

# Software-Engineering Programmmentwurf

---

## Thema: Hochzeitsplaner

des Studiengangs Angewandte Informatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

**Max Lenk & Kai Müller**

09.08.2016

Kurs	TINF14B1
Semester	3./4. Semester
Dozent	Dr.-Ing. R. Lutz

# Inhaltsverzeichnis

<b>Aufgabenstellung</b>	4
1 Einleitung	4
2 Lastenheft	4
2.1 Zielsetzung	4
2.2 Anwendungsbereiche	4
2.3 Zielgruppen, Benutzerrollen und Verantwortlichkeiten	4
2.4 Zusammenspiel mit anderen Systemen	4
2.5 Produktfunktionen	5
2.6 Produktdaten	6
2.7 Produktleistungen	6
2.8 Qualitätsanforderung	6
3 Vereinfachungen für den Programmentwurf	7
<b>Analyse</b>	8
4 Einleitung	8
5 Lastenheft	8
5.1 Zielsetzung	8
5.2 Anwendungsbereiche	10
5.3 Zielgruppen, Benutzerrollen und Verantwortlichkeiten	10
5.4 Zusammenspiel mit anderen Systemen	12
5.5 Produktfunktionen	12
5.6 Produktdaten	23
5.7 Produktleistungen	23
5.8 Qualitätsanforderung	24
5.9 Entitäten und Attribute	24
6 Use-Case-Diagramme	27
6.1 Einleitung	27
6.2 Diagramm Aktion verwalten	27
6.3 Diagramm Hilfsmittel verwalten	27
6.4 Diagramm Caterer verwalten	28
6.5 Diagramm Lebensmittel verwalten	28
6.6 Diagramm Hochzeitsveranstaltung verwalten	29
6.7 Akteure	29
6.8 Use Cases	30

7	Analyse-Klassendiagramm.....	33
7.1	Einleitung.....	33
7.2	Hinweis .....	33
7.3	Pattern.....	33
7.4	Diagramm .....	33
7.5	Entities.....	34
8	Sequenzdiagramm.....	36
8.1	Einleitung:.....	36
8.2	Vorbedingungen .....	36
8.3	Szenario .....	36
8.4	Pseudocode .....	36
8.5	Diagramm Hochzeitsveranstaltung anlegen .....	37
8.6	Diagramm Unterhaltsbeitrag anlegen.....	38
9	Aktivitätsdiagramm .....	39
9.1	Einleitung.....	39
9.2	Diagramm Trauung anlegen .....	39
9.3	Diagramm Beleg für Trabat erstellen .....	39
9.4	Diagramm Ort Standesamt anlegen.....	39
9.5	Diagramm Hochzeitsveranstaltung anlegen .....	40
9.6	Grundlegendes .....	40
9.7	Hauptdiagramm.....	40
9.8	Teildiagramm: Aktion anlegen .....	41
9.9	Teildiagramm Hilfsmittel erstellen .....	41
9.10	Subdiagramm Hochzeitsveranstaltung anlegen .....	42
9.11	Subdiagramm Ort „Standesamt“ anlegen.....	42
9.12	Subdiagramm Beleg „Beleg für Trabant“ erstellen .....	42
9.13	Pseudocode .....	43
	<b>Entwurf.....</b>	<b>44</b>
10	Datenbankentwurf .....	44
10.1	Voraussetzungen .....	44
10.2	Kurzeinführung SQL.....	44
10.3	SQL .....	45
10.4	Erstellen des Schemas .....	51
10.5	Setzen des aktuellen Schemas .....	51
10.6	Erzeugen der Tabellen mit Fremdschlüsselbeziehungen .....	51
10.7	Bekannte Entitäten.....	51
10.8	Weitere Entitäten .....	52

10.9 Relationstabellen.....	52
10.10 Weitere Besonderheiten.....	52
10.11 Programmtabellen .....	52
11 Entwurfsklassendiagramm .....	53
11.1 Einleitung.....	53
11.2 Diagramm .....	53
11.3 Package util.....	54
11.4 Package model .....	55
11.5 Package event.....	56
11.6 Package exception.....	56
11.7 Package datenbank .....	57
11.8 Package gui.....	57
12 GUI-Entwurf.....	60
12.1 Ablaufplan .....	60
12.2 Aktionen verwalten .....	62
12.3 Neue Aktion anlegen .....	64

# Aufgabenstellung

## 1 Einleitung

Wir die EMSIG GmbH (Event Management Schulze Irrwisch Gimpel GmbH) sind ein führendes mittelständisches Unternehmen für die Planung und Durchführung mittelgroßer Veranstaltungen (ca. 100 -1000 Teilnehmern). Hierfür setzen wir seit Jahren ein bewährtes Softwarewerkzeug ein.

Speziell für die Planung von Hochzeiten benötigen wir ein neues Werkzeug, welches zum einen für unsere eigene Firma eingesetzt und zum anderen auch für Privatpersonen als günstige Planungssoftware angeboten werden soll.

## 2 Lastenheft

### 2.1 Zielsetzung

Ziel des Entwicklungsauftrags soll eine Software für die Verwaltung von Hochzeiten sein. Dabei soll auf eine zentrale Datenbasis zugegriffen werden können (Server), damit sämtliche Daten von mehreren PCs und Laptops aus verwaltet werden können. Daneben sollen mehrere Personen gemeinsam an der Hochzeitsplanung teilnehmen können.

Ein Import und Export ausgewählter Daten muss zur besseren Wiederverwendbarkeit, für Backups und zum Datenaustausch möglich sein.

Eine intuitive, leicht bedienbare Benutzeroberfläche setzen wir als selbstverständlich voraus.

Es sollen keine besonderen Computerkenntnisse zur Bedienung der Software erforderlich sein.

### 2.2 Anwendungsbereiche

Die Software soll ausschließlich für die Verwaltung von Hochzeiten eingesetzt werden. Sie soll bei uns in der Firma im Tagesgeschäft eingesetzt werden sowie von Privatpersonen erwerbbar sein.

### 2.3 Zielgruppen, Benutzerrollen und Verantwortlichkeiten

Als Zielgruppe kommen zwei Rollen infrage: die eigentliche planungsverantwortliche Person, welche auf sämtliche Daten lesend und schreibend Zugriff hat (Hochzeitsmanager).

Ausnahme: da oftmals das Brautpaar selbst planen und managen will, soll es möglich sein, die persönlichen Unterhaltungsbeiträge für die Hauptplaner zu verstecken. Hierfür soll es eine zweite Rolle geben, die lesenden Zugriff auf die grundlegenden Hochzeitsdaten hat (Zeiten, Datumsangaben, Orte, ... ), ansonsten aber ausschließlich die Unterhaltungsbeiträge verwalten kann (Unterhaltungsmanager).

### 2.4 Zusammenspiel mit anderen Systemen

Das zu entwickelnde Softwaresystem soll auch ohne Netzverbindung lauffähig sein. Hierzu sollen sämtliche Daten einer Hochzeit lokal gespeichert und auf Wunsch des Benutzers mit den Serverdaten synchronisiert werden können.

## 2.5 Produktfunktionen

- /LF10/ Eine Zugangsberechtigung soll mittels eines einfachen Loginvorgangs verifiziert werden.
- Der Zugriff auf einzelne Daten soll je nach Berechtigung unterschiedlich erfolgen.  
→ siehe Abschnitt 2.3: „Zielgruppen, Benutzerrollen und Verantwortlichkeiten“
- /LF20/ Der jeweilige Benutzer muss die Möglichkeit haben, über eine grafische Benutzeroberfläche (GUI) alle für ihn relevanten Daten einfach und übersichtlich verwalten zu können.
- /LF30/ Eine Hochzeitsveranstaltung fasst viele Einzelaktionen zusammen: angefangen von der standesamtlichen und kirchlichen Trauung, Buchung bzw. Reservierung von Veranstaltungsorten, Catering (Essen und Getränke getrennt verwaltbar) über Erstellung und/oder Druck von Einladungen, Tischkarten, diverse Besorgungen, Organisation und Buchung von Übernachtungen und dem Hochzeitsfahrzeug bis zur Organisation und Durchführung der Dekoration (in Standesamt, Kirche und Festsaal) uvm.
- /LF40/ Jede Aktion beginnt und endet zu einem bestimmten Zeitpunkt, es müssen verantwortliche Personen und Teilnehmer der Aktion benannt werden können. Die verantwortlichen Personen und Teilnehmer sollen automatisch per E-Mail benachrichtigt werden können (z.B. durch starten eines vorhandenen Mail-Tools über die grafische Benutzeroberfläche).
- Jede Aktion kann an mehreren Orten stattfinden, mit Hilfsmitteln aus einer Liste durchgeführt werden. Anfallende Kosten (Rechnungen, Belege) sollen jeweils mit angegeben werden können.
- Für den Benutzer soll es leicht möglich sein, die aktuellen Zustände aller Aktionen zu erkennen (geplant, in Arbeit, abgeschlossen, usw.) und zu ändern. Der Benutzer soll geeignet dabei unterstützt werden, bestimmte vorgegebene Aktionsarten anlegen und durchführen zu können. Dabei soll der Benutzer einfach erkennen können, welche Aktionen bereits angelegt sind und welche noch nicht.
- Es soll darüber hinaus möglich sein, einer Aktion verschiedene Medien (Dokumente, Bilder, Videos, usw.) zuzuordnen.
- /LF50/ Jede verantwortliche Person und jeder Teilnehmer kann bei den üblichen Kontaktdaten mehrere E-Mail-Adressen und mehrere Telefonnummern besitzen.
- /LF60/ Das Catering kann entweder von einem kommerziellen Catering-Service als auch von ausgewählten Personen durchgeführt werden.
- /LF70/ Die oben erwähnte Liste der Hilfsmittel soll auf einfache Weise erweiterbar und zuweisbar sein. Sie sollen für sämtliche Hochzeitsveranstaltungen im System verfügbar sein.
- /LF80/ Zur Kostenkontrolle soll es möglich sein, sämtliche bisher angefallenen Kosten auf einfache Weise addieren zu können. Zur Kostenabschätzung sollen auch geschätzte Kosten angegeben und addiert werden können.
- /LF90/ Die Auswahl der Daten soll möglichst über (eventuell durchsuchbare) Auswahllisten erfolgen. Dies gilt vor allem für Zuordnungen von ausgewählten Personen zu den Aktionen usw.

/LF100/ Vor dem Hinzufügen von neuen Daten soll eine Überprüfung stattfinden, ob diese eventuell schon vorhanden sind. Das gilt in besonderem Maße für Personen und Aktionen.

## 2.6 Produktdaten

/LD10/ Die Daten sollen zentral verwaltet und in einer Datenbank abgespeichert werden.

## 2.7 Produktleistungen

/LL10/ Das Laden gewünschter Daten soll für eine sinnvolle Benutzung im Sekundenbereich erfolgen.

/LL20/ Die Anzahl der zu verwaltenden Elemente wird auf ca. 100000 geschätzt.

/LL30/ Die Daten müssen bei unserer eigenen Verwendung aus rechtlichen Gründen 10 Jahre online verfügbar sein.

/LL40/ Um bei Anschaffungen und Neuerungen flexibel zu bleiben, ist auf Plattformunabhängigkeit besonders zu achten.

## 2.8 Qualitätsanforderung

Produktqualität	sehr gut	gut	normal	nicht relevant
Funktionalität	X			
Zuverlässigkeit		X		
Effizienz			X	
Benutzbarkeit	X			
Änderbarkeit			X	
Übertragbarkeit			X	
Gestaltung	X			

### 3 Vereinfachungen für den Programmentwurf

1. Es muss nicht dafür gesorgt werden, dass auf dieselben Daten der Datenbank nicht gleichzeitig zugegriffen werden kann, d.h. es ist kein Locking-Mechanismus erforderlich.
2. Eine Protokollierfunktion ist für die Anwendung nicht erforderlich (in der Realität natürlich schon!).
3. Ein Loginvorgang und eine Benutzerverwaltung müssen in den Klassendiagrammen nicht modelliert und später auch nicht implementiert werden
4. Die Synchronisation der Daten muss bei der Implementierung nicht berücksichtigt werden.

P.S.: Kopieren Sie den Aufgabentext (d.h. ohne Frontseite) vollständig als erstes Kapitel „Aufgabenstellung“ an den Anfang Ihrer PE-Dokumentation und verwenden Sie den Aufgabentext zusätzlich als Rahmen für Ihre Lastenheftanalyse („ausfüllen“ mit Fragen und Antworten)!



# Analyse

## 4 Einleitung

Wir die EMSIG GmbH (Event Management Schulze Irrwisch Gimpel GmbH) sind ein führendes mittelständisches Unternehmen für die Planung und Durchführung mittelgroßer Veranstaltungen (ca. 100 -1000 Teilnehmern). Hierfür setzen wir seit Jahren ein bewährtes Softwarewerkzeug ein.

**Kann es bei diesen Veranstaltungen zu kleineren oder größeren Teilnehmerzahlen kommen?**

Ja, dies könnte theoretisch durchaus möglich sein. Allerdings ist dies praktisch noch nie vorgekommen.

Speziell für die Planung von Hochzeiten benötigen wir ein neues Werkzeug, welches zum einen für unsere eigene Firma eingesetzt und zum anderen auch für Privatpersonen als günstige Planungssoftware angeboten werden soll.

**Welches Preissegment ist unter günstig zu verstehen?**

Der Preis soll sich im Bereich zwischen 50 und 60 € bewegen.

**Bezieht sich die oben genannte Teilnehmerzahl auch auf die gewünschte Software (sowohl die kommerzielle als auch die private)?**

Ja, da sowohl private als auch kommerzielle Nutzer sollen Veranstaltungen in dieser Größe planen können.

## 5 Lastenheft

### 5.1 Zielsetzung

Ziel des Entwicklungsauftrags soll eine Software für die Verwaltung von Hochzeiten sein. Dabei soll auf eine zentrale Datenbasis zugegriffen werden können (Server), damit sämtliche Daten von mehreren PCs und Laptops aus verwaltet werden können. Daneben sollen mehrere Personen gemeinsam an der Hochzeitsplanung teilnehmen können.

**Was für ein Server soll Verwendung finden?**

Die Auswahl der Serversoftware ist für uns nicht relevant, es sollte nur ein Remote Login verfügbar sein.

**Sind an diesen Server spezielle Anforderungen gestellt?**

Der Server sollte rund um die Uhr erreichbar sein und entsprechende Sicherheit in Bezug auf Ausfälle und Zugriffe bieten.

**Ist der Server ein eigener Server der Firma oder ein neuer?**

Da wir noch keine Serverinfrastruktur besitzen, benötigen wir einen neuen Server.

**Wer sollte die Server- und Netzwerkeinrichtung übernehmen?**

Wir werden externe Firma damit beauftragen, falls Rückfragen bestehen, können wir den Kontakt zu Ihnen herstellen.

**Wie soll die Softwareinstallation geschehen?**

Sie werden Zugriff zum System dafür bekommen.

**Ist der Server nur für diese Software ausgelegt oder läuft auf diesem noch andere Dienste?**

Er wird voraussichtlich nur für diese Software ausgelegt werden.

Ein Import und Export ausgewählter Daten muss zur besseren Wiederverwendbarkeit, für Backups und zum Datenaustausch möglich sein.

**Wie und wann soll das Backup geschehen?**

Das Backup geschieht alle 4 Stunden automatisch und kann zusätzlich manuell angestoßen werden. Ein Backup steht für 2 Tage zur Verfügung bis es gelöscht wird. Diese Funktion soll auf dem Server geschehen, sodass kein Client dafür zuständig sein muss.

**Auf was wird das Backup gespeichert?**

Es wird auf einem zusätzlichen, speziell auf für Backups ausgelegten Server gespeichert.

**Wie sieht die Backupspeicherung bei Privatpersonen aus?**

Bei Privatpersonen soll das Backup genauso geschehen. Allerdings werden diese die Backups eher auf externen Festplatten speichern.

**Sind die Dateiformate für den Imprt/Export und dem Backup verschieden?**

Ja, für das Backup soll ein Backup der Datenbank im Datenbankhersteller üblichen Format geschehen. Die Import/Export Format soll unabhängig von der Datenbank sein.

**Wie soll ein Backup z.B. nach einem Systemcrash wiedereingespielt werden?**

Dafür ist die Wiederherstellungsfunktion der Datenbank zuständig und der Systemadministrator.

**Wie soll das Import/Export-Format aussehen?**

Sowohl der Import als auch der Export sollte im XML(Extended Markup Language) Format möglich sein. Eine Serialisierung der Entitäten in XML sollte für die Speicherung ausreichen. Für den Ablaufplan ist es möglich diesen als ical-Datei zu exportieren.

**Wie soll man auswählen können welche Daten exportiert werden sollen?**

Grundsätzlich ist jeder Entitätstyp exportierbar, was über einen Menüpunkt „exportieren“ geschieht, den man auswählen kann wenn man die einzelne Entität verwaltet. Zusätzlich gibt es die Möglichkeit, mehrer Entitäten als Entitätsset zu exportieren, was in den Bereichen geschieht, wo entsprechende Entitätstypen verwaltet werden. Für den Ablaufplan lässt sich auswählen welche Aktionen in die iCal-Datei übernommen werden.

**Woe soll die Import-Funktion aussehen?**

Die Importfunktion ist kontextunabhängig überall verfügbar. Dabei soll einfach eine Datei ausgewählt werden. Danach sollen dem Nutzer alle Entitäten übersichtlich angezeigt werden, die nun importiert werden können. Diese kann der Nutzer (de-)selektieren.

Eine intuitive, leicht bedienbare Benutzeroberfläche setzen wir als selbstverständlich voraus.

Es sollen keine besonderen Computerkenntnisse zur Bedienung der Software erforderlich sein.

**Was versteht man unter einer intuitiven, leicht bedienbaren Benutzeroberfläche?**

Eine Oberfläche, die vom einen Nutzer ohne Vorkenntnisse benutzt werden kann.

## 5.2 Anwendungsbereiche

Die Software soll ausschließlich für die Verwaltung von Hochzeiten eingesetzt werden. Sie soll bei uns in der Firma im Tagesgeschäft eingesetzt werden sowie von Privatpersonen erwerbbar sein.

**Soll es Unterschiede zwischen der Firmen- und der Privatpersonen-Software geben?**

Privatpersonen erhalten eine „Slim-Version“. Hierbei muss z.B. nicht auf die Verwaltung von vielen Hochzeiten geachtet werden, da Endnutzer immer nur eine Hochzeit planen können. Ebenso besitzt sie eine schlankere Benutzerverwaltung was den Administrationsaufwand um einiges verringert.

**Was ist unter einer schmaleren Nutzerverwaltung zu verstehen?**

Die Administratorrolle ist überflüssig. Auch eine Registrierung ist überflüssig. Auch verschiedene Benutzer (die Hochzeiten erstellen) wie in einer großen Firma von Nöten sind, ist überflüssig.

**Muss auf Internationalisierung geachtet werden?**

Auf Internationalisierung wie Right-To-Left muss nicht geachtet werden, allerdings auf die Möglichkeit der Sprachänderung. Der Kunde soll die Möglichkeit haben die Sprache innerhalb der laufenden Applikation zu ändern.

**Welche Sprachen sollen unterstützt werden?**

Vorerst wollen wir Deutsch, Englisch, Spanisch, Italienisch und Französisch unterstützen. Wir behalten uns vor in einem Update weitere Sprachen hinzuzufügen.

**Muss bei der Software von Privatpersonen auch ein Server mit Datenbank verfügbar sein?**

Ja, allerdings können Privatpersonen unsere Server, gegen eine zusätzliche monatliche Gebühr, nutzen um dort ihre Daten zu zentralisieren.

**Kann eine Privatperson, sofern ein Server Pflicht ist, einen bei der Firma mieten oder kann sie auch einen eigenen Server nutzen?**

Ein Server wird immer benötigt, allerdings muss dieser nicht unbedingt installiert sein. Das heißt, wenn kein externer Server verfügbar ist, dann wird die Software automatisch einen lokalen Server erstellen.

Privatpersonen können einen Server bei uns mieten, sind allerdings nicht dazu verpflichtet. Das heißt, dass sie, insofern sie über entsprechende Kenntnisse verfügen, einen eigenen Server verwenden können.

## 5.3 Zielgruppen, Benutzerrollen und Verantwortlichkeiten

Als Zielgruppe kommen zwei Rollen infrage: die eigentliche planungsverantwortliche Person, welche auf sämtliche Daten lesend und schreibend Zugriff hat (Hochzeitsmanager).

Ausnahme: da oftmals das Brautpaar selbst planen und managen will, soll es möglich sein, die persönlichen Unterhaltungsbeiträge für die Hauptplaner zu verstecken. Hierfür soll es eine zweite Rolle geben, die lesenden Zugriff auf die grundlegenden Hochzeitsdaten hat (Zeiten, Datumsangaben, Orte, ...), ansonsten aber ausschließlich die Unterhaltungsbeiträge verwalten kann (Unterhaltungsmanager).

**Ist der Hochzeitsmanager immer das Brautpaar?**

Da eine Hochzeit ein sehr spezielles Ereignis ist und viele unserer Kunden schon gewisse Vorstellungen haben, unterstützen wir unsere Kunden mit Erfahrung und Kenntnissen. Das Brautpaar kann somit auch der Hochzeitsmanager sein, was wir natürlich so begrüßen und unterstützen. Dennoch gibt es Fälle in denen das Hochzeitspaar nicht als Hochzeitsmanager in Erscheinung tritt. Sofern dies der Fall ist, wird einer unserer Planer als Hochzeitsmanager eingetragen werden.

**Wie soll vermerkt werden, dass der Hochzeitsmanager das Brautpaar ist?**

Bei der Vergabe des Hochzeitsmanagers an die Hochzeit soll dies festgelegt werden. Hierbei tritt das Brautpaar als ein Nutzer auf.

**Welche Rolle bekommt das Brautpaar, falls es kein Hochzeitsmanager ist?**

Selbst, falls das Brautpaar nicht der „Hochzeitsmanger“ ist, erhält es die Rolle Hochzeitsmanager in der Anwendung, um entsprechende Änderungen vornehmen zu können, da es dennoch einen erheblichen Einfluss auf die Planung hat.

**Wie soll die Benutzerverwaltung aussehen?**

Jeder Benutzer muss sich zunächst beim ersten Start registrieren, insofern er noch keinen Account hat. Nachdem der Nutzer registriert ist, prüft der Systemadministrator die Anfrage und genehmigt diese und weist ihm gegenfalls spezielle Rechte zu.

**Wie geschieht die Registrierung?**

Der Nutzer wird in einem Registrierungsbildschirm dazu aufgefordert seine E-Mail, Telefonnummer, Nutzernamen, sowie sein Passwort anzugeben. Nachdem er das Formular ausgefüllt hat, erhält der Administrator eine Anfrage mit denen Daten (exklusive Passwort).

**Was geschieht nach der erfolgreichen Registration?**

Der zu registrierende Nutzer erhält eine Bestätigung auf seine angegebene E-Mail Adresse.

**Wer übernimmt die Administration des Servers?**

Bei Firmen übernimmt dies der Netzwerkadministrator bzw. eine dafür geeignete Person. Bei Privatpersonen der Hochzeitsmanager.

**Wer übernimmt die Rechteverwaltung?**

Der Administrator übernimmt diese Aufgabe.

**Wer kann neue Hochzeitsveranstaltungen erstellen?**

Per se natürlich der Administrator. Allerdings ist geplant, dass ein Hochzeitsmanager eine Hochzeit erstellen kann, in die er dann automatisch als Hochzeitsmanager eingetragen wird.

**Sollen Benutzer auf verschiedene Hochzeitsplanungen eingeschränkt werden können?**

Nein, für die private Software ist dies nicht nötig. Innerhalb der Firma sollen für die Planer nur die Hochzeiten dargestellt werden, in denen sie Hochzeitsmanager sind. Das Gleiche soll für die Unterhaltungsmanager gelten.

**Wie erlangen Unterhaltungsmanager Recht an einer Hochzeit?**

Die Unterhaltungsmanagerrechte erhalten sie vom Administrator, den Zugriff auf die Hochzeit vom Hochzeitsmanager.

**Soll es möglich sein, auch den Ablauf der Hochzeit zu planen und wenn ja, wie soll dies aussehen?**

Eine Hochzeit besteht aus verschiedenen Aktionen, die einzeln verwaltet werden können.

Die Aktionen sind quasi die einzelnen Punkte in einem Ablaufplan. Es soll die Möglichkeit geben, sich die Aktionen als einen Ablaufplan anzeigenzulassen. Man kann bei diesem Anzeigen auch aus Aktionen selektieren, die nicht dargestellt werden sollen. Der Ablaufplan existiert als souveräner GUI Bestandteil, der alle Aktion darstellen kann und Meilensteine hervorhebt.

**Soll es die Möglichkeit geben den oben genannten Ablaufplan zu exportieren, z.B. auf ein Mobilfunkgerät?**

Ja, eine Export-Möglichkeit ist wünschenswert. Am Besten wäre dies in Form eines abrufbaren Kalenders in ical-Format. Sofern der Server Internetzugang hat kann man den Ablaufplan synchronisieren, ohne die Datei an sich zu erneuern. Somit hat der Nutzer die Termine und deren Metadaten auch mobil verfügbar.

**Auf welche Objekte hat der Unterhaltungsmanager lesenden Zugriff?**

Auf alle Informationen einer Hochzeit und die Aktionen, die er angelegt hat.

**Auf welche Objekte hat der Unterhaltungsmanager schreibenden Zugriff?**

Er hat auf alle Aktionen schreibenden Zugriff, die er angelegt hat bzw als verantwortliche Person eingetragen ist.

## 5.4 Zusammenspiel mit anderen Systemen

Das zu entwickelnde Softwaresystem soll auch ohne Netzverbindung lauffähig sein. Hierzu sollen sämtliche Daten einer Hochzeit lokal gespeichert und auf Wunsch des Benutzers mit den Serverdaten synchronisiert werden können.

**Wie soll auf die Synchronisierung der Daten geachtet werden?**

Der Datenstand sollte, sofern eine Internetverbindung besteht, dauerhaft aktuell gehalten werden. Dadurch soll garantiert werden, dass Nutzer immer die neusten Daten bearbeiten. Wenn der Nutzer kein Internet hat, dann soll sobald er wieder Zugriff hat ein Abgleich durchgeführt werden und entsprechende Konflikte behoben werden.

**Wie erhält der Nutzer offline Zugang zu den Daten?**

Die Daten sollen lokal gespeichert werden. Dies ist allerdings nur bei den Privatpersonen der Fall, da bei diesen dies Speicherplatztechnisch möglich ist. Bei der Firmensoftware sollen nur die Daten lokal gespeichert werden, die für den Nutzer relevant sind.

**Welche Daten sind für einen Nutzer der Firmensoftware relevant?**

Alle Hochzeitsveranstaltungen mit denen er zu tun hat und alle damit verbundenen Daten sind relevant für ihn. Somit kann es also sein, dass er z.B. keine vollständige Hilfsmittelliste besitzt.

**Im welchem Format sollen die Dateien lokal gespeichert werden?**

Die Daten sollen im CSV Format entsprechend dem Datenbankschema gespeichert werden.

**Wie sollten eventuelle Konflikte behandelt werden?**

Der Nutzer wird auf die Konflikte hingewiesen und erhält verschiedene Optionen:  
Er kann entweder seine Änderungen verwerfen, anfragen die Änderungen auf dem Server zu überschreiben oder seine Änderungen, sofern sie sich nicht komplett überschneiden, mit den Daten auf dem Server zu vereinigen.

## 5.5 Produktfunktionen

/LF10/ Eine Zugangsberechtigung soll mittels eines einfachen Loginvorgangs verifiziert werden.

Der Zugriff auf einzelne Daten soll je nach Berechtigung unterschiedlich erfolgen.  
➔ siehe Abschnitt 2.3: „Zielgruppen, Benutzerrollen und Verantwortlichkeiten“

**Was versteht sich unter einem einfachen Loginvorgang?**

Der Nutzer soll seinen Nutzernamen und Passwort eingeben und wird anschließend bei korrekter Eingabe eingeloggt.

**Wer hat die Benutzerverwaltung inne?**

Der Systemadministrator kümmert sich um die Freigabe und Verwaltung der Benutzer.

**Gibt es einen Benutzer, der auf alle Objekte lesenden sowie schreibenden Zugriff hat (in Hinblick auf eine Administrator-Rolle)?**

Der Systemadministrator hat Zugriff auf alle Objekte um das System gut administrieren zu können.

**Welche Rollen gibt es?**

Wie oben erwähnt gibt es den Hochzeitsmanager und den Unterhaltungsmanager. Es wäre allerdings auch praktisch den Systemadministrator als Rolle zu betrachten.

**Soll der Zugangsberechtigte als Entität betrachtet werden und wenn ja, welche Attribute hat er?**

Ja, das soll er. Er hat einen Benutzernamen, ein Passwort, eine oder mehrere Rollen und ist einer Person zugeordnet.

**Kann es sein, dass eine Person mehrere Systemnutzer innehat?**

Nein, dies kann nicht vorkommen.

**Wie sollen die Rollen dargestellt werden und braucht eine solche irgendwelche Attribute?**

Ein Rolle muss nicht als gesonderte Entität dargestellt werden. Sie ist lediglich ein Attribut in der Entität des Benutzers.

**Soll das Passwort mit speziellen Sicherheitsvorkehrungen abgespeichert werden?**

Um einen Datenschutz garantieren zu können, ist das verschlüsselte Speichern des Passwortes von unbedingter Notwendigkeit. Das Hashverfahren mit Salt eignet sich unserer Meinung nach sehr gut für diesen Zweck.

**Kann der Hochzeitsmanager nach der Hochzeit auf die Unterhaltungen Zugriff erlangen?**

Ja, damit er die Möglichkeit hat eventuelle Ereignisse und gegebenenfalls Medien privat zu archivieren.

**Soll das Brautpaar getrennte Accounts oder ein gemeinsamen verwenden?**

Das Brautpaar soll einen gemeinsamen Account verwenden.

/LF20/ Der jeweilige Benutzer muss die Möglichkeit haben, über eine grafische Benutzeroberfläche (GUI) alle für ihn relevanten Daten einfach und übersichtlich verwalten zu können.

**Was sind die relevanten Daten des jeweiligen Benutzers?**

Je nach Benutzerrolle und aktuellem Screen sollen spezielle Daten angezeigt werden. Somit soll z.B. auf der Startseite nur die wichtigen Informationen der Hochzeit angezeigt wird, wohingegen in der Aktionsliste die Aktionen in Kurzform gelistet werden sollen.

**Was versteht sich unter einer einfachen und übersichtlichen Verwaltung?**

Es soll schnell ersichtlich sein welche Sachen neu erstellt, verändert oder gelöscht werden können. Dies soll relativ intuitiv geschehen.



### Was für GUI Bestandteile soll der Nutzer zur Verfügung haben?

Es soll folgende graphischen Oberfläche geben: Aktion verwalten, Hilfsmittel verwalten, Caterer verwalten, Lebensmittel verwalten, Orte verwalten. Also im Endeffekt gibt es für jede Entität einen Dialog/GUI Bestandteil zum Verwalten derer.

### Welche weiteren Graphischen Oberflächen gibt es?

Weitere Oberfläche sind das Hauptmenü, wo der Nutzer die wichtigsten Informationen präsentiert bekommt: Datum, Neue Aktionen, Wichtige Änderungen, Online Status, sowie ob er synchron mit der Datenbank ist. Weiterhin hat er von dort die Möglichkeit mit Klick auf den entsprechenden Button zu folgenden Elementen zu gelangen: Ablaufplan, Aktionen verwalten, Hilfsmittel verwalten, Caterer verwalten, Lebensmittel verwalten und Orte verwalten.

Ein weitere Oberfläche ist der Ablaufplan, auf diesem hat der Nutzer die Möglichkeit in Form einer Timeline alle Aktionen in ihrer Reihenfolge zu sehen, dies wird in einem separaten Punkt genauer erläutert.

Ein zusätzlicher zentraler und wichtiger Bestandteil ist die Toolbar im oberen Bereich der Oberfläche: sie dient einerseits als Navigationshilfe, in dem der Nutzer dort die entsprechenden Bestandteile des Programms erreichen kann. Andererseits dient ebenso als Kontextmenü denn über den Drop Down Optionen hat der Nutzer die Möglichkeit Funktionen für den entsprechenden Screen auszuwählen. Beispiel: im Teil Aktion verwalten kann er dort Aktion erstellen auswählen und exportieren. Das funktioniert entsprechend bei jeder Entität Verwalten Oberfläche so. Bei allen gleich ist dort der Punkt synchronisieren, worüber der Nutzer die Möglichkeit hat die lokale Datenbasis mit der der Datenbank zu synchronisieren

### Gibt es sonstige kleinere GUI Bestandteile?

Ja es existierten diverse Konfirmationsdialoge, welche immer angezeigt werden, wenn der Nutzer Daten löschen bzw. ändern will oder mit ungespeicherten Daten weiternavigieren will bzw. verwerfen will. Ebenso gibt es im Fehlerfall entsprechende Fehlermeldungen. Zusätzlich gibt es auch einen Dialog der auftaucht, wenn der Nutzer das Programm schließen will, wo der Nutzer die Möglichkeit hat nochmal zu entscheiden ob er das Programm wirklich beenden will.

### Gibt es eine Konfigurationsgui?

Ja es gibt eine graphische Oberfläche in der der Nutzer Einstellungen vornehmen kann, diese ist über das Zahnrad neben dem Schließen Button erreichbar. Dort kann der Nutzer folgendes konfigurieren: Schriftgröße, Hintergrundfarbe, Schriftart und Schriftfarbe

### Gibt es sonst speziell gestaltete GUI's?

Nein die GUI's zur Verwaltung der Entitäten sind alle ähnlich (siehe Skizze), lediglich der Ablaufplan ist eine Besonderheit. Allerdings sind alle dafür ausgelegt übersichtlich zu sein. Und nur die wichtigsten Informationen zu enthalten um nicht überladen zu wirken.

### Über welchen GUI Bestandteil lassen sich neue Bestandteile z.B. eine Aktion erstellen?

In den entsprechenden Verwalten GUI's kann man neue Objekte erstellen und dies öffnet einen Dialog mit einem Formular, welches die Attribute des entsprechenden Objektes enthält. Diese kann der Nutzer ausfüllen und insofern alle Pflichtfelder ausgefüllt wurden wird das Objekt dann erstellt und gespeichert.

**Gibt es spezielle Anforderungen für die grafische Oberfläche?**

Nein, sie muss weder barrierefrei sein, noch irgendwelche Sonderfunktionen innerhaben.

**Sollte es rollenspezifische Oberflächen geben?**

Ja, wie oben erwähnt.

**Was versteht sich unter „verwalten“?**

Der Usecase „verwalten“ umfasst lesende, schreibende und löschende Aktionen. Zu diesen gehören die Abläufe anzeigen, erstellen, bearbeiten, sowie löschen.

**Welche Sachen sind löschtbar?**

Per se sind alle Daten löschtbar bis auf die Daten die von Werk aus in der Software gespeichert sind.

**Welche Nutzer dürfen was löschen?**

Der Systemadministrator darf alles löschen; der Hochzeitsmanager darf alles löschen bis auf Nutzer; der Unterhaltungsmanager kann nur die von ihm verwalteten Aktionen löschen.

/LF30/ Eine Hochzeitsveranstaltung fasst viele Einzelaktionen zusammen: anfangen von der standesamtlichen und kirchlichen Trauung, Buchung bzw. Reservierung von Veranstaltungsorten, Catering (Essen und Getränke getrennt verwaltbar) über Erstellung und/oder Druck von Einladungen, Tischkarten, diverse Besorgungen, Organisation und Buchung von Übernachtungen und dem Hochzeitsfahrzeug bis zur Organisation und Durchführung der Dekoration (in Standesamt, Kirche und Festsaal) uvm.

**Welche Attribute hat eine Hochzeitsveranstaltung?**

Die Hochzeitsveranstaltung soll einen Titel, eine Liste von Aktionen, das Hochzeitspaar, die Gäste, den Hochzeitsmanager, die Unterhaltungsmanager und den Caterer speichern.

**Kann eine Hochzeitsveranstaltung auch ein Motto haben?**

Ja, das kann sie. Allerdings soll dies optional sein.

**Soll das Catering einzeln verwaltbar sein?**

Um die Übersicht über eventuell verschiedene Caterer, sowie deren Kosten und Kontaktmöglichkeiten zu bewahren ist es sinnvoll diese verwaltbar zu machen.

**Kann auch eine Privatperson als Caterer fungieren?**

Ja, man legt dazu einfach einen neuen Caterer an und trägt als Kontaktperson eben diesen Person ein. Somit kann eine Hochzeitsveranstaltung mehrere Caterer haben, wobei von einem kommerziellen und optionalen mehreren privaten Caterern ausgegangen wird.

**Welche Attribute hat ein Cateringservice?**

Ein Caterer hat eine Person als Kontaktperson, einen Namen, eine Beschreibung (optional), eine Liste von Belegen, sowie eine Liste die Essen und Getränke enthält.

**Was soll in die Attribute Essen und Trinken gespeichert werden?**

In diesen Attributen soll das Essen und das Trinken des Caterers, welches er auf die Hochzeit liefert, festgehalten werden.



**Soll eine Vergleichsfunktion für Caterer implementiert werden?**

Ja, es soll eine solche Funktion geben. Es ist gut zuerst Angebote einzuholen und dann den passenden Caterer auszuwählen. So kann man unter Umständen einiges an Geld sparen.

**Wie werden Caterer, die nur zum Vergleichen da sind von denen unterschieden, die auch liefern werden?**

Ein Flag beim Caterer soll diese Unterscheidung ermöglichen.

**Welche Attribute hat das Essen?**

Das Essen hat einen Namen, eine Beschreibung (optional), eine Menge und eine Mengenbeschreibung.

**Wie wird das Essen und das Trinken dem Caterer zugewiesen?**

Mithilfe von Dialogen soll dies möglich sein. Man kann aus dme bisher existenten Essen und Trinken auswählen oder man kann neues erstellen.

**Was soll die Mengenbeschreibung speichern?**

In der Mengenbeschreibung soll die Maßeinheit der Menge stehen, z.B.kg, Flaschen.

**Welche Attribute hat das Trinken?**

Das Trinken hat einen Namen, eine Beschreibung (optional), eine Menge und eine Mengenbeschreibung.

**Soll es die Möglichkeit geben eine Gästeliste zu exportieren?**

Ja, dazu sollen die Kontaktdetails aller Gäste tabellarisch in einer pdf-Datei gespeichert werden.

/LF40/ Jede Aktion beginnt und endet zu einem bestimmten Zeitpunkt, es müssen verantwortliche Personen und Teilnehmer der Aktion benannt werden können. Die verantwortlichen Personen und Teilnehmer sollen automatisch per E-Mail benachrichtigt werden können (z.B. durch starten eines vorhandenen Mail-Tools über die grafische Benutzeroberfläche).

Jede Aktion kann an mehreren Orten stattfinden, mit Hilfsmitteln aus einer Liste durchgeführt werden. Anfallende Kosten (Rechnungen, Belege) sollen jeweils mit angegeben werden können.

Für den Benutzer soll es leicht möglich sein, die aktuellen Zustände aller Aktionen zu erkennen (geplant, in Arbeit, abgeschlossen, usw.) und zu ändern. Der Benutzer soll geeignet dabei unterstützt werden, bestimmte vorgegebene Aktionsarten anlegen und durchführen zu können. Dabei soll der Benutzer einfach erkennen können, welche Aktionen bereits angelegt sind und welche noch nicht.

Es soll darüber hinaus möglich sein, einer Aktion verschiedene Medien (Dokumente, Bilder, Videos, usw.) zuzuordnen.

**Welche Attribute hat eine Aktion?**

Titel, Beschreibung, Anfangsdatum, Enddatum, Art, Zustand, eine Liste von Orten, eine Liste von Organisatoren, eine Liste von Teilnehmern, eine Liste von Hilfsmitteln, eine Liste von Medien und eine Liste von Belegen. Außerdem soll gespeichert werden, ob die Aktion für das Brautpaar sichtbar ist.

**Soll bei dem Anfangs- und Enddatum auch die Zeit gespeichert werden?**

Ja, bei diesem Attributen soll auch die Zeit mitgespeichert werden

**Kann ein Hilfsmittel mehrmals in einer Aktion vorkommen?**

Da dies durchaus vorkommen kann, soll die Anzahl des jeweiligen Hilfsmittels mitgespeichert werden.

**Was ist mit dem Attribut Art gemeint?**

Die Art soll eine Kategorie angeben in welche die Aktion fällt.

**Soll es eine Vorauswahl von Arten geben und soll diese erweiterbar sein?**

Ja, es soll eine von Werk aus gegebene Auswahl geben, die sich erweitern lässt.

**Wie soll die Erweiterung von „Art“ durch den Benutzer durchgeführt werden?**

Er hat die Option alle „Arten“ in einem separaten Menüpunkt zu verwalten, dort erhält die Möglichkeit neue hinzuzufügen, sowie die andere zu löschen.

**Gibt es die Möglichkeit Prioritäten zu setzen?**

Der Nutzer hat die Option eine Priorität zu vergeben, was über das korrespondierende Attribut geregelt wird. Somit wird eine Priorisierung von Aktionen durch den Nutzer ermöglicht.

**Sind die Prioritätszustände vordefiniert?**

Ja, es gibt nur folgende vordefinierte Zustände: Hoch, Mittel, Niedrig.

**Erhält der Benutzer die Möglichkeit einen Aktion mit Notizen zu versehen?**

Ja, der Benutzer kann Notizen zu der entsprechenden Aktion angeben, wo er detailliertere Informationen zum aktuellen Status oder ähnliches festhält. Dies ist optional.

**Wie werden die Daten eingepflegt?**

Es soll einen Dialog (Formular) geben, in welche die Aktionen eingetragen werden können.

**Soll es eine Vorauswahl von Aktionen geben und wenn ja, welche?**

Nein, da bei den Aktionen jedwede Menge personalisierter Daten gespeichert werden soll, es eine Liste von Aktionen geben, die dem Nutzer vorgeschlagen werden anzulegen. Dabei sollen schon gewisse Felder der neuen Aktion vorausgefüllt werden.

**Welche Felder sollen schon ausgefüllt werden?**

Es sollen der Titel, die Beschreibung mit einem passenden Text, Meilenstein, versteckt, und der Zustand.

**Da die Attribute sehr spezifisch für eine Hochzeit sind, kann eine Aktion auch ohne direkte Hochzeit existieren, z.B. zur Wiederverwendung?**

Nein, da wie gesagt die Attribute sehr spezifisch sind, ist eine Aktion immer einer Hochzeit direkt zugewiesen. Zur Wiederverwendung soll man Templates erstellen können. Außerdem gibt es ja noch die Möglichkeit der Nutzung der Standardaktionen.

**Somit muss also eine Hochzeit auch keine Liste von Aktionen mehr speichern?**

Ja, dies ist somit nicht mehr nötig.

**Wie sollen die Templates funktionieren?**

Man soll bei der Erstellung einer neuen Aktion diese per Knopfdruck als Template speichern können. Dabei soll nicht auf die Vollständigkeit der Angaben geachtet werden. Diese Templates sollen mit der oben beschriebenen Import/Export-Funktion funktionieren. Das neue Template lässt sich dann einfach als Datei auf den Server hochladen, dies ist aber optional. Außerdem soll es natürlich die Möglichkeit geben bei der Erstellung einer Aktion ebenso ein Template zu laden. Damit sollen dann die im Template hinterlegten Felder ausgefüllt werden.

**Sind die Aktionen eher als To-Do Liste gedacht oder als Softwarefunktion?**

Die Aktionen sollen als Form einer To-Do Liste fungieren um einen besseren Überblick zu gewähren. Zusätzlich soll es die Möglichkeit geben die Aktionen als Ablaufplan einzusehen, sowie diesen exportieren zu können.

**Sollte es eine Erinnerungsfunktion geben?**

Ja, diese soll von dem Benutzer selbst eingeschaltet und definiert werden können. Sofern eine Erinnerung erfolgen soll, wird eine Email an den Nutzer geschickt.

**Wie sollen diese Erinnerungen gespeichert und vom wem gesendet werden?**

Die Erinnerungen sollen weder clientseitig noch in der Datenbank gespeichert werden. Es soll ein passender SMTP-Server verwendet werden, der Emails mit Verzögerung verschicken kann.

**Wie werden die Erinnerungen realisiert?**

Die Erinnerung wird mittels einer E-Mail realisiert, die der entsprechende Nutzer dann auf seine preferierte E-Mail erhält. Diese E-Mail wird automatisiert von der Software verschickt.

**Können Dienstleister hinzugefügt werden?**

Ja, diese sollen als Spezialversion einer Person existieren, am besten über einen separaten Attribut.

**Wie soll die Medien gespeichert werden?**

Diese sollen in ihrem Ursprungsformat auf den Server abgelegt werden.

**Existieren Medien als auch Objekte und wenn ja, welche Attribute hat dieses?**

Ein Medium hat einen Typ, der das Dateiformat beschreibt und eine URI(Uniform Resource Identifier), die den Pfad zu der Datei beschreibt als auch einen Titel.

**Was soll konkret in der Art eines Mediums gespeichert werden?**

Es soll dort die Dateiendung gespeichert werden.

**Welche Zustände haben Aktionen?**

Es gibt Standard Zustände, diese umfassen: Geplant, in Arbeit, beendet, sowie wartend. Zusätzlich hat der Nutzer die Möglichkeit eigene zu definieren.

**Welche Information soll zu einem Ort gespeichert werden?**

Es sollen Straße, Hausnummer, Adresszusatz, welcher optional ist sowie Stadt und Postleitzahl gespeichert werden. Außerdem soll er einen Titel bekommen können, der optional ist. Dafür erscheint es sinnvoll einen Ort als eigene Entität auszulagern.

**Was soll bei der Anzeige passieren, wenn der Titel nicht verfügbar ist?**

Es soll aus der Straße und der Hausnummer ein temporärer Titel gebildet werden.

**Soll in Hinblick auf die Internationalisierung auch das Land gespeichert werden und wenn ja wie?**

Ja das soll es. Es soll eine Auswahl der Länder geben wenn ein Ort erstellt wird. Standardmäßig soll das Land der aktuellen Sprache ausgewählt sein. Gespeichert werden soll der ISO 3166 ALPHA-3 Codes des Landes. Außerdem auch die Provinz (in manchen Ländern üblich) als simpler Text.

**Wie sollen die E-Mails verschickt werden, d.h. über welches Programm?**

Der E-Mail Versand geschieht über einen simplen SMTP Mail-Server auf den Servern.

**Was soll der Inhalt der E-Mails sein?**

Die Daten der Aktion in einer angemessenen lesbaren Version.

**Soll es möglich sein besondere Aktionen, beziehungsweise Aktionen mit hohem Stellenwert als „Meilenstein“ kennzeichnen?**

Ja das Hervorheben einer bestimmten Aktion als Meilenstein hilft wesentliche Punkte im Ablaufplan zu erkennen und von wenig wichtigeren zu differenzieren.

**Soll es eine Beleg/Rechnungsverwaltung geben und wenn ja, wie sollen diese Objekte verwaltet werden?**

Man kann zu einer Aktion Rechnungen/Belege als Objekt anhängen.

**Welche Attribute hat ein Beleg?**

Ein Beleg hat einen Titel, eine Beschreibung (optional), eine Liste von Medien und einen Zahl, die die Kosten beschreibt, eine Währung der Kosten, sowie eine Währungseinheit.

**Wie soll die Währung gespeichert werden?**

Die Währung sollen nach ISO 4217 als alphabetischer Code gespeichert werden. Dies Währungskürzel sollen nicht in der Datenbank gespeichert werden, sondern als Datei des Programms.

**Was ist unter Hilfsmitteln zu verstehen?**

Alles was nötig ist um eine Aktion durchführen zu können.

**Welche Attribute hat ein Hilfsmittel?**

Ein Hilfsmittel hat einen Titel, eine Beschreibung (optional), eine Art und eine Liste von Belegen.

**Soll die Art eines Hilfsmittels speziell kategorisiert werden könne, d.h. soll es eine Auswahl an Arten geben?**

Ja, es soll eine Auswahl an Arten geben. Dies soll der Einfachheit halber nicht erweiterbar sein. Wenn der Nutzer ein neues Hilfsmittel angelegt, kann er somit aus den gegebenen Arten eine Auswählen oder selber eine auswählen.

**Werden bei Änderungen auch die verantwortlichen Personen benachrichtigt werden?**

Ja, es werden bei Änderungen auch die verantwortlichen Personen benachrichtigt.

**Sind die Zustände einer Aktion vordefiniert oder ist es möglich eigene Zustände zu definieren?**

Es soll die oben genannten Zustände vordefiniert geben und welche, die der Nutzer selbst definieren kann.

**Wie kann der Benutzer eigene Zustände definieren?**

In einem separaten Menüpunkt kann er die Zustände verwalten und somit dort neue anlegen, beziehungsweise löschen.

**Soll es eine List aller Aktionen geben, oder wie soll die Darstellung passieren?**

Ja, innerhalb einer Liste, die sortierbar und filterbar ist. Außerdem soll für jede Aktion eine Detailansicht existieren.

**Sollen Personen über gewisse Ereignisse informiert?**

Die Information einzelner oder mehrerer Personen via E-Mail erscheint durchaus sinnvoll.

Werden immer alle Teilnehmer und verantwortlichen Personen per E-Mail informiert?

Im Standardfall werden alle zu dem Ereignis in Verbindung stehende Teilnehmer informiert. Allerdings ist es möglich die Notifikation auszustellen falls eine Person nicht wünscht benachrichtigt zu werden. Ebenfalls hat der auslösende Nutzer die Möglichkeit die Benachrichtigung nur an spezielle Nutzer zu senden.

/LF50/ Jede verantwortliche Person und jeder Teilnehmer kann bei den üblichen Kontaktdaten mehrere E-Mail-Adressen und mehrere Telefonnummern besitzen.

Gibt es eine Personenverwaltung?

Ja, es gibt eine Personenverwaltung. Mit der Registrierung wird ein neuer Benutzer angelegt.

Was versteht sich unter den üblichen Kontaktdaten?

Die Kontaktdaten eines Nutzers enthalten folgende Eigenschaften: Name, Adresse, Telefonnummern, sowie dessen EMail-Adresse(n).

Soll die Adresse genauso wie der Ort einer Aktion verwaltet werden?

Ja, dazu soll ein Ort verwendet werden.

Sollen noch weitere Personen verwaltet werden können, z.B. Dienstleister?

Ja, es soll möglich die Dienstleister zu verwalten.

Wie sollen diese Dienstleister verwaltet werden.

Dienstleister sollen wie normale Personen behandelt werden. Ein entsprechendes Attribut soll festlegen, ob die betreffende Person ein Dienstleister ist oder nicht.

Wer kann die Kontaktdaten ändern?

Der Systemadministrator kann die Kontaktdaten eines jeden Nutzers verwalten und ein Nutzer kann ebenfalls seine eigenen Daten anpassen.

Wer kann Personen anlegen?

Der Hochzeitsmanager und der Systemadministrator kann neue Personen anlegen. Dies ist nicht nötig, wenn die Person auch ein Systemnutzer ist, denn dann wird eine Person automatisch angelegt..

/LF60/ Das Catering kann entweder von einem kommerziellen Catering-Service als auch von ausgewählten Personen durchgeführt werden.

Kann das Catering noch von anderer Stelle durchgeführt werden?

Ja, eine Mischung aus den oben genannten Möglichkeiten soll auch möglich sein.

Sollte es eine Suchfunktion von Cateringservices geben?

Nein, für die private Software wäre die Komplexität zu hoch. Für unsere Version der Software ist dies auch nicht nötig, da wir unsere Dienstleister in einem anderen System selbst verwalten.

Soll der Speiseplan/ das Essen verwaltet werden können?

Ja, diese Daten sollen über Objekte hinzugefügt werden können. Ein Speiseplan/Karte soll generierbar sein.

### Wie genau soll der Generierungsprozess ausschene?

Nachdem der Nutzer die Generierung angetriggert hat, kann er auswählen zu welchem Gang eine Speise gehört. Kategorien wie kalt, alkoholisch, etc. wird in der Beschreibung des Nahrungsmittels gespeichert. Diese kann auf dem Speiseplan angezeigt werden. Der Nutzer kann aber auch weitere Details angeben, ohne dass diese in der Datenbank gespeichert werden. Eine Getränkearte ist simultan dazu erstellbar.

### Kann man diese Speisekarte abspeichern?

Ja, und zwar auf zwei Weisen. Erstens soll man den Speiseplan als Bild (png, jpg) oder als pdf-Datei abspeichern können. Zweitens, damit man nicht immer alle Metadaten neu angeben muss, soll die Speisekarte wie oben genannt auch exportiert/importiert werden können. Diese Datei kann man dann auf den Server hochladen.

/LF70/ Die oben erwähnte Liste der Hilfsmittel soll auf einfache Weise erweiterbar und zuweisbar sein. Sie sollen für sämtliche Hochzeitsveranstaltungen im System verfügbar sein.

### Soll es eine Standard-Hilfsmittelauswahl von Werk aus geben?

Ja, es soll eine solche Auswahl geben. Da eine solche Liste sich aber mit der Zeit ändern kann, soll diese von unserem Firmenserver bezogen werden können, d.h. beim ersten Start des Programms bei einer Privatperson wird die Liste heruntergeladen.

### Kann man diese Hilfsmittel aus dem System löschen?

Die standardmäßigen Hilfsmittel nicht, die manuell hinzugefügt schon.

### Soll es eine Suche innerhalb der Hilfsmittelliste geben?

Ja, es soll dem Nutzer möglich sein innerhalb der Hilfsmittel zu suchen.

### Wie sollen neue Hilfsmittel angelegt werden können?

Über einen Dialog sollen diese angelegt werden können.

### Kann man bei der Erzeugung von Hilfsmitteln entscheiden ob diese global im System verfügbar sein sollen?

Nein, alle Hilfsmittel sind global verfügbar.

### Kann es in Hinblick auf Unterhaltung versteckte Hilfsmittel geben?

Nein, es soll keine versteckten Hilfsmittel geben, da es sonst in Hinblick auf die Kostenkontrolle zu Problemen kommen könnte.

/LF80/ Zur Kostenkontrolle soll es möglich sein, sämtliche bisher angefallenen Kosten auf einfache Weise addieren zu können. Zur Kostenabschätzung sollen auch geschätzte Kosten angegeben und addiert werden können.

### Soll die Kostenkontrolle für jeden zugänglich sein?

Nein, nur für den Hochzeitsmanager, da es für die anderen Nutzer nicht von unbedingter Relevanz ist. Ebenfalls spielt der Aspekt des Datenschutzes eine Rolle.

### Soll jeder Kosten hinzufügen können?

Jeder der Aktionen definieren kann und/oder die Hochzeit mitplant kann Kosten hinzufügen.

### Sollte es eine Analyse geben um doppelte Kosten auszumerzen?

Nein, da doppelte Kosten zu entdecken zu schwierig wäre und den Nutzer überfordern würde.



Wie sollen geschätzte Kosten erkennbar sein?

Sie bewegen sich in einen Bereich (10-20€).

Sollen geschätzte Kosten und angefallene Kosten zusammen addiert oder getrennt behandelt werden?

Es sollen zwei Werte verfügbar sein. Einmal alle angefallenen Kosten und die geschätzten Kosten mit ihren Minimalwert und zudem auch alle angefallenen Kosten mit dem Maximalwert der geschätzten Kosten.

Wie sehen die geschätzten Kosten aus (ein Wert oder ein min/max Wert)?

Sie bewegen sich in einem min-max-Wert.

Gibt es Kosten, die nicht mit addiert werden sollen?

Der Nutzer hat die Option Ausgaben als „privat“ zu kennzeichnen, diese Kosten (also Belege) werden dann nicht dazu addiert und tauchen nicht in der Kostenkontrolle auf.

In Hinblick auf die oben erwähnte Möglichkeit der Angabe der Währung, wie soll damit innerhalb der Kostenkontrolle umgegangen werden?

In der Kostenkontrolle soll man die Währung auswählen können. Alle Kosten, die nicht in dieser Währung sind, sollen mit dem aktuellen Umrechnungskurs umgerechnet werden. Dieser soll von einer geeigneten API bezogen werden.

/LF90/ Die Auswahl der Daten soll möglichst über (eventuell durchsuchbare) Auswahllisten erfolgen. Dies gilt vor allem für Zuordnungen von ausgewählten Personen zu den Aktionen usw.

Wie soll diese Suche aussehen?

Es soll eine Stichwortsuche vorhanden sein, um den Nutzer in seiner Suche zu unterstützen, da eine Volltextsuche oft langsam und überfordernd ist. Zusätzlich ist eine Volltextsuche oft nicht notwendig.

Kann man gezielt filtern?

Ja, es ist möglich nach speziellen Attributen zu filtern.

Wo sollen diese Auswahllisten Verwendung finden?

Überall wo man mehrer Arten einer Entität zue einer anderen hinzufügen kann.

Sollen nur Daten aus der aktuellen Hochzeit verwendet werden können oder sollen auch systemweite Daten angezeigt werden?

Es sollen nur Daten zur aktuellen Hochzeit verfügbar sein, da dies die Suchgeschwindigkeit optimiert.

/LF100/ Vor dem Hinzufügen von neuen Daten soll eine Überprüfung stattfinden, ob diese eventuell schon vorhanden sind. Das gilt in besonderem Maße für Personen und Aktionen.

Wie soll der Nutzer auf Doppellungen hingewiesen werden?

Mit einer Warnung als Popup, die die doppelten Einträge verlinkt.

Wie erkennt man Doppelungen?

Doppelungen werden lediglich anhand der Aktionsart und dem Titel der Aktion festgestellt. Dies reicht allerdings aus um das Problem zu vermeiden.

## 5.6 Produktdaten

/LD10/ Die Daten sollen zentral verwaltet und in einer Datenbank abgespeichert werden.

Was für eine Datenbank soll Verwendung finden?

Dies ist der implementierenden Firma freigestellt. Allerdings haben wir intern gute Erfahrungen mit relationalen Open-Source-Datenbanken gemacht.

Wie soll das Datenbankschema aussehen?

Die implementierende Firma soll hierzu ein passendes Schema erarbeiten.

## 5.7 Produktleistungen

/LL10/ Das Laden gewünschter Daten soll für eine sinnvolle Benutzung im Sekundenbereich erfolgen.

Wie viele Sekunden sind das Maximum?

10 Sekunden soll die maximale Wartezeit sein. Allerdings wird eher eine Zeit von unter 3 Sekunden erwartet.

Was soll bei einer sehr langsamen Internetverbindung passieren?

Es soll dem Nutzer eine Warnung ausgegeben werden und z.B. Listeneinträge nur nach und nach geladen werden.

/LL20/ Die Anzahl der zu verwaltenden Elemente wird auf ca. 100000 geschätzt.

Bezieht sich diese Zahl auf die firmeninterne Software und/oder auf die private?

Diese Angabe bezieht sich nur auf die Firmensoftware. Bei der privaten wird sie 5000 geschätzt.

/LL30/ Die Daten müssen bei unserer eigenen Verwendung aus rechtlichen Gründen 10 Jahre online verfügbar sein.

Inwieweit muss bei Updates auf diese Verfügbarkeit geachtet werden?

Sofern ein Update (auch von Drittanbietern) nötig ist, muss auf die Kompatibilität geachtet werden. Sollte dies nicht möglich sein, müssen die alten Daten transformiert werden. Die Ursprungsdaten müssen dabei allerdings als Nachweis dennoch erhalten bleiben.

/LL40/ Um bei Anschaffungen und Neuerungen flexibel zu bleiben, ist auf Plattformunabhängigkeit besonders zu achten.

Welche Plattformen sollen berücksichtigt werden?

Die Software soll auf Linux, OS X sowie Windows lauffähig sein.

Gibt es Plattformen, die präferiert werden?

Nein, wir haben keine Präferenzen bezüglich der Plattformen.



## 5.8 Qualitätsanforderung

Produktqualität	sehr gut	gut	normal	nicht relevant
Funktionalität	X			
Zuverlässigkeit		X		
Effizienz			X	
Benutzbarkeit	X			
Änderbarkeit			X	
Übertragbarkeit			X	
Gestaltung	X			

## 5.9 Entitäten und Attribute

- Person
  - Name
  - E-Mailadressen
  - Adresse
  - Telefonnummer(n)
  - istDienstleister
- Systemnutzer
  - Nutzernamen
  - Passwort
  - Rollen
  - Person
- Hochzeitsveranstaltung
  - Titel
  - Gäste
  - Brautpaar
  - Hochzeitsmanager
  - Caterer (auch mehrere)
  - Motto
  - Unterhaltungsmanager
- Hilfsmittel
  - Titel
  - Beschreibung
  - Art
  - Belege
- Aktionshilfsmittel
  - Aktion
  - Hilfsmittel
  - Anzahl

- Aktion
  - Hochzeit
  - Anfang
  - Ende
  - Teilnehmer
  - Orte
  - Titel
  - Notizen
  - Priorität
  - Beschreibung
  - Organisator(en)
  - Art
  - Hilfsmittel
  - Medien
  - Belege
  - istVersteckt
  - Zustand
  - istMeilenstein
- Caterer
  - Name
  - Beschreibung
  - Kontaktperson
  - Belege
  - Essen
  - Trinken
  - zumVergleich
- Ort
  - Titel
  - Straße
  - Hausnummer
  - Adresszusatz
  - Stadt
  - Postleitzahl
  - Provinz
  - Land
- Beleg
  - Titel
  - Beschreibung
  - Medien
  - Kosten
  - Währung
  - istPrivat
- Medium
  - Titel
  - Typ
  - URI

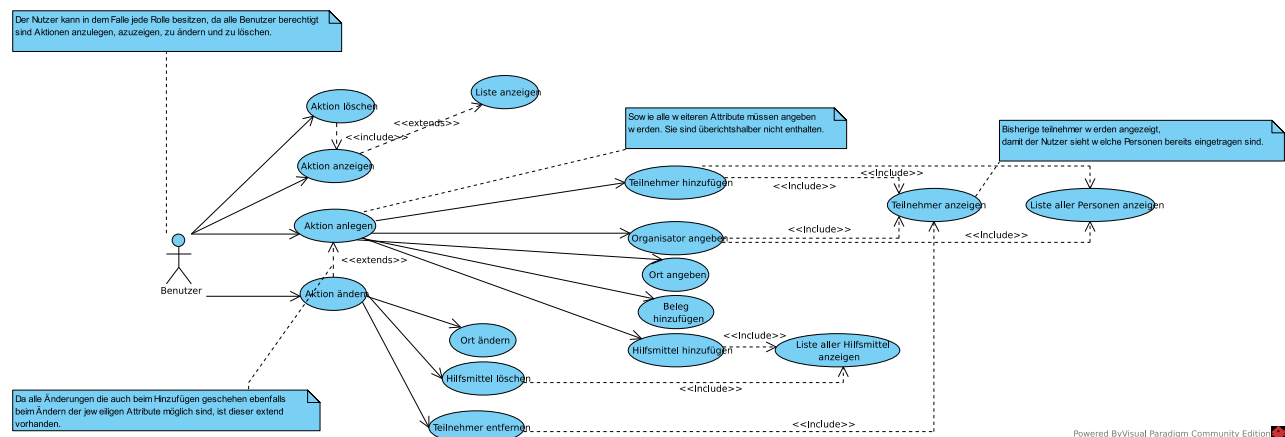
- Trinken
  - Titel
  - Beschreibung
  - Menge
  - Mengenbeschreibung
- Essen
  - Titel
  - Beschreibung
  - Menge
  - Mengenbeschreibung

## 6 Use-Case-Diagramme

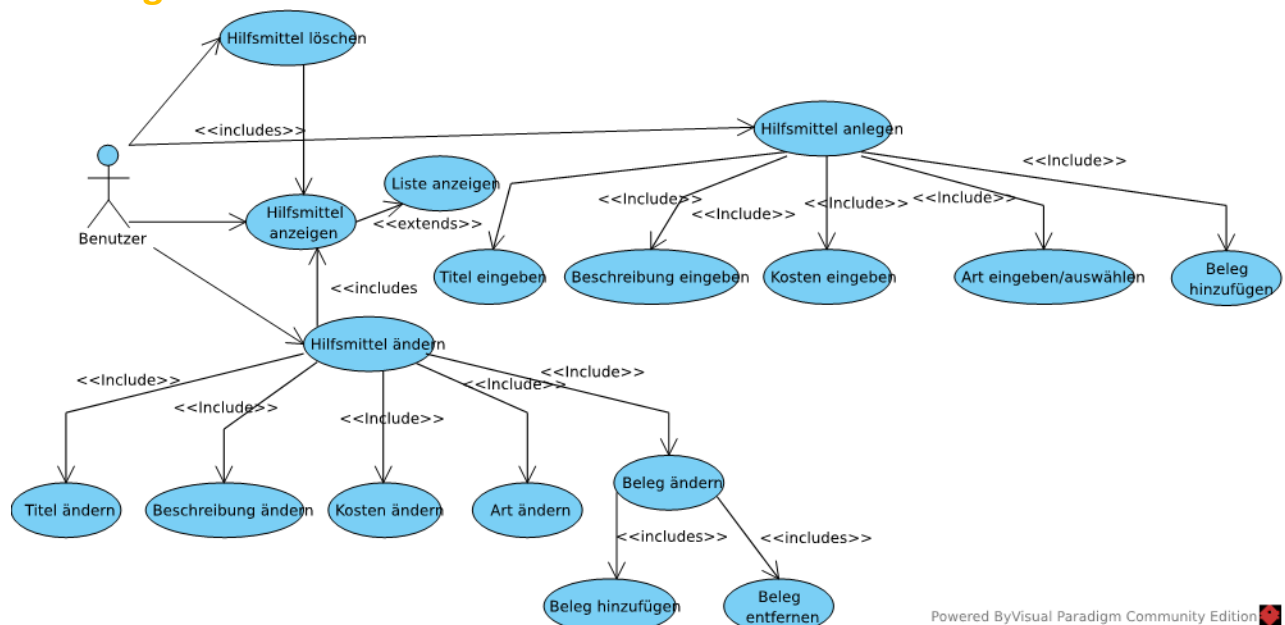
### 6.1 Einleitung

Das Use Case Diagramm, oder auch als Anwendungsfalldiagramm bezeichnet, beschreibt alle möglichen Anwendungsfälle innerhalb einer Anwendung. Es besteht im Endeffekt lediglich aus Akteuren sowie Anwendungsfälle, welche mehr oder weniger detailliert dargestellt werden können. Es stellt somit das erwartete Verhalten eines Systems dar und wird verwendet um die Anforderungen an ein System zu spezifizieren.

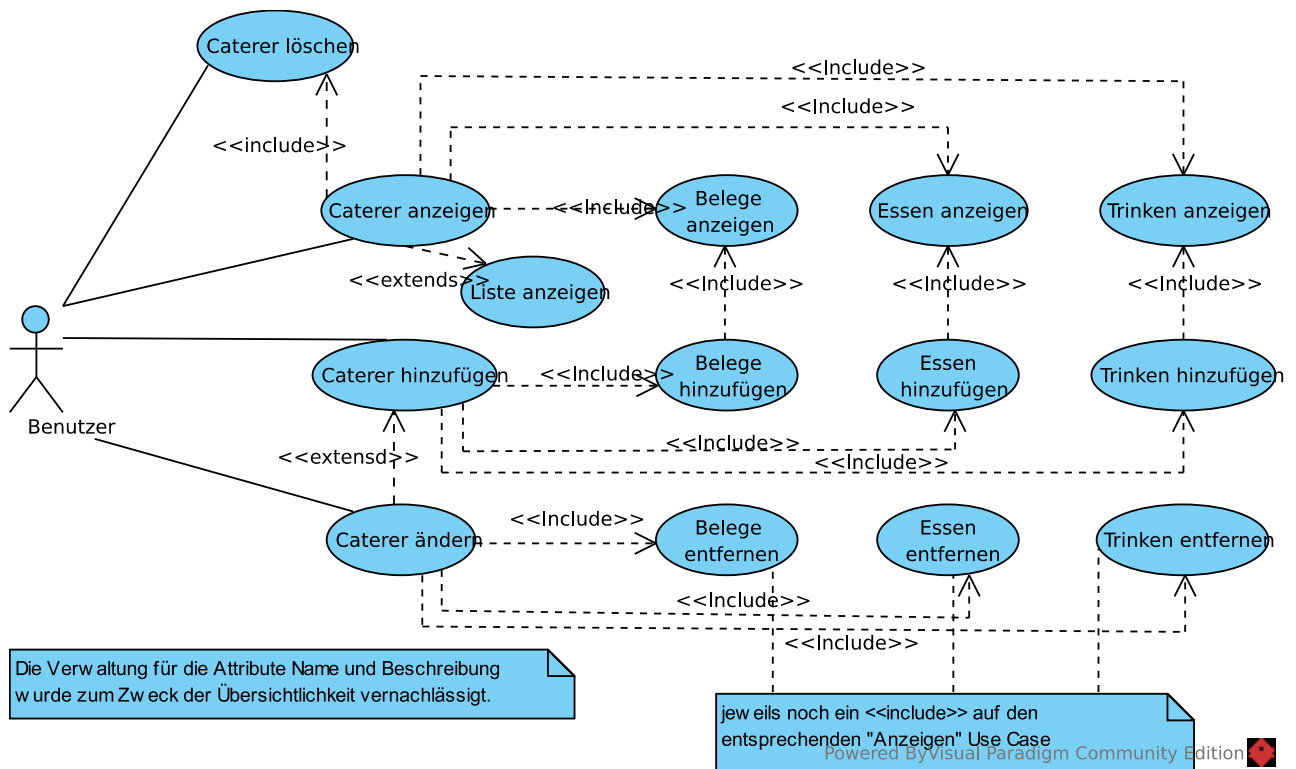
### 6.2 Diagramm Aktion verwalten



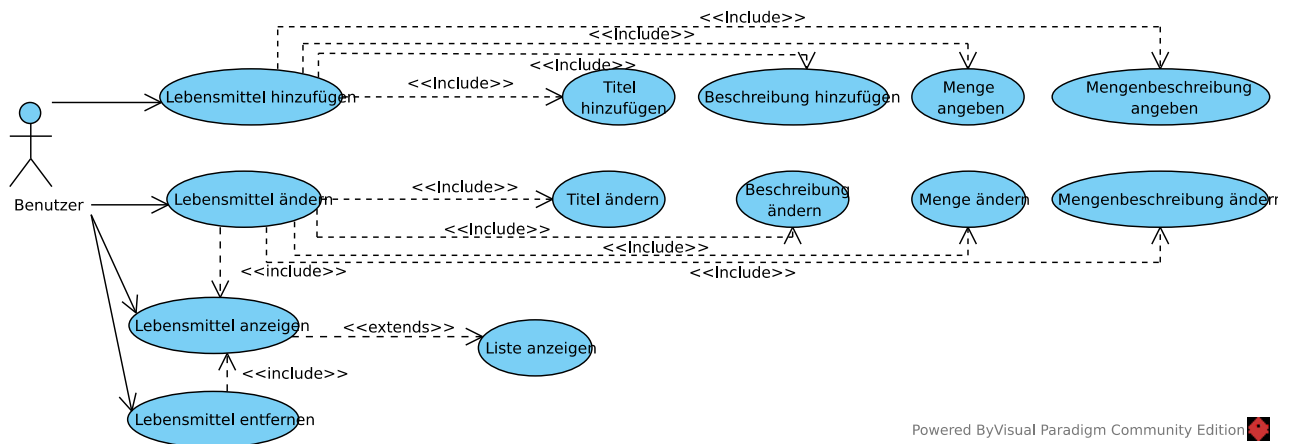
### 6.3 Diagramm Hilfsmittel verwalten



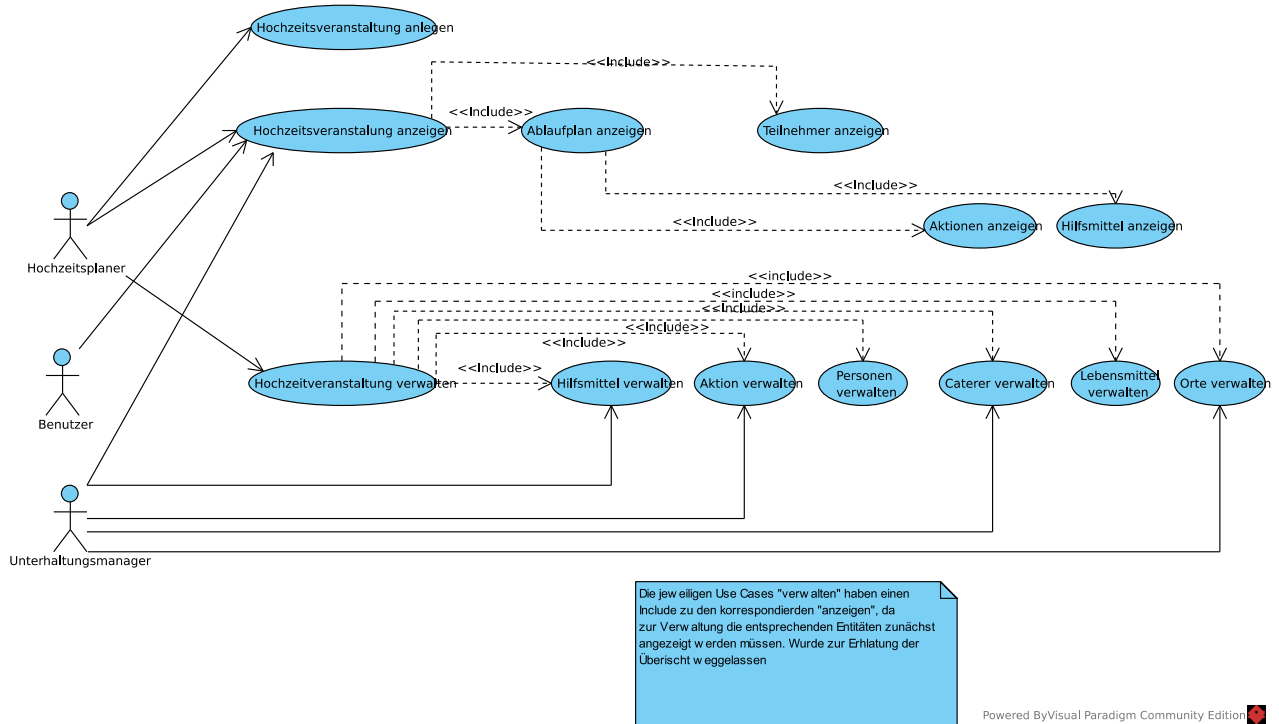
## 6.4 Diagramm Caterer verwalten



## 6.5 Diagramm Lebensmittel verwalten



## 6.6 Diagramm Hochzeitsveranstaltung verwalten



Powered By Visual Paradigm Community Edition

## 6.7 Akteure

### 6.7.1 Hochzeitsplaner

Der Hochzeitsplaner ist der zentrale Akteur, da er praktisch die hauptverantwortliche Person für die Hochzeitsplanung ist. Er hat auf fast alles Zugriff, bis auf die Nutzerverwaltung, welche der Administrator inne hat.

### 6.7.2 Unterhaltungsmanager

Der Akteur Unterhaltungsmanager kommt dann ins Spiel sobald es um Aktionen geht. Er kann solche anlegen und diese ebenfalls verwalten.

### 6.7.3 Administrator

Der Akteur Administrator ist lediglich für die Personenverwaltung relevant, da dies seine Hauptaufgabe, neben der generellen Systemadministration, ist. Allerdings verfügt er über eine Sonderstellung, da er als Administrator natürlich über alle Rechte verfügt und somit auch Zugriff auf alles hat.

### 6.7.4 Benutzer

Der Akteur Benutzer ist der Basisakteurtyp, das heißt: der Hochzeitsplaner, der Unterhaltungsmanager, sowie der Administrator sind ebenfalls Benutzer, haben jedoch zusätzliche Berechtigungen. Dieser Akteur dient somit als Verallgemeinerung für Anwendungsfälle die theoretisch von jedem Benutzer ausgeführt werden kann.

## 6.8 Use Cases

### 6.8.1 Hochzeitsveranstaltung managen

Der Anwendungsfall Hochzeitsveranstaltung managen ist der „Haupt“-Use Case, denn er beinhaltet fast alle weiteren Use Cases der Anwendung. Ebenfalls ist die Verwaltung auch der Hochzeit auch der Hauptaspekt des Programmes. In diesem Anwendungsfall sind drei Akteure vertreten: Hochzeitsplaner, Unterhaltungsmanager und Benutzer.

Der Hochzeitsplaner hat eine zentrale Rolle in diesem Diagramm, denn er ist für die Verwaltung zuständig und kann somit eine Hochzeitsveranstaltung anlegen, anzeigen, sowie verwalten. „Hochzeitsveranstaltung anlegen“ ist ein Fall der lediglich durch den Hochzeitsplaner selbst ausgeführt wird. Der Anwendungsfall Hochzeitsveranstaltung verwalten enthält weitere Use Cases, dazu zählen: Hilfsmittel verwalten, Aktion verwalten, Personen verwalten, Caterer verwalten, Lebensmittel verwalten und Orte verwalten. Diese Use Cases sind noch detaillierter betrachtbar, allerdings zur Wahrung der Übersichtlichkeit nicht in diesem Diagramm enthalten. Ebenso wurde der Übersichtlichkeit wegen die Include Pfeile zu den korrespondierenden „anzeigen“-Use Cases weggelassen.

Da ein Unterhaltungsmanager nicht die komplette Hochzeit verwalten kann, hat er dementsprechend nicht auf alle Bestandteile des Anwendungsfalles Zugriff. Daher kann nur die Use Cases: Hilfsmittel verwalten, Aktionen verwalten, Orte verwalten und Caterer verwalten ausführen.

Ein weiterer Bestandteil ist der separater Anwendungsfall „Hochzeitsveranstaltung anzeigen“, welcher von allen in diesem Diagramm vorhandenen Akteuren ausführbar ist, da sich alle Benutzer die Daten der Hochzeit anzeigen lassen können. Es gibt jedoch eine Ausnahme: Falls eine Aktion dem Hochzeitspaar verborgen bleiben soll und dies durch den Unterhaltungsmanager so gesetzt ist, kann das Hochzeitspaar diese Aktion nicht sehen. Dieser Use Case enthält folgende weitere Use Cases: Ablaufplan anzeigen sowie Teilnehmer anzeigen.

### 6.8.2 Aktion verwalten

Der Use Case „Aktion verwalten“ ist wichtiger Bestandteil der Anwendung, da die Aktionen ein Kernbestandteil der Hochzeit sind. Er besteht im Wesentlichen aus den folgenden Anwendungsfällen: Aktion anlegen, Aktion anzeigen, Aktion ändern, Aktion löschen. Es handelt sich um eine Verfeinerung.

„Aktion anzeigen“ ist ein dieser separater Faller, bei diesem der Akteur, in dem Fall ein beliebiger Benutzer, sich eine Liste der relevanten Aktionen anzeigen lassen kann. Er erbt vom dem Use Case „Liste anzeigen“, was ein generalisierter Use Case zum Anzeigen von Listen ist.

Der Anwendungsfall „Aktion anlegen“ stellt das Anlegen einer Aktion dar, welcher ebenfalls aus kleineren Use Cases besteht, dazu gehören Teilnehmer hinzufügen und Organisator angeben. Diese inkludieren beide weiterhin die Fälle Teilnehmer anzeigen, sowie Liste aller Nutzer anzeigen. Dies ist notwendig damit Nutzer hinzugefügt werden können, beziehungsweise als Organisator angegeben werden können. Der Fall Organisator angeben ist nur von Bewandnis, insofern der Ersteller nicht der Organisator sein sollte. Weiterhin enthält „Aktion anlegen“ die Use Cases: Ort angeben, Beleg anhängen, Hilfsmittel hinzufügen(include auf „Liste aller Hilfsmittel anzeigen“, da die Hilfsmittel angezeigt werden müssen, bevor sie hinzugefügt werden können). Ebenfalls würden hier noch der Use Case „... angeben“ für alle Attribute der Entität Aktion vorhanden sein, diese sind aber der Übersichtlichkeit halber weggelassen.

Der Use Case „Aktion ändern“ dient dazu eine bereits angelegte Aktion zu modifizieren. Dazu kann der Benutzer die einzelnen Attribute der Entität Aktion ändern. Der Anwendungsfall erbt durch einen „extend-Pfeil“ von „Aktion anlegen“, da alle Use Cases dessen auch in dieser Anwendung finden. Zudem sind unter anderem die Anwendungsfälle „Ort ändern“, „Hilfsmittel löschen“, sowie „Teilnehmer löschen“ Bestandteil dieses Falles.

Letztlich kann der Benutzer auch mit dem Use Case „Aktion löschen“ angelegte Aktionen wieder entfernen, nachdem sie angezeigt wurden.

### 6.8.3 Hilfsmittel verwalten

Der Use Case „Hilfsmittel“ ist ähnlich wie die vorangehenden Use Cases ein Fall in dem eine Entität verwaltet wird. Das heißt es werden die Prozesse Anzeigen, Erstellen, Ändern und Löschen abgebildet. Akteur ist in diesem Fall ein beliebiger Nutzer, der die entsprechende Berechtigung hat.

Das Diagramm besteht aus vier wesentlichen Bestandteilen: Hilfsmittel löschen, Hilfsmittel anzeigen, Hilfsmittel anlegen, sowie Hilfsmittel ändern. Hilfsmittel anzeigen: Dieser Use Case zeigt eine Liste oder auch nur 1 Hilfsmittel an, dafür gibt es ein extend auf den Use Case „Liste anzeigen“. Der Anwendungsfall Hilfsmittel löschen tritt auf wenn der Nutzer ein Hilfsmittel entfernen möchte, dafür wird ihm die Liste der Hilfsmittel angezeigt, weswegen der entsprechende Use Case ein include hat. Selbes Include gilt ebenfalls für den Anwendungsfall Hilfsmittel ändern, mit welchem der Nutzer die Attribute einer Entität Hilfsmittel verändern kann. Dafür wird ebenfalls eine Anzeige der Hilfsmittel benötigt, wo der korrespondierende Anwendungsfall wieder ins Spiel kommt. Weiterhin besitzt der Use Case includes auf folgende atomaren Anwendungsfälle, welche jeweils das entsprechende Attribut modifizieren: „Titel ändern“, „Beschreibung ändern“, „Kosten ändern“, „Art ändern“, sowie „Beleg ändern“. Letzterer hat ebenfalls Includes auf die Use Cases „Beleg hinzufügen“ und „Beleg entfernen“. Letztlich kann der Akteur ebenfalls ein Hilfsmittel anlegen, wofür der Anwendungsfall „Hilfsmittel anlegen“ existiert. Dieser hat wiederum includes auf die atomaren Use Cases: „Titel eingeben“, „Beschreibung eingeben“, „Kosten eingeben“, „Art eingeben“ und „Beleg hinzufügen“, welche jeweils das entsprechende Attribut der Entität setzen.

### 6.8.4 Caterer verwalten

Das Diagramm Caterer verwalten bildet den Use Case „Caterer verwalten“ ab. Die Verwaltung der Attribute Name sowie Beschreibung wurden zur Wahrung der Übersichtlichkeit vernachlässigt. Akteur ist in diesem Fall ein beliebiger Nutzer, der die entsprechende Berechtigung hat. Es werden allen möglichen Anwendungsfälle dargestellt, die im Rahmen der Verwaltung von der Entität „Caterer“ auftreten können. Dazu gehören maßgeblich die Fälle: „Caterer anzeigen“, „Caterer hinzufügen“, „Caterer ändern“, sowie Caterer löschen“. Der Anwendungsfall Caterer anzeigen hat eine extend auf Liste anzeigen, da er eine List von Catereren anzeigt. Dafür benötigt er Includes auf Use Case „Anzeigen“ der entsprechenden Entitätsmengen, die er als Attribute besitzt. Dazu gehören: „Beleg anzeigen“, „Essen anzeigen“, „Trinken anzeigen“. Der Anwendungsfall „Caterer löschen“ includiert „Caterer anzeigen“, da zum Löschen der entsprechenden Entität sie zunächst dem Nutzer angezeigt werden muss. Ein weiterer Use Case ist Caterer hinzufügen, in welchem der Akteur eine neue Entität vom Typ „Caterer“ erstellt. Dafür besitzt dieser (Use Case) includes auf „Beleg hinzufügen“, „Essen hinzufügen“ und „Trinken hinzufügen“. Diese Use Cases wiederum besitzen ein include auf die jeweils korrespondierenden „Anzeigen“ Anwendungsfall, da die entsprechenden Entität vor dem hinzufügen angezeigt werden müssen. Der letzte Use Case in diesem Diagramm ist „Caterer ändern“ welcher ein extends auf Caterer hinzufügen hat, da er diesen erweitert. Er erhält zusätzliche includes auf „Beleg entfernen“, „Essen entfernen“ sowie „Trinken entfernen“, diese



hätten jeweils auch ein Include auf den entsprechenden „Anzeigen“ Use Case, diese wurden aber aufgrund der Übersichtlichkeit nicht eingefügt.

### 6.8.5 Lebensmittel verwalten

Das Diagramm „Lebensmittel verwalten“ bildet den Use Case „Lebensmittel verwalten“ ab. Akteur ist in diesem Fall ein beliebiger Nutzer, der die entsprechende Berechtigung hat. Es werden allen möglichen Anwendungsfälle dargestellt, die im Rahmen der Verwaltung von der Entität „Lebensmittel“ auftreten können. Dazu gehören maßgeblich die Fälle: „Lebensmittel anzeigen“, „Lebensmittel hinzufügen“, „Lebensmittel ändern“, sowie „Lebensmittel löschen“. Der Use Case „Lebensmittel anzeigen“ hat ein extend auf List anzeigen, da es eine Liste von Lebensmitteln anzeigen kann. Diesen inkludiert der Anwendungsfall „Lebensmittel entfernen“, da vor dem Löschen einer Entität diese zunächst angezeigt werden muss. Weiterhin existiert der Use Case Lebensmittel hinzufügen, welcher den Prozess abbildet wenn der Nutzer eine neue Entität vom Typ „Lebensmittel“ erstellen möchte. Dafür hat dieser Anwendungsfall includes auf die atomaren Fälle „Titel hinzufügen“, „Beschreibung hinzufügen“, „Menge angeben“ sowie „Mengenbeschreibung angeben“. Diese setzen jeweils das entsprechende Attribut der Entität. Ähnlich ist der Use Case „Lebensmittel ändern“ aufgebaut. Jedoch erstellt dieser kein neues Objekt, sondern modifiziert ein bestehendes. Dafür besitzt er die entsprechenden Includes auf die Anwendungsfälle „Titel ändern“, „Beschreibung ändern“, „Menge ändern“ sowie „Mengenbeschreibung ändern“. Diese sind ändern jeweils das korrespondierende Attribut der Entität.

## 7 Analyse-Klassendiagramm

## 7.1 Einleitung

Das Analyseklassendiagramm stellt den ersten Entwurf der Klassen dar, die man aus dem Lastenheft identifizieren kann. Hierbei geht es vor allem um die Klassen, die Entities abbilden und nicht um solche, die nur um des Programms willen existieren (wie z.B. Datenbankmanager, etc.).

## 7.2 Hinweis

Da bei der Modellierung nicht die Personenverwaltung berücksichtigt werden muss, wurde der Systemnutzer nicht modelliert.

Auf die Labels der Relationen wurde aus Übersichtszwecken verzichtet. Außerdem werden die Attribute die einer Rolle zugewiesen sind immer auch als Attribut vermerkt, d.h. die Rollen ersetzen keine Attribute

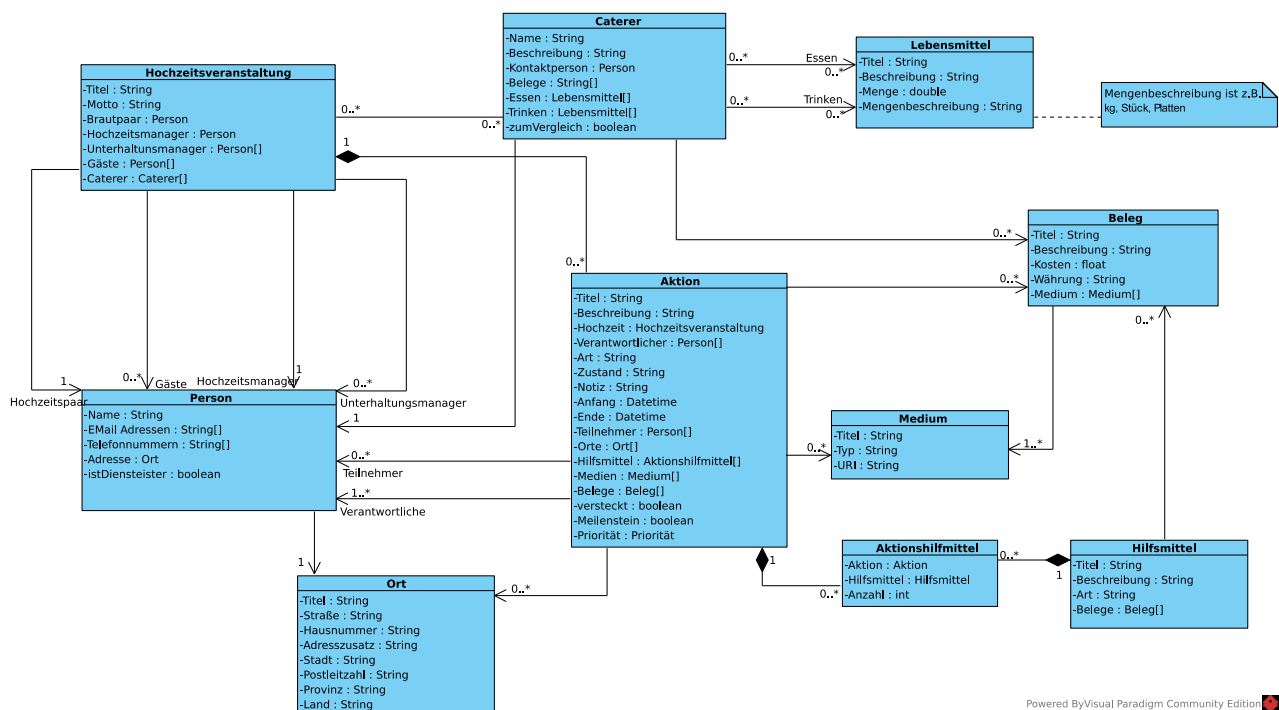
## 7.3 Pattern

Für das Diagramm werden das Rollen- und das Koordinatorpattern verwendet.

Das Rollenpattern bietet sich deswegen an, da einige Klassen mehr als eine Referenz auf eine Klasse haben. Daher agieren sie hierbei als Rollen mit entsprechenden Rollennamen.

Das Koordinatorpattern wird nur einmal verwendet, mehr dazu an der passenden Stelle.

## 7.4 Diagramm



## 7.5 Entities

### 7.5.1 Hochzeitsveranstaltung

Eine Hochzeitsveranstaltung hat einen Titel und ein Motto. Um die verschiedenen Rollen, die eine Person innerhalb einer Hochzeitsveranstaltung innehaben kann abzubilden, benutzt sie hier das Rollen-Pattern. Es gibt zwei Personen, die das Brautpaar abbilden, 1 Person als Berater, 0..\* als Gäste und als Unterhaltungsmanager

### 7.5.2 Person

Eine Person hat einen Namen, eine Liste von Email Adressen und eine Liste von Telefonnummern, beiden vom Typ String. Für die Telefonnummern wurde der Typ String gewählt da man Telefonnummern eher selten als wirkliche Zahlen betrachtet. Außerdem in einer 1..1 Relation der Adresse ein Ort zugewiesen. Als letztes kennzeichnet ein boolsches Flag, ob es sich bei der Person um einen Dienstleister handelt.

### 7.5.3 Aktion

Eine Aktion hat einen Titel und eine Beschreibung. Außerdem ist die Aktion einer Hochzeit per Komposition zugeordnet, denn es macht keinen Sinn eine Aktion zu behalten, wenn die dazugehörige Hochzeit nicht mehr existiert. Des Weiteren besitzt eine die Aktion eine Art, einen Zustand und eine Notiz. Diese alle sind vom Typ String. Der Anfang und das Ende der Aktion werden als DateTime realisiert. Die Teilnehmerliste ist eine 0..\* Relation zu den Personen, die Ortsliste eine 0..\* zu den Orten. Es gibt auch eine Liste an Medien und Belegen die mit 0..\* mit den Medien bzw. Belegen verbunden ist. Die Hilfsmittel sind mit den Aktionshilfsmitteln über eine Komposition verbunden. Das versteckt und das Meilenstein Attribute sind vom Typ boolean. Außerdem gibt es noch eine 1..\* Relation zu den Personen, die die Verantwortlichen beschreibt. Als letztes regelt ein int die Priorität.

### 7.5.4 Caterer

Der Caterer besitzt einen Namen und Beschreibung. Außerdem besitzt er 0 bis beliebig viele Belege. Um die Generalisierung des Essens und des Trinkens wieder zu spezialisieren hat er ja ein Attribut für das Essen und das Trinken vom Typ Lebensmittel mit der Kardinalität 0..\*. Hier greift das Rollen-Pattern. Des Weiteren hat er eine (1) Kontaktperson vom Typ Person. Ein boolsches Flag namens zumVergleich regelt, ob dies ein Caterer ist, der liefert oder der nur ein Vergleichsangebot repräsentiert.

### 7.5.5 Ort

Da sich bei der Analyse ergeben hat, dass zu einer Adresse viele Zusatzattribute gehören, wurde dies als einzelne Klasse ausgelagert. Ein Ort hat somit eine Straße, eine Hausnummer und für besondere Fälle einen Adresszusatz. Außerdem soll er einen Titel tragen können. Alles ist vom Typ String. Des Weiteren gehören zu einem Ort eine Stadt und eine Postleitzahl. Da es in manche Länder üblich ist die Provinz/Land/Bundesland mit abzuspeichern, gibt es das Attribut Provinz. Als letztes wird das Land abgespeichert.

### 7.5.6 Lebensmittel

Bei der Analyse wurde festgestellt, dass die Attribute von Essen und Trinken gleich sind. Außerdem sind sie in der realen Welt daraus einer Überkategorie zuteilbar. Daher wurden das Essen und das Trinken in einer Entität, das „Lebensmittel“ vereinigt.

Es hat einen Titel und eine Beschreibung, eine Menge und eine Mengenbeschreibung. Die Menge ist hierbei als double zu sehen, da hier verschiedenste Werte, also auch Gleitkommazahlen stehen können. Um nun zu wissen was die Zahl ausdrücken will, also z.B. Kilogramm, Liter, Gläser, Flaschen existiert geradeeben die Mengenbeschreibung.

Lebensmittel kann in der Kardinalität 0..\* beim Caterer auftreten. Dies wurde aus dem Lastenheft entnommen, da ein Lebensmittel durchaus ohne Caterer existieren kann, aber auch aufgrund der Wiederverwendbarkeit auch mehreren zugewiesen sein kann.

### 7.5.7 Beleg

Ein Beleg stellt eine Rechnung oder eine Quittung der realen Welt dar. Um diesen zu identifizieren hat er einen Titel und eine Beschreibung. Für die Kostenkontrolle speichert der die Kosten inkl. der Währung. Außerdem referenziert er beliebig viele aber mindestens ein (1..\*) Medium.

### 7.5.8 Hilfsmittel

Ein Hilfsmittel hat ebenfalls einen Titel und eine Beschreibung, beide vom Typ String. Außerdem ist ein Hilfsmittel von einer Art. Diese ist ebenfalls ein String. Beliebige viele (also 0..\*) Belege lassen sich dem Hilfsmittel zuordnen.

### 7.5.9 Medium

Ein Medium hat einen Titel, einen Typ und eine URI, die den Pfad zu der repräsentierten Datei darstellt. Alle sind vom Typ String.

### 7.5.10 Aktionshilfsmittel

Diese Klasse ist nach dem Koordinator-Pattern entstanden. Da eine Aktion mehrere Hilfsmittel mit einem Attribut nämlich der Menge speichern soll, muss dies in eine separate Klasse ausgegliedert werden. Das Aktionshilfsmittel hat somit eine Referenz auf die Aktion und das Hilfsmittel gespeichert. Außerdem die Menge des jeweiligen Hilfsmittels, welches die Aktion verwendet. Somit ist eine Komposition zu der Aktion als auch zu dem Hilfsmittel gegeben, denn wenn eines der beiden Entitäten nicht mehr existiert, macht die Relation keinen Sinn mehr. Somit ist die Instanz der Klasse existenzabhängig von den beiden Entitäten.

## 8 Sequenzdiagramm

### 8.1 Einleitung:

In der Hochzeitsverwaltungssoftware will ein Unterhaltungsmanager eine neue Aktion anlegen.

### 8.2 Vorbedingungen

Es wird ein leeres, aber angelegtes und funktionierendes mit Administrator versehen System angenommen. Da keine Benutzerverwaltung modelliert werden muss, wird davon ausgegangen, dass ein Unterhaltungsmanager, ein Hochzeitsmanager existieren und eingeloggt sind.

### 8.3 Szenario

Der Hochzeitsmanager erstellt zunächst eine neue Hochzeit. Dafür zeigt das UI einen Erstellungsdialog, der direkt auf eine neue Instanz der Klasse „Hochzeitsveranstaltung“ gemappt ist. Der Nutzer gibt nun nacheinander die Attribute ein: Titel, Motto, Brautpaar und den Hochzeitsmanager. Danach wird der Datenbank dass die Instanz der Hochzeitsveranstaltung übergeben. Diese erstellt diese Hochzeit dann.

Der Unterhaltungsmanager lässt sich nun zuerst alle Hochzeiten anzeigen. Dafür schickt er das UI eine entsprechende Methode in der Datenbank auf. Diese Hochzeiten werden dann dem Unterhaltungsmanager dargestellt. Danach sucht sich der Unterhaltungsmanager eine aus. Die Details werden wieder per Methodenaufruf geladen.

Er, also der Unterhaltungsmanager erstellt nun innerhalb einer Dialog eine neue Aktion. Er gibt alle Attribute ein, danach wird wieder die Datenbank aufgerufen, die Aktion zu speichern. Die Datenbank gibt einen boolschen Wert zurück, der aussagt ob es sich um eine doppelte Aktion handelt oder nicht. Sondern ja, wird dem Nutzer eine Nachricht angezeigt, die ihm dies mitteilt. Wenn nicht, dann bekommt er eine Erfolgsmeldung.

### 8.4 Pseudocode

#### 8.4.1 Anmerkungen

Es gibt zwei globale Klassen UI und Datenbank, die das UI und die Datenbank repräsentieren. Außerdem soll noch auf die besondere Schreibweise hingewiesen werden, dass wenn nach einem Methodenaufruf etwas in geschweiften Klammern steht, dies den Inhalt der Methode darstellen soll.

#### 8.4.2 Hochzeitsveranstaltung anlegen

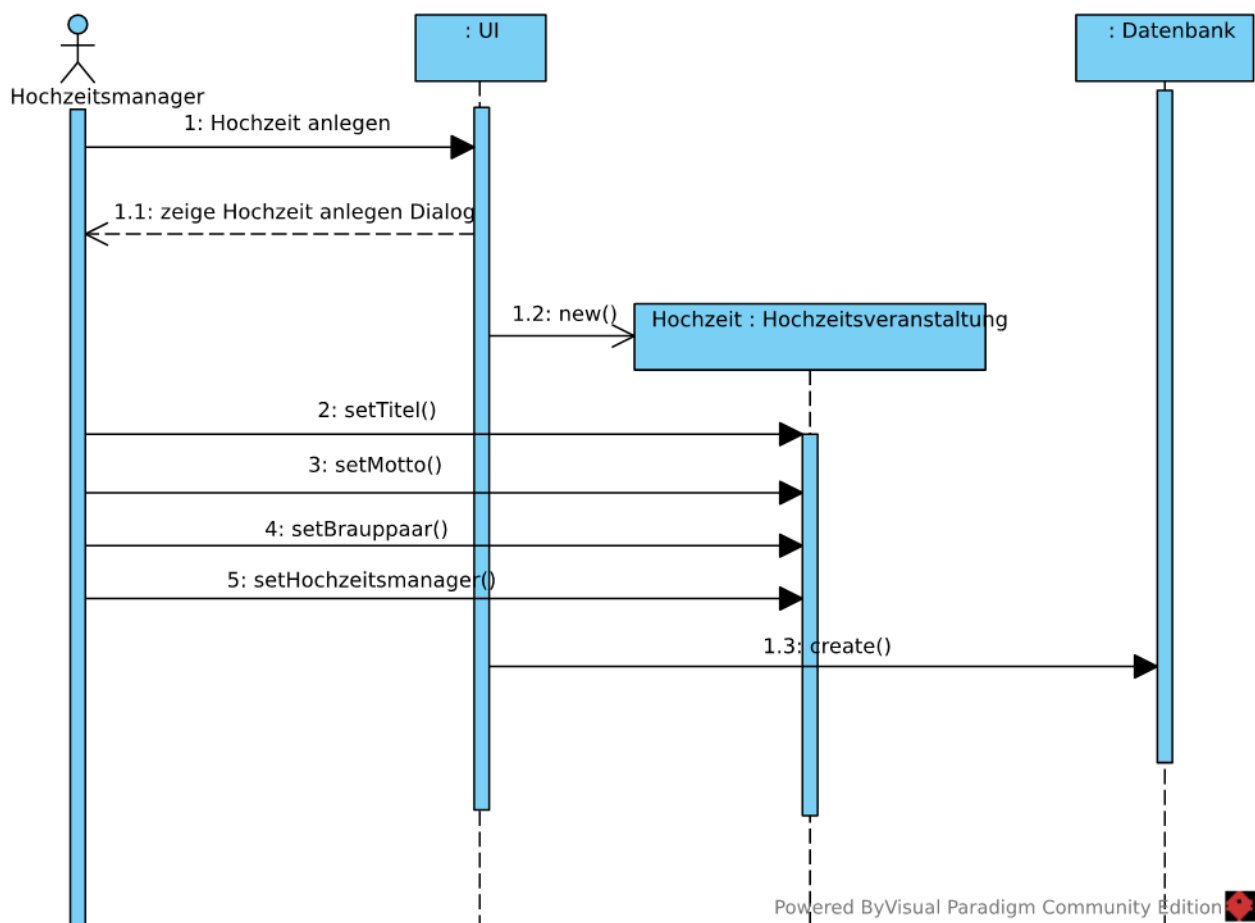
```
1  Hochzeitsveranstaltung hochzeit = UI.showHochzeitAnlegen();  
2  hochzeit.setTitel();  
3  hochzeit.setMotto();  
4  hochzeit.setBrautpaar();  
5  hochzeit.setHochzeitsmanager();  
6  
7  Datenbank.create(hochzeit);
```

### 8.4.3 Unterhaltungsbeitrag anlegen

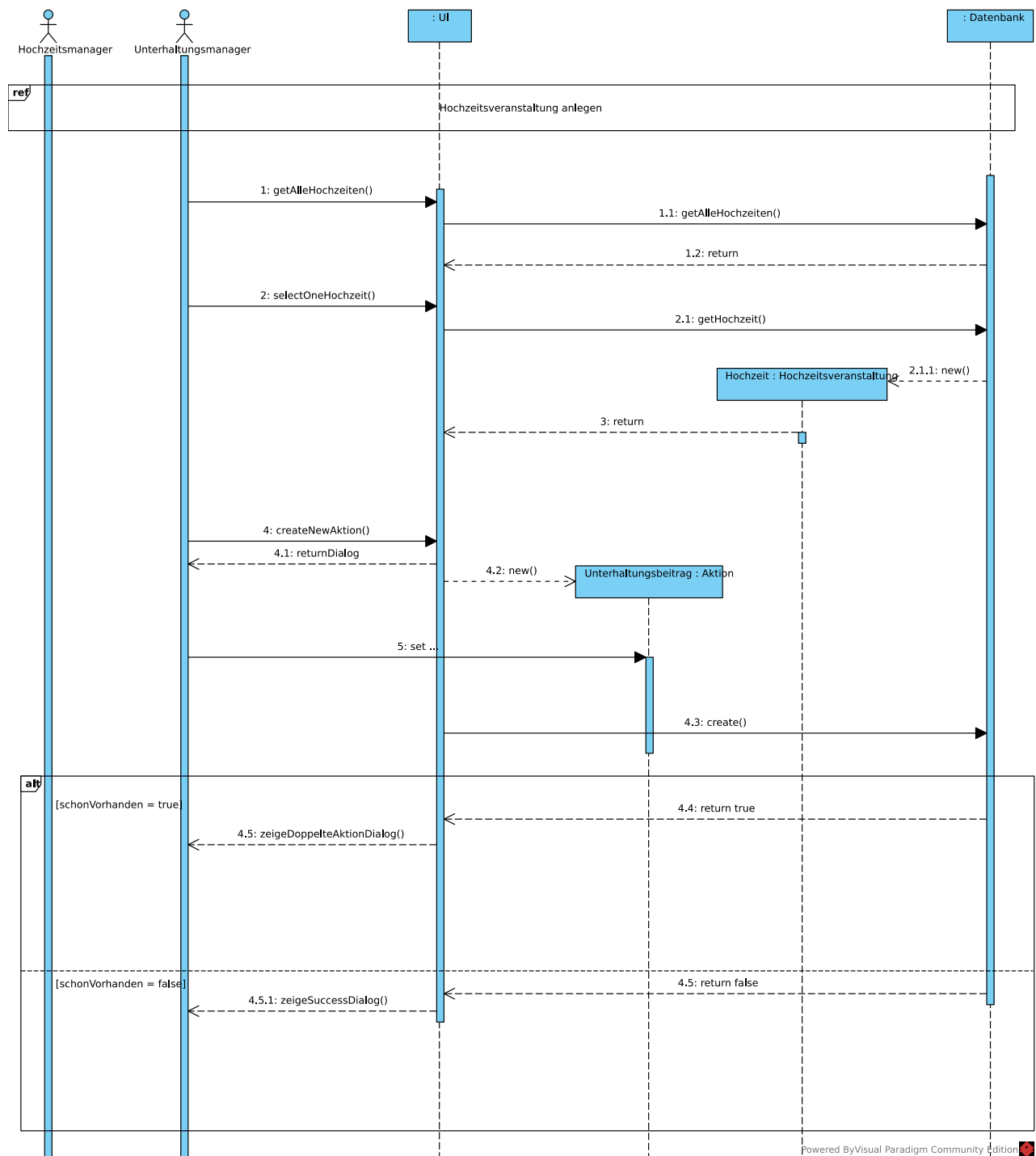
```

1  UI.zeigeAlleHochzeiten() {
2      Datenbank.getAllHochzeiten();
3  }
4
5  Hochzeitsveranstaltung hochzeit = UI.zeigeHochzeit(27) {
6      Datenbank.getHochzeit(27);
7  }
8
9  Aktion aktion = UI.showCreateAktion();
10 aktion.set... //setze alle Attribute
11
12 boolean success = ui.save() {
13     return Datenbank.create(aktion);
14 }
15
16 if(success == true) {
17     ui.showAllesInOrdnung();
18 } else {
19     ui.showDoppelteAktion();
20 }
  
```

### 8.5 Diagramm Hochzeitsveranstaltung anlegen



## 8.6 Diagramm Unterhaltungsbeitrag anlegen

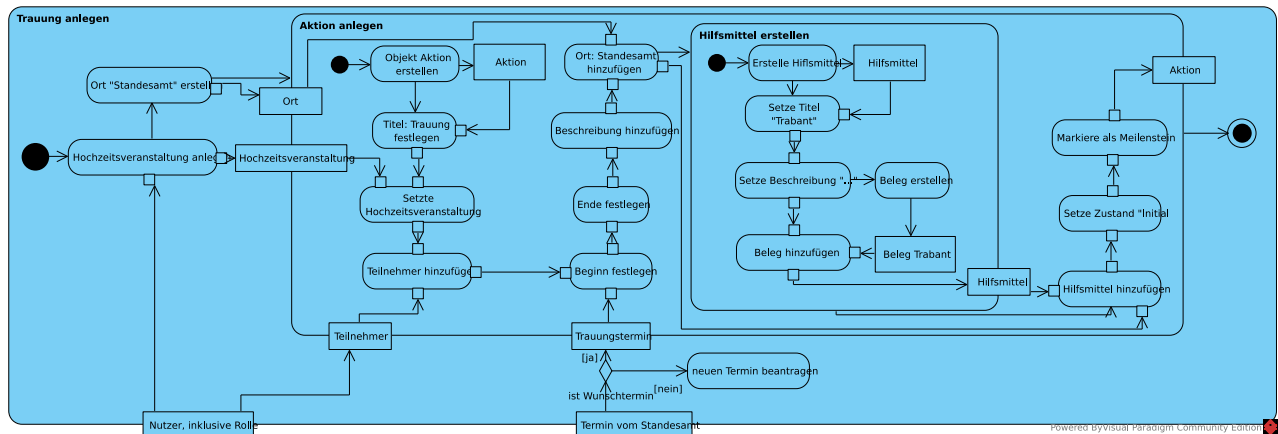


## 9 Aktivitätsdiagramm

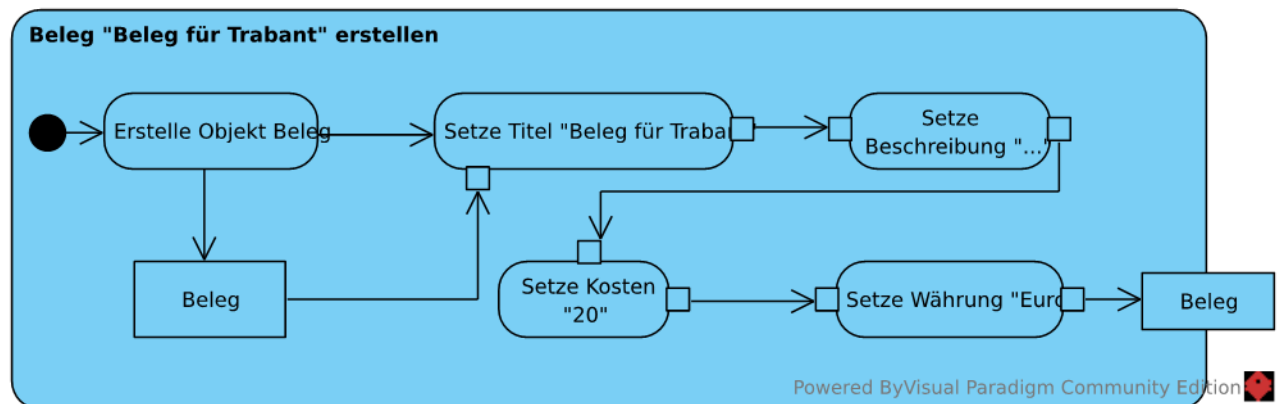
## 9.1 Einleitung

Bei einem Aktivitätsdiagramm handelt es sich um ein Verhaltensdiagramm der Unified Modeling Language(UML). Es beschreibt den Ablauf eines Anwendungsfalls und es lassen sich grundsätzlich alle Aktivitäten innerhalb eines Systemes modellieren.

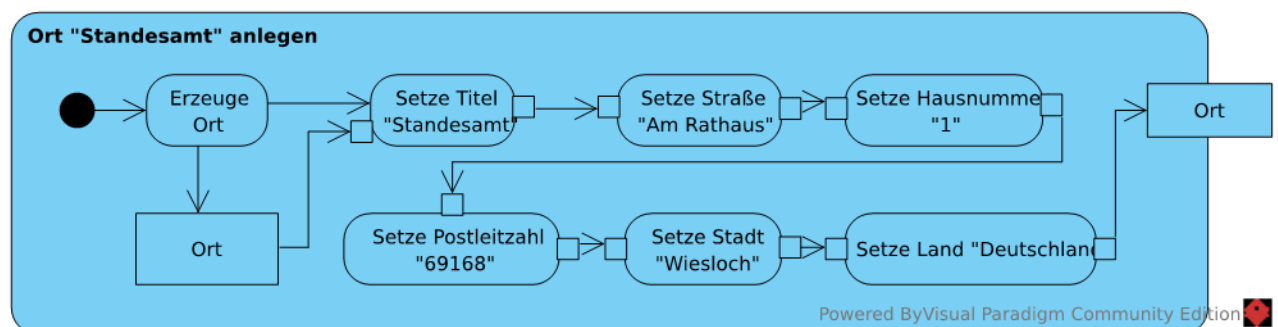
## 9.2 Diagramm Trauung anlegen



### 9.3 Diagramm Beleg für Trabat erstellen

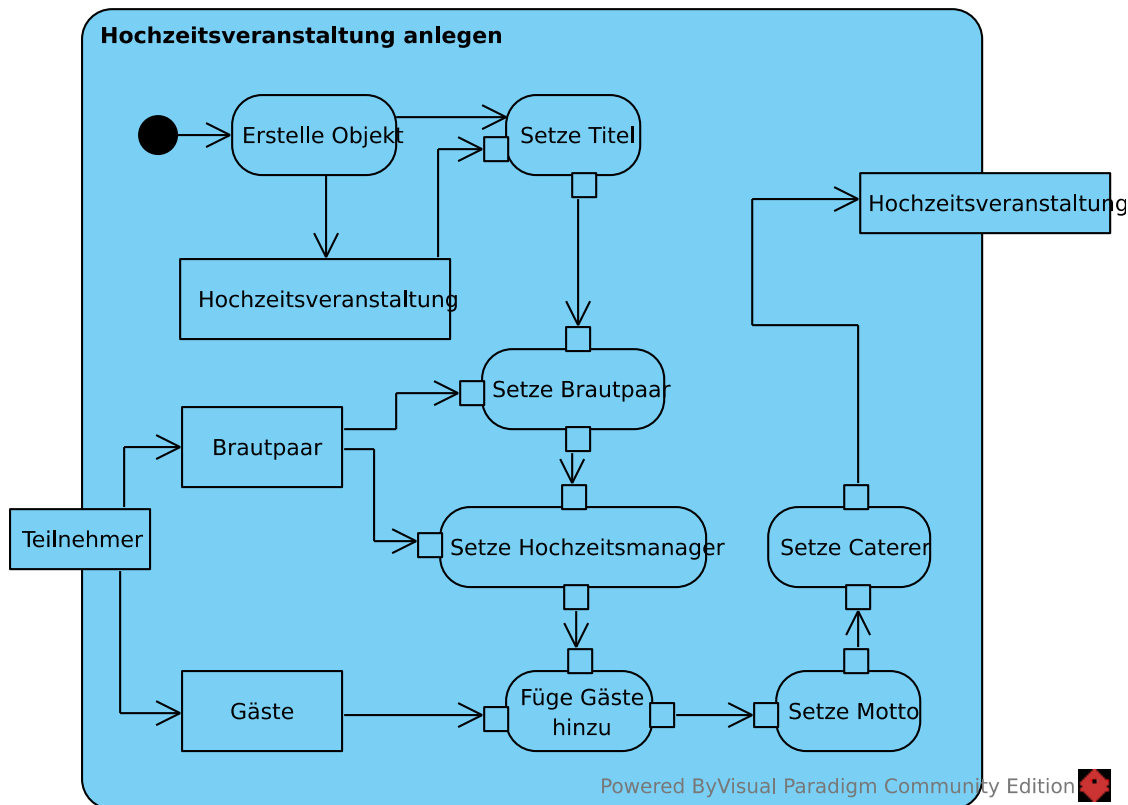


## 9.4 Diagramm Ort Standesamt anlegen





## 9.5 Diagramm Hochzeitsveranstaltung anlegen



## 9.6 Grundlegendes

In diesem Aktivitätsdiagramm wird dargestellt wie die Aktivität „Trauung anlegen“ verlaufen könnte, dies ist allerdings stark abhängig von der jeweiligen Nutzung des Benutzer sowie des Systemstatus. In diesem Diagramm wird angenommen, dass die Hochzeitsveranstaltung noch nicht angelegt ist und die Aktion somit das erste ist, was nach der Veranstaltung angelegt wird. Weiterhin ist vorausgesetzt, dass die Systemnutzer für alle relevanten Benutzer bereits vorhanden sind. Der agierende Akteur ist in diesem Diagramm das Hochzeitspaar, welches die Rolle des Hochzeitsplaner einnimmt.

## 9.7 Hauptdiagramm

Das Hauptdiagramm „Trauung anlegen“ hat 2 Parameter die von außen eingehen, dabei handelt es sich um den Termin vom Standesamt die beide dem Brautpaar vorliegen, da sie zuvor einen Termin beim Standesamt beantragt haben. Der zweite Parameter sind die Systemnutzer, inklusive ihrer Rolle, die für die Hochzeitsveranstaltung relevant sind. Dazu gehören unter anderem: das Brautpaar, die Gäste, sowie Unterhaltungsmanager.

Die erste Aktion, die ausgeführt wird ist das Anlegen der Hochzeitsveranstaltung an sich. Die Aktion erhält einen Objektfluss der Systemnutzer, die für die Hochzeitsveranstaltung relevant sind und hat als Ausgangsparameter die Hochzeitsveranstaltung, die an eine nachfolgende Aktion weitergereicht wird. Der genaue Ablauf dieser Aktivität wird in einem Subdiagramm beschrieben, was extra aufgeführt ist um die Übersichtlichkeit zu wahren. Darauf folgt die Aktion „Ort ‚Standesamt‘ erstellen“, welche ein Objekt Ort erzeugt und dieses ebenfalls an eine nachfolgende Aktion weitergibt. Hierbei handelt es sich auch um eine Aktion für die ein Subdiagramm existiert.

Als nächste Handlung wird eine Entität Aktion mittels „Aktion anlegen“ erzeugt. Diese Aktion erhält in diesem Fall 4 Eingangsparameter: Termin, Teilnehmer, Ort und Hochzeitsveranstaltung, sowie einen Ausgangsparameter: die erzeugte Aktion. Diese Aktion ist im Diagramm verfeinert dargestellt und entspricht somit einem Teil-Diagramm. Nachdem diese Aktion beendet ist, wird die erzeugte Aktion als Ausgangsparameter exportiert und wird. Nach dieser Aktivität folgt lediglich der Endknoten. Damit ist das Ende dieses Aktivitätsdiagramm erreicht und als Ergebnis ist eine Hochzeitsveranstaltung mit einer Aktion „Trauung“ entstanden.

## 9.8 Teildiagramm: Aktion anlegen

Das Subdiagramm hat wie bereits oben erwähnt 4 Eingangsparameter und einen Ausgangsparameter. Es besteht aus einer relativ hohen Anzahl atomarer Aktionen, die sich jeweils mit dem Setzen eines konkreten Attribut befasst. Somit ist die Reihenfolge theoretisch gesehen nicht relevant, da ein Attribut keine Voraussetzung für ein anderes ist. Ausnahme ist hier allerdings Hilfsmittel hinzufügen, das Hilfsmittel muss zuvor erstellt worden sein um es hinzufügen zu können. Als erste Aktion des Teildiagrammes „Objekt Aktion erstellen“ wird ein Objekt erzeugt, welches im weiteren Verlauf von jeder Aktion bearbeitet wird und schließlich dann als Ausgangsparameter auftritt. Zunächst wird die Aktion Titel festlegen ausgeführt: dabei erhält die Entität „Aktion“ einen Wert für das Attribut „Titel“. Daraufhin wird das Attribut „Hochzeitsveranstaltung“ mit der Aktivität „Setze Hochzeitsveranstaltung“ festgelegt. Hierfür wird der korrespondierende Eingangsparameter genutzt. In der nachfolgenden Aktion wird das Attribut „Teilnehmer“ gepflegt, wobei mögliche Werte als Eingangsparameter übergeben wurden. Danach wird durch die Aktionen „Beginn festlegen“ und „Ende festlegen“ der Zeitraum festgelegt. Dafür wird ebenfalls ein Eingangsparameter, welcher den Termin spezifiziert, verwendet. Anschließend wird die Aktion „Beschreibung hinzufügen“ ausgeführt, bei welcher das Attribut Beschreibung einen Wert erhält. Anschließend wird ein Ort spezifiziert, welcher im Vorfeld angelegt wurde und mittels Eingangsparameter übergeben wurde. Als nächstes soll ein Hilfsmittel zu der Aktion hinzugefügt werden, da noch keines in der Datenbank vorhanden ist muss ein eins angelegt werden, dies geschieht mittels der Aktivität „Hilfsmittel erstellen“. Diese erzeugt ein Objekt Aktivität und exportiert jenes als Ausgangsparameter. Hierbei handelt es sich wieder um ein Teildiagramm, welches im entsprechenden Abschnitt beschrieben ist. Nachdem ein Hilfsmittel erstellt ist, kann es mithilfe der Aktion „Hilfsmittel hinzufügen“ dem entsprechenden Attribut als Wert übergeben werden. Schlussendlich werden mittels der Aktionen „Setze Zustand“, sowie „Markiere als Meilenstein“ die Attribute „Zustand“ und „istMeilenstein“ entsprechend gesetzt, bevor das Objekt als Ausgangsparameter zur Verfügung steht.

## 9.9 Teildiagramm Hilfsmittel erstellen

In diesem Teildiagramm wird eine Entität Hilfsmittel erstellt und als Ausgangsparameter zur Verfügung gestellt. Es hat keinen Eingangsparameter und hat somit keine direkten Abhängigkeiten. Zunächst wird durch die Aktion „Erstelle Hilfsmittel“ ein Objekt Hilfsmittel erstellt, welches von den nachfolgenden Aktionen jeweils modifiziert wird und am Ende der Aktion als Ausgangsparameter zur Verfügung steht. Als nächste Aktion folgt „Setze Titel“, welches das Attribut „Titel“ mit dem entsprechenden Wert versieht. Danach wird das Attribut „Beschreibung“ durch die Aktion „Setze Beschreibung“ festgelegt. Anschließend wird die Aktion „Beleg erstellen“ ausgeführt, welche als Subdiagramm existiert. Sie erzeugt ein Objekt „Beleg“ welches für die nachfolgende Aktion „Beleg hinzufügen“ benötigt wird. Letztere fügt den erstellten Beleg dem Hilfsmittel als Wert für das korrespondierende Attribut hinzu und das Teildiagramm ist somit durchlaufen.

## 9.10 Subdiagramm Hochzeitsveranstaltung anlegen

Das Diagramm Hochzeitsveranstaltung anlegen erhält einen Eingangsparameter: die Teilnehmer inklusive Brautpaar und hat einen Ausgangsparameter: die Hochzeitsveranstaltung. Das Diagramm beginnt mit der Aktion Erstelle Objekt, welche wie der Name impliziert ein Objekt vom Typ Hochzeitsveranstaltung erzeugt. Dieses Objekt wird dann an die nächste Aktion weitergeben und genauso verfahren die nachfolgenden Aktionen bis alle Attribute der Entität gesetzt sind. Bei der nächsten Aktion handelt es sich um die Aktivität: „Setze Titel“, welche den Titel der Hochzeitsveranstaltung setzt. Sie erhält als Eingabe das Objekt und gibt es auch entsprechend weiter. Die nachfolgenden zwei Aktionen: „Setze Brautpaar“ und „Setze Hochzeitsmanager“ erhalten beide als Eingabe das Objekt Brautpaar, welches aus dem Eingabeobjekt der Teilnehmer entnommen wird. In diesem Falle ist dementsprechend das Brautpaar Hochzeitsmanager. Als nächstes folgt die Aktion Gäste hinzufügen, welche die entsprechenden Gäste aus dem dem Objekt Teilnehmer entnimmt und als Gäste hinzufügt. Anschließend wird noch mithilfe der Aktion „Setze Motto“ das Motto hinzugefügt und schließlich mit „Setze Caterer“ der Caterer hinzugefügt.

## 9.11 Subdiagramm Ort „Standesamt“ anlegen.

Dieses Diagramm beschreibt den Ablauf des Erstellen eines Objektes „Ort“, im konkreten Fall der Ort „Standesamt“. Es hat keine Eingangsparameter und einen Ausgangsparameter, das erzeugte Objekt Ort mit entsprechenden Attributen. Zu Beginn wird mithilfe der Aktion „Erzeuge Ort“ die Entität Ort erzeugt, welches an dann jeweils an die entsprechenden nachfolgenden Aktionen durchgereicht wird. Die Aktion „Setze Titel“ setzt den Titel des Ortes, sie folgt auf die vorhergehende Aktion und erhält das Objekt „Ort“ als Eingabe. Danach wird durch die Aktion „Setze Straße“ die Straße gesetzt. Anschließend folgen nacheinander die Aktionen „Setze Postleitzahl“, „Setze Stadt“, sowie „Setze Land“ legen jeweils die Attribute Postleitzahl, Stadt und Land der Entität Ort fest. Schließlich gibt die letzte Aktion das Objekt Ort als Ausgangsparameter frei.

## 9.12 Subdiagramm Beleg „Beleg für Trabant“ erstellen

Dieses Diagramm stellt das Aktivitätsdiagramm zur Erzeugung eines Objektes Beleg inklusive dessen Attribute dar. Es hat keinen Eingangsparameter und einen Ausgangsparameter, das erzeugt Objekt Beleg mit den entsprechenden Attributen. Zu Beginn wird mithilfe der Aktion „Erzeuge Beleg“ das Objekt erzeugt, welches an dann jeweils an die entsprechenden nachfolgenden Aktionen durchgereicht wird. Die Aktion „Setze Titel“ setzt den Titel des Beleges, in diesem Fall „Beleg für Trabant“, sie folgt auf die vorhergehende Aktion und erhält das Objekt „Beleg“ als Eingabe. Anschließend setzt die Aktivität „Setze Beschreibung“ das Attribut Beschreibung. Die nachfolgenden Aktion „Setze Kosten“, bestimmt den Wert für die Eigenschaft Kosten. Schließlich sorgt die letzte Aktion „Setze Währung“ dafür, dass der Wert für das korrespondierende Attribut gesetzt wird und das Objekt dann als Ausgangsparameter übergeben wird.

## 9.13 Pseudocode

```
1  Gästeliste Gäste = new Gästeliste(importiere(Gästeliste));
2  Hochzeitsveranstaltung Hochzeit = new HochzeitsVeranstaltung();
3  Hochzeit.setzeTitel(„Hochzeit“);
4  Hochzeit.setzeBrautpaar(Gäste.getBrautpaar());
5  Hochzeit.setzeHochzeitsmanager(Gäste.getBrautpaar());
6  Hochzeit.addGäste(Gäste.getAlleGäste());
7  Hochzeit.setzeMotto(„keins“);
8  Hochzeit.setzeUnterhaltungsmanager(Gäste.getUnterhaltungsmanager());
9  Hochzeit.setzeCaterer(null); //noch keine vorhanden
10 Datum Start = new Datum(importiere(Startdatum));
11 Datum Ende = new Datum(importiere(Enddatum));
12 Ort Standesamt = new Ort();
13 Standesamt.setzeTitel(„Standesamt“);
14 Standesamt.setzteStraße(„Am Rathaus“);
15 Standesamt.setzeHausnummer(„1“);
16 Standesamt.setzePostleitzahl(„69168“);
17 Standesamt.setzeStadt(„Wiesloch“);
18 Standesamt.setzeLand(„DE“);
19 Beleg BelegTrabant = new Beleg();
20 BelegTrabant.setzeTitel(„Beleg für Trabant“);
21 BelegTrabant.setzteBeschreibung(„Das ist der Beleg für das Ausleihen und
    Benzinkosten“);
22 BelegTrabant.setzeKosten(20);
23 BelegTrabant.setzteWährung(„EUR“);
24 Hilfsmittel Trabant = new Hilfsmittel();
25 Trabant.setzteTitel(„Hochzeitsfahrzeug: Trabant“);
26 Trabant.setzteBeschreibung(„Hochzeitsfahrzeug: Trabant 601“);
27 Trabant.addBeleg(BelegTrabant);
28 Aktion Trauung = new Aktion();
29 Trauung.setzeTitel(„Trauung“);
30 Trauung.setzteBeschreibung(„Trauung im Standesamt“);
31 Trauung.setzePriorität(„Hoch“);
32 Trauung.setzeHochzeit(Hochzeit);
33 Trauung.setzeStart(Start);
34 Trauung.setzeEnde(Ende);
35 Trauung.addOrt(Standesamt);
36 Trauung.addHilfsmittel(Trabant);
37 Trauung.addTeilnehmer(Hochzeit.getGäste());
38 Trauung.addOrganisator(Hochzeit.getBrautpaar());
39 Trauung.setzteZustand(„Initial“);
40 Trauung.setzteMeilenstein(True);
```

# Entwurf

## 10 Datenbankentwurf

### 10.1 Voraussetzungen

Um das SQL ausführen zu können ist ein MySQL oder MariaDB Datenbankmanagementsystem (DBMS) von Nöten. Außerdem darf noch keine Datenbank (DB) bzw. Schema mit dem Namen *HOCHZEITSPLANER* existieren, wobei Groß- und Kleinschreibung egal ist.

### 10.2 Kurzeinführung SQL

- **CREATE DATABASE:** Erstellt eine neue Datenbank/Schema
- **USE:** Benutze die angegebene Schema
- **CREATE TABLE:** Erstelle eine Tabelle mit den angegebenen Attributen
- **INT, TIME, VARCHAR ...:** Datentypen
- **NOT NULL:** Der Attributwert muss explizit gesetzt werden, darf also nie NULL sein
- **UNIQUE:** Der Attributwert darf innerhalb der Spalte nur einmal vorkommen
- **UNSIGNED:** Bei Datentypen mit Zahlenwerten können keine negativen Zahlen eingetragen werden. Dafür erhöht sich der positive Zahlenraum um das Doppelte.
- **AUTO\_INCREMENT:** Der Wert dieses Attributes startet stadtmäßig bei 1 und zählt dann nach oben. Man muss sich dadurch nicht selbst darum kümmern, sondern kann dies dem DBMS überlassen
- **DEFAULT:** Standardwert des Attributs
- **PRIMARY KEY:** Attributmenge, die den Primärschlüssel der Tabelle bilden
- **FOREIGN KEY:** Fremdschlüsselbeziehung; Man gibt das Attribut an auf die der Fremdschlüssel zutrifft
  - **REFERENCES:** Der Fremdschlüssel mit Tabelle und Attribut
  - **ON UPDATE:** Aktion, die ausgeführt werden soll, wenn der Fremdschlüssel sich ändert
  - **ON DELETE:** Aktion, die ausgeführt werden soll, wenn der Fremdschlüssel gelöscht wird
  - **CASCADE:** Ändere/Lösche die Fremdschlüsselbeziehung (beim Löschen wird somit der ganze Datensatz gelöscht, also die korrespondierenden Zeilen)
  - **RESTRICT:** Das Ändern/Löschen wird sowohl in der Tabelle mit dem Fremdschlüssel also auch beim Fremdschlüssel verhindert.

## 10.3 SQL

```
1  -- Datenbank erzeugen
2  CREATE DATABASE HOCHZEITSPLANER;
3
4
5  -- Benutze für alles folgende die gerade eben erzeugte Datenbank
6  USE HOCHZEITSPLANER;
7
8
9  -- Erzeuge Entity-Hilfstabellen
10
11 CREATE TABLE AktionsZustaende (
12     aktionsZustandID INT UNSIGNED NOT NULL UNIQUE AUTO_INCREMENT,
13     beschreibung VARCHAR(250) NOT NULL,
14     PRIMARY KEY ( aktionsZustandID )
15 );
16
17 CREATE TABLE AktionsArten (
18     aktionsArtID INT UNSIGNED NOT NULL UNIQUE AUTO_INCREMENT,
19     beschreibung VARCHAR(250) NOT NULL,
20     PRIMARY KEY ( aktionsArtID )
21 );
22
23 CREATE TABLE HilfsmittelArten (
24     hilfsmittelArtID INT UNSIGNED NOT NULL UNIQUE AUTO_INCREMENT,
25     beschreibung VARCHAR(250) NOT NULL,
26     PRIMARY KEY ( hilfsmittelArtID )
27 );
28
29 CREATE TABLE Telefonnummern (
30     telefonnummerID INT UNSIGNED UNIQUE NOT NULL AUTO_INCREMENT,
31     telefonnummer VARCHAR(100) NOT NULL,
32     PRIMARY KEY ( telefonnummerID )
33 );
34
35 CREATE TABLE EmailAdressen (
36     emailAdresseID INT UNSIGNED UNIQUE NOT NULL AUTO_INCREMENT,
37     emailAdresse VARCHAR(100) NOT NULL,
38     PRIMARY KEY ( emailAdresseID )
39 );
40
41
42 -- Erzeuge Entity-Tabellen
43
44 CREATE TABLE Personen (
45     personID INT UNSIGNED UNIQUE NOT NULL AUTO_INCREMENT,
46     name VARCHAR(250) NOT NULL,
47     adresse VARCHAR(250) NOT NULL,
48     istDienstleister BOOL NOT NULL DEFAULT FALSE,
49     PRIMARY KEY ( personID )
50 );
51
52 CREATE TABLE Hochzeitsveranstaltungen (
53     hochzeitsID INT UNSIGNED UNIQUE NOT NULL AUTO_INCREMENT,
54     titel VARCHAR(250) NOT NULL,
55     motto VARCHAR(250),
56     hochzeitsmanagerID INT UNSIGNED NOT NULL,
57     hochzeitspaarID INT UNSIGNED NOT NULL,
```

```

58     PRIMARY KEY ( hochzeitsID ),
59     FOREIGN KEY ( hochzeitsmanagerID ) REFERENCES Personen(personID)
60         ON UPDATE CASCADE
61         ON DELETE RESTRICT,
62     FOREIGN KEY ( hochzeitspaarID ) REFERENCES Personen(personID)
63         ON UPDATE CASCADE
64         ON DELETE RESTRICT
65 );
66
67 CREATE TABLE Caterer (
68     catererID INT UNSIGNED UNIQUE NOT NULL AUTO_INCREMENT,
69     beschreibung VARCHAR(255),
70     kontaktPerson INT UNSIGNED NOT NULL,
71     zumVergleich BOOL NOT NULL DEFAULT FALSE,
72     PRIMARY KEY ( catererID ),
73     FOREIGN KEY ( kontaktPerson ) REFERENCES Personen(personID)
74         ON UPDATE CASCADE
75         ON DELETE RESTRICT
76 );
77
78 CREATE TABLE Aktionen (
79     aktionID INT UNSIGNED UNIQUE NOT NULL AUTO_INCREMENT,
80     titel VARCHAR(250) NOT NULL,
81     beschreibung VARCHAR(250) NOT NULL,
82     hochzeitsID INT UNSIGNED NOT NULL,
83     notiz VARCHAR(250),
84     prioritaet TINYINT NOT NULL,
85     anfang DATETIME NOT NULL,
86     ende DATETIME NOT NULL,
87     aktionsArt INT UNSIGNED NOT NULL,
88     versteckt BOOL NOT NULL DEFAULT true,
89     aktionsZustand INT UNSIGNED NOT NULL,
90     meilenstein BOOL NOT NULL DEFAULT false,
91     PRIMARY KEY ( aktionID ),
92     FOREIGN KEY ( aktionsZustand ) REFERENCES
    AktionsZustaende(aktionsZustandID)
93         ON UPDATE CASCADE
94         ON DELETE RESTRICT,
95     FOREIGN KEY ( aktionsArt ) REFERENCES AktionsArten(aktionsArtID)
96         ON UPDATE CASCADE
97         ON DELETE RESTRICT,
98     FOREIGN KEY ( hochzeitsID ) REFERENCES
    Hochzeitsveranstaltungen(hochzeitsID)
99         ON UPDATE CASCADE
100        ON DELETE CASCADE
101);
102
103CREATE TABLE Nahrungsmittel (
104    nahrungsmittelID INT UNSIGNED UNIQUE NOT NULL AUTO_INCREMENT,
105    beschreibung VARCHAR (250),
106    menge DOUBLE UNSIGNED NOT NULL,
107    mengenbeschreibung VARCHAR(50) NOT NULL,
108    PRIMARY KEY ( nahrungsmittelID )
109);
110
111CREATE TABLE Medien(
112    mediumID INT UNSIGNED UNIQUE NOT NULL AUTO_INCREMENT,
113    uri VARCHAR(250) NOT NULL UNIQUE,
114    title VARCHAR(250) NOT NULL,

```



```

115     PRIMARY KEY ( mediumID )
116);
117
118CREATE TABLE Belege(
119     belegID INT UNSIGNED UNIQUE NOT NULL AUTO_INCREMENT,
120     kosten DOUBLE UNSIGNED NOT NULL,
121     währung CHAR(3) NOT NULL,
122     title VARCHAR(250) NOT NULL,
123     beschreibung VARCHAR(250),
124     private BOOL NOT NULL,
125     PRIMARY KEY ( belegID )
126);
127
128CREATE TABLE Hilfsmittel (
129     hilfsmittelID INT UNSIGNED UNIQUE NOT NULL AUTO_INCREMENT,
130     title VARCHAR(250) NOT NULL,
131     beschreibung VARCHAR(250),
132     hilfsmittelArt INT UNSIGNED NOT NULL,
133     PRIMARY KEY ( hilfsmittelID ),
134     FOREIGN KEY ( hilfsmittelArt ) REFERENCES
        HilfsmittelArten(hilfsmittelArtID)
135         ON UPDATE CASCADE
136         ON DELETE RESTRICT
137);
138
139CREATE TABLE Orte (
140     ortID INT UNSIGNED NOT NULL UNIQUE AUTO_INCREMENT,
141     titel VARCHAR(250),
142     strasse VARCHAR(250) NOT NULL,
143     hausnummer VARCHAR(10) NOT NULL,
144     adressZusatz VARCHAR(250),
145     stadt VARCHAR(250),
146     postleitzahl VARCHAR(50) NOT NULL,
147     provinz VARCHAR(250) NOT NULL,
148     land CHAR(3) NOT NULL,
149     PRIMARY KEY ( ortID )
150);
151
152
153-- Erzeuge Relationen
154
155CREATE TABLE Gaeste (
156     hochzeitsID INT UNSIGNED NOT NULL,
157     personID INT UNSIGNED NOT NULL,
158     PRIMARY KEY ( hochzeitsID, personID ),
159     FOREIGN KEY ( hochzeitsID ) REFERENCES
        Hochzeitsveranstaltungen(hochzeitsID)
160         ON UPDATE CASCADE
161         ON DELETE CASCADE,
162     FOREIGN KEY ( personID ) REFERENCES Personen(personID)
163         ON UPDATE CASCADE
164         ON DELETE CASCADE
165);
166
167CREATE TABLE HochzeitsCaterer (
168     hochzeitsID INT UNSIGNED,
169     catererID INT UNSIGNED,
170     PRIMARY KEY ( hochzeitsID, catererID ),

```



```

171     FOREIGN KEY ( hochzeitsID ) REFERENCES
      Hochzeitsveranstaltungen(hochzeitsID)
172         ON UPDATE CASCADE
173         ON DELETE CASCADE,
174     FOREIGN KEY ( catererID ) REFERENCES Caterer(catererID)
175         ON UPDATE CASCADE
176         ON DELETE CASCADE
177);
178
179CREATE TABLE HochzeitsUnterhaltungsmanager (
180     hochzeitsID INT UNSIGNED,
181     unterhaltungsmanagerID INT UNSIGNED,
182     PRIMARY KEY ( hochzeitsID, unterhaltungsmanagerID ),
183     FOREIGN KEY ( hochzeitsID ) REFERENCES
      Hochzeitsveranstaltungen(hochzeitsID)
184         ON UPDATE CASCADE
185         ON DELETE CASCADE,
186     FOREIGN KEY ( unterhaltungsmanagerID ) REFERENCES Personen(personID)
187         ON UPDATE CASCADE
188         ON DELETE CASCADE
189);
190
191CREATE TABLE AktionsVerantwortliche (
192     aktionID INT UNSIGNED NOT NULL,
193     personID INT UNSIGNED NOT NULL,
194     PRIMARY KEY ( aktionID, personID ),
195     FOREIGN KEY ( aktionID ) REFERENCES Aktionen(aktionID)
196         ON UPDATE CASCADE
197         ON DELETE CASCADE,
198     FOREIGN KEY ( personID ) REFERENCES Personen(personID)
199         ON UPDATE CASCADE
200         ON DELETE CASCADE
201);
202
203CREATE TABLE AktionsTeilnehmer (
204     aktionID INT UNSIGNED NOT NULL,
205     personID INT UNSIGNED NOT NULL,
206     PRIMARY KEY ( aktionID, personID ),
207     FOREIGN KEY ( aktionID ) REFERENCES Aktionen(aktionID)
208         ON UPDATE CASCADE
209         ON DELETE CASCADE,
210     FOREIGN KEY ( personID ) REFERENCES Personen(personID)
211         ON UPDATE CASCADE
212         ON DELETE CASCADE
213);
214
215CREATE TABLE CatererEssen (
216     catererID INT UNSIGNED NOT NULL,
217     essenID INT UNSIGNED NOT NULL,
218     PRIMARY KEY ( catererID, essenID ),
219     FOREIGN KEY ( catererID ) REFERENCES Caterer(catererID)
220         ON UPDATE CASCADE
221         ON DELETE CASCADE,
222     FOREIGN KEY ( essenID ) REFERENCES Nahrungsmittel(nahrungsmittelID)
223         ON UPDATE CASCADE
224         ON DELETE CASCADE
225);
226
227CREATE TABLE CatererTrinken (

```

```
228     catererID INT UNSIGNED NOT NULL,
229     trinkenID INT UNSIGNED NOT NULL,
230     PRIMARY KEY ( catererID, trinkenID ),
231     FOREIGN KEY ( catererID ) REFERENCES Caterer(catererID)
232         ON UPDATE CASCADE
233         ON DELETE CASCADE,
234     FOREIGN KEY ( trinkenID ) REFERENCES Nahrungsmittel(nahrungsmittelID)
235         ON UPDATE CASCADE
236         ON DELETE CASCADE
237);
238
239CREATE TABLE AktionsOrte (
240     aktionID INT UNSIGNED NOT NULL,
241     ortID INT UNSIGNED NOT NULL,
242     PRIMARY KEY ( aktionID, ortID ),
243     FOREIGN KEY ( aktionID ) REFERENCES Aktionen(aktionID)
244         ON UPDATE CASCADE
245         ON DELETE CASCADE,
246     FOREIGN KEY ( ortID ) REFERENCES Orte(ortID)
247         ON UPDATE CASCADE
248         ON DELETE CASCADE
249);
250
251CREATE TABLE AktionsMedien (
252     aktionID INT UNSIGNED NOT NULL,
253     mediumID INT UNSIGNED NOT NULL,
254     PRIMARY KEY ( aktionID, mediumID ),
255     FOREIGN KEY ( aktionID ) REFERENCES Aktionen(aktionID)
256         ON UPDATE CASCADE
257         ON DELETE CASCADE,
258     FOREIGN KEY ( mediumID ) REFERENCES Medien(mediumID)
259         ON UPDATE CASCADE
260         ON DELETE CASCADE
261);
262
263CREATE TABLE MedienBelege (
264     mediumID INT UNSIGNED NOT NULL,
265     belegID INT UNSIGNED NOT NULL,
266     PRIMARY KEY ( mediumID, belegID ),
267     FOREIGN KEY ( mediumID ) REFERENCES Medien(mediumID)
268         ON UPDATE CASCADE
269         ON DELETE CASCADE,
270     FOREIGN KEY ( belegID ) REFERENCES Belege(belegID)
271         ON UPDATE CASCADE
272         ON DELETE CASCADE
273);
274
275CREATE TABLE AktionsBelege (
276     aktionID INT UNSIGNED NOT NULL,
277     belegID INT UNSIGNED NOT NULL,
278     PRIMARY KEY ( aktionID, belegID ),
279     FOREIGN KEY ( aktionID ) REFERENCES Aktionen(aktionID)
280         ON UPDATE CASCADE
281         ON DELETE CASCADE,
282     FOREIGN KEY ( belegID ) REFERENCES Belege(belegID)
283         ON UPDATE CASCADE
284         ON DELETE CASCADE
285);
286
```

```

287CREATE TABLE AktionsHilfsmittel (
288    aktionID INT UNSIGNED NOT NULL,
289    hilfsmittelID INT UNSIGNED NOT NULL,
290    anzahl INT UNSIGNED NOT NULL DEFAULT 1,
291    PRIMARY KEY ( aktionID, hilfsmittelID ),
292    FOREIGN KEY ( aktionID ) REFERENCES Aktionen(aktionID)
293        ON UPDATE CASCADE
294        ON DELETE CASCADE,
295    FOREIGN KEY ( hilfsmittelID ) REFERENCES Hilfsmittel(hilfsmittelID)
296        ON UPDATE CASCADE
297        ON DELETE CASCADE
298);
299
300CREATE TABLE HilfsmittelBelege (
301    hilfsmittelID INT UNSIGNED NOT NULL,
302    belegID INT UNSIGNED NOT NULL,
303    PRIMARY KEY ( hilfsmittelID, belegID ),
304    FOREIGN KEY ( hilfsmittelID ) REFERENCES Hilfsmittel(hilfsmittelID)
305        ON UPDATE CASCADE
306        ON DELETE CASCADE,
307    FOREIGN KEY ( belegID ) REFERENCES Belege(belegID)
308        ON UPDATE CASCADE
309        ON DELETE CASCADE
310);
311
312CREATE TABLE PersonenMailadressen (
313    personID INT UNSIGNED NOT NULL,
314    emailAdresseID INT UNSIGNED NOT NULL,
315    PRIMARY KEY ( personID, emailAdresseID ),
316    FOREIGN KEY ( personID ) REFERENCES Personen(personID)
317        ON UPDATE CASCADE
318        ON DELETE CASCADE,
319    FOREIGN KEY ( emailAdresseID ) REFERENCES Emailadressen(emailAdresseID)
320        ON UPDATE CASCADE
321        ON DELETE CASCADE
322);
323
324CREATE TABLE PersonenTelefonnummern (
325    personID INT UNSIGNED NOT NULL,
326    telefonnummerID INT UNSIGNED NOT NULL,
327    PRIMARY KEY ( personID, telefonnummerID ),
328    FOREIGN KEY ( personID ) REFERENCES Personen(personID)
329        ON UPDATE CASCADE
330        ON DELETE CASCADE,
331    FOREIGN KEY ( telefonnummerID ) REFERENCES
    Telefonnummern(telefonnummerID)
332        ON UPDATE CASCADE
333        ON DELETE CASCADE
334);
335
336
337-- Erzeuge Programm-Tabellen
338
339CREATE TABLE NichtInformierteNutzer(
340    personID INT UNSIGNED NOT NULL,
341    aktionID INT UNSIGNED NOT NULL,
342    PRIMARY KEY ( personID, aktionID ),
343    FOREIGN KEY ( personID ) REFERENCES Personen(personID)
344        ON UPDATE CASCADE
  
```

```
345          ON DELETE CASCADE,  
346      FOREIGN KEY ( aktionID ) REFERENCES Aktionen(aktionID)  
347          ON UPDATE CASCADE  
348          ON DELETE CASCADE  
349 );
```

## 10.4 Erstellen des Schemas

Als erstes muss ein Schema, oder wie auch manchmal genannt eine Datenbank erstellt werden. In dieser werden dann die Tabellen erstellt.

## 10.5 Setzen des aktuellen Schemas

Mit dem USE Befehl kann man erreichen dass alle nachfolgenden Befehle auf diesem Schema ausgeführt werden. Dadurch muss bei der Tabellenerstellung nicht immer das Schema mit angegeben werden.

## 10.6 Erzeugen der Tabellen mit Fremdschlüsselbeziehungen

Da hier die Tabellen direkt mit den Fremdschlüsselbeziehungen erstellt werden, müssen die Tabellen in einer logischen Reihenfolge erstellt werden, sodass jede Tabelle die ein Fremdschlüssel sein wird schon erzeugt worden ist.

## 10.7 Bekannte Entitäten

Zu den bekannten und schon beschrieben Entitäten gehören:

- Hochzeitsveranstaltung
- Caterer
- Aktion
- Nahrungsmittel
- Medium
- Beleg
- Ort

Die Tabellennamen sind im Plural des Entitätsnamen gehalten.

Diese Entitäten haben alle ein neues Attribut – eine ID. Diese ID wird automatisch mit dem Einfügen einer neuen Entität in die Tabelle inkrementiert. Diese ID bildet auch den Primärschlüssel.

Alle Attribute die optional sind erlauben NULL-Werte, alle andere verlangen explizit Werte.

Sofern einfache Referenzen, d.h. Kardinalitäten mit 0/1 je Seite vorhanden waren wird direkt der Fremdschlüssel erzeugt. Das Updaten der Fremdschlüsselbeziehung ist per se überall erlaubt. Beim Löschen nur, wenn das dazugehörige Attribute NULL-Werte erlaubt, ansonsten wird es verhindert.

Alle anderen Attribute, die in Listen auftauchen wurden in weitere Tabellen ausgelagert.

## 10.8 Weitere Entitäten

Bei der Analyse des Lastenheftes hat sich ergeben, dass gewisse Attribute eine eigene Tabelle benötigen. Da dies unter anderem aus datenbanktechnische Gründen passiert als auch, dass man hier gewisse Attribute aus logischer Sicht auslagern muss, damit eine Speicherung möglich ist, gibt es damit Entitäten, die zwar in der Datenbank existieren, aber nicht im AKD.

Zu zweiten gehören die Aktionsarten, die Aktionszustände und die Hilfsmittelarten. An sich lassen sich diese als einfach String abbilden, aber da man nicht benutze Zustände und Arten nicht „vergessen“ möchte, braucht man hierzu eigene Tabellen.

Außerdem brauchen die Telefonnummern und die Emailadresse eigene Tabellen, da ansonsten ein unsauberer Primärschlüssel in gewissen Tabellen, die Relationen enthalten entstehen würde.

## 10.9 Relationstabellen

Alle Attribute, die Listen verlangen, brauchen im relationalen Datenbankmodell eine extra Tabelle. Diese Tabellen haben bis auf Ausnahmen nur zwei Attribute, nämlich die ID der eigentlichen Entität und die des „Listeneintrags“. Man kann natürlich auch Relationsattribute, wie z.B. die Menge bei Aktionshilfsmittel mitspeichern. Die Fremdschlüssel verweisen immer auf die Entitätstabelle. Sofern eine Entität gelöscht wird, löscht sich dieser Eintrag gleich mit, da er logischerweise nicht mehr von Nöten ist.

## 10.10 Weitere Besonderheiten

Das Attribut Priorität in der Tabelle Aktionen wurde auf eine spezielle Art umgesetzt. Da es nur eine feste und nicht erweiterbare Anzahl an Prioritäten gibt, muss dafür keine extra Tabelle erzeugt werden. Um nun nicht den Prioritätstext speichern zu müssen kann auch einfach eine Zahl, die diese Priorität repräsentiert gespeichert werden. Die Applikation muss später nur noch zwischen den Zahlen und dem Text konvertieren können.

## 10.11 Programmtabellen

Diese Tabellen speichern keine Entitäten an sich, sondern für die Applikation wichtige Daten.

Die Tabelle „NichtInformierteNutzer“, ist dafür da abzuspeichern, welche Nutzer bei welchen Aktionen keine Informationen bei Aktionen erhalten wollen. Dafür speichert diese Tabelle zu einem die Person wie auch die Aktion, wobei die ID's der beiden den Primärschlüssel bilden und entsprechende Fremdschlüssel ebenfalls vorhanden sind.

# 11 Entwurfsklassendiagramm

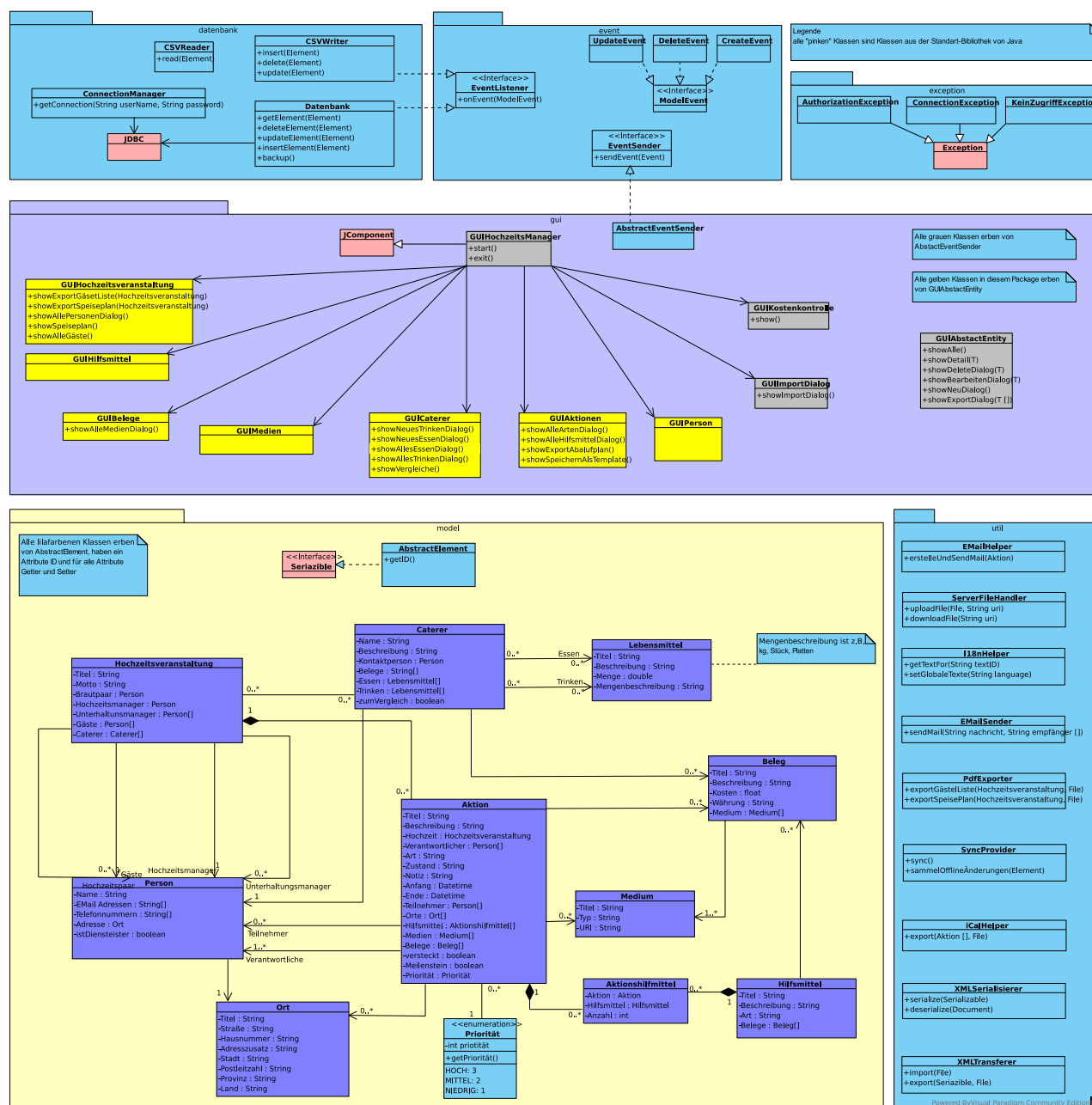
## 11.1 Einleitung

Um eine bessere Übersichtlichkeit zu gewähren wurden bei Parameters sofern der Datentyp (z.B. File) schon genug Beschreibung ist, der Parametername weggelassen. Allerdings bei z.B. „String“ wurde meist ein Name (z.B. URI) verwendet.

Außerdem wurde für eine bessere Lesbarkeit bei Parametern anstatt „AbstractElement“ nur „Element“ verwendet.

Außerdem werden ebenfalls aus Übersichtszwecken alle „throws“-Deklarationen nur der Beschreibung genannt.

## 11.2 Diagramm



## 11.3 Package util

Das Package „util“ enthält alle Hilfsklassen, die von der Applikation gebraucht werden.

### 11.3.1 Klasse l18nHelper

Diese Klasse verwaltet die übersetzten Texte innerhalb der Applikation. Es ist geplant alle Texte, die dem Nutzer gezeigt werden, übersetzbar zu machen. Daher wird für jeden Text eine ID vergeben. Während der Laufzeit, in der die Sprache feststeht und diese dieser Klasse bekannt ist, wird die Methode „getTextFor(String textID)“ aufgerufen mit der Text ID als Parameter und liefert den entsprechenden Text in der gewünschten Sprache als Rückgabewert. Außerdem besitzt diese Klasse noch eine Methode, in der die globale Sprache gesetzt werden kann, also die Sprache in der die Texte von den oben beschriebenen zurückgegeben werden.

### 11.3.2 Klasse EMailSender

Diese Klasse bietet nur eine Methode öffentlich an, mit der eine E-Mail versendet werden kann. Dabei übergibt man den Body, also den darzustellenden Text als String, und als String-Array die E-Mail Adressen der Empfänger. Intern wird dann eine Verbindung zu dem SMTP-Server des Servers hergestellt, der den Mailversand dann übernehmen wird. Sofern keine Internetverbindung besteht, werden die E-Mails lokal zwischengespeichert und gesendet, sobald wieder Internetempfang besteht.

### 11.3.3 Klasse PdfExporter

Diese Klasse ist auf das Exportieren zu pdf-Dateien optimiert. Allerdings ist dies keine allgemeingültige Klasse, sondern ist nur für die beiden Anwendungsfälle in denen ein Export zu pdf-Dateien stattfindet zuständig. Dafür bietet sie zwei Export-Methoden an, einmal für die Gästeliste, und einmal für den Speiseplan. Beide Male wird die derzeitige Hochzeitsveranstaltung als Parameter erwartet, da darüber leicht die entsprechenden Daten geladen werden können. Außerdem kann auch beides Mal ein File mitgegeben werden, der die Datei repräsentiert, die die pdf-Datei werden wird.

### 11.3.4 Klasse SyncProvider

Diese Klasse ist für die Synchronisation mit dem Server zuständig. Normalerweise sollten keine Konflikte bestehen. Allerdings wenn eine Person offline gearbeitet hat, müssen diese Daten mit dem Server synchronisiert werden. Dafür ist die Methode „sync“ zuständig. Sie vergleicht die Änderungen mit der derzeitigen Datenbasis, sucht nach Konflikten und versucht, sofern welche existieren sollten diese zu lösen. Allerdings kann dies unter Umständen manchmal nicht automatisch gelöst werden. Deswegen wird bei einem solchen Konflikt ein Dialog erstellt, welcher dem Nutzer dies mitteilt. Der Nutzer bekommt dann verschiedene Optionen angezeigt, wie er den Konflikt lösen kann.

Die Methode „sammleOfflineÄnderungen“ ist ihrerseits für das Sammeln aller Operationen im Offline-Betrieb zuständig. Sie wird von Datenbankklasse aufgerufen, sofern ein Verbindungsfehler zum Server geschieht.

### 11.3.5 Klasse iCalHelper

Der „iCalHelper“ nimmt in seiner Export-Funktion ein Aktions-Array entgegen, aus dem er dann eine iCal-Datei generiert. Diese Datei entspricht dem Ablaufplan der Aktionen. Außerdem nimmt die Methode noch einen File entgegen, der spezifiziert wo und mit welchem Namen die iCal-Datei auf dem lokalen System abgespeichert werden soll.

### 11.3.6 Klasse XMLSerialisierer

Um die XML-Objekte in Entitäten umzuwandeln und andersherum, bietet diese Klasse genau zwei Methoden an, die genau dies tun. Die Methode „serialize“ nimmt eine Entität, genauer ein Serializable, um dieses in ein XML-Dokument umzuwandeln. Die gegenorientierte Methode nimmt ein „Document“ als Parameter, mit dem die Java XML-Klasse gemeint ist und wandelt dieses in eine Entität um.

### 11.3.7 Klasse XMLTransferer

Diese Klasse fungiert quasi als Wrapper um die Klasse „XMLSerialisierer“. Die import-Funktion erlaubt das importieren von XML-Dokumenten, die danach vom „XMLSerialisierer“ in einen Entität umgewandelt wird. Die Export-Funktion macht genau das Gegenteil, wobei ein weiterer Parameter die Zielfile bestimmt.

### 11.3.8 Klasse EMailHelper

Diese Klasse ist wiederum als Wrapper um die Klasse „EMailSender“ gedacht. Sie hilft die Nachricht zu erstellen und den, beziehungsweise die Empfänger herauszufinden. Hierzu wird der Klasse in ihre einzige Methode eine Aktion übergeben, die geändert werden soll, nachdem die Änderungen an der Klasse vorgenommen worden sind, aber bevor diese in der Datenbank gespeichert wurden. Anhand der Daten sucht sich die Methode die „alten“ Daten aus der Datenbank heraus und kann so eine E-Mail mit den Änderungen erstellen. Mithilfe der Aktion kennt sie auch die Empfänger. Die so nun generierten/herausgesuchten Daten werden dem „EMailSender“ übergeben.

### 11.3.9 Klasse ServerFileHandler

Um Daten auf dem Server hochzuladen und herunterzuladen bietet diese Klasse die passenden Methoden an. Für den Upload wird der passenden Methode eine URI übergeben, die dem Pfad auf dem Server entspricht und ein File, der die hochzuladende Datei ist. Der Download funktioniert ähnlich, wobei in diesem Fall nur die URI für die Datei auf dem Server gebraucht wird. Sollte ein Verbindungsfehler auftauchen (also eine ConnectionException geworfen werden), werden die Daten solange in der Applikation gehalten, bis wieder eine Verbindung aufgebaut werden kann.

## 11.4 Package model

Das Package „model“ enthält das Modell der Applikation, d.h. alle Entitätsklassen und dazugehörigen Enums.

### 11.4.1 Enum Priorität

Diese Enumeration enthält die Zahlenwerte der Prioritäten. Es gibt somit die Attributdomaine für das Attribut Priorität fest. Die Domaine ist fest definiert und kann nicht angepasst werden.



### 11.4.2 Klasse AbstractElement

Um Entitäten direkt Methoden übergeben zu können, ohne in den Methodenköpfen alle Möglichkeiten einer Entität (also z.B. Aktion, Person, etc.) nennen zu müssen ist eine zentrale Oberklasse die beste Lösung. Sie bietet als einzige Methode „getID“ an, die die ID des Elements zurückgibt. Außerdem implementiert sie das Interface „EventSender“. Dadurch dass diese Klasse als Superklasse aller Entitäten dieses Interface implementiert müssen als Unterklassen die Methoden des Interfaces implementieren.

### 11.4.3 Klasse Aktion

Die Klasse Aktion hat eine kleine Änderung erfahren. Ihr Prioritätsattribut ist jetzt vom Typ Priorität, hat also einen Wert vom Enum „Priorität“

### 11.4.4 Hinweis zu den Entity-Hilfs-Klassen

In der Datenbank gibt es mehrere Entity-Hilfs-Klassen (EMail-Adresse, Aktionszustände). Diese muss man allerdings nicht im Model abbilden, sondern kann bei der Abfrage gleich die Relation auflösen.

## 11.5 Package event

Das Package „event“ enthält alle nötigen Klassen und Interfaces um ein Eventhandling innerhalb der Applikation zu ermöglichen.

### 11.5.1 Interface EventListener

Dieses Interface müssen alle Klassen implementieren die innerhalb der Applikation Events empfangen wollen.

### 11.5.2 Interface EventSender

Dieses Interface müssen alle Klassen implementieren die innerhalb der Applikation Events senden wollen. Als Methode wird das Senden des Events angeboten, wobei ein beliebiges Event übergeben werden kann.

### 11.5.3 Interface ModelEvent

Um wieder ein zentrales Element zu haben, müssen alle Events dieses Interface implementieren. Außerdem sorgt es dafür, dass über eine einheitliche Schnittstelle auf die Event-Daten zugegriffen werden kann.

### 11.5.4 Event-Klassen

Es gibt drei unterschiedliche Event-Arten. Jedes davon hat seine eigenen Klassen, die die Methoden des Interfaces „ModelEvent“ implementiert. Es handelt sich um die Klassen „CreateEvent“, „DeleteEvent“ und „UpdateEvent“, wobei die Namen selbsterklärend sind.

## 11.6 Package exception

Alle Exceptions, die innerhalb der Applikation geworfen werden, sind in diesem Package gesammelt.

### 11.6.1 Exception-Klassen

Alle Exception-Klassen haben als Superklasse die Java-Klasse „Exception“. Es gibt die „KeinZugriffException“, die geworfen wird, wenn man keinen Zugriff auf ein Element hat. „ConnectionException“, wird geworfen, wenn zum Server keine Verbindung aufgebaut werden kann und die „AuthorizationException“ taucht auf, wenn der Login fehlschlägt.

## 11.7 Package datenbank

Das Package Datenbank enthält alle Komponenten, die für das Zusammenspiel mit der Datenbank relevant. Angefangen bei dem bloßen Verbindungsmanagement bis hin zu lokalen Caches. Somit ist es eines der grundlegendsten Bestandteile der Software.

### 11.7.1 Klasse Datenbank

Diese Klasse ist für das Schreiben und Lesen in/aus der Datenbank verantwortlich. Um die Entsprechenden Daten zu bekommen, implementiert sie das Interface „EventListener“. Sie bietet für das Lesen (get), Löschen (delete), Ändern (update) und das Anlegen (insert) entsprechende Methoden an. Alle Methoden erwarten als Parameters die entsprechende Entität. Des Weiteren können alle Methoden die Exceptions „ConnectionException“ und „KeinZugriffException“ werden. Außerdem besitzt sie eine Backup-Methode, die die Datenbank dazu veranlasst, ein Backup zu erstellen.

### 11.7.2 Klasse ConnectionManager

Diese Klasse stellt eine Verbindung zur Datenbank her, dies geschieht mit einem entsprechenden Nutzernamen und Passwort. Sie kann die „AuthorizationException“ oder die „ConnectionException“ werfen.

### 11.7.3 Klasse CSVReader & CSVWriter

Diese Klassen sind wichtig, damit offline gearbeitet werden kann. Die Writer-Klasse implementieren genauso wie die Datenbank das Interface „EventListener“ um die offline Daten synchron zur Datenbank zu halten. Beim initialen Starten werden die wichtigen Daten heruntergeladen und als CSV-Datei gespeichert. Die Reader-Klasse ist nun für den Fall da, dass keine Verbindung zum Server besteht. Während nun die Datenbank-Klasse einen Fehler wirft wird die Anfrage an diese Klasse weitergeleitet, die dann aus der lokalen Kopie die Daten liest und zurückgibt.

## 11.8 Package gui

Dieses Package enthält alle Klassen, die für das Anzeigen der GUI nötig sind. Außerdem ist bezüglich des Eventhandling anzumerken, dass alle GUI-Events wie „onClick“ intern behandelt werden und nicht nach außen über Sender und Listener gegeben werden.

### 11.8.1 Klasse GUIHochzeitsmanager

Diese Klasse bildet den Einstieg in graphische Oberfläche dieser Applikation. Sie ist für das Management des Fensters (Frames) und etwaigen Status- und Menüleisten zuständig. Sie bietet eine Start und eine End-Methode an. Außerdem erbt sie von der Java-Klasse „JComponent“.

### 11.8.2 Klasse AbstractEventSender

Diese Klasse implementiert das Interface „EventSender“. Hiervon erben alle GUI-Klassen, die nicht schon von „GUIAbstractEntity“ erben. Diese Klasse (also GUIAbstractEntity) erbt auch von AbstractEventSender um eine Mehrfachvererbung zu vermeiden, die es in JAVA nicht gibt.

### 11.8.3 Klasse GUIAbstractEntity

Alle Klassen, die direkt einer Entität zugehörig sind, wie z.B. „GUIAktionen“, erben von dieser abstrakten Klasse. Dies hat den Vorteil, dass alle Klassen dieselben Standardmethoden haben. Zu diesen gehört es alle Entitäten anzuzeigen, Details zu einer anzuzeigen, Dialog zu löschen, anlegen Exportieren und zum Bearbeiten anzuzeigen mit entsprechenden Parametern. Genau diese Parameter sind generisch und deswegen mit „T“ anstatt einer speziellen Entität gekennzeichnet.

#### 11.8.3.1 POC

Um nun die Funktionsweise dieser generischen Vererbung zu veranschaulichen folgt ein kleiner POC:

```

1  /**
2   *   Diese Klasse ist eine abstrakte Beschreibung der möglichen Aktionen,
3   *   die eine GUI haben wenn sie zu einer bestimmten Entität (extends
4   *   AbstractElement)gebunden ist.
5   */
6  public abstract class GUIAbstractEntity<T extends AbstractElement> {
7      public abstract void showAll();
8      public abstract void showDetail(T entity);
9      public abstract void showLöschenDialog(T entity);
10 }
11
12 /**
13 *   Diese Klasse ist eine konkrete Implentierung für alle GUI-
14 *   Operationen an der Entität Aktion
15 */
16 public class GUIAktionen extends GUIAbstractEntity<Aktion> {
17     @Override
18     public void showAll() {}
19     @Override
20     public void showDetail(final Aktion entity) {}
21     @Override
22     public void showLöschenDialog(final Aktion entity) {}
23 }
24
25 /**
26 *   Diese Klasse ist die Überklasse aller Entitäten
27 */
28 public abstract class AbstractElement {
29 }
30
31 /**
32 *   Entity-Klassen für Aktionen
33 */
34 public class Aktion extends AbstractElement {
35 }

```

Die Klasse „GUIAktionen“ zeigt nun wie das Ganze funktioniert. Sie erweitert die Klasse „GUIAbstractEntity“ generisch mit „Aktion“, wobei „Aktion“ von der Klasse „AbstractElement“ erbt. Die Klasse „GUIAbstractEntity“ nimmt alle generischen Operatoren an, die von „AbstractElement“ erben. Wie man nun in „GUIAktionen“ sieht, sind alle Parameter, die vorher generisch waren nun vom Typ Aktion.

#### 11.8.4 Alle Entity GUI-Klassen

Alle GUI-Klassen, die von GUIAbstractEntity erweitern diese Klasse mit ihrer Entität generisch. Sie bieten neben den vererbten Methoden noch ein paar weitere an, je nachdem. Die Methodennamen sind selbsterklärend. Folgend nun noch ein paar besondere Methoden:

#### 11.8.5 GUIHochzeitsveranstaltung

Diese Klasse bietet unter anderem Export-Dialoge für die Gästeliste und den Speiseplan. Beide Male wird der „pdfExporter“ benutzt. Außerdem kann man sich alle Gäste und den Speiseplan nur so anzeigen lassen.

#### 11.8.6 GUICaterer

In diesem View kann man neues Essen und Trinken anlegen, da eine komplette Verwaltung dieser in separaten Views zu weit gehen würde. Außerdem kann man sich den Caterervergleich anzeigen lassen. Dafür werden die einzelnen Leistungen der Catere inklusive Preis verglichen und gegenüber gestellt. Die Nutzbarkeit ist stark abhängig davon, wie genau die Angebote spezifiziert wurden. Es bietet dem Nutzer somit einen schnellen Überblick über die Angebote.

#### 11.8.7 GUIAktionen

Diese View bietet eine visuelle Exportfunktion der Aktionen als Ablaufplan. Hierzu wird der „iCalHelper“ benutzt. Außerdem kann man hier auch die Aktion als Template speichern lassen. Hierzu wird auch Hilfe von der Klasse „XMLTransferer“ gebraucht.

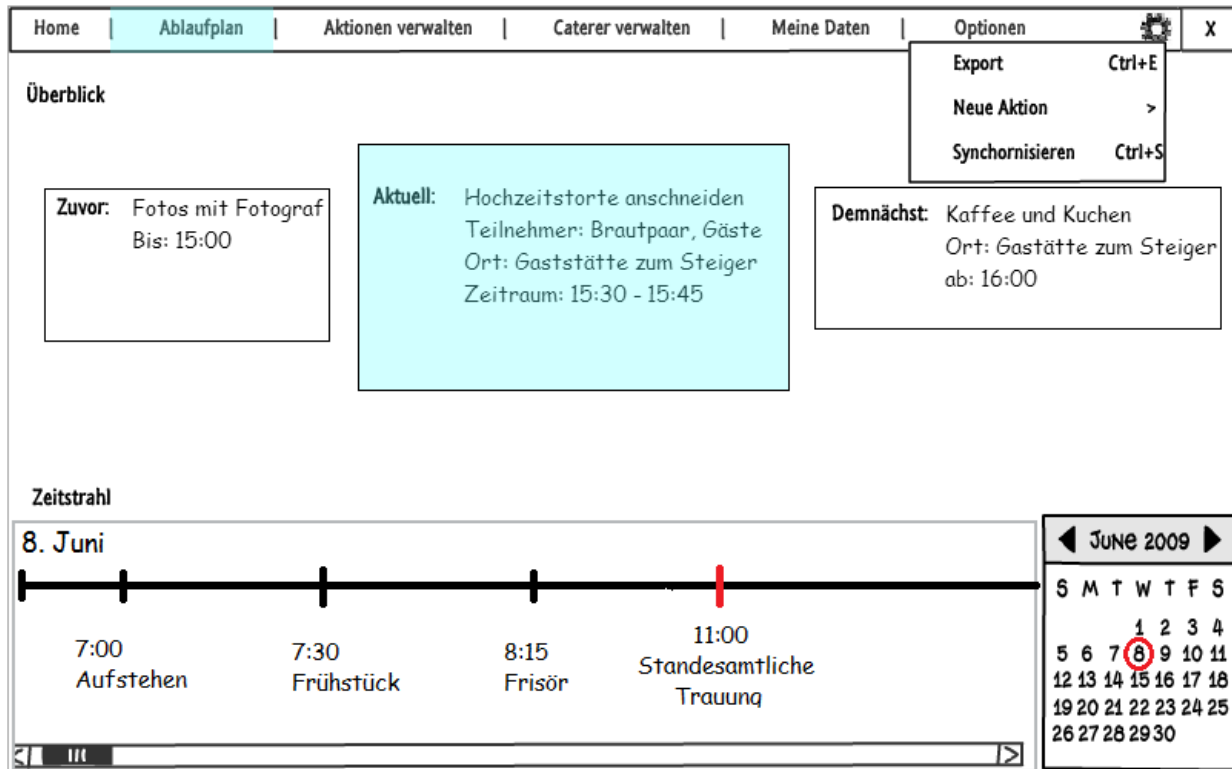
#### 11.8.8 Alle anderen GUI-Klassen

Alle anderen GUI-Klassen erben direkt von „AbstractSender“ und bieten Funktionalität an, die zu keiner Entität direkt gehört. Dazu gehören das Anzeigen des Import-Dialogs, der von überall erreichbar sein soll und die Kostenkontrolle.

## 12 GUI-Entwurf

### 12.1 Ablaufplan

Die GUI Skizze stellt den Ablaufplan dar, welcher dazu dient die Aktionen in ihrer Reihenfolge anzeigt. Sie besteht im wesentlichen aus 3 Teilen: Der Toolbar, dem Überblick und einem Zeitstrahl.



#### 12.1.1 Toolbar

Die Toolbar dient hier zur schnellen Navigation, dazu enthält sie folgende Elemente:

##### 12.1.1.1 Home

Beim klicken dieses Element gelangt der Nutzer zurück zum Home Screen, also dem Einstiegsbildschirm.

##### 12.1.1.2 Ablaufplan

Dieser Menüpunkt ist momentan ausgewählt und bringt einen Nutzer zu der Ansicht Ablaufplan in der sich die GUI momentan befindet.

##### 12.1.1.3 Aktionen verwalten

Dieser Punkt öffnet die Ansicht „Aktionen verwalten“. In dieser hat der Nutzer die Möglichkeit seine Aktion zu verwalten

#### 12.1.1.4 Caterer verwalten

Ähnlich zu der Ansicht „Aktionen verwalten“, kann in der Ansicht, welcher dieser Eintrag öffnet, die Caterer der Hochzeit verwalten

#### 12.1.1.5 Meine Daten

In dieser Ansicht hat der Nutzer die Möglichkeit seine persönlichen Daten anzupassen beziehungsweise zu aktualisieren.

#### 12.1.1.6 Optionen

Dieser Menüpunkt öffnet ein Dropdownmenü, welches vom Kontext der aktuellen Ansicht abhängig ist. Hier sind es die Optionen: Export, welches den Ablaufplan als iCal Format exportiert; Neue Aktion: öffnet einen Dialog um eine neue Aktion anzulegen; Synchronisieren: startet den Synchronisationsprozess mit der zentralen Datenbank.

#### 12.1.1.7 Einstellungen

Beim Klick auf das Zahnrad, öffnet sich dem Nutzer ein Fenster in dem Einstellungen vornehmen kann.

#### 12.1.1.8 Beenden

Beim Klick auf das Beenden-Symbol erscheint dem Nutzer zunächst ein Dialog ob er das Programm wirklich beenden möchte, wenn die Option „Ja“ wählt wird das Programm beendet. Insofern er sich für „Nein“ entscheidet läuft die Anwendung weiter.

### 12.1.2 Überblick

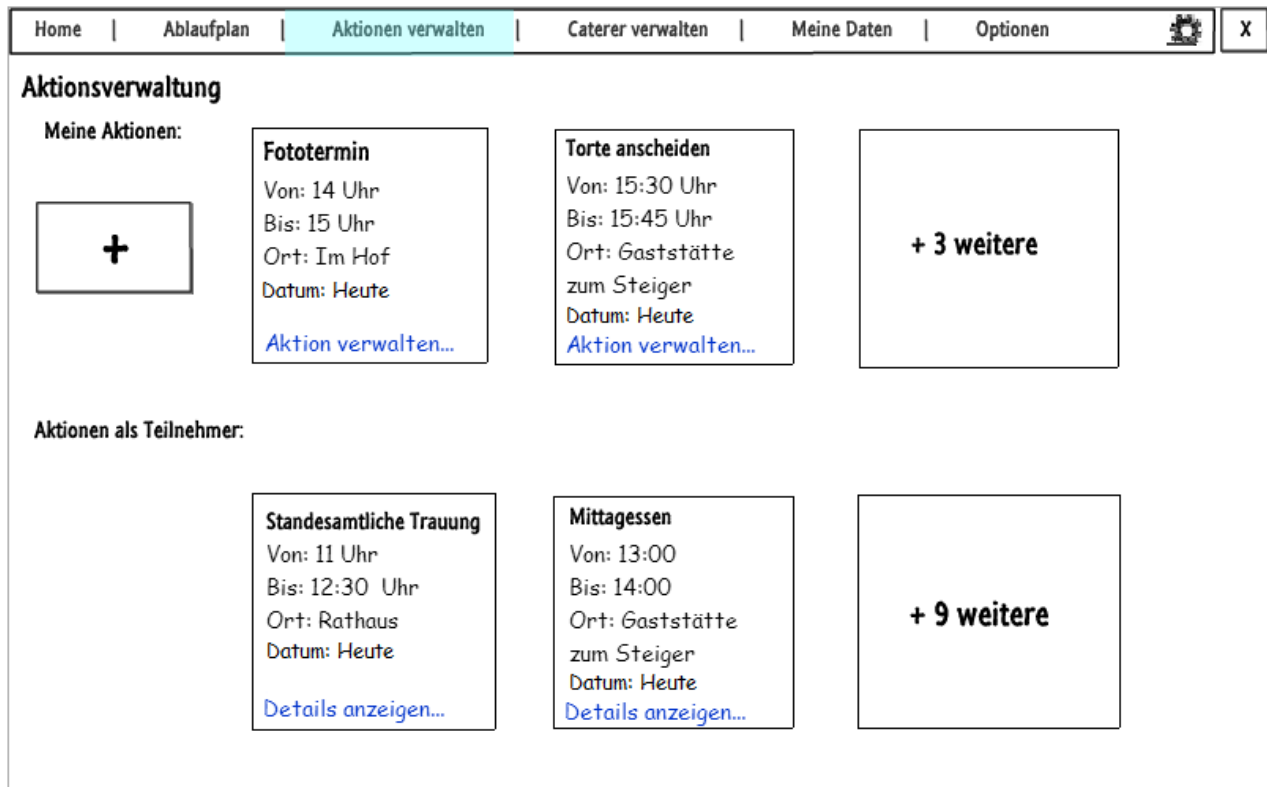
Dieser Teil der GUI enthält, die aktuell für den Nutzer interessanten Daten, das heißt konkret es werden, in sofern vorhanden, 3 Aktionen inklusive ihrer Informationen angezeigt: die aktuell stattfindende, die vorrangegangene, sowie die auf die aktuelle folgende Aktion. Das ermöglicht dem Nutzer mit einem Blick alle momentan relevanten Daten zu erfassen und ihm einen Überblick zu verschaffen. Wenn man auf einer der Aktion klickt, wird die entsprechende Aktion mit ihren Details geöffnet.

### 12.1.3 Zeitstrahl

In diesem Teil wird ein Zeitstrahl mit allen Aktionen angezeigt, an denen man an diesem Tag teilnimmt. Der Tag wird über einen Kalender rechts vom Zeitstrahl ausgewählt, standardmäßig ist der aktuelle Tag ausgewählt. Aufgrund der eventuellen Länge des Zeitstrahls ist der Bereich mit einer Scrollbar ausgestattet, damit er dennoch lesbar ist. Meilensteine werden von den normalen Aktionen hervorgehoben, in dieser Skizze ist die Standesamtliche Trauung als Meilenstein markiert. Beim Klick auf eine Aktion wird die korrespondierende Aktion geöffnet.

## 12.2 Aktionen verwalten

Diese Ansicht besteht ebenfalls aus 3 Teilbereichen und bietet dem Nutzer die Möglichkeit eine Übersicht über Aktionen sowie Verwaltungsmöglichkeiten.



### 12.2.1 Toolbar

#### 12.2.1.1 Home

Beim klicken dieses Element gelangt der Nutzer zurück zum Home Screen, also dem Einstiegsbildschirm.

#### 12.2.1.2 Ablaufplan

Dieser Menüpunkt öffnet die Ansicht der Ablaufplanes.

#### 12.2.1.3 Aktionen verwalten

Dieser Punkt ist momentan ausgewählt.

#### 12.2.1.4 Caterer verwalten

Ähnlich zu der Ansicht „Aktionen verwalten“, kann in der Ansicht, welcher dieser Eintrag öffnet, die Caterer der Hochzeit verwalten

#### 12.2.1.5 Meine Daten

In dieser Ansicht hat der Nutzer die Möglichkeit seine persönlichen Daten anzupassen beziehungsweise zu aktualisieren.

#### 12.2.1.6 Optionen

Dieser Menüpunkt öffnet ein Dropdownmenü, welches vom Kontext der aktuellen Ansicht abhängig ist. Hier sind es die Optionen: Export, welches Aktionen als iCal Format exportiert; Neue Aktion: öffnet einen Dialog um eine neue Aktion anzulegen; Synchronisieren: startet den Synchronisationsprozess mit der zentralen Datenbank.

#### 12.2.1.7 Einstellungen

Beim Klick auf das Zahnrad, öffnet sich dem Nutzer ein Fenster in dem Einstellungen vornehmen kann.

#### 12.2.1.8 Beenden

Beim Klick auf das Beenden-Symbol erscheint dem Nutzer zunächst ein Dialog ob er das Programm wirklich beenden möchte, wenn die Option „Ja“ wählt wird das Programm beendet. Insofern er sich für „Nein“ entscheidet läuft die Anwendung weiter.

### 12.2.2 Meine Aktionen

Hier werden dem Nutzer seine erstellten Aktionen angezeigt, die mit einem Mausklick auf die entsprechende Aktion auch verwalten kann. Ebenfalls hat er die Möglichkeit mit einem Klick auf der „Hinzufügen“-Kachel eine neue Aktion zu erstellen. Um den Nutzer nicht mit Informationen zu überladen, werden nur die aktuellsten Aktionen zu Beginn angezeigt, er hat aber die Möglichkeit mithilfe der „<Anzahl> weitere“ Kachel die restlichen Aktionen anzuzeigen. Diese werden dann nachgeladen und ebenfalls angezeigt.

### 12.2.3 Aktionen als Teilnehmer

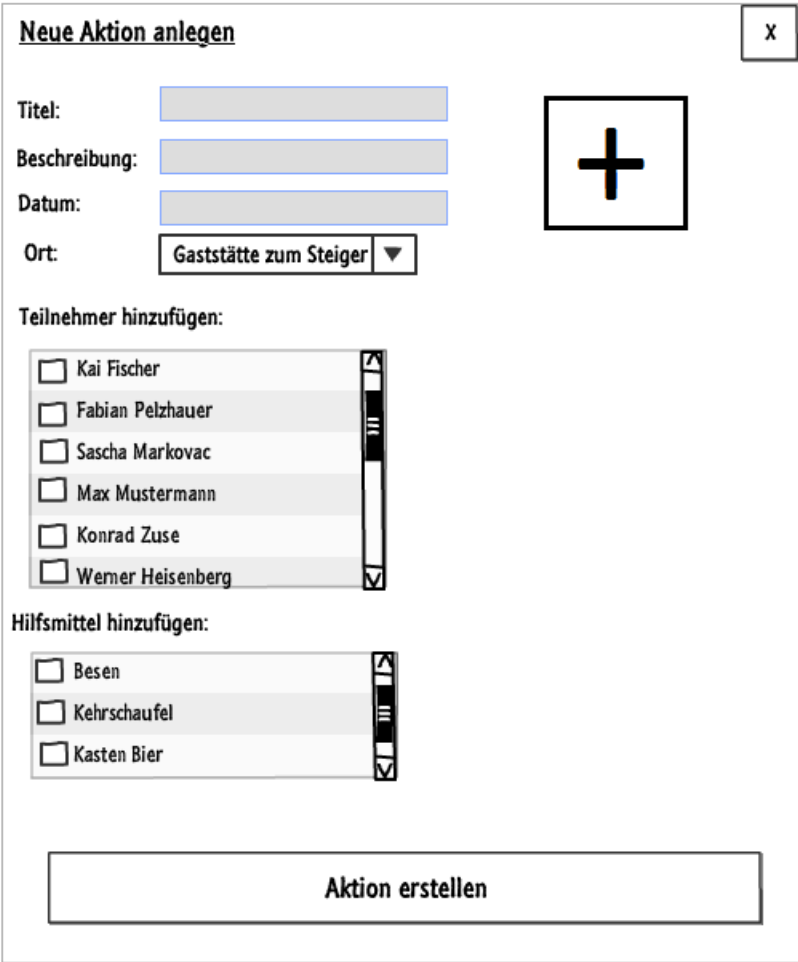
Hier werden dem Nutzer die Aktionen angezeigt an denen er als Teilnehmer eingetragen ist. Es werden nur die wichtigsten Daten zur Aktion in den Kacheln angezeigt. Durch Klick auf eine Aktionskachel wird ihm die Aktion in Detailansicht angezeigt und der Nutzer kann alle Daten einsehen. Ähnlich wie im Bereich „Meine Aktionen“ werden zu Beginn nicht alle Aktionen angezeigt und der Nutzer kann sich diese noch nachladen.



## 12.3 Neue Aktion anlegen

Dieser Dialog wird geöffnet wenn der Nutzer eine neue Aktion anlegen möchte, dies kann über verschiedene Wege geschehen, dazu zählen: über Optionen in der Toolbar oder über „neue Aktion anlegen“ in Einstiegsbildschirm, sowie in Menüpunkt Aktionen verwalten.

Der Dialog besteht aus einem Formular in dem der Nutzer grundlegende Daten zu der Aktion in textueller Form angeben kann, dazu gehören: Titel, Beschreibung und Datum. Weiterhin kann er aus einen Ort mittels eines Dropdown Menüs auswählen. Diese Felder sind die wichtigsten um eine Aktion zu erstellen, werden bereits die anderen benötigt kann der Nutzer diese mit einem klick auf die Plus-Kachel anzeigen lassen. Allerdings gilt hier wieder die Oberfläche übersichtlich zu halten und nicht zu überladen, daher werden die restlichen Felder erst versteckt. Ebenfalls kann er bereits Teilnehmer und Hilfsmittel hinzufügen. Dies ist jeweils mit einer List realisiert in der Nutzer dann mittels Checkbox markieren kann welche Hilfsmittel beziehungsweise Teilnehmer hinzugefügt werden sollen. Schließlich kann er mithilfe des Buttons „Aktion erstellen“ die Aktion erstellen, insofern kein Fehler auftritt wird der Dialog dann geschlossen. Falls der Nutzer doch keine Aktion erstellen möchte dann kann er das Fenster mittels des „Schließens-Buttons“ oben rechts schließen.



**Neue Aktion anlegen** [X]

Titel:

Beschreibung:

Datum:

Ort:

Teilnehmer hinzufügen:

- ☐ Kai Fischer
- ☐ Fabian Pelzhauer
- ☐ Sascha Markovac
- ☐ Max Mustermann
- ☐ Konrad Zuse
- ☐ Werner Heisenberg

Hilfsmittel hinzufügen:

- ☐ Besen
- ☐ Kehrschaufel
- ☐ Kasten Bier

**Aktion erstellen**