TUDas - Organisationsapp der Technischen Universität Darmstadt

Ergebnis des E-Learning Projektpraktikums WS18-19

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

24. Dezember 2018

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhaltsverzeichnis

1 Motivation **2**

2 Technische Spezifikationen **3**

3 Funktionalitäten **4**
3.1 Campus-Navigation 4
3.2 Stundenplan 4
3.3 Variabel einsetzbare, moderierbare Aufgabenlisten 4

4 Dokumentation der Implementierung **5**
4.1 Stundenplan 5
4.1.1 Modellierung der Anfangs- und Endzeitbedingung 5

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 2 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 Technische Spezifikationen

Aus Zeit- und Komplexitätsgründen wird die App zunächst nur für das *Android*-Betriebssystem[1] entwickelt.

[1] https://www.android.com/ [zuletzt aufgerufen: 24.11.2018]

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

3.2 Stundenplan

Im Stundenplan sollen die Termine des Nutzers angezeigt werden. Dies sind sowohl die selbst erstellten Einträge des Nutzers als auch die Termine von abbonierten Veranstaltungen. Üblicherweise beinhaltet ein Stundenplan Einträge, die sich wöchentlich wiederholen. Fristen oder Termine in abbonierten Listen haben einen überwiegend einmaligen Charakter. Daher können hier zwei grundsätzliche Typen von Terminen unterschieden werden:

- Einfache Termine
- Wiederholende Termine

Die Charakteristik der wiederholenden 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. Zwei Tage, das heißt also der aktuelle Tag und der darauffolgende Tag, scheint eine sinnvolle Wahl zu sein, da man an diesen Tagen gegenwärtig am meisten interessiert ist. Der entsprechende Wochentag mit dem dazugehörigen Datum wird in einer Kopfzeile 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

4 Dokumentation der Implementierung

4.1 Stundenplan

Wie in Abschnitt 3.2 beschrieben, müssen grundlegend zwei Arten von Terminen unterschieden werden. 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.

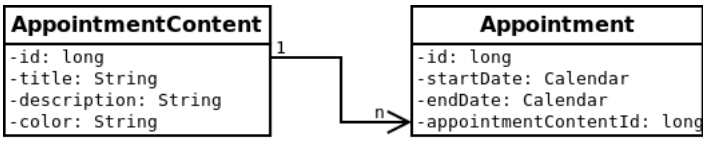

Abbildung 4.1: UML-Diagramm der Klassen AppointmentContent und Appointment

4.1.1 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:

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

4.1.2 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. Für die Anzeige der Termine eines Tages sind also folgende Schritte notwendig:

1. Laden der Termine des Tages d als geordnete Menge A_d nach ihren Anfangszeitpunkten.