2015

Hausarbeit Softwaretechnik Projekt
Thema: "Wer zahlt was?"

Stefan Sander
Jan Wieland
Michael Trotzer

11.07.2015

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Wie ist es zum Erstellen der Web-Applikation „Wer zahlt was?“ gekommen? Alles begann mit einem Gruppenurlaub an der Ostsee. Im Laufe des Kurzurlaubes, Dauer 5 Tage, hatte jeder Mitreisende einige Einkäufe und Eintrittsgelder bezahlt. Wieder in Berlin angekommen stellten sich die Urlauber die Frage nach der Aufteilung der Ausgaben. Als endlich alle Rechnungen zusammengetragen worden sind und diese dann auch den jeweiligen Personen zugeordnet wurden, konnte mithilfe einer Excel Tabelle berechnet werden, wer wie viel und vor allem wer welche Ausgaben zu tragen hat.

Die Rechnung wurde mithilfe einer Excel Tabelle erstellt. Der Aufwand und die Komplexität so eine Tabelle zu erstellen, hält sich zwar in Grenzen, dennoch stellte sich die, in der IT allgegenwärtige, Frage:

- *Kann man das nicht einfacher lösen?*

1.1 Die Idee

Die Idee vom Tool "Wer zahlt was?" ist es, auf clevere und faire Weise, Ausgaben unter Freunden auszurechnen und zu regeln. "Wer zahlt was?" hilft bei den folgenden Fragestellungen:

- Wer hat, für was, Geld vorgestreckt?
- Hatte Jan nicht Geld an Stefan verliehen?
- Wer zahlt denn nun am Ende welchen Betrag?

Die große Frage lautet also: ***Wer schuldet wem was?***

Ein Anwender legt eine Gruppe und die dazugehörigen Gruppenmitglieder für das Wochenende bei "Wer zahlt was?" an. Anschließend können die Gruppenmitglieder Ihre Ausgaben in wenigen Schritten eingeben. Am Ende berechnet das Tool für jedes Gruppenmitglied die offenen Ausgaben gegenüber den Anderen. Jeder sieht exklusiv wer wem was und wie viel schuldet. Somit werden die Schulden schnell bezahlt und alle Gruppenmitglieder haben mehr Zeit für die wichtigen Dinge im Leben.

1.2 Vorüberlegungen

Ziel ist die Erstellung einer Anwendung zum unkomplizierten Abrechnen von Ausgaben in kleinen Gruppen. Um die Ausführung der Anwendung möglichst unkompliziert und so vielen Anwendern wie möglich zu gestatten haben wir uns für die Umsetzung als Web-Applikation entschieden. Die mobile Optimierung ist beabsichtigt, wird aber vermutlich nicht mit dem ersten Release umgesetzt werden

können. Die Umsetzung wird mittels des Django Framework realisiert, dieses haben wir im letzten Semester im Modul Internettechnologien Client/Server kennengelernt.

In dieser Anwendung soll es keine klassischen Benutzerkonten geben. Wir verwenden eine Art Gruppenanmeldung über einen Gruppen-Token. Dies soll eine möglichst unkomplizierte Verwendung ermöglichen.

Es gibt drei Arten von Objekten: **Gruppen, Personen** und **Ausgaben**.

Gruppen:
Gruppen repräsentieren den Container für alle anderen Aktivitäten. Um Personen oder anschließend Ausgaben anzulegen, muss als erstes eine Gruppe erstellt werden. Beim Erstellen einer Gruppe wird ein aus 19 Zeichen langer Gruppen-Token generiert. Dieser Token wird verwendet, um die Gruppe aufzurufen und zu bearbeiten. Der Token ist nicht änderbar und jeder, der Kenntnis von diesem erlangt, kann auf die Gruppe zugreifen. Ein Passwortschutz der Gruppe ist derzeit nicht vorgesehen. Gruppen haben eine Beschreibung und einen Anzeigenamen. Außerdem wird das Datum des letzten Zugriffs gespeichert. Dies ist notwendig, um festzustellen, welche Gruppen nicht mehr verwendet werden. Gruppen sollen nach 90-tägiger Nichtverwendung gelöscht werden. Wenn eine Gruppe gelöscht wird, gehen auch alle dieser Gruppe zugehörigen Ausgaben und Personen verloren.

Personen:
Personen können erst erstellt werden, wenn eine Gruppe angelegt wurde. Personen haben einen Namen. Sie können sowohl Besitzer als auch Teilhaber von Ausgaben sein.

Ausgaben:
Das Anlegen von Ausgaben ist erst möglich, wenn eine Gruppe und mindestens eine Person in dieser Gruppe, angelegt wurde. Ausgaben haben einen Besitzer. Das ist die Person, welche für die Ausgaben aufgekommen ist. Die Ausgaben können auf eine bis beliebig viele Personen aufgeteilt werden. Dabei muss der Besitzer nicht unbedingt eine der Personen sein. Dadurch ist es möglich festzuhalten, ob Geld an eine andere Person verliehen wurde.

Zum Abschluss wird es eine Ausgabe der Schulden geben, einen sogenannten Report. In diesem Report werden wir aufschlüsseln "wer wem was" schuldet. Es ist nicht vorgesehen, dass Ausgaben für einzelne Personen als bezahlt markiert werden können.

2. Anforderungen

Die Anforderungen an das Tool wurden aus den Vorüberlegungen abgeleitet. Es wurde eine Aufteilung der Anforderungen in zwei Gruppen *(funktionale und nichtfunktionale Anforderungen)* vorgenommen. Nachdem alle Anforderungen zu den beiden Gruppen gefunden und grob aufgeschrieben wurden, erfolgte die Priorisierung der Anforderungen. Die folgenden Gruppeneinteilungen haben wir bei der Priorisierung gewählt:

- *1. (PFLICHT - muss)*
- *2. (WUNSCH – soll/sollte)*
- *3. (ABSICHT - wird)*
- *4. (VORSCHLAG - kann)*

Nachfolgend sind die Anforderungen nach ihrer Gruppenzugehörigkeit und der jeweiligen Priorität aufgegliedert.

2.1 Funktionale Anforderungen

Funktionale Anforderungen sind Aussagen über eine zu erfüllende Eigenschaft oder eine zu erbringende Leistung unseres Systems (Tab.1). Genauer gesagt beschreibt eine funktionale Anforderung: *„was das System leistet“*

ID	Bezeichnung	Priorität
1	Umsetzung als Web-Applikation	1
2	Gruppen erstellen	1
3	Personen erstellen	1
4	Ausgaben erstellen	1
5	Report ausgeben	1
6	Report als PDF exportieren	3
7	Gruppe exportieren	2
8	Gruppe importieren	2
9	Gruppe löschen, inklusive zugehörige Personen und Ausgaben	2
10	Löschen der Gruppen nach 90- tägiger Nichtverwendung	3
11	Aufruf einer Gruppe über Gruppen-Token	2
12	Gruppen-Token erstellen	1

TAB. 1 FUNKTIONALE ANFORDERUNGEN FÜR WZW

Folgende funktionale Anforderungen wurden als 1. *(PFLICHT – muss)* Anforderungen definiert:

- Umsetzen als Web-Applikation
- Gruppen erstellen
- Personen erstellen
- Ausgaben erstellen
- Report ausgeben
- Gruppen-Token erstellen

Jeder dieser Anforderungen ist explizit nötig, um ein funktionierendes Tool zu gewährleisten. Ein Beispiel wäre es, dass Gruppen, Personen und Ausgaben unbedingt erstellbar sein müssen, um überhaupt das Grundkonzept von „Wer zahlt was?" zu garantieren, nämlich ausrechnen zu lassen "wer wem was" schuldet. Ohne das Anlegen von Gruppen, Personen und Ausgaben wäre das Tool nicht zu benutzen.

Weitere Anforderungen die in die Kategorie *2. (WUNSCH – soll/sollte)* fallen, sind:

- Gruppe exportieren
- Gruppe importieren
- Gruppe löschen, inklusive zugehörige Personen und Ausgaben
- Aufruf einer Gruppe über Gruppen-Token

Diese Anforderungen sind nicht explizit nötig, um die Funktionalität des Tools zu gewährleisten. Darunter zählt das Exportieren, zur lokalen Sicherung der Gruppe und Importieren einer kompletten Gruppe. Außerdem das Löschen einer Gruppe inklusive aller zugehörigen Personen und Ausgaben.

Anforderungen der Kategorie *3. (ABSICHT – wird)* werden nicht unbedingt benötigt, um das Tool überhaupt bedienen zu können, allerdings sind es sinnvolle Anforderungen, welche dem Nutzer als Hilfestellungen, für die Weiterverarbeitung, oder Hinweise dienen. Folgende Anforderungen fallen darunter:

- Report als PDF exportieren
- Löschen der Gruppen nach 90-tägiger Nichtverwendung

2.2 Nichtfunktionale Anforderungen

Im Gegensatz zu den funktionalen Anforderungen geben nichtfunktionale Anforderungen an, wie gut ein System etwas leisten soll (qualitativ). Nachfolgend die nichtfunktionalen Anforderungen, mit Priorisierung:

ID	Bezeichnung	Priorität
13	SSL-Verschlüsselung für gesicherte Verbindung	3
14	Daten der Anwender werden vertraulich behandelt	1
15	Veröffentlichung unter freier Lizenz	2
16	Veröffentlichung unter GitHub	2
17	Mobil-Optimiert	4
18	Gleichzeitig bis zu 100 Anwender im System	2
19	Kontrastreiches Farbschema für bessere Lesbarkeit	2
20	Barrierefrei	3
21	Feedback Fenster	3

TAB. 2 NICHTFUNKTIONALE ANFORDERUNGEN FÜR WZW

Folgende nichtfunktionale Anforderung wurde als 1. *(PFLICHT – muss)* Anforderung definiert:

- Daten der Anwender werden vertraulich behandelt

Wie bereits bei den funktionalen Anforderungen beschrieben, muss diese Anforderung im Tool umgesetzt sein, damit es den datenschutzrechtlichen Standards entspricht. Nur so darf das Tool überhaupt "live geschaltet" werden.

Weitere nichtfunktionale Anforderungen, die in die Kategorie *2. (WUNSCH – soll/sollte)* fallen, sind:

- Kontrastreiches Farbschema für bessere Lesbarkeit
- Gleichzeitig bis zu 100 Anwender im System
- Veröffentlichung unter GitHub
- Veröffentlichung unter freier Lizenz

Diese Anforderungen spielen für den Anwender eine große Rolle, erst das Veröffentlichen des Tools ermöglicht eine allgemeine Nutzung.

Anforderungen der Kategorie *3. (ABSICHT – wird)* werden nicht unbedingt benötigt, um das Tool überhaupt bedienen zu können, allerdings sind es sinnvolle Anforderungen, welche dem Nutzer zum einen eine bestimmte Datensicherheit gewährleisten (SSL-Verbindung), oder dem Nutzer einräumen Verbesserungsvorschläge und Wünsche direkt an das Entwicklerteam, über ein Kontaktfenster, zu senden.

- SSL-Verschlüsselung für gesicherte Verbindung
- Barrierefrei
- Feedback Fenster

Anforderungen der Kategorie *4. (VORSCHLAG - kann),* haben eher einen zusätzlichen Charakter und können je nach Entwicklungsstand und Nachfrage der Anwender miteingebunden werden. Bei uns wäre das im Moment nur:

- Mobil-Optimierung

3. Diagramme

Um in der Entwicklung der Web-Applikation immer einen genauen Überblick über die Anforderungen zu erhalten, haben wir uns nachfolgende Diagramme überlegt. An Hand derer wir unsere Web-Applikation gestaltet haben.

3.1 Use-Case-Diagramm

Mittels eines Use-Case Diagramms wird veranschaulicht, wer welche Aufgaben im System durchführen kann. Dazu haben wir uns zwei Use-Case Diagramme überlegt, die dies veranschaulichen sollen. Im ersten Use-Case zeigen wir grob, was der Anwender mit dem System machen kann. Auch der Sysadmin wird hier mit aufgezeigt (Abb. 1). Im zweiten Use-Case gehen wir dann näher auf die einzelnen Unteraufgaben ein, die das System dem Anwender und Sysadmin bietet (Abb. 2).

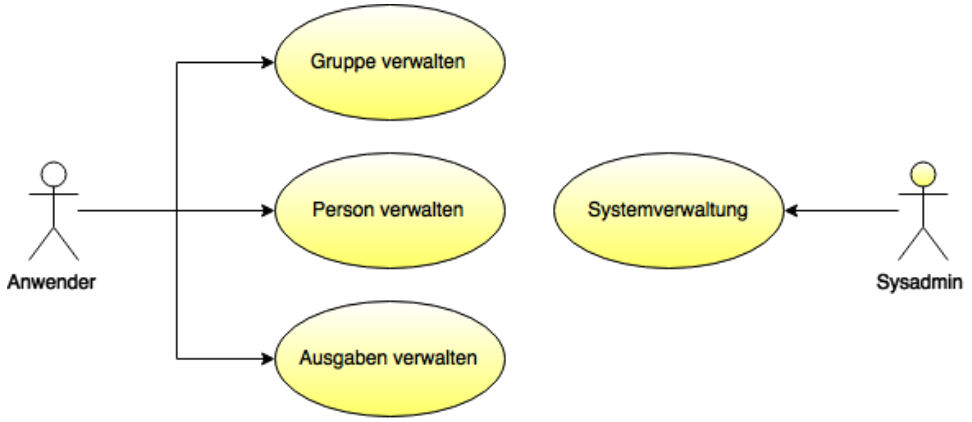

ABB. 1 USE-CASE GROB

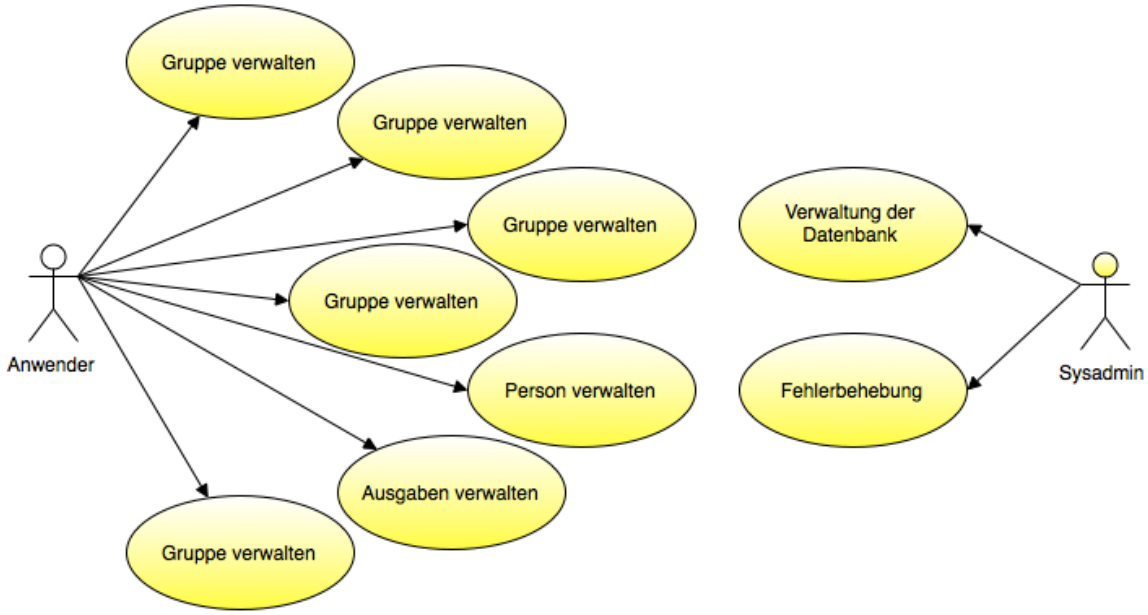

ABB. 2 USE-CASE FEIN

3.2 Sequenzdiagramm

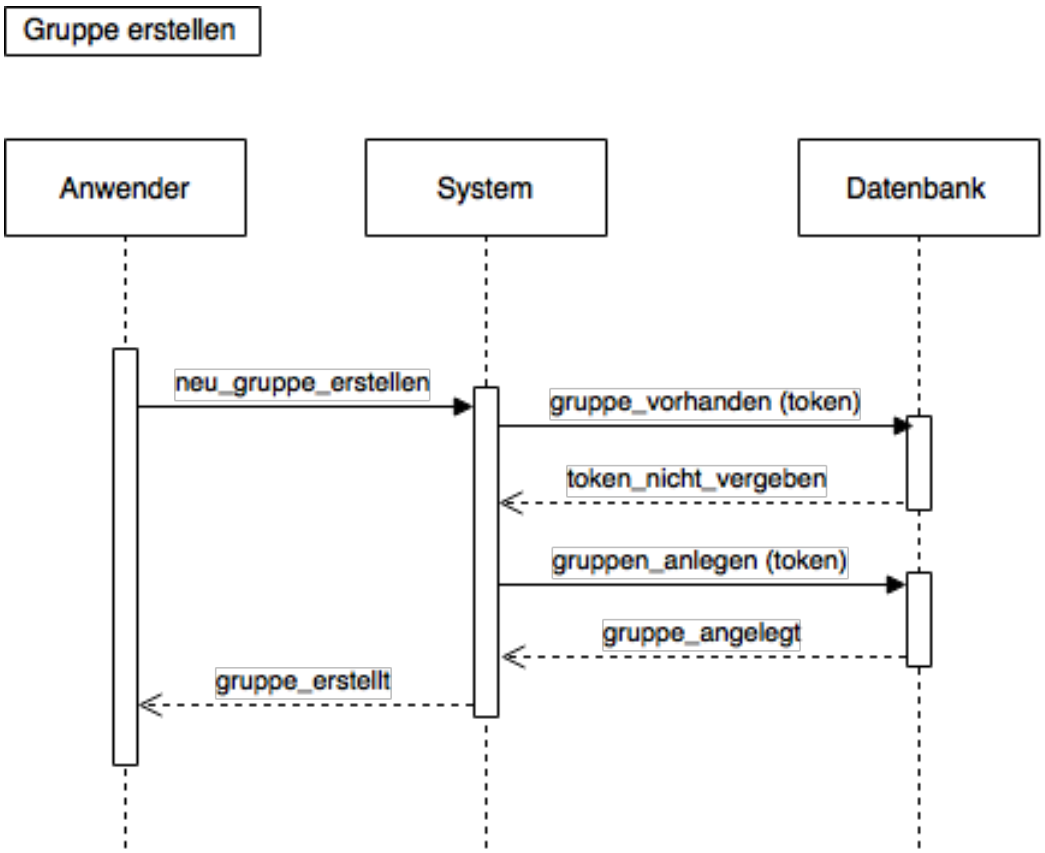

ABB. 3 SEQUENZ DIAGRAMM – ERSTELLEN EINER NEUEN GRUPPE

Bei der Abfolge „Gruppe erstellen“ haben wir uns bewusst dazu entschieden, nur den Fall abzudecken, dass eine Gruppe mit ihrem speziellen Gruppen-Token noch nicht vorhanden ist. Die Möglichkeit, dass die Gruppe bereits existiert, ist schwindend gering, weshalb wir diesen Fall im Sequenzdiagramm vernachlässigen konnten. Sonst hätte er als eine Alternative mit aufgezeigt werden müssen.

3.3 Klassendiagramm

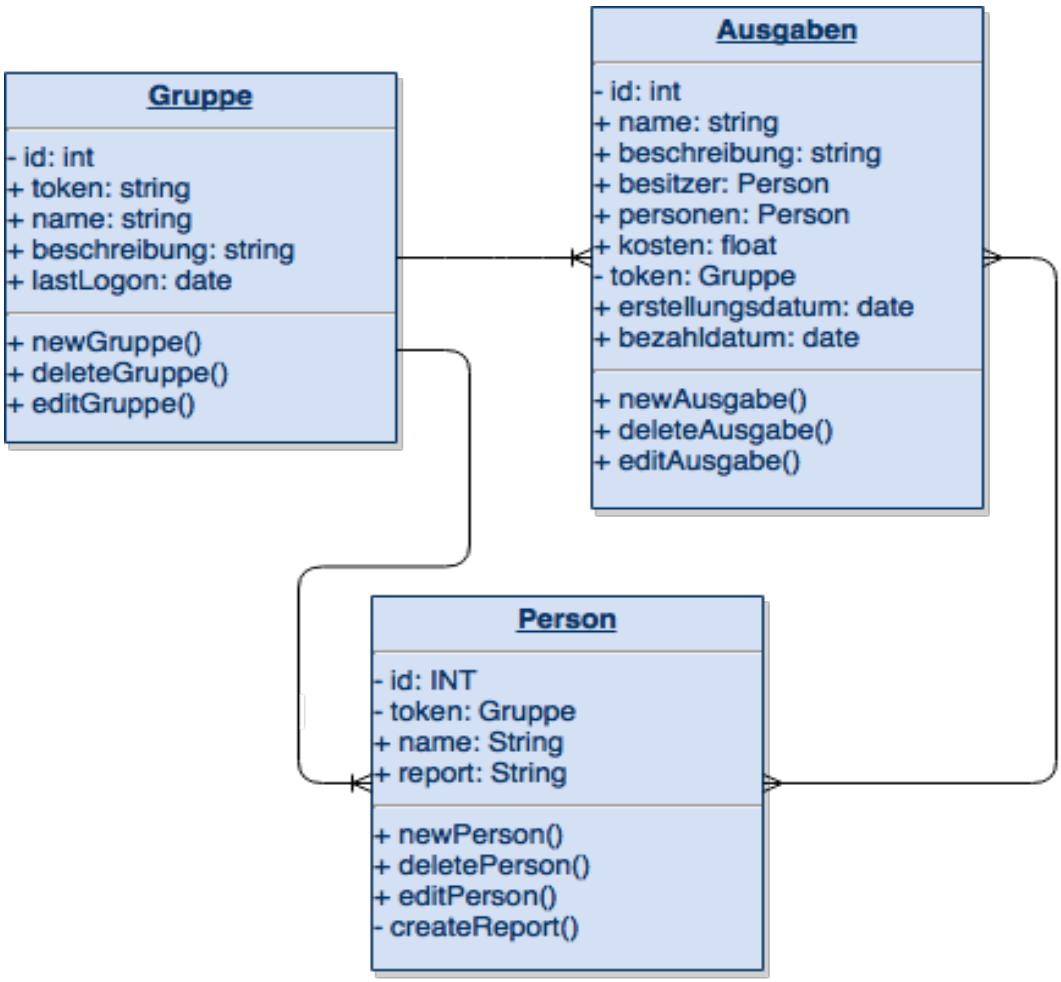

ABB. 4 KLASSENDIAGRAMM

Im Klassendiagramm sind alle Klassen dargestellt, die wir für unser Softwareprojekt brauchen. Wir beschränken uns hier auf die drei oben dargestellten Klassen „Gruppe“, „Ausgaben“ und „Person“. Des Weiteren sind die zu den Klassen gehörenden Attribute und Methoden aufgeführt, die später umgesetzt werden sollen. Das Attribut „Token“ ist als einziges Attribut nicht sichtbar, da es nicht veränderbar sein soll.

3.4 Struktogramm

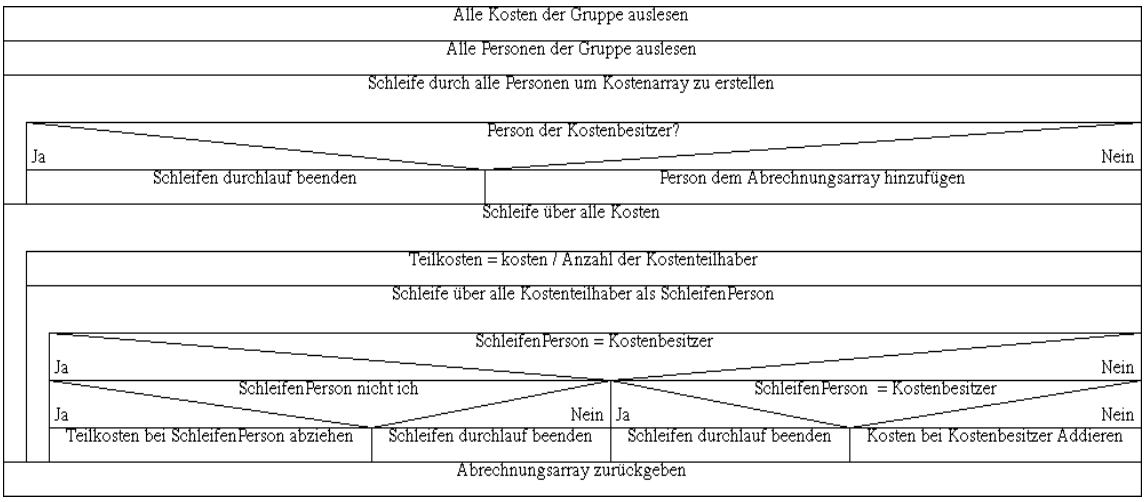

ABB. 5 STRUKTOGRAMM - REPORT ERSTELLEN

In diesem Struktogramm ist zu sehen, was alles erfüllt sein muss, um den Report zu erstellen.

3.5 GUI

Für die Darstellung der Web-Applikation im Webbrowser haben wir uns für eine einfache und übersichtliche Anzeige entschieden. Farben spielen hier noch keine Rolle. Diese werden erst später genau definiert und ins Projekt integriert. Nachfolgend kann man sehen, wie wir uns die Umsetzung der Anzeigen vorgestellt haben. Dazu haben wir exemplarisch die Startseite (Abb. 5), die Personenanzeige (Abb. 6) und die „Ausgaben erstellen“-Seite (Abb. 7) abgebildet, wie nachfolgend zu sehen ist.

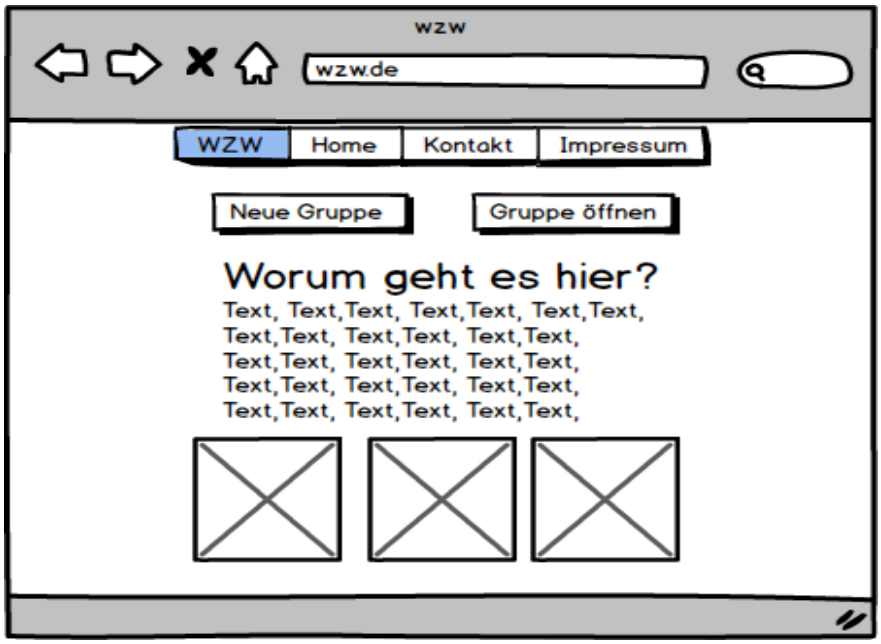

ABB. 6 STARTSEITE

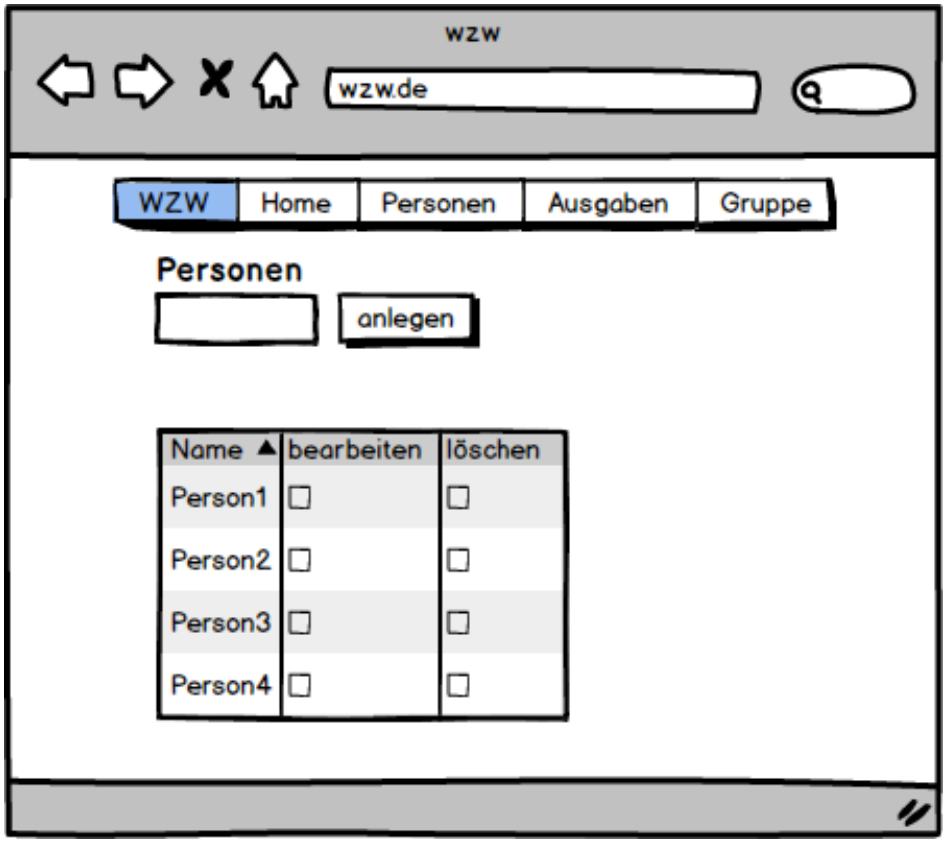

ABB. 7 PERSONENANZEIGEN

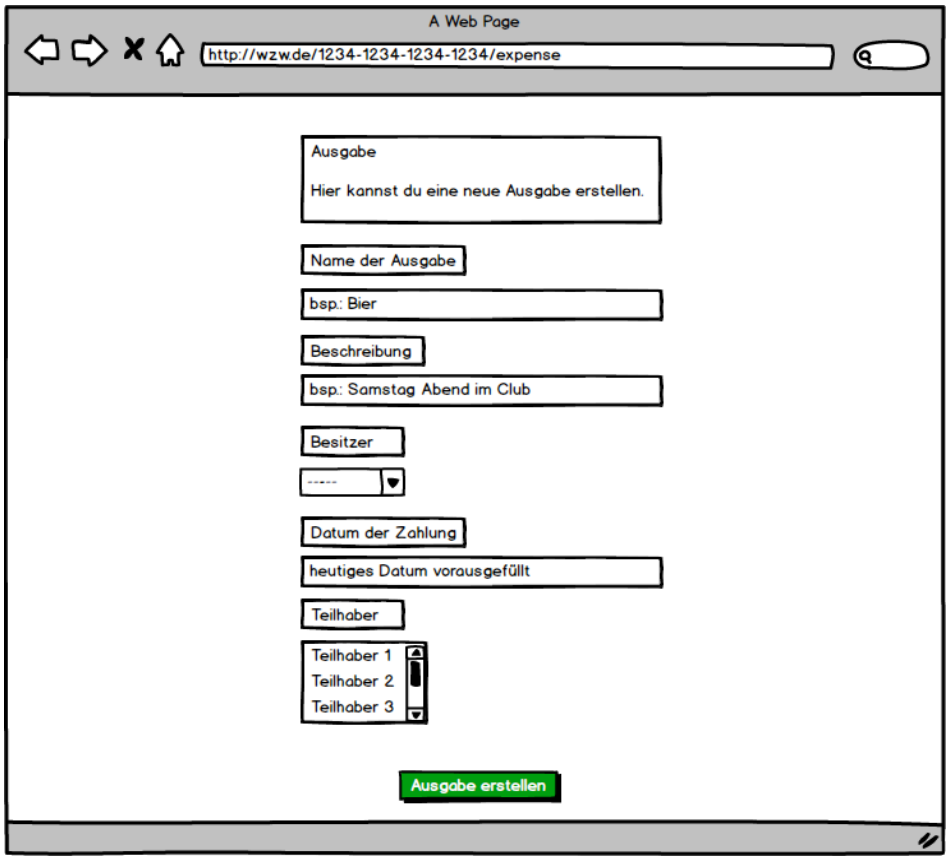

ABB. 8 AUSGABE ERSTELLEN

4. Datenmodell

Das Datenmodell ist eine Erweiterung des Klassendiagramms. Hier zeigen wir auf, welche Tabellen mit welchen Werten belegt werden und wie diese Tabellen untereinander verknüpft sind. Hierbei fällt auf, dass die Ausgabentabelle die wichtigste Tabelle ist, da sie mit den anderen Tabellen eine direkte Verknüpfung aufbaut.

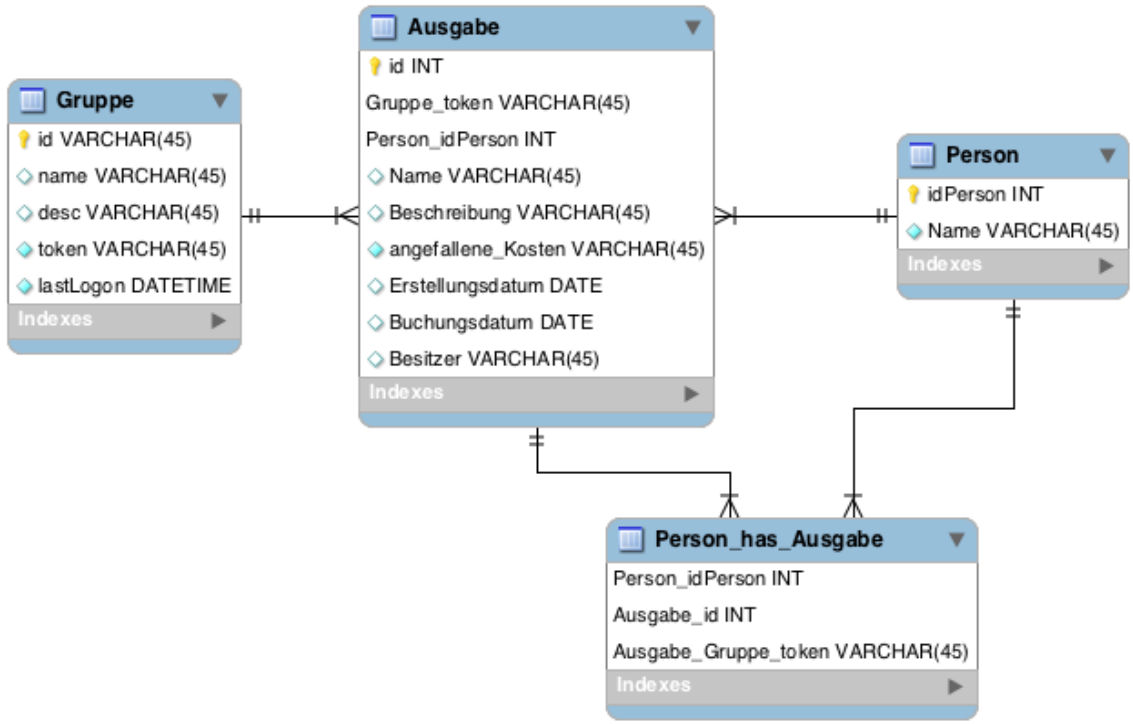

ABB. 9 DATENBANKMODELL

5. Technologien

Die nachfolgenden Technologien und Programme wurden zur Umsetzung des Projektes benutzt. Es handelt sich hierbei um verschiedene Programme, welche individuelle Aufgaben bei der Entwicklung des Projektes abbilden. In den einzelnen Abschnitten sind diese Programme definiert und die Verwendung zum Projekt wird näher erläutert.

5.1 Versionskontrolle

Um ein gemeinsames Arbeiten im Team zu ermöglichen, wurde am Anfang der Entwicklung des Projektes ein Versionskontrollsystem eingeführt. Wir haben uns für GitHub entschieden. GitHub basiert, wie der Name vermuten lässt, auf GIT. Dieses wurde von Linus Torvald entwickelt. Es ist zurzeit das meist verwendete Versionskontrollsystem. GitHub verwendet ein Optimistic-Locking, dabei wird die Datei zum bearbeiten nicht gesperrt. Es kann somit passieren, dass zwei Personen die selbe Datei bearbeiten und es zu Konflikten kommt. Um dies zu umgehen, haben wir in eigenen Zweigen gearbeitet. Somit hat jeder in seiner

Repository Branche die Daten bearbeitet und anschließend die Änderungen in der Master Branche zusammengeführt.

5.2 Programmiersprache

Für die Programmierung haben wir uns für Python entschieden. Python ist eine universell einsetzbare Sprache, die auf so gut wie jedem Betriebssystem läuft. Die Erfinder haben sich auf die Fahne geschrieben eine möglichst kompakte und einfach zu lesende Sprache zu entwickeln.

5.3 Framework

Bei der Erstellung der Web-Applikation haben wir zwei verschiedene Frameworks verwendet. Für die Logik haben wir uns für Django entschieden und das Layout wird mit Hilfe von Bootstrap erstellt.

5.3.1 Backend

Für die Bearbeitung des Backend wurde Django ausgewählt. Es ist ein in Python geschriebenes quelloffenes Framework für Web-Applikationen und wurde im Jahre 2005 veröffentlicht. Ursprünglich wurde es entwickelt, um die News-Seite „Lawrence Journal-World“ zu verwalten. Es hat sich allerdings schnell für andere Themengebiete einen Bekanntheitsgrad verschafft.

5.3.2 Frontend

Für die Bearbeitung des Frontend wurde Bootstrap benutzt. Es bietet vorkonfigurierte HTML- und CSS-Befehle, um einfache und schnelle Vorlagen zu generieren. Diese können nach den eigenen Vorgaben angepasst werden. Mit wenigen HTML und CSS Kenntnissen ist es so mit diesem Programm möglich ein schönes Design zu entwickeln. Entwickelt wurde es von Twitter Mitarbeitern und diente über ein Jahr als Styleguide für interne Tools.

5.4 Datenbank

Wir verwenden eine SQLite Datenbank, da diese die Standard Datenbank Engine von Django ist. SQLite verwendet die Standard SQL Abfrage Syntax und benötigt keinen separaten Server. Die Datensätze werden in eine einzige Datei geschrieben und von dort gelesen. Diese Datei ist plattformunabhängig und kann in ein anderes System kopiert und weiterverwendet werden. Wir haben uns daher für keine andere entschieden, da es die Standard Einstellung ist und wir die Datenbank leicht über mehrere Systeme kopieren konnten.

5.5 Entwicklungsumgebung

Als Entwicklungsumgebung haben wir uns für PyCharm entschieden. Es ist eine integrierte Entwicklungsumgebung (IDE) der Firma JetBrains für die Programmiersprache Python. Wir

haben mit der aktuellen Version 4.5.1 gearbeitet, diese wäre auch durch Plug-Ins erweiterbar. PyCharm ermöglicht die einfache Verwendung mehrerer Refactoring Methoden, von denen wir am häufigsten die Methode "rename" benutzten. Ein weiterer Vorteil der Entwicklungsumgebung war der integrierte GitHub Support, welchen wir als Versionsverwaltung, wie oben geschrieben, benutzt haben.

5.6 Sonstiges

Weitere Technologien, die wir zur Unterstützung des Projektes benutzt haben, sind:

- Draw.io - Webanwendung zum Erstellen von UML Diagrammen
- FreeMind - OpenSource Anwendung zum Erstellen von Mindmaps

6. Software Architektur

Bei der Software Architektur haben wir uns der MVC-Architektur bedient. Diese ist in Django eher eine MTV-Architektur.

Funktionsweise von MVC

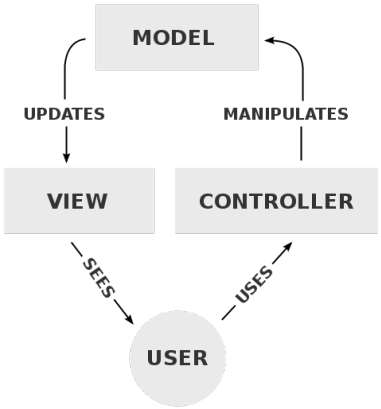

ABB. 10 MVC DIAGRAMM

Beim Aufruf einer Anwendung mit einer MVC-Architektur wird die Anzeige von der View generiert. Die View wiederrum definiert, was auf dem Bildschirm konkret angezeigt werden soll. Wenn der Anwender mit der Anwendung interagiert, kommuniziert er mit dem Controller.

Funktionsweise von MTV

Laut der offiziellen Dokumentation von Django ist es eher eine MTV-Architektur. Der Controller wird weggelassen, da die meisten Aktionen von den Models, den Templates und den Views gehandhabt werden. Der Controller ist trotzdem vorhanden und wird in diesem Fall durch das Framework abgebildet.

Model

Das Model beinhaltet alles, was mit Daten zu tun hat. Das bedeutet, welche Klassen es gibt, welche Felder und Methoden jede Klasse hat, wie diese Felder validiert werden und wie auf diese zugegriffen wird. Definiert werden die Models in der models.py im Unterverzeichnis der Applikation.

Template

Ein Template ist vergleichbar mit den klassischen Views. Hier wird definiert wie etwas wiedergegeben werden soll. Templates sind meist HTML-Seiten, die mit Platzhaltern versehen sind. Diese Platzhalter werden mit Werten, die von der View übergeben werden, gefüllt. Templates enthalten sehr wenig Programmlogik.

View

Die View enthält die Logik, hier wird bestimmt welche Daten angezeigt werden und welches Template dafür verwendet werden soll.

6.1 Komponenten von Django

Das Projektverzeichnis (Abb. 11) gibt einen Überblick, wo Django die zu erstellenden Dateien speichert. Die meisten Dateien werden beim ersten Erstellen eines Projektes durch Django vorgeben. Diese kann man erweitern oder zusätzliche Dateien erstellen.

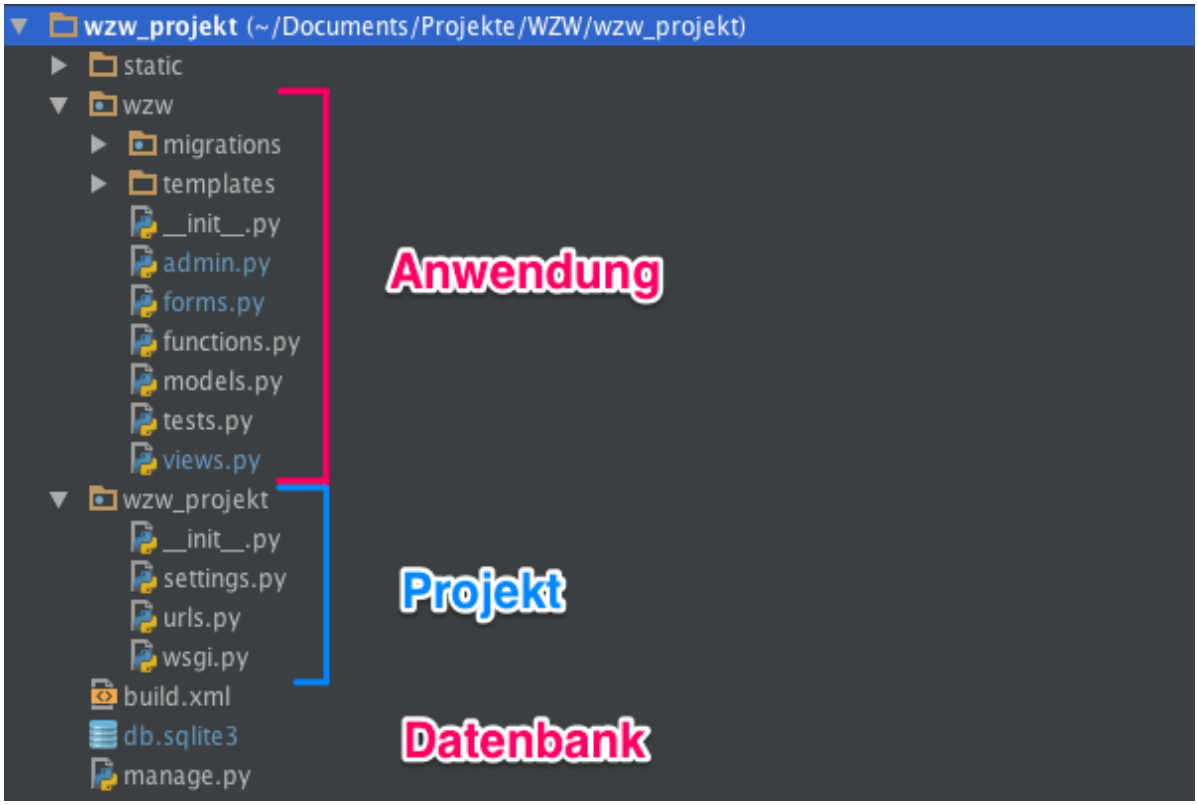

ABB. 11 DJANGO PROJEKTVERZEICHNIS

Komponentendiagramm

Das in Abb. 12 dargestellte Komponentendiagramm soll einen Überblick darüber geben, wie Django es möglich macht, dass eine Webseite auf dem Webbrowser dargestellt werden kann und welche Komponenten im Django-Framework dafür zuständig sind.

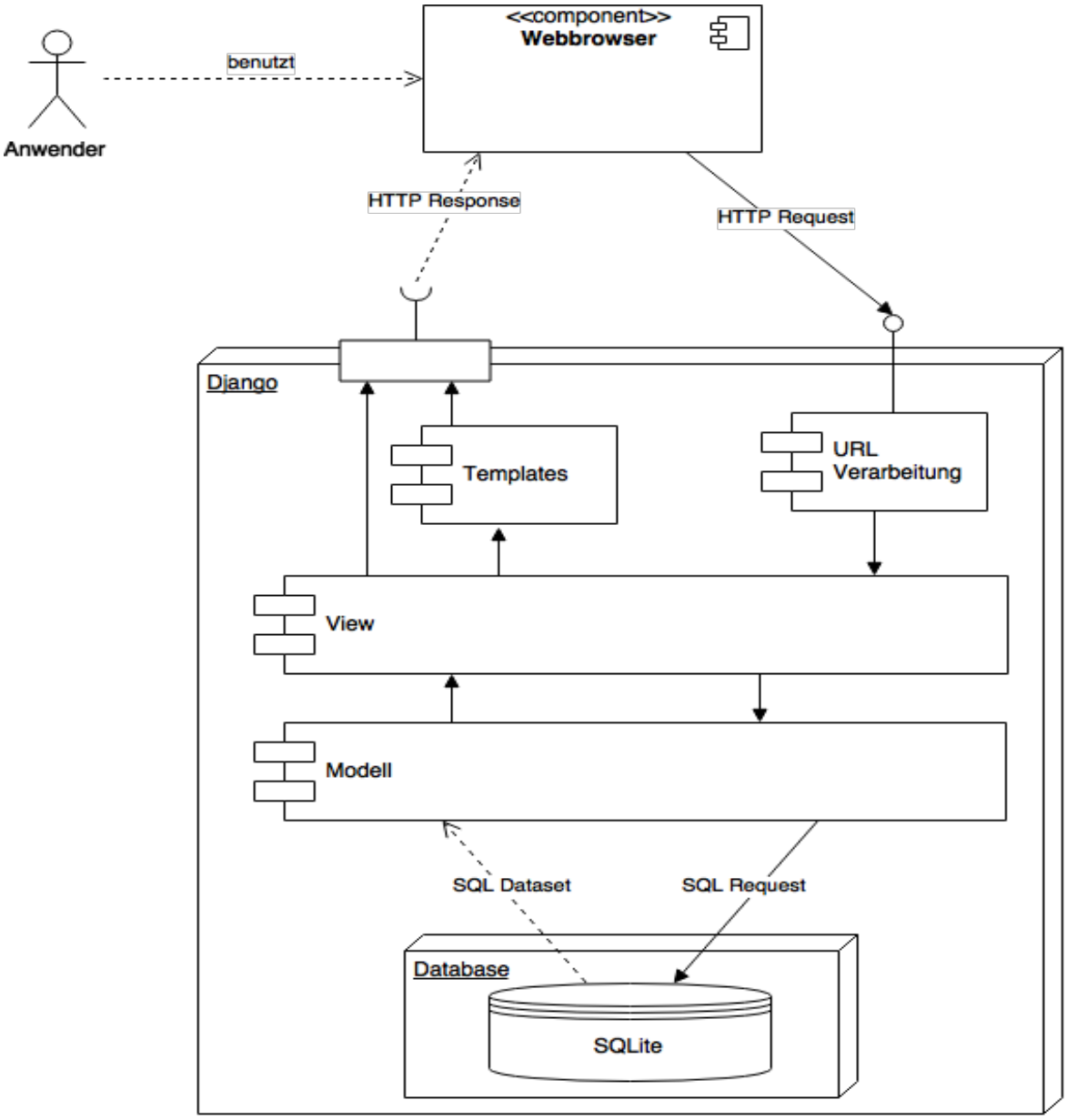

ABB. 12 KOMPONENTENDIAGRAMM

6.1.1 URL's

Die URL´s befinden sich im Projekt Verzeichnis des Django Projekts, dort sind alle URL`s hinterlegt. Es wird beschrieben, welche View verwendet werden soll. Dazu gibt es eine Startseite, sowie eine Admin Seite, die beide hinterlegt wurden. Es gibt zu jeder Klasse eine der folgenden URL´s / Views:

- Neu
- Edit
- Delete

Diese URL´s verweisen auf die entsprechenden Views.

6.1.2 Views

Die Views befinden sich im Unterverzeichnis der Anwendung. Simultan zu den URL´s gibt es Views für jede Klasse zum:

- Edit
- Delete
- New
- Anzeige aller Personen und Ausgaben

Alle Views, die innerhalb einer Gruppe aufgerufen werden, prüfen zuerst, ob die Gruppe existiert, wenn nicht, wird ein 404 HTTP Fehler zurückgeworfen. Änderungen werden nur mit POST Request übergeben, dabei werden die Daten validiert.

Grundsätzlich sieht der Aufbau einer View wie folgt aus:

- GET Request
  - Laden von Daten und Übergabe von Formularen
- POST Request
  - Überprüfen was für ein Formular ausgeführt wurde (z.B. Gruppe ändern oder löschen)
  - Daten validieren
  - Aktion ausführen
  - Zurück auf die Hauptseite mit HTTP Redirect

6.1.3 Model

Das Model befindet sich im Verzeichnis der Anwendung. Aus den Models wird die Datenbank generiert. Models werden meist zum Validieren der Daten benutzt. In unserem Projekt existieren drei Models: Gruppen, Personen und Ausgaben. Besonderheiten von Models:

- Gruppen
  - Die "speichern" Methode wurde überschrieben, um den Token einzufügen
  - Die "löschen" Methode wurde überschrieben, um alles zu löschen, was zu einer Gruppe gehört
- Personen
  - Es gibt eine Methode, die beim Abfragen des Objekts ein Report erstellt

6.1.4 Templates

Die Templates befinden sich im Unterverzeichnis "Templates" der Anwendung. Es sind sogenannte HTML Seiten, die mit Platzhaltern versehen sind. Platzhalter werden mit Werten gefüllt, die von der View übergeben werden.

6.1.5 Formulare

Die Formulare befinden sich in der Datei forms.py im Unterverzeichnis der Anwendung (Abb. 11). Es gibt zwei Möglichkeiten Formulare zu definieren:

- Formular anhand eines Models
  - Erbt alle Felder des Models (der Klasse)
  - Es können Felder ausgeblendet / hinzugefügt oder deaktiviert werden
- Formular erstellen
  - Definition ähnlich zu Model

6.2 MTV Erklärt am Beispiel: "Neue Gruppe anlegen"

Aufruf der Startseite

Der Anwender ruft die Seite im Webbrowser auf. Als erstes prüft Django die urls.py. Wird eine passende URL gefunden, wird die ihr zugeordnete View aufgerufen.

```
url(r'^$', IndexView.as_view()),
```

ABB. 13 AUSSCHNITT AUS DER URLS.PY, AUFRUF DER STARTSEITE

Beim Aufruf der Startseite (http://127.0.0.1:8000/) leitet der passende reguläre Ausdruck (Abb. 13) den HTTP-Aufruf an die IndexView-Klasse weiter.

```
...
class IndexView(View):
    @staticmethod
    def get(request):

        form_open_group = OpenGroupForm()
        # TODO message uebergeben
        return render(request, 'Wzw/index.html', {'form_open_group': form_open_group})
...
```

ABB. 14 AUSSCHNITT AUS DER VIEWS.PY, ZEIGT DEN AUFRUF DER STARTSEITE

Die Views unterscheiden zwischen GET und POST Aufrufen. Ein einfacher Aufruf der Startseite erfolgt per GET (Abb. 14). Danach wird das Formular für die Startseite geladen. Formulare werden in der forms.py definiert. Anschließend wird die HTTP-GET-Anfrage und die beiden Formulare an das Template index.html übergeben.

```
<div class="col-md-4">
    <h2>Gruppe erstellen</h2>

    <form id="new_group" method="post">
        {% csrf_token %}
        <input class="btn btn-lg btn-success" type="submit" value="Los geht's" name="new_group"/>
    </form>
</div>
```

ABB. 15 AUSSCHNITT AUS DEM TEMPLATE INDEX.HTML

Dort werden die Formulare von Django gerendert und an den Webbrowser übergeben. Um Formulare verwenden zu können, muss ein CSRF-Token eingetragen werden. Das soll verhindern, dass jede Webseite POST Anfragen an diese Webseite schicken kann. Abschließend wird die Webseite auf dem Browser des Anwenders wiedergegeben.

Erstellen einer Neuen Gruppe

Um eine neue Gruppe anzulegen, muss auf der Starseite der Link zum Erstellen einer neuen Gruppe ausgewählt werden "Gruppe erstellen". Dies ist das Formular aus der Abb. 15. Beim Drücken des Buttons wird ein POST-Request auf der Startseite erzeugt.

```
class IndexView(View):
    @staticmethod
    def get(request):

        ...
    @staticmethod
    def post(request):
        if 'new_group' in request.POST:
            group = Group.objects.create()
            # TODO message uebergeben
            return HttpResponseRedirect('group/' + group.token + '/group/new')

...
```

ABB. 16 AUSSCHNITT AUS DER VIEWS.PY, NEUE GRUPPE ERSTELLEN

Die URL-Verarbeitung von Django leitet die Anfrage wieder an die IndexView weiter. Dieses Mal wurde aber ein POST-Request übergeben. Da es mehrere Formulare und damit verschiedene Möglichkeiten der Verarbeitung geben kann, haben wir noch eine IF-Schleife implementiert. Wie in Abb. 15 zu sehen ist, wird der Name ‚new_group' übergeben. Deshalb wir die Schleife in Abb. 16 betreten. Als erstes wird eine neue Gruppe erstellt, dafür wird ein neues Objekt vom Typ "Gruppe" erzeugt.

```
class Group(models.Model):
    name = models.CharField(max_length=32, blank=True, default='')
    description = models.CharField(max_length=128, blank=True, default='')
    token = models.CharField(max_length=19, editable=False, unique=True)
    lastLogon = models.DateField('Last Logon', auto_now=True, blank=False)

    def __str__(self):  # __unicode__ on Python 2
        return self.token

    def save(self, *args, **kwargs):
        if not self.pk:
            self.token = create_token()

        super(Group, self).save(*args, **kwargs)

    def delete(self, using=None):
        person = Person.objects.filter(group=self)
        person.delete()

        expense = Expense.objects.filter(group=self)
        expense.delete()

        super(Group, self).delete()
```

ABB. 17 AUSSCHNITT AUS MODELS.PY, DEFINITION FÜR EINE GRUPPE

Die Definition einer Gruppe ist in Abb. 17 zu sehen. Als erstes werden alle Felder definiert, dabei ist deutlich zu sehen, dass nur ein Feld ausgefüllt werden muss. Anschließend werden die drei Standardfunktionen zum Speichern, Löschen und Ausgeben überschrieben.

Wenn das Objekt "Person" aufgerufen wird, gibt es seinen Token zurück.

Beim Speichern wird überprüft, ob das Objekt einen Primärschlüssel hat, wenn nicht, wird die Funktion „create_token“ aufgerufen (Abb. 18) und ein Token für die Gruppe erstellt (ein Primärschlüssel ist nicht vorhanden wenn das Objekt noch nicht gespeichert wurde). Anschließend wird die ursprüngliche "Speichern"-Methode aufgerufen. Beim Löschen einer Gruppe werden zuerst alle zugehörigen Personen und anschließend alle zugehörigen Ausgaben gelöscht. Danach wird wieder die ursprüngliche Methode zum Löschen der Gruppe aufgerufen.

Nachdem das Objekt "Gruppe" erstellt wurde, wird mit dem Token die Seite "Neue Gruppe" aufgerufen. Dort kann jetzt ein Name und eine Beschreibung für die eben erstellte Gruppe vergeben werden.

```
def create_token():
    unique = True
    while unique:
        token = ''
        for x in range(0, 4):
            token += ''.join(random.choice(string.ascii_uppercase + string.digits) for _ in range(4))
            if x <= 2:
                token += '-'
        unique = token_existing(token)

    return token
```

ABB. 18 AUSSCHNITT AUS FUNCTIONS.PY, GENERIERUNG DES TOKENS EINER GRUPPE

7. Zukunft des Projekts

Nachdem der erste Prototyp entwickelt wurde und das Feedback dazu auch ausgewertet worden ist, soll das Projekt auf jeden Fall „Live geschaltet“ werden. Ziel ist es, dass Projekt weiterzuentwickeln und nach der Fertigstellung bei GitHub öffentlich verfügbar zu machen. Außerdem soll eine lauffähige Version des Projekts unter „wzw.stiki.de“ gehostet werden. Weitere optionale Anforderungen sollen dem Projekt hinzugefügt werden, um eine höhere Usability für die Anwender zu gewährleisten. Diese Anforderungen sollen mit dem Feedback der Anwender erschlossen werden. Dazu wird ein sogenanntes „Feedback Fenster“ im Projekt erstellt, das soll ermöglichen, dass jeder seine Wünsche und Ideen direkt an das Entwicklerteam senden kann.

8. Quellen

- https://en.wikipedia.org/wiki/Model-view-controller (03.07.15)
- https://docs.djangoproject.com/en/1.8/faq/general (03.07.15)
- http://www.djangobook.com/en/2.0/chapter05.html#the-mtv-or-mvc-development-pattern (04.07.15)
- https://github.com/ (06.07.15)
- https://en.wikipedia.org/wiki/Git_(software) (06.07.15)
- https://en.wikipedia.org/wiki/GitHub (06.07.15)
- http://getbootstrap.com/ (06.07.15)
- https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework) (06.07.15)
- https://de.wikipedia.org/wiki/Django_(Framework) (06.07.2015)
- https://de.wikipedia.org/wiki/Python_(Programmiersprache (06.07.2015)
- https://de.wikipedia.org/wiki/PyCharm (06.07.2015)
- https://www.draw.io/ (06.07.15)
- http://freemind.sourceforge.net/wiki/index.php/Main_Page
- https://www.google.de/search?hl=de&q=Repository&gws_rd=ssl#hl=de&q=framework (06.07.2015)
- http://www.sqlite.org/about.html (07.07.2015)
- https://de.wikipedia.org/wiki/SQLite (07.07.2015)
- Skript Softwaretechnik, Professor Dr. Edlich (09.07.2015)

9. Glossar

WZW

- Wer zahlt was

Excel

- Programm zur grafisch orientierte Tabellenkalkulation

IT

- Oberbegriff für die Informations- und Datenverarbeitung

Tool

- kleine Anwendungssoftware oder Dienstprogramm

Report

- Journalistische Darstellungsform ist ausführlicher und oft anspruchsvoller als ein Bericht

Web-Applikation

- Anwendung die im "Internet" ausgeführt wird
- Darstellung meist im Browser

Release

- In der Softwaretechnik, eine Version oder ein Stand einer Software

Django

- In Python geschriebenes quelloffenes Web-Applikation Framework

Token

- Eine Zeichenfolge die zur Zutrittskontrolle verwendet werden kann

Container

- abstraktes Objekt, das Elemente des gleichen Typs speichert

SSL-Verschlüsselung

- Transport Layer Security, ist ein hybrides Verschlüsselungsprotokoll zur sicheren Datenübertragung

Freie Lizenz

- Nutzungslizenz, die die Nutzung, Weiterverbreitung und Änderung urheberrechtlich geschützter Werke erlaubt

GitHub

- Webbasierter Hosting-Dienst für Software-Entwicklungsprojekte

Use-Case

- Anwendungsfall

Sysadmin

- verwaltet Computersysteme auf der Basis umfassender Zugriffsrechte auf das System

Sequenzdiagramm

- Verhaltensdiagramm

Klassendiagramm

- Strukturdiagramm der Unified Modeling Language (UML) zur grafischen Darstellung (Modellierung) von Klassen

Struktogramm

- ein Diagrammtyp zur Darstellung von Programmentwürfen im Rahmen der Methode der strukturierten Programmierung

GUI

- Grafische Benutzeroberfläche oder auch grafische Benutzerschnittstelle

GIT

- freie Software zur verteilten Versionsverwaltung von Dateien

Optimistic-Locking

- regelt Änderungskonflikte im letzten Augenblick

Repository

- Ein Repository, ist ein verwaltetes Verzeichnis zur Speicherung und Beschreibung von digitalen Objekten

Branche

- Zweig

Python

- Universelle, üblicherweise interpretierte höhere Programmiersprache

Framework

- Ein *Framework* (englisch für Rahmenstruktur) ist ein Programmiergerüst

Backend

- Bezeichnet den Teil einer Software-Anwendung, die auf dem Server läuft und die Daten verwaltet

Frontend

- Programm, das die Daten auf dem Client darstellt

HTML

- textbasierte Auszeichnungssprache zur Strukturierung digitaler Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten

CSS

- Computersprache für die Gestaltung digitaler, vorwiegend Web-basierter Dokumente

Styleguide

- beschreibt, wie bestimmte Elemente eines Druckerzeugnisses oder einer Website zu gestalten sind

Datenbank

- System zur elektronischen Datenverwaltung

SQLite

- Programmbibliothek, die ein relationales Datenbanksystem enthält

SQL

- Datenbanksprache zur Definition von Datenstrukturen in relationalen Datenbanken sowie zum Bearbeiten (Einfügen, Verändern, Löschen) und Abfragen von darauf basierenden Datenbeständen

Syntax

- Regelsystem zur Kombination elementarer Zeichen (z. B. Wörter) zu zusammengesetzten Zeichen (z. B. Wortfolgen) in natürlichen oder künstlichen Zeichensystemen

Server

- Computerprogramm oder ein Computer für den Zugriff auf eine zentrale Ressource oder Dienst in einem Netzwerk

Mindmaps

- beschreibt eine kognitive Technik, die man zum Erschließen und visuellen Darstellen eines Themengebietes, zum Planen oder für Mitschriften nutzen kann

MVC

- MVC ist eine Architektur Muster und steht für Model, View und Controller

Controller

- verarbeitet die Daten der HTTP-Requests und stößt schließlich die Erstellung eines neuen Views an

Model

- Objekt innerhalb einer strukturierten Softwarearchitektur

Template

- Vorlagen, die mit Inhalt gefüllt werden können

View

- logische Tabelle in der Datenbanktechnik

Programmlogik

- interne Logik eines Programmes

Komponentendiagramm

- bestimmte Sicht auf die Struktur des modellierten Systems

URL

- identifiziert und lokalisiert eine Ressource

HTTP

- ist ein zustandsloses Protokoll zur Übertragung von Daten auf der Anwendungsschicht über ein Rechnernetz

Reguläre Ausdrücke

- ist in der theoretischen Informatik eine Zeichenkette, die der Beschreibung von Mengen von Zeichenketten mit Hilfe bestimmter syntaktischer Regeln dient

CSRF

- ist ein Angriff auf ein Computersystem, bei dem der Angreifer eine Transaktion in einer Webanwendung durchführt

Primärschlüssel

- ein Schlüssel dient in einer relationalen Datenbank dazu, die Tupel einer Relation eindeutig zu identifizieren

Hosten

- ist die Unterbringung von Internetprojekten, die sich in der Regel auch öffentlich durch das Internet abrufen lassen

Usability

- ist die Arbeit hin zu leicht verständlicher und schnell benutzbarer Software unter den gebotenen technischen Möglichkeiten und unter der Einhaltung definierter bzw. empirisch entstandener Standards und Styleguides