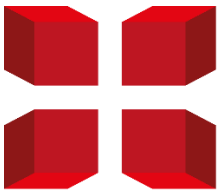
HÖHERE TECHNISCHE BUNDES-
LEHRANSTALT WIEN 16

Abteilung für Informatik für Berufstätige



HTL
WIEN
WEST

DIPLOMARBEIT

Studybuddy

Ausgeführt im Schuljahr 2020/21

Ibrahim Farghali (6BBKIF)

Joe Martin (6BBKIF)

Betreuer: DI Walter Jaich

Selbstständigkeitserklärung

Wir erklären, dass wir die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht haben.

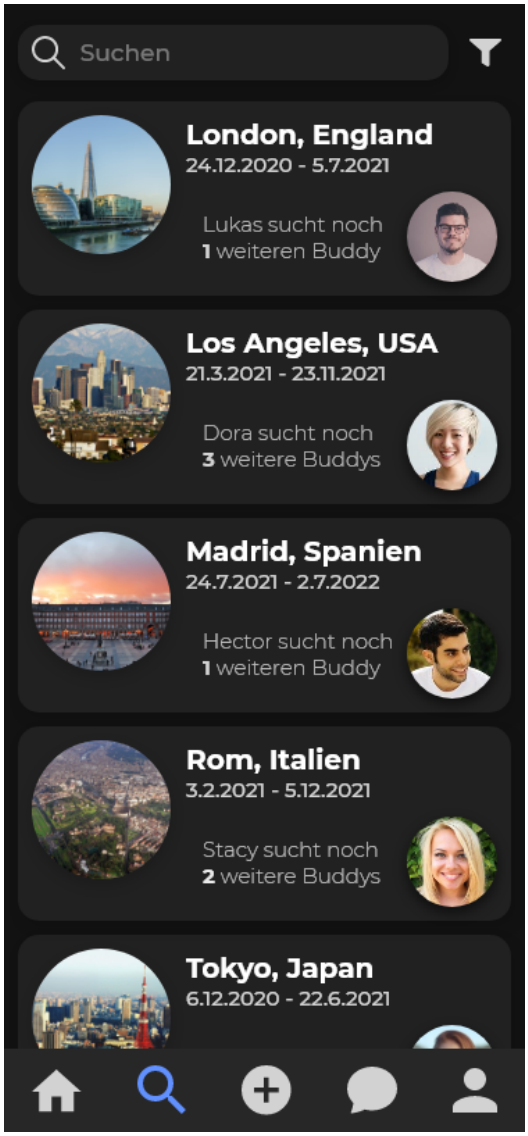
Wien, am

Ibrahim Farghali	Joe Martin

Dokumentation der Diplomarbeit

Verfasser	Ibrahim Farghali, Joe Martin
Jahrgang / Klasse Schuljahr	2020/21, 6BBKIF
Thema	Plattform für die Planung gemeinsamer Auslandssemester.
Kooperationspartner	Keine
Aufgabenstellung	Entwicklung einer Plattform für Studenten zur Planung von Auslandssemestern. Es soll eine App sowie eine mobile Webseite entwickelt werden, über welche angehende Studenten Kontakt zu anderen Studenten an verschiedenen Universitäten aufnehmen können, um Erfahrungen austauschen können.
Realisierung	***Client:*** Entwicklung einer IOS App mit Xcode / Swift, sowie einer mobilen Webseite mittels der Javascript-Bibliothek „React“. ***Server:*** Das Backend wurde mit dem open Source Framework „Spring“ realisiert, welches die Daten in einer Maria-DB Datenbank abspeichert.
Ergebnis	Der Entwurf und das Design einer benutzerfreundlichen Benutzeroberfläche stand am Anfang der Entwicklung. Der Client (iOS App bzw. die mobile Webseite) kommuniziert über eine Rest API -Schnittstelle zum Server. Zusätzlich wurde ein Benutzer-Account-System mittels dem JWT (JSON-Web-Token) Verfahren realisiert.

Teilnahme an Wettbewerben	keine	
Möglichkeiten der Einsicht-nahme der Arbeit	Bibliothek, Abteilung BI	
Abgabevermerk	Datum:	Übernommen von:
Approbation	Datum:	Prüfer: DI Walter Jaich
	Datum:	Abteilungsvorstand: Prof. DI Andreas Heinbach

Studybuddy App	

Kurzfassung

Das Ziel dieses Projekts war es eine Plattform zu entwickeln, welche die Planung gemeinsamer Auslandssemester vereinfacht. Die Hauptzielgruppe sind Schüler und Studenten mit dem Drang zu Reisen und die Welt zu erkunden, oder einfach nur mit dem Ziel eine neue Sprache zu erlernen.

Für die Umsetzung einer solchen Plattform gibt es einige Möglichkeiten. Von der Wahl der zu verwendenden Technologien bis hin zur Wahl der richtigen Softwarearchitektur, jede Entscheidung muss sorgfältig abgewogen werden, um das bestmögliche Ergebnis liefern zu können. Um das Ziel eines voll funktionstüchtigen Prototypen im Zeitrahmen eines Semesters zu erreichen, wurden hauptsächlich Technologien gewählt, die im Laufe der Ausbildung gelehrt wurden.

Die Planung spielte im Laufe der Arbeit die wichtigste Rolle, denn ohne umfangreiche Entwürfe jeder einzelnen Komponente des Projekts, wäre die Umsetzung nahezu unmöglich gewesen.

Das Endergebnis dieser Arbeit ist in gewisser Weise die Zusammenführung aus so gut wie allen zuvor angeeigneten Disziplinen der Informatik. Der Großteil der Arbeit ist dem Bereich Softwareentwicklung zuzuschreiben, welche jedoch in den verschiedensten Formen auftritt.

Abstract

The goal of this Project was the development of a platform that simplifies the process of planning joint semesters abroad. The main target audience are students with the urge to travel and explore the world, or simply with the goal to learn a new language.

Such a platform can be implemented in a variety of ways. From the choice of which technologies to use, to choosing the right software architecture, every decision involves trade-offs that need to be evaluated carefully to deliver the best outcome possible. To achieve the goal of a fully functional prototype in the time frame of one semester, mainly technologies that were taught during the course were chosen.

Planning played the most important role during the development of the project, as without extensive drafts of each component of the project, implementation would have been nearly impossible.

The outcome of this project is, in a way, the combination of just about all the previously acquired disciplines of computer science. The majority of the work can be attributed to the area of software development, which, however, occurs in the most diverse forms.

Inhaltsverzeichnis

1 Einleitung

Ausgearbeitet von Joe Martin

Als Student ein Auslandssemester zu planen ist nie einfach, von der Suche nach Unterkunft bis hin zum Finden sozialer Kontakte und möglicherweise dem Lernen einer neuen Sprache, alles stellt eine Herausforderung dar. Die wichtigsten Aspekte der Planung eines Auslandssemesters werden mit der Plattform „Study Buddy“ vereinfacht.

1.1 Grundidee

Da die Planung eines Auslandssemesters einige Schwierigkeiten mit sich bringen kann, war die grundsätzliche Idee eine Plattform zu entwickeln die Abhilfe schaffen kann. Dies wird dadurch erreicht, dass man Benutzern die Möglichkeit gibt Gleichgesinnte zu finden wodurch diese Aufgabe als Team erledigt werden kann. Gleichzeitig wird dadurch die Sorge genommen vor Ort womöglich noch niemanden zu kennen.

Ein typischer Anwendungsfall könnte wie folgt aussehen: Ein Student, der das kommende Semester im Ausland studieren möchte und auch schon eine Stadt in nähere Betrachtung gezogen hat, registriert sich auf der Plattform. Weil die Zeit fehlt, um umfangreich zu recherchieren und zu planen kann man nun nach der gewünschten Stadt filtern und nach einem passenden Inserat suchen. Beim Finden eines passenden Inserats, kann sich der Benutzer mit dem Inserierenden austauschen, eigene Ideen einbringen und diese im besten Fall auch umsetzen.

Für den Fall, dass der kommende Aufenthalt bereits geplant ist oder es kein Inserat in der gewünschten Stadt oder dem gewünschten Zeitraum gibt, besteht die Möglichkeit selbst eine Anzeige auf die Plattform zu stellen.

1.2 Gliederung

Die Diplomarbeit wurde im Sinne der wichtigsten Aspekte der Entwicklung eines modernen Webservice und den dazugehörigen Benutzeroberflächen gegliedert. Dies beinhaltet das Design der Benutzeroberfläche, die Datenbankentwicklung, die Software in allen Aspekten und di e Sicherheit.

Dabei behandelt das Kapitel „User Interfaces (UI) / User Experience (UX)“ die gestalterische Umsetzung der Benutzeroberfläche. Das Kapitel „Datenbank“ behandelt die Wahl, Planung, Umsetzung und Erweiterung einer Datenbank. Das Kapitel „Software“ beschäftigt sich mit der Entwicklung von Server- sowie Clientsoftware. Das

Kapitel „Sicherheit“ behandelt den Umgang mit sensitiven Daten und beschreibt das Benutzer-Account-System genauer.

1.3 Arbeitspakete

Die Arbeitspakete wurden mit dem Start des Projekts definiert, um klare Verantwortlichkeiten in der Entwicklung zu schaffen und somit einen effizienten Ablauf zu gewährleisten.

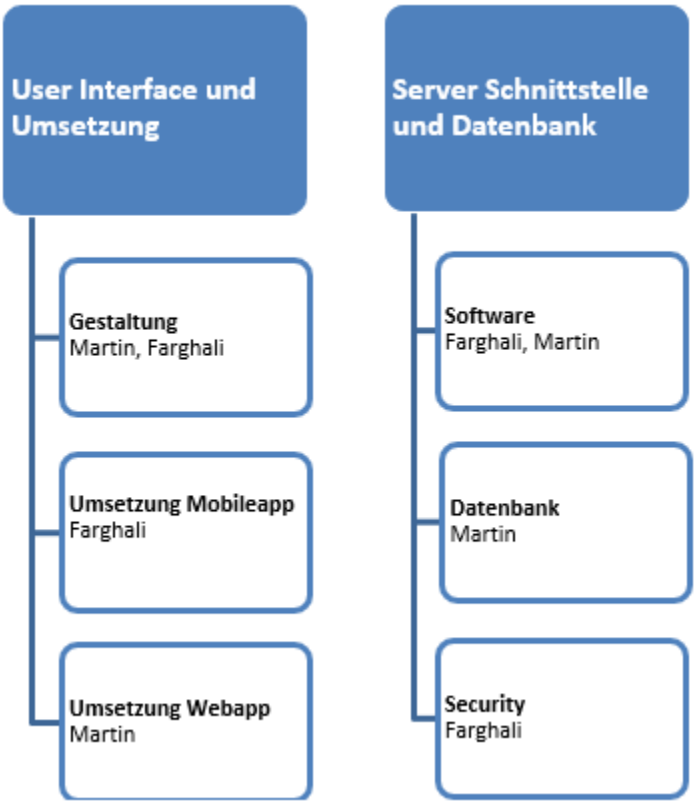
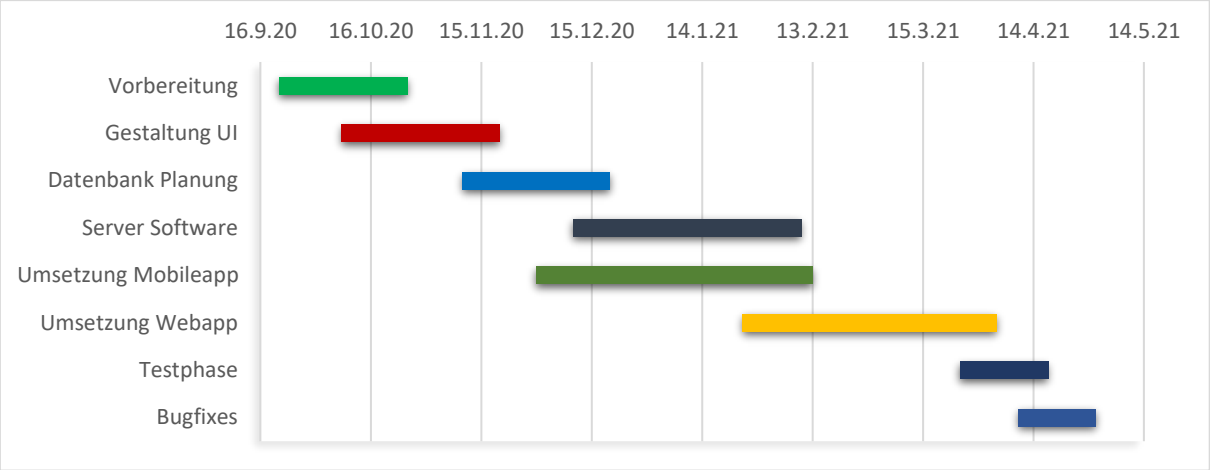


Abbildung 1: Arbeitspakete

1.4 Zeitplan

Zu Beginn der Arbeit wurde zudem ein Zeitplan über die Abarbeitung der einzelnen Arbeitspakete erstellt.



2 User Interfaces (UI) / User Experience (UX)

Ausgearbeitet von Joe Martin

Da ein Programm nur so gut ist wie die Person, die es bedient, stellt das User Interface eines der wichtigsten Faktoren jedes Programms und jeder Website dar. Die Gestaltung des User Interfaces hat direkten Einfluss auf den Benutzer und wie dieser damit umgeht. Somit beeinflusst das Design des User Interfaces direkt die „User Experience“. Eine gutaussehende Benutzeroberfläche hat keinen Mehrwert, wenn sie kompliziert und unpraktikabel ist.[1]

An erster Stelle jedes User Interfaces steht somit die Funktionalität. Zudem ist die Übersichtlichkeit und die Durchschaubarkeit einer Benutzeroberfläche von höchster Priorität. Um dieses Ziel zu erreichen und eine gutaussehende, hoch funktionale und intuitive Benutzeroberfläche zu schaffen können eine Reihe an Gestaltungsprinzipien angewandt werden.

2.1 Gestaltgesetze

1923 wurden von Max Wertheimer, im Zuge seiner Arbeit an den Grundlagen der Gestalttheorie, sechs grundlegende Faktoren formuliert, die Zusammenhänge in der Wahrnehmung widerspiegeln.

Das Wissen über diese Gesetze erleichtert die Gestaltung von User Interfaces weitgehend. Besonders relevant sind in diesem Zusammenhang zwei dieser Faktoren:

Das Gesetz der Nähe

Es besagt, dass Elemente mit geringem Abstand zueinander als zusammengehörig interpretiert werden. Beim bewussten Anwenden dieses Faktors, ist es leichter übersichtliche Benutzeroberflächen zu gestalten, da durch das selektive Gruppieren Zusammenhänge geschaffen werden können, die nicht weiter explizit erwähnt werden müssen.

[1] Business Insider (2019)

Das Gesetz der Ähnlichkeit

Dieses Gesetz besagt, dass Elemente, die einander ähnlich sind, eher als zusammengehörig wahrgenommen werden. Dabei macht es keinen Unterschied in welcher Weise die Elemente sich ähneln. Dieser Faktor findet in der User Interface Gestaltung bei der Form wiederkehrender Elemente, der Farbe, Schriftart, Schriftstärke und Schriftfarbe Anwendung.

2.2 Erstellen eines Entwurfs

Mit der Ausarbeitung des Designs werden Richtlinien festgelegt, die sich durch das gesamte Projekt ziehen. Es ist von großer Wichtigkeit, dass sich im User Interface ein roter Faden in Sachen Gestaltung erkennen lässt. Es macht die Oberfläche nicht nur anschaulicher, sondern auch berechenbarer, was wiederum die Benutzung erleichtert.

Von der Farbwahl bis hin zur Wahl der Schriftart ist es gängige Praxis jedes Detail im Voraus zu planen und in mehreren Designentwürfen festzuhalten. Der finale Entwurf dient dann als Vorlage bei der Umsetzung in der Entwicklung.

2.2.1 Farbschema

Um dem Benutzer die Funktionalität der Applikation so schnell wie möglich verständlich zu machen, wiederholen sich die wichtigsten Elemente des Designs immer wieder, dieses Prinzip gilt auch für die Farbwahl. Es wurde Blau (Farbcode: #6190FF) als primäre Farbe gewählt, um auf Aktionen oder aktive Elemente hinzuweisen und die Farbe Rot (Farbcode: #FF4949) für Fehlermeldungen. Das restliche Farbschema basiert auf diversen Grautönen, die hauptsächlich Hintergründe voneinander unterscheiden.

Abbildung 2: Farbschema

Die Entscheidung zu diesem Farbschema geschah mit dem Ziel vor Augen die Benutzeroberfläche zu vereinfachen und wurde unterstützt durch die Grundlage, dass so gut wie alle der derzeit meistverwendeten Applikationen ebenfalls auf den Stil eines „Dark Modes“ setzen.

2.2.2 Anwendung der Prinzipien und des Farbschemas

Auf der Abbildung rechts ist der finale Entwurf der Inserat Benutzeroberfläche, die im Rahmen der in diesem Projekt entwickelten Plattform Studybuddy entstanden ist, zu sehen.

Abbildung 3: UI: Inserat

Die Elemente wurden hierarchisch nach der Wichtigkeit angeordnet und farblich betont, um dem Benutzer die Informationsaufnahme zu erleichtern.

Überschriften wurden durch einen helleren Farbton und fette Schriftstärke hervorgehoben, was den Benutzer sofort auf das wichtigste aufmerksam machen sollte. Zusätzlich ein oben angeordneter prägnanter Button in Primärfarbe, welcher in diesem Fall die direkte Interaktion mit dem Inserat kennzeichnet. Das Hauptaugenmerk ist auf diese zwei Elemente gerichtet, was ein Überfordern des Benutzers mit Informationen oder Aktionen verhindern soll.

Alle nachfolgenden Informationen sind in einem blasseren Farbton dargestellt, da diese nur bei weiterem Interesse für den Nutzer von Bedeutung sind. Zudem wurden die Abstände der jeweiligen Elemente so gewählt, dass Gruppierungen entstehen.

Die erste dieser Gruppen beinhaltet die Überschrift, den primären Button „Anfragen“ und die wichtigsten Informationen wie Datum, wie viele Nutzer noch gesucht werden, die Universität und die gesprochenen Sprachen. Sie beinhaltet die wichtigsten Informationen eines Inserats und ist somit an oberster Stelle.

Als nächstes kommt die Beschreibung des Inserats, die als Ein-Element Gruppe gesehen werden kann.

Zu guter Letzt eine Gruppe, die den Inserat Ersteller und die bereits beigetretenen Teilnehmer beinhaltet, mit denen bei weiterem Interesse interagiert werden kann.

Somit sollte, durch die Aufteilung der Informationen und der Reduzierung von Ablenkungen, auf den ersten Blick klar sein, wo auf der Oberfläche gesucht werden muss, um die gewünschten Informationen zu erhalten.

2.2.3 Iterationen

Die Gestaltung eines User Interface ist, wie bei allen anderen kreativen und gestalterischen Disziplinen, ein Prozess. Die bestmögliche Zusammenführung aus Form und Funktionalität kristallisiert sich erst nach mehrfachen Durchläufen heraus.

Dies war auch bei der Entwicklung unseres Diplomprojekts der Fall. Das regelmäßige Überprüfen und neu Überarbeiten des ursprünglichen Entwurfs in der Anfangsphase des Projekts führte zu einem weitaus ausgereifteren Endergebnis.

Die Farbgestaltung des ersten User Interface (UI) Entwurfs von Studybuddy war gekennzeichnet von der Verwendung von sehr hellen und markanten Farben. Im Verlauf der weiteren Gestaltung und Weiterentwicklung der Funktionalität wurde klar, dass eine schlichtere und weniger ablenkende Oberfläche sinnvoller ist, was schließlich zur Umsetzung eines einfachen dunklen Designs führte.

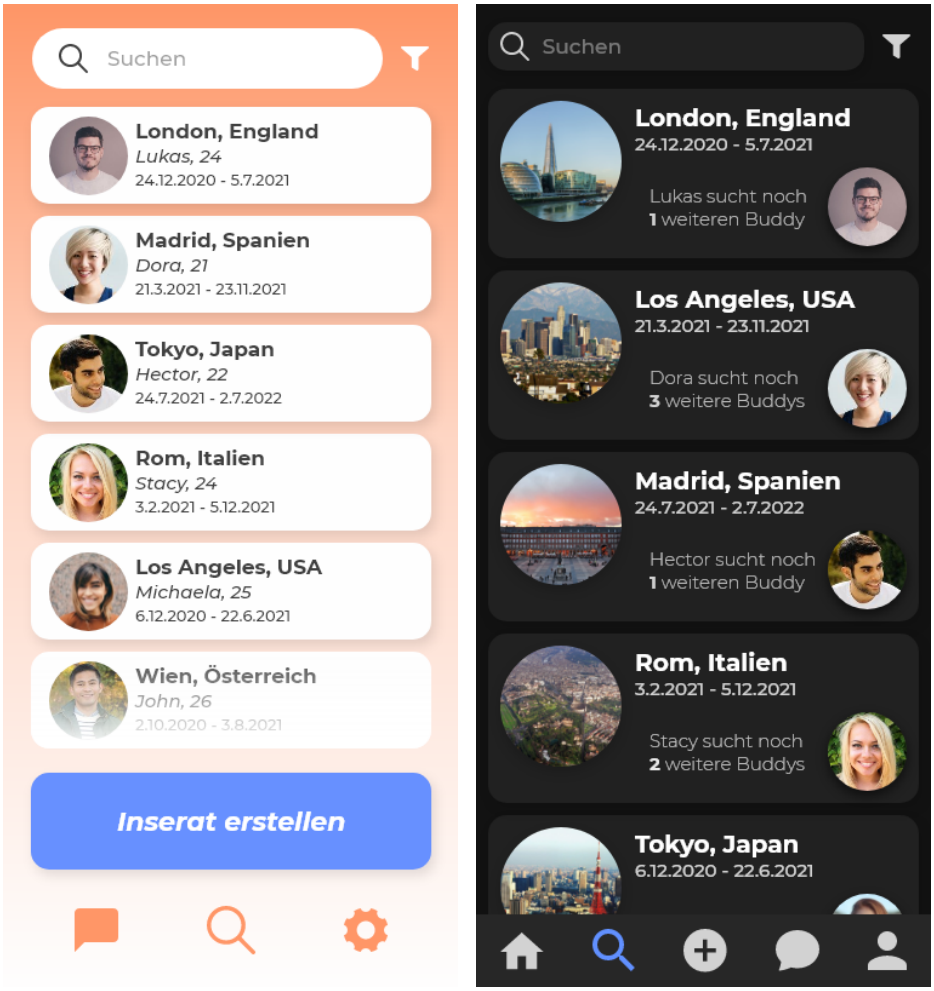


Abbildung 4: UI: Erstentwurf und Umsetzung

2.3 Adobe XD

Adobe XD ist ein Programm mit dem User Interfaces gestaltet und simple Prototypen erstellt werden können. Es wurde in der Entwicklung von Studybuddy verwendet, um die Benutzeroberfläche der mobilen App zu gestalten.

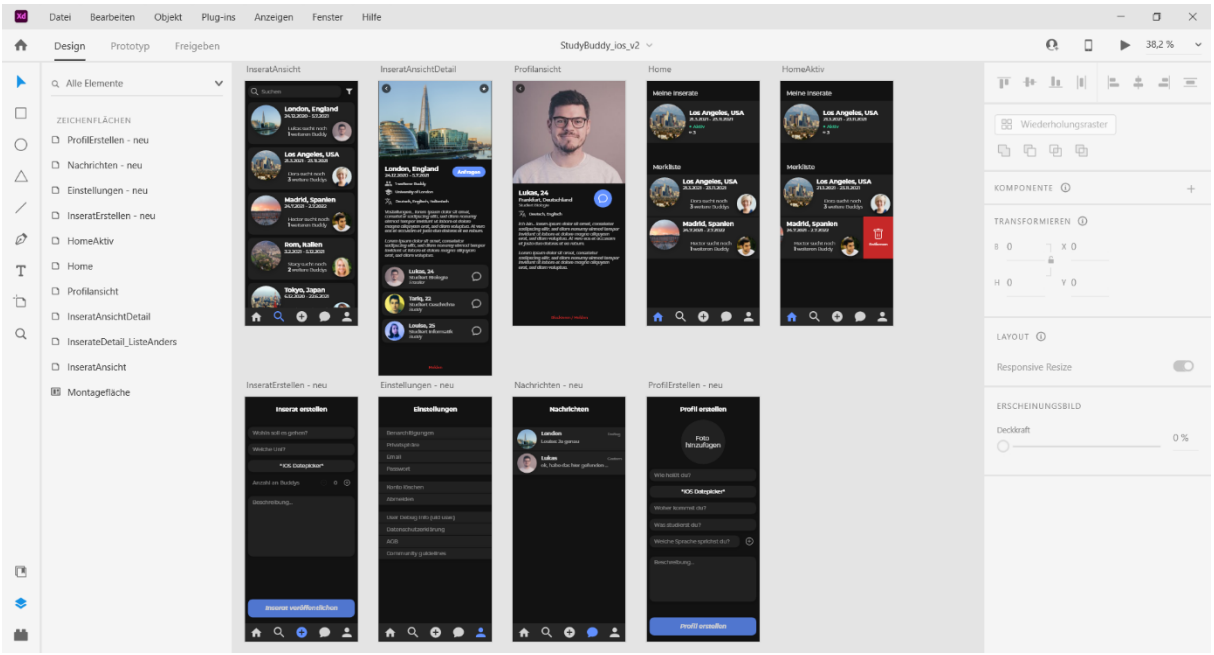


Abbildung 5: Adobe XD

Es lassen sich schnell mehrere Zeichenflächen anlegen, welche die einzelnen Teile des User Interfaces widerspiegeln. Bei Bedarf können Komponenten angelegt oder Zeichenflächen verbunden werden, um interaktive Prototypen ohne eine einzige Zeile Code zu erstellen. Die Gespeicherten Designs und Prototypen können auch über die Adobe App auf dem entsprechenden Endgerät direkt angezeigt und getestet werden.

2.3.1 Erstellen eines Projekts in Adobe XD

Nach dem Anlegen eines Projekts im Home-Menü (durch das Auswählen einer Vorlage oder dem Angeben der gewünschten Maße der Zeichenfläche) befindet man sich in der Design-Ansicht des Projekts und man kann nun die Benutzeroberfläche nach Belieben gestalten. Mithilfe der Toolleiste können nun diverse Formen oder neue Zeichenflächen erstellt und Texte eingefügt werden.

Bei dem Entwerfen von Studybuddy wurden von Anfang an hauptsächlich runde Formen eingearbeitet. So sind zum Beispiel alle Buttons oder Inputfelder, die in vielen Fällen als einfache Rechtecke definiert werden, abgerundet. Dies kann in Adobe XD

nach der Erstellung und Auswahl eines Rechtecks durch das Kontextmenü auf der rechten Seite durch die Anpassung des Eckenradius erreicht werden.

Auch Text lässt sich über die Auswahl des entsprechenden Werkzeugs in der Toolbar einfügen. Über das Kontextmenü können Eigenschaften wie Schriftart, -größe, -stärke oder -farbe bearbeitet werden. Passend zur Umsetzung des „Dark Mode Designs“ wurde eine helle Schriftfarbe gewählt. Die Schriftgröße ist in mehrere Kategorien unterteilt: Es wurden drei Abstufungen an Überschrift Größen, sowie Schriftgröße und -stärke für Primär- und Sekundärparagraphen gewählt.

Um Icons für die Umsetzung von Menüs zu verwirklichen, können entweder Grafiken importiert werden, oder eines der vielen Plugins für Adobe XD geladen werden, um eine Auswahl an frei verwendbaren Icons ohne große Umwege zu erhalten. Die zweite Möglichkeit kam bei der Erstellung des Entwurfs von Studybuddy zum Einsatz.

2.3.2 Prototypen

Durch das Auswählen des Reiters Prototyp können die im vorherigen Schritt erstellten Zeichenflächen nun in gewisser Weise zum Leben erweckt werden. Es kann mit dem Klick auf ein als Button vorgesehenes Element eine Verbindung zu einer anderen Zeichenfläche erstellt werden, welche in der Prototypen Vorschau dann nach einem Klick auf diesen Button gezeigt wird. Durch solch einfache Zusammenhänge können schnell Prototypen erstellt werden, welche einen guten Eindruck auf das Endprodukt geben.

Für Studybuddy ist die Funktion der Prototypenerstellung in einfachster Form zum Einsatz gekommen. Es wurden die wichtigsten interaktiven Elemente zusammengefügt, um für den Ablauf eine Idee zu bekommen.

Der dritte und letzte Reiter ist freigeben. Erstellte Prototypen können über einen von Adobe generierten Link an andere Personen versendet werden, die diesen dann testen können.

3 Datenbank

Ausgearbeitet von Joe Martin

3.1 Wahl der Datenbank

Es kann zwischen sehr grundlegend verschiedenen Datenbankmodellen gewählt werden. Von objektorientierten zu relationalen oder dokumentorientierten Datenbanken, gibt es sehr viele Möglichkeiten, um Daten abzubilden. Aufgrund des Vorwissens über relationalen Datenbanken und der Datenbanksprache SQL, fiel die Entscheidung auf das Datenbankmanagementsystem MariaDB.

MariaDB gilt als zuverlässig und leicht skalierbar, ist also auch für große Datensätze geeignet. Die Skalierbarkeit wird in der Umsetzung im Rahmen dieser Arbeit zwar keine große Rolle spielen, wenn Studybuddy aber in Produktion gehen würde und eine große Anzahl von Benutzern sich registrieren würden, wäre die Möglichkeit große Datensätze verwalten zu können durchaus von Vorteil.

3.1.1 Relationale Datenbanken

Die Daten in relationalen Datenbanken werden in verschiedenen miteinander in Beziehung stehenden Tabellen gespeichert. Diese bestehen aus Zeilen und Spalten, auch als Tupel und Attribut bezeichnet. Jedes Tupel verfügt somit über eine bestimmte Anzahl an Attributwerten. Jeder Datenbankeintrag benötigt einen aus einem oder mehreren Attributen bestehenden Schlüssel, um ihn eindeutig identifizierbar zu machen.

Dadurch dass ein Schlüssel einen Datensatz eindeutig identifiziert, können verschiedene Tabellen in Relation gesetzt werden. Dazu wird der Primärschlüssel einer Tabelle als Fremdschlüssel in einer anderen gespeichert.

Die Möglichkeit Tabellen über Schlüssel verknüpfen zu können ermöglicht es, Daten ohne jegliche Redundanzen zu sichern und vermeidet so Inkonsistenzen bei späteren Änderungen.

3.2 Planung der Datenbank

Bei der Planung einer relationalen Datenbank werden Entitäten, deren Beziehungen zueinander und deren Attribute erstmals festgehalten. Im Optimalfall kann das mittels eines Entity-Relationship-Diagramm (ER-Diagramm) erreicht werden. Danach wird ein relationales Schema erstellt, welches die Abbildung in der Datenbank widerspiegelt.

3.2.1 Entity-Relationship-Diagramm

Bei der Erstellung eines ER-Diagramms werden Entitäten in Form eines Quadrats und Attribute als mit Linien verbundene Ellipsen dargestellt. Bei der Definition von Beziehungen gibt es verschiedene Notationen. In dem ER-Diagramm, dass für Studybuddy ausgearbeitet wurde, wurde die Chen-Notation angewendet. Beziehungen zwischen den Entitäten werden dabei als Rauten abgebildet. Die Kardinalität wird jeweils auf die Seite der Raute geschrieben.

Es gibt drei verschiedene Beziehungsgrundarten:

1 : 1
Zu jeder Entität einer Entitätsmenge gibt es genau eine Entität einer anderen Entitätsmenge. Das gleiche gilt in Gegenrichtung.

1 : n
Zu jeder Entität einer Entitätsmenge gibt es viele Entitäten einer anderen Entitätsmenge. In Gegenrichtung wird die Entität allerdings nur genau einer anderen zugeordnet.

n : m
Zu jeder Entität einer Entitätsmenge gibt es viele Entitäten einer anderen Entitätsmenge. Das gleiche gilt in Gegenrichtung.

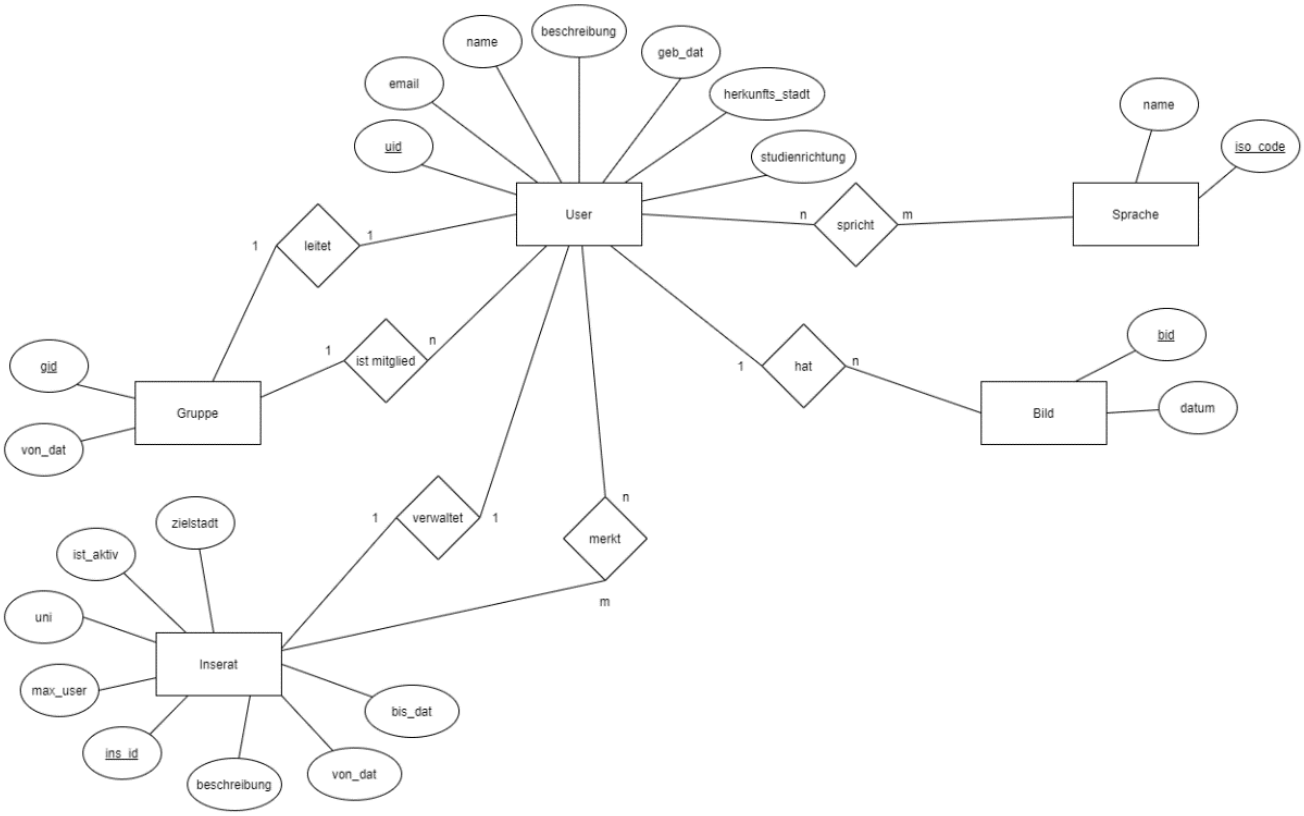


Abbildung 6: Entity-Relationship-Diagramm

Im ersten Entwurf wurden die Entitäten „User“, „Gruppe“, „Inserat“, „Sprache“ und „Bild“ abgebildet.

Zur Entität „User“ wird eine ID, die E-Mail-Adresse, der Name, eine Profilbeschreibung, das Geburtsdatum, die Herkunftsstadt und die Studienrichtung definiert. Da die Vernetzung von Benutzern in der Applikation eines der wichtigsten Bestandteile ist, hat die Entität „User“ die meisten Beziehungen.

Die Entität „Sprache“ ist mit dem Gedanken erstellt worden, Redundanzen in der Datenbank zu vermeiden. „Bild“ soll den Namen der auf dem Webserver abgespeicherten Bilder im Zusammenhang zum Benutzer abbilden. Die Entität „Gruppe“ ist nicht wie im obigen ER-Diagramm umgesetzt worden, da in der Umsetzung klar wurde, dass keine Notwendigkeit dazu besteht.

Inserate werden durch die Entität „Inserat“ widergespiegelt. Ein Inserat beinhaltet eine Zielstadt, die gewählt Universität, eine Beschreibung, ein Von- und Bis-Datum und hat eine maximale Anzahl an Benutzern, die beitreten können. Es wurde anfangs erstmals vorgesehen, dass ein Benutzer es erstellt und somit verwaltet und, dass andere Benutzer es merken können.

3.2.2 Relationales Modell

Mittels des relationalen Modells wird das ER-Diagramm für ein relationales Datenbanksystem umsetzungsfähig gemacht und bestimmt wie viele Tabellen erstellt werden müssen und wie Beziehungen mittels Fremdschlüsseln umgesetzt werden. Dabei gibt es zwei Schritte. Erstmals wird ein Initialentwurf erstellt, der danach verfeinert wird.

Die Allgemeine Darstellung einer Entität im relationalen Modell ist wie folgt:

```
Entität: {[AttrPri, Attr1, Attr2, …, AttrN]}
```

Nach diesem Schema werden erstmals alle Entitäten und auch ihre Beziehungen angeschrieben.

Die Verfeinerung dient dazu die Tabellen, die beim Initialentwurf entstehen nach Möglichkeit zusammenzulegen, da nicht für jede Beziehung unbedingt eine Tabelle erstellt werden muss. Beispielsweise muss für die Beziehung „verwaltet“ keine extra Tabelle erstellt werden, da der Schlüssel des Verwalters als Fremdschlüssel in der Tabelle „Inserat“ gespeichert werden kann.

Das relationale Schema für das zuvor erstellte ER-Diagramm sieht, ohne Einbeziehung der Entität „Gruppe“ wie folgt aus:

```
User: {[uid, email, name, beschreibung, geb_dat, passwort, gid]}
Sprache: {[iso_code, name]}
Bild: {[bid, datum, uid]}
Inserat: {[ins_id, ist_aktiv, uni, max_user, beschreibung, von_dat, bis_dat, ersteller_id]}

spricht: {[uid, iso_code]}
merkt: {[uid, ins_id]}
```

3.3 Umsetzung

3.3.1 Installieren von MariaDB

Am einfachsten lässt sich MariaDB über die Kommandozeile auf einem auf Linux oder Unix basierendem Betriebssystem installieren.

Auf Ubuntu geht das über den APT Package Manager mit folgendem Kommandozeilenbefehl:

```
$ sudo apt install mariadb-server
```

Danach sollte das „mysql_secure_installation“ Skript ausgeführt werden, welches abfragt, ob man diverse Sicherheitsschritte umsetzen möchte. Es empfiehlt sich alle Empfehlungen zu akzeptieren und ein Passwort für den Root-Benutzer zu setzen.

3.3.2 Anlegen einer Datenbank

Um die Datenbank und deren Tabellen zu erstellen, wurde der „MySQL Command-line Client“ verwendet. Damit können über die Kommandozeile mithilfe von SQL-Befehlen alle notwendigen Operationen durchgeführt werden um die Datenbank und weitergehend auch Tabellen zu erstellen, bearbeiten und abzufragen.

Um über den „MySQL Command-line Client“ zum Beispiel über den Root-Benutzer zuzugreifen wird folgender Befehl auf der Kommandozeile ausgeführt:

```
$ mysql -u root -p
```

Nun kann mittels SQL-Befehlen gearbeitet werden, um Datenbanken und Tabellen oder auch neue Benutzer zu erstellen oder diverse andere Operationen durchzuführen.

Das Erstellen der Datenbank wird mit einem einzigen SQL-Befehl erledigt:

```
CREATE DATABASE studybuddy;
```

Danach können die einzelnen Tabellen mittels dem „CREATE TABLE“ SQL-Befehl erstellt werden. Für die Tabelle „Bild“ lautet der Befehl wie folgt:

```
CREATE TABLE bild (
`bid` BIGINT(20) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
`datum` DATE NOT NULL,
`uid` BIGINT(20),
CONSTRAINT `bild_uid_fk` FOREIGN KEY (`uid`) REFERENCES `user` (`uid`)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

Nach diesem Schema wurden alle Tabellen angelegt. Somit steht das Grundgerüst der Datenbank und es können Daten hinzugefügt werden.

Alternativ dazu könnte eine SQL-Datei geschrieben werden, in der dieselben Befehle stehen. Diese kann dann über die Kommandozeile in MariaDB importiert werden.

3.3.3 Erweiterung

Im Laufe der Entwicklung von Studybuddy wurde klar, dass noch weitere Tabellen benötigt werden, um alle Daten abbilden zu können. Diese wurden ebenfalls über den „MySQL Command-line Client“ hinzugefügt. Zur Darstellung der erweiterten Version der Datenbank wurde ein Diagramm generiert.

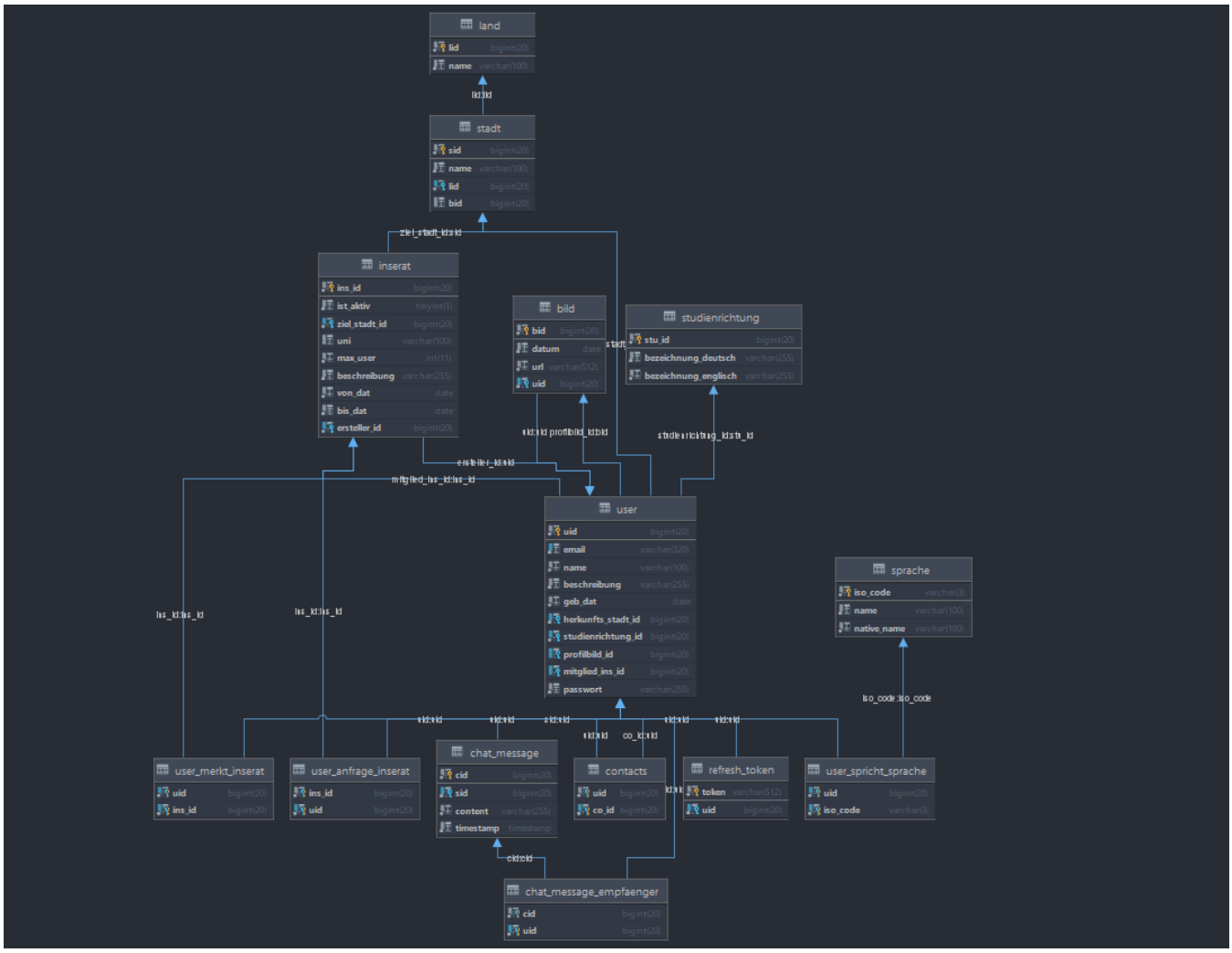


Abbildung 7: Datenbank Erweiterung

Da Redundanzen und Inkonsistenzen in relationalen Datenbanken stets zu vermeiden sind, sind die hinzugefügten Tabellen normalisiert worden.

3.3.3.1 Normalisierung

Das Ziel der Normalisierung ist die Vermeidung von Redundanzen und deren Folgen. Dazu gibt es mehrere Schritte, die das Ausmaß der Normalisierung bestimmen. Die sogenannten Normalformen. Für die Entwicklung von Studybuddy wurde die Datenbank bis zur „Dritten Normalform“ normalisiert.

„Eine Tabelle befindet sich in der 3. Normalform, wenn alle Datenfelder nur vom gesamten Schlüssel abhängig sind und untereinander keine Abhängigkeiten auftreten. Sobald ein Nicht-Schlüsselfeld nur über ein anderes Nicht-Schlüsselfeld identifizierbar ist, wird von „transitiver Abhängigkeit“ gesprochen. Transitive Abhängigkeiten verursachen ebenfalls Datenredundanz und Dateninkonsistenz.“[2]

[2] Fuchs, Elmar (2018): S. 56.

4 Software

4.1 Serverseitige Programmierung

Ausgearbeitet von Joe Martin

Um Daten an einen Browser oder eine Mobile App zu liefern, benötigt es ein Web-Service. Dieses sollte in der Lage sein HTTP (Hypertext Transfer Protocol) Anfragen zu interpretieren und die gewünschten Daten zurückzusenden. Dazu wird ein Programm benötigt, dass Anfragen entgegennimmt, und mittels Zugriffe auf die Datenbank die gewünschten Daten speichert oder abfragt und dem Client eine Antwort sendet.

Der beste Weg, um diese Funktionalität umzusetzen ist die Verwendung der REST (Representational State Transfer) Architektur, da sie durch die strikte Trennung von Client und Server sehr flexibel und skalierbar ist.

Eine weitere Anforderung an das Web-Service ist die Benutzerauthentifizierung. Da eine REST API grundsätzlich zustandslos ist, muss die Authentifizierung über ein Token Verfahren abgewickelt werden. Das bedeutet, dass bei so gut wie jeder Anfrage ein eindeutiger Wert mitgesendet werden muss, der den Benutzer verifiziert.

4.1.1 REST API

REST wurde im Jahr 2000 von Roy Thomas Fielding im Zuge seiner Doktorarbeit definiert. Einer der wichtigsten Punkte ist, dass REST Abstraktion bietet. Es wird über den Aufruf eines URI (Uniform Resource Identifier) auf Ressourcen zugegriffen.

Um mit den Ressourcen interagieren zu können, werden Endpunkte mittels den verschiedenen HTTP Methoden aufgerufen. Um eine Applikation zu verwirklichen, die hauptsächlich auf Datenbankoperationen basiert, werden folgende Methoden benötigt:

- GET
  - Dient dazu Ressourcen abzurufen
- POST
  - Dient dazu eine neue Ressource zu erstellen
- PUT
  - Dient dazu eine Ressource zu aktualisieren
- DELETE
  - Dient dazu eine Ressource zu löschen

4.1.2 Spring

Spring ist ein Open-Source Framework, dass darauf ausgelegt ist die Entwicklung mit der Programmiersprache Java zu vereinfachen. Es bietet einige Erweiterungen, die einem vor allem bei der Entwicklung von REST Services das Leben erleichtern. Um ein Spring Projekt zu erstellen, können unter start.spring.io mithilfe des „spring initilizr“ alle notwendigen Erweiterungen hinzugefügt werden.

Für die Entwicklung der Studybuddy REST API wurden die Erweiterungen „Spring Web“, „Spring Security, „Spring Data JPA“ und „MariaDB Driver“ gewählt.

Spring Web

In dieser Erweiterung werden die Erweiterungen „Spring MVC“, „Tomcat“ und „Jackson“ gebündelt, damit sie nicht manuell zum Projekt hinzugefügt werden müssen. Es ist die grundlegende Erweiterung zur Entwicklung von REST Services.[3]

Spring Security

Spring Security dient dazu die Umsetzung von Authentifizierung und Autorisierung zu erleichtern. Es lässt sich leicht für den eigenen Gebrauch erweitern, um eigene Anforderungen umsetzen zu können. [4]

[3] Vgl. Baeldung (2020)

[4] Vgl. Spring (2021)

Spring Data JPA

Diese Erweiterung erleichtert die Implementierung der JPA (Java Persistence Api). JPA ist eine Schnittstelle, um Datenbankeinträge mittels Java Objekten zu manipulieren und abbilden zu können.

MariaDB Driver

Treiber für die Einbindung der MariaDB Datenbank.

4.1.2.1 Umsetzung

Nachdem das Projekt erstellt wurde, kann mit der Planung und Programmierung angefangen werden. Für die Umsetzung von Studybuddy war vorab klar, dass einige Endpunkte benötigt werden, um alle Interaktionen mit der Datenbank zu ermöglichen.

Das Projekt wird übersichtshalber in mehrere Pakete aufgeteilt. Sie trennen beispielsweise den Datenbankzugriff von der Entgegenahme und Verarbeitung von Anfragen.

In der Abbildung rechts ist die Ordnerstruktur des Spring Projekts zu erkennen. Es werden nach der Grundlage des MVC (Model View Controller) Modells Komponenten voneinander getrennt. Weiters wurden Verzeichnisse für „DTOs“ (Data Transfer Objects), „Exceptions“, die „Spring Security Komponenten“, „Service Abstraktionen“ und „Utility Klassen“ erstellt.

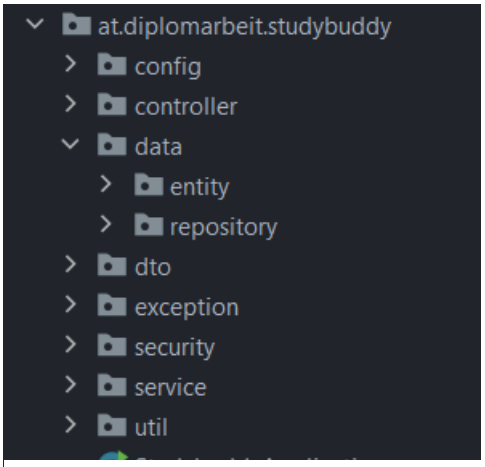


Abbildung 8: Ordnerstruktur Spring

Für die Umsetzung eines einfachen Endpunktes, der auf Anfrage Daten zurücksendet, werden folgende Komponenten benötigt:

Entity

Hier wird eine Datenbank Entität mit allen ihren Feldern als POJO (Plain Old Java Object) abgebildet. Dazu wird eine einfache Java Klasse mitsamt Attributen, Konstruktor Gettern und Settern erstellt und mit der „@Entity“ Annotation versehen. Um ein Schlüsselfeld zu kennzeichnen, wird es mit der „@Id“ Annotation versehen.

Repository

Das Repository Interface erbt vom von Spring zur Verfügung gestellten „CrudRepository“ Interface. Das „CrudRepository“ Interface bekommt zwei „Generic Types“

übergeben, die dazugehörige Entitätsklasse und den Datentyp des Primärschlüssels. Es bietet eine Liste an vorgefertigten CRUD (Create Read Update Delete) Operationen mit denen Datensätze manipuliert werden können an.

Controller

In der Controller Klasse werden die Endpunkte und deren URIs definiert. Die Klasse wird mittels der „@RestController“ Annotation markiert. Es gibt für die jeweiligen HTTP Methoden Annotationen wie zum Beispiel „@PostMapping“, „@GetMapping“, „@PutMapping“ und „@DeleteMapping“ mit denen die jeweiligen Java Methoden versehen werden. Zu diesen Annotationen kann der Pfad und wenn notwendig Pfadvariablen oder Anfrageparameter angegeben werden. Eine Methode für eine GET Anfrage könnte wie folgt aussehen:

```
@GetMapping(path = "/{id}")
public Inserat getInserat(@PathVariable Long id) {
    return inseratRepository.findById(id).orElseThrow(() ->
new InseratNotFoundException(id));
}
```

Exception

Eine Exception Klasse wird definiert, um zum Beispiel beim nicht Vorhandensein eines Datensatzes eine entsprechende Fehlermeldung zu liefern. Die einfachste Variante ist es von der Klasse „RuntimeException“ zu erben und eine neue Meldung zu definieren.

Für die Umsetzung des Benutzer-Account Systems wurde ein „/login“ Endpunkt erstellt. Dieser ist mittels einer POST Anfrage erreichbar und erwartet ein Objekt mit den Parametern „email“ und „passwort“ im JSON (Javascript Object Notation) Format. Als Antwort bekommt der Client ein „User“ Objekt in dem die wichtigsten Informationen sowie ein JSON Web Token enthalten ist. Mit diesem JSON Web Token werden alle weiteren Anfragen ermöglicht.

4.2 Web-Applikation

Ausgearbeitet von Joe Martin

Für die Umsetzung der Webseite wurde eine Single-Page-Applikation entwickelt. Sie besteht aus einer einzigen HTML Datei. Die gesamte Logik und Navigation ist mittels der Programmiersprache JavaScript realisiert. Der große Vorteil daran ist, dass während der Nutzung keine neue Seite geladen werden müssen. Alle notwendigen Daten werden bei Bedarf im Hintergrund mittels JavaScript vom Backend abgerufen. Dieses Verhalten vermittelt dem Benutzer eine hohe Geschwindigkeit und macht die Benutzung somit angenehmer.

Um die Umsetzung solcher Single-Page-Applikationen zu vereinfachen, gibt es einige JavaScript Frameworks. Die drei bekanntesten sind „React“ (wobei React eigentlich eine Library ist), „Angular“ und „Vue“. In Bezug auf den Entwicklungsablauf, funktionieren alle drei nach einem ähnlichen Prinzip. Es werden einzelne Komponenten definiert, die wiederverwertet werden können.

Um die Single-Page-Applikation für Studybuddy zu realisieren, wurde React gewählt.

4.2.1 React

React ist eine von Facebook entwickelte JavaScript Library, mit dem Zweck leichter und schneller Single-Page-Applikationen zu entwickeln. React ist komponentenbasiert und verwendet zur Darstellung JSX (JavaScript XML). JSX ist eine eigens entwickelte JavaScript Syntax die HTML ähnelt.

Der schnellste Weg, um ein React Projekt zu erstellen ist mittels „Create React App“. Dazu benötigt man eine Installation von „Node.js“, denn mit dem dort enthaltenen Package-Manager „npm“ kann mittels des Kommandozeilenbefehls `npx create-react-app <name>` ein Projekt generiert werden.

Um den Entwicklungsserver zu starten, muss nur in das Projektverzeichnis gewechselt werden und der Befehl `npm start` ausgeführt werden.

Wenn das Projekt für die Veröffentlichung auf einem öffentlichen Server bereit ist, kann mit dem Befehl `npm run build` eine optimierte Version des Projekts generiert werden.

Die Basisstruktur einer Komponente sieht wie folgt aus:

```
import React from 'react'

const hello = () => {
    return (
        <div>
            <h1>hello world</h1>
        </div>
    )
}

export default hello
```

Komponenten können Übergabewerte bekommen sogenannte „props“. So kann Beispielsweise eine Eltern Komponente Daten von der REST API abrufen und diese dann an eine andere weitergeben, um sie darzustellen.

4.2.1.1 Hooks

Hooks sind eine Weiterentwicklung von React. Sie erlauben unter anderem das Verwalten von Zuständen, ohne Klassen schreiben zu müssen. Damit sind Zustände von Variablen gemeint, deren Änderung auch in der Darstellung eine Änderung mit sich ziehen. Dafür wird der „useState-Hook“ verwendet. Er wird wie folgt definiert:

```
const [counter, setCounter] = useState(0);
```

„counter“ ist in diesem Fall der Wert, der mit 0 initialisiert wird. „setCounter“ ist eine Funktion mit der, der Wert „counter“ neu gesetzt werden kann.

Ein weiterer wichtiger Hook ist der „useEffect-Hook“. Dieser ermöglicht es Code auszuführen, wenn ein erneutes Rendern geschieht. Dabei gibt es mehrere Möglichkeiten. Standardmäßig wird er bei jedem Rendern einer Komponente ausgeführt, doch es kann ein Array mit übergeben werden, um diese Eigenschaft zu überschreiben. Dann würde er nur ausgeführt werden, wenn sich der Zustand des Inhalts des Arrays ändert. Wenn ein leeres Array übergeben wird, wird der „useEffect-Hook“ nur einmal beim

ersten Rendern der Komponente ausgeführt. Die Syntax des useEffect-Hook ist wie folgt:

```
useEffect(() => {
    alert(count);
});
```

4.2.1.2 Routen

Auch bei Single-Page-Applikationen soll Navigation mittels Links und unterschiedlichen URLs möglich sein. Aus diesem Grund müssen Routen definiert werden, unter denen bestimmte Komponenten gefunden werden können. Um das umzusetzen, wurde die Erweiterung „React Router“ verwendet. Diese kann über den Node Package Manager installiert werden. Sie verfügt über diverse Komponenten, die in das vorhandene Projekt importiert werden können, um Routen und Links zu erstellen. Um den Inhalt einer Route anzugeben, wird die Komponente als Child der Route hinzugefügt.

Route:

```
<Route path="/login">
  <Login />
</Route>
```

Link:

```
<Link className="nav-item nav-link" to="/">
  <span className="icons">home</span>
</Link>
```

4.2.1.3 Private Routen

Da Studybuddy über ein Benutzer-System verfügt, müssen gewisse Daten geschützt werden, die nur für Benutzer der Plattform bestimmt sind. Dazu wurde die „React Router“ Komponente erweitert. Es wurde eine Komponente namens „PrivateRoute“ erstellt, welche überprüft, ob der Benutzer bereits eingeloggt ist, bevor der Inhalt einer Route gezeigt wird. Wenn das nicht der Fall ist, findet ein Redirect zum Login statt.

```
const PrivateRoute = ({ children, ...rest }) => {
  const [user, , , , isLoading] = useContext(AuthContext);

  if (isLoading) {
    return <Spinner className="spinner-full" />;
  }

  return (
    <Route
      {...rest}
      render={({ location }) => {
        if (user !== null) {
          return children;
        } else {
          return (
            <Redirect
              to={{
                pathname: "/login",
                state: { from: location },
              }}
            />
          );
        }
      }}
    />
  );
};

export default PrivateRoute;
```

Die Komponente “PrivateRoute” kann nach dem gleichen Schema wie „Route“ verwendet werden.

4.2.2 Umsetzung

Das Design der Webseite ähnelt dem der iOS-App stark. Die meisten Elemente wurden vergrößert, Menüs und diverse andere Eingabefelder wurden ein wenig angepasst.

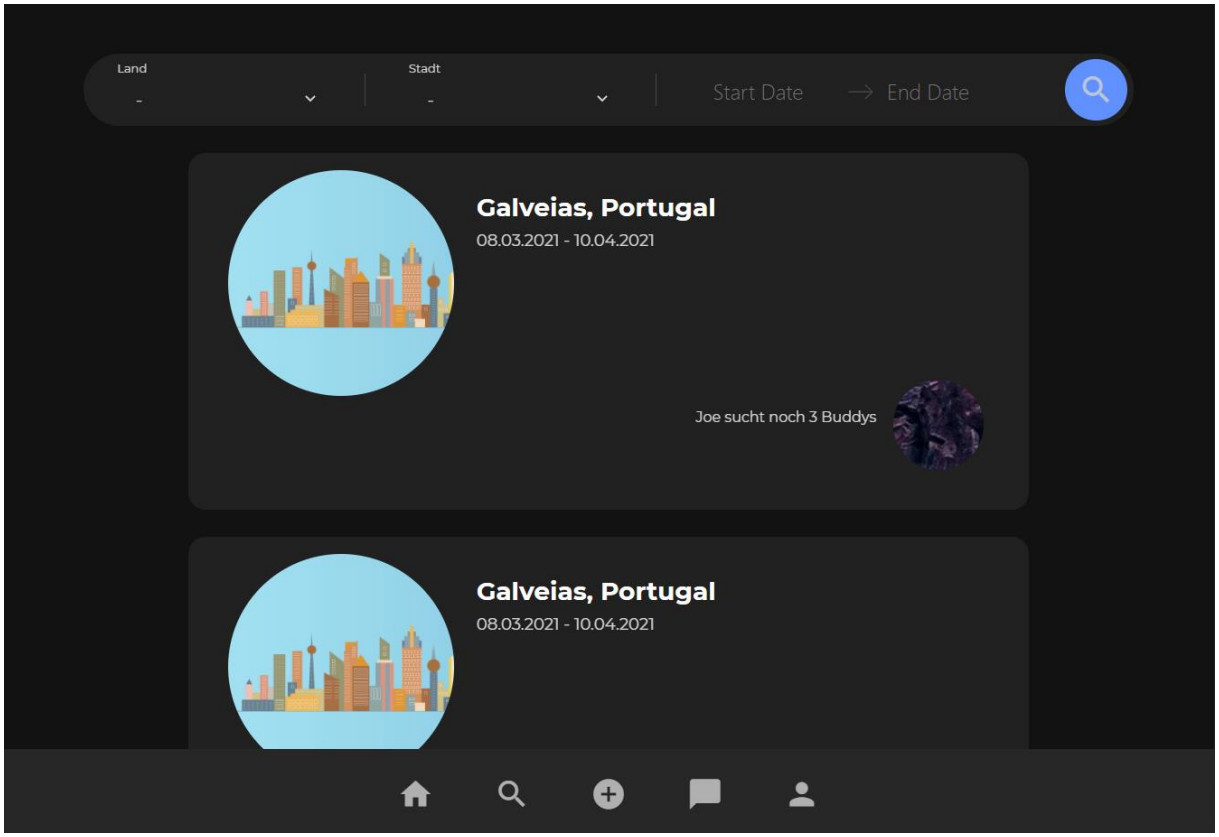


Abbildung 9: Webseite – Suchansicht

4.2.2.1 Sicherung des JWT

Nach einem erfolgreichen Login muss der JWT („Json Web Token“) gespeichert werden, um weitere Anfragen an den Server machen zu können. Der Token sollte allerdings nicht unbedingt im LocalStorage gespeichert werden, da ansonsten die Möglichkeit einer Cross-Site-Scripting Attacke besteht. Aus diesem Grund wird der Token als State Variable gespeichert. Zusätzlich dazu wird automatisch ein Refresh-Token als httpOnly-Cookie gespeichert. Bei jedem neuen Laden einer Seite oder beim Ablauf der Gültigkeit des JWT wird der Endpunkt „/refresh“ der REST API aufgerufen, um zu überprüfen, ob der Refresh-Token gültig ist. Wenn das der Fall ist, dann bekommt der Benutzer einen neuen JWT und einen neuen Refresh-Token.

4.3 Xcode

Ausgearbeitet von Ibrahim Farghali

4.3.1 Was ist Apple Xcode?

Xcode ist eine integrierte Entwicklungsumgebung (kurz IDE), die speziell zum Schreiben von Programmen verwendet wird, die für verschiedene Apple-Betriebssysteme geeignet sind. Xcode ist derzeit nur für Mac-Computer verfügbar.

Mit Xcode können Entwickler Software für macOS, iOS, watchOS und tvOS schreiben. Daher kann es als Plattform für alle derzeit relevanten Betriebssysteme verwendet werden, die in Apple-Geräten zum Einsatz kommen.

Xcode ist hauptsächlich für Entwickler gedacht, die Objective-C oder den moderneren Swift verwenden möchten, um Programme für diese Betriebssysteme zu schreiben. Gleichzeitig werden jedoch allgemeinere Sprachen wie C und C ++ sowie JavaScript und Ruby unterstützt. Wie von Apple gewohnt bietet Xcode eine klare und saubere Benutzeroberfläche. Wenn man ein neues Projekt startet, hilft der Assistent, die richtigen Einstellungen vorzunehmen. Neben Programmieranwendungen unterstützt das Tool auch das visuelle Design nachfolgender Benutzeroberflächen sowie Test- und Debugging-Software.

4.3.2 Bedeutung von Xcode

Da Xcode das einzige Tool ist, das Anwendungen für die beliebtesten Apple-Geräte entwickeln kann, ist Xcode für Entwickler, die diese Betriebssysteme verwenden, von entscheidender Bedeutung. Um diese Entwickler und Unternehmen zum Schreiben von Programmen zu ermutigen, können Entwickler Xcode kostenlos verwenden.

Das Hochladen der fertigen Anwendung zum Apple „App-Store“ ist jedoch nicht kostenlos. Für die Entwickler ist eine kostenpflichtige Mitgliedschaft erforderlich. Xcode verfügt über mehr als eine Million verschiedene Anwendungen für iOS und eine große Anzahl von Anwendungen für Mac OS. Im Vergleich zu Visual Studio und ähnlichen umfangreichen Umgebungen ist Xcode zu einer der wichtigsten IDEs geworden.

4.3.3 Funktionen der Entwicklungsumgebung

Bei Verwendung von Xcode übernimmt die IDE alles, was für die moderne Softwareentwicklung erforderlich ist. Das Schreiben, Kompilieren, Debuggen von Code, sowie das Verwalten großer Datenmengen und Projekte. Die Apple-Entwicklungsumgebung verfügt über eine sehr umfangreiche Dokumentations- und Hilfebibliothek, einschließlich SDK-Dokumentation und -Informationen, Codierungsrichtlinien, API-Referenzen und Beispielcodes. Da auf die API-Dokumentation direkt von Xcode aus zugegriffen werden kann, ist die Entwicklung auch für Anfänger einfach. Um eine grafische Benutzeroberfläche (GUI) für Mac OS und iOS zu erstellen, gibt es das Interface Design Tool seit Version 4.0 in Xcode.

Der iPhone-Simulator ermöglicht die Anzeige der iPhone-Oberfläche auf dem Mac-Computer. Entwickler können damit Programme schreiben und auf virtuellen Geräten testen, ohne ein iPhone vor Ort haben zu müssen. Es gibt jedoch einige geringfügige Einschränkungen. Mit dem Aufkommen von Geräten wie Apple TV und Apple Watch erweiterte das Unternehmen diese Funktion und stellt beispielsweise auch einen watchOS-Emulator zur Verfügung.

Xcode wird mit einem Quellcode-Prüfer geliefert. Das Tool arbeitet in Echtzeit. Dies bedeutet, dass Xcode während der Eingabe alle Fehler anzeigt. Mit der automatischen Vervollständigungsfunktion von Xcode verbringt man auch weniger Zeit mit dem Tippen. Die Fehlerprüfung gilt für C ++, Objective C, Swift und C.

Um die Leistung des Programms zu überwachen, beinhaltet Xcode das Programm „Instruments“. Dies umfasst Analysefunktionen, wie zum Beispiel Überwachung der CPU-Auslastung und des Speicherverbrauchs. Daher kann das „Instruments“ auch beim Debuggen hilfreich sein.

4.3.4 Xcode User Interface Design

Xcode wird mit einem grafischen Interface-Design-Tool namens Interface Builder geliefert. Hier entwickelt man die Benutzeroberfläche. Man kann damit Menüs entwerfen, Fenster zusammenstellen, Steuerelemente entwerfen und andere visuelle Elemente. Diese können aus vorhandenen Objekten in der integrierten Bibliothek extrahiert werden.

4.3.5 Backups und Version Control

Durch die automatische Speicherfunktion von Xcode besteht nur ein geringes Risiko, dass Änderungen an den Projekt- oder Quelldateien verloren gehen. Es ist nicht erforderlich, einen Setup-Vorgang durchzuführen, um diese Funktion zu aktivieren. Es passiert einfach automatisch. Wenn man die Änderungen rückgängig machen möchten, helfen die Befehle "Rückgängig" und "Zurücksetzen" dabei.

Wenn man eine eingehende Änderungsverfolgung benötigt, helfen die Tools zur Verwaltung der Quellcodeverwaltung. Man könnte dazu „Subversion“ oder „Git“ verwenden. Man kann ein Remote-Repository verwenden oder ein eigenes Repository erstellen und lokal speichern.

4.4 Swift

Ausgearbeitet von Ibrahim Farghali

4.4.1 Was ist Swift?

Apple hat Swift im Jahr 2014 als neue Programmiersprache für die Programmierung im Apple-Ökosystem eingeführt. Diese Sprache ist besonders relevant für die Entwicklung von iOS-Anwendungen, kann aber auch in der macOS-Umgebung verwendet werden. Swift ist jedoch nicht auf Apple-Systeme beschränkt, es kann auch für Linux verwendet werden, da Swift als Open Source zur Verfügung gestellt wird.

Swift verfügt über eine leistungsstarke statische Typisierung und basiert auf einer grundlegenden Syntax, die vereinfacht wurde, um eine optimale Lesbarkeit zu erzielen. Apple versucht, eine einfache und leicht verständliche Struktur zu erstellen, die Anfänger und Konvertierte schnell verstehen sollten.

Vor einigen Jahren war „Objective-C“ der Standard für die Anwendungsprogrammierung für macOS oder iOS. Apple hat 2014 seine eigene interne Programmiersprache Swift eingeführt, die bestimmte Konzepte verbessert oder vereinfacht.

Obwohl Swift auf „Objective-C“ basiert, sollte es bequemer und benutzerfreundlicher sein. Swift verwendet typsichere Variablen und die Syntax wurde stark vereinfacht.

4.4.2 Einsteigerfreundliche Programmiersprache

Swift eliminiert viele Dinge, die die Syntax komplizieren. Der Zweck besteht darin, den Code leicht lesbar zu machen und damit Anfängern eine komfortable Einführung zu bieten. In Swift befindet sich beispielsweise kein Semikolon am Ende des Ausdrucks. Apple bietet Unterstützung für Programmieranfänger über seine eigene Lern-App, damit der Einstieg in die Swift-Programmierung leichter wird.

Die iPad-App "Swift Playgrounds" vermittelt auf spielerische Weise die Grundlagen, insbesondere für Anfänger ohne Programmierkenntnisse. Man kann auch kleine Apps für Roboter und Drohnen schreiben.

4.4.3 Variablen mit optionalem Inhalt

Ein „optionaler Typ“ ist ein Variablencontainer, der zwei mögliche Zustände annehmen kann: Die Variable ist leer (null) oder enthält einen Wert, der dem deklarierten Typ entspricht. Intern handelt es sich um einen Aufzählungstyp.

Optionaler Inhalt dient hauptsächlich dazu, den Code einfacher und damit lesbarer zu machen. Sie haben ein Fragezeichen nach dem Datentyp, z. B. "Int?" Oder "String?".

Optionaler Inhalt kann verwendet werden, um Situationen abzudecken, in denen eine Variable nicht unbedingt einen Wert enthalten muss. Wenn beispielsweise ein Fehler auftritt, ist der Wert möglicherweise versehentlich leer: Man versucht, eine nicht vorhandene Datei zu lesen. Die Informationen sind jedoch nicht immer verfügbar: Nicht jeder hat einen zweiten Vornamen, und nicht jede Suche im Array führt zu einer Übereinstimmung.

4.4.4 Generics in Swift

Swift ermöglicht generische Funktionen, die jeden Typ akzeptieren können, sodass es auf viele Arten flexibel wiederverwendet werden kann. Swift kennt auch die generischen Typen und Strukturen, die nach Bedarf definiert werden können. Wörterbücher, optionale Variablen oder Arrays können auch intern mit generischen Typen verwendet werden, sodass jeder Datentyp akzeptiert werden kann.

4.4.5 Code-Beispiele

```swift
// variables
var implicitInteger = 70
var implicitDouble = 70.0
var explicitDouble: Double = 70.0

// constants
let apples = 3
let oranges = 5
let appleSummary = "Ich habe \(apples) Äpfel."
let fruitSummary = "Ich habe \(apples + oranges) Früchte."

print("Hallo Welt!")

// loops
let people = ["Anna": 67, "Julia": 8, "Hans": 33, "Peter": 25]
for (name, age) in people {
    print("\(name) ist \(age) Jahre alt.")
}

// functions (aka "named closures")
func sayHelloTo(yourName name: String) -> Void {
    print("Hello \(name)")
}

sayHelloTo(yourName : "Otto")

// multiline preformatted text
let text  = """
Hello Otto,

why don't you take a break?
"""
```

Abbildung 10: Swift Code-Beispiel[5]

[5] Wikipedia (2021)

4.4.6 UIView-Controller

In der iOS-Entwicklung werden verschiedene View-Controller zum Verwalten von Inhaltsansichten verwendet, z.B. UIView-Controller, Table-View-Controller, Collection-View-Controller, PageView-Controller usw.

Der View Controller ist die Grundlage für die interne Struktur der Anwendung. Jeder View-Controller verwaltet einen Teil der Benutzeroberfläche und die Interaktion zwischen der Benutzeroberfläche und den zugrunde liegenden Daten. Der View-Controller hilft auch beim Umschalten zwischen verschiedenen Teilen der Benutzeroberfläche. Jede Anwendung verfügt über mindestens einen View-Controller, dessen Inhalt das Hauptfenster ausfüllt.

UIView-Controller ist eine übergeordnete Klasse, mit der alle View-Controller von iOS-Anwendungen erstellt werden, einschließlich integrierter View-Controller wie Collection-View-Controller und Table-View-Controller. In iOS-Anwendungen muss die UIView-Controller-Klasse nicht direkt instanziiert werden. Stattdessen kann man Klassen definieren, die UIView-Controller erben und Lebenszyklusmethoden hinzufügen, um die Ansichtshierarchie zu verwalten.

Um eine Echtzeit-iOS-Anwendung zu erstellen, müssen mehrere Ansichts-Controller kombiniert werden, von denen jeder einen anderen Teil der Anwendung darstellt.

Jede iOS-Anwendung enthält mindestens eine UIView-Controller-Unterklasse. Um eine iOS-Anwendung zu erstellen, muss man mehrere benutzerdefinierte View-Controller erstellen, um die Gesamtfunktionalität der Anwendung zu definieren.

4.4.6.1 Ansichtsverwaltung

Die Stammansicht der Ansichten Hierarchie wird in der Eigenschaft „view“ von UIView-Controller angegeben. Die Stammansicht ist der Container für die restlichen Ansichten in der Ansichtshierarchie. Größe und Position der Stammansicht werden durch das Objekt bestimmt, das sie besitzt, d. h. entweder durch einen übergeordneten View-Controller oder das Window der App. Der View-Controller, der dem Window der App

gehört, ist der Root-View-Controller der App, und seine Ansicht ist so groß, dass sie das Window der App ausfüllt.

Es gibt die folgenden Möglichkeiten, die Ansichten in iOS-Anwendungen zu spezifizieren.

1) Die bevorzugte Methode zum Festlegen einer benutzerdefinierten Ansicht ist die Verwendung eines Storyboards. Man kann die Ansicht im Storyboard angeben und eine Verbindung zwischen der Ansicht und der entsprechenden View-Controller-Klasse herstellen. Die Beziehung zwischen den verschiedenen View-Controllern der Anwendung kann man auch im Storyboard selbst angeben. Um den View-Controller aus dem Storyboard zu laden, ruft man die InstantiateView-Controller-Methode „withidentifier“ der UIStoryboard-Klasse auf. Das Storyboard-Objekt erstellt das Viewcontroller-Objekt und gibt es an den Code zurück.

2) Man kann die NIB-Datei verwenden, um die Ansicht anzugeben. Die NIB-Datei erleichtert das Angeben einer Ansicht für eine einzelne View-Controller-Klasse, ermöglicht jedoch nicht das Definieren der Beziehung zwischen verschiedenen View-Controllern.

3) Man kann die Views für einen View-Controller mit der loadView()-Methode spezifizieren, in der man die View-Hierarchie programmatisch erstellt und die Root-View der Hierarchie der View-Eigenschaft UIView-Controller zuweist.
Im Storyboard wird die Root-View des View-Controllers immer so dimensioniert, dass sie in den zugewiesenen Bereich des View-Controllers passt. Für alle benutzerdefinierten Ansichten, die man dem Storyboard hinzufügt, muss man die Auto-Layout-Einschränkungen definieren, die die Größe und Position der Ansichten auf verschiedenen Bildschirmgrößen bestimmen.

4.4.6.2 Ansichtszustände

Ändert sich der Ansichtszustand eines View-Controllers, so werden folgende Methode aufgerufen:

viewWillAppear(): Diese Methode wird aufgerufen, wenn das Erscheinen des View-Controllers bevorsteht. In dieser Methode bereitet man alle Views auf das Erscheinen am Bildschirm vor.

viewDidAppear(): Diese Methode wird aufgerufen, wenn der View-Controller erschienen ist. Diese Methode enthält den Code, der ausgeführt werden soll, sobald die Ansichten auf dem Bildschirm erschienen sind.

viewWillDisappear(): Diese Methode wird aufgerufen, wenn der View-Controller im Begriff ist, zu verschwinden. Den Code zum Speichern der Änderungen oder anderer Statusinformationen kann man in dieser Methode platzieren.

ViewDidDisappear(): Diese Methode wird aufgerufen, sobald der View-Controller verschwunden ist.

4.4.7 Hierarchische Navigation mit UINavigation-Controller

Der Navigations-Controller verwaltet einen oder mehrere Child-View-Controller in der Navigationsoberfläche. Navigations-Controller werden in fast jeder iOS-Anwendung verwendet. Auch wenn ein oder mehrere untergeordnete View-Controller im Navigationsstapel verwaltet werden, erscheint nur ein View-Controller auf dem Bildschirm in einer Instanz. Wenn man ein Element im View-Controller auswählt, wird ein neuer View-Controller aktiviert.

Alle in einen Navigations-Controller eingebetteten View-Controller enthalten eine Navigationsleiste, die den Titel des View-Controllers, die Schaltfläche "Zurück" und andere Elemente enthält. Durch Tippen auf die Schaltfläche "Zurück" wird der oberste View-Controller vom Navigationsstapel entfernt. Die Schaltfläche "Zurück" gilt jedoch nicht für die Stammansicht des Stapels.

Ein Delegate-Objekt verwaltet das Verhalten eines Navigations-Controllers. Es kann die benutzerdefinierten Animationen für das Ein- und Ausblenden der View-Controller bereitstellen. Es kann auch die bevorzugte Ausrichtung für die Navigationsoberfläche festlegen.

4.4.7.1 Navigation Controller Methoden

Siehe Anhang

4.4.7.2 UINavigation-Controller Implementierung

Das Bild zeigt den Login bzw. den Registrierungsprozess anhand der Klasse UINavigation-Controller in Xcode für die App. Es wird ein Navigations-Controller erstellt und ein Segue-Übergang, der eine Root View Controller-Beziehung zwischen den Controllern herstellt. Segues werden für Übergänge und die Übergabe der Daten von einer Szene zur anderen verwendet.

Der Haupt-Controller hat zwei Buttons, mit denen man entweder zu den Login View-Controller oder zu den Sign up View-Controllern weitergeleitet werden kann. Um zwischen den Bildschirmen zu wechseln, kann man die Push- und Pop-Methode verwenden.

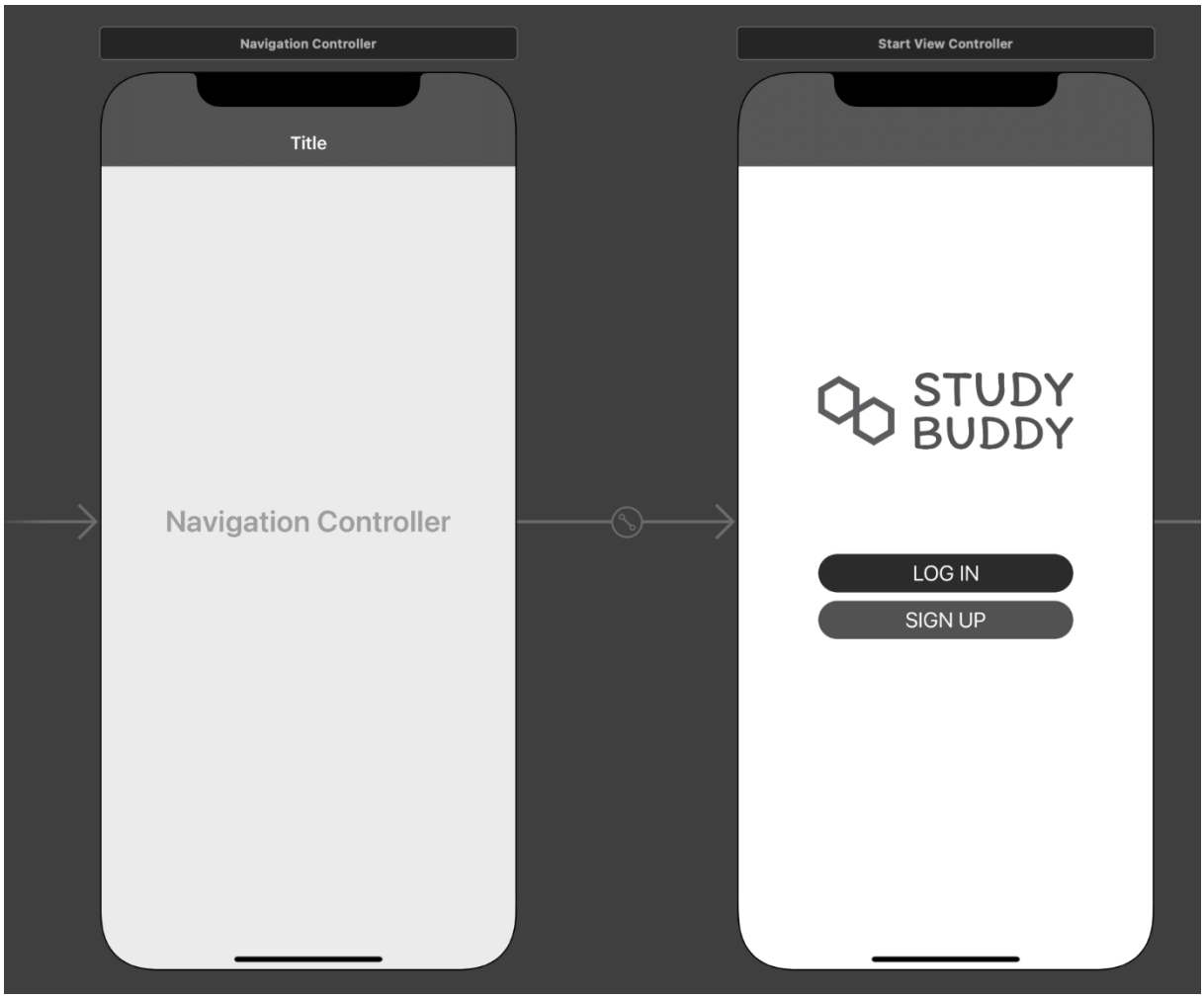


Abbildung 11: Start View Controller in Xcode

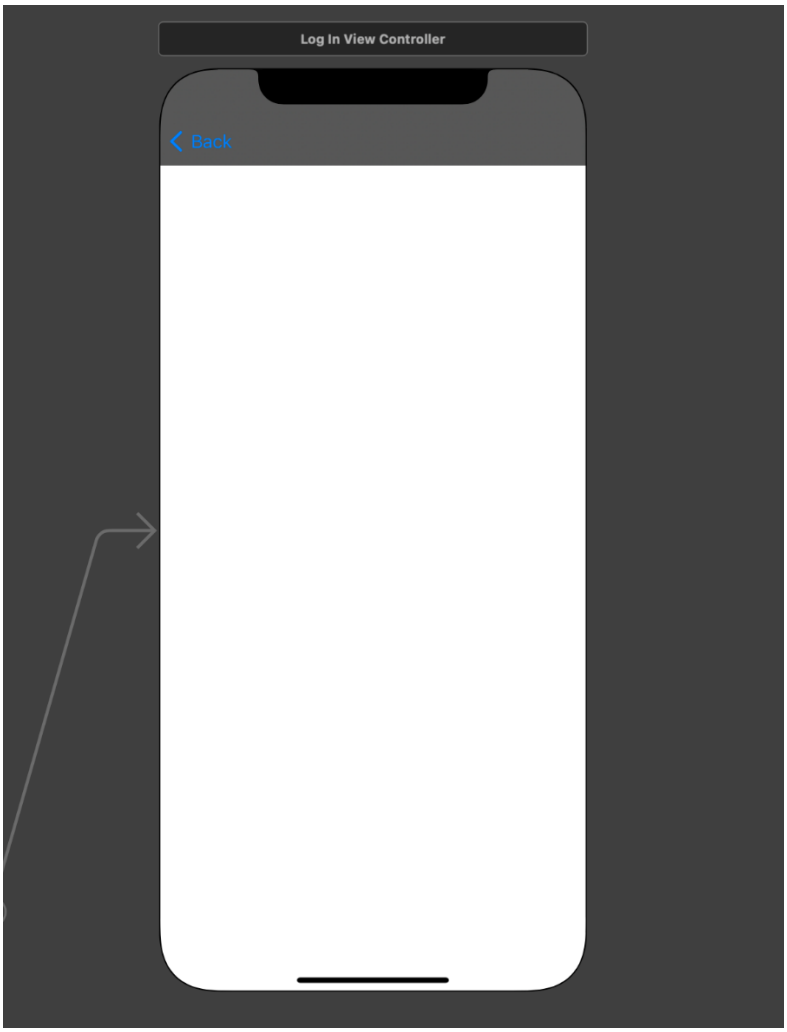


Abbildung 12: Log In View Controller in Xcode

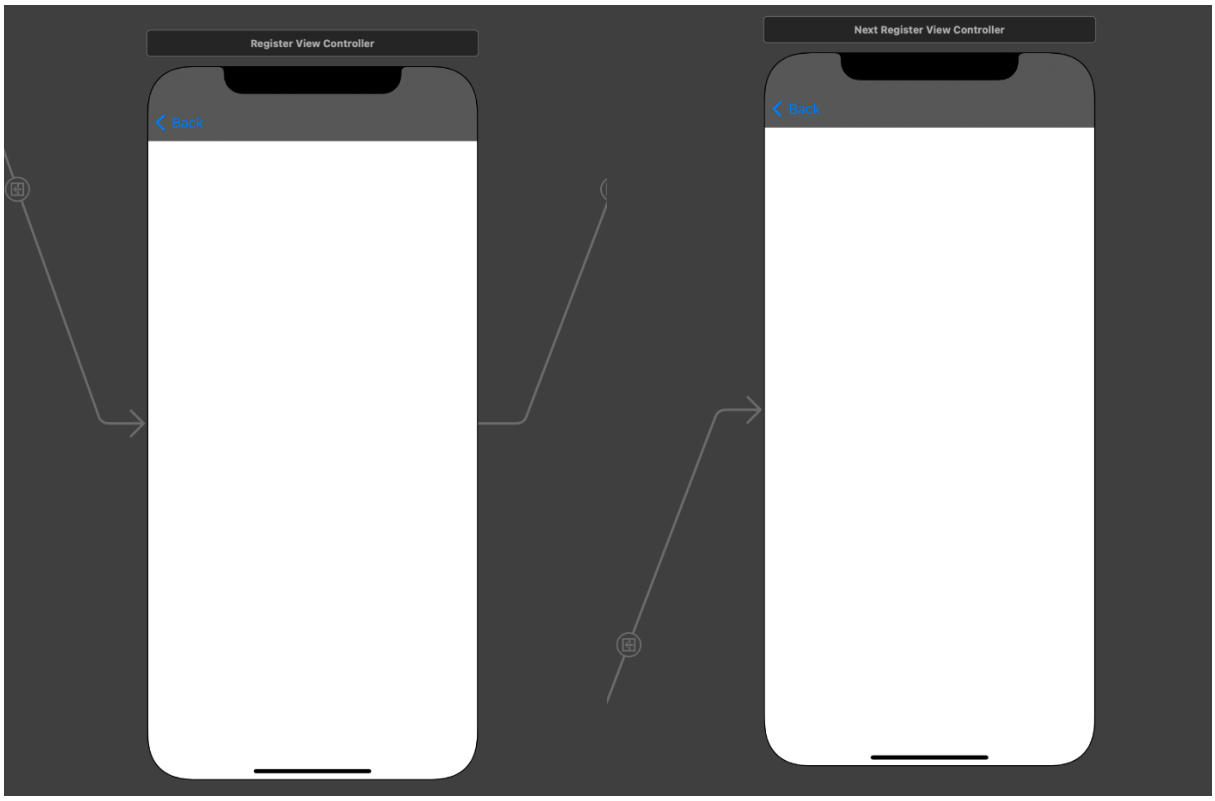


Abbildung 13: Die zwei Register Controller in Xcode

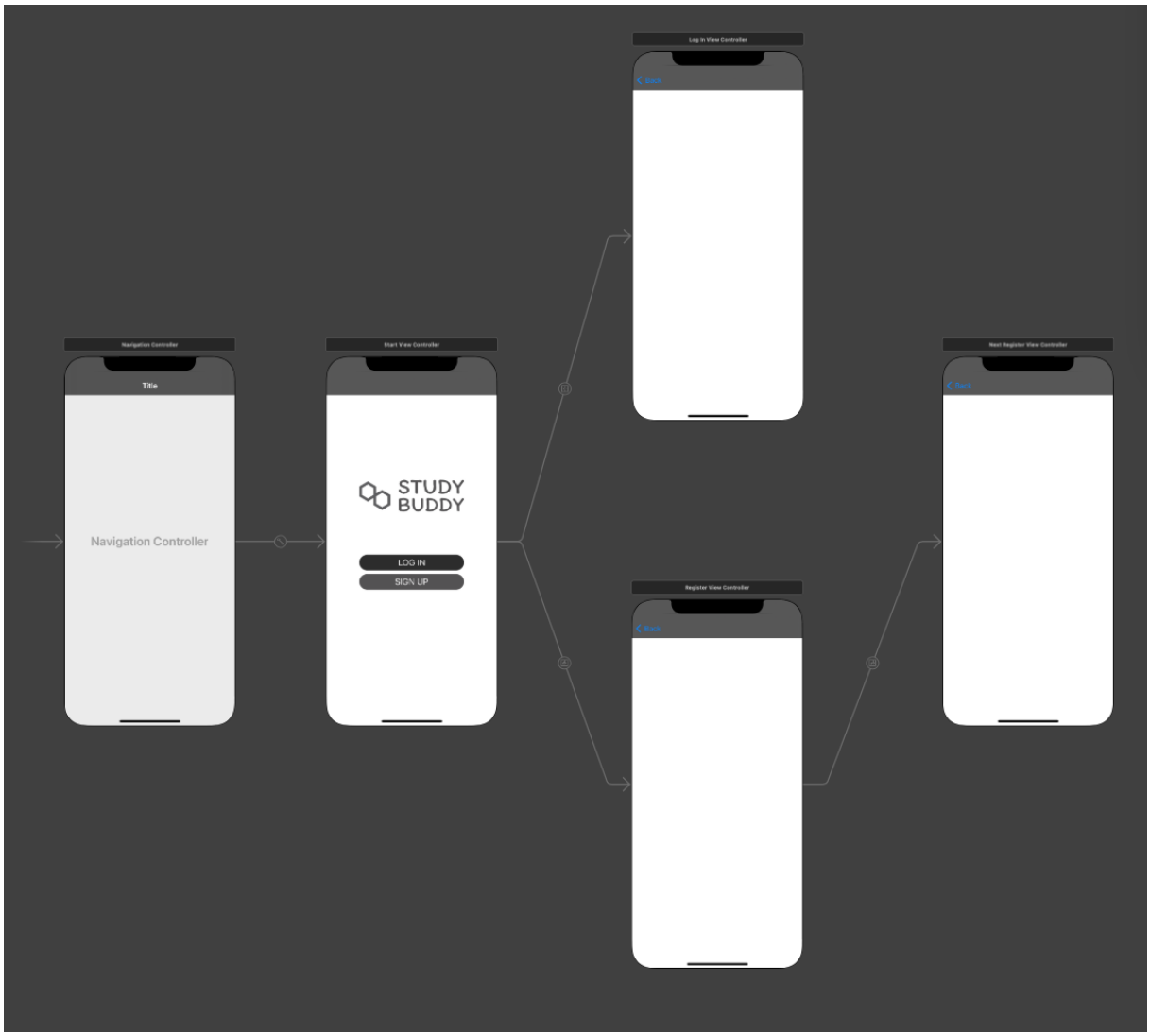


Abbildung 14: Das gesamte Bild des Logins bzw. den Registrierungsprozess

4.4.8 Tab-Bar-Controller

Ein Tab-Bar-Controller ist ein Container-View-Controller, der eine Multiselection-Schnittstelle verwaltet, wobei die Auswahl bestimmt, welcher untergeordnete View-Controller angezeigt werden soll.

Das Interface der Tab-Bar zeigt am unteren Rand des Fensters Registerkarten zur Auswahl zwischen den verschiedenen Modi.

Jede Registerkarte eines Tab-Bar-Controller-Interfaces ist mit einem benutzerdefinierten View-Controller verknüpft. Wenn der Benutzer eine bestimmte Registerkarte

auswählt, zeigt das Interface die Stammansicht des entsprechenden View-Controllers an und ersetzt damit alle vorherigen Ansichten.

Tab-Bar-Controllers werden in vielen Apps dazu verwendet, um dem Benutzer Zugriff auf die wichtigsten Teile der App zu geben. Man kann dies auch in Standard-iOS-Apps sehen, z. B. in den Apps "Telefon" und "Uhr".

4.4.8.1 Tab-Bar-Controller Implementierung

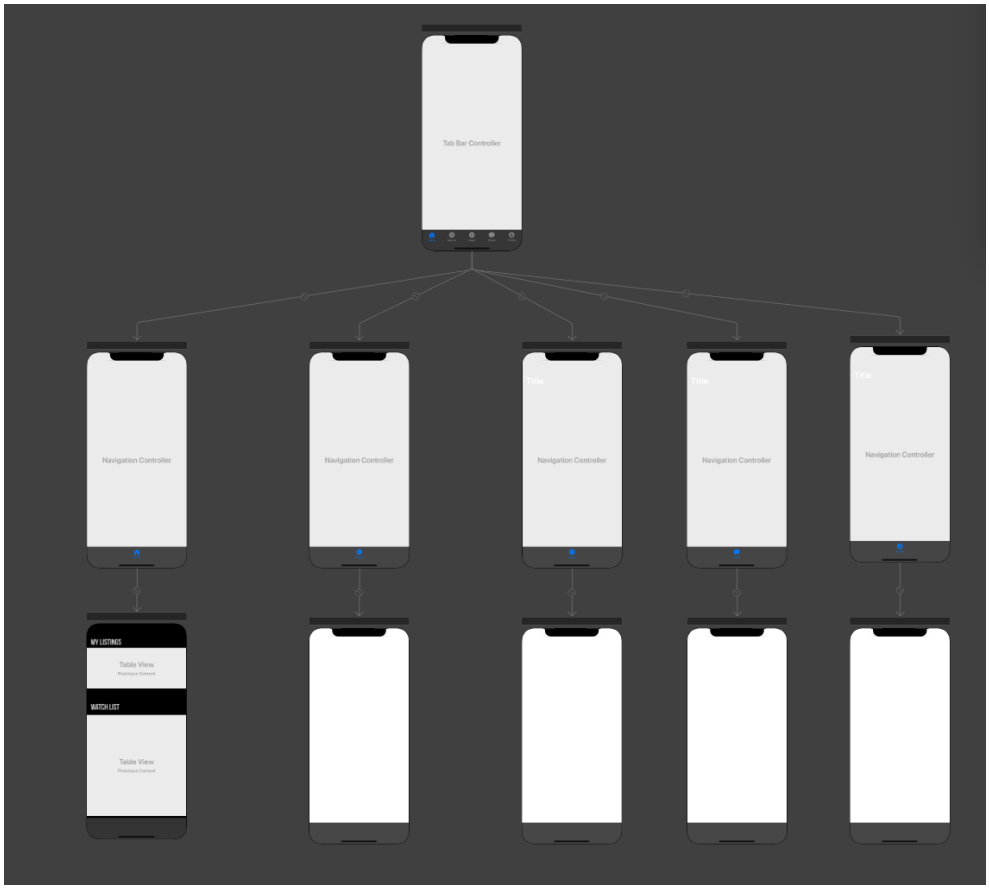


Abbildung 15: Tab-Bar-Controller

Die obere Szene besteht aus einem Tab-Bar-Controller mit fünf zusätzlichen ViewControllern. Dazwischen befinden sind fünf Navigation-Controller, die in den Tab-Bar-Controller eingebettet sind. Diese sind dafür zuständig, dass beim Wechsel zum neuen View-Controller, die Registerkartenleiste im User Interface erhalten bleibt. Die Beziehung zwischen dem Tab-Bar-Controller und den Ansichten wird durch einen Segue

dargestellt. Der oberste View-Controller verwaltet ein Array von View-Controllern, aus denen der Nutzer wählen kann.

In der unteren Abbildung stellt die erste Registerkarte in der Leiste die Home-Benutzeroberfläche dar. Die nächste Registerkarte, ein Lupensymbol, ermöglicht es dem Benutzer, in der App nach Inseraten zu suchen und filtern. Der dritte Tab, mit einem Plussymbol, ist das Interface für das Erstellen und Veröffentlichen eines Inserates. Um mit den verschiedenen Usern Mitteilungen auszutauschen, kann man auf der vierten Registerkarte eine Unterhaltung starten oder fortführen. Im letzten Tab in der Leiste findet man die Profileinstellungen, wo der Benutzer z.B. das Kennwort oder das Profilbild ändern kann.

Abbildung 16: Das Interface der Tab-Bar

4.4.9 Table-View

Table-View kann als eine Ansicht definiert werden, die Daten basierend auf Zeilen in einer einzelnen Spalte anordnen kann. Es wird in fast jeder iOS-Anwendung wie Kontakten, Facebook, Instagram usw. verwendet. Table-View ist eine Instanz der UITable-View-Klasse, die von der UIScroll-View-Klasse geerbt wurde.

Wenn man in einer iOS-App eine einzelne Spalte mit vertikal scrollendem Inhalt anzeigen möchten, wird eine Table-View verwendet. In der Table-View können mehrere Datensätze (in Zeilen unterteilt) angezeigt und bei Bedarf vertikal gescrollt werden.

Jede Zeile der Tabellenansicht repräsentiert jeden Datensatz der Datenquelle. In der iOS-Anwendung Kontakt wird beispielsweise der Name jedes Kontakts in einer separaten Zeile der Tabellenansicht angezeigt. Wenn man auf die Zeile klickt, erhält man Details zu diesem Kontakt

Man kann die Abschnitte in der Table-View erstellen, um die zusammengehörigen Zeilen zu gruppieren, oder die lange Liste der Datensätze in mehreren Zeilen ohne Abschnitte anzeigen.

Das Aussehen der Table-View wird von der Klasse UITable-View verwaltet, die von UIScroll-View erbt. In der Table-View wird die Zeile durch das Objekt der Klasse UI-Table-View-Cell simuliert, mit dem der eigentliche Inhalt angezeigt werden kann. Man kann die Table-View-Zellen anpassen, um beliebige Inhalte in der iOS-Anwendung anzuzeigen.

Um die Table-View zu verwenden, muss man ihre Delegaten- und DataSource-Methoden einstellen. Die Table-View ist ein datengesteuertes Objekt, d.h., sie bezieht die anzuzeigenden Daten aus dem Datenquellenobjekt. In den realen Anwendungen enthält das Datenquellenobjekt die Daten, die durch einen API-Aufruf vom Datenbankserver zurückgegeben werden.

4.4.9.1 Table-View Delegate Methoden

Die Table-View-Delegate-Methoden sind definiert, um der Table-View die folgenden Funktionen hinzuzufügen.

Siehe Anhang

4.4.9.2 Table-View DataSource Methoden

Um die Daten zu pflegen, die von der Table-View angezeigt werden sollen, muss man ein DataSource-Objekt pflegen, das das UITableViewDataSource-Protokoll implementiert. Das Datasource-Objekt verwaltet die Daten der Table-View.

Die wichtigsten im Protokoll definierten Methoden.

Siehe Anhang

4.4.9.3 Implementierung der Table-View

Der einfachste und bequemste Weg, eine Table-View zu erstellen, ist ein Storyboard. In der Objektbibliothek gibt es ein entsprechendes UI-Element, das einem vorhandenen View-Controller hinzugefügt werden kann. In den folgenden Abbildungen sieht man zwei Table-Views. Die obere Tabelle ist für die Auflistung der Inserate, die von dem Benutzer veröffentlicht wurde, während die untere für die Auflistung der Inserate ist, die von dem User als Favorit gekennzeichnete wurde.

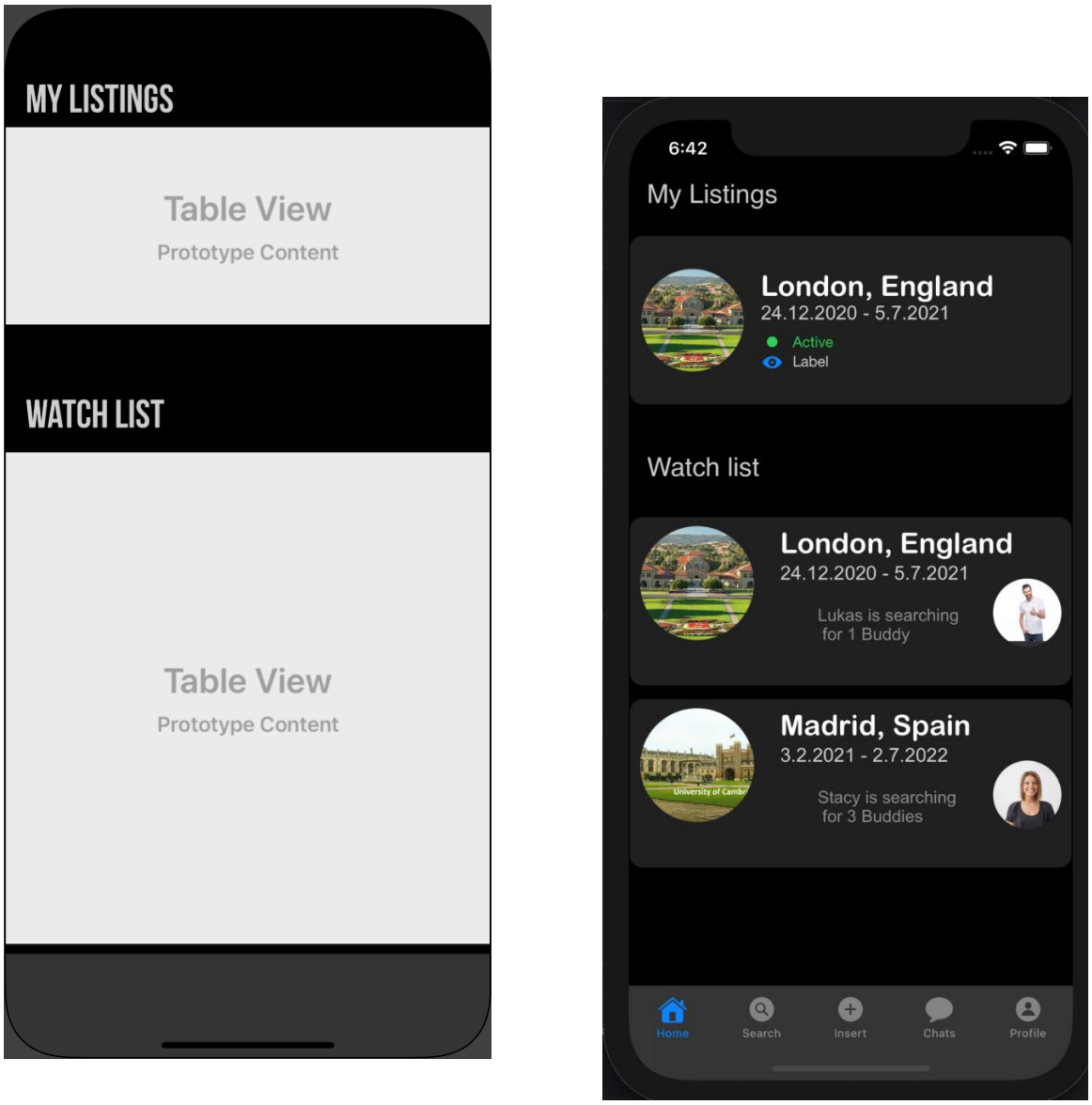


Abbildung 17: Table-View beim Erstellen im Storyboard und das Graphical User Interface

Die Table-View (siehe folgendes Bild) zeigt eine Instanz der UITableViewCell-Klasse für jede Zelle. Um sie so effizient wie möglich zu nutzen, erstellt man in der Regel zuerst Vorlagen auf Basis von UITable-View-Cell. Diese Vorlagen entsprechen dem gewünschten Erscheinungsbild der Zellen der Table-View. Dafür erstellt man eine NIB-Datei, in der man die Benutzeroberfläche und den Inhalt der Zellen der Table-View gestalten kann. Table-View-Cell ist eine Vorlage und enthält keine spezifischen Informationen. Es bestimmt nur das Erscheinungsbild der Zellen, die von der Anwendung benötigt werden.

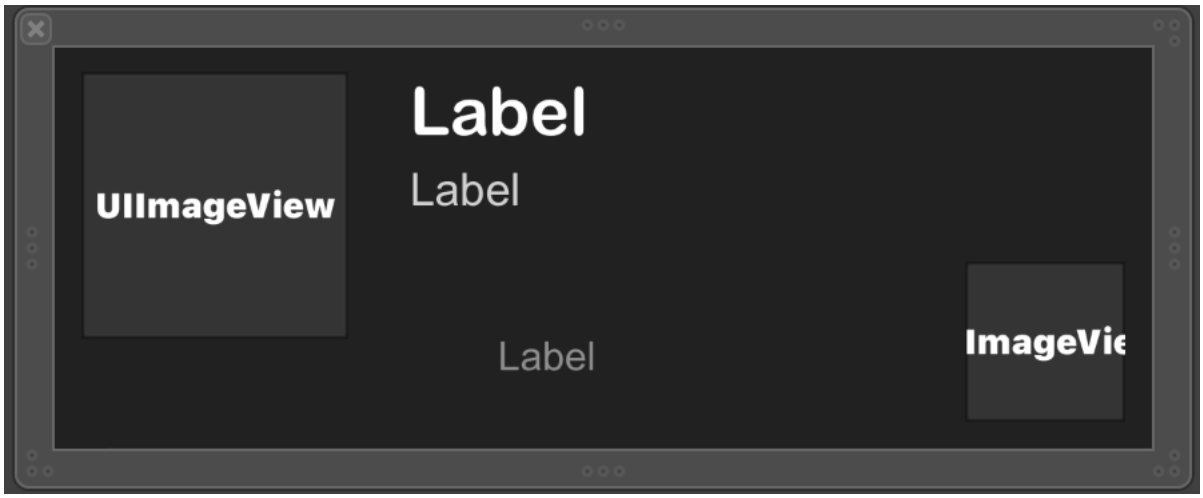


Abbildung 18: Vorlage der Zelle für die Table-View

Um die Konfiguration der Tabellenansichtszellen abzuschließen, muss man ihr noch einen passenden Identifier geben. Dadurch kann die Zellenvorlage im Code identifiziert und entsprechend geladen werden. Nachdem der Entwurf abgeschlossen ist, teilt man der Table-View im View-Controller mit, dass dieser Zelltyp für jede Zelle verwendet werden soll.

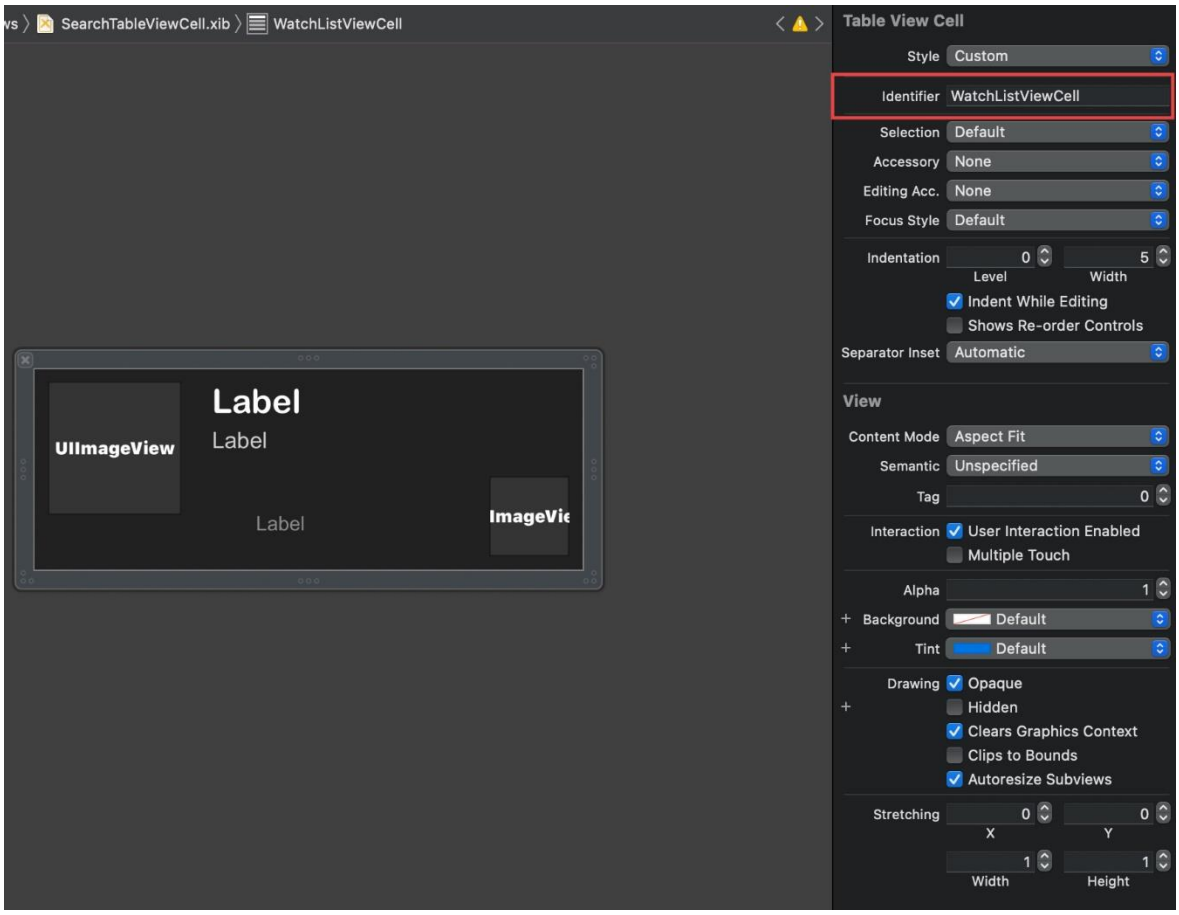


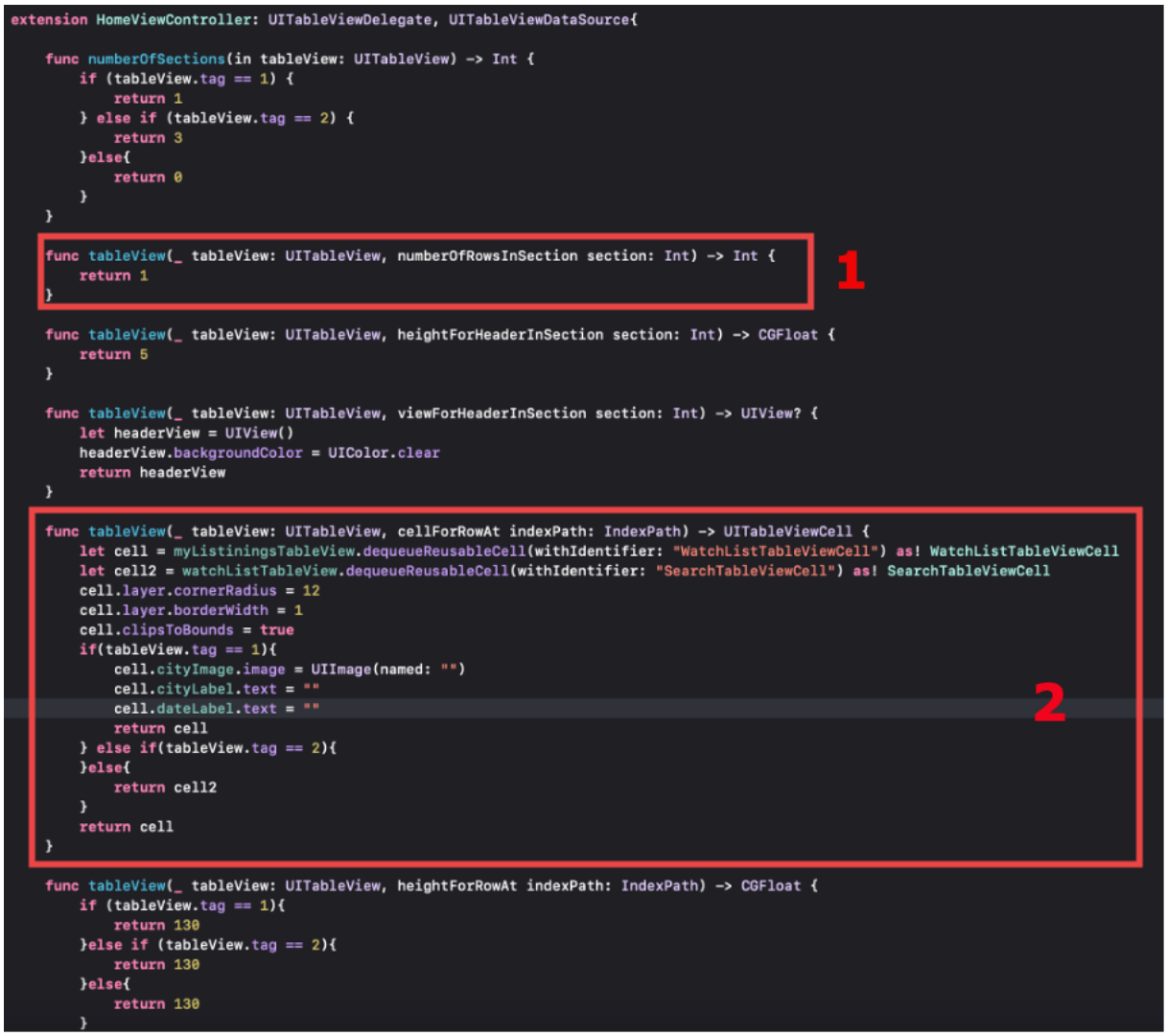
Abbildung 19: Identifier für die Nutzung der Zelle geben

```
let nib2 = UINib(nibName: "WatchListTableViewCell", bundle: nil)
watchListTableView.register(nib2, forCellReuseIdentifier: "WatchListTableViewCell")
```

Abbildung 20: Table-View mitteilen, dass dieser Zelltyp für jede Zelle verwendet werden soll

Der Inhalt der Table-View wird vom UITableViewDataSource-Protokoll abgeleitet. Im nächsten Bild sieht man, die beiden wichtigsten (und auch zwingend zu implementierenden) Methoden des UITableViewDataSource-Protokolls die wie folgt lauten:

1. tableView(_:numberOfRowsInSection)
2. tableView(_:cellForRowAt:)




```
extension HomeViewController: UITableViewDelegate, UITableViewDataSource{

    func numberOfSections(in tableView: UITableView) -> Int {
        if (tableView.tag == 1) {
            return 1
        } else if (tableView.tag == 2) {
            return 3
        }else{
            return 0
        }
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return 1
    }

    func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
        return 5
    }

    func tableView(_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {
        let headerView = UIView()
        headerView.backgroundColor = UIColor.clear
        return headerView
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = myListiningsTableView.dequeueReusableCell(withIdentifier: "WatchListTableViewCell") as! WatchListTableViewCell
        let cell2 = watchListTableView.dequeueReusableCell(withIdentifier: "SearchTableViewCell") as! SearchTableViewCell
        cell.layer.cornerRadius = 12
        cell.layer.borderWidth = 1
        cell.clipsToBounds = true
        if(tableView.tag == 1){
            cell.cityImage.image = UIImage(named: "")
            cell.cityLabel.text = ""
            cell.dateLabel.text = ""
            return cell
        } else if(tableView.tag == 2){
        }else{
            return cell2
        }
        return cell
    }

    func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
        if (tableView.tag == 1){
            return 130
        }else if (tableView.tag == 2){
            return 130
        }else{
            return 130
        }
```

Abbildung 21: Zwingend zu implementierenden Methoden

Die erstgenannte Methode liefert als Parameter einerseits die Table-View, für die die Anzahl der Zellen zurückgegeben werden soll, als auch die Section, für die die Anzahl der Zellen benötigt wird.

Die zweite Methode ist etwas komplizierter. Zusätzlich zur Table-View wird ein Index-Path bereitgestellt, der die Position der von dieser Methode zurückzugebenden Zelle enthält.

Wenn die Table-View geladen wird, wird zunächst überprüft, wie viele Zellen angezeigt werden sollen (basierend auf der tableView (_: numberOfRowsInSection :) -Methode).

Anschließend ruft die Table-View die tableView (_: cellForRowAt :) -Methode für jede dieser Zellen auf und übergibt hierbei die aktuelle Position.

Zuerst wird die erste Zelle angezeigt, dann die zweite, dann die dritte und so weiter.

Innerhalb dieser Methode erstellt man eine Instanz von UITable-View-Cell, die der gewünschten Vorlage entspricht, und konfiguriert sie. Man verwendet zum Laden einer geeigneten Instanz die Methode dequeueReusableCell (withIdentifier: for :), die man auf der übergebenen UITableView-Instanz aufrufen kann.

Zum einen übergibt man die Kennung und Position der gewünschten Zelle in Form des Index-Path.

Die auf diese Weise erhaltene UITableViewCell-Instanz wird im nächsten Schritt konfiguriert. Dazu kann man auf deren Eigenschaften zugreifen und entsprechende Werte festlegen.

Dafür verwendet man den Index-Path und dessen Property „section“, der diese Informationen enthält. Die so erstellte und konfigurierte Zelle gibt man im Anschluss mittels return zurück.

5 Security

Ausgearbeitet von Ibrahim Farghali

5.1 Was ist ein JWT?

JSON Web Token (JWT) ist ein offener Standard (RFC 7519), der ein kompaktes und eigenständiges Verfahren für eine sichere Übertragung von Informationen als JSON-Objekte definiert. Diese Informationen sind digital signiert, sodass sie überprüft und als vertrauenswürdig eingestuft werden können. Zum Signieren verwendet JWT das „RSA“- oder „ECDSA“-Verfahren

Während verschlüsselte Token Ansprüche vor anderen Parteien verbergen, können signierte Token die Integrität der darin enthaltenen Ansprüche überprüfen.

Wenn ein öffentliches / privates Schlüsselpaar zum Signieren eines Tokens verwendet wird, kann die Signatur beweisen, dass nur die Partei, die über den privaten Schlüssel verfügt, die Partei ist, die ihn signiert hat.

5.1.1 Verwendung von JSON-Web-Token

Autorisierung: Dies ist die häufigste Lösung für die Verwendung von JWT. Sobald sich der Benutzer anmeldet, bekommt der Client eine JWT vom Server, und jede nachfolgende Anforderung enthält das JWT und ermöglicht somit dem Benutzer den Zugriff auf die vom Token zugelassenen Routen, Dienste und Ressourcen. Informationsaustausch: JSON-Web-Token sind eine hervorragende Möglichkeit, Informationen sicher zwischen Parteien zu übertragen. Indem der JWT mit einem öffentlichen / privaten Schlüsselpaar signiert wird, kann sichergestellt werden, dass der Absender, der ist, für den er sich auch ausgibt. Da die Signatur anhand des Titels und der Benutzerdaten berechnet wird, kann man prüfen, ob der Inhalt manipuliert wurde.

5.1.2 Die Struktur des JSON-Web-Token

In seiner kompakten Form besteht JSON-Web-Tokens aus drei von Punkten getrennten Teilen:

Header.Payload.Signature

Die Ausgabe besteht aus drei durch Punkte getrennten Base64-URL-Zeichenfolgen, die in HTML- und HTTP-Umgebungen problemlos übergeben werden können.

Im Folgenden wird ein JWT gezeigt:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Abbildung 22: JSON-Web-Token[6]

[6] Tanmai Gopal, Hasura (2020)

5.1.2.1 Header

Der Header besteht normalerweise aus zwei Teilen: der Art des Tokens, d.h. JWT und der verwendete Signaturalgorithmus wie z.B. HMAC SHA256 oder RSA.

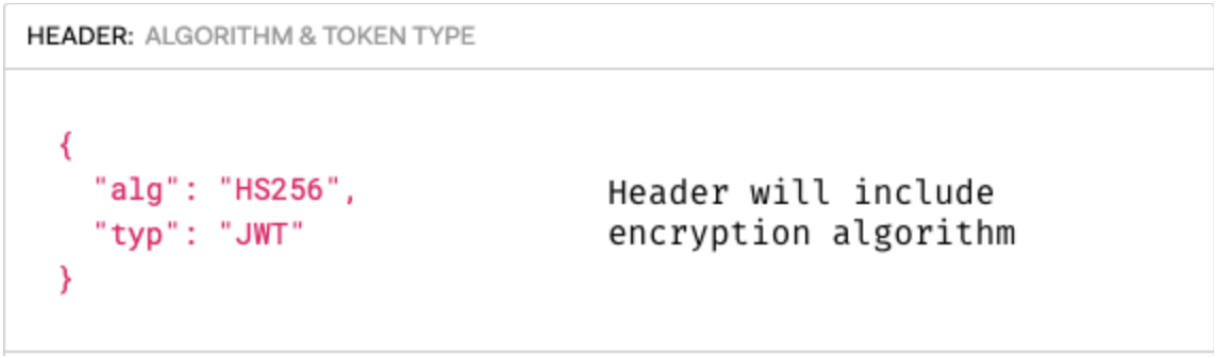


Abbildung 23: JWT-Header[7]

Anschließend wird dieser JSON mit Base64Url codiert, um den ersten Teil des JWT zu bilden.

5.1.2.2 Payload

Der zweite Teil des Tokens ist die Payload, die die Claims enthält. Claims sind Angaben zu einer Entität (zum Beispiel einer Person) und zusätzlichen Daten. Es gibt drei Arten von Claims: registrierte, öffentliche und private Claims.

Registered Claims: Hierbei handelt es sich um eine Reihe nicht obligatorischer vordefinierter Deklarationen. Es wird jedoch empfohlen, eine Reihe nützlicher interoperabler Deklarationen bereitzustellen. Einige von ihnen sind: iss (Absender), exp (Ablaufzeit), sub (Betreff), aud (Publikum) und andere.

Public Claims: Diese können von denjenigen, die JWTs verwenden, beliebig definiert werden. Um Kollisionen zu vermeiden, sollte man jedoch

[7] Tanmai Gopal, Hasura (2020)

in der IANA JSON Web Token Registry oder als URI definiert werden, der einen Antikollisions-Namensraum enthält.

Private Claims: Benutzerdefinierte Claims, werden erstellt, um Informationen zwischen Parteien auszutauschen, die sich auf ihre Verwendung einigen und die weder registrierte noch öffentliche Claims sind.

Alle Claims sind optional. Daher muss man nicht alle registrierten Claims verwenden. Im Allgemeinen kann die Payload eine beliebige Anzahl von Claims enthalten. Es wird jedoch empfohlen, die Informationen im JWT auf ein Minimum zu beschränken. Je größer die JWT, desto mehr Ressourcen werden für die (Decodierungs-) Codierung benötigt.

Abbildung 24: JWT-Payload[8]

Die Payload werden dann Base64Url-kodiert, um den zweiten Teil des JSON-Web-Tokens zu bilden.

5.1.2.3 Signatur

Die Signatur des JSON-Web-Tokens wird mithilfe der Base64-Codierung des Headers und der Payload sowie der angegebenen Signatur- / Verschlüsselungsmethode erstellt. Die Struktur wird durch JSON Web Signature (JWS) definiert, einen Standard, der gemäß RFC 7515 standardisiert ist. Damit die Signatur funktioniert, muss ein

[8] Tanmai Gopal, Hasura (2020)

geheimer Schlüssel verwendet werden, der nur der ursprünglichen Anwendung bekannt ist. Einerseits überprüft diese Signatur, ob die Nachricht unterwegs geändert wurde, andererseits kann die Verwendung eines mit einem privaten Schlüssel signierten Tokens sicherstellen, dass der Absender des JWT korrekt ist.

Um den Signaturteil zu erstellen, muss man den verschlüsselten Header und die verschlüsselte Payload mit dem im Header angegebenen Algorithmus abrufen und anschließend signieren.

Wenn man beispielsweise den HMAC SHA256-Algorithmus verwenden möchten, kann man eine Signatur auf folgende Weise erstellen:

Abbildung 25: JWT-Signatur

5.1.3 Funktionsweise des JSON-Web-Token

Wenn sich der Benutzer am Client bei der Authentifizierung erfolgreich mit seinen Anmeldeinformationen anmeldet, wird ein JSON-Web-Token zum Client zurückgegeben.

Da es sich bei dem Token um einen Berechtigungsnachweis handelt, muss besonders darauf geachtet werden das die Aufbewahrungszeit des Tokens die erforderliche Zeit nicht überschreitet.

Wenn ein Benutzer auf eine geschützte Route oder Ressource zugreifen möchte, sollte der Benutzeragent normalerweise das Trägerschema verwenden, um die JWT im Autorisierungsheader zu senden. Der Inhalt des Headers sollte folgendermaßen aussehen:

```
Authorization: Bearer <token>
```

In einigen Fällen kann dies ein zustandsloser Autorisierungsmechanismus sein. Die geschützte Route des Servers sucht im Autorisierungsheader nach einem gültigen JWT und ermöglicht dem Benutzer den Zugriff auf die geschützte Ressource.

Die folgende Abbildung zeigt den Anmeldevorgang, wie man ein JWT erhält und wie man mit JWT auf API oder Ressourcen zugreift:

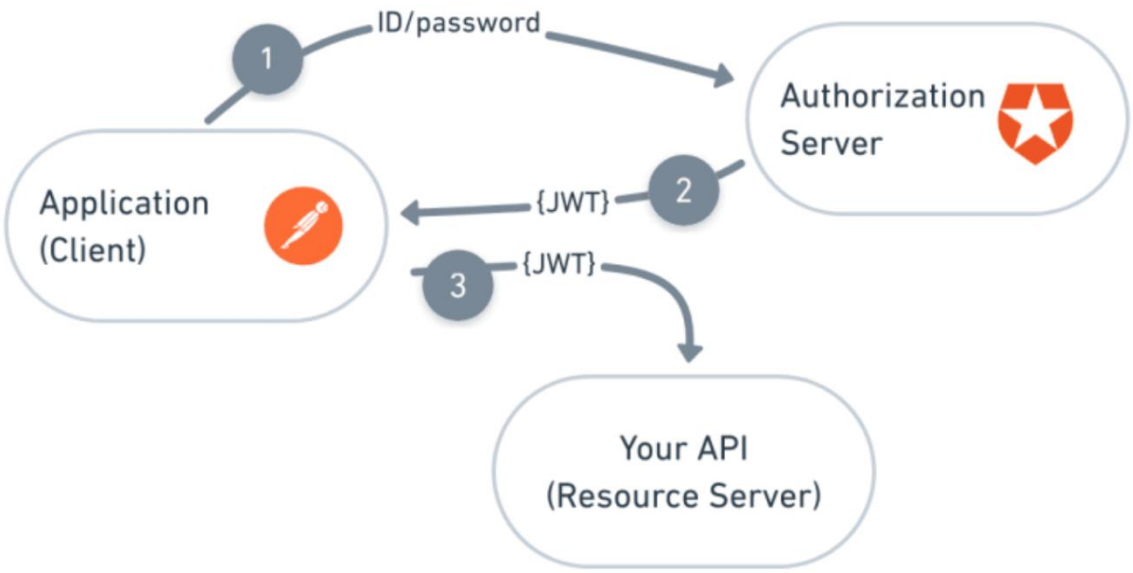


Abbildung 26: Anmeldevorgang[9]

[9] Tanmai Gopal, Hasura (2020)

1. Die Anwendung fordert die Autorisierung an, indem sie die erforderlichen Anmeldeinformationen (z.B. die Benutzer ID und das Passwort) übermittelt.
2. Der Autorisierungsserver gibt den JWT (Zugriffstoken) an die Anwendung zurück.
3. Die Anwendung verwendet den Zugriffstoken, um auf geschützte Ressourcen (z. B. API) zuzugreifen.

5.1.4 Funktionsweise eines Refresh-Tokens

Wenn der Access Token abgelaufen ist oder ungültig wird, kann ein Refresh Token verwendet werden, um einen neuen Access Token vom Autorisierungsserver anzufordern. Der Refresh Token ist ebenfalls für eine begrenzte Zeit gültig. Die Zeit wird normalerweise höher gewählt als beim Access Token.
Wie der Access Tokens wird der Refresh Token vom Autorisierungsserver an den Client gesendet, nachdem der Ressourcenbesitzer es autorisiert hat. Da der Refresh Token selbst bereits die Autorisierung des Ressourceneigentümers darstellt, muss für diese neue Access Token Anforderung keine weitere Autorisierung vom Ressourceneigentümer eingeholt werden.
Die Verwendung von Access Token und Refresh Token bietet den folgenden Vorteil: Die Lebensdauer des Access Token kann sehr kurzgehalten werden, sodass die Sicherheit des Protokolls erhöht wird.

Auch wenn der Ressourcenserver die Autorisierung nur bei der ersten Anforderung überprüft hat, hat dies keine Konsequenzen für den Widerruf der Berechtigung. Auf diese Weise kann der Client weiterhin auf die Daten und Dienste auf dem Ressourcenserver zugreifen, ohne dass Registrierungsdaten abgefragt werden müssen.

Die folgende Abbildung zeigt den Anmeldevorgang erneut mit der Verwendung eines Refresh-Tokens:

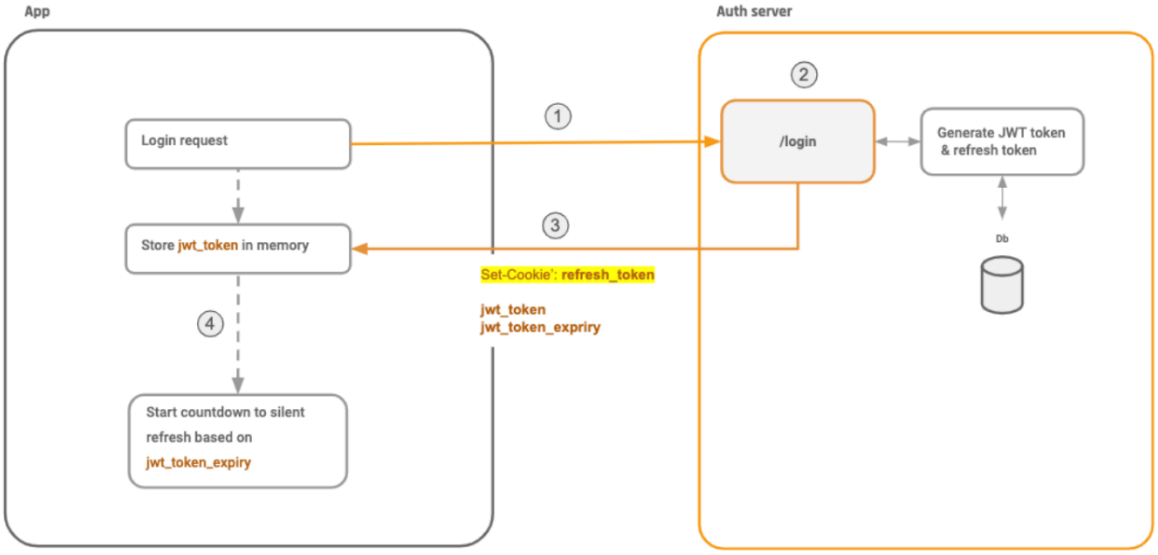


Abbildung 27: Anmeldung mit der Verwendung eines Refresh-Tokens[10]

1. Der Benutzer verwendet die Anmelde-API, um die Anmeldung aufzurufen.
2. Der Server generiert einen JWT-Token und einen Refresh-Token.
3. Der Server verwendet den Refresh-token, um das HttpOnly-Cookie zu setzen. „jwt_token“ und „jwt_token_expiry“ werden als Benutzerdaten im JSON Format an den Client zurückgegeben.
4. jwt_token wird gespeichert

Die folgende Abbildung zeigt einen stillen Refresh, nachdem der jwt_token abgelaufen ist:

[10] Tanmai Gopal, Hasura (2020)

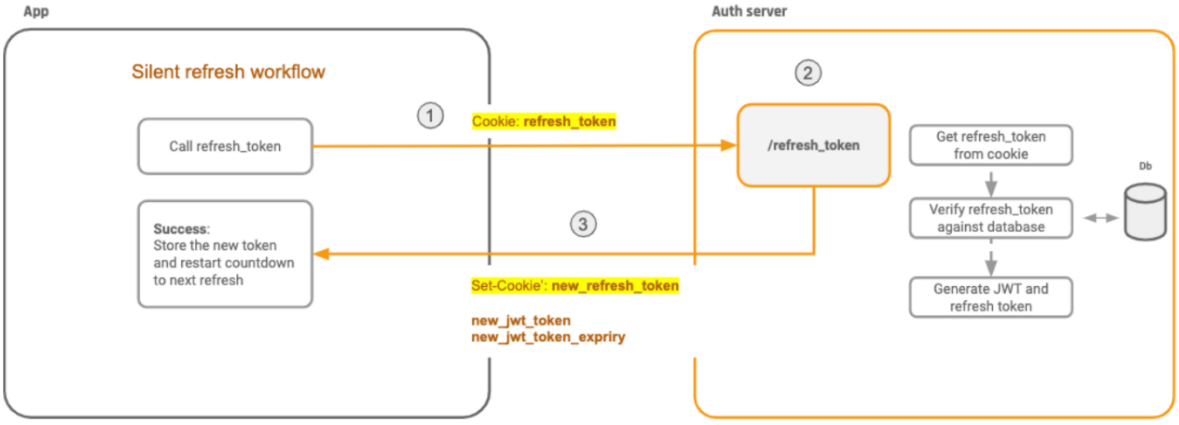


Abbildung 28: Stiller Refresh[11]

1. Refresh-Token wird vom Authentifizierungsserver angefordet.
2. Der Server liest das httpOnly-Cookie und wenn ein gültiges refresh_token gefunden wird, dann ...
3. ... gibt der Server das neue jwt_token und das jwt_token_expiry an den Client zurück und setzt über den gesetzten Cookie-Header ein neues Aktualisierungstoken-Cookie.

5.1.5 Warum sollte man JSON-Web-Token verwenden?

Da die JSON-Syntax mit weniger Zeichen als die Beschreibung mit XML auskommt, ist JWT kompakter als SAML. Dies macht JWT zu einer guten Wahl für die Bereitstellung in HTML- und HTTP-Umgebungen.

Die Security Assertion Markup Language, kurz SAML, ist ein XML-basiertes Open-Source-Framework zum Austausch von Authentifizierungs- und Autorisierungsinformationen. Es definiert eine Reihe von Regeln/Protokollen, sodass für den Zugriff auf die Webanwendung nur eine Anmeldung erforderlich ist.

[11] Tanmai Gopal, Hasura (2020)

JSON-Parser sind in den meisten Programmiersprachen üblich, da sie direkt Objekten zugeordnet werden. Im Gegensatz dazu gibt es keine natürliche Zuordnung von Dokumenten zu Objekten mit XML. Dies macht die Verwendung von JWT einfacher als die von SAML.

JWT wird im Internet in großem Umfang eingesetzt. Dies unterstreicht die einfache Verarbeitung von JSON-Web-Token auf der Clientseite auf mehreren Plattformen, insbesondere auf Mobilgeräten.

5.1.6 JWT-Code-Implementierung

```
@Configuration
public class WebSecurityConfiguration extends WebSecurityConfigurerAdapter {
    @Autowired
    private StudyBuddyUserDetailService userDetailsService;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.cors().configurationSource(request -> {
            var cors = new CorsConfiguration();
            cors.setAllowedOrigins(List.of("http://localhost:3000", "http://127.0.0.1:3000"));
            cors.setAllowedMethods(List.of("GET","POST", "PUT", "DELETE", "OPTIONS"));
            cors.setAllowedHeaders(List.of("*"));
            cors.setAllowCredentials(true);
            return cors;
        });

        http.authorizeRequests()
                // Endpoints ohne Authentifizierung
                .antMatchers( ...antPatterns: "/registrieren/**").permitAll()
                .antMatchers( ...antPatterns: "/check").permitAll()
                .antMatchers( ...antPatterns: "/login").permitAll()
                .antMatchers( ...antPatterns: "/refresh").permitAll()
                .antMatchers( ...antPatterns: "/sprachen").permitAll()
                .antMatchers( ...antPatterns: "/studienrichtungen").permitAll()
                .antMatchers( ...antPatterns: "/staedte").permitAll()
                .antMatchers( ...antPatterns: "/chat/**").permitAll()
                .antMatchers( ...antPatterns: "/topic/messages/**").permitAll()
                // Alle anderen
                .anyRequest().authenticated();

        http.csrf().disable();

        // Keine Sessions werden von spring security erstellt
        http.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);

        http.addFilterBefore(new JwtFilter(userDetailsService), UsernamePasswordAuthenticationFilter.class);
    }

    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }
```

Abbildung 29: Java-Code WebSecurityConfiguration

Zunächst möchte man verhindern, dass jeder den Endpunkt sehen kann. Daher erstellt man eine Konfiguration, um den Zugriff auf den Endpunkt einzuschränken. Beispielsweise wurde eine neue Klasse mit dem Namen *WebSecurityConfiguration* hinzugefügt, die die *WebSecurityConfigurerAdapte*r-Klasse in Spring Security implementiert.

@Configuration gibt an, dass die Klasse eine oder mehrere *@Bean-Methoden* deklarieren kann, und der Spring-Container sie verarbeiten kann, um zur Laufzeit Bean-Definitionen und Serviceanforderungen für diese Beans zu generieren. In Spring

werden die Objekte, die das Rückgrat der Anwendung bilden und vom Spring Container verwaltet werden, als Beans bezeichnet. Eine Bean ist ein Objekt, das vom Spring Container instanziiert, kompiliert und verwaltet wird.

Das Feld mit *@Autowired* bedeutet, dass es automatisch durch die Abhängigkeitsinjektionsfunktion von Spring verbunden wird.

Die Methode *authenticationManagerBean ()* erstellt ein Bean, der vom Spring-Container verwaltet wird.

Die Methode *configure* überschreibt man, um das Standardverhalten des HTTP-Sicherheitsobjekts zu ändern. Man teilt Spring Security mit, wie es mit verschiedenen APIs umgehen soll und welche Anforderungen authentifiziert werden müssen.

```java
package at.diplomarbeit.studybuddy.security;

import io.jsonwebtoken.*;
import io.jsonwebtoken.security.Keys;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import javax.crypto.SecretKey;
import java.util.*;


@Component
public class JwtProvider{

    private final SecretKey secretKey;
    private long validityInMilliseconds;

    @Autowired
    public JwtProvider(@Value("${security.jwt.token.expiration}")long validityInMilliseconds) {

        this.secretKey = Keys.secretKeyFor(SignatureAlgorithm.HS384);
        this.validityInMilliseconds = validityInMilliseconds;
    }

    public String createToken(String username) {
        Date now = new Date();

        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(now)
                .setExpiration(new Date(now.getTime() + validityInMilliseconds))
                .signWith(secretKey)
                .compact();
    }
```

Abbildung 30: Java-Code JwtProvider

@Component teilt Spring mit, dass diese Klasse als Kandidat für die automatische Erkennung bei der Verwendung von annotationsbasierter Konfiguration und Klassenpfad-Scanning betrachtet wird.

@Value weist Spring an, den angegebenen Schlüssel zu verwenden, um den Wert aus der Eigenschaftendatei abzurufen.

Die Methode *createToken* verwendet die jjwt-Bibliothek, um ein JWT-Token zu erstellen.

Der *Benutzername* ist das Subjekt, auch als Nutzlast bekannt. Es verwendet die Schlüsselsignatur in der Properties-Datei, und die Gültigkeit des Tokens wird auch in der Properties-Datei angegeben.

Das Format des generierten Tokens lautet *Header.Payload.Signature.*

```
package at.diplomarbeit.studybuddy.security;

import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.web.authentication.preauth.PreAuthenticatedAuthenticationToken;
import org.springframework.web.filter.GenericFilterBean;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;
import java.util.Optional;

public class JwtFilter extends GenericFilterBean {
    private static final String BEARER = "Bearer";
    private StudyBuddyUserDetailService userDetailsService;

    public JwtFilter(StudyBuddyUserDetailService userDetailsService ) {
        this.userDetailsService = userDetailsService;

    }

    @Override
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain filterChain)
            throws IOException, ServletException {
        //Check for Authorization:Bearer JWT
        String headerValue = ((HttpServletRequest)req).getHeader( s: "Authorization");
        getBearerToken(headerValue).ifPresent(token-> {
            //Pull the Username and Roles from the JWT to construct the user details
            userDetailsService.loadUserByJwtToken(token).ifPresent(userDetails -> {
                //Add the user details (Permissions) to the Context for just this API invocation
                SecurityContextHolder.getContext().setAuthentication(
                        new PreAuthenticatedAuthenticationToken(userDetails,  aCredentials: "", userDetails.getAuthorities()));
            });
        });

        filterChain.doFilter(req, res);
    }

    private Optional<String> getBearerToken(String headerVal) {
        if (headerVal != null && headerVal.startsWith(BEARER)) {
            return Optional.of(headerVal.replace(BEARER,  replacement: "").trim());
        }
        return Optional.empty();
    }
}
```

Abbildung 31: Java-Code JwtFilter

Die *JwtFilter*-Klasse implementiert die *GerericFilterBean*-Klasse.

Das Spring Web-Modul stellt die *GerericFilterBean*-Klasse als allgemeine Oberklasse für jeden Filtertyp bereit. Die JwtFilter-Klasse deklariert eine JWT-Filterklasse,

extrahiert die JWT-Informationen aus der HTTP-Anforderung und verwendet den Schlüssel „*secretKey*“ zur Authentifizierung der JWT. Diese Klasse überschreibt die Methode *doFilter*(). Diese Methode extrahiert das JWT-Token aus dem Anforderungsheader und delegiert die Authentifizierung an den injizierten *AuthenticationManager*.

Wenn das Token nicht gefunden wird, wird eine Exception geworfen, die die Verarbeitung der Anforderung stoppt. Man muss das *HttpServletRequest*-Objekt als Parameter an die *setAuthentication*()-Methode übergeben.

Das zurückgegebene Authentifizierungsobjekt wird dem *SecurityContext*-Objekt hinzugefügt.

5.2 Handhabung von Passwörtern mit Spring Boot und Spring Security

Wenn man den Benutzer auf der Serverseite authentifizieren will, muss man die folgenden Schritte ausführen:

1. Man holt sich den Benutzernamen und das Passwort des zu authentifizierenden Benutzers.
2. Man sucht den Benutzernamen im Speicher, normalerweise in der Datenbank.
3. Das Kennwort, das der Benutzer angegeben hat, wird mit dem Kennwort des Benutzers aus der Datenbank verglichen.

5.2.1 Hashing

Man muss sich mit der Tatsache auseinandersetzen, dass Benutzerkennwörter zum Vergleich während der Authentifizierung im System gespeichert werden müssen. Das Speichern des Passworts im Klartext in der Datenbank ist natürlich eine schlechte Idee.

Es sollte davon ausgegangen werden, dass der Angreifer das Kennwort verwenden kann, um die Datenbank zu stehlen oder über andere Methoden (z. B. SQL-Injection) auf das Kennwort zuzugreifen.

In diesem Fall kann der Angreifer das Kennwort sofort verwenden, um auf die Anwendung zuzugreifen. Daher muss man das Kennwort in einer Form speichern, die ein Angreifer nicht zur Authentifizierung verwenden kann.

Hashing löst das Problem des sofortigen Zugriffs auf das System mit offenen Passwörtern.

Hashing ist eine Einwegfunktion, welche die Eingabe in eine Reihe von Symbolen mit fester Länge umwandelt. Nach dem *Hashing* der Daten ist es schwierig, den Hash wieder in die ursprüngliche Eingabe umzuwandeln

Man muss das Kennwort in zwei Fällen hashen:

1. Wenn sich der Benutzer in der Anwendung registriert, hasht man das Kennwort und speichert es in der Datenbank.
2. Wenn der Benutzer sich authentifizieren möchte, hasht man das angegebene Kennwort und vergleicht es mit dem Kennwort-Hash aus der Datenbank.

Wenn ein Angreifer jetzt den Hashwert des Kennworts erhält, kann er das Kennwort nicht für den Zugriff auf das System verwenden. Jeder Versuch, den Klartext aus dem Hash-Wert zu finden, erfordert vom Angreifer einen enormen Aufwand. Wenn der Hash-Wert lang genug ist, können Brute-Force-Angriffe sehr teuer sein.

Die Brute-Force-Methode ist eine beliebte Angriffsmethode, die verwendet wird, um Passwörter zu finden oder Daten zu entschlüsseln. Es wird damit versucht, ein Passwort durch automatisches, zufälliges Ausprobieren herauszufinden. Je mehr Kombinationen getestet werden, desto größer die Erfolgschancen.

Mit Hilfe von Regenbogentabellen kann der Angreifer jedoch weiterhin erfolgreich sein. Die Regenbogentabelle ist eine vorberechnete Hash-Tabelle für viele Passwörter. Es gibt viele Regenbogentabellen im Internet, von denen einige Millionen von Passwörtern enthalten.

5.2.2 Salting des Passworts

Ein Hash-Wert, der aus einem Passwort und einem Salt besteht, kann verwendet werden, um Angriffe mit Regenbogentabellen zu verhindern. Das Salt ist eine zufällig generierte Folge von Bytes, die zusammen mit dem Passwort gehasht werden.

Immer wenn ein Benutzer versucht, sich zu authentifizieren, wird das Benutzerkennwort mit dem gespeicherten Salt gehasht, und das Ergebnis sollte mit dem gespeicherten Kennwort übereinstimmen.

Die Wahrscheinlichkeit, dass die Kombination aus Passwort und Salt in der Regenbogentabelle vorberechnet wird, ist sehr gering. Wenn das Salt lang und zufällig genug ist, ist es unmöglich, den Hash in der Regenbogentabelle zu finden.

5.2.3 Passwortverwaltung mit Spring Security

Alle Passwort-Encoder implementieren das PasswordEncoder-Interface.

Dieses Interface definiert die encode () -Methode zum Konvertieren des einfachen Passworts in eine verschlüsselte Form und die match () -Methode zum Vergleichen des einfachen Passworts mit dem verschlüsselten Passwort.

BCryptPasswordEncoder:

```
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder( strength: 10);
}
```

Abbildung 32: Java-Code für das Verschlüsseln des Passwortes

BCryptPasswordEncoder hat den Parameter *strength*. Der Standardwert in Spring Security ist 10. Es wird empfohlen, einen *SecureRandom* als Salt-Generator zu verwenden, da dieser eine kryptografisch starke Zufallszahl liefert.

Das Passwort, das dann in der Datenbank gespeichert wird, sieht wie folgt aus:

```
$2a$10$.7xgrcPjcohH1uC5XRT.7uKKF9RlGbGOwkcpLVpWkIV6obCjuoAsi
```

Abbildung 33: Gehashtes Passwort

6 Zusammenfassung

Das Ergebnis des Projekts ist ein Umfangreiches Web-Service, welches Benutzer Authentifizieren kann und die gewünschten Daten auf Anfrage liefert. Es interagiert dazu mit den Daten auf der eigens erstellten Datenbank. Es ist dabei nicht nur auf eine bestimmte Art von Clientsoftware beschränkt, sondern kann flexibel eingesetzt werden, was die Anbindung weiterer Clients ohne weitere Änderung oder Erweiterung der Serverseitigen Software ermöglicht.

Zusätzlich dazu wurden Clientseitige Implementierungen verwirklicht, welche nicht nur auf Funktionalität, sondern auch auf eine angenehme Benutzererfahrung ausgelegt sind. Dies bietet einen vollständigen Web- und App-Auftritt, welcher in der heutigen Zeit für Applikationen dieser Art der Standard ist. Erweiterung wäre in der Entwicklung einer Applikation für Android Systeme möglich.

Es wurde bei allen Schritten der Entwicklung auch auf die Sicherheit geachtet, um Benutzer und deren Daten zu schützen. Dafür wurde nach ausgiebiger Recherche entschieden was die größten Bedrohungen sind und was getan werden muss, um den bestmöglichen Schutz zu bieten.

Die Entwicklung von Studybuddy hat einen erheblichen Lernerfolg im Zusammenhang mit den verwendeten Technologien, der Projektentwicklung und der Umsetzung eines Softwareprojekts als Ganzes gebracht.

7 Abbildungsverzeichnis

8 Tabellenverzeichnis

9 Literaturverzeichnis

Baeldung. 2020. Spring Boot Starters. [Online] 28. Jänner 2020. https://www.baeldung.com/spring-boot-starters.

Business Insider. 2019. User Interface. [Online] 1. Jänner 2019. https://www.businessinsider.de/gruenderszene/lexikon/begriffe/user-interface/.

Fuchs, Elmar. 2018. *SQL Grundlagen und Datenbankdesign.* s.l. : Herdt Verlag, 2018, S. 56.

Gopal, Tanmai. 2020. Hasura. [Online] 03. Jänner 2020. https://hasura.io/blog/best-practices-of-using-jwt-with-graphql/.

Spring. 2021. Spring Security. [Online] 2021. https://spring.io/projects/spring-security.

Wikipedia. 2021. Swift. [Online] 24. April 2021. https://de.wikipedia.org/wiki/Swift_(Programmiersprache).

10 Anhang

Tabelle 1: Navigation Controller Methoden

Methode	Beschreibung
init(rootViewController: UIViewController)	Gibt einen neu erstellten Navigations-Controller mit dem angegebenen Root-View-Controller zurück.
init(navigationBarClass: AnyClass?, toolbarClass: AnyClass?)	Initialisiert und gibt einen neu erstellten Navigations-Controller mit der angegebenen Klasse für die Navigationsleiste zurück.
func setViewControllers([UIViewController], animated: Bool)	Wird verwendet, um die View-Controller zu ersetzen, die derzeit vom Navigations-Stack verwaltet werden.
func pushViewController(UIViewController, animated: Bool)	Mit dieser Methode wird der angegebene View Controller auf den Stack des Empfängers geschoben und die Anzeige aktualisiert.
func popViewController(animated: Bool) -> UIViewController?	Gibt einen View-Controller zurück, der aus dem Navigations-Controller gepoppt (entfernt) wird. Diese Methode wird automatisch ausgelöst,

	wenn der Benutzer in der Navigationsleiste auf die Schaltfläche "Zurück" tippt.
func popToRootViewController(animated: Bool) -> [UIViewController]?	Es werden alle View-Controller auf dem Stapel außer dem Root-View-Controller gepoppt und dann die Anzeige aktualisiert.
func popToViewController(UIView-Controller, animated: Bool) -> [UIViewController]?	Diese Methode wird verwendet, um zum angegebenen View-Controller im Navigationsstapel zu springen.
func setNavigationBarHidden(Bool, animated: Bool)	Diese Methode wird verwendet, um die mit dem Navigations-Controller verbundene Navigationsleiste auszublenden.
func setToolbarHidden(Bool, animated: Bool)	Diese Methode wird verwendet, um die mit dem Navigations-Controller verbundene Symbolleiste auszublenden.
func show(UIView-Controller, sender: Any?)	Diese Methode wird verwendet, um einen angegebenen View-Controller anzuzeigen.

Tabelle 2: Table-View Delegate Methoden:

Methode	Beschreibung
func tableView(UITableView, willDisplay: UITableViewCell, forRowAt: IndexPath)	Der Table-View benachrichtigt diesen Delegaten, wenn er im Begriff ist, eine Zelle für eine bestimmte Zeile zu zeichnen.
func tableView(UITableView, willSelectRowAt: IndexPath) -> IndexPath?	Der Table-View benachrichtigt diese Delegate-Methode, wenn die angegebene Zeile ausgewählt werden soll.
func tableView(UITableView, didSelectRowAt: IndexPath)	Dieser Delegat wird benachrichtigt, wenn die angegebene Zeile des Table-Views ausgewählt wird.
func tableView(UITableView, willDeselectRowAt: IndexPath) -> IndexPath?	Dieser Delegat wird benachrichtigt, wenn die jeweilige Zelle abgewählt werden soll.
func tableView(UITableView, didDeselectRowAt: IndexPath)	Dieser Delegat wird benachrichtigt, wenn die bestimmte Zeile abgewählt wird.

func tableView(UITableView, viewFor-HeaderInSection: Int) -> UIView?	Diese Delegate-Methode gibt eine UIView zurück, die den Kopf des Table-Views darstellt.
func tableView(UITableView, viewFor-FooterInSection: Int) -> UIView?	Diese Delegate-Methode gibt die UI-View zurück, die die Fußzeile des Table-Views darstellt.
func tableView(UITableView, willDis-playHeaderView: UIView, forSection: Int)	Diese Delegate-Methode wird be-nachrichtigt, wenn der Table-View den Headerview für den jeweiligen Abschnitt anzeigen soll.
func tableView(UITableView, willDis-playFooterView: UIView, forSection: Int)	Diese Methode wird benachrichtigt, wenn der Table-View die Fußzeilen-ansicht für den jeweiligen Abschnitt anzeigen wird.
func tableView(UITableView, height-ForRowAt: IndexPath) -> CGFloat	Diese Delegaten Methode gibt die Höhe für die Zeile zurück.
func tableView(UITableView, height-ForHeaderInSection: Int) -> CGFloat	Diese Delegaten Methode liefert die Höhe der Kopfzeile des Abschnitts in der Table-View.

func tableView(UITableView, heightForFooterInSection: Int) -> CGFloat	Diese Methode gibt die Höhe für die Fußzeile eines bestimmten Abschnitts in der Table-View zurück.
func tableView(UITableView, estimatedHeightForRowAt: IndexPath) -> CGFloat	Fragt den Delegierten nach der geschätzten Höhe der Zeile an einer bestimmten Stelle.
func tableView(UITableView, estimatedHeightForHeaderInSection: Int) -> CGFloat	Fragt den Delegierten nach der geschätzten Höhe der Kopfzeile an einer bestimmten Stelle.
func tableView(UITableView, estimatedHeightForFooterInSection: Int) -> CGFloat	Fragt den Delegierten nach der geschätzten Höhe für die Fußzeile in dem jeweiligen Abschnitt

Tabelle 3: Table-View DataSource Methoden:

Methoden	Beschreibung
func tableView(UITableView, numberOfRowsInSection: Int) -> Int	Diese Methode gibt die Anzahl der Zeilen zurück, die im Bereich des Table-Views angezeigt werden sollen.
func numberOfSections(in: UITableView) -> Int	Diese Methode liefert die Anzahl der Abschnitte, die in der Table-View angezeigt werden sollen.
func tableView(UITableView, cellForRowAt: IndexPath) -> UITableViewCell	Diese Methode liefert das Objekt einer UITableViewCell, die den aktuellen Inhalt einer bestimmten Zeile in der Table-View anzeigt. Diese Methode fügt die Zelle für eine bestimmte Zeile in der Table-View ein.
func tableView(UITableView, titleForHeaderInSection: Int) -> String?	Diese Methode gibt eine Zeichenkette zurück, die den Titel der Kopfzeile im Abschnitt des Table-Views darstellt.
func tableView(UITableView, titleForFooterInSection: Int) -> String?	Diese Methode gibt eine Zeichenkette zurück, die den Titel der Fußzeile im Abschnitt des Table-Views darstellt.

func tableView(UITableView, canEditRowAt: IndexPath) -> Bool	Fragt die DataSource, ob die jeweilige Zeile editierbar ist oder nicht.
func tableView(UITableView, canMoveRowAt: IndexPath) -> Bool	Bittet die DataSource zu prüfen, ob die bestimmte Zeile an eine andere Stelle in der Table-View verschoben werden kann.
func tableView(UITableView, moveRowAt: IndexPath, to: IndexPath)	Diese Methode verschiebt die bestimmte Zeile an eine andere Stelle in der Table-View.
func sectionIndexTitles(for: UITableView) -> [String]?	Gibt das Array mit der Zeichenkette zurück, die die Titel für die Abschnitte in der Table-View enthält.