TUDas - Organisationsapp der Technischen Universität Darmstadt

Ergebnis des E-Learning Projektpraktikums WS18-19

Benedikt Lins (1799381) und Stefan Thaut (1800351)
Fachbereich 20 - Informatik

29. März 2019

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhaltsverzeichnis

1 Motivation

Jeder Student oder der, der einmal einer gewesen ist, kennt es: Du kommst das erste Mal in die Uni und hast keine Ahnung, was zu tun ist. Alles ist komplett neu und Du bist froh, wenn Du die Räume findest, in denen Du Dich laut Deinem Willkommensbrief einfinden sollst. Selbst wenn Du dann nach ein paar Tagen herausgefunden hast, welche Straßenbahn Dich in die Uni bringt, bist Du nach ein paar Wochen immer noch ratlos, welche Dokumente du gegebenenfalls wo nachreichen musst, wo Du Dich zu welchen Veranstaltungen und Prüfungen anmelden sollst und in welchem Kellerraum nun dieser Treffpunkt Mathe stattfindet.

Diesem Problem soll sich die (Android-)App *TUDas* widmen. Die App soll als Organisationsplattform für Studenten verschiedener Fachbereiche während ihres ersten Semesters dienen. Dabei soll TUDas vorhandene Plattformen, wie *Moodle*[1] oder die *OAPP* [2] unterstützen und nicht ersetzen.

Diese Dokumentation soll einerseits die Funktionalitäten der App TUDas festhalten und andererseits als Übersicht für zukünftige Entwickler dienen. Kapitel **??** behandelt die technischen Spezifikationen der App. In Kapitel 3 werden die Features beschrieben, die von der App in ihrer jeweils aktuellen Version angeboten werden. Und Kapitel 4 beinhaltet eine Dokumentation des Codes.

[1] www.moodle.tu-darmstadt.de [zuletzt aufgerufen: 23.11.2018]

[2] www.oapp.tu-darmstadt.de [zuletzt aufgerufen: 23.11.2018]

2 Einrichtung

Für das Funktionieren der App muss auf Seiten der Entwickler, abgesehen von der Verbreitung der App, lediglich ein Webserver mit zwei Frameworks aufgesetzt werden. Für das Webinterface *Edidas* wird das Framework cakePHP[1] in der Version 3.7 verwendet. Die REST-API wird über das Slim-Framework[2] realisiert. Ist der Webserver entsprechend der in den Fußnoten genannten Anleitungen eingerichtet, so muss lediglich der gelieferte Code auf den Webserver kopiert werden.

[1] https://book.cakephp.org/3.0/en/installation.html [zuletzt aufgerufen: 28.03.2019]

[2] http://www.slimframework.com/docs/v3/start/installation.html [zuletzt aufgerufen: 28.03.2019]

3 Funktionalitäten

Dieses Kapitel beschreibt die Funktionalitäten, die die App in ihrer jeweils aktuellen Version anbietet. Dabei sollen die jeweiligen funktionalen sowie nicht-funktionalen Anforderungen aufgezeigt werden und nicht die programmiertechnische Umsetzung, die dann in Kapitel 4 folgt.

3.1 Campus-Navigation

Die TU Darmstadt belegt Gebäude und Räume, die über das gesamte Stadtgebiet verteilt sind. Daher kann es schwer fallen, stets zu wissen, wohin man gehen muss, um pünktlich zu dem Raum zu gelangen, in dem die nächste Veranstaltung stattfindet. Der Nutzer soll deswegen die Möglichkeit haben, zu dem entsprechenden Ort einer Veranstaltung navigieren zu können.

3.2 Stundenplan

Im Stundenplan werden die selbst erstellten Termine des Nutzers angezeigt. Hier können grundsätzlich zwei Typen von Terminen unterschieden werden:

- Einfache Termine
- Wiederholende Termine

Die Charakteristik der einfachen Termine impliziert eine einmalige Anzeige im Stundenplan an einem bestimmten Datum. Daher ist es sinnvoll, den Stundenplan datumsabhängig anzuzeigen. Da die Bildschirmbreite des Smartphones in der vertikalen Ausrichtung deutlich begrenzt ist, muss auch die Anzahl der angezeigten Tage beschränkt werden. Diese Beschränkung kann der Nutzer selbst vornehmen und zwischen zwei und fünf anzuzeigenden Tagen wählen. Weniger als zwei Tage ist nicht sinnvoll, da sich dies mit der Funktionalität überschneiden würde, die in 3.3 beschrieben ist. Bei mehr als fünf anzuzeigenden Tagen ist die Grenze der visuellen Darstellbarkeit schnell erreicht. Darüber hinaus besteht eine Werktagswoche in der Regel aus fünf Tagen. Der ausgewählte Zeitraum wird in einer Kopfzeile über den Tageslisten angezeigt. Auf der linken Seite sind in vertikaler Ausrichtung die Uhrzeiten in stündlichem Abstand untereinander angeordnet. Dabei beginnt der Stundenplan oben mit der maximalen ganzen Stunde, die kleiner als oder gleich allen Anfangsuhrzeiten von Terminen im angezeigten Zeitraum ist und endet unten mit der minimalen ganzen Stunde, die größer als oder gleich allen Enduhrzeiten der gleichen Termine sind.

3.3 Variabel einsetzbare, moderierbare Aufgabenlisten

Da Erstsemesterstudenten oft nicht wissen, welche Termine und Veranstaltung für sie wichtig sind, ist es sinnvoll, diese Termine bereitzustellen. Dies setzt voraus, dass solche Termine von einer externen Institution erstellt und editiert werden. Eine solche Funktionalität benötigt einen zentralen Speicherort, wo die erstellten Termine gespeichert und von der App abgerufen werden können.
Auf der anderen Seite gibt es spezifische Termine, die nicht jeder Student angezeigt bekommen soll, um nicht mit Informationen überflutet zu werden. Hierfür soll eine Art Abonnement-System etabliert werden, mittels dem Studenten verschiedene Labels abonnieren können, unter denen Informationen zu dem gleichen Thema zusammengefasst werden. Diese Labels können vollkommen frei gewählt werden und können beispielsweise für verschiedene Lehrveranstaltungen (z.B. eine Vorlesung) aber auch für die Termine der Orientierungsphase stehen.
Um solche Termine zu erstellen und zu bearbeiten, wird ein Webinterface mit einer zugrundeliegenden Datenbank verwendet, welches über einen herkömmlichen Browser erreichbar ist. Fachschaften und andere Institutionen benutzen oftmals schon eine Software für die Verwaltung ihrer Termine. Daher soll das Webinterface auch eine Importfunktion besitzen, die das Hochladen von Kalenderdateien ermöglicht und für verschiedene Dateiformate erweiterbar ist.

Die Termine von abonnierten Terminlisten werden dem Nutzer in einer Tagesansicht des jeweils aktuellen Tags angezeigt, welche gleichzeitig auch die Startseite der App darstellt. Zusätzlich enthält diese Ansicht ebenfalls alle privaten Termine, die der Nutzer im Stundenplan erstellt hat. Die Termine werden hier unabhängig von ihrer Länge in gleich großen Blöcken angezeigt und nach ihrem Startzeitpunkt sortiert.
Viele Nutzer verwenden üblicherweise eine andere Applikation zur Verwaltung ihres Kalenders. Daher werden für alle angezeigten Termine in der Tagesansicht überprüft, ob sich diese mit einem Termin einer anderen installierten Kalenderapp überschneidet. Ist dies der Fall, wird dies in dem Block des Termins durch ein Icon angezeigt.

4 Dokumentation der Implementierung

4.1 Campus-Navigation

Um die in Abschnitt 3.1 beschriebene Funktionalität erfüllen zu können, kann für einen Termin ein Ort gespeichert werden. Dabei ist es dem Nutzer selbst überlassen, was er hierbei eingibt. Für die Realisierung der Navigation haben sich verschiedene Möglichkeiten geboten:

- Nutzen einer Karten-API, um sowohl den kürzesten Weg zu finden als auch den Weg auf einer Karte anzuzeigen
- Crawlen von Kartenmaterial und der Entwurf eines eigenen Routing-Algorithmus
- Weiterleitung an eine Navigationsapp

Die meisten APIs, die einen Navigationsalgorithmus bereitstellen, sind entweder kostenpflichtig oder begrenzen die Anzahl der Zugriffe für einen bestimmten Zeitraum auf ein Maß, das für den erwarteten Nutzungsumfang von TUDas unzureichend ist. Aus Gründen der Kosteneffizienz der App ist die Nutzung einer Karten-API daher nicht möglich.
Eine weitere Möglichkeit, die Campus-Navigation zu realisieren, ist das Crawlen, also das Beschaffen von frei zugänglichem Kartenmaterial wie zum Beispiel von Openstreetmap. Hier wird Kartenmaterial in einem verarbeitungsfähigen Format bereitgestellt. Auf dieser Grundlage kann dann ein Algorithmus wie etwa der Dijkstra entwickelt, bzw. angewandt werden, um eine kürzeste Route von der aktuellen Position des Nutzers bis zum angegebenen Ziel zu berechnen. Auch dieser Ansatz muss aus verschiedenen Gründen abgelehnt werden. Einerseits enthalten die verfügbaren Kartendaten sehr viele Informationen, die für die Anwendung der App unnötig sind, weshalb ein hoher Aufwand aufgebracht werden muss, um die Daten zu bereinigen. Darüber hinaus müssten die Daten ständig aktualisiert werden, um mögliche Straßensperrungen oder Änderungen des Kartenmaterials einbinden zu können. Andererseits muss die Korrektheit des Routing-Algorithmus gewährleistet werden, was im Umfang dieser Implementierung nicht möglich ist.
Daher wird im Umfang dieses Projekts die Aufgabe der Navigation an eine externe App abgegeben. Hier bietet sich die Navigationsapp von Google "Google Mapsän. Diese App ist in der Regel auf allen Android Smartphones installiert oder verfügbar. Darüber hinaus stellt diese App eine API bereit, die Navigationsanfragen von anderen Apps begünstigt.

4.2 Stundenplan

Wie in Abschnitt 3.2 beschrieben, müssen grundlegend zwei Arten von Terminen unterschieden werden. Um zukünftig auch wiederholende Termine unterstützen zu können, ist die Modellierung im Vorhinein dahingehend angepasst. Ein wiederholender Termin hat im Vergleich zu einem einfachen Termin zusätzlich ein Startdatum, an dem die Wiederholung des Termins beginnt und ein Enddatum, wann die Wiederholung endet. Notwendig ist ebenfalls eine Wiederholungsvorschrift. Möchte man nun eine Teilmenge von wiederholenden Terminen löschen, so ist es notwendig, dass die Termine nicht zur Laufzeit auf Grundlage der Wiederholungsvorschrift berechnet werden sondern die Termine schon gespeichert wurden. Ansonsten wäre es nicht möglich zu spezifizieren, welche Termine von der Wiederholungsvorschrift ausgeschlossen werden sollen. Die wiederholenden Termine unterscheiden sich dann nur noch in den Daten. Der Titel und die Beschreibung beispielsweise sind für alle wiederholenden Termine jeweils gleich. Um Redundanzen zu vermeiden, werden solche Attribute von den reinen Datumsangaben getrennt. Die Klasse AppointmentContent beinhaltet die statischen Informationen einer Terminsammlung und die Klasse Appointment enthält die Datumsangaben. Die Attribute der Klassen sind im UML-Diagramm in Abbildung 4.1 zu sehen.
Für eine mathematische Beschreibung des Projekts müssen diverse Hilfsfunktionen definiert werden. Sei A die Menge aller Termine des Benutzers und $\mathbb{D}$ die Menge aller Datumsangaben, die auch den Zeitpunkt am Tag beinhalten. Die Funktion

$$\text{start} : A \to \mathbb{D} \tag{4.1}$$

liefert für einen gegebenen Termin den Startzeitpunkt. Analog liefert die Funktion

$$\text{end} : A \to \mathbb{D} \tag{4.2}$$

den Endzeitpunkt des gegebenen Termins. Für eine bestimmte Datumsangabe gibt die Funktion

$$\text{hour} : \mathbb{D} \to \{0, \ldots, 23\} \tag{4.3}$$

die Stunde der Angabe aus.

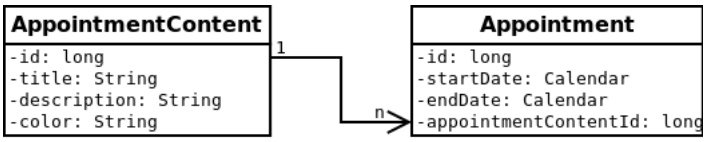

Abbildung 4.1: UML-Diagramm der Klassen AppointmentContent und Appointment

4.2.1 Modellierung der Anfangs- und Endzeitbedingung

Die in Abschnitt 3.2 beschriebene Anforderung, dass der Stundenplan mit der maximalen Stunde beginnen soll, die kleiner als die Startzeiten aller Termine im betrachteten Zeitraum ist, kann wie folgt formalisiert werden: Für ein gegebenes Start- und Enddatum $s \in \mathbb{D}$, bzw. $e \in \mathbb{D}$ ist

$$A_f = \{a \in A : s \leq a \leq e\} \tag{4.4}$$

die Menge aller Termine im gegebenen Zeitraum. Dann ist die gesuchte Stunde h, zu der der Stundenplan beginnen soll, mithilfe der Funktionen 4.1 und 4.3 wie folgt definiert:

$$h_s = \max\{h_s \in \{0, \ldots, 23\} : \forall a \in A_f : h_s \leq \text{hour}(\text{start}(a))\} \tag{4.5}$$

Analog ist die Stunde der Endzeit h_e definiert:

$$h_e = \min\{h_e \in \{0, \ldots, 23\} : \forall a \in A_f : h_e \geq \text{hour}(\text{end}(a))\} \tag{4.6}$$

4.2.2 Positionierung der Termine im Stundenplan

Die Termine sollen in einem tabellenartigen Format angezeigt werden. Dabei soll jede Spalte der Tabelle einen Tag repräsentieren. Innerhalb eines Tages sollen die Termine untereinander angezeigt werden. Programmtechnisch wird jeder Tag über ein Layout beschrieben, dem Elemente vertikal hinzugefügt werden können. Für Zeiten, zu denen kein Termin existiert, wird dem Layout ein leeres Feld hinzugefügt. Damit beschränkt sich die korrekte Positionierung der Termine auf das Berechnen der Höhen der einzelnen Elemente. Dazu wird eine Konstante L eingeführt, die die Anzahl der Pixel pro Minute angibt. Die Dauer eines Termins wird dann in Minuten berechnet und mit der Konstante multipliziert, um die entsprechende Höhe zu berechnen.

5 Codespezifische Dokumentation

In diesem Kapitel soll ein konkreter Überblick über den Code gegeben werden.

5.1 Struktur

TUDas ist ein Android-Projekt und hält sich an das MVVM-Entwurfsmuster[1]. Die Klassen in der Projektstruktur sind nach ihren Aufgaben in Packages unterteilt. Es existieren die folgenden Packages:

- adapters: Adapter zur Darstellung von Listen mit variabler Größe
- customWidgets: Eigens erstellte Komponenten zur Anzeige innerhalb von Views
- fragments
- listeners: Listener, die in mehreren Klassen referenziert werden
- localdatabase: Klassen für die lokale Persistenz über eine Room-Datenbank
- model: Model-Klassen, die in der Datenbank gespeichert werden können
- repositories: Datenrepositories, die verschiedene Datenquellen zusammenfassen
- utils: Utility-Funktionen, die in mehreren Klassen referenziert werden
- viewmodels: Viewmodels für alle Activities
- views: Eigens erstellte Views, die in Activities genutzt werden

Das Projekt beinhaltet darüber hinaus fünf Activities:

- DailyApointmentsActivity: Tagesansicht der Termine, die aus der Room-Datenbank und der zentralen Datenbank auf dem Server geladen werden; Einstiegspunkt der App
- ManageLabelActivity: Verwalten der abonnierten Listen
- NewAppointmentActivity: Interface zum Hinzufügen eines neuen Termins in den Stundenplan
- SettingsActivity: Einstellungen der App
- TimeTableActivity: Darstellung eines Stundenplans mit mehreren Tagen

Die App ist für den Export auf andere Entwicklungsumgebungen wartungsarm gestaltet und muss kaum konfiguriert werden. Die URL der REST-API, über die die Termine in der zentralen Datenbank auf dem Server abgerufen werden, kann in der Klasse *TudasServiceGenerator* im Package *Model* gesetzt und verändert werden.

[1] https://developer.android.com/jetpack/docs/guide [zuletzt aufgerufen: 28.03.2019]

6 Ausblick

In diesem Kapitel sollen zuletzt noch Vorschläge unterbreitet werden, wie das Projekt weitergeführt werden kann. Diese Vorschläge haben es teilweise aus zeitlichen Gründen nicht in die aktuelle Version geschafft, da erst einmal die Grundfunktionalitäten erstellt werden mussten.

6.1 Erstellen von sich wiederholenden Terminen

Wie in Abschnitt 4.2 beschrieben, wurde die Modellierung der Termine so gestaltet, dass sich Termine auch wiederholen können. Hier kann die Benutzerschnittstelle dahingehend angepasst werden, dass der Nutzer eine Wiederholungsvorschrift definieren kann und die Termine entsprechend dieser Vorschrift berechnet und mit dem Termininhalt assoziiert werden. Auch für die Weboberfläche von Edidas kann diese Funktionalität erstellt werden.

6.2 Wechseln des Starttages im Stundenplan durch Wischen

In der Stundenplanaktivität kann der angezeigte Zeitraum verändert werden, indem der Starttag über den Datepicker, der über die Datumszeile erreichbar ist, gewechselt wird. Ein erwartetes Verhalten im Stundenplan ist das Wechseln des Tages über eine horizontale Wischoperation. Hierfür muss das Layout der Stundenplanaktivität angepasst werden und ein eigener Gesture-Listener erstellt werden.

6.3 Tutorial

Sollte sich bei dem ersten Einsatz der App oder der Weboberfläche herausstellen, dass diese nicht selbst erklärend ist, sollte überlegt werden, eine Hilfe- oder Erklärungsfunktion in beide Systeme einzubauen. In dieser sollen dann die Funktionen der App oder der Weboberfläche erläutert werden.