegt.

Ein `View`-Objekt hat folgende Attribute:

`id`
[String]
Der eindeutige Datenbank-Identifizierer der Ansicht.

`name`
[String]
Der Name der Ansicht.

app
[*App*]
Die App zu der diese Ansicht gehört.

order
[Integer]
Die Reihenfolge der Ansicht, d.h. nach dieser Zahl sind die Ansichten im `App`-Attribut `views` sortiert.

lang
[String]
Die im FormBuilder konfigurierte Sprache für den View.

width
[Integer]
Die von Benutzer festgelegte Breite in Pixeln für diese Ansicht.

height
[Integer]
Die von Benutzer festgelegte Höhe in Pixeln für diese Ansicht.

start
[Datum oder `None`]
Die Anzeige einer Ansicht kann zeitlich beschränkt werden. Ist `start` nicht `None`, so wird Ansicht erst zu diesem Zeitpunkt aktiv, vorher ist sie inaktiv.

end
[Datum oder `None`]
Ist `end` nicht `None`, so wird die Ansicht zu diesem Zeitpunkt deaktiviert.

login_required
[Bool]
`login_required` ist `True` wenn im FormBuilder der Haken bei *Login wird benötigt* gesetzt ist. In dem Falle muß der Benutzer eingeloggt sein und Rechte an der App besitzen, um das Eingabeformular benutzen zu können.

result_page
[Bool]
`result_page` ist `False` wenn im FormBuilder der Haken bei *Keine Ergebnisseite anzeigen* gesetzt ist. In dem Falle wird der Benutzer nach dem Absenden des Formulars zum „Bearbeiten“-Formular weitergeleitet.

use_geo
[String]
`use_geo` legt fest, ob das Update-Template zu dieser Ansicht auf den Standort des Endgerätes zugreifen soll oder nicht. Es gibt drei mögliche Werte:

"no"
Es erfolgt kein Zugriff auf die Standort-Daten.

"once"
Es erfolgt ein einmaliger Zugriff auf die Standortdaten bei der ersten Benutzerinteraktion, oder wenn *Verwendung des aktuellen Ortes erlauben* am Fuß des Eingabeformulars angeklickt wird.

"watch"
Es erfolgt ein anfänglicher Zugriff auf die Standortdaten (wie bei `"once"`) sowie jedes Mal wenn sich der Standort ändert.

focus_control
[*Control* oder `None`]
Dasjenige `Control`-Objekt, das den Eingabefokus hat. Wird nur beim Anlegen neuer Datensätze automatisch anhand der Tab-Reihenfolge gesetzt.

focus_first_control()
[Methode() → `None`]
Setzt den Fokus auf das Eingabefeld mit dem kleinsten Tab-Index.

controls
[Dictionary(String → *ViewControl*) oder `None`]
Die View-Varianten der Controls (und zwar nur die die für diesen View konfiguriert sind). Dazu muß aber in der Datenquelle bei *Felder* mindestens *Alle Felder* gewählt werden.

c_<identifier>
[*Control*]
`Control`-Objekte stehen auch über „Shortcut"-Attribute zur Verfügung. `view.c_beispiel` ist äquivalent zu `view.controls.beispiel`.

layout_controls
[Dictionary(String → *LayoutControl*) oder `None`]
Die Layout-Controls dieses Views (falls in der Datenquelle bei *Felder Alle Felder und Layout-Felder* ausgewählt wurde oder `None` falls nicht).

lc_<identifier>
[*LayoutControl*]
„Shortcut"-Attribut zum Zugriff auf die Layout-Controls. `view.lc_beispiel` ist äquivalent zu `view.layout_controls.beispiel`.

27.1.21 ViewControl

Ein `ViewControl`-Objekt enthält *View*-spezifische Attribute eines *Control*s. Im Gegensatz zu *Control* gibt es von `ViewControl` keine Unterklassen. Daher beinhaltet ein `ViewControl`-Objekt evtl. auch Attribute, die für das zugehörige *Control*-Objekt irrelevant sind. Da über den aktiven *View* diese *View*-spezifischen Attribute auch in den *Control*-Objekten zugänglich sind, können aber prinzipiell die `ViewControl`-Objekte ignoriert werden.

Attribute sind:

id
[String]
Der eindeutige Datenbank-Identifizierer des Feldes.

label
[String]
View-spezifische Feld-Beschriftung

identifier
[String]
Der `identifier` des zugehörigen *Control*-Objekts.

view
[*View*]
Das *View*-Objekt zu dem dieses `ViewControl`-Objekt gehört.

control
[*Control*]
Das *Control*-Objekt zu dem dieses `ViewControl`-Objekt gehört.

type
[String]
Der `type` des zugehörigen *Control*-Objekts.

subtype
[String oder `None`]
Der `type` des zugehörigen *Control*-Objekts.

top
[Integer]
Vertikale Position des Eingabefeldes in diesem *View*.

left
[Integer]
Horizontale Position des Eingabefeldes in diesem *View*.

width
[Integer]
Breite des Eingabefeldes in diesem *View*.

height
[Integer]
Höhe des Eingabefeldes in diesem *View*.

z_index
[Integer]
Überlagerungs-Reihenfolge des Eingabefeldes in diesem *View*.

default
[Objekt]
Standardwert für diesen Feld in diesem *View*.

tabindex
[Integer]
Tab-Reihenfolge des Eingabefeldes in diesem *View* (zum Durchschalten der Eingabefelder mit Hilfe der Tab-Taste).

minlength
[Integer oder `None`]
Die minimale String-Länge (`None` bedeutet ohne Einschränkung). (Nur für Felder vom Typ `"string"` relevant). .. index:: maxlength, ViewControl.maxlength

maxlength
Die maximale String-Länge (`None` bedeutet ohne Einschränkung). (Nur für Felder vom Typ `"string"` relevant).

required
[Bool]
`True` wenn dieses Feld in diesem *View* als Pflichfeld konfiguriert wurde.

placeholder
Der Platzhalter für das HTML-Eingabefeld. (Nur für Felder vom Typ `"string"` relevant).

mode
[`"DISPLAY"` oder `"EDIT"`]
Kann der Wert dieses Feldes in diesem *View* bearbeitet werden (`"EDIT"`) oder wird der Wert nur angezeigt (`"DISPLAY"`)? (`"DISPLAY"` wird verwendet, wenn das Feld im FormBuilder mit *Inhalt zur zum Lesen anzeigen* konfiguriert wurde.

labelpos
[`"TOP"`, `"LEFT"`, `"RIGHT"`, `"BOTTOM"` oder `None`]
Position der Feldbeschriftung relativ zum Eingabefeld in diesem *View*. Bei `None` wird die Beschriftung gar nicht angezeigt.

lookup_none_key
[String oder `None`]
Ist `none_key` nicht `None` wird im Eingabeformular eine „Nichts ausgewählt"-Option angeboten. Der Wert von `none_key` wird als Wert dieser Auswahl-Option verwendet. Ist `none_key` `None` wird keine „Nichts ausgewählt"-Option angezeigt.

(Nur für Felder vom Typ `"lookup"`, `"multiplelookup"`, `"applookup"` oder `"multipleapplookup"`).

lookup_none_label
[String oder `None`]
Wird als Beschriftung der „Nichts ausgewählt"-Option verwendet, wenn diese „Nichts ausgewählt"-Option angeboten wird. Ist `none_label` `None` sollte eine generische Beschriftung verwendet werden.

(Nur für Felder vom Typ `"lookup"`, `"multiplelookup"`, `"applookup"` oder `"multipleapplookup"`).

lookupdata
[Dictionary(String → *ViewLookupItem*)]

`lookupdata` beinhaltet die Auswahlmöglichkeiten als Dictionary (in der Reihenfolge wie die Optionen vom Benutzer angelegt wurde). Die Schlüssel sind die Identifierer der Option und die Werte sind *ViewLookupItem*-Objekt.

`labelwidth`
[Integer]
Breite des Labels in diesem *View*.

`autoalign`
[Bool]
`True` wenn die Breite des Labels automatisch vom FormBuilder berechnet wird. `False` wenn sie der Benutzer selbst angepasst hat.

`autoexpandable`
[Bool]
`True` wenn in FormBuilder für das Feld *Auto-Hinzufügen aktivieren* konfiguriert wurde.

`x_<identifier>`
[Objekt]
Es werden beliebige zusätzliche Attribute unterstützt deren Namen mit `x_` beginnt.

27.1.22 `ViewLookupItem`

Ein `ViewLookupItem` stellt den *View*-spezifischen Wert eines `LookupControls` dar (d.h. einer Auswahlbox oder eines Optionsfeldes) wenn nicht die Datensätze einer anderen App als Auswahlmöglichkeiten verwendet werden. Ein `ViewLookupItem`-Objekt hat folgende Attribute:

`key`
[String]
Die eindeutige Kennung der Auswahl-Option.

`label`
[String]
Die Beschriftung der Auswahl-Option.

`visible`
[Bool]
`False` wenn diese Option in diesem *View* nicht angezeigt werden soll. `True` wenn sie angezeigt werden soll.

> **Bemerkung**
>
> `label` und `visible` können sich von *View* zu *View* unterscheiden.)

27.1.23 `Attachment`

Ein `Attachment`-Objekt beinhaltet die Daten zu einem Anhang. Genauer gesagt gibt es vier verschiedene Typen von Anhängen: `NoteAttachment`, `URLAttachment`, `FileAttachment` und `JSONAttachment`. Gemeinsam sind ihnen alle folgende Attribute:

`id`
[String]
Der interne Datenbank-Identifizierer für diesen Anhang.

`type`
[String]
Der Typ des Anhangs, einer der Werte `"note"`, `"url"`, `"file"`, `"image"` oder `"json"`.

`record`
[*Record*]
Der Datensatz zu dem dieser Anhang gehört.

label
[String]
Die Beschriftung des Anhangs.

active
[Bool]
Ist diser Anhang aktiv oder nicht?

Weiterhin haben `NoteAttachment` und `URLAttachment` ein Attribut `value`, das die Notiz bzw. die URL als String beinhaltet. Bei `JSONAttachment` ist das `value`-Attribut das JSON-Objekt. Bei `FileAttachment` ist das `value`-Attribut ein *File*-Objekt.

27.1.24 User

Ein `User`-Objekt stellt einen LivingApps-Benutzeraccount dar und hat folgende Attribute:

id
[String]
Der eindeutige Datenbank-Identifizierer des Benutzers.

gender
[String]
Das Geschlecht des Benutzers (`"m"` oder `"f"`).

firstname
[String]
Der Vorname des Benutzers.

surname
[String]
Der Nachname des Benutzers.

initials
[String]
Die Initialen des Benutzers.

email
[String]
Die Email-Adresse des Benutzers (gleichzeitig auch der Accountname).

lang
[String]
Die bevorzugte Sprache des Benutzers (beispielsweise `"de"` für Deutsch und `"en"` für Englisch).

image
[*File* oder `None`]
Das Benutzer-Portrait in Original-Größe.

Wird das Portrait in einer anderen Größe benötigt, kann *scaled_url(...)* benutzt werden.

keyviews
[Dictionary(String → *KeyView*) oder `None`]
Dieses Attribut ist nur dann nicht `None`, wenn es sich bei diesem Benutzer um den eingeloggten Benutzer handelt. In dieser Fall ist `keyviews` ein Dictionary das die „Key-Views“ dieses Users enthält, d.h. die Template-Ansichten die für diesen Benutzer ohne Login zur Verfügung stehen. Die Schlüssel sind die Identifizierer der jeweiligen Ansicht und die Werte sind die entsprechenden `KeyView`-Objekte.

streetname
[String]
Der Straßenname.

streetnumber
[String]
Die Hausnummer.

zip
[String]
Die Postleitzahl.

city
[String]
Die Stadt.

phone
[String]
Die Telefonnummer.

fax
[String]
Die Faxnummer.

summary
[String]
Selbstbeschreibung des Benutzers.

interests
[String]
Die Interessen.

personal_website
[String]
Die URL der persönlichen Website.

company_website
[String]
Die URl der Unternehmens-Website.

company
[String]
Der Firmenname.

position
[String]
Die Position innerhalb der Firma.

department
[String]
Die Abteilung innerhalb der Firma.

change(oldpassword, newpassword, newemail)
[Methode(String, String, String) → Liste(String)]
Mit der Methode `change` können Passwort und E-Mail-Adresse geändert werden. Die Angabe von `oldpassword` ist verpflichtend. Wenn ein `new`-Parameter `None` ist, wird der Wert auch nicht geändert. Die zurückgegebene Liste enthält die Meldungen zu den aufgetretenen Fehlern.

27.1.25 File

`File`-Objekte werden verwendet für Icons, Bilder und andere Dateien (also z.B. für die Icons der *App* und für Datei-Upload-Felder). Ein Datei-Objekt hat folgende Attribute:

url
[String]
Die absolute URL unter der die Datei abrufbar ist.

archive_url
[String]
Wenn sich diese Datei in einem ZIP-Archiv befindet kann diese URL benutzt werden, um dann mittels relativer URLs in dieser Datei auf andere Dateien im ZIP-Archiv zuzugreifen. Normalerweise trifft dies nur für Dateien zu, die LivingApps selbst verwendet. Ist diese Datei nicht Bestandteil eines ZIP-Archivs, hat `archive_url` den selben Wert wie `url`.

filename
[String]
Der ursprüngliche Dateiname der Datei.

mimetype
[String]
Der MIME-Typ der Datei (also z.B. `image/jpeg` für JPEG-Bilder oder `application/pdf` für PDF-Dateien)

size
[Integer]
Die Größe der Datei in Bytes.

width
[Integer oder `None`]
Falls es sich bei der Datei um ein Bild handelt, ist `width` die Breite des Bildes in Pixeln (ansonsten `None`).

height
[Integer oder `None`]
Die Höhe der Bilder in Pixeln (oder `None` falls es sich nicht um ein Bild handelt).

duration
[Integer oder `None`]
Wenn ich sich bei der Datei um ein Video handelt, ist `duration` die Dauer des Videos in Millisekunden (und `None` für andere Datei-Typen).

geo
[*Geo* oder `None`]
Enthält die Datei Geo-Daten (z.B. den Ort an dem das Bild oder Video aufgenommen wurde), so werden diese aus der Datei extrahiert und im Attribut `geo` gespeichert.

Damit jedoch Geo-Daten tatsächlich extrahiert werden, muß für die App bei der die Datei hochgeladen wird, dies erst aktiviert werden. Dies ist unter *Konfiguration* → *Erweitert* unter *Uploads* möglich (bzw. unter *Meine LivingApps* → *LivingApps* bei der jeweiligen App).

recordedat
[Datum]
Der Zeitpunkt zu dem diese Datei erzeugt wurde (wird aus den EXIF-Daten gelesen, falls vorhanden).

archive
[`File` oder `None`]
Ist diese Datei Bestandteil eines ZIP-Archivs, verweißt `archive` auf dieses Archiv, ansonsten ist `archive` `None`.

createdat
[Datum]
Der Zeitpunkt zu dem diese Datei angelegt wurde.

27.1.26 Geo

`Geo`-Objekte stellen einen geographische Ort auf der Erdoberfläche dar und werden als Werte von `GeoControl`-Felder verwendet. Ein `Geo`-Objekt hat folgende Attribute:

lat
[Zahl]
Der Breitengrad. Orte mit positivem Breitengrad befinden sich nördlich des Äquators, solche mit negativem Breitengrad südlich.

long
[Zahl]
Der Längengrad. Orte mit positivem Längengrad befinden sich östlich des Nullmeridians, solche mit negativem Längengrad westlich.

info
[String oder `None`]
Eine Beschreibung des durch `lat` und `long` bestimmten Ortes, z.B. eine Adresse.

> **Bemerkung**
>
> Wenn `Geo`-Objekte dargestellt werden, muss Open Streetmap als Datenquelle genannt werden. Mehr Informationen zur Nennung[14] und zur Lizenz[15].

27.1.27 `Installation`

Ein `Installation`-Objekt beschreibt einen Installationsvorgang bei dem eine App installiert wurde. Diese Information ist im *App*-Attribut `installation` zu finden.

Ein `Installation`-Objekt hat folgende Attribute:

id
[String]
Der interne Datenbank-Identifizierer für diese Installation.

name
[String]
Der Name der Installation.

27.1.28 `Category`

Ein `Category`-Objekt stellt eine Kategorie dar, der eine App zugeordnet ist. Dabei kann eine App mehreren Kategorien zugeordnet werden. Diese Zuordnung ist im *App*-Attribut `categories` zu finden.

Ein `Category`-Objekt hat folgende Attribute:

id
[String]
Der interne Datenbank-Identifizierer für diese Kategorie.

identifier
[String]
Der vom Benutzer vergebene Identifizierer für diese Kategorie. Dieser Identifizierer ist eindeutig innerhalb der „Geschwister"-Kategorien dieser Kategorie.

name
[String]
Der Name der Kategorie.

order
[Integer]
Die Reihenfolge, d.h. nach dieser Zahl sind die Kategorien innerhalb der übergeordneten Kategorie sortiert.

parent
[*Category* oder `None`]
Die übergeordnete Kategorie (bzw. `None` falls sich diese Kategorie bereits auf oberster Ebene befindet).

children
[Dictionary(String → *Category*) oder `None`]
Wenn in der Datenquellen-Konfiguration *Kategorien* die Option *Kategorien-Pfade* ausgewählt, so ist `children` `None`. D.h. über die den Apps zugeordneten Kategorien können über das `parent`-Attribut die Pfade zu diesen Kategorie im Kategorien-Baum rekonstriert werden, aber nicht die Bäume selbst.

Dazu wird das `children`-Attribut benötigt, das aber nur gefüllt wird, wenn in der Datenquellen-Konfiguration *Kategorien* entweder *Kategorien-Bäume* oder *Kategorien-Bäume mit Apps* ausgewählt wurde.

[14] https://www.openstreetmap.de/faq.html#lizenz
[15] https://opendatacommons.org/licenses/odbl/summary/

Dann ist `children` ein sortiertes Dictionary. Die Schlüssel sind die internen Datenbank-Identifizierer der untergeordneten Kategorie und die Werte sind die entsprechende `Category`-Objekte.

apps

[Dictionary(String → *App*) oder `None`]
`apps` beinhaltet die dieser Kategorie zugeordneten Apps, wenn in der Datenquellen-Konfiguration *Kategorien* die Option *Kategorien-Bäume mit Apps* ausgewählt ist. `apps` ist ein sortiertes Dictionary. Die Schlüssel sind die internen Datenbank-Identifizierer der jeweiligen App und die Werte sind die entsprechenden `App`-Objekte.

Ist in der Datenquellen-Konfiguration *Kategorien* die Option *Kategorien-Bäume mit Apps* nicht ausgewählt, ist `apps` `None`.

27.1.29 KeyView

`KeyView`-Objekte werden dazu benutzt, um einem Benutzer den Zugriff auf bestimmte Templates zu ermöglichen, ohne dass dieser Benutzer sich im LivingApps-System einloggen muß. Die einzige Voraussetzung ist, dass der Benutzer den ihm zugewiesenen Schlüssel (`key`) kennt.

`KeyView`-Objekte haben folgende Attribute:

id

[String]
Der interne Datenbank-Identifizierer für diese Kategorie.

identifier

[String]
Der vom Benutzer vergebene Identifizierer für diesen `KeyView`. Dieser Identifizierer ist eindeutig für diesen Benutzer.

name

[String]
Der Name für diesen `KeyView`, der z.B. dem Benutzer zur Navigation angezeigt werden kann.

key

[String]
Der Schlüssel-Wert (eine sechzigstellige Zeichenkette), die als Bestandteil der URL Zugriff auf das Template gewährt. Ist dieser Wert also z.B. `key`, ist unter der URL

http://my.living-apps.de/gateway/keyviews/key

die zugehörige Ansicht verfügbar.

user

[*User*]
Der Benutzer, dem dieser `KeyView` gehört. Mit den Rechten dieses Benutzers wird das Template aufgerufen.

27.1.30 ViewTemplateInfo

`ViewTemplateInfo`-Objekte werden dazu benutzt, um Informationen über Anzeige-Templates zur Verfügung zu stellen.

`ViewTemplateInfo`-Objekte haben folgende Attribute:

id

[String]
Der interne Datenbank-Identifizierer für dieses Anzeige-Template.

app

[*App*]
Die App zu der dieses Anzeige-Template gehört.

identifier

[String]

Der vom Benutzer vergebene Identifizierer für dieses Anzeige-Template. Dieser Identifizierer ist eindeutig für diese App.

name
[String]
Der Name dieses Anzeige-Templates.

icon
[String]
Das Icon für dieses Anzeige-Template.

type
[String]
Der Typ dieses Anzeige-Templates. Mögliche Werte sind: `list`, `listdatamanagement`, `listdefault`, `detail`, `detailresult`, `detaildatamanagement` oder `support`.

mimetype
[String]
Der Mime-Type dieses Anzeige-Templates.

permission_level
[String]
Die Zugriffsberechtigung dieses Anzeige-Templates. Mögliche Werte: `all`, `loggedin`, `app`, `appedit`, `appadmin`.

27.2 Sonstiges

27.2.1 Variablen

Einem Anzeige- oder E-Mail-Template werden folgende Variablen übergeben:

globals
[*Globals*]
Globale Informationen

datasources
[Dictionary(String → *DataSource*)]
`datasources` beinhaltet die konfigurierten Datenquellen. Die Schlüssel sind die Identifizierer der Datenquellen und die Werte sind `DataSource`-Objekte.

record
[*Record*]
Einen Detail-Template (d.h. einem *Anzeige-Template*, bei dem unter *Typ Detail* ausgewählt wurde), wird zusätzlich noch der Datensatz übergeben.

Ebenso wird bei einen *E-Mail-Template* der Datensatz, der die E-Mail-Versendung auslöst, übergeben.

Bei anderen Templates ist diese Variable `None`.

app
[*App*]
Die App zu der dieses Anzeige- oder E-Mail-Template gehört.

27.2.2 HTTP-Request-Parameter

Einige HTTP-Request-Parameter werden von der LivingAPI selbst verwendet zur Steuerung welches Anzeige-Template zur Datenanzeige verwendet wird, und welche Daten diesem Anzeige-Template zur Verfügung gestellt werden.

Alle Parameter-Namen die mit `la-` beginnen sind für die LivingAPI reserviert und sollten nicht für andere Zwecke verwendet werden.

Template

Der Request-Parameter `template` bestimmt welches Anzeige-Template verwendet wird. D.h. der Parameter-Wert muß der *Identifizierer* eines Anzeige-Templates sein. Ist der Parameter `template` nicht angegeben wird das Standard-Anzeige-Template verwendet (d.h. das Template dessen *Standard*-Haken gesetzt ist). Gibt es kein Standard-Anzeige-Template erhalten Sie einen 404-HTTP-Fehler.

Paging

Wenn eine Datenquelle so konfiguriert ist, daß die Datensätze dieser App einem Anzeige-Template zur Verfügung gestellt werden, kann es vorkommen, daß die Anzahl der Datensätze zu groß ist um sie auf einer HTML-Seite anzuzeigen. In diesem Fall können Sie durch Request-Parameter festlegen, daß für diese Datenquelle nur ein Teil der Datensätze zur Verfügung gestellt wird. Wenn Sie z.B. eine Datenquelle mit dem Identifizierer `personen` haben, können Sie mit folgendem Request-Parameter festlegen, daß Sie nur die ersten 20 Datensätze haben wollen:

```
la-ds-personen-paging=0_20
```

`0_20` bedeutet dabei, daß beginnend beim ersten Datensatz (also mit dem Index 0) 20 Datensätze ausgeliefert werden sollen.

Dieselbe Funktionalität steht für untergeordnete Datensätze zur Verfügung. Wenn z.B. bei obiger Datenquelle `personen` noch eine Konfiguration für untergeordnete Datensätze mit den Identifizierer `bestellungen` existiert, können Sie folgenden Request-Parameter verwenden:

```
la-dsc-personen-1234567890abcdef12345678-bestellungen-paging=20_10
```

Damit werden dann bei dem Personen-Datensatz mit der Id `1234567890abcdef12345678` aus der Datenquelle `personen` von den untergeordneten `bestellungen`-Datensätzen nur der 20.-29. ausgeliefert. Ist der Parameter für diesen Datensatz nicht angegeben, wird zusätzlich noch der Parameter

```
la-dsc-personen-bestellungen-paging
```

verwendet (d.h. dieser Parameter kann verwendet werden um ein Default-Paging für die untergeordneten BEstellungs-Datensätze *aller* Personen-Datensätze anzugeben).

27.2.3 E-Mails über die LivingAPI versenden

Über die beiden Methoden *App.send_mail()* und *Record.send_mail()* können Sie direkt E-Mails versenden.

Wenn Sie aus Ihren LivingApps heraus E-Mail versenden, sind diese immer der jeweiligen LivingApp zugeordnet, und damit unter *Versendete E-Mails* zu finden. Die trifft auch auf die E-Mails zu, die sie über diese beiden Methoden versenden. Beim Versand über `Record.send_mail()` wird die versendete E-Mail jedoch auch noch dem Datensatz zugeordnet, für den die E-Mail versendet wurde.

Bei Aufruf dieser Methoden müssen alle Argumente per Keyword übergeben werden. Argumente die E-Mail-Adressen sind unterstützen als Format sowohl eine reine E-Mail-Adresse als auch eine E-Mail-Adresse mit Anzeige-Name, also entweder

```
name@example.org
```

oder

```
Vorname Nachname <name@example.org>
```

Folgende Argumente werden unterstützt:

`from`: String oder `None`

Absender der E-Mail.

Dieses Argument gibt den Absender der E-Mail an. Es enthält die E-Mail-Adresse der Person oder Organisation, die die E-Mail sendet. Dieses Argument zeigt an, wer die Nachricht verfasst hat.

Es wird jedoch nur der Anzeige-Name dieser E-Mail-Adresse verwendet, die E-Mail-Adresse selber ist jedoch immer `benachrichtigung@living-apps.de`. D.h. wenn Sie hier als Absender-E-Mail-Adresse `Vorname Nachname <name@example.org>` eingeben, wird die E-Mail von der Absender-E-Mail-Adresse `Vorname Nachname <benachrichtigung@living-apps.de>` versendet werden. Wird dieser Parameter nicht übergeben (oder `None`) wird `LivingApps-Benachrichtigungssystem <benachrichtigung@living-apps.de>` verwendet.

`reply_to`: String oder `None`

E-Mail-Adresse für Antworten (falls abweichend).

Dieses Argument gibt die E-Mail-Adresse an, an die Antworten gesendet werden sollen. Dies kann eine andere Adresse als die im `from`-Argument sein.

Sie können es verwenden, wenn Sie möchten, daß Antworten an eine andere E-Mail-Adresse gesendet werden. (Was normalerweise immer der Fall ist, da die Absender-Adresse ja immer `benachrichtigung@living-apps.de` ist).

`to`: String oder `None`

Hauptempfänger.

Dieses Argument enthält die E-Mail-Adresse(n) der Hauptempfänger der E-Mail. Dies sind die Personen, an die die E-Mail direkt gerichtet ist. Es ist auch möglich mehrere E-Mail-Adressen anzugeben, diese müssen dann durch Kommas getrennt angegeben werden.

Wird `None` übergeben, so wird die E-Mail nicht versendet.

`cc`: String oder `None`

Weitere Empfänger, für alle sichtbar.

Dieses Argument wird verwendet, um andere Personen zu informieren, die die E-Mail erhalten sollen, aber nicht die Hauptempfänger sind. Die im `cc`-Argument übergebenen Adressen sind für alle Empfänger der E-Mail sichtbar. `cc` wird häufig genutzt, um Personen in den Verteiler aufzunehmen, die die Information erhalten sollen, ohne direkt involviert zu sein.

Auch hier können möglich mehrere E-Mail-Adressen durch Kommas getrennt übergeben werden.

`bcc`: String oder `None`

Weitere Empfänger, für andere unsichtbar.

Dieses Argument erlaubt es, E-Mails an Empfänger zu senden, ohne daß die anderen Empfänger diese Adressen sehen können. Die im `bcc`-Argument übergebenen E-Mail-Adressen sind für andere Empfänger unsichtbar. Dies ist nützlich, wenn man eine E-Mail an eine große Gruppe senden möchte, ohne daß die E-Mail-Adressen der Empfänger für alle sichtbar sind.

Auch hier können möglich mehrere E-Mail-Adressen durch Kommas getrennt übergeben werden.

`subject`: String oder `None`

Der Betreff der E-Mail.

Der Betreff sollte eine kurze Bechreibung oder Zusammenfassung des Inhalts der E-Mail enthalten. Er sollte kurz und prägnant sein, und dem Empfänger eine Vorstellung davon geben, worum es in der E-Mail geht (und dies idealerweise auch noch Wochen oder Monate später).

Ein gut formulierter Betreff hilft, die E-Mail effizient zu organisieren und priorisieren, sowohl für den Absender als auch für den Empfänger.

`body_text`: String oder `None`

Der Inhalt der E-Mail als Text.

Diese ist der Hauptteil einer E-Mail, der die eigentliche Nachricht enthält.

Ist dieses Argument nicht angegeben oder `None` wird eine Text-Version automatisch aus `body_html` erzeugt.

Sie sollten jedoch bei `body_text` immer eine Text-Version des Nachrichten-Inhalts übergeben, da die automatisch erzeugte Version möglicherweise nicht optimal formatiert ist, und Empfänger aus Sicherheitsgründen die Anzeige der HTML-Version unterbunden haben.

`body_html`: String oder `None`

Der Inhalt der E-Mail als HTML.

Mit der HTML-Version können Sie eine ansprechend formattierte Version der Nachricht erzeugten, und Bilder, Links und andere Design-Elemente verwenden.

`body_html` sollte dabei eine vollständige HTML-Seite (d.h. inklusive `<html>`- und `<head>`-Element) sein. Dies ermöglicht es Ihnen auch CSS-Stylesheets einzubinden.

Um die Darstellung in E-Mail-Programmen zu verbessern und zu verhindern, daß das E-Mail-Programm Ihre CSS-Angaben ignoriert wird vor der Versendung der E-Mail der Inhalt der CSS-Stylesheets in *style*-Attribute verlagert.

`attachments`: *File* oder `None`

Eine Datei, die zusammen mit der E-Mail gesendet wird.

E-Mail-Anhänge sind ein nützliches Werkzeug, um zusätzliche Informationen, Dokumente und Dateien effizient und direkt zu kommunizieren. Bei größeren Dateien sollten Sie jedoch dem Empfänger besser über ein Anzeige-Template die Möglichkeit geben, diese Datei direkt herunterzuladen.

KAPITEL 28

Statische Resourcen

Das Einbinden von externen Inhalten ist bei *Externe Pakete* dokumentiert.

Wenn das LivingApps-Logo oder ein Platzhalter-Icon für eine App oder einen User eingebunden werden sollen, kann das mittels einer der folgenden relativen URLs geschehen:

```
/static/LivingApps/0.1/livingapps-logo.svg
/static/LivingApps/0.1/app-placeholder.jpg
/static/LivingApps/0.1/user-placeholder.jpg
```

KAPITEL 29

Bilderskalierung

Über die URLs https://my.living-apps.de/imgproxycache/insecure und https://my.living-apps.de/imgproxy/insecure können skaliere Bilder erzeugt werden.

Näheres dazu finden Sie unter *scaled_url(…)*.

KAPITEL 30

QR-Codes

Über die URL https://my.living-apps.de/qr/generate kann ein QR-Code erzeugt werden. Es müssen folgende Parameter übergeben werden:

data
: Der durch den QR-Code kodierte Text. Normalerweise ist dies eine URL.

size
: Die Breite und Höhe des Bildes.

Um z.B. einen QR-Code für die URL https://my.living-apps.de/ in der Größe 300×300 Pixel zu erzeugen, sieht die URL folgendermaßen aus:

```
https://my.living-apps.de/qr/generate?size=300&data=https%3A%2F%2Fmy.living-apps.de%2F
```

Da die Zeichen `:` und `/` in URLs ein spezielle Bedeutung haben, müssen sie mittels der `%`-Notation ersetzt werden. Wenn Sie so eine URL in einem Anzeige-Template erzeugen wollen, können Sie für diese Ersetzung durch die UL4-Funktion `urlquote()` verwenden. D.h. in einem Anzeige-Template könnten Sie obige URL folgendermaßen erzeugen:

```
<?code url = "https://my.living-apps.de/"?>
<?code size = 300?>

https://my.living-apps.de/qr/generate?data=<?print urlquote(url)?>&size=<?print size?>
```

D.h. wenn Sie in einen HTML-Anzeige-Template ein Bild mit diesem QR-Code verwenden wollen, geht das folgendermaßen:

```
<?code url = "https://my.living-apps.de/"?>
<?code size = 300?>

<img src="https://my.living-apps.de/qr/generate?data=<?printx urlquote(url)?>&size=<?
↪printx size?>" width="<?printx size?>" height="<?printx size?>" />
```

Diese Arbeit nimmt Ihnen die Funktion `globals.qrcode_url()` ab, sodaß sie obiges Bild auch folgendermaßen erzeugen können:

```
<?code url = "https://my.living-apps.de/"?>
<?code size = 300?>
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
<img src="<?printx globals.qrcode_url(url, size=size)?>" width="<?printx size?>"␣
↪height="<?printx size?>" />
```

KAPITEL 31

Externe Pakete

Sie können auf alle NPM-Pakete die auf https://www.npmjs.com/ zu finden sind unter einer LivingApps-URL zugreifen.

https://my.living-apps.de/npm/ ist ein Cache für https://cdn.jsdelivr.net/npm/. Eine Beschreibung dieses Services finden Sie unter https://www.jsdelivr.com/. Wenn Sie in den dortigen Beispielen `cdn.jsdelivr.net` durch `my.living-apps.de` ersetzen, erhalten Sie funktionierende URLs, also z.B.:

```
https://my.living-apps.de/npm/jquery@3.6.4/dist/jquery.min.js
```

Bemerkung

Sie sollten bei diesen URL immer eine vollständige Versionsnummer verwenden (also in obigem Beispiel `3.6.4`), da sich bei anderen URLs (also z.B. `https://my.living-apps.de/npm/jquery@3.6/dist/jquery.min.js` oder gar `https://my.living-apps.de/npm/jquery@latest/dist/jquery.min.js`) jederzeit die dahinter liegende Versionsnummer verändern kann, sie aber vom Cache noch einige Zeit die alte Version erhalten würden.

KAPITEL 32

Programmierschnittstellen

LivingApps bietet einige Programmierschnittstellen, mit deren Hilfe Sie mit Ihren Daten in LivingApps arbeiten können.

32.1 REST-Service

Unter der URL `https://my.living-apps.de/rest/` ist eine REST-Schnittstelle zugänglich, die Ihnen über HTTP ermöglicht, Daten aus Ihren LivingApps abzurufen, neue Datensätze zu erzeugen, sowie vorhandene Datensätze zu ändern, oder zu löschen.

32.1.1 Autorisierung

Um die REST-Schnittstelle benutzen zu können, benötigen Sie einen Login-Token, den Sie unter *Konfiguration* → *Erweitert* bei *Account* → *Login-Token* anlegen können.

Dieser Login-Token gewährt Ihren HTTP-Anfragen auf die REST-Schnittstelle dieselben Zugriffsrechte, die Sie als Benutzer auch haben.

In obigem Beispiel ist der Login-Token folgender Text:

```
6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeWI
```

Diesen Login-Token müssen Sie bei allen HTTP-Anfragen an die REST-Schnittstelle im `Authorization`-Header mitgeben und zwar folgendermaßen:

```
Authorization: Bearer␣
↪6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeWI
```

D.h. wenn Sie die Liste Ihrer LivingApps abrufen wollen, geht das beispielsweise mit `curl` folgendermaßen:

```
$ curl -s -X 'GET' \
  'https://my.living-apps.de/rest/apps' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer␣
↪6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeWI
↪'
```

Falls Sie `requests` in Python verwenden, geht das folgendermaßen:

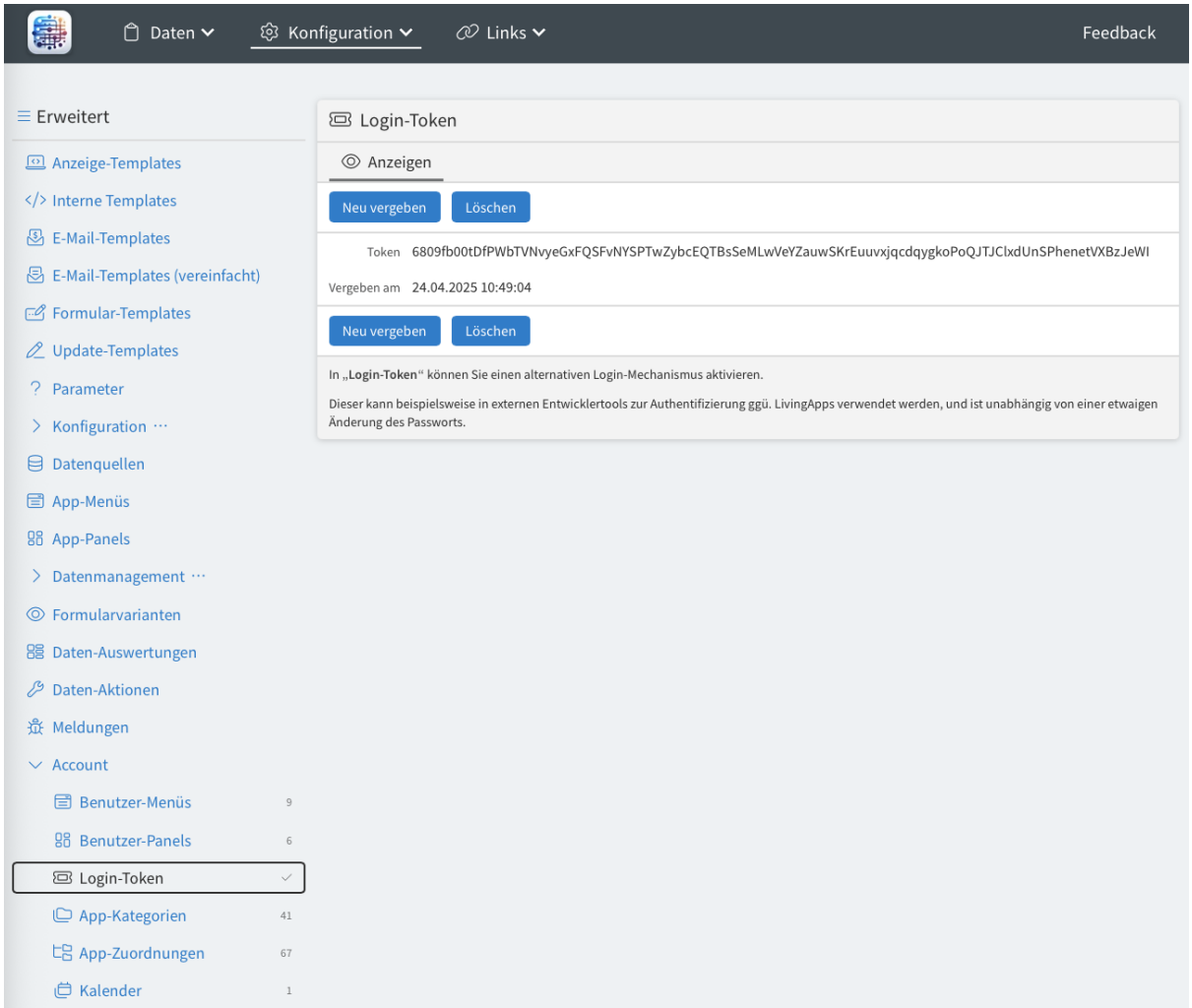

Abb. 1: Login-Token anlegen, neu vergeben, oder löschen

```
Python 3.13.2 (main, Feb  4 2025, 14:51:09) [Clang 16.0.0 (clang-1600.0.26.6)] on␣
↪darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps",
...     headers={
...         "Accept": "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
... )
>>> response.json()
...
```

32.1.2 Verfügbare Endpoints

GET https://my.living-apps.de/rest/apps
Gibt Informationen über alle Ihre LivingApps zurück.

GET https://my.living-apps.de/rest/apps/{app_id}
Gibt Informationen über die LivingApp mit der ID `{app_id}` zurück.

GET https://my.living-apps.de/rest/apps/{app_id}/records
Gibt Informationen über die Datensätze der LivingApp mit der ID `{app_id}` zurück.

POST https://my.living-apps.de/rest/apps/{app_id}/records
Legt in der LivingApp mit der ID `{app_id}` einen neuen Datensatz an.

GET https://my.living-apps.de/rest/apps/{app_id}/records/{record_id}
Gibt Informationen über den Datensatz mit der ID `{record_id}` in der LivingApp mit der ID `{app_id}` zurück.

PUT https://my.living-apps.de/rest/apps/{app_id}/records/{record_id}
Ändert den Datensatz mit der ID `{record_id}` in der LivingApp mit der ID `{app_id}`. Nicht übergebene Feldwerte werden auf „leer“ gesetzt.

PATCH https://my.living-apps.de/rest/apps/{app_id}/records/{record_id}
Ändert den Datensatz mit der ID `{record_id}` in der LivingApp mit der ID `{app_id}`. Nicht übergebene Feldwerte werden nicht geändert.

DELETE https://my.living-apps.de/rest/apps/{app_id}/records/{record_id}
Löscht den Datensatz mit der ID `{record_id}` in der LivingApp mit der ID `{app_id}`.

POST https://my.living-apps.de/rest/files
Löscht den Datensatz mit der ID `{record_id}` in der LivingApp mit der ID `{app_id}`.

Unter `https://my.living-apps.de/rest/docs` finden Sie eine Liste dieser verfügbaren Endpoints und eine Erklärung der unterstützen Parameter.

32.1.3 Fehlermeldungen

Folgende Fehlermeldungen können bei der Verwendung des REST-Services auftreten:

„Bearer“-Token ungültig (`bearer-token-invalid`)

Der im `Authorization`-Header bei der HTTP-Abfrage übergebene Login-Token ist ungültig.

Beispiel:

```
>>> import pprint, requests
>>> login_token = "falsch"
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps",
...     headers={
...         "Accept": "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
... )
>>> pprint.pprint(response.json())
{'detail': 'Bearer token is invalid.',
 'status': 401,
 'title': 'Unauthorized access',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪bearer-token-invalid'}
```

„Basic“-Autorisierung ungültig (`basic-auth-invalid`)

Es ist auch möglich sich mittels „Basic“-Autorisierung (also über Benutzername und Passwort) zu autorisieren. Dieser Fehler tritt auf, wenn Benutzername oder Passwort ungültig sind.

Beispiel:

```
>>> import pprint, requests
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps",
...     headers={
...         "Accept": "application/json",
...     },
...     auth=("user@example.org", "falsch"),
... )
>>> pprint.pprint(response.json())
{'detail': 'Basic authentication credentials are invalid.',
 'status': 401,
 'title': 'Unauthorized access',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪bascic-auth-ivalid'}
```

`Authorization`-Header fehlt (`authorization-missing`)

Der `Authorization`-Header wurde bei der HTTP-Abfrage gar nicht mitgesendet.

Beispiel:

```
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps",
...     headers={
...         "Accept": "application/json",
...     },
... )
>>> pprint.pprint(response.json())
{'detail': 'Header `Authorization` is missing.',
 'status': 401,
 'title': 'Unauthorized access',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪authorization-missing'}
```

Autorisierung ungültig (`authorization-invalid`)

Der bei der HTTP-Abfrage gesendete `Authorization`-Header enthält weder eine „Basic“-Autorisierung noch eine „Bearer“-Autorisierung.

Beispiel:

```
>>> import pprint, requests
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps",
...     headers={
...         "Accept": "application/json",
...         "Authorization": "falsch",
...     },
... )
>>> pprint.pprint(response.json())
{'detail': 'Header `Authorization` contains neither `Basic` nor `Bearer`.',
 'status': 401,
 'title': 'Unauthorized access',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪authorization-invalid'}
```

LivingApp unbekannt (`unknown-app`)

Die bei der HTTP-Abfrage gesendete LivingApp-ID ist ungültig.

Beispiel:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps/000000000000000000000000",
...     headers={
...         "Accept": "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
... )
>>> pprint.pprint(response.json())
{'app_id': '000000000000000000000000',
 'detail': 'The LivingApp specified via `app_id` does not exist.',
 'status': 404,
 'title': 'LivingApp not found',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪unknown-app'}
```

Keine Zugriffrechte auf LivingApp (`no-app-permission`)

Der bei der HTTP-Abfrage übergebene Benutzer hat nicht die nötigen Zugriffrechte auf der spezifizierten LivingApp.

Beispiel:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps/680a201a8f45887b855d1b99",
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
...     headers={
...         "Accept": "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
... )
>>> pprint.pprint(response.json())
{'app_id': '680a201a8f45887b855d1b99',
 'detail': 'You don't have permission to access the LivingApp specified via `app_id`.
↪',
 'status': 403,
 'title': 'LivingApp not accessible',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪no-app-permission'}
```

Keine Rechte zum Erzeugen einer LivingApp (`no-permission-app-creation`)

Der bei der HTTP-Abfrage übergebene Benutzer hat nicht die nötigen Rechte, um einen neue LivingApp anzulegen.

Beispiel:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.post(
...     "https://my.living-apps.de/rest/apps",
...     headers={
...         "Accept": "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
...     json={
...         "name": "My App",
...         "description": "My new App",
...         "gramgen": "m",
...         "typename_nom_sin": "Test",
...         "typename_gen_sin": "Test",
...         "typename_dat_sin": "Test",
...         "typename_acc_sin": "Test",
...         "typename_nom_plu": "Test",
...         "typename_gen_plu": "Test",
...         "typename_dat_plu": "Test",
...         "typename_acc_plu": "Test",
...         "controls": {
...             "my_string": {"label": "My_string", "fulltype": "string/text",
↪"description": "My description"},
...             "my_number": {"label": "My Number", "fulltype": "number", "required":␣
↪True},
...         },
...     }
... )
>>> pprint.pprint(response.json())
{'detail': 'You don't have permission to create a new LivingApp.',
 'status': 403,
 'title': 'No permission to create a new LivingApp',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la_erw_rest_service_error_
↪no_permission_app_creation'}
```

Datensatz nicht gefunden (`record-not-found`)

Der Datensatz mit der bei der HTTP-Abfrage übergebenen ID existiert nicht in der spezifizierten LivingApp (oder Sie haben in dieser LivingApp nur das Recht, die Ihnen zugewiesenen Datensätze zu sehen, und der spezifizierte Datensatz gehört nicht dazu).

Beispiel:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps/680a23234731c68fdf429d23/records/
↪000000000000000000000000",
...     headers={
...         "Accept": "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
... )
>>> pprint.pprint(response.json())
{'app_id': '680a23234731c68fdf429d23',
 'detail': 'The record specified via `record_id` does not exist in the '
           'LivingApp specified via `app_id` (or is not accessible).',
 'record_id': '000000000000000000000000',
 'status': 404,
 'title': 'Record not found',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪record-not-found'}
```

Feld nicht gefunden (`field-not-found`)

Der Feld-Identifizierer, der bei der HTTP-Abfrage zum Anlegen oder Ändern von Datensätzen übergeben wurde, existiert nicht in der spezifizierten LivingApp.

Beispiel:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.post(
...     "https://my.living-apps.de/rest/apps/680a23234731c68fdf429d23/records",
...     headers={
...         "Accept" : "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
...     json={
...         "fields": {
...             "falsch": 42,
...         },
...     },
... )
>>> pprint.pprint(response.json())
{'app_id': '680a23234731c68fdf429d23',
 'control_identifier': 'falsch',
 'detail': 'The field with the identifier in `control_identifier` does not '
           'exist in the LivingApp specified via `app_id`.',
 'status': 400,
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
 'title': 'Field identifier not found',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪field-not-found'}
```

Ungültiger Feld-Wert (`illegal-field-value`)

Der Feld-Wert, der bei der HTTP-Abfrage zum Anlegen oder Ändern von Datensätzen übergeben wurde, ist für diesen Feld-Typ ungültig, d.h. er hat den falschen Typ oder einen ungültigen Wert.

Beispiele:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.post(
...     "https://my.living-apps.de/rest/apps/5bffc841c26a4b5902b2278c/records",
...     headers={
...         "Accept": "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
...     json={
...         "fields": {
...             "nachname": 42,
...         },
...     },
... )
>>> pprint.pprint(response.json())
{'app_id': '5bffc841c26a4b5902b2278c',
 'control_identifier': 'nachname',
 'control_type': 'string',
 'detail': 'The given value must be `None` or of type `str`, but is of type '
           '`int`.',
 'field_type': 'int',
 'field_value': '42',
 'status': 400,
 'title': 'Unsupported field value',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪illegal-field-value'}
```

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.post(
...     "https://my.living-apps.de/rest/apps/5bffc841c26a4b5902b2278c/records",
...     headers={
...         "Accept": "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
...     json={
...         "fields": {
...             "geburtsdatum": "2000",
...         },
...     },
... )
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
>>> pprint.pprint(response.json())
{'app_id': '5bffc841c26a4b5902b2278c',
 'control_identifier': 'geburtsdatum',
 'control_type': 'date/date',
 'detail': 'The given `date/date` value must conform to the format `%Y-%m-%d`, '
           "but it doesn't.",
 'field_type': 'str',
 'field_value': "'2000'",
 'status': 400,
 'title': 'Unsupported field value',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪illegal-field-value'}
```

Ungültiger Ziel-Applikation (`invalid-target-app`)

Der Wert, der bei Anlegen einer App für ein App-Lookup-Feld als Parameter `lookup_app` angegeben wurde ist ungültig.

Dies kann beispielsweise sein, wenn `None` übergeben wurde:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.post(
...     "https://my.living-apps.de/rest/apps",
...     headers={
...         "Accept": "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
...     json={
...         "name": "My App",
...         "description": "My new App",
...         "gramgen": "m",
...         "typename_nom_sin": "Test",
...         "typename_gen_sin": "Test",
...         "typename_dat_sin": "Test",
...         "typename_acc_sin": "Test",
...         "typename_nom_plu": "Test",
...         "typename_gen_plu": "Test",
...         "typename_dat_plu": "Test",
...         "typename_acc_plu": "Test",
...         "controls": {
...             "my_applookupselect": {"label": "my_applookupselect", "fulltype":
↪"applookup/select", "lookup_app": None},
...         },
...     }
... )
>>> pprint.pprint(response.json())
{'detail': '`lookup_app` is missing or empty',
 'status': 400,
 'title': 'Invalid target app',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la_erw_rest_service_error_
↪invalid_target_app'}
```

Oder auch wenn die URL nicht mit der Basis-URL übereinstimmt:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.post(
...     "https://my.living-apps.de/rest/apps",
...     headers={
...         "Accept": "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
...     json={
...         "name": "My App",
...         "description": "My new App",
...         "gramgen": "m",
...         "typename_nom_sin": "Test",
...         "typename_gen_sin": "Test",
...         "typename_dat_sin": "Test",
...         "typename_acc_sin": "Test",
...         "typename_nom_plu": "Test",
...         "typename_gen_plu": "Test",
...         "typename_dat_plu": "Test",
...         "typename_acc_plu": "Test",
...         "controls": {
...             "my_multipleapplookupselect": {"label": "my_multipleapplookupselect",
↪"fulltype": "multipleapplookup/select", "lookup_app": "https://www.example.com"},
...         },
...     }
... )
>>> pprint.pprint(response.json())
{'detail': 'The given `lookup_app` value must be a valid URL for a target LivingApp,␣
↪so it must start with `https://my.living-apps.de/rest/apps/`, but it doesn't.',
 'status': 400,
 'title': 'Invalid target app',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la_erw_rest_service_error_
↪invalid_target_app'}
```

Ungültiger vSQL-Ausdruck (`invalid-vsql-expression`)

Der vSQL-Ausdruck, der bei der HTTP-Abfrage, die alle Datensätze einer LivingApp zurückliefert, als Parameter `filter` oder `orderby` übergeben wurde, ist ungültig.

Dies kann beispielsweise sein, weil eine ungültige Syntax verwendet wurde:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps/5bffc841c26a4b5902b2278c/records",
...     headers={
...         "Accept" : "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
...     params={
...         "filter": "now() >= ++x",
...     },
... )
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
>>> pprint.pprint(response.json())
{'detail': 'The vSQL expression `now() >= ++x` is invalid',
 'exception': 'antlr3.exceptions.NoViableAltException: '
              "NoViableAltException(38!=['''])",
 'expression': 'now() >= ++x',
 'status': 400,
 'title': 'Invalid vSQL expression',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪invalid-vsql-expression'}
```

oder weil ein Feld angegeben wird, das nicht existiert:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps/5bffc841c26a4b5902b2278c/records",
...     headers={
...         "Accept" : "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
...     params={
...         "filter": "r.v_falsch.startswith('prefix')",
...     },
... )
>>> pprint.pprint(response.json())
{'cause': {'detail': 'The attribute name `v_falsch` is unknown.',
           'expression': 'r.v_falsch',
           'operation': 'Attribute access operation (`A.name`)'},
 'context': 'where',
 'detail': "The `where` vSQL expression `r.v_falsch.startswith('prefix')` is "
           'invalid',
 'exception': 'll.la.vsql.VSQLUnknownNameError',
 'root': {'expression': "r.v_falsch.startswith('prefix')",
          'operation': 'Method call (`A.name(B, ...)`)'},
 'status': 400,
 'title': 'Invalid vSQL expression',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪invalid-vsql-expression'}
```

oder weil ein nicht existierendes Attribut verwendet wird:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps/5bffc841c26a4b5902b2278c/records",
...     headers={
...         "Accept" : "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
...     params={
...         "filter": "r.v_geburtstag.year >= today().falsch",
...     },
... )
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
>>> pprint.pprint(response.json())
{'cause': {'detail': 'The attribute name `falsch` is unknown.',
           'expression': 'today().falsch',
           'operation': 'Attribute access operation (`A.name`)'},
 'context': 'where',
 'detail': 'The `where` vSQL expression `r.v_geburtstag.year >= '
           'today().falsch` is invalid',
 'exception': 'll.la.vsql.VSQLUnknownNameError',
 'root': {'expression': 'r.v_geburtstag.year >= today().falsch',
          'operation': 'Greater-than or equal comparison (`A >= B`)'},
 'status': 400,
 'title': 'Invalid vSQL expression',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪invalid-vsql-expression'}
```

oder weil eine Funktion mit den falschen Argumenten aufgerufen wird:

```
>>> import pprint, requests
>>> login_token =
↪"6809fb00tDfPWbTVNvyeGxFQSFvNYSPTwZybcEQTBsSeMLwVeYZauwSKrEuuvxjqcdqygkoPoQJTJClxdUnSPhenetVXBzJeW
↪"
>>> response = requests.get(
...     "https://my.living-apps.de/rest/apps/5bffc841c26a4b5902b2278c/records",
...     headers={
...         "Accept" : "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
...     params={
...         "filter": "r.v_geburtstag >= >= date(2000, 2, False)",
...     },
... )
>>> pprint.pprint(response.json())
{'cause': {'detail': 'Type combination `INT`, `INT`, `BOOL` is not supported.',
           'expression': 'date(2000, 2, False)',
           'operation': 'Function call (`name(A, ...)`)'},
 'context': 'where',
 'detail': 'The `where` vSQL expression `r.v_geburtstag >= date(2000, 2, '
           'False)` is invalid',
 'exception': 'll.la.vsql.VSQLSubnodeTypesError',
 'root': {'expression': 'r.v_geburtstag >= date(2000, 2, False)',
          'operation': 'Greater-than or equal comparison (`A >= B`)'},
 'status': 400,
 'title': 'Invalid vSQL expression',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪invalid-vsql-expression'}
```

Request-Parameter ungültig (`request-validation-failed`)

Dieser Fehler tritt dann auf, wenn Parameter bei der HTTP-Abfrage ein ungültiges Format haben, also beispielsweise, wenn eine App- oder Datensatz-ID nicht dem vorgeschriebenen Format entspricht.

Beispiel:

```
>>> login_token =
↪"680a032bdtUrdhiMqYLfCRtAiVLMXxmQuKdvpPSBDGuTGsDpdliRYMGaKYoBOxNYDynSxANtfGyJJwpMGtevJtqEiRwDMfZGF
↪"
>>> response = requests.get(
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
...     "https://my.living-apps.de/rest/apps/falsch",
...     headers={
...         "Accept": "application/json",
...         "Authorization": f"Bearer {login_token}",
...     },
... )
>>> pprint.pprint(response.json())
{'errors': [{'ctx': {'pattern': '^[0-9a-f]{24,}$'},
             'input': 'falsch',
             'loc': ['path', 'app_id'],
             'msg': "String should match pattern '^[0-9a-f]{24,}$'",
             'type': 'string_pattern_mismatch'}],
 'status': 422,
 'title': 'Request validation error.',
 'type': 'https://my.living-apps.de/docs/REST-Service.html#la-erw-rest-service-error-
↪request-validation-failed'}
```

32.1.4 Interne Fehlermeldungen

Die folgenden Fehlermeldungen sind interne Fehler, die bei der normalen Verwendung des REST-Services nicht auftreten können (aber hier trotzdem der Vollständigheit halber aufgeführt werden).

Sollte einer dieser Fehler bei Ihnen auftreten, so wenden Sie sich bitte an LivingLogic.

Unbekannter Benutzer (`unknown-user`)

Bei der Verwendung einer internen API konnte der Benutzer nicht bestimmt werden.

„Bearer“-Token fehlt (`bearer-token-missing`)

Bei der Verwendung einer internen API konnte der „Bearer“-Token nicht gefunden werden.

Datensatz konnte nicht angelegt werden (`record-not-created`)

Ein interner Datenbank-Fehler ist während das Anlegens eines Datensatzes aufgetreten.

Datensatz konnte nicht geändert werden (`record-not-updated`)

Ein interner Datenbank-Fehler ist während das Ändern eines Datensatzes aufgetreten.

Datensatz konnte nicht gelöscht werden (`record-not-deleted`)

Ein interner Datenbank-Fehler ist während das Löschens eines Datensatzes aufgetreten.

32.2 GraphQL LivingAPI

Bemerkung

Diese Seite ist für Entwickler gedacht. Manche Links leiten auf englischsprachige Webseiten weiter und diese Seiten sind meist für Entwickler geschrieben.

Mithilfe der GraphQL LivingAPI können Sie in jeder Sprache, die einen Https Client unterstützt, mit LivingApps interagieren.

Die GraphQL LivingAPI (GQL LAPI) ermöglicht es auf Daten einer in einem Template definierten Datenquelle zuzugreifen und diese Daten zu aktualisieren. Zusätzlich können neue Datensätze angelegt werden oder existierende gelöscht werden.

Die Antworten sind hierbei immer gültiges JSON.

Bemerkung

Die LivingApps Daten werden als JSON zurückgegeben und nicht wie beim Python SDK oder Javascript SDK als UL4on übergeben. Dadurch sind GQL LAPI nutzende Anwendungen unabhängiger von LivingAPI und UL4 Updates als zuvor. Falls es durch ein Update zu einer Veränderung an dem Aufbau der LivingAPI kommt, so werden betroffene Felder mit polyfills versehen und als deprecated markiert. Von da an haben Sie noch 90 Tage Zeit die Anwendung an die neue API anzupassen.

Bemerkung

Auf dieser Seite wird nicht auf die allgemeine Nutzung von GraphQL eingegangen. Falls Sie GraphQL noch nicht kennen, sehen Sie sich folgende Seite[16] an.

32.2.1 Einloggen

Die GQL LAPI unterstützt zwei Einloggmöglichkeiten. Zum einen können der Nutzername und das Passwort mit jedem Request mitgeschickt werden. Zum anderen wird der OpenId-Connect Standard unterstützt.

Ersteres eignet sich besonders gut für Backend oder Entwicklungssetups.

Zweiteres benötigt Zugang zu einem von LivingApps unterstützten OIDC (OpenId-Connect) Provider. Wenden Sie sich dafür bitte an unser Team.

32.2.2 Login mit Nutzernamen und Passwort

Es müssen die Header

- `x-la-gql-user`
- `x-la-gql-pwd`

gesetzt werden.

Bspw.

```
curl --request POST \
    --url https://graphql-pwa-la-prod.living-apps.de/ \
    --header 'Content-Type: application/json' \
    --header 'x-la-gql-pwd: <meinGeheimesPasswort>' \
    --header 'x-la-gql-user: <max@mustermann.de>' \
    --data '{"query":"query {...}"}'
```

32.2.3 Login mit OIDC

Für den Login mit OIDC muss von dem OIDC Provider ein Accesstoken geladen werden. Dieser Accesstoken muss dann im Header „authorization“ mitgeschickt werden.

Bemerkung

Achtung: die Accesstokens haben eine relativ kurze Lebenszeit (bspw. 5 Minuten) und müssen stetig erneuert werden. Bitte machen Sie sich mit OIDCs „Authorization Code Flow“[17] bekannt.

[16] https://graphql.org/learn/
[17] https://openid.net/specs/openid-connect-core-1_0.html#CodeFlowAuth

```
curl --request POST \
    --url https://graphql-pwa-la-prod.living-apps.de/ \
    --header 'Content-Type: application/json' \
    --header 'authorization: Bearer <your accesstoken>' \
    --data '{"query":"query {...}"}'
```

Bemerkung

Für Frontendprojekte wie PWAs ist „Login mit OIDC“ zu bevorzugen.

32.2.4 Dokumentation

Wir bieten für die GQL LAPI einen Playground[18] inklusive kurzer englischsprachiger Typendokumentation an. Falls eine deutschsprachige Dokumentation benötigt wird, können die Typen auch in der LivingAPI Dokumentation nachgelesen werden.

32.2.5 GraphQL und Maps

In der LivingAPI werden häufig Maps genutzt aber in GraphQL gibt es keine Maps. Die Alternative sind Listen die Schlüssel-Wert Objekte enthalten. Falls Mapeinträge den Schlüssel selbst als Feld gespeichert haben wird in GraphQL eine Liste der Werte zurückgegeben.

Bspw.

```
// livingapi
{
    [recordId: string]: Record
}
// graphql
RecordV1[]
```

Wenn Maps benötigt werden, müssen diese auf der Clientseite aus der (Schlüssel-)Wert Liste bebildet werden.

32.2.6 Antworten Caching

Die GQL LAPI wird innerhalb der LivingLogic für die Entwicklung von Progressive Web Apps genutzt. Progressive Webs Apps sind Webseiten die wie native Anwendungen genutzt werden können. So können auch PWAs bei keiner oder sehr schlechter Internetverbindung genutzt werden. Bei einer sehr schlechten Internetverbindung kann es dazu kommen, dass eine Anfrage abgeschickt wird und die Antwort nicht empfangen werden kann. Um zu verhindern, dass dann Mutations doppelt ausgeführt werden können Anfragen mit einer Id versehen werden. Wird eine Anfrage mit der gleichen Id vom gleichen Nutzer mehrfach innerhalb einer Stunde ausgeführt so wird das Ergebnis der ersten Anfrage zurückgegeben und die GQL LAPI führt den Query oder die Mutation nicht mehr aus.

```
curl --request POST \
    --url https://my.living-apps.de/gql/ \
    --header 'Content-Type: application/json' \
    --header 'x-la-gql-pwd: <meinGeheimesPasswort>' \
    --header 'x-la-gql-user: <max@mustermann.de>' \
    --header 'x-la-gql-reqid: <meineReqId>' \
    --data '{"query":"mutation {...}"}'
```

[18] https://graphql-pwa-la-prod.living-apps.de/

32.3 Python-SDK

32.3.1 Überblick

Mit dem Python-SDK können Sie mithilfe der Programmiersprache Python[19] mit LivingApps interagieren.

Sie können die Daten die Sie in den Datenquellen für eines Ihrer Anzeige-Templates konfiguriert haben, abrufen, neue Datensätze anlegen, sowie existierende ändern und löschen.

32.3.2 Installation

Das Python-SDK benötigt mindestens Python 3.6. Zum Anlegen eines virtuellen Environment wird außerdem das `venv`[20]-Modul benötigt. Des weiteren wird zur Installation evtl. ein C-Compiler benötigt. Wenn dies nicht bereits Teil der Standard-Installation auf Ihrem System ist, kann beides mittels:

```
$ apt-get install python3.6-venv python3.6-dev
```

installiert werden (bzw. mit dem äquivalenten Aufruf für Ihr System).

Sodann sollte eine neue virtuelle Umgebung angelegt werden, in der alle benötigten Module installiert werden. Wir nennen diese virtuelle Umgebung `livingapps` und installieren sie im Verzeichnis `~/pyvenvs`:

```
$ python3.6 -mvenv ~/pyvenvs/livingapps
```

Anschließend können wir diese Umgebung aktivieren:

```
$ . ~/pyvenvs/livingapps/bin/activate
```

Um Sie darauf hinzuweisen, daß Sie diese Umgebung aktiviert haben, wird forthin dem Prompt der Name der Umgebung als Präfix vorangestellt. D.h. der Prompt ist ab jetzt `(livingapps) $`.

Um das Python-SDK selbst zu installieren, gehen Sie folgenden Befehl ein:

```
(livingapps) $ pip install ll-la
```

Dabei werden auch die benötigten Module `ll.xist`[21], `requests`[22] und `geocoder` installiert.

Um das interaktive Ausprobieren des Python-SDK zu vereinfachen, empfielt es sich IPython zu installieren mittels:

```
(livingapps) $ pip install ipython
```

(Die folgenden Beispiele zeigen die Verwendung von IPython).

32.3.3 Beispiel-App

Als Beispiel-App benutzen wir eine App, mit der wir Daten zu berühmten Personen verwalten. Wir legen zwei Apps an. Die erste (namens „Persons“) enthält die Personen selbst. Sie hat folgende Felder:

[19] http://www.python.org/
[20] https://docs.python.org/3.10/library/venv.html#module-venv
[21] https://python.livinglogic.de/XIST.html#module-ll.xist
[22] https://requests.readthedocs.io/en/latest/api/#module-requests

Name	Identifizierer	Typ	Sonstiges
Firstname	`firstname`	`string/text`	
Lastname	`lastname`	`string/text`	
Sex	`sex`	`lookup/select`	Auswahl `male/female`
Field of activity	`field_of_activity`	`multipleapplookup/select`	Mehrfach-Auswahl aus der zweiten App
Date of birth	`date_of_birth`	`date/date`	
Date of death	`date_of_death`	`date/date`	
Grave	`grave`	`geo`	
Portrait	`portrait`	`file`	

Die zweite App (namens „Fields of activity“) verwaltet die Tätigkeitsfelder und hat zwei Felder:

Name	Identifizierer	Typ	Sonstiges
Name	`name`	`string/text`	
Parent	`parent`	`applookup/select`	Auswahl aus derselben App

32.3.4 Vorbereitungen in der App

Damit Sie auf die Daten Ihrer App zugreifen können, müssen Sie für jeden Datenexport den Sie benötigen, das passende Anzeige-Template anlegen. Dazu muß Ihr Account vom Administrator als Experten-User freigeschaltet worden sein.

In Kürze: Fügen Sie unter *Konfiguration* → *Erweitert* bei *Anzeige-Templates* ein neues Anzeige-Template hinzu. Verwenden Sie als *Identifizierer* `export`, setzen Sie den *Typ* auf *Liste* und setzen Sie *Standard?*.

Bei *Quelltext* muß nichts eingegeben werden (da wir die Daten ja gar nicht in dem Anzeige-Template anzeigen wollen, sondern mit dem SDK darauf zugreifen wollen). Bei *Berechtigung für* können Sie auswählen, wer auf diese Daten zugreifen darf. (Wenn Sie *alle Benutzer* wählen, muß der Benutzer nicht eingeloggt sein, um zugreifen zu können, d.h. die Daten sind öffentlich.) Klicken Sie auf *Speichern* um Ihr Anzeige-Template abzuspeichern.

Sodann können Sie die Datenquellen die Sie benötigen unter *Datenquellen* anlegen. Wir legen hier zwei Datenquellen für unsere beiden Apps an. Klicken Sie bei Ihrem neuen Anzeige-Template auf den Maskenlink *Datenquellen* und dann auf *Hinzufügen*. Bei *App* wählen Sie die App „Persons“ aus, als *Identifizierer* geben Sie `persons` ein, bei *Felder/Datensätze* wählen Sie *Felder und Datensätze*, sowie bei *Felder Alle Felder*. Klicken Sie dann auf *Speichern* um Ihren neue Datenquelle abzuspeichern.

Legen Sie nun eine zweite Datenquelle für die App „Fields of activity“ an (mit dem *Identifizierer* `fieldsofactivity`).

Eine ausführliche Beschreibung von Anzeige-Templates und Datenquellen findet sich unter *Anzeige-Templates*.

32.3.5 Verwendung des SDKs

Starten Sie IPython mit

```
(livingapps) $ ipython
Python 3.6.4 (default, Jan 15 2018, 09:29:23)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.
In [1]:
```

Anschließend importieren Sie das `la`-Modul:

```
In [1]: from ll import la
```

Nun können Sie sich mit Ihrem Account bei LivingApps einloggen:

```
In [2]: handler = la.HTTPHandler("https://my.living-apps.de/", "username", "password")
```

Benutzername und Passwort kann auch weggelassen werden, dann können Sie nur auf öffentliche Anzeige-Templates zugreifen, d.h. auf Templates, bei denen unter *Berechtigung für alle Benutzer* ausgewählt wurde.

Um nun von Ihrer App und Ihren Anzeige-Template Daten abzuholen, benötigen Sie den Identifizierer Ihrer App. Diesen finden sie wenn Sie sich auf der Startseite der jeweiligen App befinden. Die URL diese Seite lautet in unserem Beispiel:

```
https://my.living-apps.de/apps/5bffc841c26a4b5902b2278c.htm
```

`5bffc841c26a4b5902b2278c` ist dann der gesuchte Identifizierer.

Damit können Sie dann die Daten holen, die Sie für Ihr Template konfiguriert haben:

```
In [3]: data = handler.viewtemplate_data("5bffc841c26a4b5902b2278c")
In [4]: data
Out[4]:
{'globals': <ll.la.Globals version='7' platform='LivingApps' at 0x56d62e8>,
 'app': <ll.la.App id='5bffc841c26a4b5902b2278c' name='LA-Demo: Persons' at 0x56d6080>
↪,
 'record': None,
 'apps': {'5bffc841c26a4b5902b2278c': <ll.la.App id='5bffc841c26a4b5902b2278c' name=
↪'LA-Demo: Persons' at 0x56d6080>,
  '5bffc44c5be111d74ed79972': <ll.la.App id='5bffc44c5be111d74ed79972' name='LA-Demo:␣
↪Fields of activity' at 0x57d0550>},
 'datasources': {'persons': <ll.la.DataSource id='5bffccdb8cd2298964e2c7b1'␣
↪identifier='persons' at 0x57ca1d0>,
  'fieldsofactivity': <ll.la.DataSource id='5bffdadd22f815dd7480dfcc' identifier=
↪'fieldsofactivity' at 0x57ca208>}}
```

Wenn Sie ein weiteres Anzeige-Template angelegt haben (z.B. mit dem Identifizierer `specialexport`), können Sie die Daten holen, indem Sie diesen Template-Identifizierer als zweites Argument übergeben:

```
In [3]: data = handler.viewtemplate_data("5bffc841c26a4b5902b2278c", "specialexport")
```

(d.h. bei dem Anzeige-Template, das als *Standard?* konfiguriert wurde, muß der Template-Identifizierer nicht angegeben werden).

Das Objekt `data`, das Sie als Ergebnis erhalten, besitzt drei Attribute:

`globals`
: Globale Informationen zum System und dem eingeloggten Benutzer. Dieses Python-Objekt unterstüzt die unter *Globals* dokumentierten Attribute und Methoden der LivingAPI, die Ihnen im Sourcecode des Anzeige-Templates zur Verfügung stehen.

`datasources`
: Ein Dictionary mit den von Ihnen konfigurierten Datenquellen. Die Keys sind die Identifizierer und die Werte *DataSource*-Objekte. Diese Objekte unterstützen ebenfalls die LivingAPI.

`app`
: Die App, zur der das Anzeige-Template gehört (d.h. die App, deren Id Sie an den `get()`-Aufruf übergeben haben).

Damit können wir auf unsere App „Fields of activity“ folgendermaßen zugreifen:

```
In [3]: tfapp = data.datasources.fieldsofactivity.app
In [4]: tfapp
Out[4]: <ll.la.App id='5bffc44c5be111d74ed79972' name='LA-Demo: Fields of activity'␣
↪at 0x10b857d30>
```

Die Datensätze sind folgendermaßen zugänglich:

```
In [5]:  for r in tfapp.records.values():
......:     print(r.v_name)
......:
Sport
Politics
Literature
Industry
Music
Art
Physics
Maths
Science
Computer science
Film
In [6]:
```

Die Datensätze sind *Record*-Objekte und unterstützen die entsprechenden Attribute und Methoden.

Aus der anderen Datenquelle können wir uns die Daten zu den Personen holen:

```
In [6]: papp = data.datasources.persons.app
In [7]: list(papp.records.values())
Out[7]:
[<ll.la.Record
     id='5bf40f8f11b9ceb4ce7c75ac'
     v_firstname='Albert'
     v_lastname='Einstein'
     v_sex=<ll.la.LookupItem key='male' label='Male' at 0x10fa5bd30>
     v_fieldofactivity=[<ll.la.Record
          id='5bf40f8dc511b06d3be85c7d'
          v_name='Physics'
          at 0x10fa3ea90>]
     v_date_of_birth=datetime.date(1879, 3, 14)
     v_date_of_death=datetime.date(1955, 4, 15)
     at 0x10fa5beb8>,
 <ll.la.Record
     id='5bf40f8f2249662d2b4a8c0b'
     v_firstname='Marie'
     v_lastname='Curie'
     v_sex=<ll.la.LookupItem key='female' label='Female' at 0x10fa5be10>
     v_fieldofactivity=[<ll.la.Record
          id='5bf40f8dc511b06d3be85c7d'
          v_name='Physics'
          at 0x10fa3ea90>]
     v_date_of_birth=datetime.date(1867, 11, 7)
     v_date_of_death=datetime.date(1934, 7, 4)
     at 0x10fa3ef98>,
 <ll.la.Record
     id='5bf40f8f2ecda576c9d524fe'
     v_firstname='Muhammad'
     v_lastname='Ali'
     v_sex=<ll.la.LookupItem key='male' label='Male' at 0x10fa5bd30>
     v_fieldofactivity=[<ll.la.Record
          id='5bf40f8e29069905fe8c71f7'
          v_name='Sport'
          at 0x10fa3e748>]
     v_date_of_birth=datetime.date(1942, 1, 17)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    v_date_of_death=datetime.date(2016, 6, 3)
    at 0x10fa670b8>,
<ll.la.Record
    id='5bf40f8f3cc3d6e19dbe5c50'
    v_firstname='Marilyn'
    v_lastname='Monroe'
    v_sex=<ll.la.LookupItem key='female' label='Female' at 0x10fa5be10>
    v_fieldofactivity=[<ll.la.Record
         id='5bf40f8de2efb2a43c8185d2'
         v_name='Film'
         at 0x10fa3e8d0>]
    v_date_of_birth=datetime.date(1926, 6, 1)
    v_date_of_death=datetime.date(1962, 8, 4)
    at 0x10fa67160>,
<ll.la.Record
    id='5bf40f8f493c32a841ade504'
    v_firstname='Elvis'
    v_lastname='Presley'
    v_sex=<ll.la.LookupItem key='male' label='Male' at 0x10fa5bd30>
    v_fieldofactivity=[<ll.la.Record
         id='5bf40f8dee0ef934d6fe34de'
         v_name='Music'
         at 0x10fa3e7f0>]
    v_date_of_birth=datetime.date(1935, 1, 8)
    v_date_of_death=datetime.date(1977, 8, 16)
    at 0x10fa671d0>,
<ll.la.Record
    id='5bf40f8f58b852d36c8dc59c'
    v_firstname='Bernhard'
    v_lastname='Riemann'
    v_sex=<ll.la.LookupItem key='male' label='Male' at 0x10fa5bd30>
    v_fieldofactivity=[<ll.la.Record
         id='5bf40f8dba17a22c136c8f5b'
         v_name='Maths'
         at 0x10fa3e9e8>]
    v_date_of_birth=datetime.date(1826, 6, 17)
    v_date_of_death=datetime.date(1866, 6, 20)
    at 0x10fa67278>,
<ll.la.Record
    id='5bf40f8f692fb0cfbaf2c5b7'
    v_firstname='Carl Friedrich'
    v_lastname='Gauß'
    v_sex=<ll.la.LookupItem key='male' label='Male' at 0x10fa5bd30>
    v_fieldofactivity=[<ll.la.Record
         id='5bf40f8dba17a22c136c8f5b'
         v_name='Maths'
         at 0x10fa3e9e8>]
    v_date_of_birth=datetime.date(1777, 4, 30)
    v_date_of_death=datetime.date(1855, 2, 23)
    at 0x10fa67320>,
<ll.la.Record
    id='5bf40f8f78befe7d55dc40c2'
    v_firstname='Ronald'
    v_lastname='Reagan'
    v_sex=<ll.la.LookupItem key='male' label='Male' at 0x10fa5bd30>
    v_fieldofactivity=[<ll.la.Record
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        id='5bf40f8de2efb2a43c8185d2'
        v_name='Film'
        at 0x10fa3e8d0>,
    <ll.la.Record
        id='5bf40f8e1202ba6352c2c3c9'
        v_name='Politics'
        at 0x10fa3e0b8>]
    v_date_of_birth=datetime.date(1911, 2, 6)
    v_date_of_death=datetime.date(2004, 6, 5)
    at 0x10fa673c8>]
```

KAPITEL 33

Javascript Helfer Bibliothek

Die Javascript Helfer Bibliothek ist eine Ansammlung an Javascript Bibliotheken, welche Ihnen zur Nutzung in Ihren Anzeige Templates bereitgestellt wird.

Bemerkung

Die Bibliothek und Dokumentation wendet sich an Personen die bereits Erfahrung mit Javascript haben. Es ist vorteilhaft wenn Begriffe wie Variablen, Funktionen, Parameter und Promises bekannt sind. Viele Beispiele in der Dokumentation können mit minimalen Anpassungen genutzt werden.

Warnung

Die Bibliothek unterstützt nur moderne Browser!

33.1 Funktionen

33.1.1 Teilen

Lassen Sie Ihre Nutzer ganz leicht Links zu Ihren Formularen und Anzeigetemplates via Social Media teilen.

Bemerkung

Auf Mobilgeräten wird die Share Funktion[23] von der Web Share API genutzt. Diese wird auf dem Desktop nicht umfassend unterstützt, weshalb auf dem Desktop ein Modal zum Teilen via gängiger Social Media Kanäle bereitgestellt wird.

Installation

Importieren Sie den Stylesheet.

```
<link rel="stylesheet" type="text/css" href="/static/ll-form-utils/0.2.4/dist/share.
↪css" />
```

[23] https://developer.mozilla.org/en-US/docs/Web/API/Navigator/share

Importieren Sie das Javascript-Modul und verwenden Sie es folgendermaßen:

```
<p>
    <button id="shareUrl">Teile URL</button>
    <button id="shareText">Teile Text</button>
    <button id="shareForm">Teile Formular</button>
</p>

<!-- Importieren Sie das share Skript -->
<script src="/static/ll-form-utils/0.2.4/dist/share.js"></script>
<script>

// Teilen Sie eine URL wenn der "Teile URL" Button angeklickt wurde
document.getElementById('shareUrl').addEventListener('click', () => {
    la_share_module.la_share_url('https://my.living-apps.de');
});

// Teilen Sie einen Text mir URL wenn der "Teile Text" Button angeklickt wurde
document.getElementById('shareText').addEventListener('click', () => {
    la_share_module.la_share_text('Ich habe eine Lösung gefunden auf:', 'https://my.
↪living-apps.de');
});

// Teilen Sie eine URL zu einem Formular wenn der "Teile Formular" Button angeklickt␣
↪wurde
document.getElementById('shareForm').addEventListener('click', () => {
    la_share_module.la_share_form(
        'Bitte füllen Sie folgendes Formular aus: ',
        'https://my.living-apps.de/gateway/apps/646cab89ebc17480353d685d/new?
↪view=646cab894f066018ba59e382'
    );
});
</script>
```

Parameter

Es werden, wie im Installationsbeispiel zu sehen, drei verschiedene Funktionen angeboten. Diese erfüllen den gleichen Zweck und unterscheiden sich nur in dem Titel und Hover Effekten auf dem Desktop.

```
type DefaultSupportedLanguages = 'de' | 'en';
function la_share_url(url: string, lang: DefaultSupportedLanguages = 'de'): Promise
↪<void>;
function la_share_text(text: string, url: string, lang: DefaultSupportedLanguages =
↪'de'): Promise<void>;
function la_share_form(text: string, url: string, lang: DefaultSupportedLanguages =
↪'de'): Promise<void>;
```

Tab. 1: Parameter

Parameter	Defau	Beschreibung
`url`		Geben Sie hier den Link an den Sie teilen möchten. Falls der Link nicht auf `https://my.living-apps/` verweist wird eine Warnung angezeigt.
`text`		Geben Sie einen beschreibenden Text zu Ihrem Link an.
`lang`	`'de'`	Geben Sie an in welcher Sprache die Texte im Desktop Popup angezeigt werden soll. Dieser Parameter hat keinen Einfluss auf den übergebenen Text!

33.1.2 E-Mail-Adressen validieren

Reduzieren Sie die Eingabe von ungültigen E-Mail-Adressen indem Sie die Eingabe vom Nutzer nochmal bestätigen lassen.

Installation

Importieren Sie das Stylesheet.

```
<link rel="stylesheet" type="text/css" href="/static/ll-form-utils/0.2.3/dist/
↪validate_mail.css" />
```

Importieren Sie das Javascript-Modul und verwenden Sie es folgendermaßen:

```
<p>
    <input id="mail-input" value="mail-mit-tppfehler@mi.com" />
    <button id="validate-button">Validieren Sie die E-Mail-Adresse</button>
</p>
<p id="validate-result">Resultat: Klicken Sie auf den "Validieren Sie die E-Mail-
↪Adresse" Button</p>
<!-- Importieren Sie das validate_mail Skript -->
<script src="/static/ll-form-utils/0.2.3/dist/validate_mail.js"></script>
<script>
// Erstellen Sie Referenzen zu den DOM Elementen
const inputElement = document.getElementById('mail-input');
const resultElement = document.getElementById('validate-result');
// Registrieren Sie alle Eventlistener, welche die Funktion auslösen sollen.
document.getElementById('validate-button').addEventListener('click', async () => {
    // Lesen Sie den aktuellen Eingabewert aus.
    const value = inputElement.value;
    // Führen Sie die Funktion aus.
    const result = await la_validate_mail_module.la_validate_mail(value);
    // Geben Sie das Resultat an den Nutzer zurück
    inputElement.value = result;
    resultElement.innerText = `Resultat: ${result}`;
});
</script>
```

Parameter

```
function laValidateMail(maybeMail: string, userMail?: string, target?: HTMLElement):␣
↪Promise<string>
```

Tab. 2: Parameter

Parameter	Default	Beschreibung
maybeMa		Die String-Eingabe welche vermutlich eine E-Mail-Adresse ist.
userMai	undefir	Eine bereits bekannte E-Mail-Adresse.
target	undefir	Ein DOM Element ohne Kinder, in welches die Oberfläche gemounted wird. Ist der Wert undefined wird ein DOM Element automatisch erzeugt.

KAPITEL 34

Importfunktion

Bevor Sie Daten importieren, gibt es einige wichtige Punkte, die Sie vorab durchdenken sollten, um sicherzustellen, dass der Import erfolgreich und ohne Probleme abläuft.

34.1 Grundsätzliches

Der Datenimport gelingt, wenn Daten aus einer CSV- oder eine Excel-Datei importiert werden. Vorteil dieser Vorgehensweise ist die Möglichkeit, die Daten vorher auf ihre Qualität zu überprüfen. Das bedeutet, dass die Daten vollständig, konsistent und korrekt sind. Überprüfen Sie, ob es leere Felder oder Duplikate gibt und ob die Daten in der richtigen Form sind.

Beispielsweise sollte der Inhalt der Spalten immer in der gleichen Form vorliegen: Eine Internetadresse sollte in LivingApps als `https://ab.de` angegeben sein. Probleme bereiten unterschiedliche Einträge wie `www.ab.de` und `https://cd.de` in einer Spalte. Man kann damit aus LivingApps heraus bestimmte Funktionalitäten nicht nutzen.

34.2 Unterschiedliche Szenarien für einen Import

Es gibt unterschiedliche Anlässe, warum und wie in LivingApps Daten importiert werden:

1. Es sollen in eine bestehende LivingApp Daten übernommen werden
    a. Es kommen weitere Datensätze dazu
    b. In bestehende Datensätze werden weitere Daten hinzu importiert
2. Erzeugen einer neuen LivingApp beim Import

34.3 Übernahme von Daten in eine bestehende LivingApp

Mit der Übernahme in eine bestehende LivingApp werden weitere Datensätze dieser LivingApp hinzugefügt.

Das Vorgehen läuft folgendermaßen: Wählen Sie in der App, die mit weiteren Datensätzen ergänzt werden soll, den Menüpunkt *Daten* → *Import*.

Kopieren Sie die zu importierenden Excel-Daten mit den dazugehörigen Spaltenköpfen in die Zwischenablage und kopieren Sie die Daten in das geöffnete Import-Fenster:

Klicken Sie *Daten importieren* an.

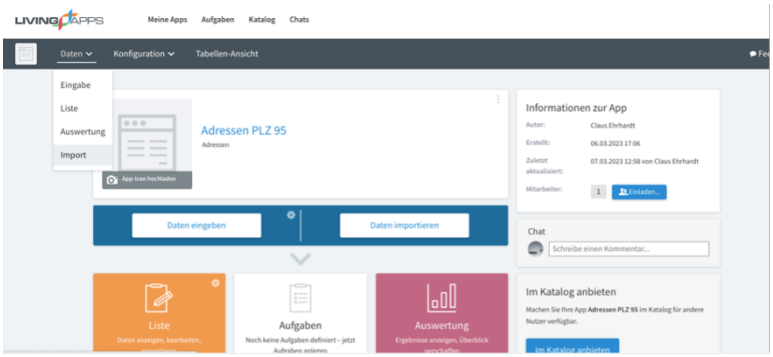

Abb. 1: Das Import-Menü

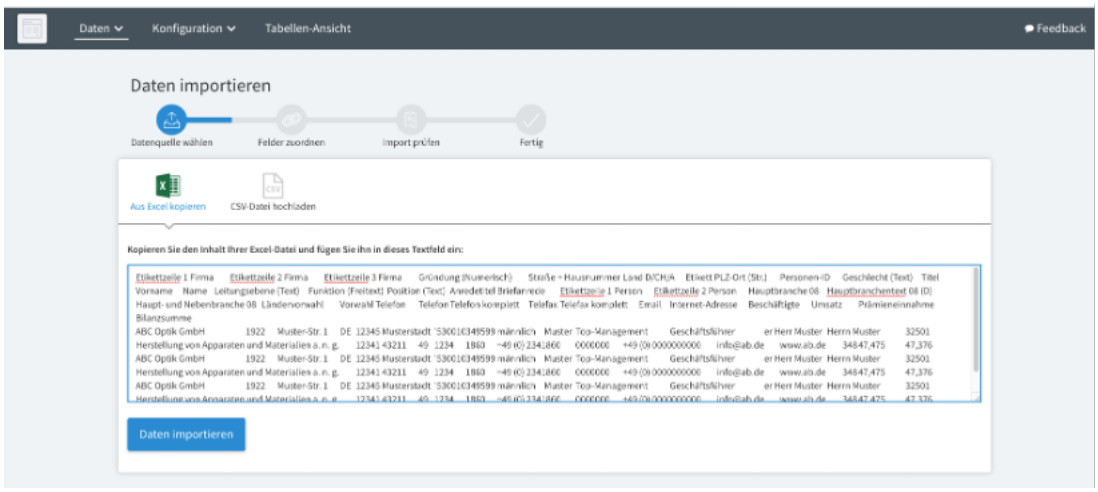

Abb. 2: Daten kopiert

Sie erhalten die Anzahl der Datensätze — hier werden 3 Datensätze angezeigt —, die das System erkannt hat sowie die Zuordnungs-Tabelle mit den jeweiligen Feldern (*Verfügbare Elemente* und *App-Eingabefelder*). Auf der linken Seite sehen Sie die aus dem Excel mitkopierten Spaltenköpfe.

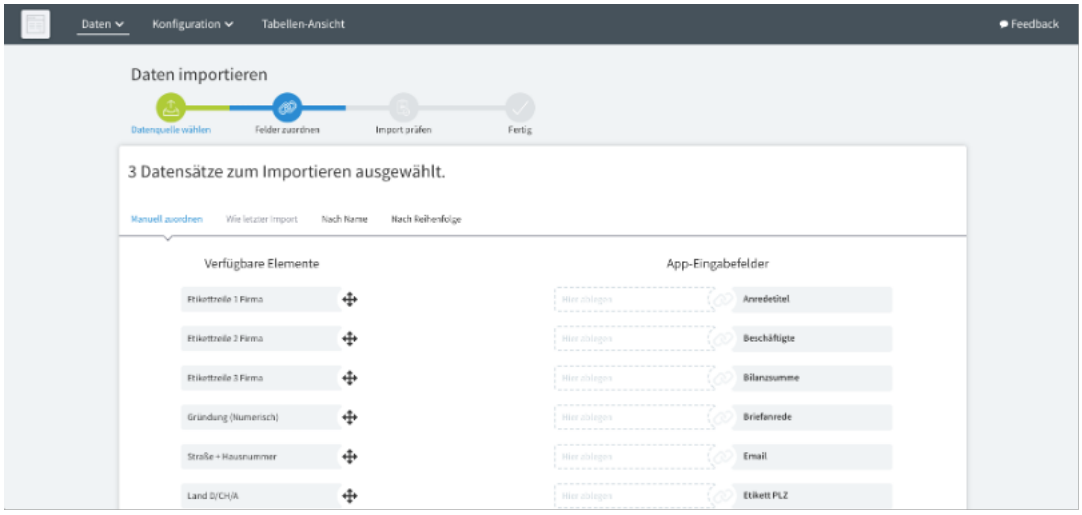

Abb. 3: Felder zuordnen

Vergleichen Sie die Anzahl der Datensätze in LivingApps mit den Zeileneinträgen in Excel, die Sie importieren wollen. Die Zahl ist ein Indiz für die korrekte Übernahme in die LivingApps. (Die Anzahl Datensätze und die Anzahl der Excelzeilen (ohne die Spaltenköpfe) sollten gleich sein.)

Nun können Sie die verfügbaren Elemente den App-Eingabefeldern zuordnen. Sie können das manuell nacheinander tun, alternativ so machen, wie Sie es beim letzten Import (wird gespeichert) zugeordnet haben, nach Namen (das System sucht automatisch nach den identischen Namen für die App-Eingabefelder) oder nach Reihenfolge (also von oben nach unten) zuordnen.

Die Spaltenköpfe müssen nicht notwendigerweise den identischen Namen haben. Sie sollten bei der manuellen Zuordnung jedoch sicher sein, dass die verfügbaren Elemente inhaltlich mit den App-Eingabefeldinhalten zusammenpassen.

Beginnen Sie im Anschluss mit einem Klick auf *Import starten* den Import. Es kann bei großen Datenmengen ein paar Minuten dauern, bis alle Daten in der Datenbank an der richtigen Stelle angekommen sind. Den Status können Sie über die Anzeige des importieren Prozentsatzes der Daten mitverfolgen.

Im Anschluss können Sie das Ergebnis über das Menü *Daten* → *Liste* begutachten.

34.4 Übernahme von Daten in eine bestehende LivingApp (Daten ergänzen)

Eine weitere Form des Imports von Daten ist das Ergänzen bestehender Datensätze mit zusätzlichen Informationen.

Wählen Sie in der App, die mit weiteren Datensätzen ergänzt werden soll, über Menü *Daten* → *Import*.

Kopieren Sie die Excel-Datei oder den Ausschnitt daraus mit den neuen, zu ergänzenden Spaltenköpfen (Feldern) sowie dem Spalteninhalt in das sich öffnende Fenster. Klicken Sie dann *Daten importieren* an.

Sie erhalten die Anzahl der Datensätze — hier werden 3 Datensätze angezeigt —, die das System erkannt hat sowie die Zuordnungs-Tabelle mit den jeweiligen Feldern (*Verfügbare Elemente* und *App-Eingabefelder*). Auf der linken Seite sehen Sie die aus dem Excel mitkopierten Spaltenköpfe.

Vergleichen Sie die Anzahl der Datensätze in LivingApps mit den Zeileneinträgen in Excel, die Sie importieren wollen. Die Zahl ist ein Indiz für die korrekte Übernahme in die LivingApps.

Zur Beachtung: Beim Aktualisieren von bestehenden Datensätzen kann es auch dazu kommen, dass ausgewählte Felder überschrieben werden, die z.B. beim Import oder durch Eingabe bereits gespeichert waren!

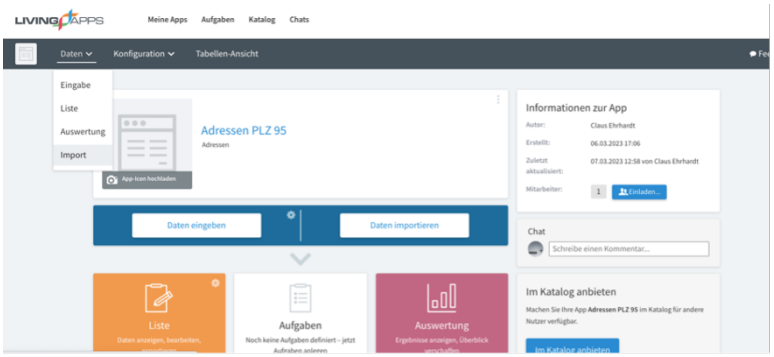

Abb. 4: Das Import-Menü

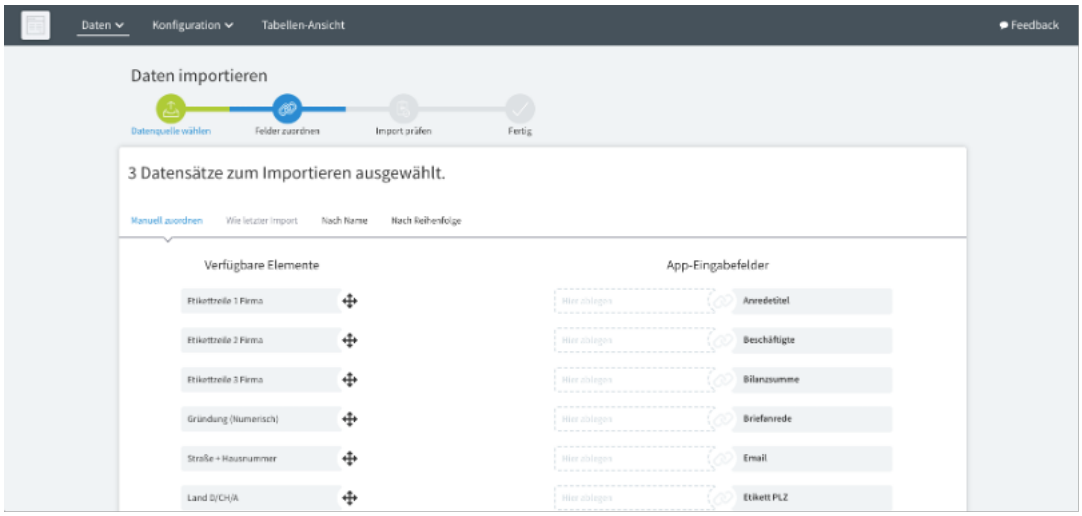

Abb. 5: Felder zuordnen

Wenn Sie den Datenbankinhalt der bestehenden Datensätze ergänzen wollen, benötigen Sie hier ein eindeutiges Schlüsselfeld. Wählen Sie ein Schlüsselfeld, z.B. der eindeutige Firmenname, und kreuzen das Kästchen hierfür an, damit das System erkennen kann, zu welchem Datensatz der Inhalt überschrieben oder hinzugefügt werden soll.

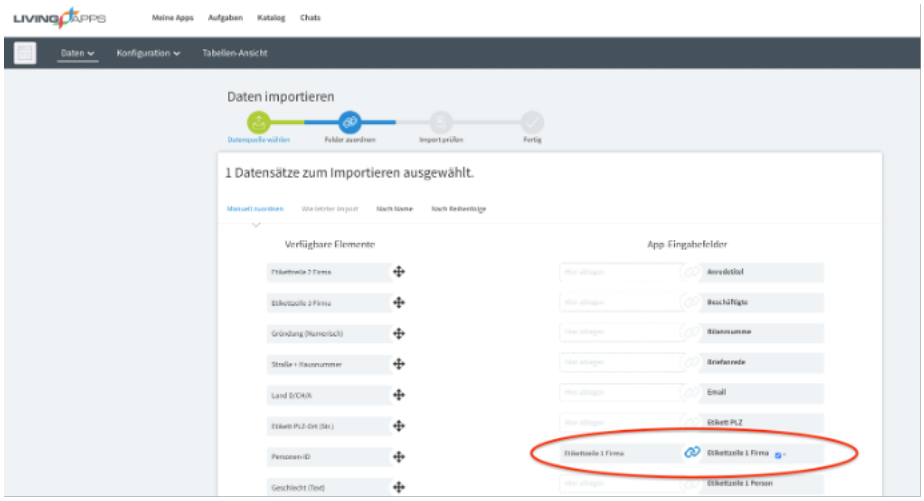

Abb. 6: Schlüsselfeld ausgewählt

Nun können Sie die restlichen benötigten Elemente den App-Eingabefeldern per „Drag and Drop“ zuordnen.

Sie können das manuell nacheinander tun, oder so machen, wie Sie es beim letzten Import (wird gespeichert) zugeordnet haben, nach Namen (das System ordnet Spalten zu Feldern mit demselben Namen) oder nach Reihenfolge (also von oben nach unten) zuordnen.

Zur Beachtung: Möglicherweise ist es notwendig, zwei oder mehrere Felder als Schlüsselfeld zu definieren, damit der passende Datensatz identifiziert werden kann. Zum Beispiel müssen Sie Nachname und Vorname ankreuzen, um den eindeutigen Datensatz zu identifizieren. Dies kann dann der Fall sein, wenn beispielsweise der Nachname „Müller“ öfter als nur einmal vorkommt.

Die Spaltenköpfe müssen nicht notwendigerweise den identischen Namen haben. Sie sollten bei der manuellen Zuordnung jedoch sicher sein, dass die verfügbaren Elemente inhaltlich mit den App-Eingabefeldinhalten zusammenpassen.

Beachten Sie, dass es zu allerlei Fehlern beim Import kommen kann, wenn bei :guilabel`Verfügbare Elemente` Spalten mit Werten ausgewählt sind, deren Inhalt aber nicht zum Feldtyp passt. Beispielsweise erwartet die LivingApp beim Typ URL einen Feldinhalt nicht in der Form `www.blafasel.de`. Richtig ist an dieser Stelle ein Eintrag in der Form `https://blafasel.de`. Darüber hinaus gibt es Probleme in den Feldinhalten, die zum Beispiel Anführungszeichen beinhalten. Bei einer Kopie direkt aus Excel heraus kann der Inhalt nicht richtig interpretiert werden und der Import bricht an dieser Stelle ab. Ein Beispiel für einen Importabbruch ist folgender Inhalt in einem Feld, welches Sie aus einer Exceltabelle kopiert haben: `Hase "Paul" GmbH`. Suchen Sie nach derartigen Inhalten und löschen Sie die Anführungszeichen.

Bemerkung

Führen Sie den Import durch Hochladen einer CSV–Datei durch, die Daten in Anführungszeichen enthält, heilt das System den Fehler und der Import gelingt problemlos.

Auch Mehrfachinhalte in den Datenzellen führen in der Regel zum Abbruch, wenn der Feldtyp einzelne Inhalte erwartet. Z.B. führt ein Eintrag `ab@cd.de, fg@df.de` für ein E-Mail-Feld zum Abbruch des Imports.

Achten Sie auf absolut saubere Daten!

Beginnen Sie im Anschluss mit einem Klick auf *Import starten* den Import. Es kann bei großen Datenmengen ein paar Minuten dauern, bis alle Daten in der Datenbank an der richtigen Stelle angekommen sind. Den Status können Sie über die Anzeige des importieren Prozentsatzes der Daten mitverfolgen.

Im Anschluss können Sie das Ergebnis über das Menü *Daten* → *Liste* überprüfen.

34.5 Import in eine neue LivingApp

Eine weitere, sehr effiziente Möglichkeit, Daten mit LivingApps zu verarbeiten, ist der Import aus einer Excel-Datei bzw. einer CSV-Datei und dem Aufbau einer neuen App aus den importierten Daten.

Hierzu geben Sie derzeit folgenden Link in den Browser ein, in dem Sie die LivingApps geöffnet haben:

```
https://my.living-apps.de/import-neue-app.htm#
```

Sie gelangen in dieses Fenster:

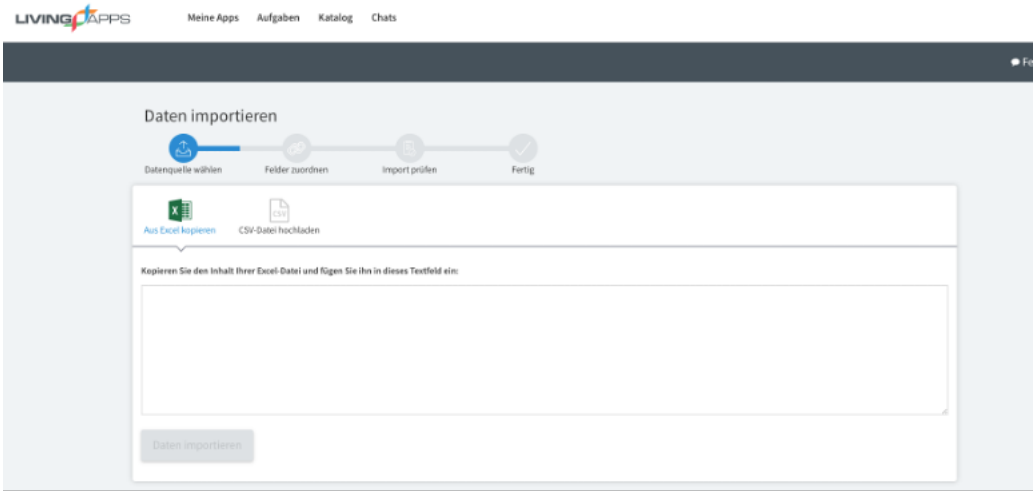

Abb. 7: Daten kopieren

Kopieren Sie die zu importierenden Excel-Daten mit den dazugehörigen Spaltenköpfen in die Zwischenablage und kopieren Sie die Daten in das geöffnete Import-Fenster:

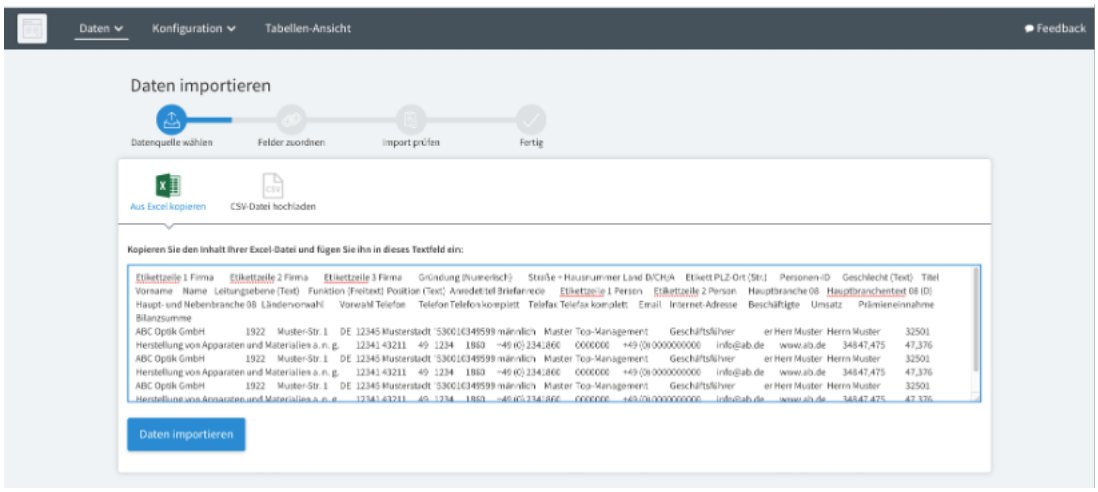

Abb. 8: Daten kopiert

Klicken Sie *Daten importieren* an.

Sie erhalten die Anzahl der Datensätze — hier wird 1 Datensatz angezeigt —, die das System erkannt hat sowie die Zuordnungs-Tabelle mit den jeweiligen Feldern (*Verfügbare Elemente*).

Vergleichen Sie die Anzahl der Datensätze, die Sie in die LivingApps importieren wollen, mit den Zeileneinträgen in Excel. Die Zahl ist ein Indiz für die korrekte Übernahme in die LivingApps. (Die Anzahl Datensätze und die

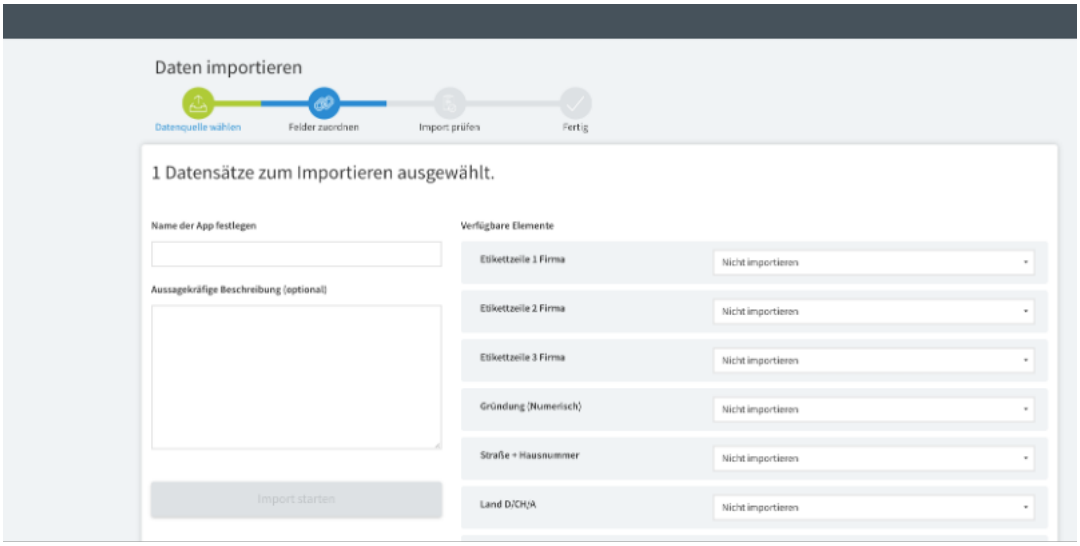

Abb. 9: Feldtypen festlegen

Anzahl der Excelzeilen ohne die Spaltenköpfe sollten gleich sein.)

Bevor Sie die Daten importieren, benötigt die neu in LivingApps erzeugte LivingApp weitere Angaben zu den Feldtypen. Sie können zu den verfügbaren Elementen folgende Feldtypen kreieren:

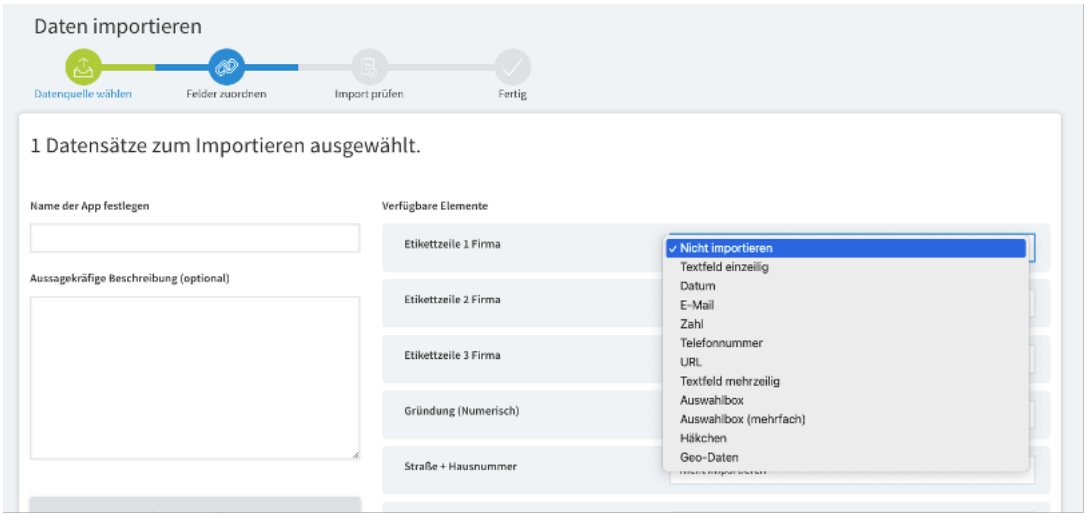

Abb. 10: Auswahl der Feldtypen

Es ist Ihnen bei der Auswahl der „Verfügbaren Elemente“ auch möglich, einzelne Felder nicht zu importieren. Hierzu lassen Sie die Voreinstellung *Nicht importieren* einfach ausgewählt stehen.

Auf der linken Seite dieser Maske werden Sie nach dem Namen der neuen LivingApp sowie einer möglichen weiteren Beschreibung der LivingApp gefragt.

Beachten Sie, dass es zu allerlei Fehlern beim Import kommen kann, wenn für die „Verfügbaren Elemente“ Feldtypen ausgewählt wurden, die nicht zu den vorhandenen Feldwerten passen. Beispielsweise erwartet die LivingApp beim Typ URL einen Feldinhalt nicht in der Form `www.blafasel.de`. Richtig ist an dieser Stelle ein Eintrag in der Form `https://blafasel.de`. Darüber hinaus gibt es Probleme bei Feldinhalten, die zum Beispiel Anführungszeichen beinhalten. Bei einer Kopie direkt aus Excel heraus kann der Inhalt nicht richtig interpretiert werden und der Import bricht an dieser Stelle ab. Ein Beispiel für einen Importabbruch ist folgender Inhalt in einem Feld, welches Sie aus einer Exceltabelle kopiert haben: `Hase "Paul" GmbH`. Suchen Sie nach derartigen Inhalten und löschen Sie die Anführungszeichen.

Bemerkung

Führen Sie den Import durch Hochladen einer CSV–Datei durch, die Daten in Anführungszeichen enthält, heilt das System den Fehler und der Import gelingt problemlos.

Auch Mehrfachinhalte in den Datenzellen führen in der Regel zum Abbruch, wenn der Feldtyp einzelne Inhalte erwartet. Z.B. führt ein Eintrag `ab@cd.de, fg@df.de` für ein E-Mail-Feld zum Abbruch des Imports.

Achten Sie auf absolut saubere Daten!

Sobald Sie im Anschluss auf *Import starten* klicken, beginnt der Import. Sollten es viele Datenzeilen sein, die geladen werden, kann sich der Import bis zu einigen Minuten hinziehen. Sie erkennen den Importstatus am Prozentsatz der geladenen Daten.

Nach dem Import erhalten Sie folgende Meldung:

Abb. 11: Import erfolgt

34.5.1 Begutachtung der neuen App

Um den Import zu prüfen und mit dem Aufbau eines Formulars der LivingApp zu beginnen, klicken Sie auf *Zur Detailansicht der App* und gelangen in das Startfenster der App. Diese sieht so aus:

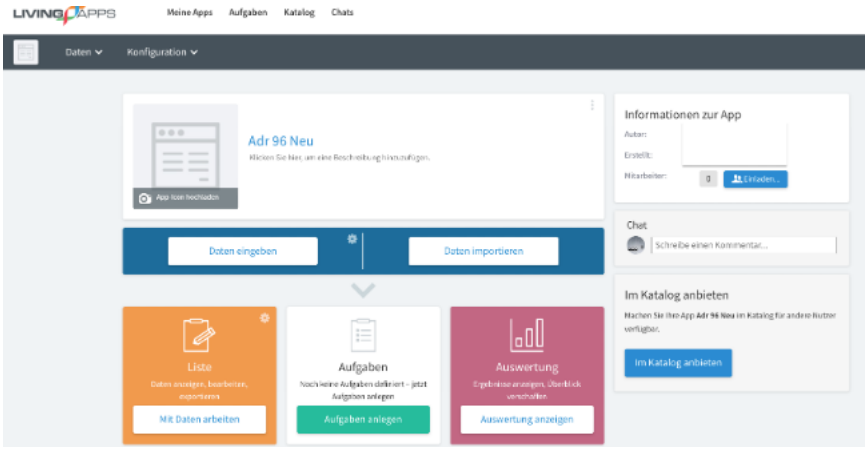

Abb. 12: App-Startseite

Um zur Anzeige der importierten Daten zu gelangen, wählen Sie im Menü *Daten* → *Liste* aus oder klicken weiter unten in der orangenen Kachel auf *Liste*.

Im Anschluss sehen Sie nun die Liste der importieren Daten:

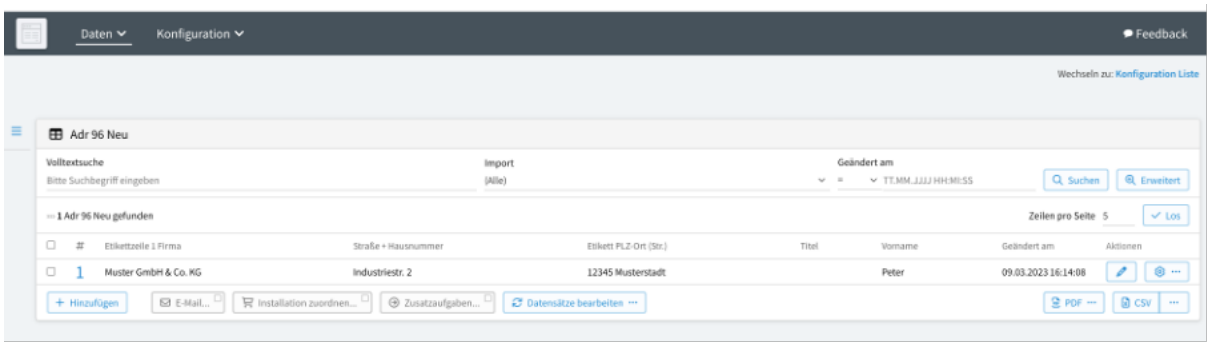

Abb. 13: Datenmanagement

34.5.2 Konfiguration der neuen App

Die Anzeige oder Reihenfolge der in der Liste verwendeten Spalten können Sie ändern, indem Sie im Menü *Konfiguration → Erweitert* wählen, in der linken Navigation *Datenmanagement*, und dort den Punkt *Felder* auswählen.

Um die Spaltenreihenfolge für alle Spalten anzupassen, klicken Sie die Spaltenüberschrift *Reihenfolge Datenmanagement* an, wählen *Bearbeiten ein* und ändern dann in der Spalte *Reihenfolge Datenmanagement* die Werte. Klicken Sie anschließend den erscheinenden Button *Alle speichern* an.

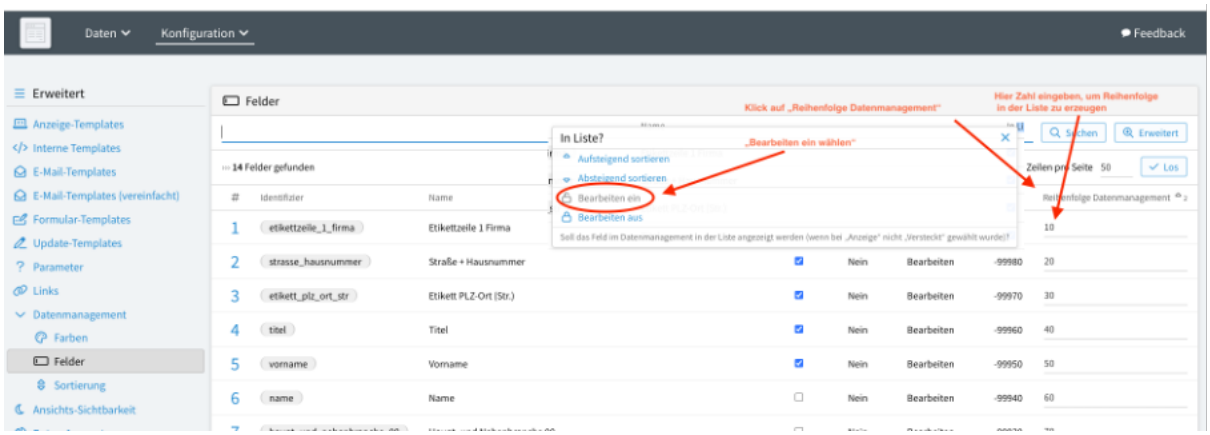

Abb. 14: Spaltenreihenfolge festlegen

Um festyzulegen, welche Spalten in der Liste dargestellt werden gehen Sie ähnlich vor: Klicken Sie die Spaltenüberschrift *In Liste?* an, wählen *Bearbeiten ein* und selektiern oder deselektieren dann in der Spalte *In Liste?* den haken. Klicken Sie anschließend den erscheinenden Button *Alle speichern* an.

Um für diese App ein Formular zu bauen, wählen Sie im Menü *Konfiguration → Eingabe*. Sie erhalten die Maske des „Formularbuilders" mit den Feldnamen zur Platzierung auf dem Formularbogen. Diese Felder finden Sie unter *Private Felder*. Gestalten Sie Ihr Formular, wie gewünscht.

34.6 Unterstützung des Imports durch aktive Hinweise auf Fehler

Beim Import können Fehler auftreten, die durch Dateninhalte verursacht werden, die nicht zu den gewählten Feldtypen passen. In diesem Falle unterstützt Sie das System mit Fehlermeldung wie oben dargestellt. Hier passt z.B. der Feldinhalt aufgrund eines vorangestellten Anführungszeichens nicht zu einem reinen Zahlenfeld. Sie haben die Möglichkeit, den Fehler zu ignorieren und den Import trotzdem zu starten. Andererseits stoppen Sie den Import und korrigieren den Fehler in Excel oder den Daten der CSV-Datei.

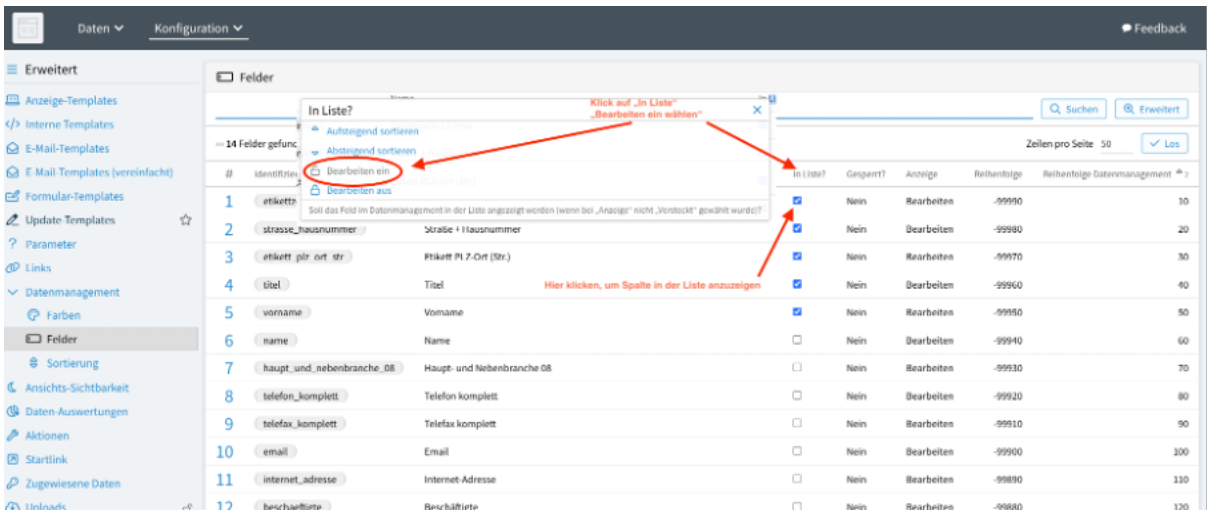

Abb. 15: Spaltenanzeige festlegen

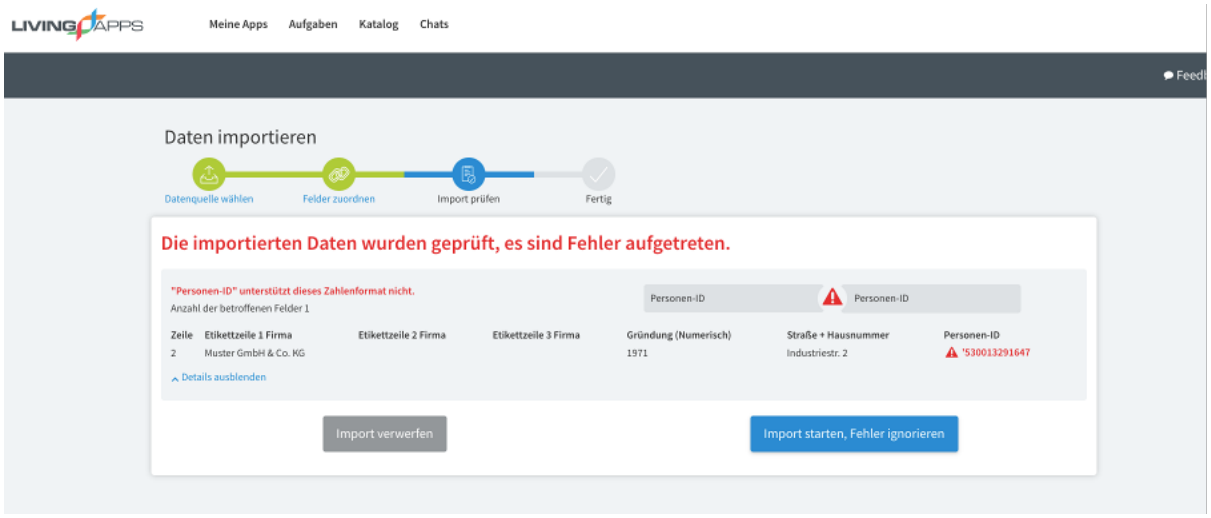

Abb. 16: Importfehler

KAPITEL 35

Howtos

Dieses Kapitel behandelt einige Anwendungsbeispiele.

35.1 Wie bekomme ich einen LivingApps-Account?

Es gibt zwei Möglichkeiten, einen LivingApps-Account zu bekommen.

1. Eigeninitiative
2. Man wird zur Mitarbeit an einer LivingApp eingeladen

35.1.1 Vorgehen bei Eigeninitiative

Registrieren Sie sich auf der Registrierungsseite. Nach Überprüfung wird Ihr Account freigeschaltet.

35.1.2 Vorgehen bei Einladung zur Mitarbeit an einer LivingApp

- Sie bekommen eine E-Mail mit Einladungslink.
- Wenn Sie die Einladung zur Mitarbeit an der LivingApp annehmen, öffnet sich automatisch die *Registrierungsseite*.
- Ihr Account ist gleich nach Registrierung freigeschaltet.

35.2 Wie bekomme ich Administrationsrechte an einer LivingApp?

35.2.1 Voraussetzung

Sie haben eine *LivingApps-Account*.

35.2.2 Vorgehen

Sie haben automatisch Administrationsrechte an einer LivingApp, wenn Sie die LivingApp erstellt haben. Wurden Sie zur Mitarbeit an einer LivingApp eingeladen, kann der Besitzer dieser LivingApp Sie als Administrator freischalten.

Registrierung für LivingApps

Die Zugänge sind gegenwärtig für Firmen und Organisationen gedacht. Notwendig ist eine zur Firma/Organisation gehörige E-Mail-Adresse sowie eine Ansprechperson. Bitte geben Sie eine Motivation an, warum Sie LivingApps nutzen möchten. Im Einzelfall können auch private Nutzer einen Zugang erhalten.

E-Mail / Benutzername*

Passwort*

Passwort bestätigen*

Bitte teilen Sie uns Ihre Motivation zur Nutzung von LivingApps mit.

Ansprechpartner

Anrede*

Vorname*

Nachname*

Firma/Organisation*

Kontaktdaten

Straße

Hsnr.

PLZ

Ort

Land

Deutschland

Telefon

Fax

Promotion

Zugangscode

Rechtliches

☐ Ich bin mit den AGB und der Datenschutzerkärung einverstanden. Die Kundeninformationen habe ich zur Kenntnis genommen.

Registrieren

Abb. 1: Registrierungsseite

Sehr geehrte Damen und Herren,

[illegible] bittet Sie, bei der LivingApp „Kontakte“ mitzuarbeiten.

Ja, gerne: http://lapp.io/V1pwkwxgWg

Nein, danke: http://lapp.io/Ekxpv1vxe-e

Mit LivingApps erledigen Sie die tausenden, täglich auf Sie einstürmenden Aufgaben effektiv, effizient, koordiniert und dokumentiert.
Klicken Sie sich Ihre eigenen Apps ohne Programmierkenntnisse und fremde Hilfe einfach selbst zusammen und erschaffen Sie Lösungen, die Ihr Leben erleichtern.

Viele Grüße
Ihr LivingApps-Team

Abb. 2: E-Mail mit Einladungslink

35.3 Wie kann ich Personen zur Mitarbeit an einer LivingApp einladen?

35.3.1 Voraussetzungen

- Sie sind Besitzer der LivingApp oder haben *Administrationsrechte* an der LivingApp, zu der Sie Personen einladen wollen.

35.3.2 Vorgehen

- Wählen Sie zuerst die App aus, zu der Sie Personen einladen wollen.
- Klicken Sie dann rechts im Panel *Informationen zur App* auf *Einladen…*.

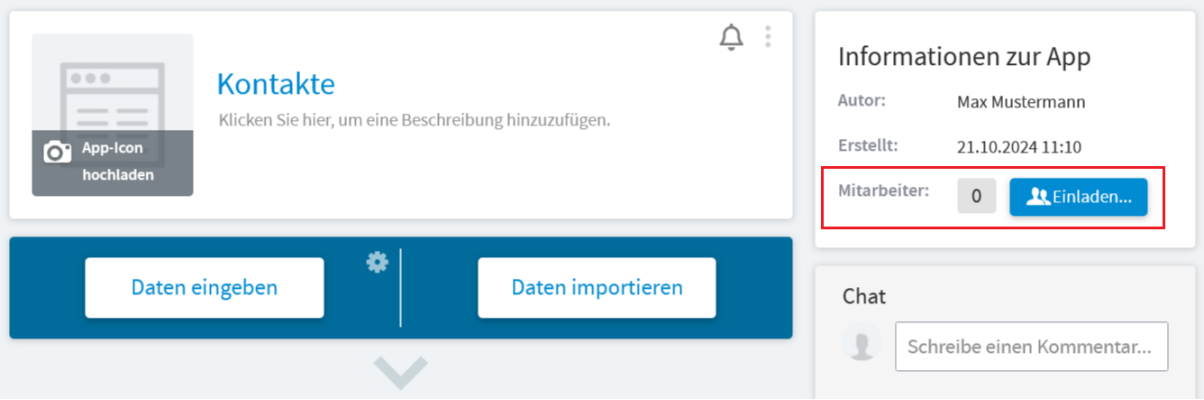

Abb. 3: Mitarbeiter einladen

- Im nun geöffneten Fenster haben Sie die Möglichkeit nach Apps oder Personen zu suchen, mit denen Sie schon zusammen an LivingApps arbeiten oder neue Personen hinzuzufügen.

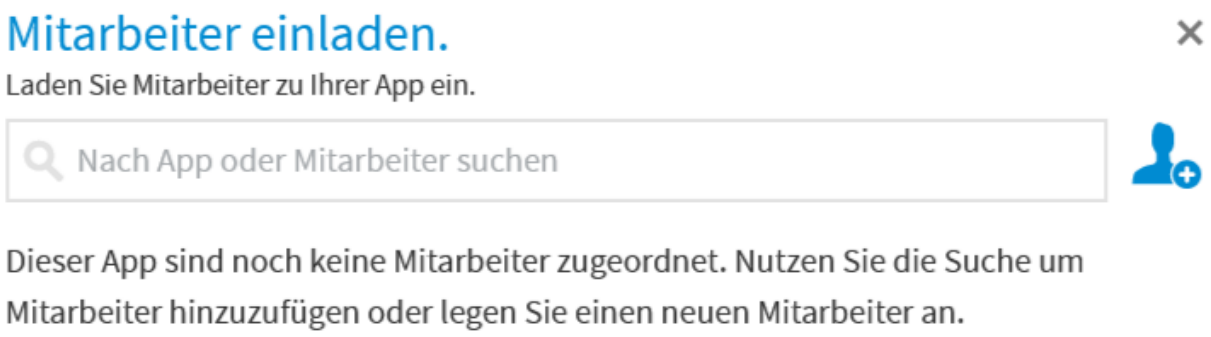

Abb. 4: Mitarbeiter auswählen

- Durch Klicken in das Suchfeld öffnet sich folgende Ansicht.

  Tippen Sie in das Eingabefeld den Namen oder die E-Mail-Adresse der einzuladenden Person.

  Wurde die gewünschte Person gefunden, können Sie das Häkchen setzen, um die entsprechende Person auszuwählen und festzulegen, welche Berechtigungen die Person an Ihrer LivingApp haben soll. Durch Klicken auf *Fertig* und danach auf *Übernehmen*, ist die Einladung abgeschlossen. Die Person erhält eine E-Mail, in der die Einladung angenommen oder abgelehnt werden kann.

  War Ihre Suche erfolglos, können Sie durch Klicken auf das + neben dem Personen-Icon eine neue Person anlegen. Es öffnet sich folgende Ansicht.
- Tragen Sie hier die Daten des neuen Mitarbeiters ein und wählen die Berechtigungsstufe aus.
- Durch Klicken auf *Jetzt hinzufügen* und dann auf *Übernehmen* ist die Einladung abgeschlossen.
- Die eingeladene Person erhält nun eine E-Mail, in der die Einladung angenommen oder abgelehnt werden kann.
- Hat die eingeladene Person bereits eine *LivingApps-Account*, gelangt Sie nach Annahme der Einladung entweder zur Loginseite oder direkt auf die App-Übersichtsseite, je nachdem, ob sie schon eingeloggt war oder nicht.
- Hat die eingeladene Person noch keinen *LivingApps-Accunt*, gelangt Sie nach Annahme der Einladung zur *Registrierungsseite*. Nach Registrierung ist der Account sofort freigeschaltet.

35.4 Wie kann ich mich als Experten-User freischalten lassen?

35.4.1 Voraussetzung

Sie haben *Administrationsrechte* an einer LivingApp.

35.4.2 Vorgehen

Schreiben Sie eine E-Mail oder rufen Sie uns an. Die Freischaltung kann nur durch einen unserer Mitarbeiter vorgenommen werden.

35.5 Wie kann ich mehrere Datensätze auf einmal ändern?

35.5.1 Voraussetzungen

- Sie haben mindestens Datenverarbeitungsrechte an der LivingApp.

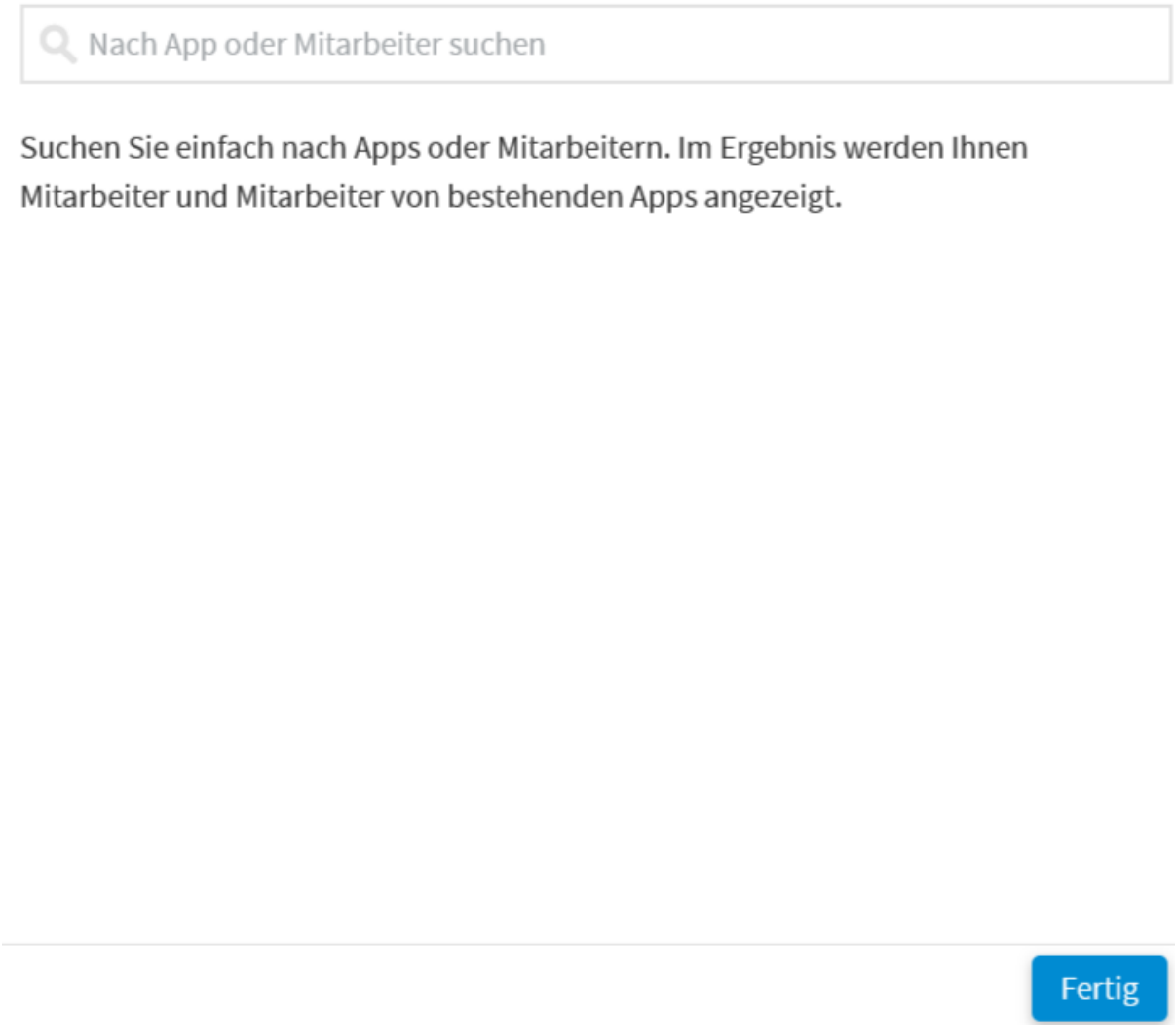

Abb. 5: Mitarbeiter suchen

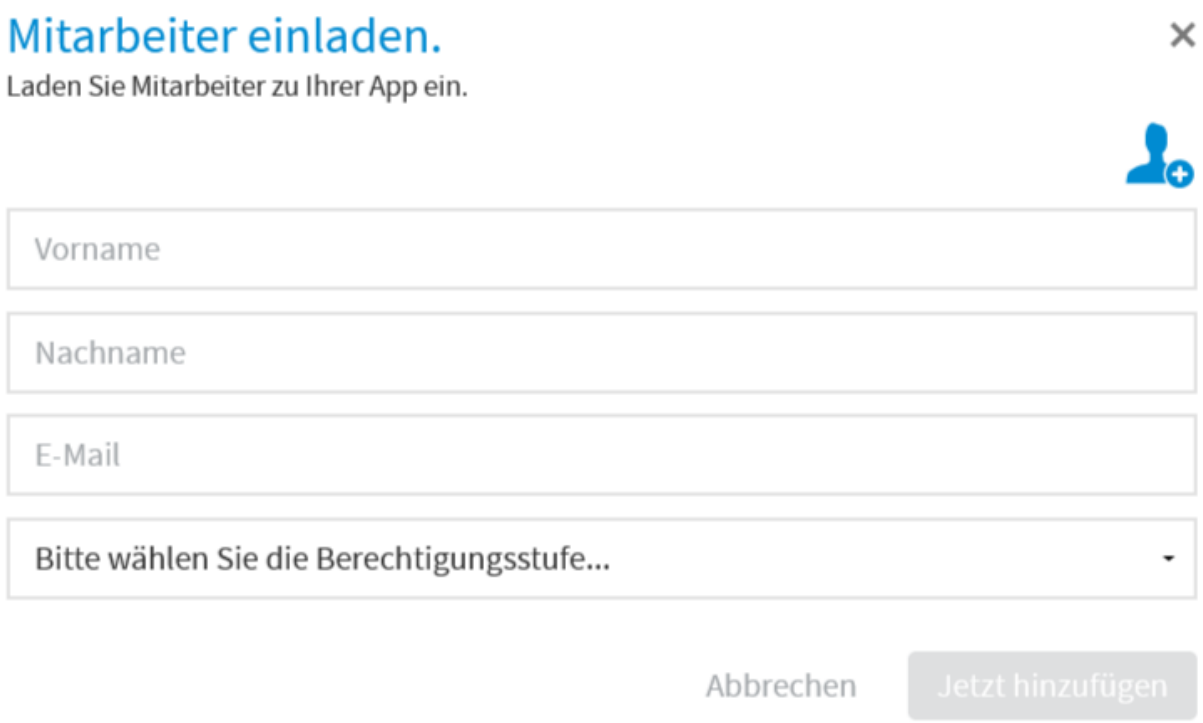

Abb. 6: Mitarbeiter neu hinzufügen

35.5.2 Vorgehen

- Klicken Sie im Menü *Daten* → *Liste* an, um ins Datenmanagement zu gelangen.
- Setzen Sie in der Datenliste das Häkchen bei den Datensätzen, die Sie ändern wollen. Sie können die Datensätze einzeln oder, indem Sie in das Kästchen in der Überschriften-Zeile klicken, alle Datensätze auf dieser Seite oder alle gefundenen Datensätze auswählen.

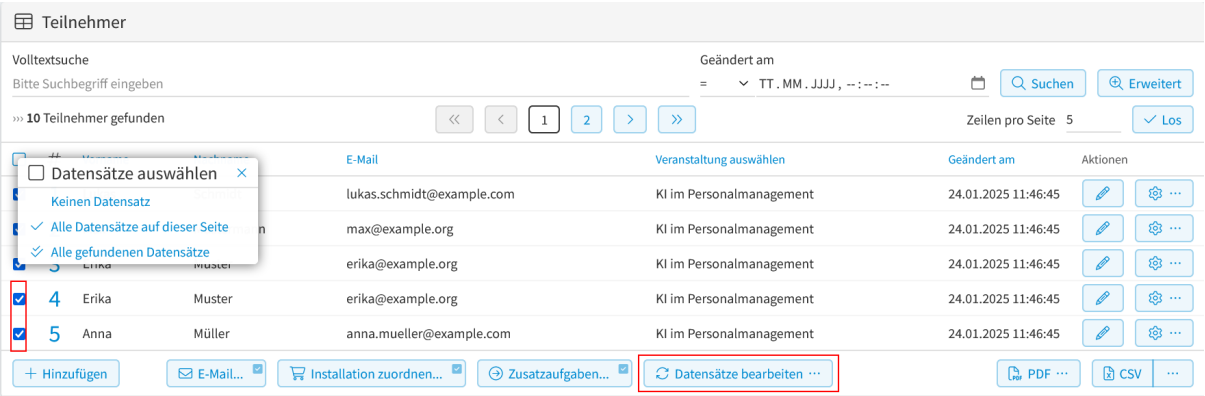

Abb. 7: Datensätze auswählen

- Klicken Sie anschließend in der Fußzeile der Datenliste auf *Datensätze bearbeiten...* und wählen *Datensätze ändern*.
- Es öffnet sich ein Fenster mit einer Übersicht aller Felder Ihrer LivingApp.
- Wählen Sie hier das Feld aus, das in Ihren ausgewählten Datensätzen verändert werden soll. Möchten Sie z. B. mehreren Teilnehmern eine Veranstaltung zuordnen, klicken Sie bei *Feld* auf „Veranstaltung auswählen“, bei *Aktion* auf „Feld auf Wert setzen“ und bei *Wert* auf die entsprechende Veranstaltung. Mit Klick auf *Ändern* werden die Änderungen übernommen.
- Genausogut könnten Sie bei allen gewählten Datensätzen das Feld „Veranstaltung auswählen“ leeren, indem

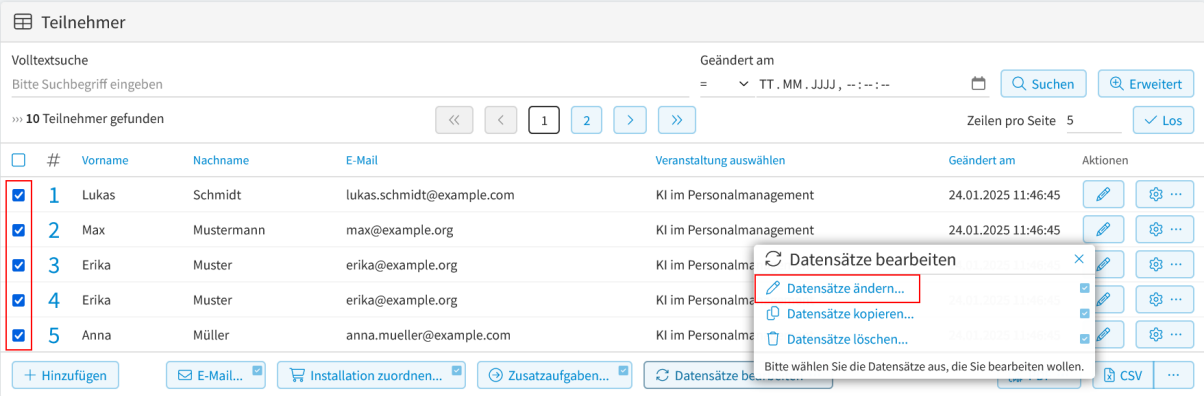

Abb. 8: Datensätze ändern

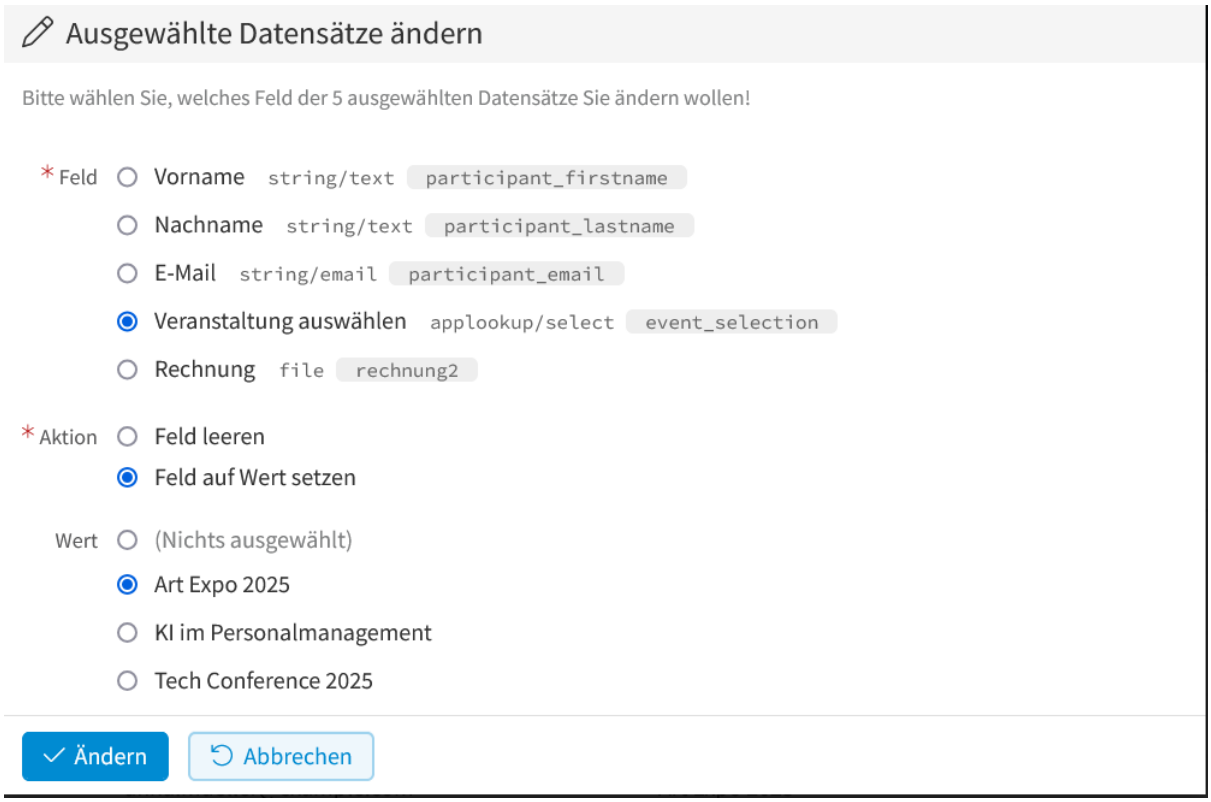

Abb. 9: Feld auswählen

Sie bei *Aktion* „Feld leeren“ auswählen.

- Möchten Sie für verschiedene Felder Änderungen vornehmen, müssen Sie diesen Vorgang für jedes zu ändernde Feld wiederholen.

35.6 Wie kann ich mehrere Datensätze auf einmal kopieren?

Sie haben die Möglichkeit, Felderwerte in ausgewählten Datensätzen beim Kopieren zu ändern. Wenn Sie kein Feld auswählen, werden die Datensätze ohne Änderung kopiert.

Im Beispiel werden Veranstaltungen kopiert und für die kopierten Datensätze ein neues Veranstaltungsdatum festgelegt.

35.6.1 Voraussetzungen

- Sie haben mindestens Datenverarbeitungsrechte an der LivingApp.

35.6.2 Vorgehen

- Klicken Sie im Menü *Daten* → *Liste* an, um ins Datenmanagement zu gelangen.
- Setzen Sie in der Datenliste das Häkchen bei den Datensätzen, die Sie kopieren wollen. Sie können die Datensätze einzeln oder, indem Sie in das Kästchen in der Überschriften-Zeile klicken, alle Datensätze auf dieser Seite oder alle gefundenen Datensätze auswählen.

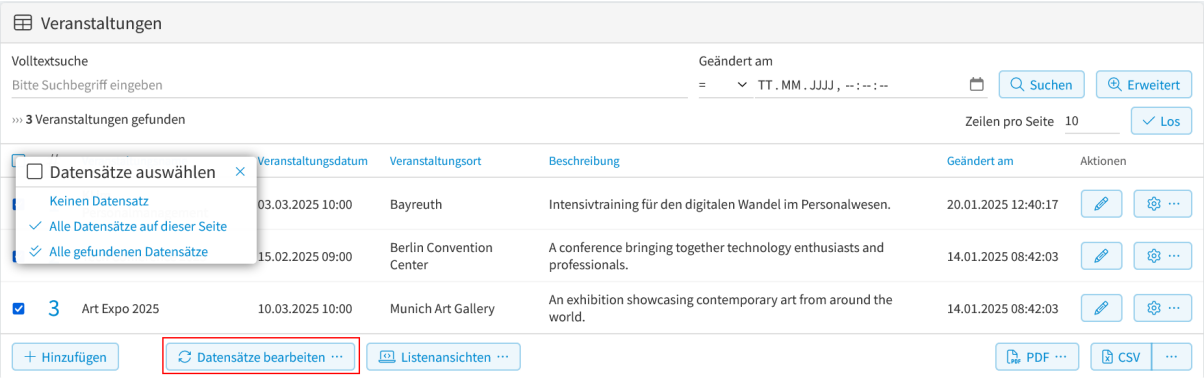

Abb. 10: Datensätze auswählen

- Klicken Sie anschließend in der Fußzeile der Datenliste auf *Datensätze bearbeiten…* und wählen *Datensätze kopieren*.

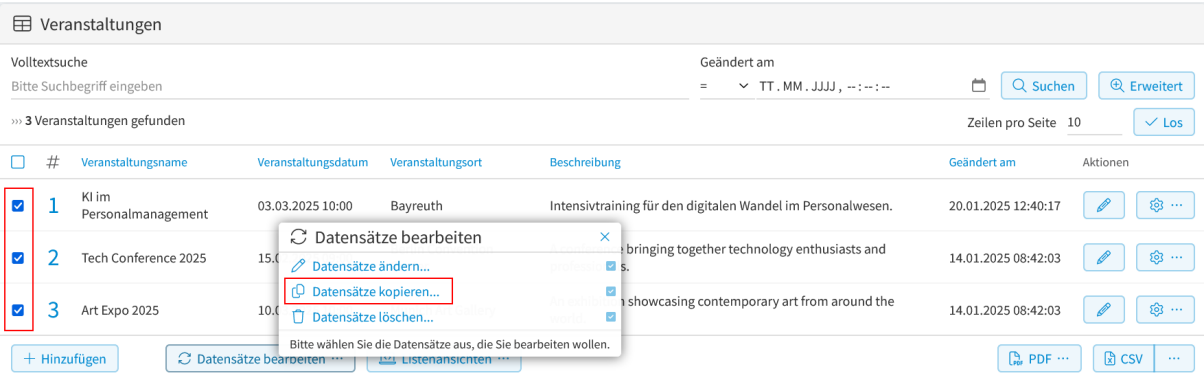

Abb. 11: Datensätze kopieren

- Es öffnet sich ein Fenster mit einer Übersicht aller Felder Ihrer LivingApp.

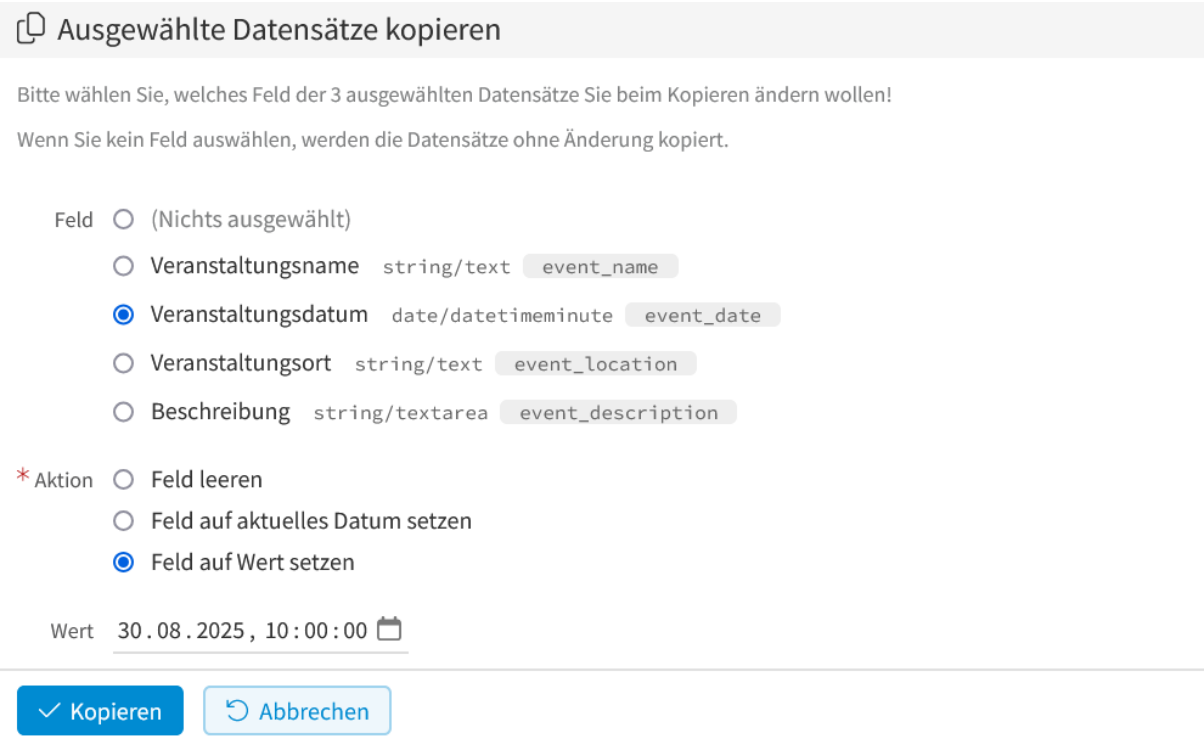

Abb. 12: Feld auswählen

- Wählen Sie hier das Feld aus, dessen Wert in den Kopien Ihrer ausgewählten Datensätze verändert werden soll. In unserem Beispiel wird bei *Feld* „Veranstaltungsdatum“ ausgewählt und bei *Aktion* „Feld auf Wert setzen“. Ergänzen Sie bei *Wert* das gewünschte Datum. Mit Klick auf *Kopieren* werden die Datensätze mit neuem Veranstaltungsdatum angelegt.

  Wählen Sie hier kein Feld, werden die Datensätze ohne Änderung kopiert.

- Genausogut könnten Sie bei allen gewählten Datensätzen das Feld „Veranstaltungsdatum“ in den Kopien leeren, indem Sie bei *Aktion* „Feld leeren“ auswählen.

35.7 Wie kann ich mehrere Datensätze auf einmal löschen?

35.7.1 Voraussetzungen

- Sie haben mindestens Datenverarbeitungsrechte an der LivingApp.

35.7.2 Vorgehen

- Klicken Sie im Menü *Daten* → *Liste* an, um ins Datenmanagement zu gelangen.
- Setzen Sie in der Datenliste das Häkchen bei den Datensätzen, die Sie löschen wollen. Sie können die Datensätze einzeln oder, indem Sie in das Kästchen in der Überschriften-Zeile klicken, alle Datensätze auf dieser Seite oder alle gefundenen Datensätze auswählen.
- Klicken Sie anschließend in der Fußzeile der Datenliste auf *Datensätze bearbeiten…* und wählen *Datensätze löschen*.
- Bestätigen Sie die Löschung im zweiten Schritt nochmal mit Klick auf *Löschen*.

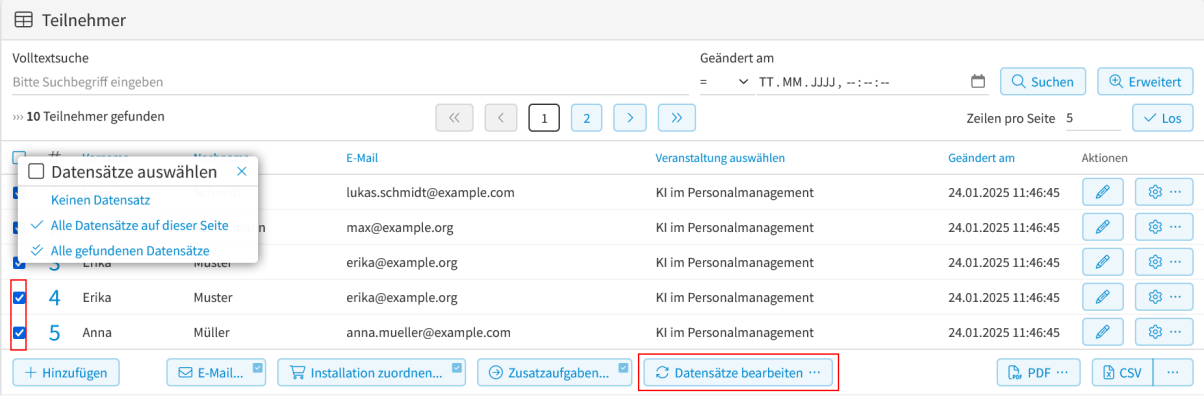

Abb. 13: Datensätze auswählen

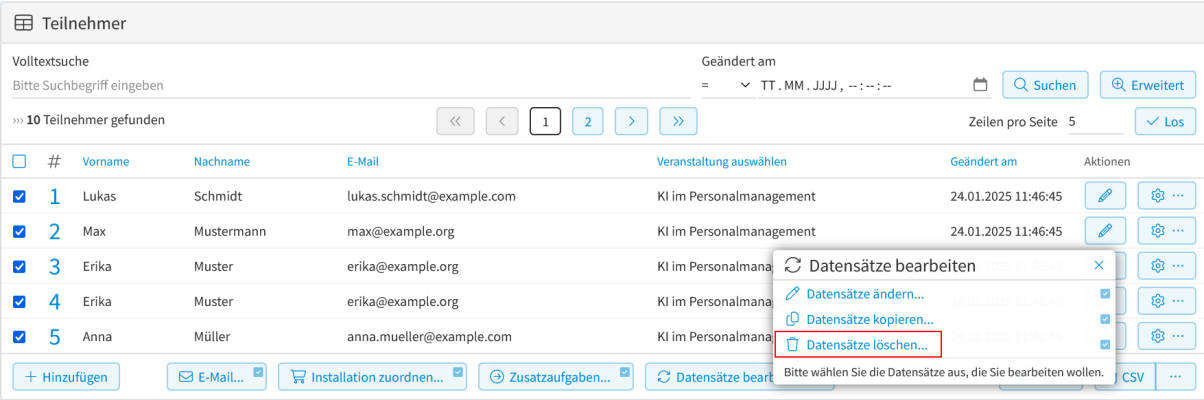

Abb. 14: Datensätze löschen

35.8 Wie kann ich Anhänge (Dateien, URLs, Notizen) für einen Datensatz hochladen?

35.8.1 Voraussetzung

- Sie haben mindestens Datenverarbeitungsrechte an der LivingApp, in der Sie Anhänge hochladen möchten.

35.8.2 Vorgehen

- Wählen Sie die entsprechende LivingApp aus und klicken dort auf *Daten eingeben*. Nachdem Sie Ihre Daten eingegeben und gespeichert haben, erscheint rechts unterhalb vom Eingabeformular das Panel *Anhänge*.

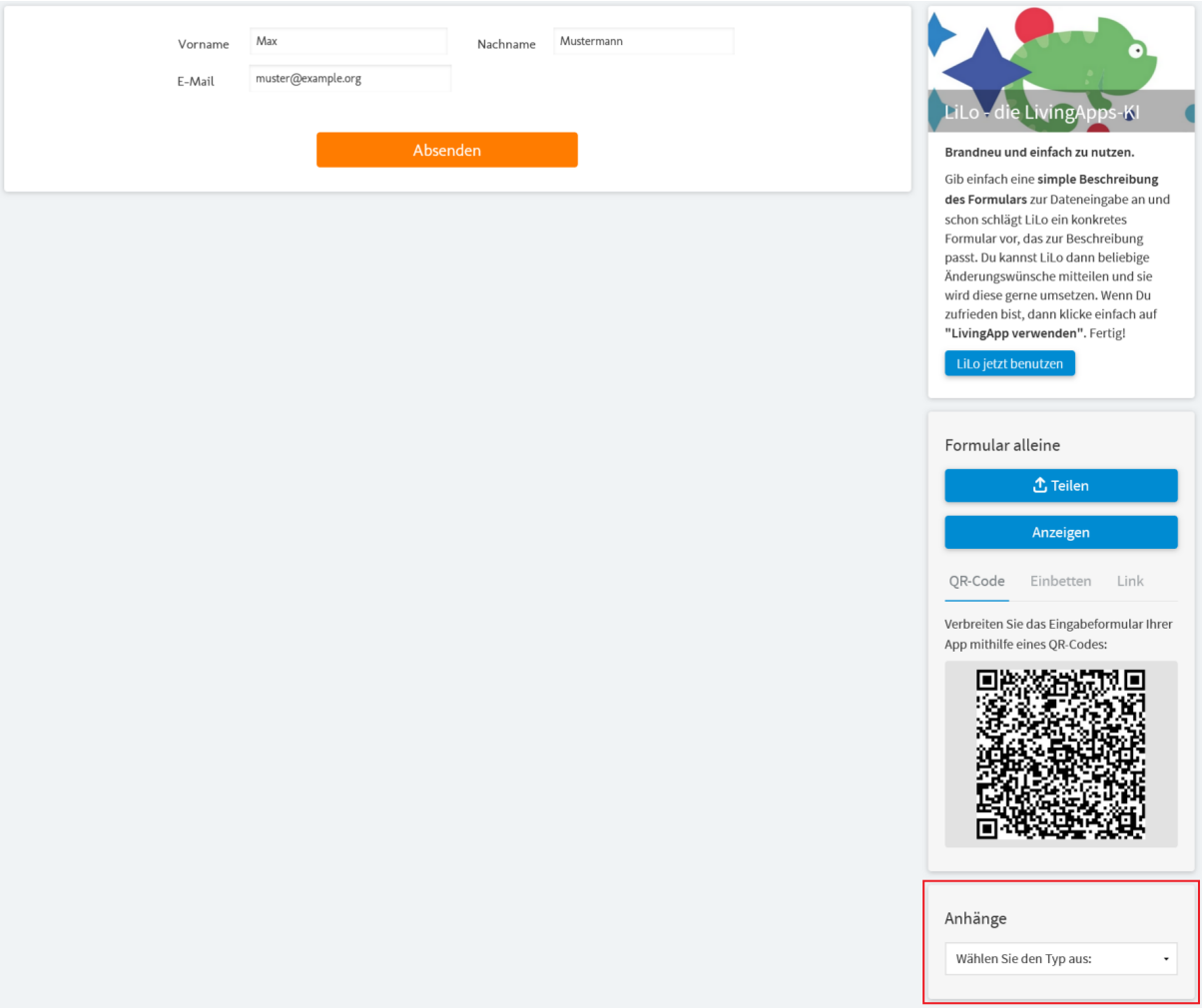

Abb. 15: Anhänge hochladen

- Wählen Sie dort den Typ des Anhangs (Datei, Notiz, URL) aus, füllen die erforderlichen Felder und klicken dann auf *Anhang speichern*.
- Möchten Sie nachträglich einen Anhang hochladen, müssen Sie über die Datenliste gehen. Klicken Sie dafür im Menü *Daten* → *Liste* und dann beim entsprechenden Datensatz auf den Stift zum Bearbeiten.

  Sie gelangen dann wieder in das *Formular* der App. Gehen Sie hier, wie oben beschrieben vor.
- Das *Ergebnis* zeigt eine hochgeladene Datei, eine hochgeladene URL und eine hochgeladene Notiz.
- Die hochgeladenen Anhänge zu einem Datensatz können sie sehen, wenn Sie im Menü auf *Datenliste* → *Liste* und dann beim entsprechenden Datensatz auf den Stift zum Bearbeiten klicken.

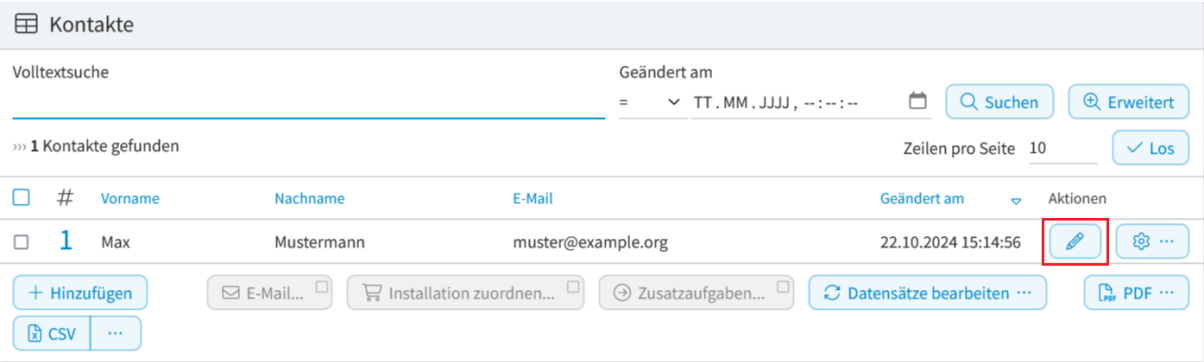

Abb. 16: Anhänge nachträglich hochladen

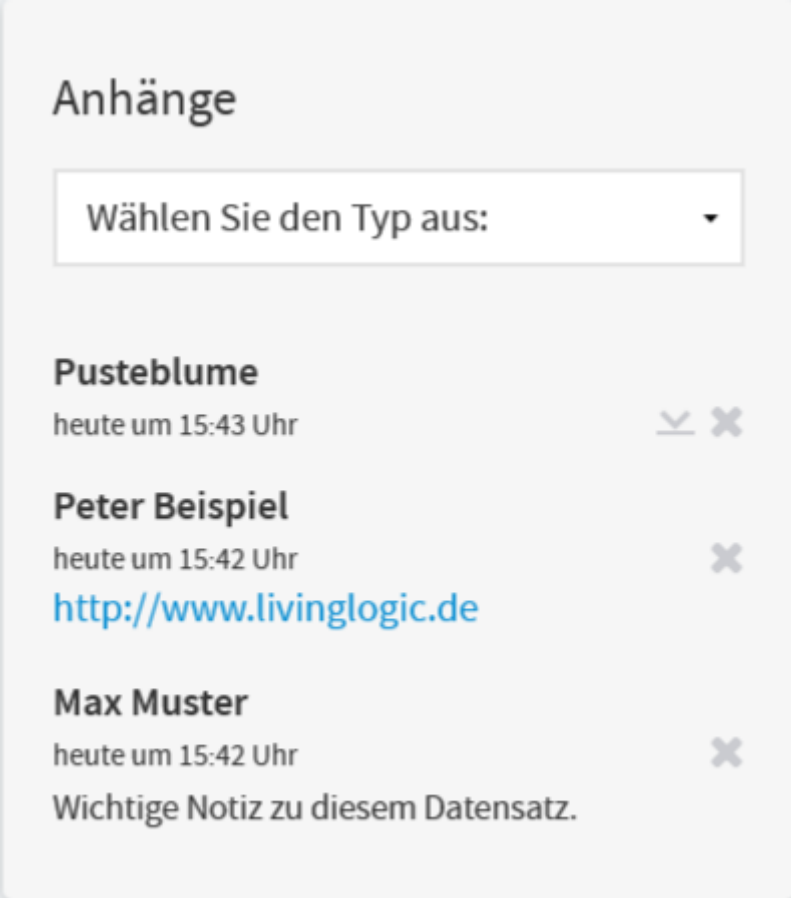

Abb. 17: Ergebnis

- Außerdem können die Anhänge zum Datensatz in Templates ausgegeben werden. Mehr dazu kann in der Dokumentation der *LivingAPI* nachgelesen werden.

35.9 Wie kann ich die Tab-Reihenfolge für die Dateneingabe im Formular ändern?

35.9.1 Voraussetzungen

- Sie haben *Administrationsrechte* an der LivingApp.

35.9.2 Vorgehen

- Klicken Sie im Menü *Konfiguration* → *Eingabe* an, um in den Formulareditor zu gelangen.

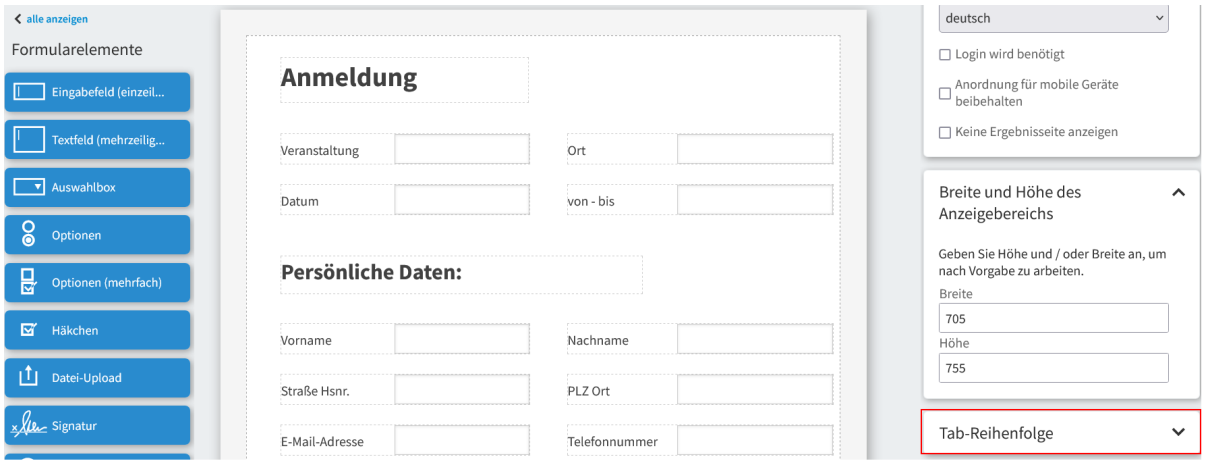

Abb. 18: Tab-Reihenfolge

- Scrollen Sie im rechten Bereich ganz nach unten und klicken auf *Tab-Reihenfolge*. Es öffnet sich eine Ansicht mit allen Feldern Ihrer LivingApp. Die Felder stehen in der Reihenfolge, wie sie bis jetzt vom Tabulator angesprungen werden.
- Möchten Sie diese Reihenfolge ändern, bringen Sie die Felder per Drag & Drop an die passende Position. Sie ändern damit nur die Tab-Reihenfolge, nicht die Position der Felder im Formular.
- Klicken Sie oben links auf *Änderungen übernehmen*.

35.10 Wie kann ich bei Verwendung eines Formulars Feldwerte vorbelegen?

Möchten Sie den Link zu einem Formular verschicken und in diesem Formular Felder mit einem Wert vorbelegen, können Sie dies über URL-Parameter machen.

So können Sie zum Beispiel die URL des Links zum Anmeldeformular für Veranstaltungen durch Parameter so ergänzen, dass im Formular die Veranstaltung bereits ausgewählt und das Feld des Veranstaltungsdatums bereits vorbelegt ist.

35.10.1 Vorgehen

- Wählen Sie im Menü *Daten* → *Eingabe* oder *Daten* → *Formular*, um in das Eingabeformular Ihrer LivingApp zu gelangen.
- Klicken Sie hier in die Zeile der URL und gehen dann ans Ende der URL.

  `https://my.living-apps.de/dateneingabe/677d2b3762fe98bff66ae82a/new?view=677d2b37282d7cfb56a26427`

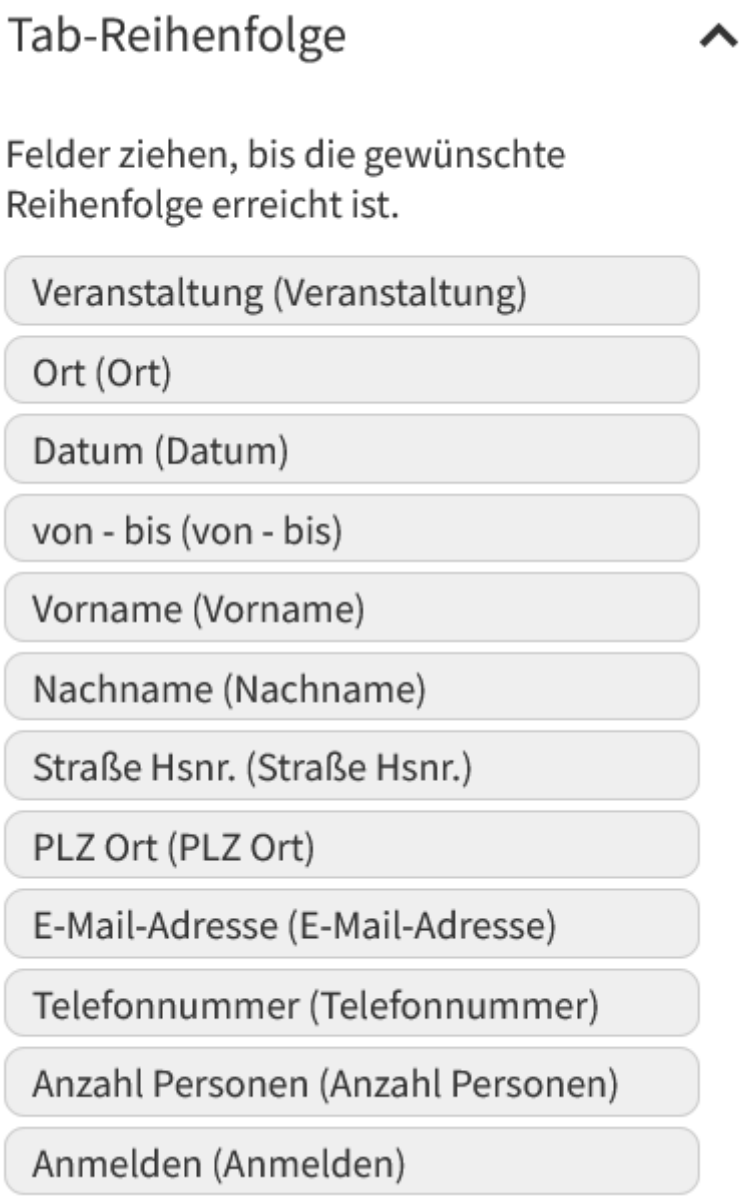

Abb. 19: Tab-Reihenfolge ändern

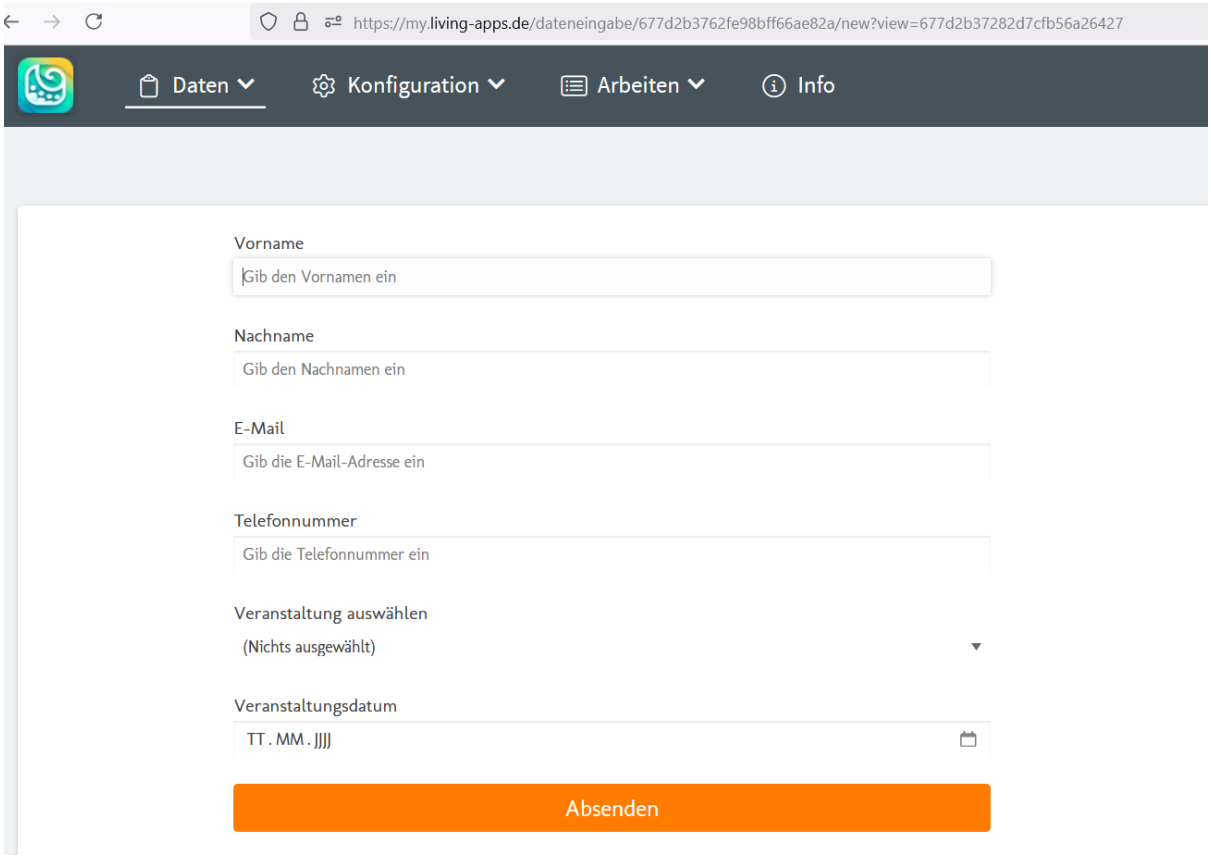

Abb. 20: Eingabeformular

Fügen Sie am Ende der URL folgendes hinzu:

`&event_selection=677d2b434bb3be9487d29695&veranstaltungsdatum=2025-05-15`

Das Ergebnis sieht dann so aus:

Die Veranstaltung ist vorausgewählt und das Datumsfeld vorbelegt.

- Ein URL-Parameter setzt sich zusammen aus dem Identifizierer des Feldes, das Sie vorbelegen möchten und dem Wert des Feldes. Um die Identifizierer der Felder einzusehen, wählen Sie im Menü *Konfiguration* → *Erweitert* unten klicken unten bei *Felder* auf das `i`.

  Es öffnet sich folgendes Fenster:

  Über Felder und deren Identifizierer können Sie in der *Einleitung* der Dokumentation der erweiterten Funktionen mehr nachlesen.

- URL-Parameter werden mit & an die bestehende URL angehängt.
- Im Folgenden wird kurz auf die einzelnen Feldtypen bei Verwendung von URL-Parametern eingegangen.

Vorname

Gib den Vornamen ein

Nachname

Gib den Nachnamen ein

E-Mail

Gib die E-Mail-Adresse ein

Telefonnummer

Gib die Telefonnummer ein

Veranstaltung auswählen

Art & Culture Festival

Veranstaltungsdatum

15 . 05 . 2025

Absenden

Abb. 21: Eingabeformular - Ergebnis

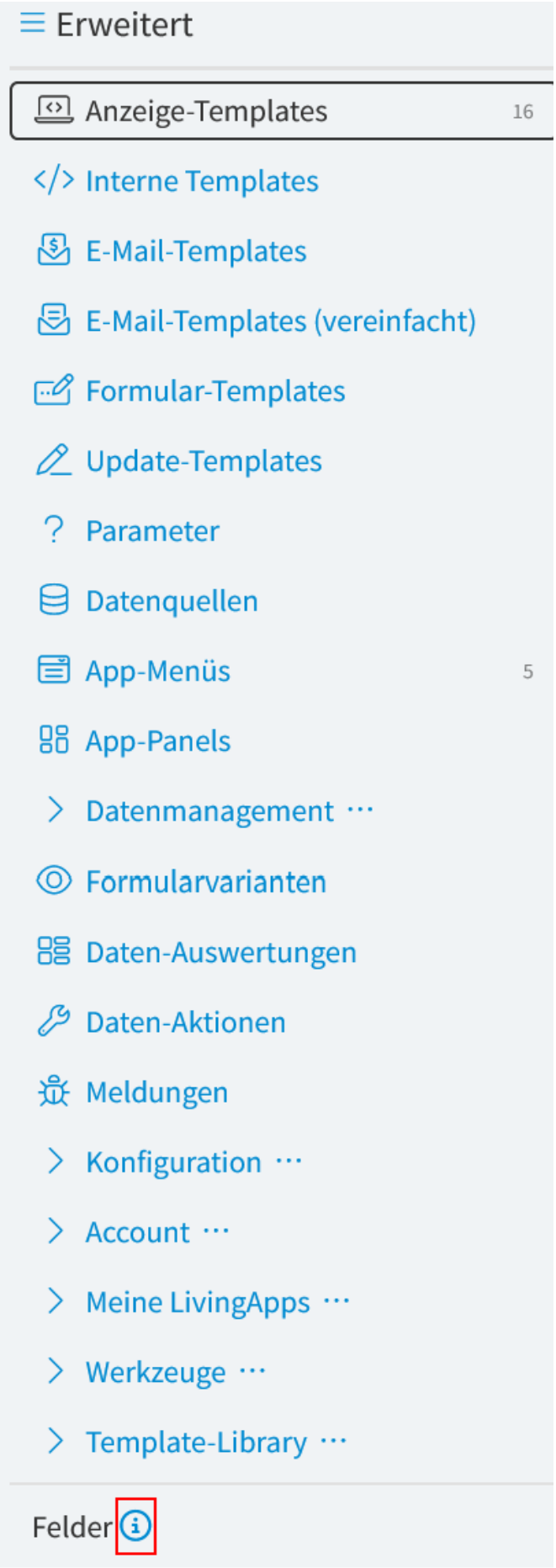

Abb. 22: Felder

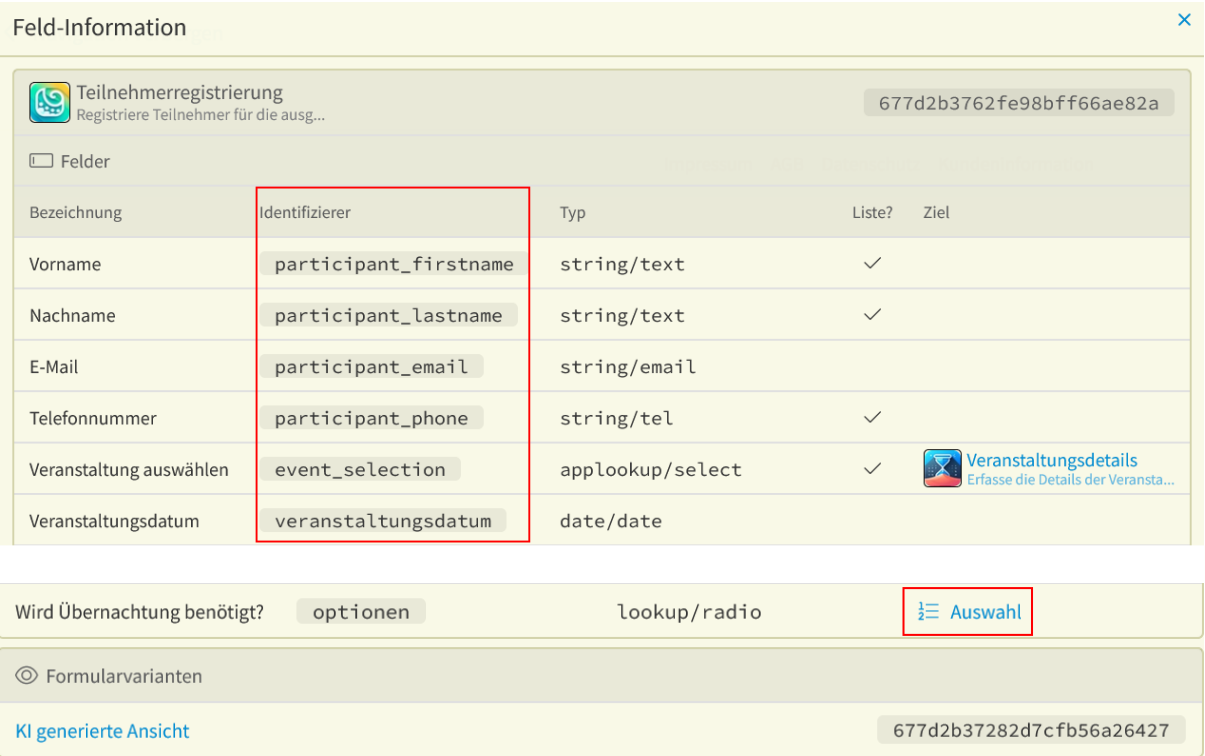

Abb. 23: Identifizierer der Felder

Feldtyp	Beispiel
`bool`	`identifizierer=true`
`string/email`	`identifizierer=beispiel@example.org`
`string/text`	`identifizierer=Text`
`date/date`	`identifizierer=2024-01-06`
`date/datetimeminute`	`identifizierer=2024-01-06 12:30`
`date/datetimesecond`	`identifizierer=2024-01-06 12:30:45`
`lookup/select`	`identifizierer=Identifizierer der Auswahl`
`lookup/radio`	`identifizierer=Identifizierer der Option`
`multiplelookup/checkbox`	`identifizierer=Identifzierer der Option`
`applookup/select`	`identifizierer=Datensatz-ID`

(bei Mehrfachbelegung jeweils weitere Parameter anhängen)

Die Identifizierer der Auswahl und Optionen bei `lookup/select`-, `lookup/radio`- und `multiplelookup/checkbox`-Feldern finden Sie in den *Feldinformationen*, indem Sie neben dem entsprechenden Feld auf *Auswahl* klicken.

Die Datensatz-ID bei Applookup-Feldern können Sie einsehen, indem Sie im Formular der LivingApp das Applookup-Feld mit Rechtsklick anklicken und dann *Untersuchen* auswählen.

Es öffnet sich folgendes Fenster:

Wählen Sie die ID aus, deren Wert im Formular vorbelegt werden soll.

35.11 Welche Möglichkeiten habe ich, Einfluss auf Eingabeformulare zu nehmen?

Über URL-Parameter, die bei allen Formularen verwendet werden können, können Aussehen und Verhalten des Formulars gesteuert werden.

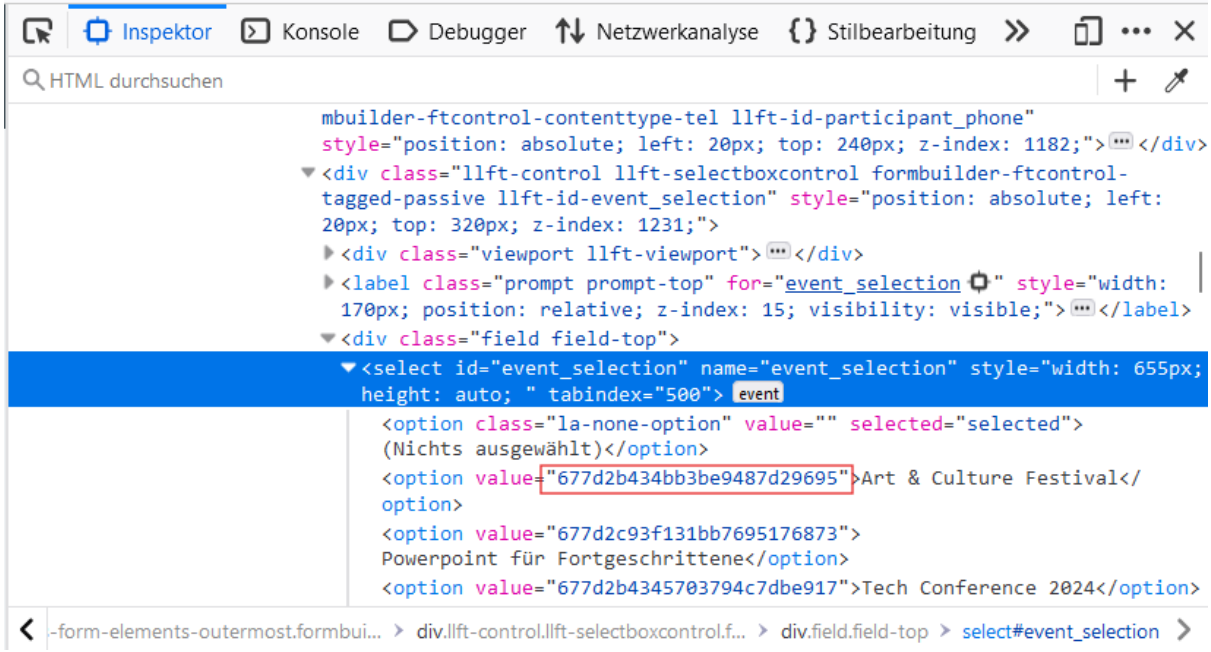

Abb. 24: Applookup-Feld - Datensatz-ID

Folgende Systemparameter stehen zur Auswahl:

`la-header`
: Hierüber kann gesteuert werden, ob das Titel-Bild oben auf der Formularseite angezeigt wird (nur in der Stand-alone-Version). Und falls ja, ob dieses bearbeitet werden darf.

`la-frameless`
: Hierüber kann gesteuert werden, ob die Formularseite (Stand-alone) ohne oder mit umgebenden Rahmen-Tags angezeigt wird.

`la-result-url`
: Hierüber kann gesteuert werden, zu welcher URL man weitergeleitet wird, nach erfolgreicher Formulareingabe.

`responsive`
: Hierüber können Sie festlegen, ob sich der Formularinhalt automatisch an unterschiedliche Bildschirmgrößen anpassen soll.

35.11.1 Vorgehen

- Wählen Sie im Menü *Daten* → *Eingabe* (für die eingebettete Version) oder *Daten* → *Formular* (für die Stand-alone-Version), um in das Eingabeformular Ihrer LivingApp zu gelangen.
- Klicken Sie hier in die Zeile der URL und gehen dann ans Ende der URL.

  `https://my.living-apps.de/dateneingabe/677d2b3762fe98bff66ae82a/new?view=677d2b37282d7cfb56a26427`

  Die Systemparameter werden jeweils mit ? bzw. & an die bestehende URL angehängt.

  `la-header` (nur in der Stand-alone-Version)

  - `la-header=no` (das Titel-Bild wird nicht angezeigt)
  - `la-header=yes` (das Titel-Bild wird angezeigt, kann aber nicht geändert werden)
  - `la-header=edit` oder ohne Parameter (das Titel-Bild wird angezeigt und kann geändert werden)

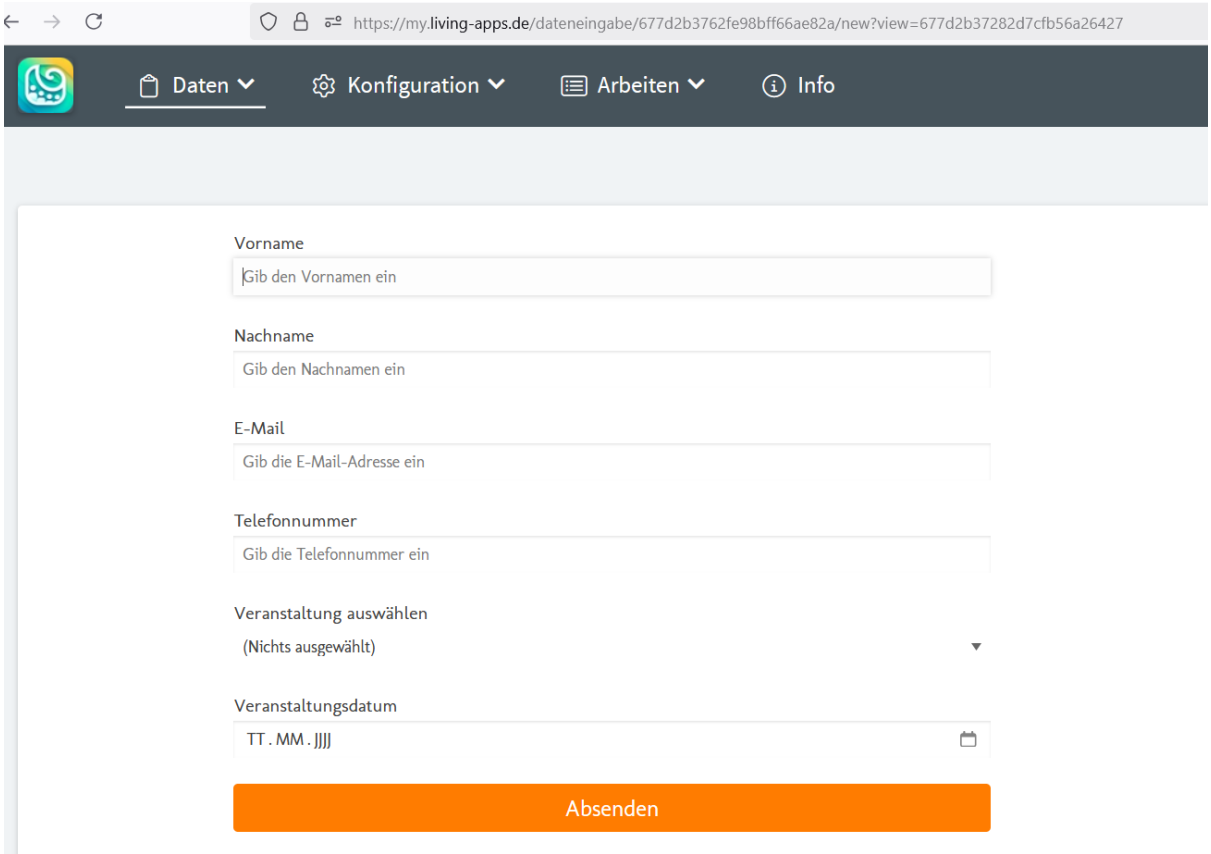

Abb. 25: Eingabeformular (eingebettet)

`la-frameless` (nur in der Stand-alone-Version)
Wird dieser Parameter an die URL angehängt (mit beliebigem Wert), wird das Formular ohne umgebende Rahmen-Tags ausgeliefert. Wird beispielsweise bei der Einbettung in iframes genutzt.

`la-result-url`
Gibt an, zu welcher Seite man nach dem erfolgreichen Speichern des Datensatzes weitergeleitet wird. In diesem Fall werden etwaige Ergebnisseiten-Templates ignoriert, und auch ob im Formulareditor *Keine Ergebnisseite anzeigen* gewählt wurde, oder nicht.

Es sind nur URLs mit dem gleichen Host und Port zulässig.

Sonderzeichen in der URL müssen ersetzt werden, und zwar:

- `:` wird ersetzt durch `%3a`
- `/` wird ersetzt durch `%2f`
- `?` wird ersetzt durch `%3f`
- `&` wird ersetzt durch `%26`
- `=` wird ersetzt durch `%3d`

Beispiel für Weiterleitung nach Speichern des Formulars zur Übersicht der Veranstaltungen:

`la-result-url=https%3a%2f%2fmy.living-apps.de%2fgateway%2fapps%2f677d2b3258bae016fb9173d9%3`

`responsive`

- `responsive=true` (Formularinhalt wird automatisch angepasst)
- `responsive=false` (Formularinhalt wird nicht automatisch angepasst)

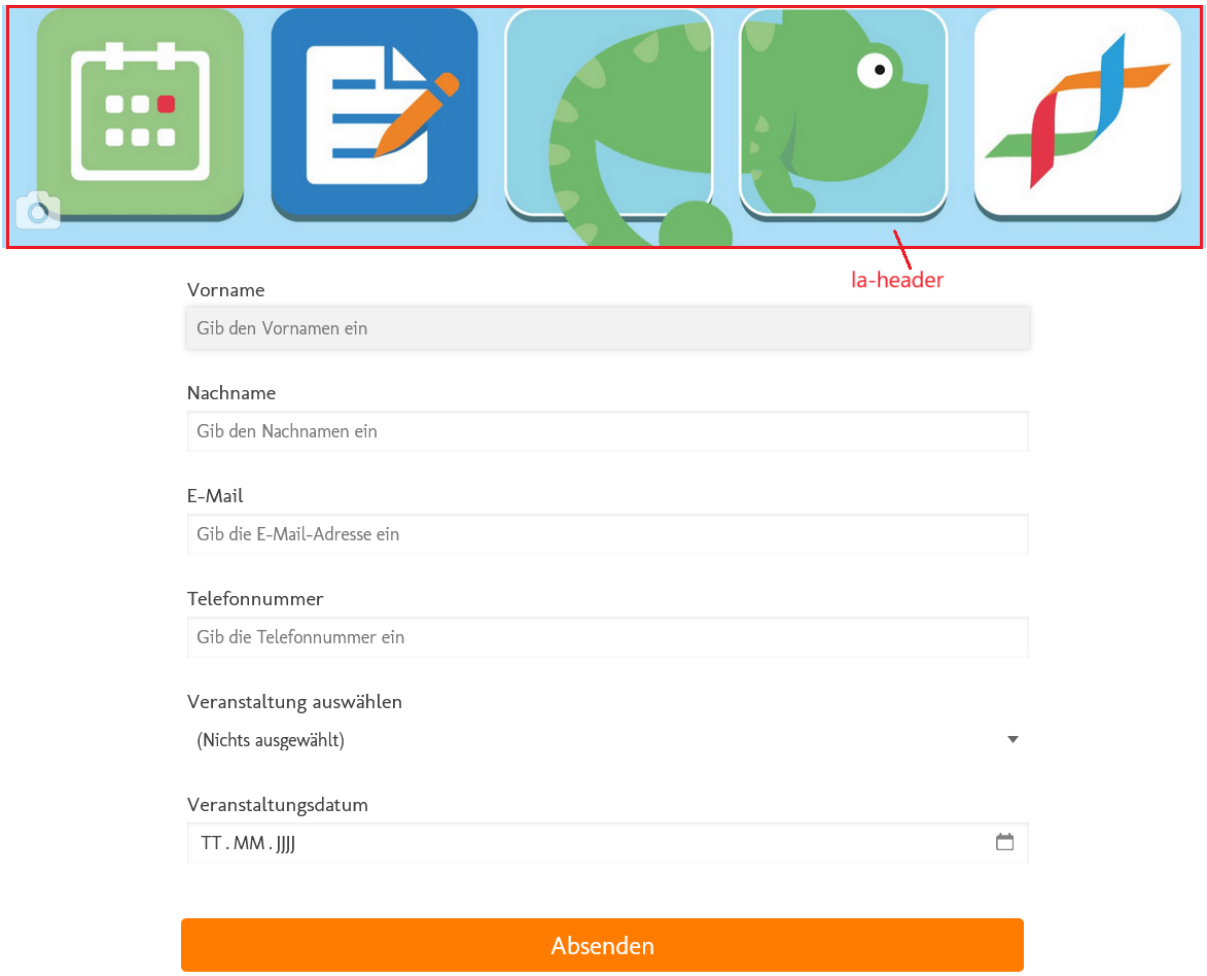

Abb. 26: Eingabeformular (Stand-alone)

35.12 Wie kann ich mehrere zusammengehörige LivingApps kopieren?

35.12.1 Voraussetzungen

- Sie sind Besitzer der LivingApps oder haben *Administrationsrechte* an den LivingApps, die Sie kopieren wollen.

35.12.2 Vorgehen

- Klicken Sie im Menü *Konfiguration* → *Erweitert* an, um in die erweiterte Konfiguration zu gelangen.
- Klicken Sie dort in der Navigation auf *Werkzeuge* → *LivingApps kopieren*.

  Dort können Sie dann die zu kopierenden LivingApps auswählen:

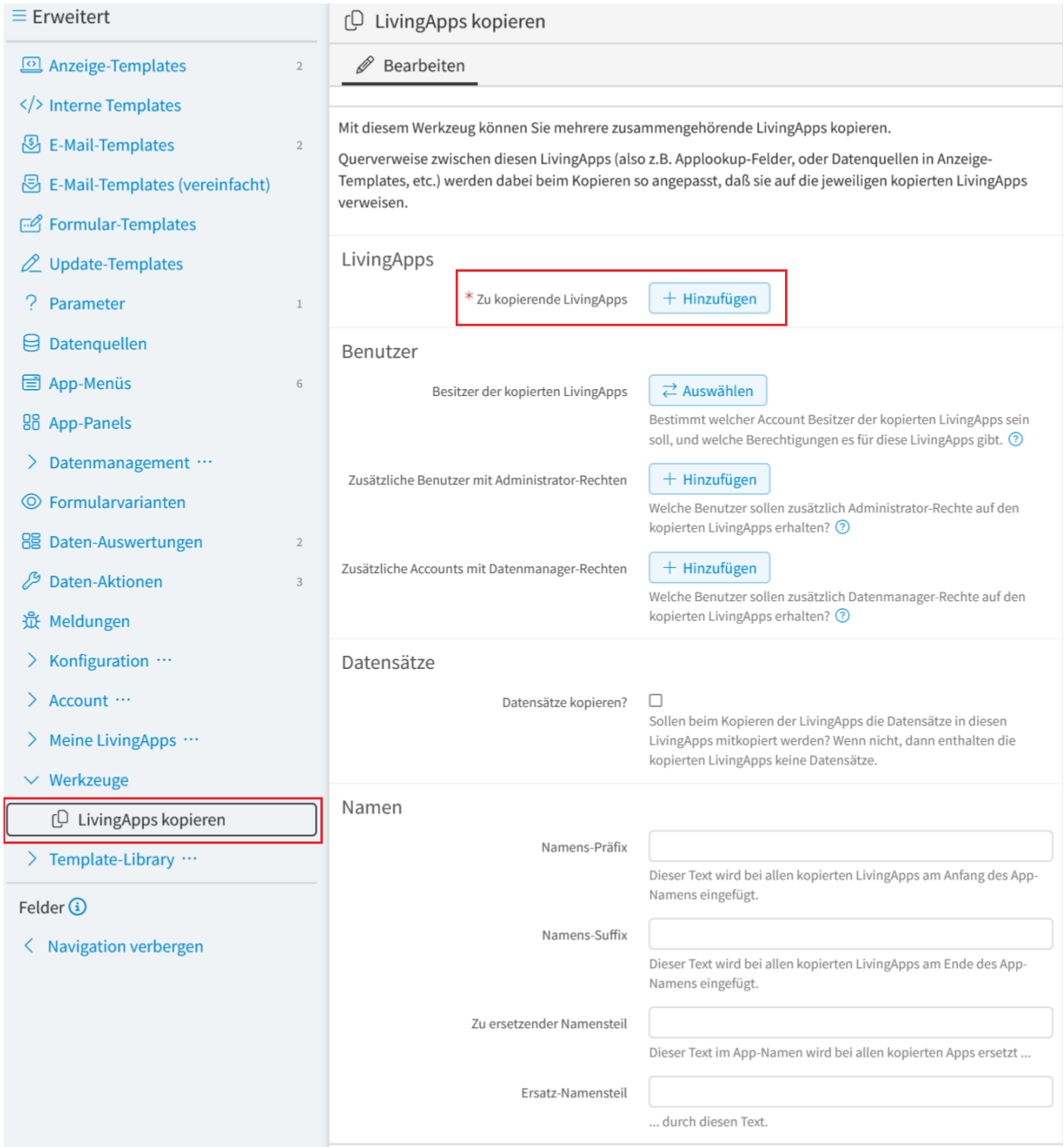

Abb. 27: LivingApps auswählen und übernehmen

- Klicken Sie bei *Zu kopierende LivingApps* auf *Hinzufügen*, um die gewünschten LivingApps auszuwählen und zu übernehmen.
- Unter *Benutzer* kann, wenn Sie die LivingApps für eine andere Person kopieren, der Besitzer der kopierten LivingApps ausgewählt werden. Außerdem können zusätzliche Benutzer mit Administrator-Rechten und zusätzliche Accounts mit Datenmanager-Rechten hinzugefügt werden.
- Sie können die LivingApps mit vorhandenen Datensätzen oder ohne Datensätze kopieren.
- Es wird empfohlen, einen *Namens-Präfix* oder *Namens-Suffix* zu vergeben oder einen vorhanden Namensteil durch einen neuen Namensteil zu ersetzen, damit die kopierten LivingApps von den Ausgangs-LivingApps zu unterscheiden sind.

35.13 Wie kann ich die Bezeichnung eines Feldes im Formular ändern?

35.13.1 Voraussetzungen

- Sie haben *Administrationsrechte* an der LivingApp.

35.13.2 Vorgehen

- Klicken Sie im Menü *Konfiguration* → *Eingabe* an, um in den Formulareditor zu gelangen.

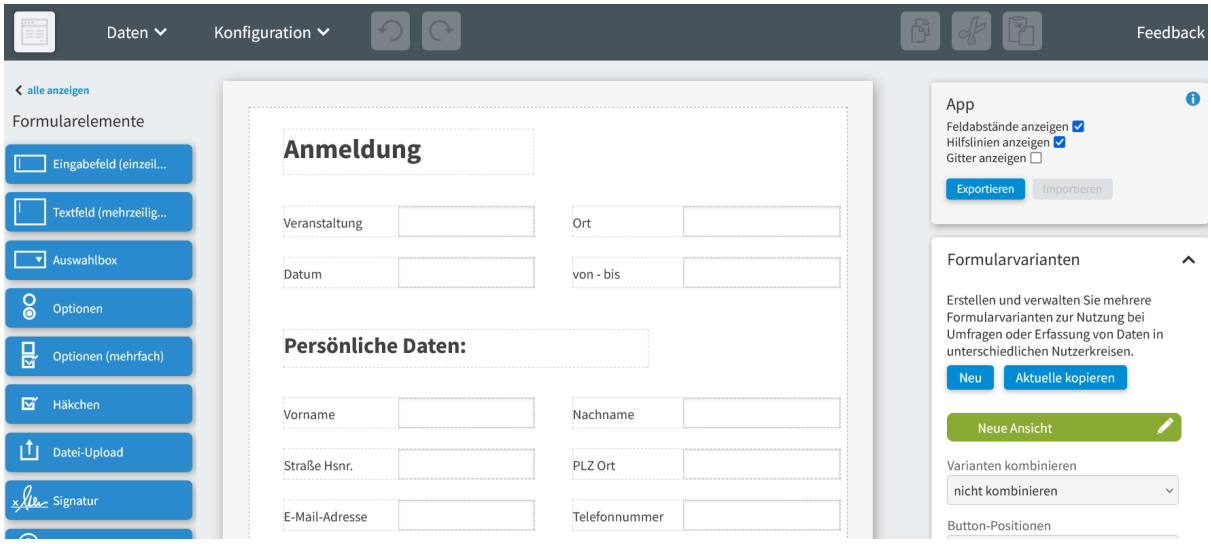

Abb. 28: Formulareditor

- Klicken Sie hier auf das Feld, dessen Bezeichnung Sie ändern möchten. Im rechten Bereich erscheinen die Konfigurationsmöglichkeiten für dieses Feld.
- In unserem Beispiel soll die Bezeichnung des Feldes „von - bis“ geändert werden in „Dauer“. Klicken Sie dafür im rechten Bereich unter *Allgemein* in das Feld *Bezeichnung* und geben die neue Feldbezeichnung ein.
- Die *Bezeichnung für das Datenmanagement* wird automatisch mit geändert. Möchten Sie im Datenmanagement die vorherige Bezeichnung belassen oder ein ganz andere Bezeichnung, geben Sie diese hier manuell ein.
- Klicken Sie oben links auf *Änderungen übernehmen*.
- Die Identifizierer der Felder sind von der Änderung der Bezeichnung nicht betroffen. Diese bleiben so, wie sie bei Erstellung der LivingApp automatisch angelegt wurden.

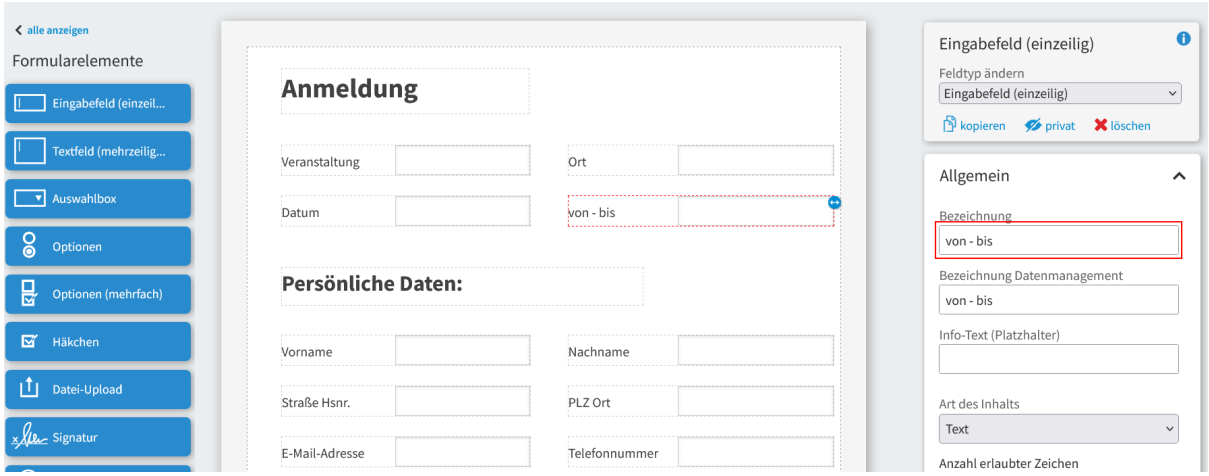

Abb. 29: Feldbezeichnung

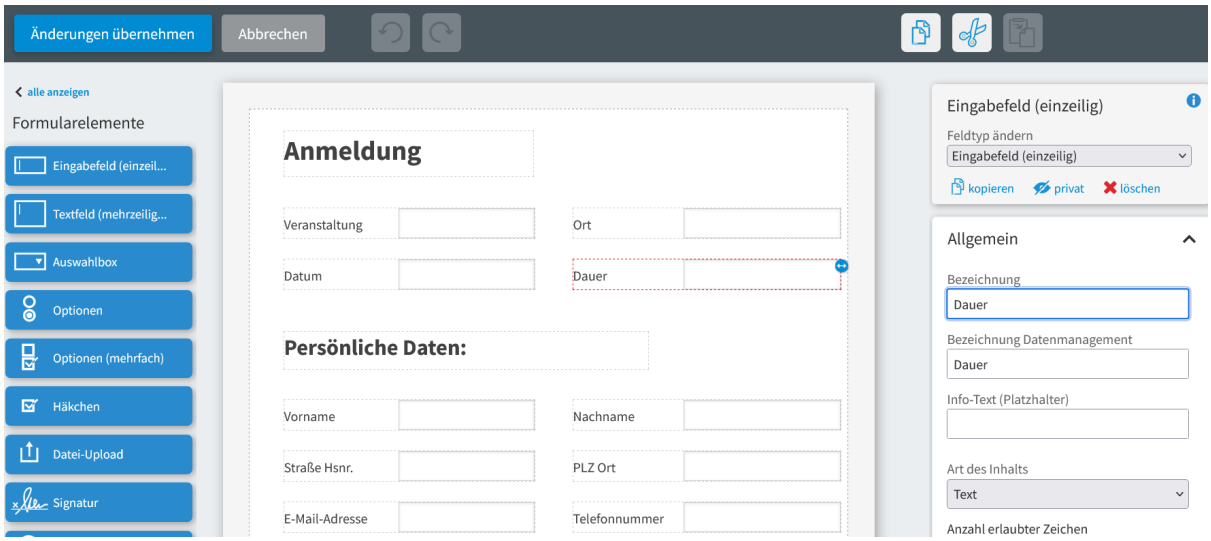

Abb. 30: Feldbezeichnung ändern

35.14 Wie kann ich Uploads in Datensätzen vor dem Zugriff von Dritten schützen?

Gibt es in Ihrer LivingApp ein Upload-Feld über das sensible Daten (z. B. Personaldaten, Gehaltsnachweise, usw.) hochgeladen werden, haben Sie die Möglichkeit dieses Feld vor dem Zugriff von Dritten zu schützen.

35.14.1 Voraussetzungen

- Sie haben *Administrationsrechte* an der LivingApp.

35.14.2 Vorgehen

- Wählen Sie dafür im Menü *Konfiguration* → *Erweitert* und klicken Sie anschließend in der linken Menüleiste auf *Konfiguration* → *Uploads schützen?*.

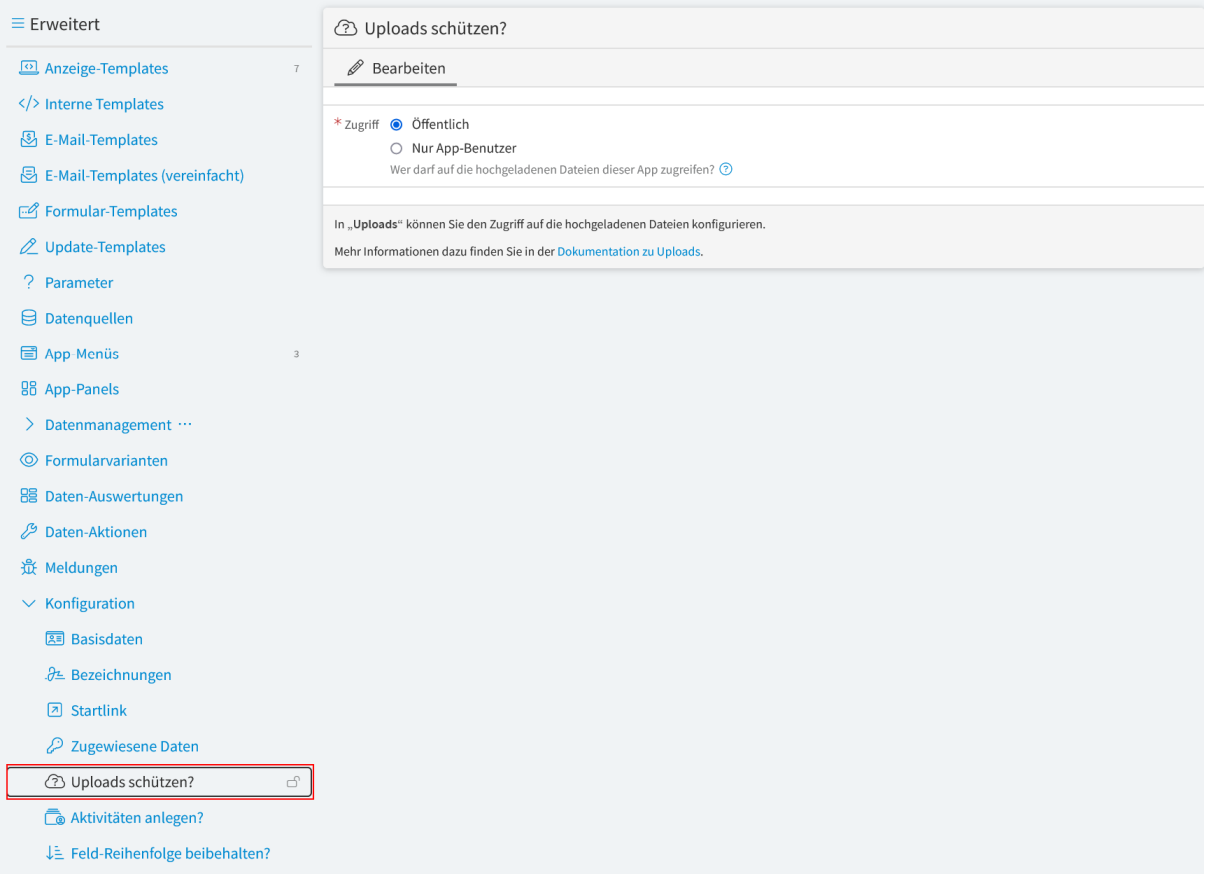

Abb. 31: Uploads schützen?

- Wählen Sie bei *Zugriff* *Nur App-Benutzer*. Die Datei kann dann nur von einem eingeloggten Benutzer heruntergeladen werden, wenn dieser Zugriffsrechte auf der LivingApp besitzt.
- Näheres können Sie in der Dokumentation *Uploads schützen?* nachlesen.

35.15 Wie kann ich E-Mails über das Datenmanagement verschicken?

35.15.1 Voraussetzungen

- Sie haben mindestens Datenverarbeitungsrechte an der LivingApp.
- In Ihrer LivingApp gibt es ein E-Mail-Feld und dieses ist gefüllt.

35.15.2 Vorgehen

- Klicken Sie im Menü *Daten* → *Liste* an, um in die Liste der Datensätze zu gelangen.

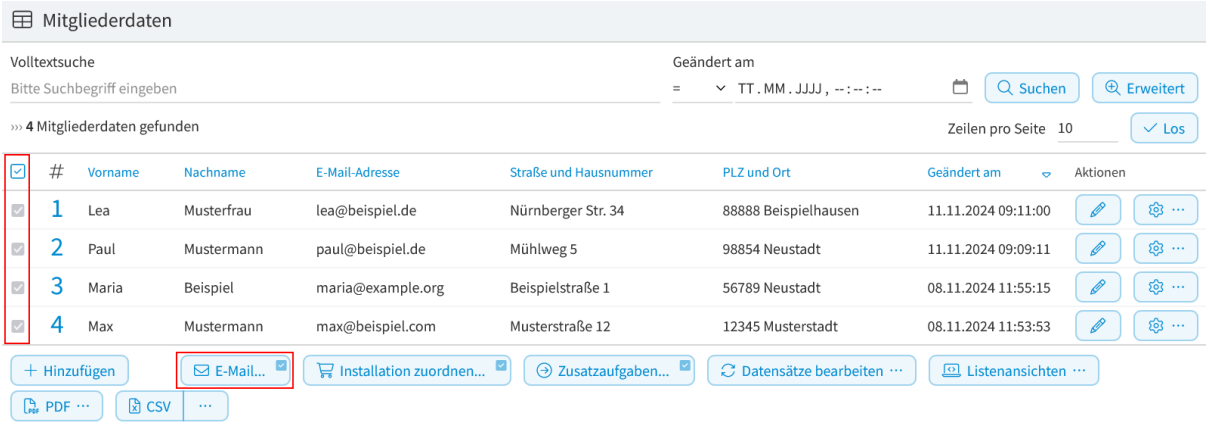

Abb. 32: Auswahl der Datensätze

- Wählen Sie hier, durch Setzen des Häkchens, einzelne oder alle Personen aus, an die Sie die E-Mail schicken wollen und klicken dann unterhalb der Liste auf *E-Mail*.
- Es öffnet sich ein Fenster mit folgenden Konfigurationsmöglichkeiten:

Mailserver
Haben Sie einen eigenen SMTP-Server konfiguriert, können Sie diesen hier auswählen. Wenn Sie hier nichts auswählen, erfolgt der Versand über den internen SMTP-Server von LivingApps.

Empfänger
Gibt es in Ihrer LivingApp mehrere E-Mail-Felder, können Sie hier auswählen, aus welchem E-Mail-Feld die Adresse verwendet werden soll. Es ist nicht möglich, die Adresse manuell einzutragen.

An
Die E-Mail geht an alle Empfänger-Adressen, die Sie in der Liste gewählt haben und deren E-Mail-Feld gefüllt ist.

E-Mail-Inhalt

Vorlage verwenden
Sie haben die Möglichkeit, hier eine Vorlage auszuwählen, um den Betreff und die E-Mail-Nachricht zu übernehmen. Diese Auswahl erscheint nur, wenn bereits eine Vorlage angelegt wurde. Wie Sie eine Vorlage (Versendung) anlegen, können Sie *hier* nachlesen.

E-Mail-Template
Sie haben außerdem die Möglichkeit, den Betreff und die E-Mail-Nachricht aus einem *E-Mail-Template* zu übernehmen, das Sie unter *Konfiguration* → *Erweitert E-Mail-Templates* angelegt haben. Wichtig ist, dass Sie bei der Anlage des E-Mail-Templates bei *Verwendung Vorlage für Nachricht an alle Datensätze* auswählen.

Diese Auswahl erscheint nur, wenn Sie bereits ein E-Mail-Template angelegt haben.

Betreff
Haben Sie eine Vorlage oder ein E-Mail-Template verwendet, ist der Betreff bereits gefüllt, kann aber noch angepasst werden.

Nachricht
Haben Sie eine Vorlage oder ein E-Mail-Template verwendet, ist das Nachrichtenfeld bereits gefüllt, kann aber noch angepasst werden.

Möchten Sie den Nachrichtentext frei gestalten, haben Sie hier auch die Möglichkeit, über *Daten einfügen...* Platzhalter in Ihre Nachricht einzufügen, die beim Versenden ersetzt werden.

Links

E-Mail versenden

Mailserver (Nichts ausgewählt)
Über diesen SMTP-Server wird die E-Mail versendet.

* Empfänger E-Mail-Adressen aus Feld "E-Mail-Adresse"
Wählen Sie hier aus, aus welchem E-Mail-Feld in Ihrer App die E-Mail-Adressen verwendet werden sollen.

An lea@beispiel.de paul@beispiel.de maria@example.org max@beispiel.com
E-Mail-Adressen, an die die E-Mail versendet wird.

E-Mail-Inhalt

Vorlage verwenden (Nichts ausgewählt)
Wählen Sie hier eine bestehende E-Mail-Versendung aus, um deren „Betreff" und „Nachricht" zu übernehmen.

E-Mail-Template (Nichts ausgewählt)
Ein E-Mail-Template erzeugt automatisch den Betreff und der Nachrichteninhalt der E-Mail.

* Betreff

* Nachricht Datei Bearbeiten Ansicht Einfügen Format Werkzeuge Tabelle
Absatz
p 0 Wörter tiny
Daten einfügen

Anhang Durchsuchen... Keine Datei ausgewählt.

Versand

Sofort Zu einem späteren Zeitpunkt nur vorbereiten
Bei „nur vorbereiten" können Sie weitere Datensätze der Versendung zuordnen. Die Versendung kann dann über das „Versende-Archiv" aktiviert werden.

Versendeprotokoll

Mit Hilfe einer Vorlage können Sie die Daten zur versendeten E-Mail aufbewahren und bei einer späteren E-Mail-Versendung wiederverwenden.

* Name Versendung
Unter diesem Namen wird diese Email-Versendung zur Protokollierung und späteren Wiederverwendung abgelegt.

Als Vorlage?
Soll diese Versendung für die Auswahl unter „Vorlage verwenden" zur Verfügung stehen?

Versendung über andere App

Sie können für jeden Adressaten Datensätze in einer Zielapp anlegen und in der E-Mail einen Link zum Bearbeiten- oder Neu-Formular für diesen Datensatz versenden.

Zielapp (Nichts ausgewählt)
Hier können Sie die App auswählen, für die Sie die Versendung durchführen wollen.

Versenden Abbrechen

Abb. 33: Konfiguration der E-Mail

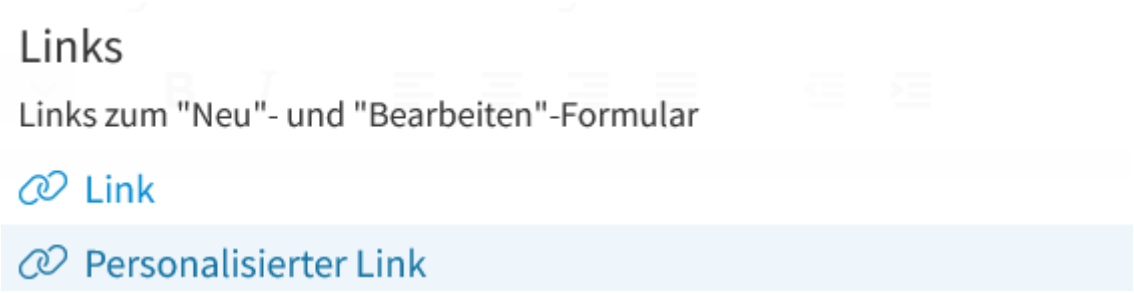

Abb. 34: Platzhalter für Link

Link
Link zum „Neu“-Formular der App, über die Sie E-Mails verschicken. Hier kann ein neuer Datensatz hinzugefügt werden.

Wenn Sie bei *Versenden über eine andere App* eine Zielapp auswählen, wird der Link zum „Neu“-Formular der Zielapp verschickt.

Personalisierter Link
Link zum „Bearbeiten“-Formular der App, über die Sie E-Mails verschicken. Hier können die eigenen Daten geändert werden.

Wenn Sie bei *Versenden über eine andere App* eine Zielapp auswählen, wird der Link zum „Bearbeiten“-Formular der Zielapp verschickt.

Felder

Abb. 35: Platzhalter für Felder des Datensatzes

Der Platzhalter wird gefüllt mit dem jeweiligen Feldinhalt des Datensatzes.

Aktionslinks

Hierfür müssen Sie unter *Konfiguration* → *Erweitert - Aktionen* eine *Aktion* als *Link für E-Mail* angelegt haben. Beim Anklicken dieses Links in der E-Mail wird die Aktion dann ausgeführt.

Abb. 36: Platzhalter für Aktionslinks

Links zu Detailansichten

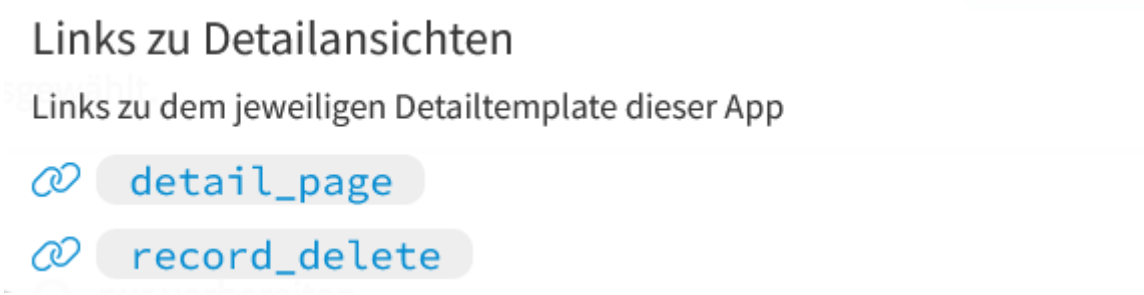

Abb. 37: Platzhalter für Links zu Detailansichten

Haben Sie in Ihrer LivingApp Detail-Templates angelegt, können Sie hier den Platzhalter zum jeweiligen Link auswählen.

Anhang
Hier kann eine Datei als Anhang zur E-Mail hochgeladen werden.

Versand

Sofort
Die E-Mail(s) werden sofort verschickt.

Zu einem späteren Zeitpunkt
Die E-Mail(s) werden zum Zeitpunkt versendet, den Sie hier eingeben.

Abb. 38: Versendezeitpunkt eingeben

Nur vorbereiten
Bei *Nur vorbereiten* können Sie weitere Datensätze der Versendung zuordnen. Die Versendung kann dann über das *Versende-Archiv* aktiviert werden.

Zum Versende-Archiv gelangen Sie, wenn Sie links von der Daten-Liste in der Navigation auf *Versendungen* klicken.

Es öffnet sich folgendes Fenster:

Hier sehen Sie alle von Ihnen erstellten Versendungen. Von hier aus können Sie die vorbereiteten Versendungen losschicken. Außerdem können hier Versendungen im Nachhinein als Vorlage erstellt werden.

Versende-Protokoll

Name
Geben Sie hier den Namen ein, unter dem Sie die Versendung abspeichern wollen.

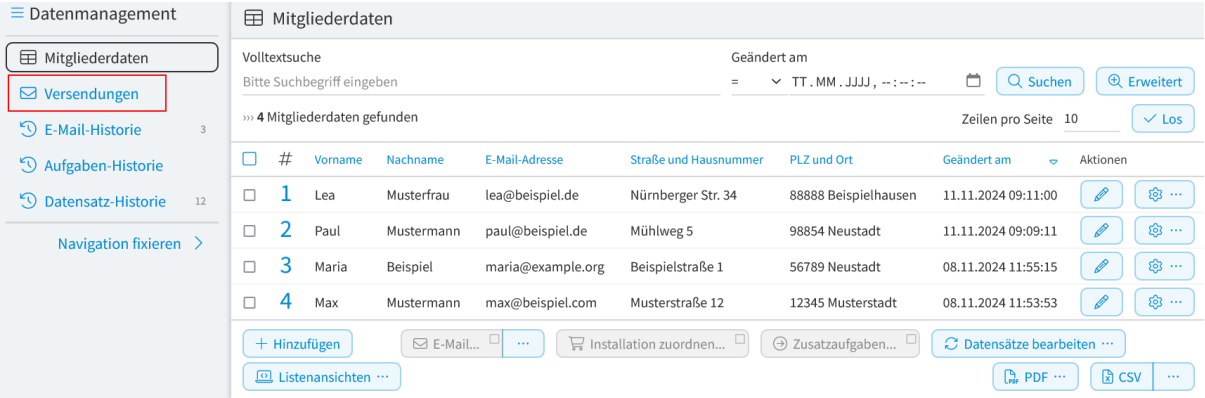

Abb. 39: Versende-Archiv aufrufen

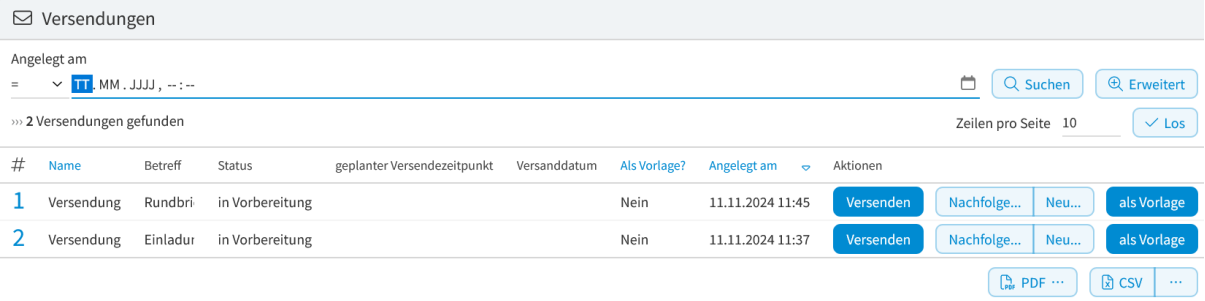

Abb. 40: Versendungen

Als Vorlage?

Möchten Sie die aktuelle E-Mail aufbewahren und sie für spätere E-Mail-Versendungen als Vorlage verwenden, setzen Sie hier das Häkchen. Diese Versendung steht dann als Auswahl unter *Vorlage verwenden* zur Verfügung.

Wie Sie eine Versendung nachträglich als Vorlage erstellen, können Sie bei der Beschreibung des *Versende-Archivs* nachlesen.

Versendung über andere App

Sie können für jeden Adressaten Datensätze in einer Zielapp anlegen und in der E-Mail einen Link zum Bearbeiten- oder Neu-Formular für diesen Datensatz versenden.

Zum Beispiel könnte man hier als Zielapp die App „Vereinsveranstaltungen“ auswählen und das Feld für die Veranstaltung vorbelegen mit der entsprechenden Veranstaltung, zu der man einlädt. Im Nachrichtentext wird der personalisierte Link mitgeschickt. Wenn der E-Mail-Empfänger dann in der E-Mail den Link anklickt, ist schon die Veranstaltung und die E-Mail-Adresse ausgefüllt. Es müssen nur noch die restliche Angaben ergänzt und gespeichert werden. Damit ist die Anmeldung zur Veranstaltung abgeschlossen.

Zielapp

Wählen Sie hier die App, für die Sie die Versendung durchführen wollen.

Daten in Zielapp erzeugen

Sobald eine App ausgewählt wurde, können Sie wählen, ob und welche Daten in der Zielapp erzeugt werden.

Nein

Es werden keine Daten in der Zielapp erzeugt.

Ja

Das Feld, das befüllt wird, sehen Sie bei der ausgewählten Zielapp in Klammern.

Sie haben außerdem die Möglichkeit, für die anzulegenden Datensätze bis zu drei Feldwerte festzulegen, die dann im personalisierten Link bereits ausgefüllt sind und in der Zielapp

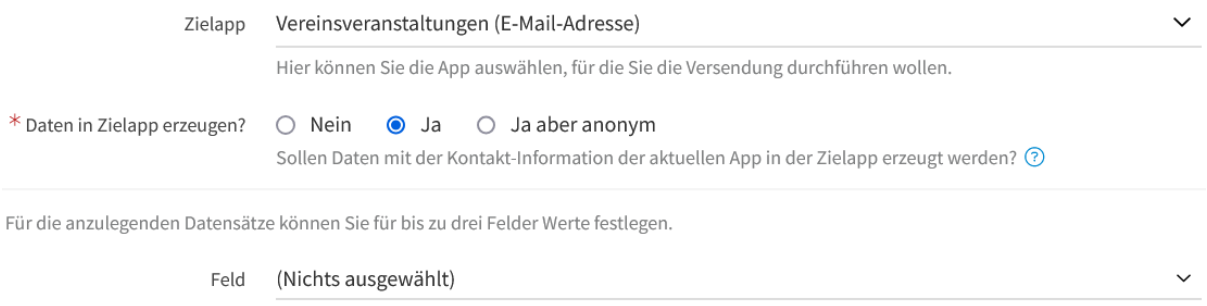

Abb. 41: Versendung über Zielapp

erzeugt werden.

Ja, aber anonym
Das Feld, das befüllt wird, sehen Sie bei der ausgewählten Zielapp in Klammern. Bei *Ja aber anonym* wird in der Zielapp das zugehörige Feld nicht befüllt.

Sie haben außerdem die Möglichkeit, für die anzulegenden Datensätze bis zu drei Feldwerte festzulegen, die dann im personalisierten Link bereits ausgefüllt sind und in der Zielapp erzeugt werden.

35.15.3 Ergebnis

Das Ergebnis soll eine E-Mail-Versendung sein, die an alle Mitglieder geht und zur Mitgliederversammlung einlädt. In dieser E-Mail wird über eine E-Mail-Template-Vorlage die Signatur eingefügt und es wird ein Aktionslink mitgeschickt, über den man den Newsletter abonnieren kann. Die Anrede und der Nachname werden über einen Platzhalter eingefügt.

Zum Schluss wird die Versendung „Mitgliederversammlung“ genannt und als Vorlage angelegt, dass sie für die nächsten Einladungen nur noch ausgewählt werden braucht.

Das Ergebnis mit oben genannten Konfigurationen sieht dann folgendermaßen aus:

35.16 Wie kann ich, wenn ein Hauptdatensatz gelöscht wird, auch alle untergeordneten Datensätze löschen?

35.16.1 Voraussetzungen

- Sie haben *Administrationsrechte* an der LivingApp, die die untergeordneten Datensätze enthält.
- Ihr Account muß als *Experten-User* freigeschaltet worden sein.
- In Ihrer LivingApp gibt es ein `applookup`-Feld.

35.16.2 Vorgehen

- Klicken Sie in der LivingApp, die die untergeordneten Datensätze enthält im Menü *Konfiguration* → *Erweitert* an, um in die erweiterte Konfiguration zu gelangen.
- Klicken Sie dort in der Navigation auf *Konfiguration* → *Felder*.

  Dort sehen Sie dann die Liste der Felder in Ihrer LivingApp:
- Klicken Sie in der Liste das `applookup`-Feld an, für das Sie eine Löschregel festlegen wollen.

  Löschen

  Nicht erlauben
  Das Löschen führt zu einer Fehlermeldung.

E-Mail-Inhalt

Vorlage verwenden: (Nichts ausgewählt)

Wählen Sie hier eine bestehende E-Mail-Versendung aus, um deren „Betreff“ und „Nachricht“ zu übernehmen.

E-Mail-Template: Signatur Vorstandschaft

Ein E-Mail-Template erzeugt automatisch den Betreff und der Nachrichteninhalt der E-Mail.

Betreff: Einladung zur Mitgliederversammlung

Nachricht

Datei Bearbeiten Ansicht Einfügen Format Werkzeuge Tabelle

Absatz

Sehr geehrte(r) {field:anrede} {field:nachname},

hiermit laden wir Sie zur außerordentlichen Mitgliederversammlung

am 30.12.2024, um 19:00 Uhr ganz herzlich ein.

Wir hoffen auf eine rege Teilnahme.

Möchten Sie immer auf dem Laufenden bleiben, dann abonnieren Sie gerne unseren Newsletter

{actionlink:newsletter}.

Mit freundlichen Grüßen

p 44 Wörter tiny

Daten einfügen ···

Abb. 42: Konfiguration E-Mail-Inhalt

Versand

Sofort | Zu einem späteren Zeitpunkt | nur vorbereiten

Bei „nur vorbereiten“ können Sie weitere Datensätze der Versendung zuordnen. Die Versendung kann dann über das „Versende-Archiv“ aktiviert werden.

Versendeprotokoll

Mit Hilfe einer Vorlage können Sie die Daten zur versendeten E-Mail aufbewahren und bei einer späteren E-Mail-Versendung wiederverwenden.

* Name: Mitgliederversammlung

Unter diesem Namen wird diese Email-Versendung zur Protokollierung und späteren Wiederverwendung abgelegt.

Als Vorlage? ☑

Soll diese Versendung für die Auswahl unter „Vorlage verwenden“ zur Verfügung stehen?

Versendung über andere App

Sie können für jeden Adressaten Datensätze in einer Zielapp anlegen und in der E-Mail einen Link zum Bearbeiten- oder Neu-Formular für diesen Datensatz versenden.

Zielapp: (Nichts ausgewählt)

Hier können Sie die App auswählen, für die Sie die Versendung durchführen wollen.

Versenden | Abbrechen

Abb. 43: Konfiguration Versendung

Sehr geehrte(r) Frau Voigt,

hiermit laden wir Sie zur außerordentlichen Mitgliederversammlung

am 30.12.2024, um 19:00 Uhr ganz herzlich ein.

Wir hoffen auf eine rege Teilnahme.

Möchten Sie immer auf dem Laufenden bleiben, dann abonnieren Sie gerne unseren Newsletter

Newsletter abonnieren.

Mit freundlichen Grüßen

--
Verein für mehr Lebensqualität
Muster und Beispiel e. V.
Britta Beispiel
95445 Bayreuth
Telefon: 12345/6789
E-Mail: britta@beispiel.de

Abb. 44: Ergebnis der Versendung

Felder

Bitte Suchbegriff eingeben | Suchen | Erweitert

›› 7 Felder gefunden | Zeilen pro Seite 11 | Los

#		Identifizier	Name	Typ	In Liste?	Gesperrt?	Anzeige	Fußzeile	Löschen	Reihenfolge	Reihenfolge Datenmanagement
1	>	veranstaltung	Verar	string/text	Ja	Nein	Bearbeiten			10	
2	>	ort	Ort	string/text	Ja	Nein	Bearbeiten			20	
3	>	datum	Datur	date/date	Ja	Nein	Bearbeiten			30	
4	>	von_bis	von -	string/text	Ja	Nein	Bearbeiten			40	
5	>	e_mail_verteiler	E-Mai	string/email	Ja	Nein	Bearbeiten			50	
6	>	icon	Icon	file	Nein	Nein	Bearbeiten			60	
7	>	zustaendiger_mitarbeiter	Zustä	applookup/select	Nein	Nein	Bearbeiten		Referenz löschen	-99990	

CSV | ···

Unter „**Felder**“ können Sie einstellen, welche Felder im Datenmanagement angezeigt werden.

Mehr Informationen dazu finden Sie in der Dokumentation zu Feldern im Datenmanagement.

Abb. 45: Liste der Felder

Felder zustaendiger_mitarbeiter Name Zuständiger Mitarbeiter Typ applookup/choice 7/7

Bearbeiten Interne Templates

Speichern Wiederherstellen

Identifizier zustaendiger_mitarbeiter
Eine eindeutige Kennung für dieses Feld. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

Name Zuständiger Mitarbeiter

Typ applookup/choice

In Liste? ☑
Soll das Feld im Datenmanagement in der Liste angezeigt werden (wenn bei „**Anzeige**“ nicht „**Versteckt**“ gewählt wurde)?

Gesperrt? ☐
Ein gesperrtes Feld kann im Formular-Editor nicht gelöscht werden und dessen Typ kann nicht geändert werden.

* Anzeige ◉ Bearbeiten ○ Anzeigen ○ Verstecken
Soll das Feld im Datenmanagement angezeigt werden, bearbeitet werden können, oder komplett versteckt werden? Dies betrifft sowohl die Listen- als auch die Detailansicht.

* Löschen ○ Nicht erlauben ◉ Diesen Datensatz löschen ○ Referenz löschen
Was soll passieren, wenn der referenzierte Datensatz gelöscht wird?

Reihenfolge 70
Die ursrpüngliche Reihenfolge, in der die Felder angelegt wurden.

Reihenfolge Datenmanagement 70
Wenn Sie hier einen von „**Reihenfolge**“ abweichenden Wert eingeben, behält LivingApps im Datenmanagement die dadurch festgelegte Reihenfolge bei.

Abb. 46: Feldkonfiguration

Diesen Datensatz löschen
Beim Löschen des referenzierten Datensatzes wird dieser Datensatz ebenfalls gelöscht.

Referenz löschen
Beim Löschen des referenzierten Datensatzes wird diese Referenz gelöscht (d.h. dieses Feld referenziert dann keinen Ziel-Datensatz mehr).

Im Beispiel ist die LivingApp „Kundendaten“ über das Feld „Zuständiger Mitarbeiter“ mit der LivingApp „Mitarbeiter“ verknüpft.

Möchten Sie alle untergeordneten Datensätze (Kundendaten) löschen, wenn ein Hauptdatensatz (Mitarbeiter) gelöscht wird, wählen Sie *Diesen Datensatz löschen*.

35.17 Wie kann ich im Datenmanagement für Zahlenfehler eine Summe anzeigen lassen?

35.17.1 Voraussetzungen

- Sie haben *Administrationsrechte* an der LivingApp.
- Ihr Account muß als *Experten-User* freigeschaltet worden sein.
- In Ihrer LivingApp gibt es ein Zahlenfeld.
- Dieses Zahlenfeld ist so konfiguriert, daß es im Datenmanagement in der Liste angezeigt wird.

35.17.2 Vorgehen

- Klicken Sie im Menü *Konfiguration* → *Erweitert* an, um in die erweiterte Konfiguration zu gelangen.
- Klicken Sie dort in der Navigation auf *Konfiguration* → *Felder*.

  Dort sehen Sie dann die Liste der Felder in Ihrer LivingApp:

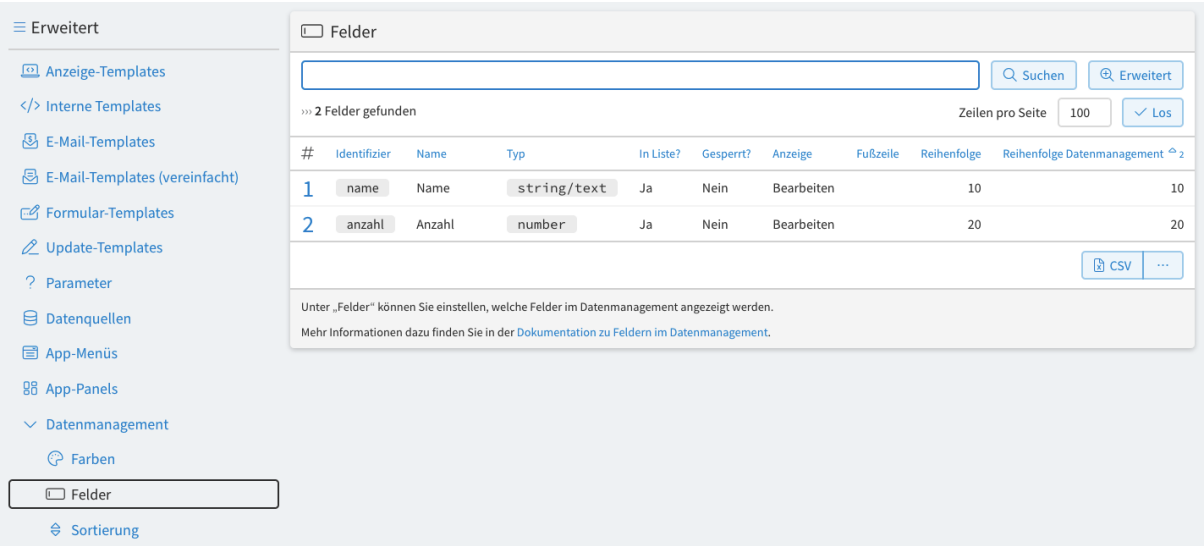

Abb. 47: Liste der Felder

- Klicken Sie in der Liste das Zahlenfeld an, für das Sie die Summe anzeigt haben wollen. Stellen sie bei *Fußzeile* von *(Nichts ausgewählt)* um auf *Summe*.
- Klicken Sie anschließend auf *Speichern*.
- Wenn Sie im Menü über *Daten* → *Liste* zurück zum Datenmanagement wechseln, sollten Sie in der Spalte Ihres Zahlenfelds unten die Summe sehen:

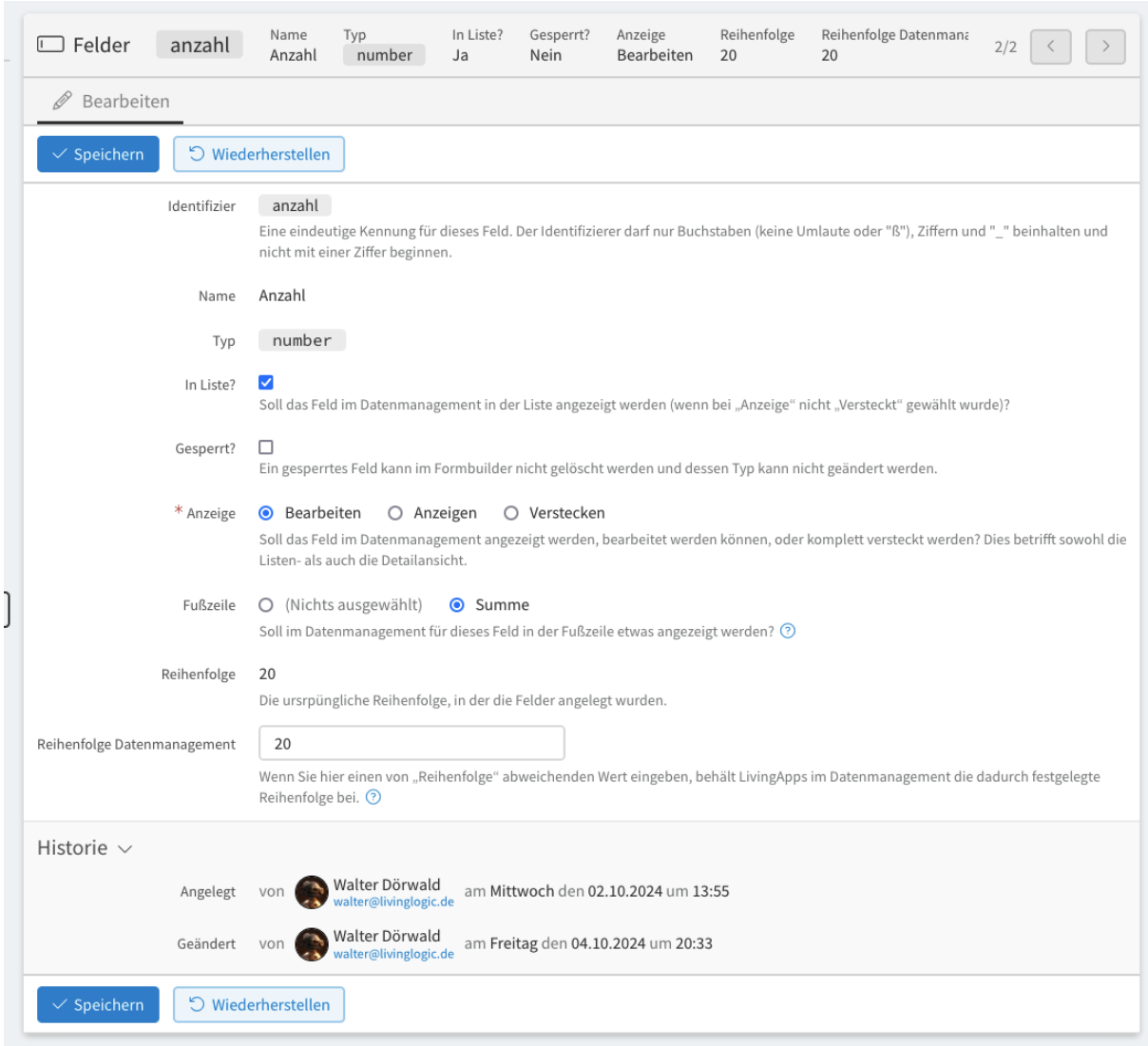

Abb. 48: Feld-Konfiguration

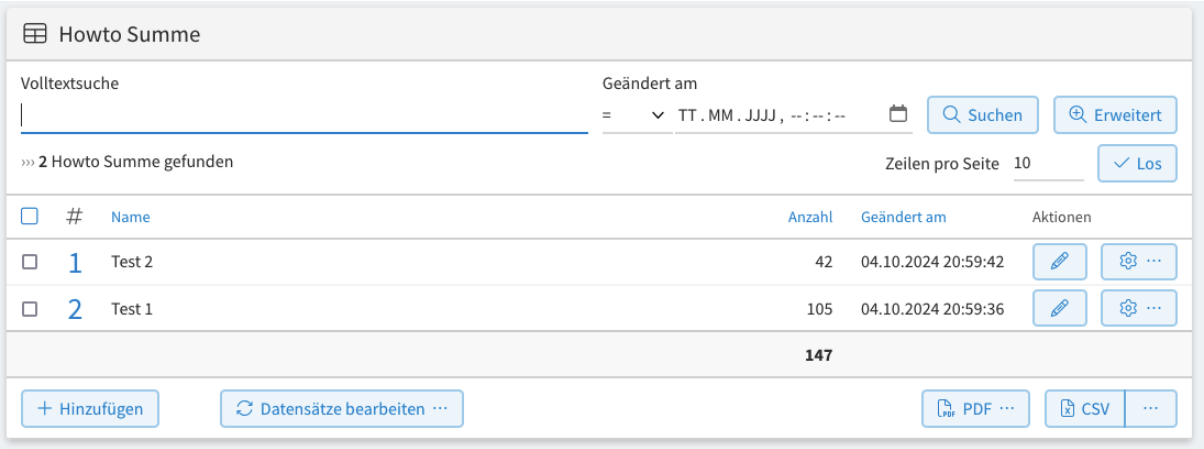

Abb. 49: Ergebnis

35.18 Wie kann ich Interne Templates und Parameter vererben?

Interne Templates und Parameter von einer anderen LivingApp zu erben ermöglicht es Ihnen, in Ihrer eigenen erbenden LivingApp die Internen Templates und Parameter der Basis-LivingApp so zu verwenden, als wären es Ihre eigenen. Insbesondere ermöglicht Ihnen dies, einen gemeinsamen Satz von Internen Templates und Parametern aus einer Basis-LivingApp in mehreren anderen LivingApps zu verwenden.

Nähere Informationen zur Vererbungslogik finden Sie unter *Überschreiben eines Templates* und unter *Überschreiben eines Parameters*.

35.18.1 Voraussetzungen

- Sie haben *Administrationsrechte* sowohl an der Basis-LivingApp, als auch an der erbenden LivingApp.
- Ihr Account muß als *Experten-User* freigeschaltet worden sein.

35.18.2 Vorgehen

- Klicken Sie in der erbenden LivingApp im Menü *Konfiguration* → *Erweitert* und dann in der linken Menüleiste auf *Parameter*.

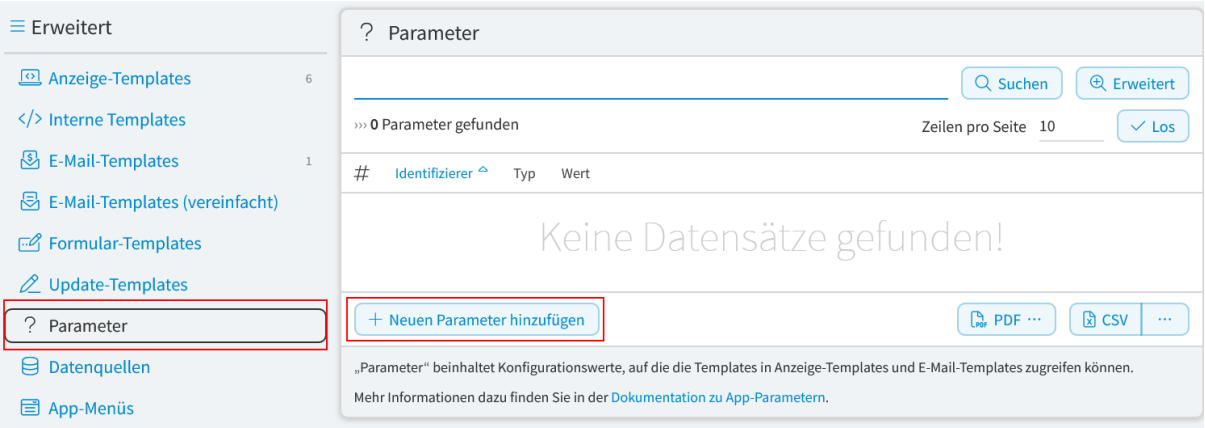

Abb. 50: Parameter hinzufügen

- Klicken Sie auf *Neuen Parameter hinzufügen*.

Identifizierer
: Geben Sie hier `1a` ein.

Typ
: Wählen Sie hier `App` aus.

Wert
: Kein Wert

App
: Wählen Sie hier die Basis-LivingApp aus, von der Sie die Internen Templates und Parameter übernehmen wollen.

Beschreibung
: Hier können Sie eine Beschreibung eingeben.

- Mit Klick auf *Speichern* ist die Erstellung abgeschlossen und die Internen Templates und Parameter der unter *App* ausgewählten Basis-LivingApp werden an die erbende LivingApp vererbt.

? Parameter Typ: App

+ Hinzufügen

✓ Speichern | ✓ Speichern und zur Übersicht | ↺ Abbrechen | ⓘ Hilfe

* Identifizierer: la

Die eindeutige Kennung für diesen Parameter. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen. ⓘ

* Typ
- ○ BOOL (Wahrheitswert)
- ○ INT (Ganze Zahl)
- ○ NUMBER (Zahl)
- ○ STR (Text)
- ○ HTML (HTML-Text)
- ○ COLOR (Farbe)
- ○ DATE (Datum)
- ○ DATETIME (Datum/Uhrzeit)
- ○ DATEDELTA (Zeitraum in Tagen)
- ○ DATETIMEDELTA (Zeitraum in Tagen/Stunden/Minuten/Sekunden)
- ○ MONTHDELTA (Zeitraum in Monaten)
- ○ LIST (Liste)
- ○ DICT (Dictionary)
- ○ Upload
- ◉ App
- ○ Control

Wert

App: Anmeldung Computerkurse | × Entfernen | ⇄ Auswählen

Beschreibung

✓ Speichern | ✓ Speichern und zur Übersicht | ↺ Abbrechen | ⓘ Hilfe

Abb. 51: Parameter konfigurieren

35.19 Wie kann ich in Anzeige-Templates Font Awesome-Icons verwenden?

35.19.1 Voraussetzungen

- Sie haben *Administrationsrechte* an der LivingApp.

35.19.2 Vorgehen

Wenn Sie in Ihren Templates das Library-Template `la_standard_page` verwenden werden die Font Awesome-Icons automatisch eingebunden. Dies sieht z.B. so aus:

```
<?renderblock globals.t_la_standard_page(title="Titel")?>
   <h1>Überschrift</h1>
   <p>Inhalt mit Icon: <i class="fas fa-paw fa-fw fa-2x"></i></p>
<?end renderblock?>
```

Wenn Sie die Icons selber einbinden wollen, können Sie das Library-Template `la_static_font_awesome` folgendermaßen verwenden:

```
<?render globals.t_la_static_font_awesome()?>
```

Dies erzeugt folgenden Link:

```
<link href="https://my.living-apps.de/npm/@fortawesome/fontawesome-free@6.7.2/css/all.
↪min.css" rel="stylesheet">
```

Zur Anzeige eines Icons können Sie wie gewohnt folgendes HTML verwenden:

```
<i class="fas fa-paw fa-fw fa-2x"></i>
```

Sie können jedoch auch das Library-Template `la_icon` verwenden. Dieses Template bietet die Möglichkeit einen Standard-Icon-Stil zu verwenden, den Sie über den Parameter `la_default_faicon_style` festlegen können. Wollen Sie für alle Ihre Icons einen anderen Stil verwenden, müssen Sie nur den Wert des Parameters ändern.

Der Aufruf des Templates sieht beispielsweise so aus:

```
<?render globals.t_la_icon("paw", fw=True, cssclass="fa-2x")?>
```

Dies gibt das selbe HTML aus, wie in obigem Beispiel:

```
<i class="fas fa-paw fa-fw fa-2x"></i>
```

35.19.3 Standard-Icon-Stil

Der Standard-Stil ist `classic/solid`. Um einen anderen Standard-Stil zu konfigurienen können Sie den Parameter `la_default_faicon_style` in Ihrer LivingApp setzen.

Gehen Sie dazu folgendermaßen vor:

- Wählen Sie im Menü *Konfiguration* → *Erweitert* und klicken anschließend in der linken Menüleiste auf *Parameter*.
- Klicken Sie dann auf den Button *Neuen Parameter hinzufügen*.
- Tippen Sie im nun geöffneten Fenster bei *Identifizierer* den Text `la_default_faicon_style` ein.
- Bei *Typ* muss *STR (Text)* ausgewählt werden.
- Tippen Sie bei *Wert* den gewünschten Stil-Namen ein.

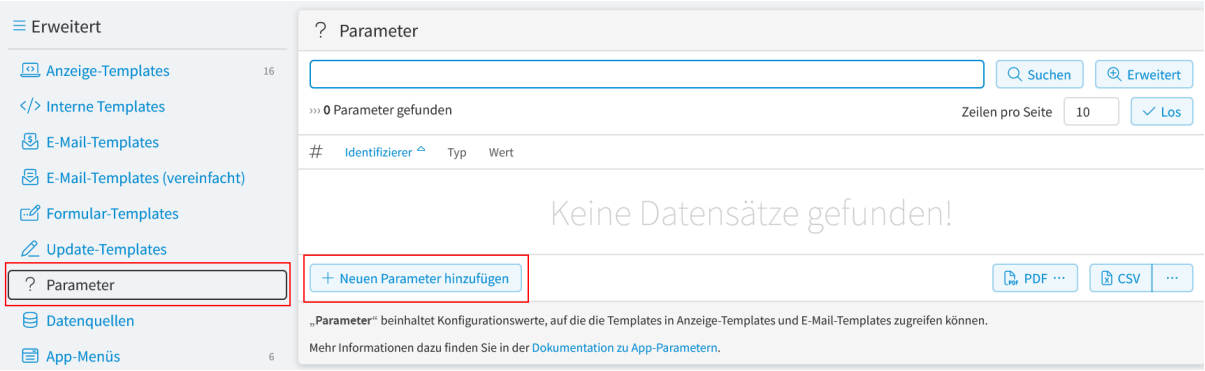

Abb. 52: Parameter hinzufügen

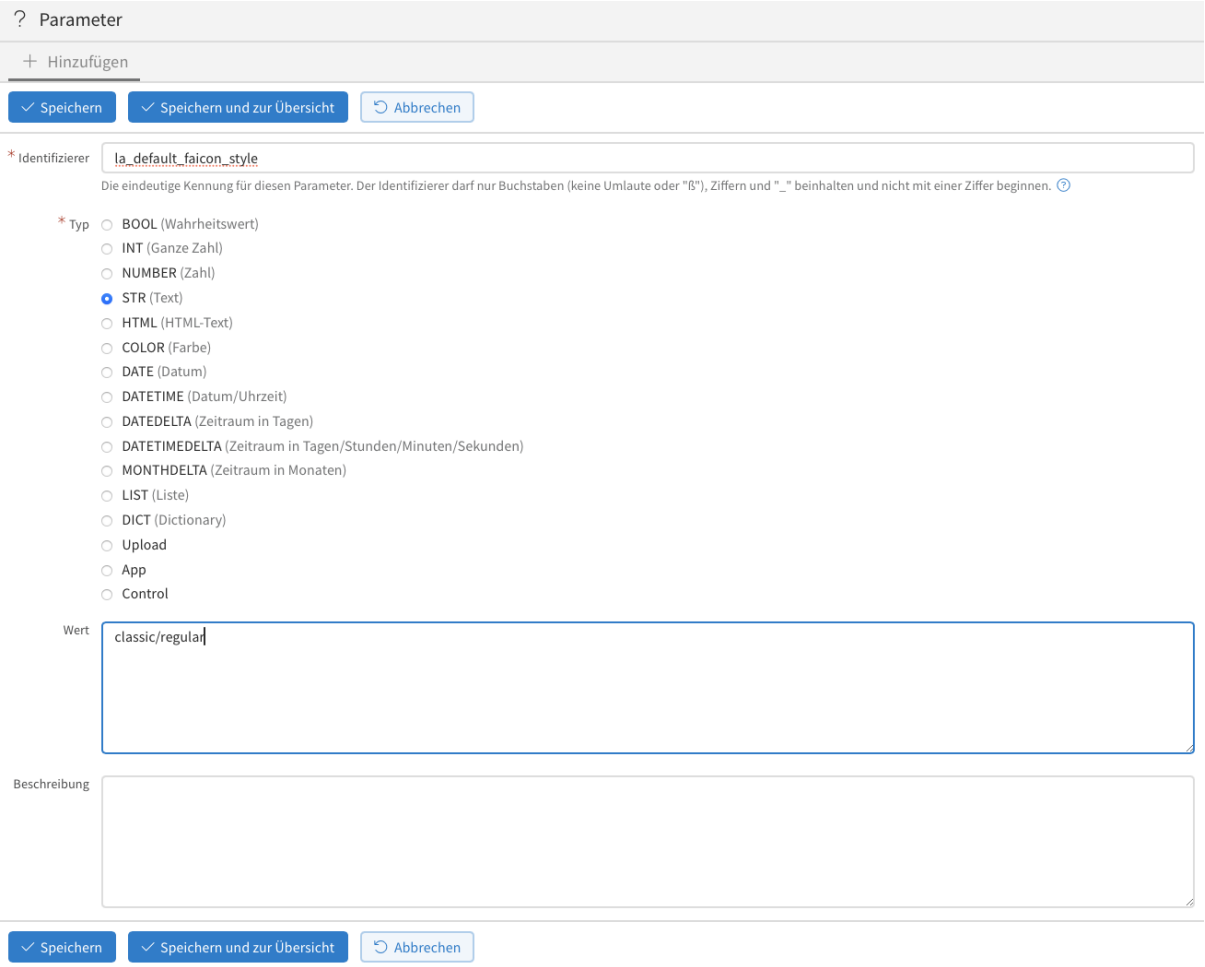

Abb. 53: Parameter konfigurieren

Bemerkung

Normalerweise wird die frei verfügbare Version der Font Awesome-Icons eingebunden. Diese bietet ca. 2000 Icons in den Stilen `classic/solid`, `classic/regular` und `brands`.

Die kommerzielle Version bietet ca. 50000 Icons in den unten aufgelisteten Stilen (ca. 500 in `brands` und jeweils ca. 3000 Icon in den anderen Stilen).

Zur Verwendung ist jedoch eine Font Awesome-Lizenz nötig. Setzen Sie sich dazu mit LivingLogic unter info@livinglogic.de in Verbindung.

Es stehen folgende Stile zur Verfügung:

Tab. 1: icon-Stile

Name	Anzahl Icons (frei, ca.)	Anzahl Icons (kommerziell, ca.)
`classic/solid`	1500	3000
`classic/regular`	150	3000
`classic/light`	0	3000
`classic/thin`	0	3000
`duotone/solid`	0	3000
`duotone/regular`	0	3000
`duotone/light`	0	3000
`duotone/thin`	0	3000
`sharp/solid`	0	3000
`sharp/regular`	0	3000
`sharp/light`	0	3000
`sharp/thin`	0	3000
`sharp-duotone/solid`	0	3000
`sharp-duotone/regular`	0	3000
`sharp-duotone/light`	0	3000
`sharp-duotone/thin`	0	3000
`brands`	500	500

35.19.4 Verwendung eines Nicht-Standard-Stils

Wollen Sie ein Icon mit einem anderen Stil verwendet als den Standard-Stil (z.B. um ein Firmenlogo aus dem `brands`-Stil anzuzeigen), können Sie den Stil mit `/` getrennt an die Icon-Namen hängen:

```
<?render globals.t_la_icon("wikipedia-w/brands")?>
```

erzeugt:

```
<i class="fab fa-wikipedia-w"></i>
```

und:

```
<?render globals.t_la_icon("paw/duotone/light")?>
```

erzeugt:

```
<i class="fadl fa-paw"></i>
```

(für den Icon-Stil `duotone/light` ist jedoch die kommerzielle Version der Font Awesome-Icons nötig.)

35.19.5 Unterschiedliche Icons je nach Font Awesome-Version

Wenn Ihr Wunsch-Icon nur in der kommerziellen Version von Font Awesome zur Verfügung steht und Sie dieses verwenden können, jedoch wollen, daß auch bei Verwendung der freien Version von Font Awesome ein Icon angezeigt wird, können Sie einen zweiten Namen an `la_icon` übergeben:

```
<?render globals.t_la_icon("paw/duotone/light", "paw")?>
```

oder

```
<?render globals.t_la_icon("chevrons-down", "chevron-up")?>
```

In diesem Fall wird dann wird das erste Icon verwendet, wenn die kommerzielle Version in Verwendung ist, und das zweite Icon wenn die freie Version in Verwendung ist.

35.20 Wie kann ich in einem Anzeige-Template ein HTML-„Akkordeon“ verwenden?

Eine „Akkordeon“-Ansicht ist eine Liste von Elementen, die entweder eine abgekürzte Version ihres Inhalts (normalerweise eine Überschrift oder einen Namen) oder den detailierten Inhalt anzeigen. Durch Klicken auf die einzelnen Elemente kann zwischen verkürzter und detailierter Anzeige umgeschaltet werden.

Bei einem „exklusiven“ Akkordeon zeigt immer nur ein Element seinen detailierten Inhalt an.

In HTML gibt es dafür das `details`-Element. Der angekürzte Inhalt muß dabei innerhalb des `details`-Elements in einem `summary`-Element stehen also beispielsweise folgendermaßen:

```
<details>
   <summary>Hinweis zur Verwendung</summary>
   <div>
      <p>Bei der Verwendung müssen Sie folgendes beachten...</p>
   </div>
</details>
```

Um ein exklusives Akkordeon zu erzeugen, müssen die `details`-Elemente die zu diesem Akkordeon gehören, ein `name`-Attribut mit denselben Wert haben.

Zusammen mit den CSS-Klassen aus der Template-Library ergibt sich dann damit folgendes HTML:

```
<div class="la-accordion la-panel">
   <details name="personen" class="la-padding">
      <summary>Bernhard Riemann</summary>
      <table class="la-detail">
         <tr>
            <th>Geboren</th><td>17.06.1826</td></tr>
            <th>Gestorben</th><td>20.06.1866</td></tr>
         </tr>
      </table>
   </details>
   <details name="personen" class="la-padding">
      <summary>Albert Einstein</summary>
      <table class="la-detail">
         <tr>
            <th>Geboren</th><td>15.03.1879</td></tr>
            <th>Gestorben</th><td>15.04.1955</td></tr>
         </tr>
      </table>
   </details>
   <details name="personen" class="la-padding">
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
      <summary>Marie Curie</summary>
      <table class="la-detail">
         <tr>
            <th>Geboren</th><td>08.01.1867</td></tr>
            <th>Gestorben</th><td>04.07.1934</td></tr>
         </tr>
      </table>
   </details>
   <details name="personen" class="la-padding">
      <summary>Carl Friedrich Gauß</summary>
      <table class="la-detail">
         <tr>
            <th>Geboren</th><td>30.04.1777</td></tr>
            <th>Gestorben</th><td>23.02.1855</td></tr>
         </tr>
      </table>
   </details>
</div>
```

Das Ergebnis sieht dann folgendermaßen aus:

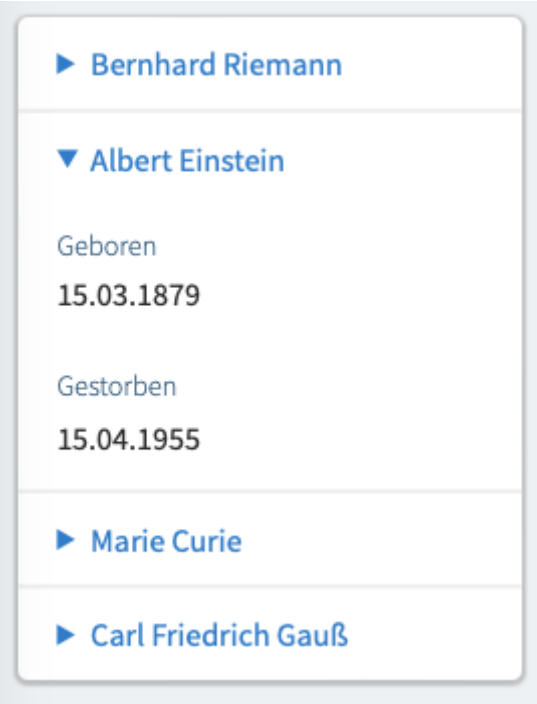

Abb. 54: Akkordeon

35.21 Wie kann ich in meinen Anzeige-Templates den Dunkel-Modus des Browsers unterstützen?

35.21.1 Voraussetzungen

- Sie haben Administrationsrechte an der LivingApp.

35.21.2 Vorgehen

- Klicken Sie in der App, in der Sie für Ihre Templates den Dunkel-Modus des Browsers unterstützen möchten, im Menü *Konfiguration* → *Erweitert* an, um in die erweiterte Konfiguration zu gelangen.
- Klicken Sie dort in der Navigation auf *Template-Library* → *Parameter*.

  Es öffnet sich die Parameter-Library. Klicken Sie auf den für Sie passenden Parameter. In unserem Fall auf `la_css_color_scheme`.

  Es öffnet sich folgendes Fenster:

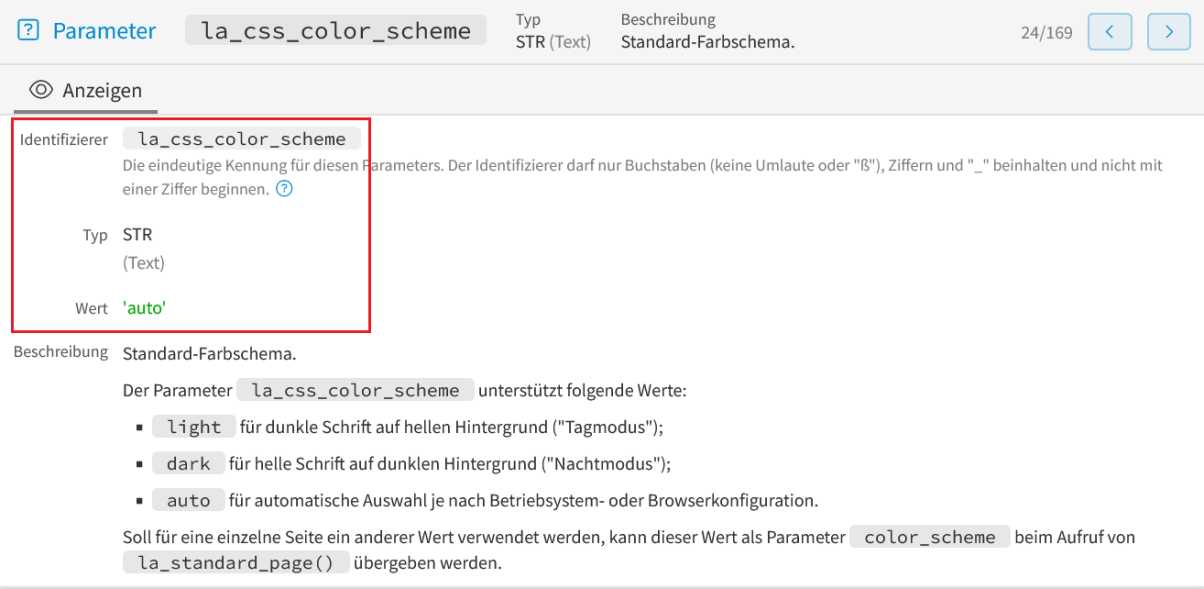

Abb. 55: Parameter aus Library auswählen

  Der Parameter `la_css_color_scheme` unterstützt folgende Werte:

  – `light` für dunkle Schrift auf hellen Hintergrund („Tagmodus“);
  – `dark` für helle Schrift auf dunklen Hintergrund („Nachtmodus“);
  – `auto` für automatische Auswahl je nach Betriebsystem- oder Browserkonfiguration.

- Fügen Sie jetzt den Parameter aus der Library den App-Parametern Ihrer LivingApp hinzu. Klicken Sie dafür unter *Konfiguration* → *Erweitert* auf *Parameter* und dann auf *Neuen Parameter hinzufügen*. Nehmen Sie folgende Konfiguratinen vor:

Identifizierer
: Kopieren Sie den Identifizierer des Parameters aus der Library und fügen ihn hier ein, also `la_css_color_scheme`.

Typ
: Wählen Sie den Typ, der beim Parameter aus der Library angegeben ist, also `STR (Text)`.

Wert
: Der Wert für den Dunkelmodus ist `dark`.

Soll nur für eine einzelne Seite ein anderer Wert verwendet werden, kann dieser Wert als Parameter `color_scheme` beim Aufruf von `la_standard_page()` übergeben werden., also:

```
<?renderblock globals.t_la_standard_page(title="Titel", color_scheme="auto")?>
   ...
<?end renderblock?>
```

? Parameter

+ Hinzufügen

✓ Speichern | ✓ Speichern und zur Übersicht | ↺ Abbrechen

* Identifizierer: la_css_color_scheme

Die eindeutige Kennung für diesen Parameter. Der Identifizierer darf nur Buchstaben (keine Umlaute oder "ß"), Ziffern und "_" beinhalten und nicht mit einer Ziffer beginnen.

* Typ
- ○ BOOL (Wahrheitswert)
- ○ INT (Ganze Zahl)
- ○ NUMBER (Zahl)
- ◉ STR (Text)
- ○ HTML (HTML-Text)
- ○ COLOR (Farbe)
- ○ DATE (Datum)
- ○ DATETIME (Datum/Uhrzeit)
- ○ DATEDELTA (Zeitraum in Tagen)
- ○ DATETIMEDELTA (Zeitraum in Tagen/Stunden/Minuten/Sekunden)
- ○ MONTHDELTA (Zeitraum in Monaten)
- ○ LIST (Liste)
- ○ DICT (Dictionary)
- ○ Upload
- ○ App
- ○ Control

Wert: dark

Beschreibung:

✓ Speichern | ✓ Speichern und zur Übersicht | ↺ Abbrechen

Abb. 56: Konfiguration Parameter

Stichwortverzeichnis

A

accessible, 365, 367
account_url, 329
active, 373
active_view, 338
add_cookie, 332
add_error, 350, 357
add_param, 363
App, 335
app, 299, 322, 334, 344, 348, 361, 364, 365, 368, 377
App.account_url, 329
App.active_view, 338
App.catalog_url, 329
App.categories, 337
App.chats_url, 329
App.children, 365, 367
App.controls, 336
App.createdat, 336
App.createdby, 336
App.custom, 342
App.datamanagement_config_url, 342
App.datamanagement_url, 341
App.datamanageview_url, 342
App.datasource, 338
App.description, 335
App.favorite, 338
App.globals, 335
App.gramgen, 336
App.group, 336
App.home_url, 328, 341
App.id, 335
App.image, 336
App.import_url, 341
App.insert, 339
App.installation, 337
App.lang, 335
App.layout_controls, 338
App.logout_url, 329
App.main, 336
App.menus, 338
App.my_tasks_url, 328
App.name, 335
App.new_embedded_url, 339
App.new_standalone_url, 340
App.panels, 338
App.params, 337
App.permissions_url, 342
App.private_uploads, 336
App.profile_url, 329
App.recordcount, 337
App.records, 337
App.save, 339
App.send_mail, 342
App.seq, 342
App.tasks_url, 341
App.template_url, 340, 354
App.templates, 337
App.typename_acc_plu, 336
App.typename_acc_sin, 335
App.typename_dat_plu, 336
App.typename_dat_sin, 335
App.typename_gen_plu, 336
App.typename_gen_sin, 335
App.typename_nom_plu, 335
App.typename_nom_sin, 335
App.updatedat, 336
App.updatedby, 336
App.views, 338
App.viewtemplates, 337
append_param, 363
AppGroup, 343
AppGroup.apps, 343
AppGroup.globals, 343
AppGroup.id, 343
AppGroup.main_app, 343
AppGroup.name, 343
AppParameter, 361
AppParameter.add_param, 363
AppParameter.app, 361
AppParameter.append_param, 363
AppParameter.createdat, 362
AppParameter.createdby, 362
AppParameter.delete, 363
AppParameter.description, 362
AppParameter.id, 361
AppParameter.identifier, 362
AppParameter.is_deleted, 363

AppParameter.is_dirty, 363
AppParameter.order, 362
AppParameter.parent, 362
AppParameter.save, 363
AppParameter.state, 363
AppParameter.type, 362
AppParameter.updatedat, 363
AppParameter.updatedby, 363
AppParameter.value, 362
apps, 334, 343, 377
archive, 375
archive_url, 374
Attachment, 372
Attachment.active, 373
Attachment.id, 372
Attachment.label, 372
Attachment.record, 372
Attachment.type, 372
attachments, 333, 349
autoalign, 346, 372
autoexpandable, 347, 372

B

background_color1, 366
background_color2, 366
bcc, 333
bodyhtml, 332
bodytext, 332
bool, 313

C

catalog_url, 329
categories, 337
Category, 376
Category.apps, 377
Category.children, 376
Category.id, 376
Category.identifier, 376
Category.name, 376
Category.order, 376
Category.parent, 376
cc, 333
change, 374
chats_url, 329
children, 349, 365, 367, 376
city, 374
clear_all_errors, 350
clear_errors, 350, 357
clear_files, 331
COLOR, 304
column, 367
company, 374
company_website, 374
Control, 343, 368
control, 335, 355, 370
Control.app, 344
Control.autoalign, 346
Control.autoexpandable, 347
Control.custom, 346
Control.description, 344
Control.encrypted, 346
Control.format, 348
Control.fulltype, 344
Control.height, 345
Control.id, 343
Control.identifier, 343
Control.in_active_view, 346
Control.in_list, 344
Control.in_mobile_list, 344
Control.in_text, 345
Control.is_focused, 346
Control.label, 344
Control.labelpos, 345
Control.labelwidth, 346
Control.left, 345
Control.lookup_app, 347
Control.lookup_controls, 347
Control.lookupdata, 347
Control.maximum, 347
Control.minimum, 347
Control.minlength, 346
Control.mode, 345
Control.none_key, 348
Control.none_label, 348
Control.order, 345
Control.placeholder, 346
Control.precision, 347
Control.priority, 344
Control.required, 345
Control.save, 346
Control.search_param_name, 348
Control.search_url, 348
Control.subtype, 344
Control.tabindex, 345
Control.target_param_name, 348
Control.top, 345
Control.type, 344
Control.width, 345
Control.z_index, 345
controls, 336, 369
cookies, 330
cos, 318
createby, 365, 368
createdat, 297, 298, 336, 348, 362, 365, 368, 375
createdby, 297, 298, 336, 348, 362
current_geo, 325
custom, 323, 342, 346, 354, 359

D

datamanagement_config_url, 342
datamanagement_url, 341
datamanageview_url, 342
datasoourcechildren, 355
DataSource, 334
datasource, 334, 338
DataSource.app, 334

DataSource.apps, 334
DataSource.id, 334
DataSource.identifier, 334
DataSourceChildren, 334
DataSourceChildren.control, 335
DataSourceChildren.datasource, 334
DataSourceChildren.id, 334
DataSourceChildren.identifier, 334
datasources, 323
date, 315
DATEDELTA, 304
DATETIMEDELTA, 304
day, 303
days, 317
default, 371
delete, 350, 363
department, 374
description, 297, 298, 335, 344, 362, 366
description_url, 366
details, 349
display_embedded_url, 352
display_standalone_url, 353
display_url, 353
dist, 325
duration, 375

E

edit_embedded_url, 351
edit_standalone_url, 352
Email, 379
email, 296, 373
Email-Request, 332
Email-Response, 332
EmailRequest.bodyhtml, 332
EmailRequest.bodytext, 332
EmailResponse.attachments, 333
EmailResponse.bcc, 333
EmailResponse.cc, 333
EmailResponse.from, 332
EmailResponse.replyto, 333
EmailResponse.subject, 333
EmailResponse.to, 332
enabled, 358
encrypted, 346
end, 369
end_time, 364, 367
endswith, 300
errors, 349, 357
executeaction, 351

F

favorite, 338
fax, 374
Field, 355
Field.add_error, 357
Field.clear_errors, 357
Field.control, 355
Field.custom, 359
Field.enabled, 358
Field.errors, 357
Field.has_custom_lookupdata, 360
Field.has_errors, 357
Field.height, 359
Field.is_dirty, 358
Field.is_empty, 358
Field.label, 355
Field.left, 359
Field.lookupdata, 358, 359
Field.mode, 357
Field.record, 355
Field.required, 359
Field.search_param_name, 360
Field.search_url, 360
Field.set_error, 357
Field.target_param_name, 360
Field.top, 359
Field.value, 355
Field.visible, 358
Field.width, 359
Field.writable, 358
fields, 349
File, 374
File.archive, 375
File.archive_url, 374
File.createdat, 375
File.duration, 375
File.filename, 374
File.geo, 375
File.height, 375
File.mimetype, 375
File.recordedat, 375
File.size, 375
File.url, 374
File.width, 375
filename, 374
find, 301
firstname, 296, 373
flash_error, 327
flash_info, 327
flash_notice, 327
flash_warning, 327
flashes, 327
FlashMessage, 333
float, 314
focus_control, 369
focus_first_control, 369
form, 324
format, 348
from, 332
fulltype, 344

G

gender, 373
GEO, 306
Geo, 375
geo, 324, 375

Geo.info, 375
Geo.lat, 375
Geo.long, 375
Globals, 320
globals, 335, 343
Globals.app, 322
Globals.current_geo, 325
Globals.custom, 323
Globals.datasources, 323
Globals.dist, 325
Globals.flash_error, 327
Globals.flash_info, 327
Globals.flash_notice, 327
Globals.flash_warning, 327
Globals.flashes, 327
Globals.form, 324
Globals.geo, 324
Globals.hostname, 322
Globals.lang, 322
Globals.log_debug, 327
Globals.log_error, 328
Globals.log_info, 327
Globals.log_notice, 328
Globals.log_warning, 328
Globals.mode, 323
Globals.platform, 322
Globals.record, 322
Globals.request, 328
Globals.response, 328
Globals.seq, 325
Globals.templates, 323
Globals.user, 322
Globals.version, 322
gramgen, 336
group, 336

H

has_custom_lookupdata, 360
has_errors, 350, 357
header_background, 366
header_type, 366
headers, 330, 331
height, 345, 359, 361, 367, 369, 371, 375
home_url, 328, 341
hostname, 322
hour, 304
hours, 317
HTMLLayoutControl.value, 361
HTTP-Request, 329
HTTP-Response, 331
HTTPRequest.cookies, 330
HTTPRequest.headers, 330
HTTPRequest.id, 330
HTTPRequest.method, 330
HTTPRequest.params, 330
HTTPRequest.url, 330
HTTPResponse.add_cookie, 332
HTTPResponse.clear_files, 331
HTTPResponse.headers, 331
HTTPResponse.send_file, 331
HTTPResponse.status, 331

I

icon, 364, 366, 378
id, 296–298, 330, 334, 335, 343, 348, 355, 360, 361, 363, 365, 368, 370, 372, 373, 376, 377
identifier, 298, 334, 343, 360, 362, 364, 365, 370, 376, 377
image, 336, 361, 366, 373
ImageLayoutControl.image, 361
import_url, 341
in_active_view, 346
in_list, 344
in_mobile_list, 344
in_text, 345
info, 375
initials, 297, 373
insert, 339
Installation, 376
installation, 297, 337
Installation.id, 376
Installation.name, 376
int, 314
interests, 374
is_deleted, 363
is_dirty, 350, 358, 363
is_empty, 358
is_focused, 346

J

join, 302

K

key, 368, 372, 377
KeyView, 377
KeyView.id, 377
KeyView.identifier, 377
KeyView.key, 377
KeyView.name, 377
KeyView.user, 377
keyviews, 373

L

la-frameless, 38
la-header, 38
label, 344, 355, 360, 364, 366, 368, 370, 372
labelpos, 345, 371
labelwidth, 346, 372
lang, 322, 335, 369
language, 373
lat, 375
layout_controls, 338, 370
LayoutControl, 360
LayoutControl.height, 361
LayoutControl.id, 360
LayoutControl.identifier, 360

LayoutControl.label, 360
LayoutControl.left, 361
LayoutControl.subtype, 361
LayoutControl.top, 361
LayoutControl.type, 361
LayoutControl.view, 360
LayoutControl.visible, 361
LayoutControl.width, 361
LayoutControl.z_index, 361
left, 345, 359, 361, 370
len, 315
Link.app, 364, 365
Link.background_color1, 366
Link.background_color2, 366
Link.column, 367
Link.createby, 365, 368
Link.createdat, 365, 368
Link.description, 366
Link.description_url, 366
Link.end_time, 364, 367
Link.header_background, 366
Link.header_type, 366
Link.height, 367
Link.icon, 364, 366
Link.id, 363, 365
Link.identifier, 364, 365
Link.image, 366
Link.label, 364, 366
Link.on_app_detail_page, 364, 367
Link.on_app_overview_page, 364, 367
Link.on_custom_overview_page, 365, 367
Link.on_form_page, 365, 367
Link.on_iframe_page, 365, 367
Link.order, 364, 367
Link.parent, 364, 366
Link.row, 367
Link.start_time, 364, 367
Link.style, 364, 367
Link.target, 364, 366
Link.target_type, 364, 366
Link.target_url, 364, 367
Link.text_color, 366
Link.title, 364, 366
Link.updateby, 365, 368
Link.updatedat, 365, 368
Link.width, 367
log_debug, 327
log_error, 328
log_info, 327
log_notice, 328
log_warning, 328
login_required, 369
logout_url, 329
long, 375
lookup_app, 347
lookup_controls, 347
lookup_none_key, 371
lookup_none_label, 371
lookupdata, 347, 358, 359, 371
LookupItem, 368
LookupItem.Control, 368
LookupItem.key, 368
LookupItem.label, 368
LookupItem.visible, 368
lower, 299
lstrip, 301

M

main, 336
main_app, 343
maximum, 347
maxlength, 346
md5, 318
MenuItem, 363
menus, 338
method, 330
mimetype, 375, 378
minimum, 347
minlength, 346, 371
minute, 304
minutes, 317
mode, 323, 345, 357, 371
month, 303
MONTHDELTA, 304
monthdelta, 316
months, 317
my_tasks_url, 328

N

name, 297, 335, 343, 368, 376–378
new_embedded_url, 339
new_standalone_url, 340
none_key, 348
none_label, 348
now, 313

O

on_app_detail_page, 364, 367
on_app_overview_page, 364, 367
on_custom_overview_page, 365, 367
on_form_page, 365, 367
on_iframe_page, 365, 367
order, 345, 362, 364, 367, 369, 376

P

Panel, 365
panels, 338
params, 330, 337
parent, 362, 364, 366, 376
permission_level, 378
permissions_url, 342
personal_website, 374
phone, 374
placeholder, 346, 371
platform, 322
position, 374

precision, 347
priority, 344
private_uploads, 336
profile_url, 329

R

random, 318
randrange, 318
Record, 348
record, 322, 355, 372
Record.add_error, 350
Record.app, 348
Record.attachments, 349
Record.children, 349
Record.clear_all_errors, 350
Record.clear_errors, 350
Record.createdat, 348
Record.createdby, 348
Record.custom, 354
Record.delete, 350
Record.details, 349
Record.display_embedded_url, 352
Record.display_standalone_url, 353
Record.display_url, 353
Record.edit_embedded_url, 351
Record.edit_standalone_url, 352
Record.errors, 349
Record.executeaction, 351
Record.fields, 349
Record.has_errors, 350
Record.id, 348
Record.is_dirty, 350
Record.save, 350
Record.update, 350
Record.updatedat, 349
Record.updatedby, 349
Record.values, 349
RecordChildren, 355
RecordChildren.datasoourcechildren, 355
RecordChildren.id, 355
RecordChildren.record, 355
RecordChildren.records, 355
recordcount, 337
recordedat, 375
records, 337, 355
replyto, 333
request, 328
required, 345, 359, 371
response, 328
result_page, 369
rfind, 302
row, 367
rstrip, 301

S

save, 339, 346, 350, 363
search_param_name, 348, 360
search_url, 348, 360
second, 304
seconds, 318
send_file, 331
send_mail, 342
seq, 325, 342
set_error, 357
sin, 318
size, 375
split, 302
sqrt, 318
start, 369
start_time, 364, 367
startswith, 300
state, 363
status, 331
str, 314
streetname, 373
streetnumber, 373
strip, 300
style, 364, 367
subject, 333
subtype, 344, 361, 370
summary, 374
surname, 296, 373

T

tabIndex, 371
tabindex, 345
tan, 318
target, 364, 366
target_param_name, 348, 360
target_type, 364, 366
target_url, 364, 367
tasks_url, 341
template_url, 340, 354
templates, 323, 337
text_color, 366
timedelta, 316
title, 364, 366
to, 332
today, 313
top, 345, 359, 361, 370
type, 344, 361, 362, 370, 372, 378
typename_acc_plu, 336
typename_acc_sin, 335
typename_dat_plu, 336
typename_dat_sin, 335
typename_gen_plu, 336
typename_gen_sin, 335
typename_nom_plu, 335
typename_nom_sin, 335

U

update, 350
updateby, 365, 368
updatedat, 297, 298, 336, 349, 363, 365, 368
updatedby, 297, 298, 336, 349, 363
upper, 299

url, 298, 330, 374
use_geo, 369
User, 373
user, 322, 377
User.change, 374
User.city, 374
User.company, 374
User.company_website, 374
User.department, 374
User.email, 373
User.fax, 374
User.firstname, 373
User.gender, 373
User.id, 373
User.image, 373
User.initials, 373
User.interests, 374
User.keyviews, 373
User.language, 373
User.personal_website, 374
User.phone, 374
User.position, 374
User.streetname, 373
User.streetnumber, 373
User.summary, 374
User.surname, 373
User.zip, 373

V

value, 298, 355, 361, 362
values, 349
version, 322
View, 368
view, 360, 370
View.app, 368
View.controls, 369
View.end, 369
View.focus_control, 369
View.focus_first_control, 369
View.height, 369
View.id, 368
View.lang, 369
View.layout_controls, 370
View.login_required, 369
View.name, 368
View.order, 369
View.result_page, 369
View.start, 369
View.use_geo, 369
View.width, 369
ViewControl, 370
ViewControl.autoalign, 372
ViewControl.autoexpandable, 372
ViewControl.control, 370
ViewControl.default, 371
ViewControl.height, 371
ViewControl.labelpos, 371
ViewControl.labelwidth, 372
ViewControl.left, 370
ViewControl.lookup_none_key, 371
ViewControl.lookup_none_label, 371
ViewControl.lookupdata, 371
ViewControl.maxlength, 346
ViewControl.minlength, 371
ViewControl.mode, 371
ViewControl.placeholder, 371
ViewControl.required, 371
ViewControl.subtype, 370
ViewControl.tabIndex, 371
ViewControl.top, 370
ViewControl.type, 370
ViewControl.view, 370
ViewControl.width, 370
ViewControl.z_index, 371
ViewControlid, 370
ViewControlidentifier, 370
ViewControllabel, 370
ViewLookupItem, 372
ViewLookupItem.key, 372
ViewLookupItem.label, 372
ViewLookupItem.visible, 372
views, 338
ViewTemplateInfo, 377
ViewTemplateInfo.app, 377
ViewTemplateInfo.icon, 378
ViewTemplateInfo.id, 377
ViewTemplateInfo.identifier, 377
ViewTemplateInfo.mimetype, 378
ViewTemplateInfo.name, 378
ViewTemplateInfo.permission_level, 378
ViewTemplateInfo.type, 378
viewtemplates, 337
visible, 358, 361, 368, 372

W

weekday, 303
width, 345, 359, 361, 367, 369, 370, 375
writable, 358

Y

year, 303
yearday, 303
years, 317

Z

z_index, 345, 361, 371
zip, 373