ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Konzeption und Realisierung eines mobilen, plattformunabhängigen Studienführers

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
David Damaschk
david.damaschk@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Nicolas Mundbrod

2014

Fassung 5. März 2014

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Deutschland ist auf dem Weg zur Informationsgesellschaft. Dies äußert sich unter anderem anhand der zunehmenden Verfügbarkeit und steigenden Bandbreite von Internetverbindungen, durch die auch für heute gängigen Smartphones Informationen mobil in hoher Güte zur Verfügung stehen. Dadurch wird auch die Entwicklung von mobilen Angeboten ermöglicht, welche das Studium an Universitäten erleichtern können.
Zurzeit ist die Organisation eines Studiums an einer Universität aufgrund unterschiedlicher Anforderungen innerhalb der Studiengänge sehr komplex. Durch Unklarheiten entstehende Fragen können dabei oft nur von unterschiedlichen Ansprechspartnern beantwortet werden.

Um derartige Probleme zu lösen, wird im Rahmen dieser Arbeit ein entsprechendes Angebot in Form einer mobilen Applikation für Studenten der Universität Ulm entwickelt. Dazu werden bereits existierende Applikationen verschiedener Universitäten analysiert und Interviews mit Studenten geführt, damit das bisherige Angebot der Universität Ulm sinnvoll ergänzt werden kann.

Die Studenten sollen ihren Studienverlauf in der Applikation abbilden und Antworten auf Fragen zum Studium finden können. Für die Umsetzung der Applikation kommt die Technologie Phonegap zum Einsatz, mit der plattformunabhängige Anwendungen entwickelt werden können.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Deutschland ist auf dem Weg zur Informationsgesellschaft [RG11]. Schon heute erleichtert zum Beispiel das Internet den Informationszugang erheblich. Ein Leben ohne Internet ist daher für viele Menschen unvorstellbar geworden und ist in ihrem Alltag fest verankert [Con10]. Durch die immer stärker wachsende Zahl an Smartphones können Informationen beinahe zu jeder Zeit und an jedem Ort abgerufen werden [Mob13]. Somit ergeben sich für alle Lebensbereiche neue Möglichkeiten, das Leben zu gestalten. Das gilt insbesondere auch im Studium, vor allem da die Anzahl der Studenten von Jahr zu Jahr in Deutschland ansteigt [Anz13]. Dies wurde schon von zahlreichen Universitäten erkannt und es werden auch entsprechende mobile Applikationen (Apps) angeboten, Informationen für das Studium über Smartphones zu beziehen [App14]. Das Angebot

an Apps ist hierbei sehr vielfältig und neben einer App für den aktuellen Speiseplan [Men13a] oder einem *Campusplan* [HSM13] gibt es noch mehr Möglichkeiten, Informationen und Funktionen rund um das Studium in Form einer App anzubieten. In dieser Bachelorarbeit wird das bestehende und mögliche Angebot an Apps für die Universität Ulm analysiert und ein mobiler Studienführer in Form einer App entwickelt, der das Angebot sinnvoll ergänzt.

1.2 Problemstellung

Die Organisation des Studiums an einer Universität kann sehr komplex sein. Das liegt mitunter daran, dass jeder Studiengang seine eigenen Anforderungen besitzt. Ist die Anzahl der zu erreichenden Leistungspunkte studienübergreifend noch grundsätzlich gleich, hängt hingegen schon die Anzahl der zu erbringenden Mindestleistungspunkte im jeweiligen Fachsemester von den einzelnen *fachspezifischen Prüfungsordnungen* ab. Es können dabei je nach Studiengang mehrere verschiedene aktive Prüfungsordnungen geben, die jeweils das Studium anders regeln. Dazu gehört beispielsweise die Verteilung der Leistungspunkte für eine Veranstaltung oder gar die Vorschrift, welche Veranstaltungen belegt werden müssen. Prüfungsordnungen können sich sogar im Nachhinein ändern, sodass es nötig ist, sich darüber zu informieren und diese zu berücksichtigen. Neben fachspezifischen Prüfungsordnungen gibt es auch die sogenannte *Allgemeine Prüfungsordnung*, in der zusätzliche Bestimmungen zum Studium geregelt sind. Prüfungsordnungen müssen rechtssicher sein, wodurch die Verständlichkeit durch eine komplexe Sprache leidet. Diese dürfen ebenfalls nicht zu konkret formuliert werden, da ansonsten bei geringen Änderungen die Prüfungsordnungen umgeschrieben werden müssen und es dadurch zu Verwirrungen unter den Studenten kommen kann. Ebenfalls wird die Übersicht durch die Aufteilung in eine allgemeine und fachspezifische Prüfungsordnung schwieriger. Durch die abstrakte Formulierung der Informationen und Abhängigkeiten als Fließtext entgehen Studenten dabei wichtige Informationen zum Ablauf ihres Studiums. Dazu zählt beispielsweise das Wissen um die Anrechenbarkeit von Praktika und Berufsausbildungen. Antworten auf verschiedene Fragen stehen teils auf mehreren Seiten verteilt oder sind nur auf Nachfrage an die entsprechenden Mitarbeiter

herauszufinden. Dies führt zu einem nicht zu unterschätzenden Zeitverlust sowohl für Studenten als auch für die Mitarbeiter. Für Letztere ist der Aufwand vor allem bei immer wiederkehrenden Fragen recht hoch.

1.3 Zielsetzung

Das Ziel dieser Bachelorarbeit soll es sein, unter Berücksichtigung der in Kapitel 1.2 genannten Probleme eine mobile Applikation mit dem Namen *Ulm University Study Guide* (ULMUS) zu entwickeln, mit der das Planen und die Organisation des Studiums einfacher und übersichtlicher zu gestalten ist. In der App soll es möglich sein, passend zum Studiengang seine Prüfungsleistungen zu verwalten. Daneben sollen auch relevante Informationen zum Studienverlauf wie zum Beispiel die Anzeige der benötigten Mindestleistungspunkte pro Semester, angeboten werden. Zuletzt soll eine FAQ integriert werden, mit der Antworten auf Fragen während des Studiums einfach und schnell nachgeschlagen werden können. Die App soll zudem neue Informationen erhalten können, sodass auf Änderungen von Studieninhalten schnell reagiert werden kann. Außerdem soll sie eine hohe Plattformunabhängigkeit besitzen, sodass sie unter möglichst vielen Betriebssystemen laufen kann, wodurch die Zeit für die Entwicklung und Wartung verringert wird.

1.4 Aufbau der Arbeit

Der Aufbau dieser Arbeit ist in Abbildung 1.1 visuell dargestellt.
In Kapitel 1 wird in die Thematik der Möglichkeiten mobiler Apps in der heutigen Zeit, eingeführt. Dabei werden Probleme genannt, die im universitären Bereich vorhanden sind und abschließend Ziele formuliert, um diese Probleme in Form einer mobilen App zu lösen.
Kapitel 2 vermittelt das benötigte fachliche Hintergrundwissen, welches für das Thema Studium wichtig ist. Anschließend werden Anwendungsfälle vorgestellt, die verschiedene Abläufe des Studiums aufzeigen.

In Kapitel 3 werden aktuelle Apps für Universitäten analysiert und anschließend Kurzinterviews mit Studenten vorgestellt. Daraus werden die funktionalen und nicht-funktionalen Anforderungen für die App erläutert.
In Kapitel 4 wird die Architektur der App beschrieben und ein Datenmodell diskutiert. Zusätzlich werden Mockups vorgestellt, die die Benutzeroberfläche zeigen und designtechnische Aspekte erläutern, mit deren eine optimale Bedienung der App erreicht wird.
In Kapitel 5 werden die verwendeten Technologien und weitere Grundlagen für die Umsetzung vorgestellt. Dazu werden anhand von Programmierbeispielen die Umsetzungen in der App und auf der Server-Seite erläutert.
In Kapitel 6 werden die Ergebnisse einer ersten Evaluation, mit dem Fokus auf die Bedienbarkeit, vorgestellt. Anschließend werden Optimierungen an benötigten Stellen diskutiert.
Kapitel 7 fasst die Bachelorarbeit zusammen und stellt zukünftige Erweiterungsmöglichkeiten von ULMUS vor.

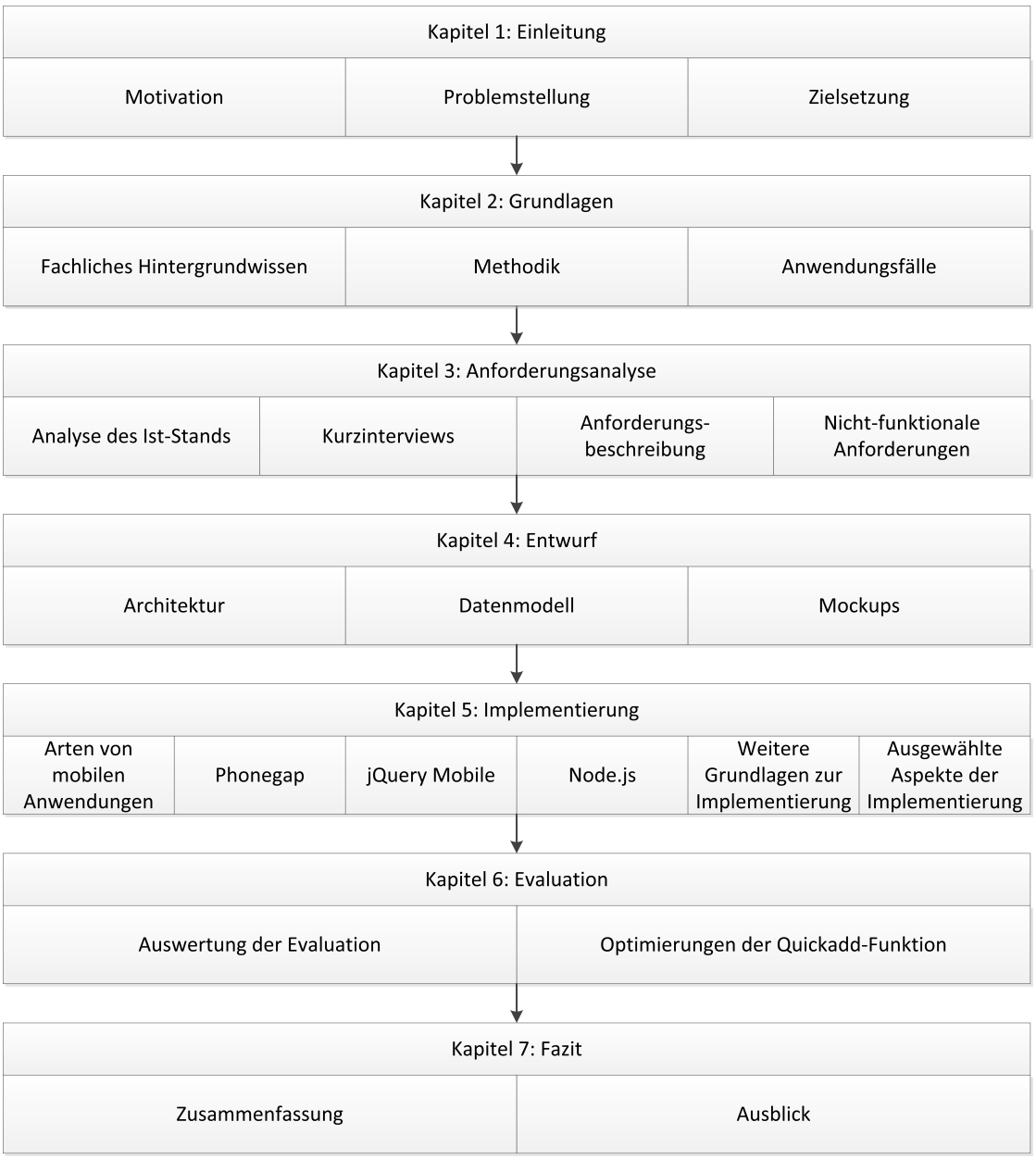


Abbildung 1.1: Aufbau der Arbeit

2

Grundlagen

In Kapitel 2.1 werden zentrale Begriffe erklärt, die für das Verständnis rund um das Thema Studium unerlässlich sind. Dabei werden für synonyme Begriffe, die in verschiedenen Studiengängen verwendet werden, ein Begriff ausgewählt, das in der gesamten Arbeit einheitlich verwendet wird.
In Kapitel 2.2 wird die Vorgehensweise für die Erstellung dieser Arbeit erläutert. Abschließend werden in Kapitel 2.3 Anwendungsfälle für verschiedene Szenarien zum Studienverlauf beschrieben.

2.1 Fachliches Hintergrundwissen

Bachelor: Der Bachelorstudiengang (BA) ist für viele Studiengänge der Einstieg ins Studium, in dem grundlegende Kenntnisse aus der Studienfachrichtung erlangt werden.

Master: Der Masterstudiengang (MA) ist ein aufbauendes Studium und vertieft die Kenntnisse, die im Bachelorstudiengang erworben wurden.

Prüfungsordnung: In einer Prüfungsordnung (PO) werden die Rahmenbedingungen und Vorgaben für ein Studium eines konkreten Studiengangs an einer Universität (Uni) festgelegt. Neben einer *Fachspezifischen PO (FSPO)* gibt es auch eine *Allgemeine PO (APO)*. Die APO ist eine allgemeine einheitliche Basis für alle Studiengänge und eine FSPO ergänzt diese um die Inhalte und den Aufbau des Studiums sowie die Leistungsanforderungen und die Ziele für die jeweiligen Studiengänge. Die FSPO umfasst unter anderem den zu erreichenden akademischen Grad, die Dauer der Regelstudienzeit, Fristen für die Anmeldung von Prüfungen und die Studieninhalte des Studiengangs. Dabei können die Ziele und Anforderungen aus mehreren sich ähnelnden Studiengängen in einer Prüfungsordnung zusammengefasst werden. Aufgrund von ständigen Anpassungen der POs, gibt es Versionen der APOs und FSPOs auf Basis von Jahreszahlen. Für Studierende gilt im Normalfall die Version, die zu ihrem Studienbeginn aktuell war. Ob ein Wechsel in eine spätere PO möglich oder zwingend ist, kann sich allerdings von PO zu PO unterscheiden. Ein Aufbau der verschiedenen POs ist zur Übersicht in Abbildung 2.1 visuell dargestellt.
Um die Bedeutung der PO für den Studienverlauf aufzuzeigen, werden die Änderungen der Prüfungsordnungen in den letzten Jahren für die Fachrichtung Informatik näher erläutert.

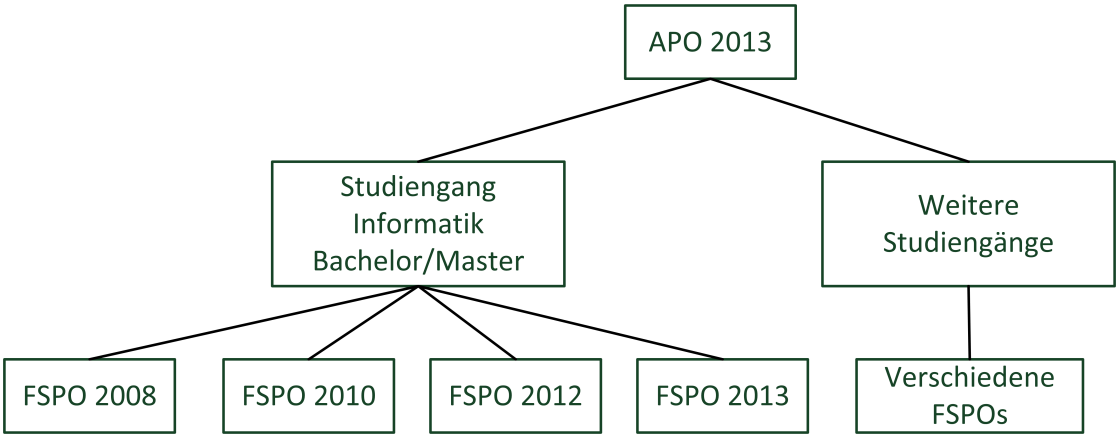


Abbildung 2.1: Verschiedene Prüfungsordnungen in den Studiengängen

Wie in Abbildung 2.1 zu sehen ist, gab es in den letzten Jahren für den Studiengang Informatik im Bachelor wie auch im Master die FSPO 2008, FSPO 2010, FSPO 2012 und die FSPO 2013. Dabei ergaben sich im direktem Vergleich der FSPO 2008 und FSPO 2010 unter anderem folgende Änderungen [po214a]:

- Änderungen von Modulleistungspunkten
- Neue Endnotenberechnung
- Abgeschwächte Fristen für Leistungspunkte
- Änderung der Pflichtmodule in der Mathematik
- Verpflichtende Präsentation der Abschlussarbeit

Im Vergleich zu den Änderungen in der FSPO 2010 ist die Grundstruktur der FSPO 2012 sehr ähnlich zur FSPO 2010 geblieben. Erwähnenswerte Neuerungen sind im Folgenden [po214b]:

- Erhöhung der Leistungspunkte in den Veranstaltungen aus der angewandten Mathematik
- Bessere Studienplanung durch größere Wahlmöglichkeiten in den Kernfächern
- Module bestehen nun aus Veranstaltungen und deren Übungsformen. Mehrere Module bilden dabei ein Fach

Hierbei ist anzumerken, dass sich dennoch auch Änderungen für die Studenten aus der FSPO 2010 durch die Einführung der FSPO 2012 ergaben. Beispielsweise wurden die Leistungspunkte für die Veranstaltungen aus der Angewandten Mathematik auch für die Studenten erhöht, die nach der FSPO 2010 studieren.
Die FSPO 2013 ist mit der FSPO 2012 weitgehend identisch. Es wurden hier nur die Studienanforderungen für den neuen Studiengang Software Engineering Master hinzugefügt.

Regelstudienzeit: Die Regelstudienzeit gibt an, wie viele Semester für ein Studium benötigt werden. Im Normalfall sind das im Bachelor sechs oder acht Semester und im Master zwei bis vier Semester.

Semester: Das Studienjahr an der Universität Ulm ist in Semestern unterteilt. Dabei wird das Semester vom 01.10. bis 31.03. als Wintersemester (WS) bezeichnet und das Semester vom 01.04. bis 30.09. als Sommersemester (SS).

Fachsemester: Fachsemester entsprechen der Studiendauer in einem bestimmten Studiengang. Urlaubssemester werden dabei nicht hinzugezählt.

Hochschulsemester: Hochschulsemester entsprechen der gesamten Studienzeit. Das heißt, hierbei zählen alle Semester, in die ein Student an der Uni immatrikuliert war/ist.

Urlaubssemester: Urlaubssemester sind Semester, in der Studierende vom ordnungsgemäßen Studium beurlaubt sind. Dabei sind sie weiterhin an der Uni immatrikuliert, können aber keine Prüfungen und Leistungsnachweise ablegen, wobei Abschlussarbeiten davon ausgeschlossen sind. Gründe für Urlaubssemester können unter anderem ein Praktikum, Krankheit, Freiwilligendienst oder ein Studium an einer ausländischen Hochschule sein [Beu14].

Veranstaltung: Es existieren verschiedene Lehrformen, die als Veranstaltungen bezeichnet werden können. Als Vorlesung wird eine Veranstaltung genannt, in der in der Regel ein Professor oder Dozent den Inhalt der Veranstaltung vor vielen Studenten erklärt. In Seminaren können Studenten in kleineren Gruppen zu verschiedenen Themen ihr Fachwissen vertiefen. Übungen sind vorlesungsunterstützende Lerneinheiten, in denen der Inhalt der im Semester parallel stattfindenden Vorlesung durch Übungsaufgaben vertieft wird. Es existieren noch weitere Lehrformen wie zum Beispiel Berufspraktika, in der die Kenntnisse in der Praxis angewendet werden.

Leistungspunkte: Leistungspunkte werden für erfolgreich besuchte Veranstaltungen vergeben. In der Regel spiegelt ein Leistungspunkt etwa 25-30 Stunden Arbeit wider und im Normalfall sollen in jedem Semester 30 Punkte erreicht werden [BaM10].

Module: Module (oder auch Fächer in anderen Prüfungsordnungen) vereinen mehrere Veranstaltungen, die sich inhaltlich ähneln oder aufeinander aufbauen können. Ein Modul gilt als bestanden, falls alle Veranstaltungen erfolgreich bestanden wurden. Eine Veranstaltung gilt nach erfolgreicher Teilnahme als bestanden, wobei die erfolgreiche Teilnahme auch eine Prüfung mit einschließen kann.
Neben den Modulen, die nur Veranstaltungen enthalten, existieren noch weitere Möglichkeiten, Module zu unterteilen. Diese weiteren Möglichkeiten müssen dabei in der späteren Umsetzung der App berücksichtigt werden. Am Beispiel des Informatik Bachelorstudiums (FSPO 2012) lässt sich ein möglicher Unterschied verdeutlichen. Im Bachelorstudium muss ein Anwendungsfach (AF) belegt werden. Im AF werden Veranstaltungen aus anderen Studienrichtungen wie zum Beispiel Biologie oder Physik besucht. Das AF besteht dabei aus verschiedenen Untermodulen, aus denen genau eines ausgewählt werden muss (siehe Abbildung 2.2).

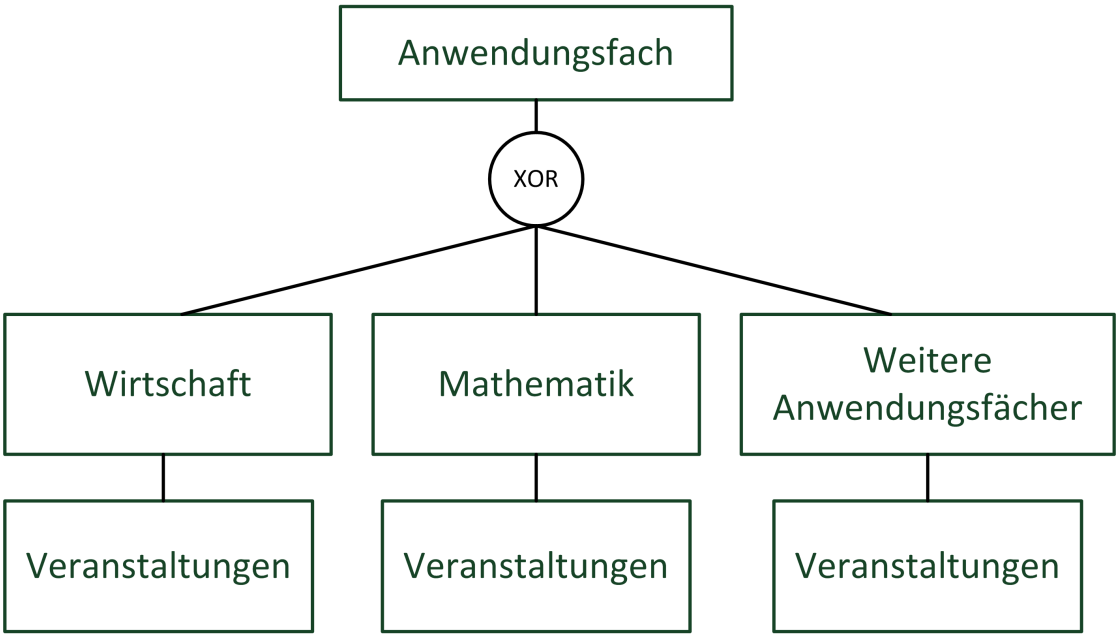


Abbildung 2.2: Anwendungsfächer unterteilt in weitere Module

2.2 Methodik

In diesem Kapitel wird die Vorgehensweise für die Erstellung der App beschrieben. Sie ist an den *Design Science Research Cycles* aus [Hev07] angelehnt und in Abbildung 2.3 visuell dargestellt.
Der *Relevanzzyklus* (Relevance Cycle) initiiert dabei die Arbeit und erarbeitet Probleme und Chancen für eine definierte Umgebung. Diese ist in der Arbeit das Studium an Universitäten, insbesondere an der Universität Ulm. Aus diesem Zyklus gehen die Anforderungen hervor, die im Bereich *Design Science Research* umgesetzt werden. Außerdem werden hier die Akzeptanzkriterien für die Evaluation der Umsetzung definiert. Im *Rigorzyklus* wird sichergestellt, dass die zu entwickelnde App innovativ ist und einen Mehrwert für die Umgebung bietet. Dabei werden auf die benötigten Werkzeuge, Kenntnisse und Erfahrungen zurückgegriffen, die für die Umsetzung benötigt werden.
Im *Designzyklus* (Design Cycle) wird die App mithilfe der Ergebnisse aus dem Relevanzzyklus und Rigorzyklus umgesetzt. Anschließend wird im Relevanzzyklus die Umsetzung getestet und die Ergebnisse im Designzyklus evaluiert. Durch die Evaluation kann es zu

Änderungen der Anforderungen kommen (Relevanzzyklus), wodurch für die Umsetzung im Designzyklus wieder auf die bestehenden Kenntnisse (Rigorzyklus) zurückgegriffen werden muss. Somit werden diese Zyklen während der Erstellung dieser Arbeit iterativ durchlaufen.

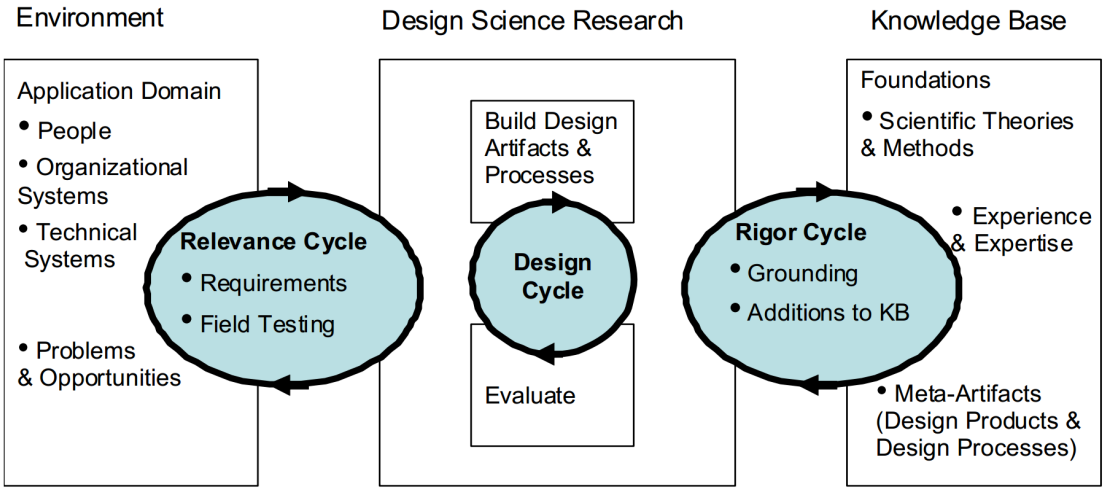


Abbildung 2.3: Vorgehensweise der Arbeit aus [Hev07]

2.3 Anwendungsfälle

Das Studium kann von Student zu Student unterschiedlich verlaufen und ist in einigen Fällen nicht linear. In diesem Kapitel sollen daher einige repräsentative Anwendungsfälle vorgestellt werden, die aufzeigen, wie verschieden das Studium von Studenten an der Universität verläuft und daher besonders auch diese Fälle in einer App berücksichtigt werden müssen.

2.3.1 Organisation des Studiums:

Im Bachelor Informatikstudium besuchen die Studenten in den ersten Semestern für gewöhnlich die Pflichtveranstaltungen. In den höheren Semestern stehen den Studenten in den Modulen Angewandte Mathematik und Schwerpunktmodul Informatik verschiedene

Wahlveranstaltungen zur Verfügung. Für das Modul Additive Schlüsselqualifikationen stehen den Studenten ebenfalls verschiedene Veranstaltungen zur Verfügung und im Anwendungsfach wird eine Vertiefung ausgewählt. Im 6. Semester wird für gewöhnlich die Bachelorarbeit geschrieben. Nach dem Bachelorstudium können Studenten mit einem guten Abschluss zum Beispiel im Masterstudiengang Informatik weiter studieren. Hier stehen ihnen wesentlich mehr Wahlveranstaltungen zur Verfügung, sodass vertiefte Kenntnisse in ausgewählten Gebieten erworben werden können.

2.3.2 Änderungen der Prüfungsordnungen:

In der FSPO 2012 des Studiengangs BA Informatik werden die Wahlveranstaltungen aus dem Modul Angewandte Mathematik für Informatiker von vier LP auf sechs LP angehoben. Dadurch müssen Studenten nicht mehr drei Veranstaltungen besuchen, sondern nur noch zwei. Diese Änderung betrifft auch Studenten der FSPO 2010. Das heißt für Studenten, die schon drei Veranstaltungen besucht haben, dass die schlechteste Prüfung in das Zusatzfach verschoben wird und sie haben die Möglichkeit, sich diese im Master anrechnen zu lassen.

2.3.3 Studiengangwechsel während des ersten Semesters:

Student Philipp beginnt sein Informatikstudium im Wintersemester an der Universität Ulm. Jedoch wird ihm in den ersten zwei Wochen des Semesters bewusst, dass er von den möglichen Anwendungsfächern, von denen eines in der Informatik belegt werden muss, nicht begeistert ist und er daher zum Studiengang Medieninformatik wechseln möchte. Dazu wendet er sich an das Studiensekretariat. Da die Frist zum Studiengangwechsel noch nicht überschritten wurde, kann er sich im gleichen Semester in den Studiengang Medieninformatik umschreiben lassen. Da die Veranstaltungen in den Studiengängen Informatik und Medieninformatik im ersten Semester bis auf wenige Ausnahmen gleich sind, beschränkt sich der Aufholbedarf des Stoffes von neuen Veranstaltungen auf ein Minimum.

2.3.4 Studiengangwechsel in höheren Fachsemestern:

Studentin Jasmin studiert seit vier Semestern BA Informatik an der Uni Ulm. Sie hat dabei während des Studiums die Pflichtprüfung zur Veranstaltung Technische Informatik 2 bereits zweimal nicht bestanden. Nach einem möglichen weiteren Fehlversuch wird Jasmin zwangsläufig exmatrikuliert und könnte auch nicht an anderen Universitäten Informatik oder verwandte Studiengänge studieren. Da Jasmin kein Risiko eingehen möchte und sie weiß, dass die Veranstaltung Technische Informatik 2 keine Pflichtveranstaltung im Studiengang Medieninformatik ist, beschließt sie den Studiengang zu wechseln. Dazu wendet sie sich an das Studiensekretariat. Da Jasmin bereits zahlreiche Veranstaltungen gehört hat, muss sie zusätzlich noch den Prüfungsausschuss konsultieren und dort ihre Studien- und Prüfungsleistungen vorlegen. Der Vorsitzende des Prüfungsausschusses schreibt die Leistungen in den neuen Studiengang um. Leistungen, die dabei in beiden Studiengängen erforderlich sind und schon im Studiengang Informatik erbracht wurden, müssen dadurch nicht noch einmal erbracht werden. Jedoch hat dieser Studiengangwechsel zur Folge, dass die Studiendauer sich verlängert, da sie noch Veranstaltungen aufholen muss, die in der Informatik nicht verpflichtend waren.

2.3.5 Überschreiten der regulären Studiendauer:

Alex studiert im 6. Semester BA Informatik. Um die Regelstudienzeit einzuhalten, die sechs Semester beträgt, muss er bis zum Ende des Semesters alle Leistungen erbracht haben. Jedoch hat er eine Prüfung nicht bestanden und es fehlen ihm noch die Leistungen für diese Veranstaltung. Da er im nächsten Semester den Master Informatik auch an der Uni Ulm beginnen möchte, beschließt er daher, seine Bachelor Studiendauer um ein Semester zu verlängern. Dazu hört er nun Veranstaltungen aus dem Master, die er sich als Zusatzfachveranstaltungen anrechnet. Diese übernimmt er dann im Masterstudium.

3

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen für die zu entwickelnde App bestimmt und beschrieben. Die in diesem Kapitel gewonnenen Erkenntnisse sind grundlegend für die spätere Entwurfs- und Entwicklungsphase. Zunächst werden in Kapitel 3.1 verschiedene bereits vorhandene Angebote für Universitäten analysiert. Auf der Analyse aufbauend werden in Kapitel 3.2 Kurzinterviews vorgestellt, die durchgeführt wurden, um die Anforderungen klarer eingrenzen. Durch die Erkenntnisse aus Kapitel 3.1 und 3.2 werden in Kapitel 3.3 die funktionellen Anforderungen von ULMUS beschrieben. Abschließend werden in Kapitel 3.4 die nicht-funktionalen Anforderungen erläutert.

3.1 Analyse des Ist-Stands

Zurzeit existiert kein Studienführer für die Universität Ulm.
Damit ULMUS einen Mehrwert für ihre Nutzer bietet, müssen bereits vorhandene Angebote analysiert werden [OFF11]. Die Analyse von Angeboten anderer Universitäten ist daher hilfreich, um realistische Ideen für noch nicht bestehende Angebote für die Uni Ulm zu finden [Lau02]. Durch den Einblick in diese Apps gewinnt man weitere Erkenntnisse für ein mögliches Design und eine optimale Bedienung.
Dazu wird im Folgenden zuerst eine Übersicht von mobilen Applikationen anderer Universitäten vorgestellt und daraufhin bereits bestehende Angebote für die Universität Ulm. Zu jedem Angebot wird kurz erklärt, für welche mobilen Betriebssysteme sie erhältlich sind. Daraufhin werden ihre Funktionalitäten, das Design und die Usability beschrieben und ein Fazit zur jeweiligen App abgegeben. Im Fazit werden unter anderem die Anzahl der Funktionen, das Design und die Usability im Vergleich zu einer nicht optimierten App für mobile Geräte, bewertet. Ebenfalls werden Screenshots der App dargestellt, um die verschiedenen Arten für die Umsetzung aufzuzeigen. Abschließend wird ein Fazit über den Ist-Stand abgegeben und diskutiert, welche Erkenntnisse aus der Analyse für die späteren Anforderungen gewonnen wurden.

3.1.1 Vergleich des Ist-Stands

3.1.1.1 Uni Tübingen

Die App *Uni Tübingen* für die gleichnamige Universität wird zurzeit für Android und iOS angeboten [Tub13].

Funktionalitäten:
Die Startseite der App stellt aktuelle Neuigkeiten dar (siehe Abbildung 3.1 links). Ein Mensaplan zeigt das aktuelle Menü an und es können mit einer Karte die Gebäude der Universität angezeigt werden und nach bestimmten Gebäuden gesucht werden (siehe Abbildung 3.1 rechts). Außerdem können mit zusätzlichen Karten Bushaltestellen sowie

Wlan-Spots angezeigt werden. Ein Kalender gibt Auskunft über anstehende Veranstaltungen und mit einem integriertem Webbrowser kann die Homepage der Uni Tübingen besucht werden.

Usability und Design:

Das Design ist sehr schlicht aufgebaut und alle Funktionalitäten befinden sich ganz oben auf der Seite. Die App wurde für mobile Geräte optimiert und daher ist ihre Bedienbarkeit in den meisten Fällen sehr angenehm. Wie in Abbildung 3.1 zu sehen ist, wird für die Navigation der Hauptfunktionen das *Tab-Menü Pattern* verwendet [Nei12]. Dadurch ist es möglich, zu jedem Zeitpunkt zu einer anderen Hauptfunktion zu wechseln. Diese Art der Navigation ist vor allem für Apps mit wenigen Hauptfunktionen sinnvoll.

Abbildung 3.1: Screenshots der App Uni Tübingen

Fazit:
Die Funktionen der App sind nicht sehr umfangreich und mit der App können nur Informationen abgerufen werden. Einstellungsmöglichkeiten sind nicht vorhanden und alle Funktionen sind vom Internet abhängig. Das Design und die Bedienbarkeit sind in Ordnung. Die interne Webseitendarstellung der Uni Tübingen ist praktisch.

3.1.1.2 HSMWmobil

Die App *HSMWmobil* ist für die Hochschule Mittweida entwickelt worden und für Android und iOS verfügbar [HSM13].

Funktionalitäten:
Die Startseite bietet viele Funktionen an (siehe Abbildung 3.2 links). Ein Campusplan zeigt mithilfe von Google Maps eine Übersicht der Stadt Mittweida an. In dem Plan können nach Gebäuden, Parkplätzen, Servicepunkten und weiteren interessanten Punkten gesucht werden. Es können Kontaktdaten zum Personal der Uni aufgerufen und detaillierte Informationen wie zum Beispiel Orte, Telefonnummern und mehr eingesehen werden.
Außerdem können Nachrichten angezeigt werden und auch der Speiseplan abgerufen werden. Auf der nächsten Seite der App kann ein eigener erstellter Stundenplan (siehe Abbildung 3.2 mitte) und die eigenen Noten (siehe Abbildung 3.2 rechts) nach Anmeldung eingesehen werden. In den Einstellungen der App kann bestimmt werden, wie lange heruntergeladene Daten wie zum Beispiel der Mensaplan zwischengespeichert werden soll. Die App stellt noch weitere Funktionen wie ein integriertes Radio zur Verfügung bereit.

Usability und Design:
Die App besitzt über alle Seiten hinweg ein einheitliches Design. Sie ist sehr übersichtlich gestaltet und ihre Struktur ist logisch aufgebaut. Für die Hauptnavigation wird das *Seitenkarussell Pattern* (engl. page Carousel pattern) verwendet, womit durch eine Wischgeste nach links oder rechts über den Bildschirm zwischen den Hauptfunktionen

gewechselt werden kann [Nei12]. Dadurch ist die App sehr angenehm zu bedienen.

Fazit:
Die App HSMWmobil bietet viele Funktionen an. Die Funktionen zur Einsicht in den eigenen Stundenplan und die Notenanzeige sind dabei sehr interessant. Die Usability und das Design sind sehr gut für mobile Geräte optimiert. Ebenfalls positiv aufgefallen ist, dass nach Kontaktdaten gefiltert werden kann und dadurch eine bessere Bedienbarkeit erreicht wird. Negativ aufgefallen ist jedoch, dass die App ohne ständige Internetverbindung nicht funktioniert und sich beendet.

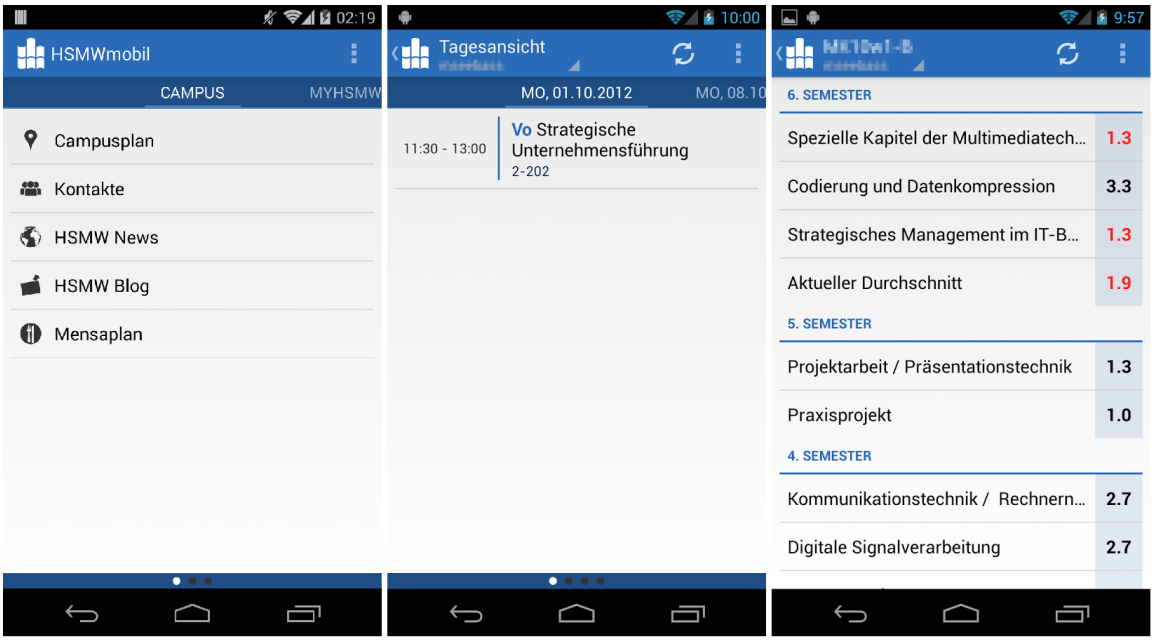


Abbildung 3.2: Screenshots der App HSMWmobil

3.1.1.3 myUDE

Die App *myUDE* für die Universität Duisburg-Essen ist für Android, iOS sowie für Windows Phone erhältlich [myU13].

Funktionalitäten:
Wie in Abbildung 3.3 links dargestellt, bietet die Startseite der App sehr viele Funktionen an. Es können Neuigkeiten unter anderem aus der Presse, Einrichtungen sowie der Fakultäten eingesehen werden. Unter Studium kann unter anderem ein eigener Stundenplan nach Anmeldung angezeigt werden (siehe Abbildung 3.3 rechts). Außerdem können weitere Informationen zu Fristen, Terminen und Fachschaften angezeigt werden. Unter Orientierung können Haltestellen, Mensen, Wlan-Hotspots, PC-Pools, Gebäude, Campus-Pläne und mehr eingesehen werden. Es werden noch viele weitere Hauptfunktionen wie zum Beispiel die Anzeige der Speisepläne, Freizeitinformationen und aktuelle Ereignisse im Kalender angeboten.

Usability und Design:
Das Design der App ist sehr angenehm und durch die verschiedenen Seiten hinweg konsistent. Die App ist sehr gut für mobile Geräte optimiert und die Funktionen sind logisch aufgebaut, sodass die Bedienung der App als intuitiv empfunden wird. Für die Navigierung zwischen den Hauptfunktionen wird das *Spring Board Pattern* verwendet [Nei12]. Das bedeutet, alle Hauptfunktionen sind über die Hauptseite (Startseite) erreichbar, was auf dem ersten Blick für eine besonders schnelle Navigation sorgt. Navigiert man jedoch in die Unterbereiche der Hauptfunktionen, ist ein Wechsel zu einer anderen Hauptfunktion nur möglich, wenn wieder zur Startseite gewechselt wird. Insgesamt ist dieses Pattern für die App sehr empfehlenswert, da durch die hohe Anzahl an Funktionen die App übersichtlich bleibt.

Fazit:
Die App myUDE bietet eine sehr hohe Anzahl an verschiedenen Funktionen an. Dennoch wirkt die App durch ihr ausgewähltes Design nicht überladen und ist sehr angenehm zu bedienen. Besonders hervorgehoben hat sich die App durch die Anzeige eines individuellen Stundenplans. Viele der Funktionen sind jedoch von einer ständigen Internetverbindung abhängig.

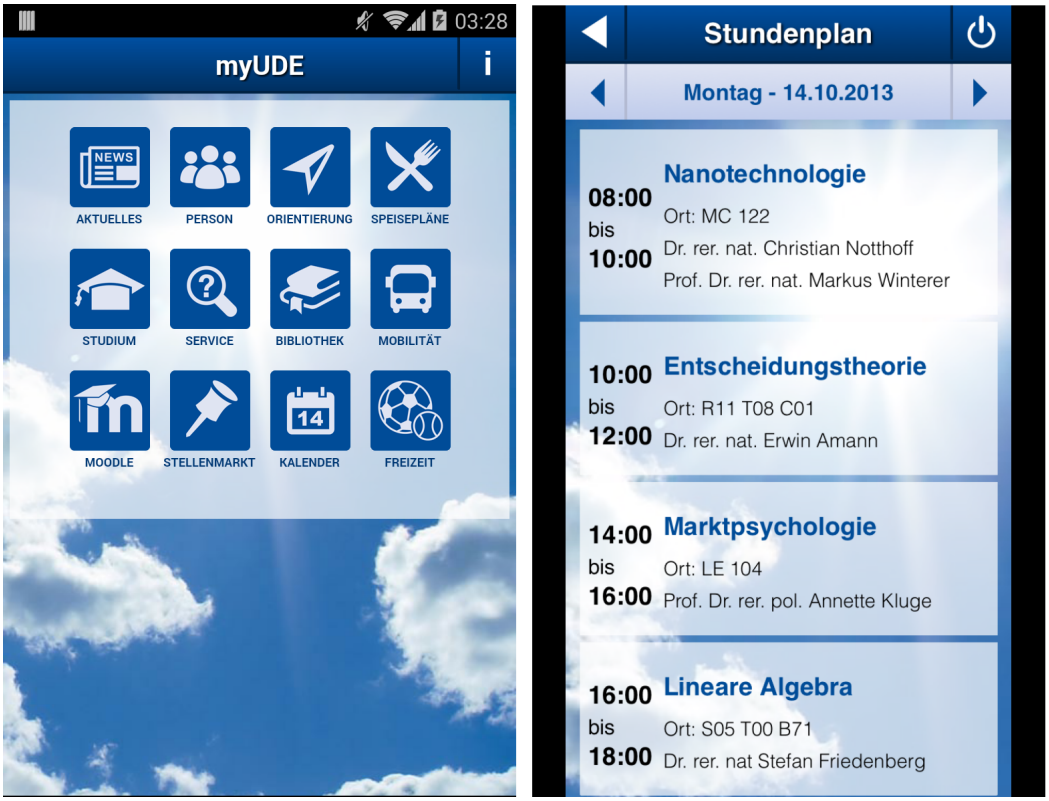


Abbildung 3.3: Screenshots der App myUDE

3.1.1.4 RUB mobile

Die App *RUB mobile* der Ruhr-Universität Bochum ist zurzeit für Android, iOS und Blackberry erhältlich [Rub13].

Funktionalitäten:

In Abbildung 3.4 links ist die Startseite der App mit den Hauptfunktionen abgebildet. Es können Nachrichten unter anderem zum Studium, zur Forschung und zum Hochschulsport eingesehen werden (siehe Abbildung 3.4 rechts). Es kann nach verschiedenen Orten gesucht werden, der aktuelle Mensplan eingesehen werden und vieles mehr. Weitere Funktionen sind auf der nächsten Seite der App vorhanden. Dort kann man

unter anderem nach Personen suchen oder sich über das Wetter informieren.

Usability und Design:
Das Design der App ist innerhalb der integrierten Funktionen konsistent. Jedoch führen viele Funktionen auf externe Seiten, wodurch ein einheitliches Design nicht vorhanden ist. Die Benutzerfreundlichkeit ist gut, da die App für den mobilen Gebrauch optimiert wurde. Positiv fällt hierbei auf, dass die externen Seiten ebenfalls für den mobilen Gebrauch optimiert sind. Jedoch stört der Wechsel zwischen der App und dem Browser beim Aufruf einer externen Funktion. Für die Navigation wird das Spring Board Pattern verwendet, womit alle Hauptfunktionen über die Hauptseiten erreichbar sind [Nei12]. Die Hauptseiten bilden dabei die Startseite und eine weitere Seite, zu der mit einer Wischgeste nach links aus der Startseite heraus navigiert werden kann. Diese Verteilung der Funktionen auf wenige Seiten ist bei einer hohen Anzahl an Hauptfunktionen sinnvoll.

Fazit:
Es werden sehr viele Funktionen in dieser App angeboten. Viele dieser Funktionen dienen jedoch als Platzhalter und verweisen auf externe Seiten. Abgesehen von dem Campusplan und den Notfallinformationen, die Auskunft über den Wachdienst und der zentralen Meldestelle bietet, sind leider alle Funktionen vom Internet abhängig. Durch die Verteilung der Funktionen auf zwei Seiten wirkt die App nicht überladen.

Abbildung 3.4: Screenshots der App RUB mobile

3.1.1.5 UniBib

UniBib ist eine App für verschiedene Universitäten wie zum Beispiel für die Universität Ulm und zurzeit für Android verfügbar [Uni13].

Funktionalitäten:
Die UniBib App besitzt nur eine Hauptfunktion, mit der der Bibliothekskatalog verschiedener Universitäten durchsucht werden kann (siehe Abbildung 3.5). Gefundene Treffer können in einer Merkliste gespeichert werden.

Usability und Design:
Die Oberfläche der App ist sehr schlicht aufgebaut und optimiert für die Verwendung der

Hauptfunktion.

Fazit:
Das Angebot an Funktionen der App ist sehr niedrig, da der Verwendungszweck nur auf das Durchsuchen des Kataloges der jeweiligen Universität beschränkt ist. Beim Testen der App ist jedoch aufgefallen, dass teilweise keine Ergebnisse bei der Suche von Büchern zurückgeliefert wurden, wodurch die App zurzeit nur eingeschränkt nutzbar ist.

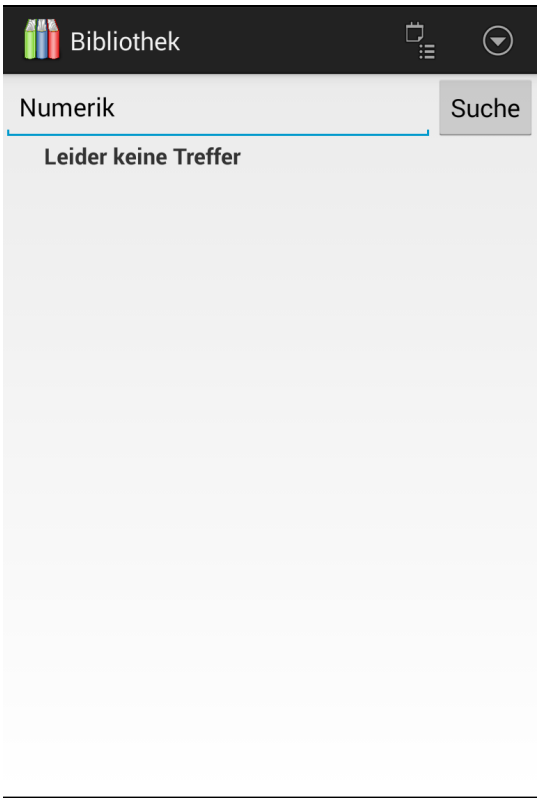


Abbildung 3.5: Screenshot der App UniBib

3.1.1.6 Mensaplan

Die App *Mensaplan* für die Uni Ulm ist zurzeit nur für Android erhältlich [Men13a].

Funktionalitäten:

Mit der App kann der aktuelle Speiseplan der Mensa, des Bistros und der Uni West für die aktuelle Woche angezeigt werden (siehe Abbildung 3.6). Zudem kann man Informationen zu den Öffnungszeiten einsehen.

Usability und Design:

Das Design ist sehr schlicht und über die einzelnen Seiten konsistent. Zwischen den Seiten wird wie in der App HSMWmobil mit dem Seitenkarussell Pattern navigiert, wodurch die App sehr bequem zu bedienen ist.

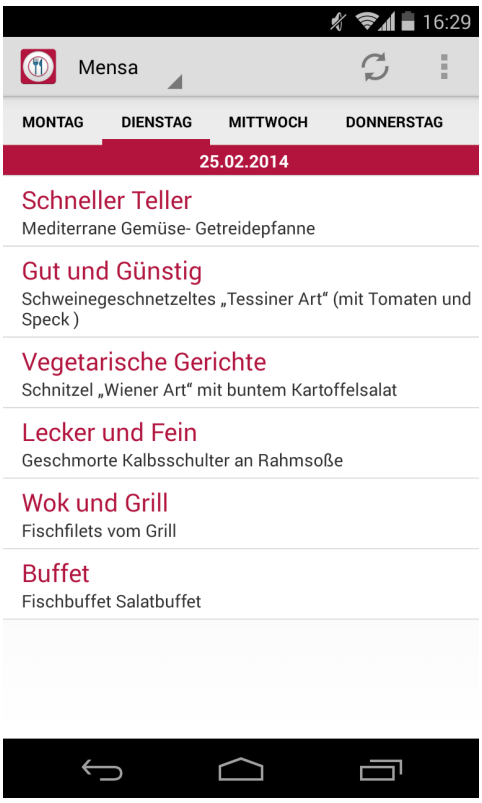


Abbildung 3.6: Screenshot der App Mensaplan

Fazit:

Die App Mensaplan ist wie die App UniBib für einen Verwendungszweck konzipiert

worden und bietet dementsprechend nur wenige Funktionen an. Im Vergleich zu den anderen Apps, die einen Speiseplan anbieten, werden leider keine Preise zu den Menüs angezeigt.

3.1.1.7 Study-App

Die App *Study-App* für die Studenten der Universität Ulm kann unter allen Betriebssystemen genutzt werden, da sie nicht auf dem Gerät installiert werden muss und stattdessen vom Browser aus abrufbar ist [Stu13].

Funktionalitäten:
Die Oberfläche der Hauptseite (siehe Abbildung 3.7 links) kann in drei Bereiche unterteilt werden. Ganz oben auf der Seite stellen die Fakultäten verschiedene Informationen für Studenten bereit. In der Mitte der Hauptseite werden verschiedene Informationen zur Universität angeboten. Hier kann nach Personen gesucht werden und Nachrichten angezeigt werden. Außerdem werden verschiedene Links zu Campusdiensten wie zum Beispiel dem Online Bibliothekskatalog, dem Hörsaalfinder, der Verwaltung, verschiedenen Lernplattformen und zu den Lageplänen der Universität Ulm bereitgestellt. Unter Beratung werden Informationen zu den Beratungsstellen bereitgestellt, unter anderem für die zentrale Studienberatung. Unter Einrichtungen sind Informationen zum Kommunikations- und Informationszentrum (KIZ) zu finden. Externe Angebote werden unten in der Hauptseite aufgelistet. Dort wird für die Fahrplanauskunft auf eine externe Seite weitergeleitet.

Usability und Design:
Das Design ist schlicht gehalten und wird über die verschiedenen Seiten hinweg beibehalten. Ausnahmen hierbei sind die externen Verlinkungen zu anderen Angeboten. Die Funktionen der Seite sind für mobile Geräte optimiert, sodass die Bedienung als sehr angenehm empfunden wird. Dies variiert jedoch bei den Verlinkungen zu externen Angeboten. Beispielsweise ist es nicht möglich, den Hörsaalfinder unter iOS oder Android zu verwenden. Das liegt daran, dass die der Hörsaalfinder Adobe Flash verwendet, welche von einigen Betriebssystemen (z.B. iOS) nicht unterstützt wird. Für die Navigierung

zwischen den Funktionen werden zwei Patterns verwendet. Um die Informationen zu einzelnen Fakultäten einzusehen, wird das Spring Board Pattern verwendet (siehe App myUDE). Für die anderen Funktionen wird jedoch das *List-Menü Pattern* verwendet [Nei12]. Im Vergleich zum Spring Board Pattern werden hier die Funktionen in einer Liste präsentiert. Ein Vorteil des List-Menü Pattern ist die Anzeige von längeren Titeln und kurzen Inhaltstexten (siehe Abbildung 3.7 rechts).
Mit dem Einsatz verschiedener Patterns können damit Funktionen gruppiert werden, wodurch ebenfalls eine bessere Übersicht erreicht wird.

Fazit:
In der App Study-App werden zahlreiche Funktionen für das Studium angeboten. Im Vergleich zu den anderen Apps, ist sie eine reine Web-App, wodurch sie von jedem Browser aus erreichbar ist und eine Installation auf dem Smartphone oder Tablet entfällt. Dadurch wird garantiert, dass das Design für verschiedene Betriebssysteme immer gleich bleibt. Außerdem muss die App nicht mehr für jedes System neu entwickelt werden, wodurch viel Zeit für die Wartung und Entwicklung entfällt. Da die App jedoch in einem Webbrowser läuft, kann sie nicht ohne eine Internetverbindung erreicht und verwendet werden. Verlinkungen zu Angeboten wie dem Hörsaalfinder sind redundant, da die verwendete Software nicht unterstützt wird und der Benutzer daher auf andere Systeme ausweichen muss.

Abbildung 3.7: Screenshots der Web-App Study-App

3.1.2 Fazit über den Ist-Stand

Anhand der verschiedenen Apps aus Kapitel 3.1.1 wird deutlich, dass für die meisten Universitäten das Angebot sehr ähnlich ist. Alle Angebote sind dabei unter Android und sehr oft für iOS erhältlich. Insgesamt wird für jede Universität eine Nachrichtenanzeige, ein Speiseplan und ein Lageplan angeboten. Einen Bibliothekskatalog, eine Personensuche, eine Lernplattformfunktion und ein Fahrplan existiert ebenfalls für die meisten Universitäten, insbesondere auch für die Universität Ulm. Die Apps myUDE und HSMWmobil bieten zusätzlich einen Zugang zu einem individuellen Stundenplan an und letztere App sogar Informationen über den Notenspiegel. Zusammenfassend kann man sagen, dass alle Apps größtenteils nur zur Darstellung von Informationen konzipiert worden sind und wenig Spielraum für individuelle Einstellungen oder einem individuellem Studienverlauf anbieten. Durch eine fehlende Internetverbindung sind ihre

Funktionen sehr stark eingeschränkt.
Aus den Ergebnissen des Ist-Stands können wichtige Erkenntnisse für die in Kapitel 3.3 zu erstellenden Anforderungen mitgenommen werden.
Unter anderem soll berücksichtigt werden, dass viele Funktionen auch ohne eine Internetverbindung funktionieren. Eine Anzeige der aktuellen Veranstaltungen kann zum Beispiel in Form eines Stundenplans erstellt werden. Für die Navigation kann bei einer hohen Anzahl an Funktionen das Spring Board Pattern oder bei einer niedrigen Anzahl das Tab-Menü oder das Seitenkarussell Pattern verwendet werden.
Ebenfalls sollte für eine verbesserte Übersicht und einfache Bedienbarkeit eine Filtermöglichkeit in Betracht gezogen werden, falls viele Informationen in einer Liste dargestellt werden. Falls nötig, soll der Benutzer die App nach seinen Präferenzen einstellen können, um eine optimale Bedienung zu ermöglichen. Das Design soll durch alle Funktionen konsistent bleiben und für den mobilen Gebrauch optimiert sein, damit die App bequem zu bedienen ist. Die Verfügbarkeit für Android und iOS sollte vorhanden sein.

3.2 Kurzinterviews

Um die Spezifikationen der Anforderungen an das System weiter zu definieren, wurden mit mehreren Studenten der Uni Ulm ein kurzes Interview geführt.
Interviews eignen sich besonders gut für die Anforderungsanalyse, da das Gespräch während des Verlaufs individuell angepasst werden kann. Dies ist z. B. nötig, um aufkommende Fragen zu klären oder das Gespräch in Richtungen zu lenken, die einen besonders hohen Informationsgewinn versprechen [Rup07]. Dadurch ist es möglich zu ermitteln, welche Angebote zurzeit verwendet werden, wo ihre Probleme liegen, welche Anforderungen realistisch umzusetzen sind und wo Probleme in der Umsetzung entstehen könnten [Lau02]. Damit Interviews nützlich sind, müssen diese vor allem mit der potenziellen Nutzergruppe durchgeführt werden [Lau02]. In diesem Fall wird die Nutzergruppe durch Studenten der Universität Ulm repräsentiert. Zusätzlich ist wichtig, welche Fragen während des Interviews gestellt werden, um auch ausreichende Informa-

tionen über mögliche Anforderungen zu erhalten [Lau02].

Das Ziel der Interviews war es, herauszufinden, wie eine neue App die Studenten im Laufe ihres Studiums, zusätzlich zu den bestehenden Angeboten, besser unterstützen kann. Während des Interviews wurden dazu Fragen zu folgenden Bereichen gestellt:

1. Welche mobilen Angebote werden bisher genutzt?
2. Wie zufrieden sind Sie mit den bisherigen Angeboten?
3. Wie könnten die verwendeten Angebote erweitert werden?
4. Welche zukünftigen Angebote könnten Sie sich vorstellen?

Vor allem für den 4. Bereich wurden die Erkenntnisse des Ist-Stands in das Interview eingebaut und den Studenten weitere Fragen in diese Richtung gestellt. Dazu wurden den Studenten unter anderem gefragt, inwieweit eine Unterstützung des Studiums in Form eines Stundenplans sinnvoll ist und ob die App einen hohen Funktionsumfang bieten sollte. Dafür wurden verschiedene Funktionen der verglichenen Apps genannt, wie zum Beispiel die Anzeige eines Speiseplans, Wetterinformationen, Sportinformationen und mehr.

Das Interview wurde insgesamt mit acht Studenten durchgeführt. Dabei hat sich ergeben, dass als einzige App der Mensaplan aus Kapitel 3.1 verwendet wurde. Angebote wie zum Beispiel eine integrierte Radiofunktion, eine Wetteranzeige, Informationen zu Sportaktivitäten oder einem Campusplan wurden von den Studenten als nicht besonders wichtig empfunden.

Jedoch wurde des Öfteren der Wunsch geäußert, eine mobile Übersicht für den Studienverlauf anzubieten. Dazu hat sich während des Gesprächs ergeben, dass Veranstaltungen individuell erstellt werden sollen und den passenden Modulen zugeordnet werden können. Dadurch kann überprüft werden, welche Veranstaltungen noch zu belegen sind. Es sollen dazu Zeiten eingetragen werden können und die Veranstaltungen aus dem aktuellem Semester dargestellt werden, um eine kurze Übersicht zu erhalten.

Zusätzlich wurde festgestellt, dass sich viele Studenten eine zentrale Informationsquelle rund um das Thema Studium wünschen. Aus Sicht der Studenten können dadurch oft gestellte Fragen zum Studium schneller beantwortet werden, was eine bessere Planung des Studiums ermöglicht.
Die Interviews mit den Studenten haben gezeigt, dass die App sich auf einige wenige zentrale Funktionen beschränken sollte.
Aus den Erkenntnissen, die durch die Kurzinterviews und der Analyse bereits bestehender Angebote gewonnen wurden, werden in Kapitel 3.3 die Anforderungen beschrieben.

3.3 Anforderungsbeschreibung

In diesem Kapitel werden zunächst die funktionalen Anforderungen grob beschrieben, die aus den Erkenntnissen der Kapitel 3.1 und 3.2 gewonnen wurden. Dabei werden die Erkenntnisse weiter ausgeführt, um eine optimale Verwendbarkeit der App zu gewährleisten.

Der Benutzer soll seinen Studienverlauf abbilden können. Dazu stehen ihm mehrere Studiengänge zur Auswahl. Er kann Veranstaltungen erstellen und diese in einer Semesterübersicht anzeigen lassen.
Für eine bequemere Bedienung soll er auch Veranstaltungen anderer Benutzer übernehmen können.
Weiterhin kann der Benutzer detaillierte Informationen zu den Semestern und seinem Studienverlauf einsehen. Der Studienverlauf soll als PDF exportierbar sein.
Außerdem kann er Antworten zu Fragen rund um das Thema Studium einsehen.
Falls nötig, sollen Filter für eine bessere Übersicht eingesetzt werden. Zusätzlich sollen dynamische Suchen gestartet werden können, um neue Informationen herunterzuladen.
Der Benutzer soll dabei die Möglichkeit erhalten, die Einstellungen der App nach seinen Wünschen anzupassen.
Im Hinblick auf das Studium ist besonders wichtig, dass ULMUS die verschiedenen Studiengänge korrekt abbildet. Dazu zählen vor allem die korrekte Bezeichnungen der

Module und auch eine Unterstützung verschiedener Modularten, wie es beispielhaft in Kapitel 2.1 erläutert wurde. Die Übernahme relevanter Veranstaltungen bei einem Studiengangwechsel soll ebenfalls berücksichtigt werden. Für die Verwendung von Funktionen auf der Seite des Servers muss sichergestellt werden, dass wichtige Daten wie zum Beispiel die Daten für die Prüfungsordnung auf der Client- und auf der Server-Seite synchron sind. In Kapitel 4.2 werden diese Anforderungen berücksichtigt.

In den folgenden Kapiteln werden die funktionalen Anforderungen einerseits mithilfe von Anwendererzählungen textuell und andererseits mithilfe von Flussdiagrammen visuell erläutert. Die Notation der Flussdiagramme ist dabei in Abbildung 3.9 dargestellt.
Die Anwendererzählungen gehen dabei näher darauf ein, welche Schritte im Einzelnen vom Benutzer vorzunehmen sind und welche Besonderheiten und Ausnahmen auftreten können. Diese Erkenntnisse sind eine hilfreiche und notwendige Basis für die spätere Entwurfsphase in Kapitel 4.

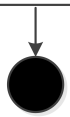
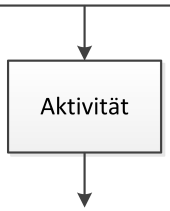
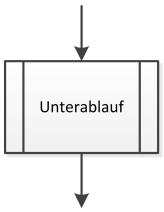
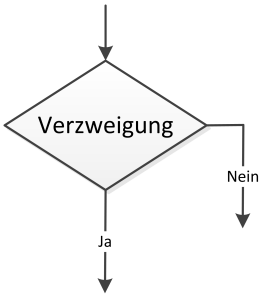
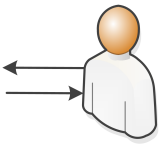
Symbol	Name	Bedeutng
	Startknoten	Startet einen Aktivitätsablauf.
	Endknoten	Endpunkt eines Aktivitätsablaufs.
	Aktivität	Stellt einen Ablaufschritt dar.
	Unterablauf	Stellt einen eigenen Aktivitätsablauf dar.
	Verzweigung	Entscheidet über den weiteren Ablauf der Aktivität. Kann auch eine Verzweigung wieder zusammenführen (XOR-Join).
	Benutzerinterkation	Beschreibt die Interaktionen mit einem Benutzer.

Abbildung 3.8: Flussdiagramm Annotation

3.3.1 Initialisierung

Die Initialisierung wird nur beim erstmaligen Start der App aufgerufen. Der Vorgang für die Initialisierung ist in Abbildung 3.9 dargestellt.

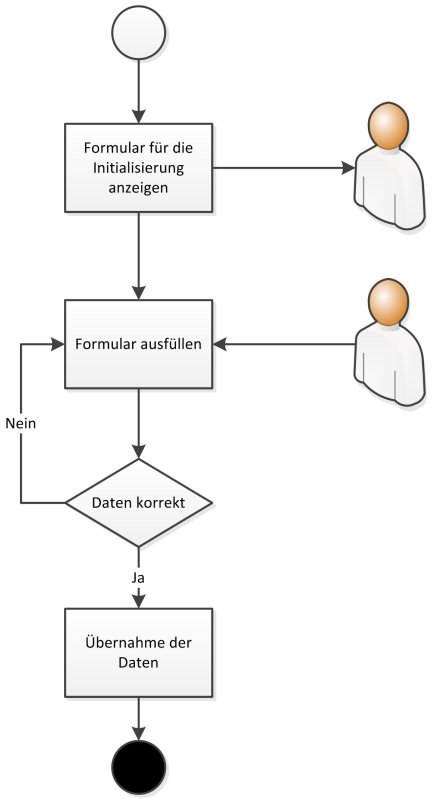


Abbildung 3.9: Ablauf zur Initialisierung

Welche Aktionen werden hier durchgeführt:
Der Benutzer muss beim erstmaligen Start der App seinen Studiengang und die für ihn geltende Prüfungsordnung auswählen (Profil). Zudem kann der Benutzer sein aktuelles Semester eintragen, wodurch alle bisherigen Semester nach der Erstellung des Profils angelegt werden. Dabei können Semester auch als Urlaubssemester markiert und im System berücksichtigt werden. Allerdings ist es zum Beispiel nicht möglich, Veranstaltungen während eines Urlaubssemesters zu erstellen. Weiterhin muss der Benutzer je nach Profil eventuell weitere Informationen angeben, wie zum Beispiel die Wahl des Anwendungsfaches (siehe Kapitel 2.1 Module).

Auf welche Ausnahmen/Besonderheiten müssen hierbei geachtet werden:
Falls der Benutzer nicht alle Daten eingetragen hat, erhält er eine Meldung und muss diese nachtragen. Dies ist zum Beispiel der Fall, wenn kein Anwendungsfach ausgewählt wurde.

3.3.2 Semesterübersicht

Die *Semesterübersicht* (SÜ) ist die Standardansicht von ULMUS und wird bei jedem Start der App angezeigt. Hier werden alle relevanten Informationen zu den Semestern angezeigt. Die möglichen Funktionen für den Benutzer werden im Folgenden erklärt.

SÜ 1: Normale Semesterinformationen anzeigen
Der Vorgang für den Abruf der Semesterinformationen ist in Abbildung 3.10 dargestellt.

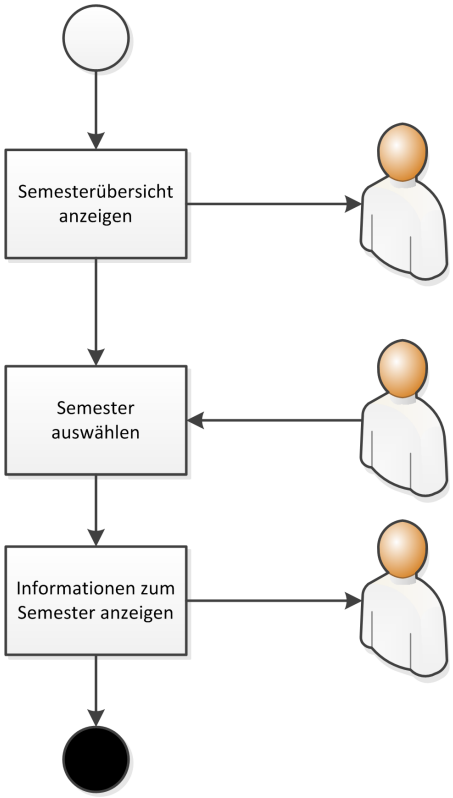


Abbildung 3.10: Ablauf zur Anzeige der Semesterinformationen

Welche Aktionen werden hier durchgeführt:

Der Benutzer kann bisherige Informationen zu den einzelnen Semestern in einer Semesterübersicht einsehen. Durch Auswählen eines entsprechenden Semesters werden ihm die dazugehörigen Veranstaltungen mit ihren jeweiligen Zeiten und Orten angezeigt.

SÜ 2: Detaillierte Semesterinformationen anzeigen

Um weitere Informationen zu einem Semester anzuzeigen, kann eine detaillierte Anzeige für ein Semester aufgerufen werden. Der dazugehörige Vorgang ist in Abbildung 3.11 dargestellt.

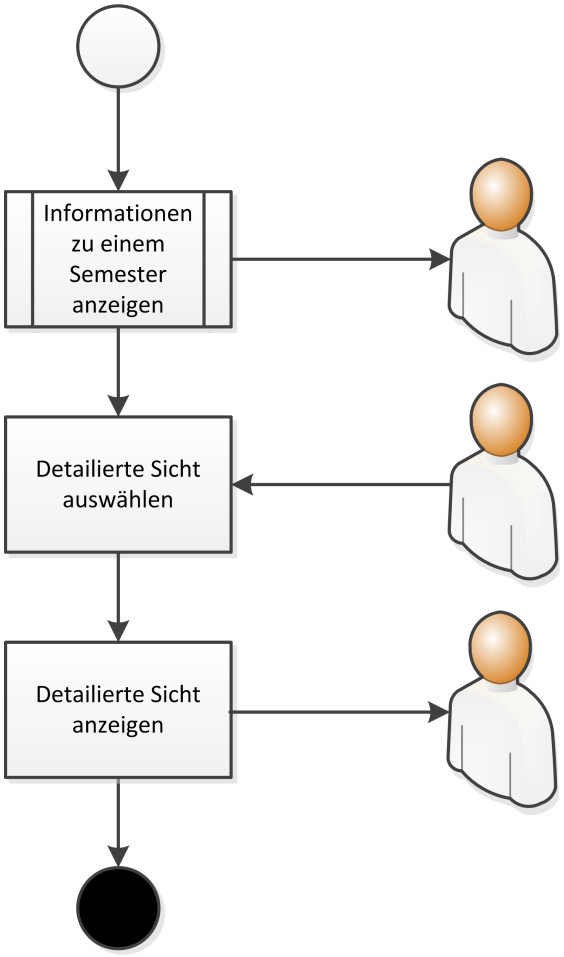


Abbildung 3.11: Ablauf zur Anzeige der detaillierten Semesterinformationen

Welche Aktionen werden hier durchgeführt:

Der Benutzer kann zu jedem Semester weitere Informationen aufrufen. Dazu wählt er das entsprechende Semester aus und wechselt von dort in eine detaillierte Semesteranzeige. Dabei werden ihm die besuchten Veranstaltungen mit ihren Leistungspunkten aufgelistet, die gesamten erbrachten Leistungspunkte in diesem Semester aufgezählt und auch angegeben, ob die benötigten Leistungspunkte im ausgewähltem Fachsemester bereits erreicht worden sind.

SÜ 3: Veranstaltungen eines Semesters ändern oder hinzufügen

Der Vorgang für die Erstellung neuer Veranstaltungen oder die Änderung einer Veranstaltung eines ausgewählten Semesters ist in Abbildung 3.12 abgebildet.

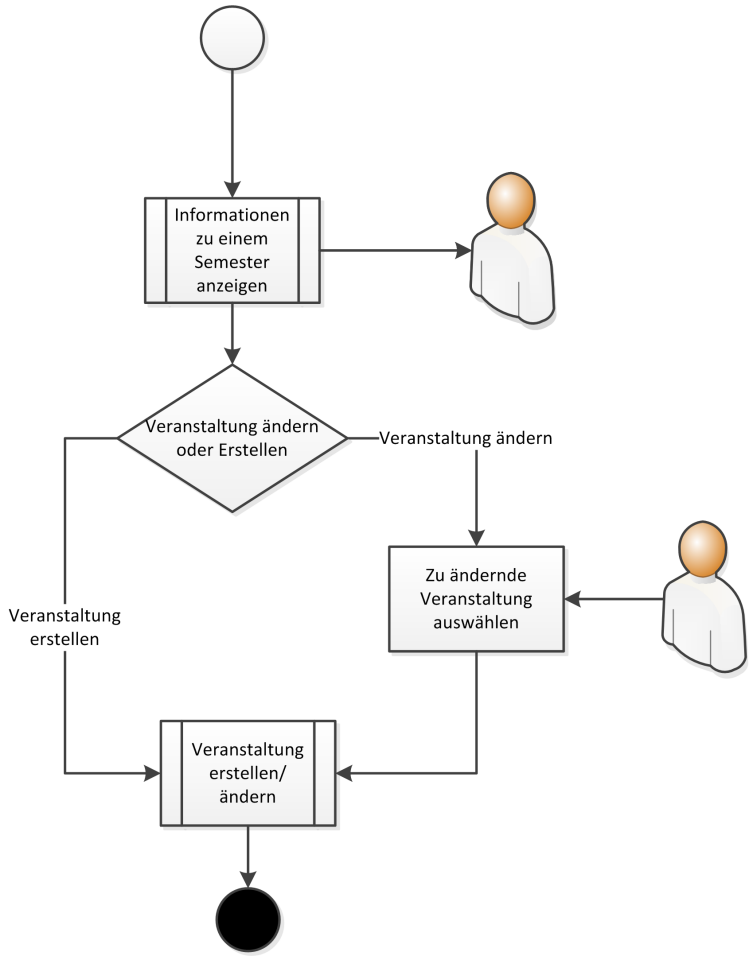


Abbildung 3.12: Ablauf zur Erstellung oder Änderung einer Veranstaltung in einem Semester

Welche Aktionen werden hier durchgeführt:

Um eine bestehende Veranstaltung zu ändern oder zu erstellen muss der Benutzer zuerst das passende Semester auswählen. Dabei werden ihm alle Veranstaltungen in diesem Semester aufgelistet. Durch das Auswählen einer Veranstaltung wird er zu einem neuem Fenster weitergeleitet, in der er die Informationen zur Veranstaltung ändern kann. Alternativ kann der Benutzer im ausgewähltem Semester auch eine neue Veranstaltung

erstellen. Der Vorgang für die Erstellung oder Änderung einer Veranstaltung wird in Kapitel 3.3.5 detaillierter erläutert.

3.3.3 Gesamtübersicht

Die Funktionen in der *Gesamtübersicht* (GÜ) sind in Abbildung 3.13 dargestellt.

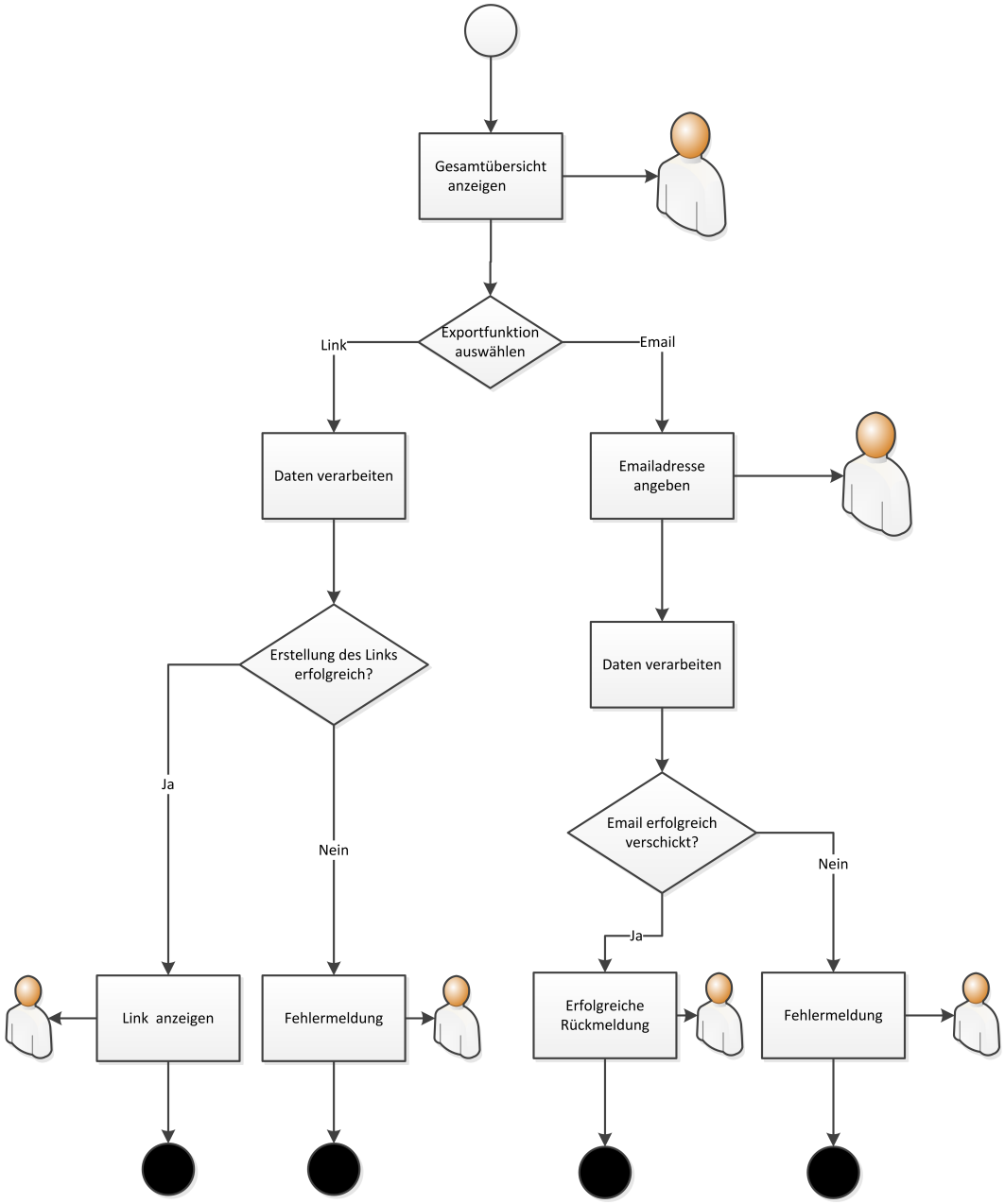


Abbildung 3.13: Ablauf zur Anzeige der Gesamtübersicht

Welche Aktionen werden hier durchgeführt:

Hier hat der Benutzer die Möglichkeit, seine bisher erreichten Leistungen einzusehen. Dazu werden ihm unter anderem die besuchten Veranstaltungen nach Modulen sortiert

aufgelistet und auch die bisher erreichten Leistungspunkte in den Modulen angezeigt. Aus dieser Gesamtübersicht kann sich der Benutzer auch eine PDF generieren lassen, die die wichtigsten Informationen zu seinem bisherigen Studienverlauf im ausgewählten Profil beinhalten. Diese PDF kann der Benutzer sich entweder als Email zuschicken lassen kann oder einen Link erhalten, mit dem er die PDF für eine gewisse Zeitdauer vom Server laden kann.

Auf welche Ausnahmen/Besonderheiten müssen hierbei geachtet werden:
Der Benutzer wird benachrichtigt, falls eine inkorrekte Email-Adresse angegeben wird oder der Export fehlschlägt.

3.3.4 Veranstaltungsübersicht

In der *Veranstaltungsübersicht* (VÜ) können eigene Veranstaltungen und vom Server heruntergeladene Veranstaltungen eingesehen werden. Die Funktionen werden im Folgenden erklärt.

VÜ 1: Veranstaltungen erstellen/ändern/übernehmen
Die Funktionsweise der Erstellung, Änderung und der Übernahme von Veranstaltungen ist in Abbildung 3.14 dargestellt.

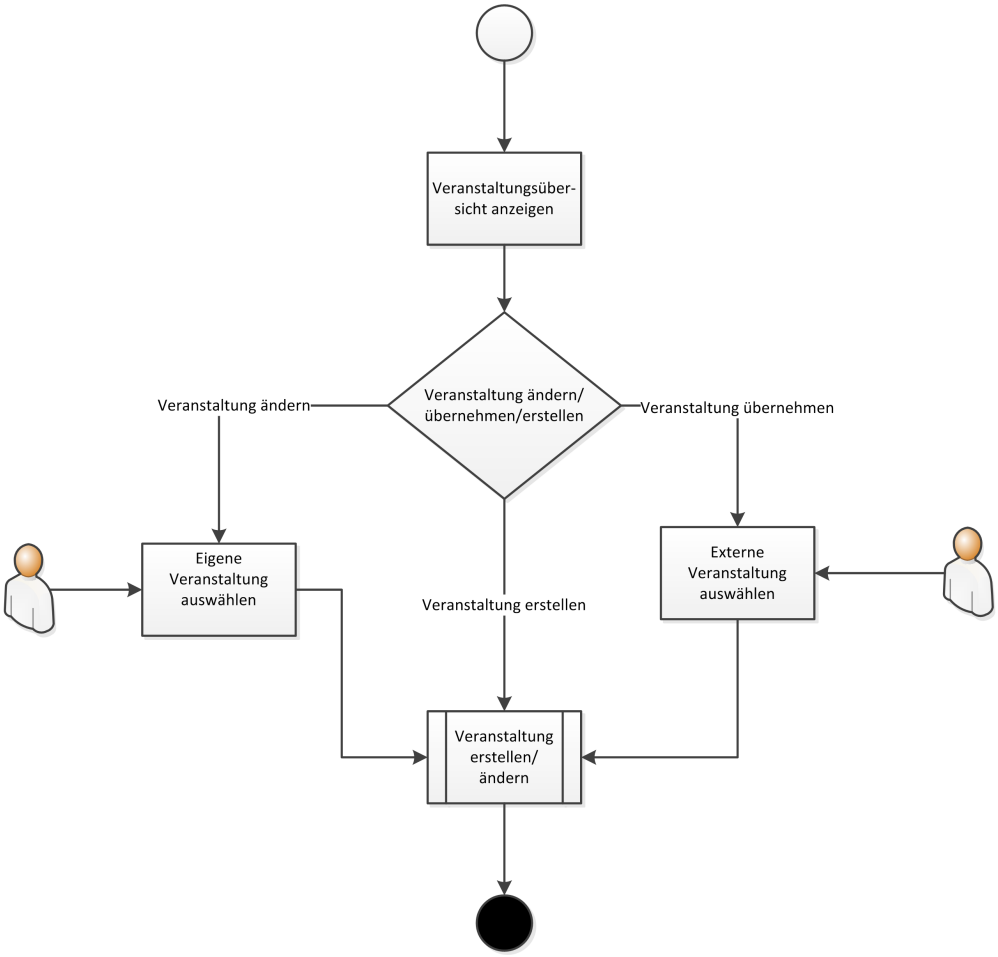


Abbildung 3.14: Ablauf zur Erstellung, Änderung und Übernahme von Veranstaltungen

Welche Aktionen werden hier durchgeführt:

Hier werden dem Benutzer verschiedene Veranstaltungen angezeigt. Dabei werden die Veranstaltungen in zwei verschiedene Komponenten unterteilt.

In der ersten Komponente kann der Benutzer seine eigenen erstellten Veranstaltungen einsehen und durch Berührung einer Veranstaltung diese verändern. In der zweiten Komponente werden ihm Veranstaltungen angezeigt, die bereits andere Studenten erstellt haben und vom Server geladen wurden. Hier hat der Benutzer die Möglichkeit, diese zu übernehmen. Der genaue Vorgang für die Erstellung beziehungsweise Änderung einer Veranstaltung ist in Kapitel 3.3.5 erläutert.

VÜ 2: Veranstaltungen filtern und dynamisch suchen

Der Vorgang für die Filterung von Veranstaltungen und der dynamischen Suche ist in Abbildung 3.14 dargestellt.

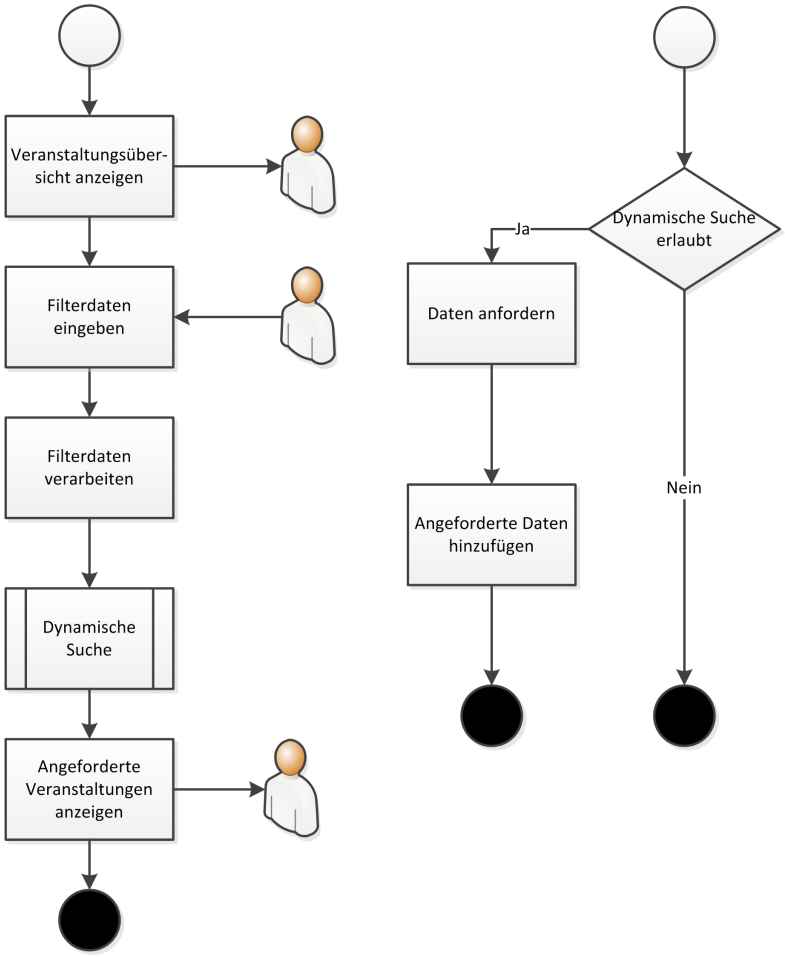


Abbildung 3.15: Ablauf für die Filterung von Veranstaltungen (links) und der dynamischen Suche (rechts)

Welche Aktionen werden hier durchgeführt:

Der Benutzer hat die Möglichkeit, externe Veranstaltungen zu filtern. Nach der Eingabe der gesuchten Veranstaltungen werden diese gefiltert und es wird eine dynamische Suche angestoßen. Falls die dynamische Suche aktiviert wurde, werden eingegebene Daten an einen Server geschickt und dem Benutzer werden neue Veranstaltungen angezeigt.

3.3.5 Veranstaltungsformular

In Abbildung 3.16 wird der Vorgang einer Erstellung und Änderung einer Veranstaltung im *Veranstaltungsformular* (VF) konkret beschrieben.

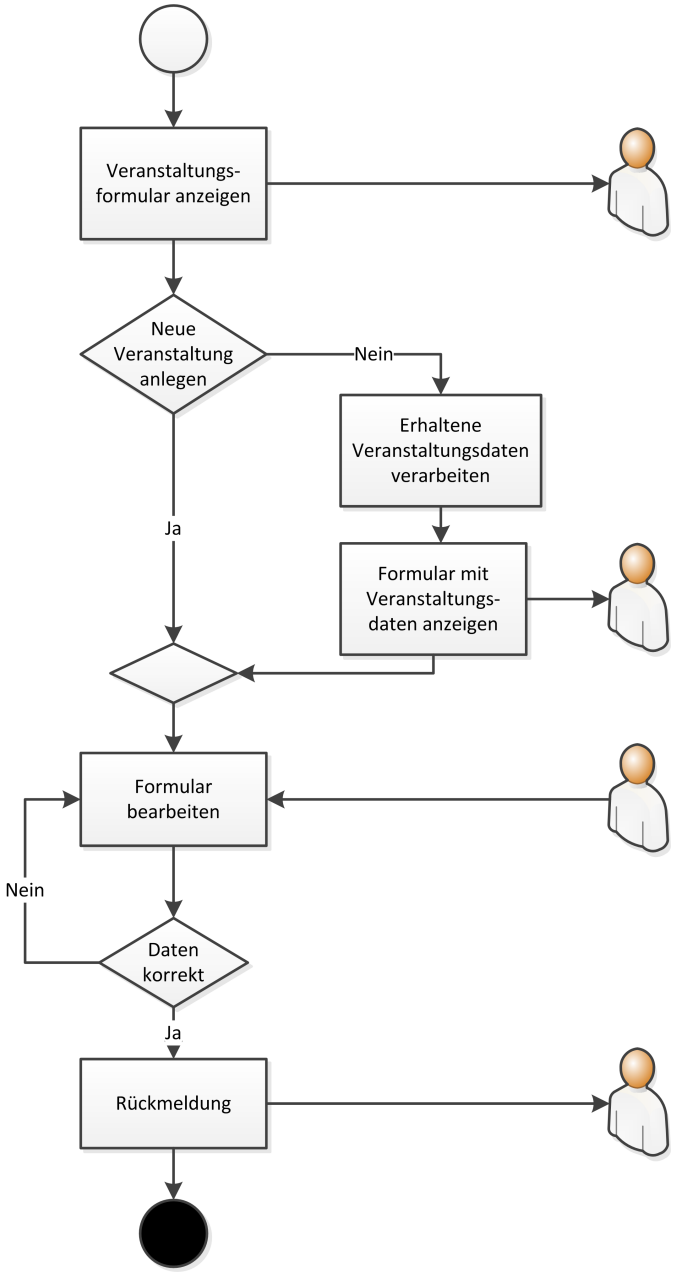


Abbildung 3.16: Ablauf im Veranstaltungsformular

Welche Aktionen werden hier durchgeführt:
In dieser Anzeige kann der Benutzer, je nachdem, mit welcher Interaktion er zu dieser Anzeige gelangt ist, Veranstaltungen anlegen oder ändern. Soll eine bestehende Veranstaltung verändert werden oder eine externe Veranstaltung vom Server übernommen werden, werden die Felder des Formulars entsprechend der erhaltenen Informationen ausgefüllt. Somit muss der Benutzer für die Veranstaltung nur noch minimale Änderungen vornehmen. Wird eine neue Veranstaltung angelegt, muss der Benutzer einen Veranstaltungsnamen eintragen, die Veranstaltung einem passendem Modul zuordnen und die Leistungspunkte eintragen. Eine optionale Angabe von Zeiten und Orten kann ebenfalls erfolgen.
Dabei werden dem Benutzer beim Eintragen der Veranstaltungsbeschreibung eine Liste aus möglichen Veranstaltungen angezeigt, aus der er die passende wählen kann. Zugleich wird automatisch das passende Modul ausgewählt. Die umgekehrte Reihenfolge ist auch möglich, falls das Beschreibungsfeld keinen Eintrag enthält. Zudem kann der Benutzer auch entscheiden, ob er die Veranstaltung nach dem Erstellen an einen Server schicken möchte, damit andere Studenten diese ebenfalls nutzen können.

Auf welche Ausnahmen/Besonderheiten müssen hierbei geachtet werden:
Falls das Feld für die Veranstaltungsbeschreibung nach dem Ausfüllen des Formulars leer ist, wird der Benutzer benachrichtigt. Ansonsten erhält der Benutzer eine erfolgreiche Rückmeldung über das Anlegen oder Ändern einer Veranstaltung. Außerdem wird der Benutzer benachrichtigt, ob die Übermittlung der Daten an den Server erfolgreich war.

3.3.6 Quickadd

In Abbildung 3.17 ist die Funktionsweise der *Quickadd*[1]-Funktion beschrieben.

[1] zu dt. „schnelles hinzufügen“

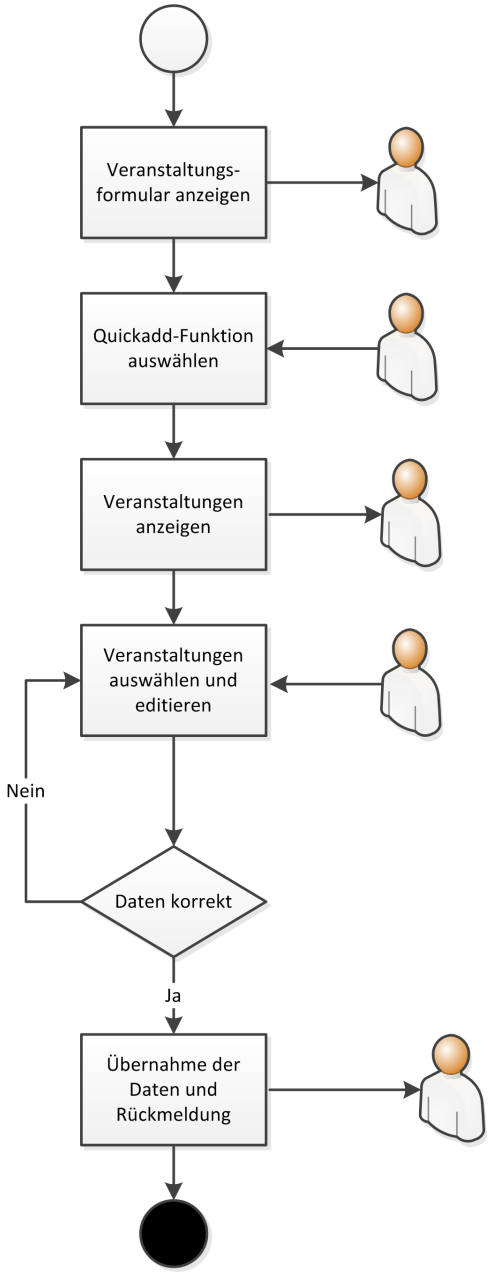


Abbildung 3.17: Ablauf der Quickadd-Funktion

Welche Aktionen werden hier durchgeführt:

Der Benutzer kann über das Veranstaltungsformular zur Quickadd-Funktion gelangen. Mit dieser hat er die Möglichkeit, mehrere Veranstaltungen auf einmal anzulegen. Dabei

wird ihm eine Liste von Veranstaltungen angezeigt, aus denen er bereits besuchte Veranstaltungen wählen kann. Zu den Veranstaltungen kann er zusätzlich noch das Semester, die Leistungspunkte und die Veranstaltungsbezeichnung eintragen.

Auf welche Ausnahmen/Besonderheiten müssen hierbei geachtet werden:
Zu den ausgewählten Veranstaltungen müssen die Bezeichnungen einen Eintrag enthalten. Falls eine Bezeichnung keinen Eintrag enthält, erhält der Benutzer eine Benachrichtigung, sodass er diese eintragen kann. Bei erfolgreicher Erstellung erhält der Benutzer eine Rückmeldung.

3.3.7 FAQ

Die FAQ[2]-Übersicht (FÜ) enthält oft gestellte Fragen rund um das Studium. Mögliche Aktionen werden im Folgenden betrachtet.

FÜ 1: Anzeigen einer Antwort zu einer Frage
Der Vorgang für die Anzeige einer Antwort ist in Abbildung 3.18 dargestellt.

[2] zu dt. „oft gestellte Fragen“

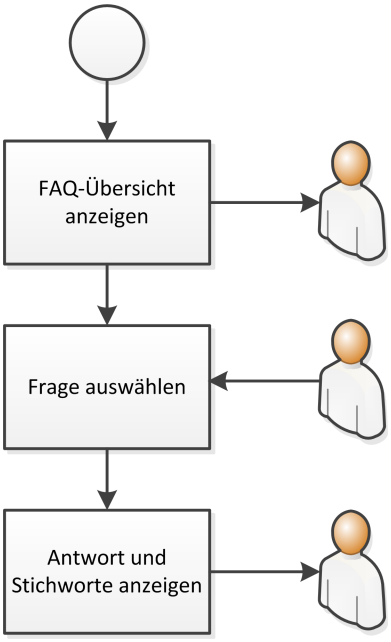


Abbildung 3.18: Ablauf zur Anzeige einer Antwort zu einer Frage

Welche Aktionen werden hier durchgeführt:

Hier kann sich der Benutzer über Fragen betreffend zum Studium informieren lassen. Dabei werden ihm die Fragen in einer Liste präsentiert. Durch berühren einer Frage werden ihm neben der Antwort auch mögliche Stichwörter angezeigt, die der Filterung dienen. Diese Filterfunktion wird nachfolgend näher erläutert.

FÜ 2: Filtern von Fragen mit einem Stichwort

Der Vorgang für die Filterung von Fragen ist in Abbildung 3.19 dargestellt.

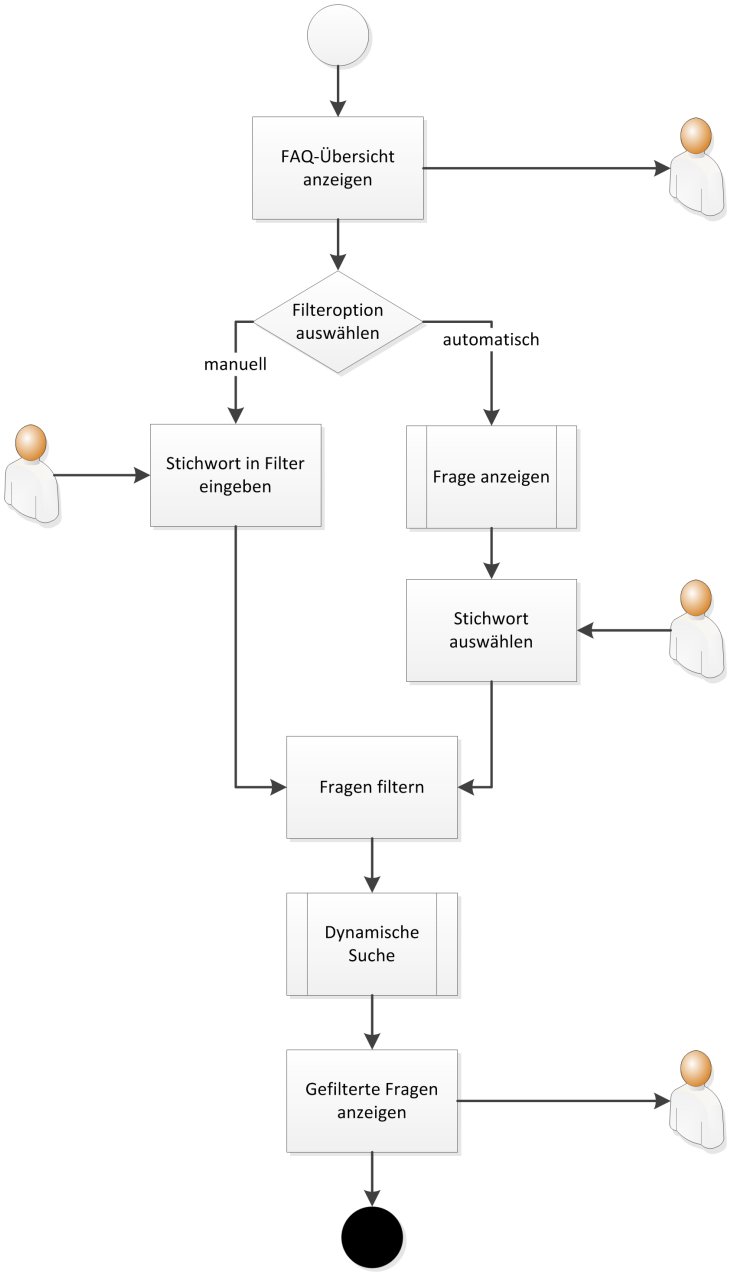


Abbildung 3.19: Ablauf zum Filtern von Fragen

Welche Aktionen werden hier durchgeführt:

Der Benutzer kann Stichwörter in einem Suchfilter eingeben, wodurch ihm nur Fragen angezeigt werden, die die entsprechenden Stichworte beinhalten. Alternativ besteht auch

die Möglichkeit, die Stichwörter in den bereits aufgeklappten Fragen nur zu berühren und es wird automatisch nach dem berührtem Stichwort gefiltert. Zudem wird nach dem Anzeigen der gefilterten Fragen auch eine dynamische Suche gestartet, in der noch nicht vorhandene Fragen mit dem Stichwort vom Server geladen und dem Benutzer angezeigt werden (siehe Kapitel 3.3.4). In diesem Fall werden dem Benutzer neue FAQ-Einträge angezeigt.

3.3.8 Optionen

In den Optionen (OP) kann der Benutzer die App nach seinen Wünschen anpassen. Dabei existieren drei Einstellungskategorien: Die Updates, das Studium und die Interneteinstellungen.

OP 1: Updates
Der Vorgang für die Updates ist in Abbildung 3.20 abgebildet.

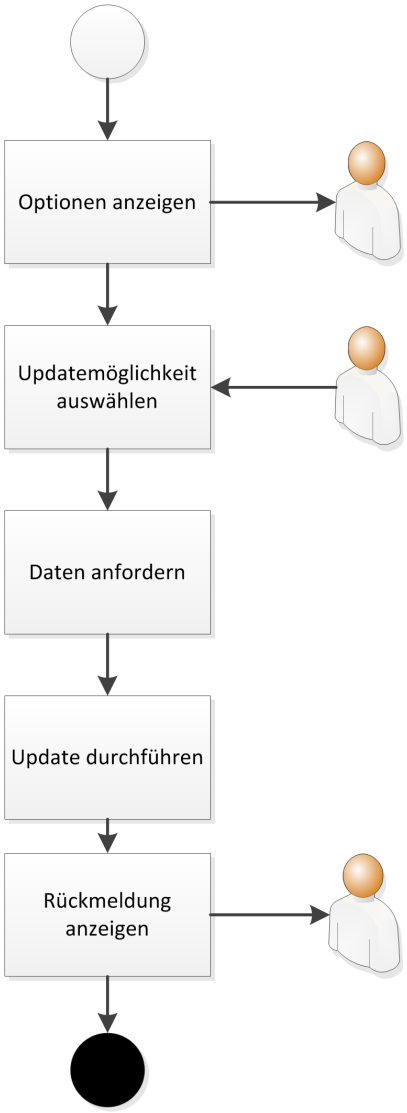


Abbildung 3.20: Ablauf für die Durchführung von Updates

Welche Aktionen werden hier durchgeführt:

In den Updates stehen dem Nutzer drei Möglichkeiten zur Verfügung. Er kann nach Updates suchen, die beispielsweise neue Profile hinzufügen oder bestehende Profile ändern. Er kann auch nach Veranstaltungen suchen, wobei er hier auswählen kann, aus welchen Profilen neue Veranstaltungen geladen werden sollen. Zudem können auch neue FAQs geladen werden.

Auf welche Ausnahmen/Besonderheiten müssen hierbei geachtet werden:
Der Benutzer erhält stets eine Rückmeldung, in dem er darüber informiert wird, ob ein Update erfolgreich durchgeführt worden ist oder ob Probleme auftraten.

OP 2: Profil
Der Vorgang für die Profiländerung ist in Abbildung 3.21 abgebildet.

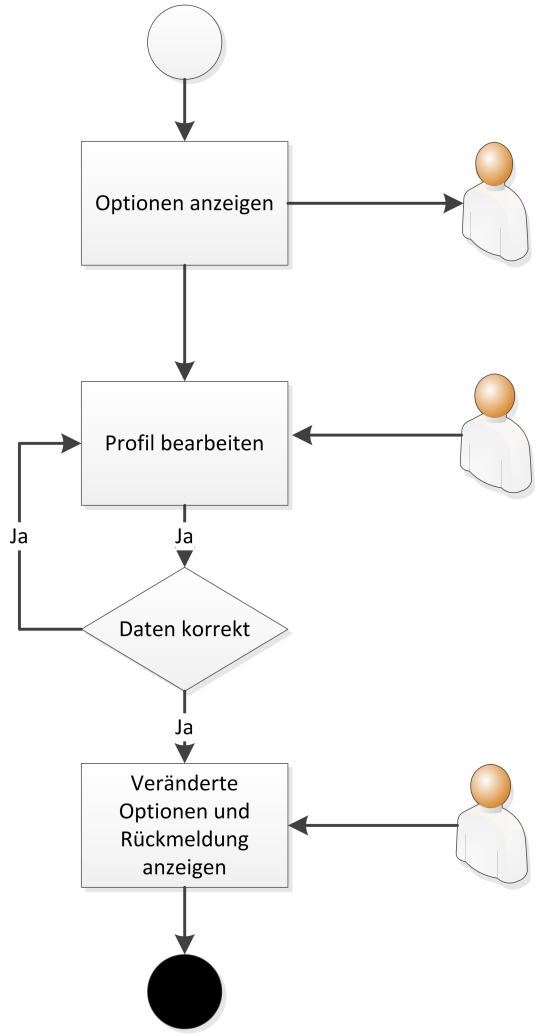


Abbildung 3.21: Ablauf zur Änderung des Profils

Welche Aktionen werden hier durchgeführt:
In den Profileinstellungen kann der Benutzer sein Profil ändern, Semester hinzufügen oder auch Semester löschen. Falls er sein Profil wechseln möchte, muss er dazu das neue Profil angeben. Unter Umständen ist auch die Angabe von Zusatzinformationen erforderlich, wie zum Beispiel die Angabe des Anwendungsfaches.
Auf welche Ausnahmen/Besonderheiten müssen hierbei geachtet werden:
Der Benutzer wird über jegliche Änderungen informiert und erhält zusätzlich Rückmeldungen bei fehlenden Daten, wenn zum Beispiel kein Anwendungsfach ausgewählt wurde.

OP 3: Interneteinstellungen
Der Vorgang für die Bearbeitung der Interneteinstellungen ist in Abbildung 3.22 abgebildet.

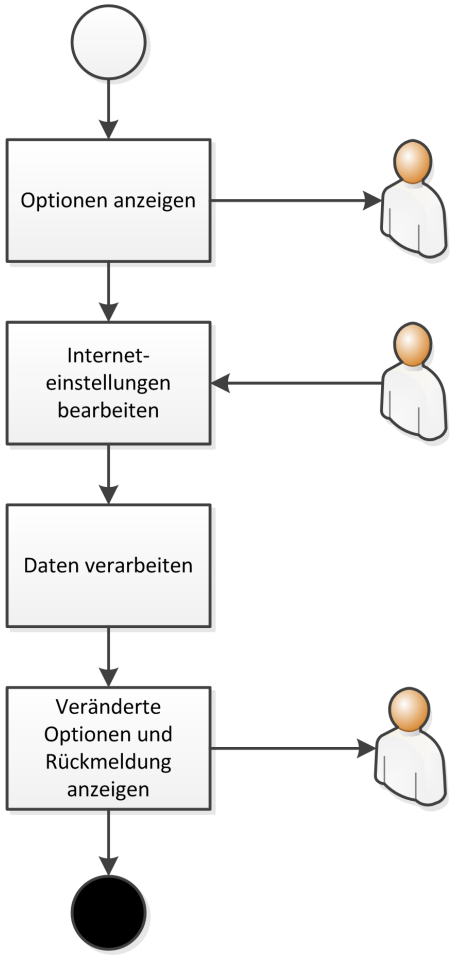


Abbildung 3.22: Interneteinstellungen bearbeiten

Welche Aktionen werden hier durchgeführt:

Der Benutzer hat hier die Möglichkeit, die Interneteinstellungen für die dynamische Suche einzustellen. Dabei kann er den Internetzugang über Wlan oder Mobilfunk einstellen.

Auf welche Ausnahmen/Besonderheiten müssen hierbei geachtet werden:

Bei einer Änderung der Interneteinstellungen erhält der Benutzer eine Rückmeldung.

3.4 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen beschreiben die Qualitätseigenschaften der zu entwickelnden App. Für die Klassifizierung der Anforderungen wird die Norm ISO/IEC 9126 verwendet [Hor07]. Dabei werden in diesem Kapitel auf die Qualitätsmerkmale dieser Norm eingegangen, die in Abbildung 3.23 dargestellt sind.

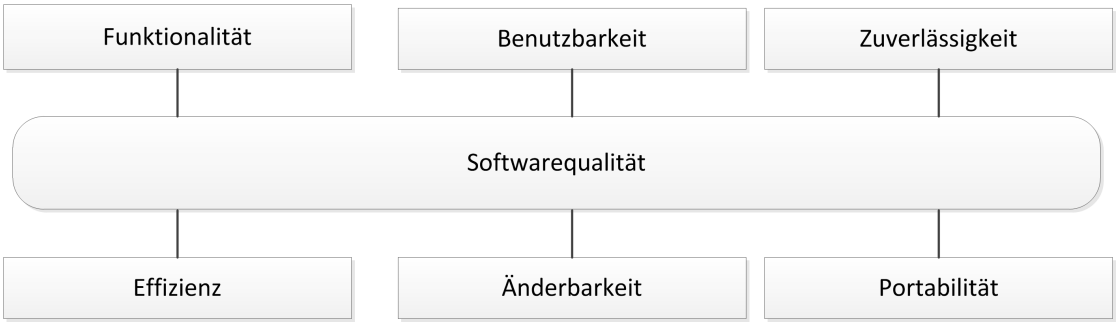


Abbildung 3.23: Qualitätsmerkmale von Softwaresystemen nach ISO 9126

Für die Funktionalität gilt, dass der Benutzer die in Kapitel 3.3 beschriebenen Anforderungen in der angegeben Form tatsächlich auch verwenden kann.
Die Bedienbarkeit soll für den Benutzer angenehm sein und der Aufwand für die Verwendung der Funktionen so niedrig wie möglich gehalten werden. Unnötige Arbeitsschritte sollen zudem vermieden werden (siehe Kapitel 4.3). Die Funktionen sollen zudem für den Benutzer verständlich und leicht erlernbar sein.
Zudem soll die App zuverlässig funktionieren. Falls während der Benutzung Fehler auftreten, müssen diese abgefangen werden und das System in einen vordefinierten Zustand gebracht werden. Rückmeldungen über Fehler sollen dem Benutzer die Möglichkeit bieten, eingegebene Daten zu korrigieren und Informationen über den Grund des Fehlers zu erhalten (siehe Kapitel 4.3.10).
Die Funktionen von ULMUS sollen dabei effizient umgesetzt werden, sodass für den Benutzer keine langen Wartezeiten entstehen. Das bedeutet unter anderem, dass nur Funktionen, die zwingend eine Internetverbindung benötigen, auf der Server-Seite umgesetzt werden. Treten Wartezeiten auf, sollen falls möglich, andere Funktionen ausgeführt werden können.

Die Architektur des Systems soll so umgesetzt werden, dass eine spätere Änderung, Erweiterbarkeit und Wartbarkeit des Systems ermöglicht wird (siehe Kapitel 4.1). Dabei sollen neue oder zu ändernde Inhalte ohne Änderungen des Systems erfolgen können (siehe Kapitel 4.2). Eine weitere wichtige Eigenschaft ist die Portierbarkeit. Dabei soll die App ohne große Änderungen auf verschiedenen Betriebssystemen funktionieren. In Abbildung 3.24 ist eine Übersicht der weltweiten Marktanteile mobiler Betriebssysteme im 2. Quartal 2013 abgebildet [Mar13]. Es ist zu sehen, dass die Mehrheit das Betriebssystem Android (79,3%) verwendet, gefolgt von iOS mit einem Marktanteil von 13,2%. Hier gilt es daher zu berücksichtigen, dass die App durch minimale Anpassungen auf diese Systeme portiert werden kann. In Kapitel 5 wird dazu näher auf die Faktoren für die Portierung eingegangen.

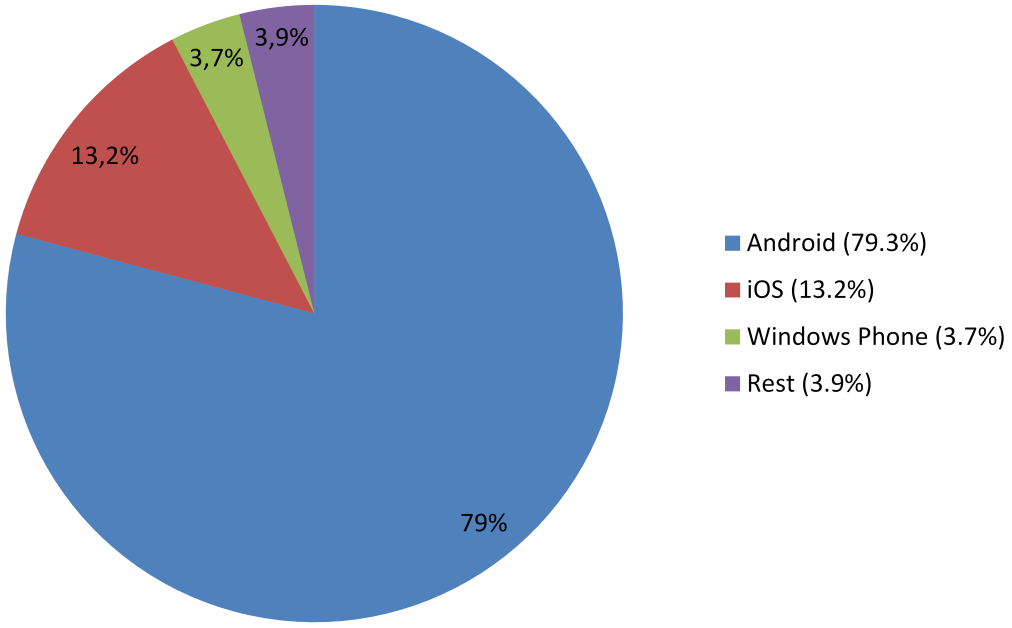


Abbildung 3.24: Marktanteile mobiler Systeme im 2. Quartal 2013

4

Entwurf

In diesem Kapitel wird auf Basis der Anforderungen ein Entwurf von ULMUS erstellt. In Kapitel 4.1 wird dazu die Architektur erläutert. Kapitel 4.2 stellt das Datenmodell vor und in Kapitel 4.3 wird das Design und die Bedienbarkeit mithilfe von Mockups erklärt.

4.1 Architektur

In diesem Kapitel wird die Client-Server Architektur beschrieben. Diese wird für einige Anforderungen aus Kapitel 3.3 benötigt, wie zum Beispiel für die dynamische Suche. Zwischen dem Client und dem Server findet dabei eine Datenübertragung statt, die für gewöhnlich vom Client initiiert wurde. Die Funktion eines Servers kann unter anderem die Datenbankhaltung sein, mit der ein Client nur Daten vom Server anfordert. Es ist auch möglich, dass der Server verschiedene Funktionen anbietet, die der Client verwen-

den kann. Im Folgenden wird erklärt, wie Clients umgesetzt werden können und wie die dazugehörige die Server-Seite aussieht [MKS11]:

Thin Client:
Ein *Thin Client* ist sehr stark von der Server-Seite abhängig. Er dient lediglich als eine Benutzerschnittstelle, während sich die eigentliche Anwendungslogik auf der Seite des Servers befindet. Dadurch ergeben sich einige Vorteile im Vergleich zu anderen Architekturen. Die Implementierung und Wartbarkeit des Clients ist einfacher, da nur die Benutzereingaben abgefragt und an den Server gesendet werden müssen. Dadurch sinken auch die Hardwareressourcen für den Client, da der Hauptrechenaufwand durch den Server übernommen wird. Außerdem ist die Wartbarkeit der App einfacher, da Änderungen der Anwendungslogik nur die Server-Seite betreffen und die Client-Seite davon unberührt bleibt. Dadurch entfällt der Aufwand für die Verteilung der Änderungen an die Clients. Nachteilig bei dieser Architektur kann eine höhere Wartezeit bei einer hohen Datenkommunikation oder eine hohe Auslastung des Servers sein. Außerdem benötigt der Client eine ständige Verbindung zum Server. Falls der Client den Server nicht erreichen kann, ist es nicht mehr möglich, Funktionen der App zu verwenden.

Thick Client:
Bei einem sogenannten *Thick Client* liegt die Anwendungslogik auf der Client-Seite und eine stetige Verbindung zu einem Server ist nicht notwendig. Die Daten können ebenfalls auf der Client-Seite gehalten werden. Nachteilig sind hierbei im Vergleich zum Thin Client die höheren Hardwareanforderungen und komplexere Aktualisierungen der Applikation, da dies in den meisten Fällen auch auf der Client-Seite nötig ist.

Hybrid Client:
Ein *Hybrid Client* ist eine Mischung aus Thick und Thin Client. Dabei können mehrere Varianten vorkommen. Zum Beispiel können verschiedene Funktionen je nach Bedarf auf der Client- oder auf der Server-Seite implementiert werden. Die App kann auch ohne ständige Anbindung zum Server funktionieren, wobei hier nicht alle Funktionalitäten

verfügbar sein müssen.

In dieser App wird eine hybrider Client verwendet. Wichtige Daten werden auf der Client-Seite gespeichert und die App kann ohne Anbindung zum Server mit einigen Einschränkungen verwendet werden. Zudem ist es möglich, neue Daten gezielt vom Server zu laden und die App dabei aktuell zu halten. Jedoch können Änderungen der Anwendungslogik oder des Benutzerinterfaces der App nicht vom Server geladen werden, sondern müssen durch eine Aktualisierung der App durchgeführt werden.

Eine Übersicht über die Gesamtarchitektur bietet dabei die Abbildung 4.1. Der Nutzer kommuniziert über die App mit dem Server. Dabei findet die Kommunikation nur über HTTP statt. Der Server kann dabei auf eine Datenbank zugreifen, die ebenfalls auf der Server-Seite vorhanden ist. Im Folgenden wird die clientseitige Architektur als auch die serverseitige Architektur näher beschrieben.

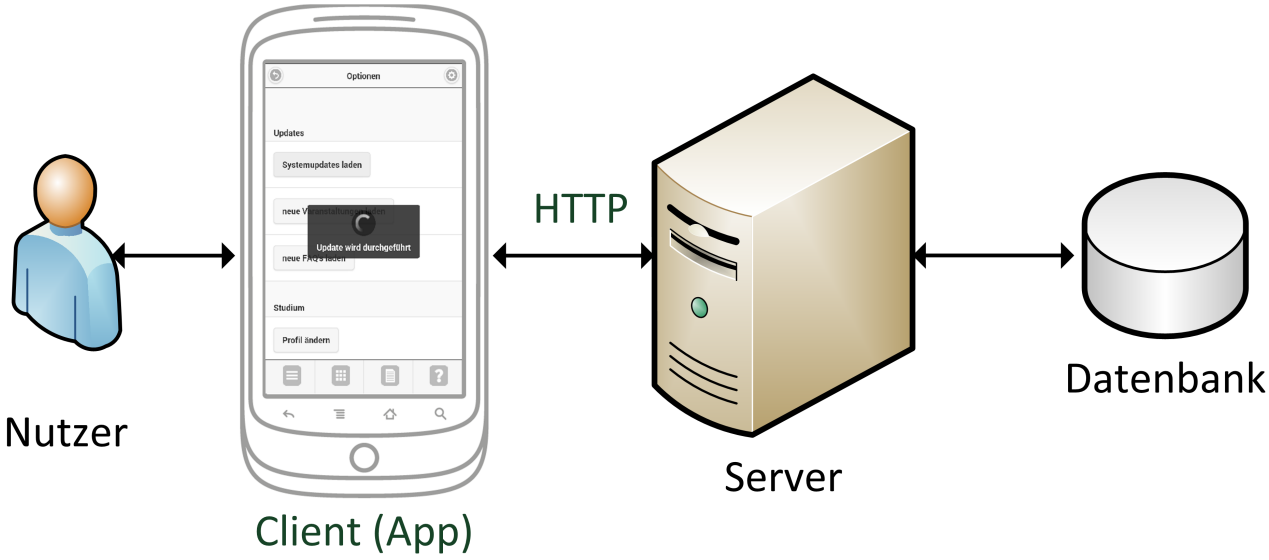


Abbildung 4.1: Gesamtarchitektur des Systems

4.1.1 Architektur auf Seite des Clients (App)

Die Architektur auf Seite des Clients (App) ist in drei Komponenten aufgebaut, der *View*, dem *Controller* und dem *Core* und erfüllt damit die Struktur des *MVC-Patterns* [LR01]. Jede dieser Komponenten ist für verschiedene Aufgaben zuständig, die im Folgenden kurz beschrieben werden. Durch die Trennung der Aufgabenbereiche in mehrere Komponenten können diese mehrfach verwendet werden. Außerdem kann das System durch Hinzufügen von neuen Komponenten einfach erweitert werden. Abbildung 4.2 beschreibt dabei die Client Architektur.

View:
Die View ist für die Darstellung der Daten auf der Benutzeroberfläche zuständig. Sie stellt zum Beispiel die Semesteranzeige (siehe Kapitel 3.3) dem Benutzer dar.

Controller:
Der Controller nimmt Eingaben des Benutzers über die View entgegen und ist für die Logik hinter der App und Delegierung der Aufgaben zuständig. Wenn der Benutzer zum Beispiel in der Semesteranzeige die detaillierten Informationen eines Semesters einsehen will, nimmt der Controller die Eingabe entgegen und kümmert sich darum, dass die notwendigen Daten in der View angezeigt werden.

Core:
Der Core dient als Unterstützung für den Controller und verarbeitet spezielle Arbeitsschritte. Zudem wird hier mit der Datenbank kommuniziert und er dient als Schnittstelle zwischen Server und Client. Möchte der Benutzer zum Beispiel neue Veranstaltungen vom Server laden (siehe Kapitel 3.3), werden im Core die angeforderten Daten an den Server gesendet.

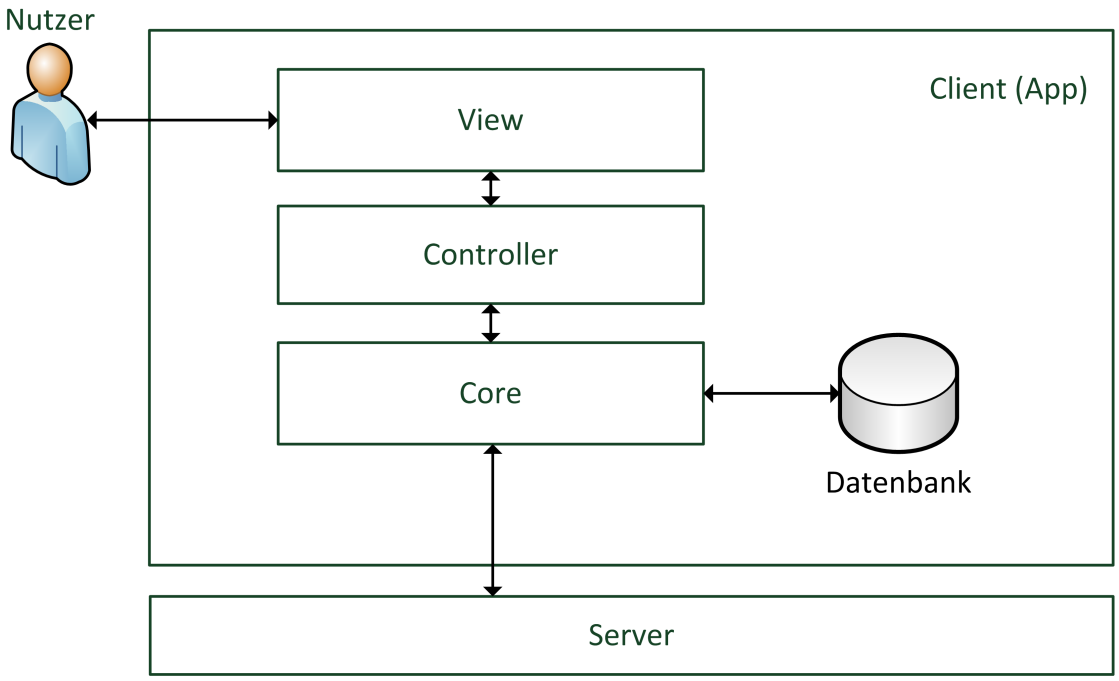


Abbildung 4.2: Client-Architektur von ULMUS

4.1.2 Architektur auf Seite des Servers

Die Server-Architektur von ULMUS kann in zwei Bereiche unterteilt werden, einem sogenannten *Dispatcher* und den *Modulen*. Diese Grundstruktur ist in Abbildung 4.3 grafisch dargestellt. Anfragen vom Client, d.h. der App, an den Server werden dabei vom Dispatcher entgegengenommen und an die jeweiligen Module weitergeleitet. Die Module verarbeiten die Anfragen und können nach Bedarf auf die Datenbank zugreifen. Abschließend werden die Ergebnisse der Verarbeitung wieder an den Client gesendet. Durch die Aufteilung der Aufgabenbereiche in verschiedene Module, das auch unter dem Prinzip *Seperation of concerns*[1] bekannt ist, wird die Programmierung erleichtert und übersichtlicher [HL95]. Ein weiterer Vorteil ist die Wiederverwendbarkeit von Komponenten.

[1] zu dt. „Trennung der Belange“

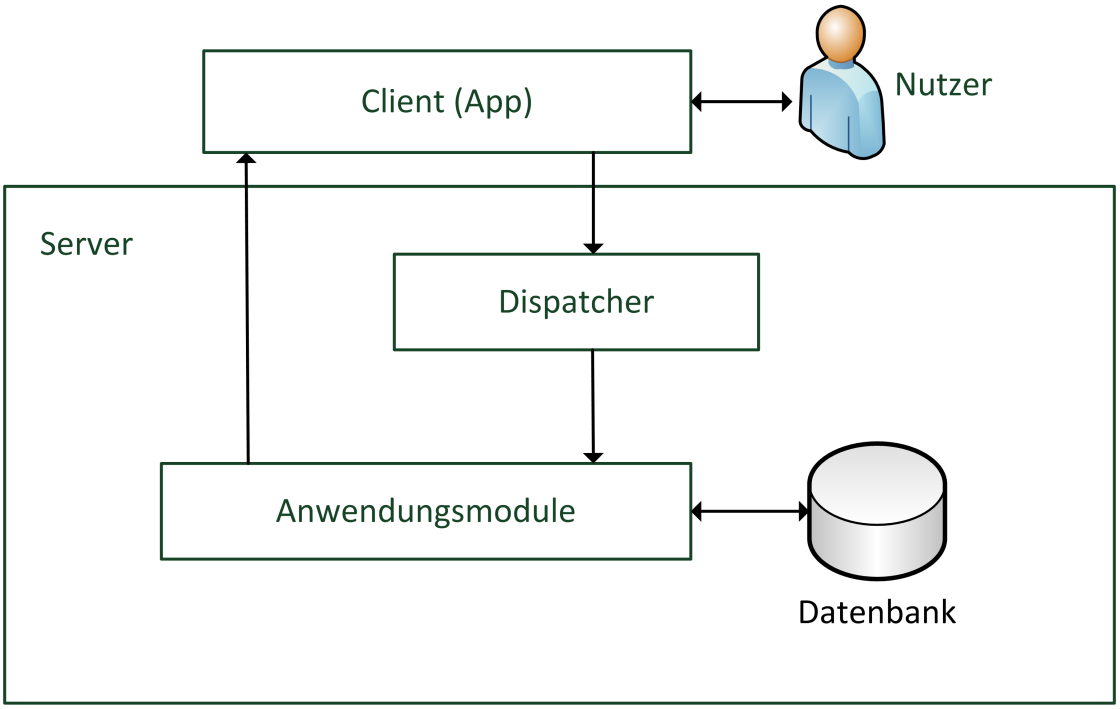


Abbildung 4.3: Server-Architektur von ULMUS

4.2 Datenmodell

Das Datenmodell wird anhand der erhobenen funktionalen Anforderungen aus Kapitel 3.3 erstellt und bietet eine wichtige Grundlage für die spätere Umsetzung von ULMUS. Es berücksichtigt dabei Anforderungen wie zum Beispiel die synchrone Datenhaltung auf der Client- wie auch auf der Server-Seite. Es wird dabei ein relationales Datenmodell verwendet. Es beschreibt, wie die Daten gehalten werden und welche Abhängigkeiten untereinander bestehen. Im Folgenden werden zwei Datenmodelle erklärt. Ersteres beschreibt das Datenmodell auf der Client-Seite. Dieses hat eine sehr große Ähnlichkeit mit dem Datenmodell der Server-Seite, da wichtige Daten auf beiden Seiten zum Zwecke der Updatemöglichkeiten gehalten werden müssen. Daher werden in Kapitel 4.2.2 nur auf die Unterschiede der beiden Datenmodelle eingegangen. Die Datenmodelle werden als ER-Diagramme in Martin-Notation dargestellt.

4.2.1 Datenmodell auf Seite des Clients (App)

Das Datenmodell auf Seite des Clients kann in vier Hauptkomponenten unterteilt werden: Dem *Profil*, der *lokalen Datenhaltung von Veranstaltungen*, der *externen Datenhaltung von Veranstaltungen* und der *FAQ-Komponente*. Diese vier Komponenten werden im Folgenden näher beschrieben.

4.2.1.1 Profil:

Die Profilkomponente des Datenmodells auf Seite des Clients ist in Abbildung 4.4 dargestellt und bildet eine wichtige Basis von ULMUS. Daher wird sie in diesem Kapitel genauer betrachtet und anhand von Beispielen aus der FSPO 2010 und FSPO 2012 Informatik erläutert.

Mit dem Datenmodell lassen sich für eine Studienrichtung vielfältige Prüfungsordnungen erstellen. Für jeden Studiengang und und für jede Prüfungsordnung existiert in der Entität Studiengang ein Eintrag. Hier können weitere Zusatzinformationen gespeichert werden, wie die Unterscheidung der Modulbezeichnungen. Weitere Informationen wie die zu erbringenden Leistungspunkte pro Fachsemester werden in einer eigenen Entität gespeichert.
Die verschiedenen Module werden in der Entität Module gespeichert. Hierbei werden die Module in drei verschiedene Arten eingeteilt.
Mit der ersten Art werden diejenigen Module bezeichnet, die nur Veranstaltungen beinhalten. Module der zweiten Art besitzen weitere Untermodule der ersten Art. Mit der zweiten Art müssen Studenten bis zu n verschiedene Untermodule auswählen, wobei n hier die festgelegte Anzahl der auszuwählenden Untermodulen definiert. Dadurch können zum Beispiel Module des Anwendungsfaches zu einem Hauptmodul zusammengefasst werden.
Zuletzt fasst die dritte Art Module in einem Hauptmodul zusammen, aus denen aus allen Modulen Veranstaltungen frei gewählt werden können. Dadurch lassen sich beispielsweise die Kernmodule aus der FSPO 2012 in ein Hauptmodul zusammenfassen. Um dabei zu berücksichtigen, dass im Kernmodul (bzw. Kernfach) Praktische und Angewandte

Informatik mindestens 12 LP zu wählen sind, kann dieses Modul getrennt als weiteres Modul der ersten Art erfasst werden. Durch die Entität Modulreferenzen werden dabei die Module der zweiten und dritten Art den Hauptmodulen zugeordnet.
Die Entität Veranstaltungen speichert alle Veranstaltungen. Der Name ist dabei in der Entität optional und dient nur zur Übersicht. Wichtiger hierbei ist die Entität ModulVeranstaltungen, die Veranstaltungen verschiedenen Modulen zuordnet, welches eine große Rolle bei einem Profilwechsel spielt.
Bei einem Wechsel von der FSPO 2010 Informatik zu Medieninformatik können dadurch Veranstaltungen automatisch den richtigen Modulen zugeordnet werden. Ist eine Zuordnung nicht möglich, da das Modul im neuen Studiengang nicht vorhanden ist, werden die betreffenden Veranstaltungen in das Zusatzmodul übertragen. Auch ein Wechsel von der FSPO 2010 Informatik zur neueren FSPO 2012 wäre möglich. Hierbei können sogar alle Veranstaltungen den richtigen Modulen zugeordnet werden.
Insgesamt ist es dadurch möglich, verschiedene Prüfungsordnungen mit dem Datenmodell abzubilden und gleichzeitig bei einem Profilwechsel dem Anwender die Übernahme der Veranstaltungen zu erleichtern.

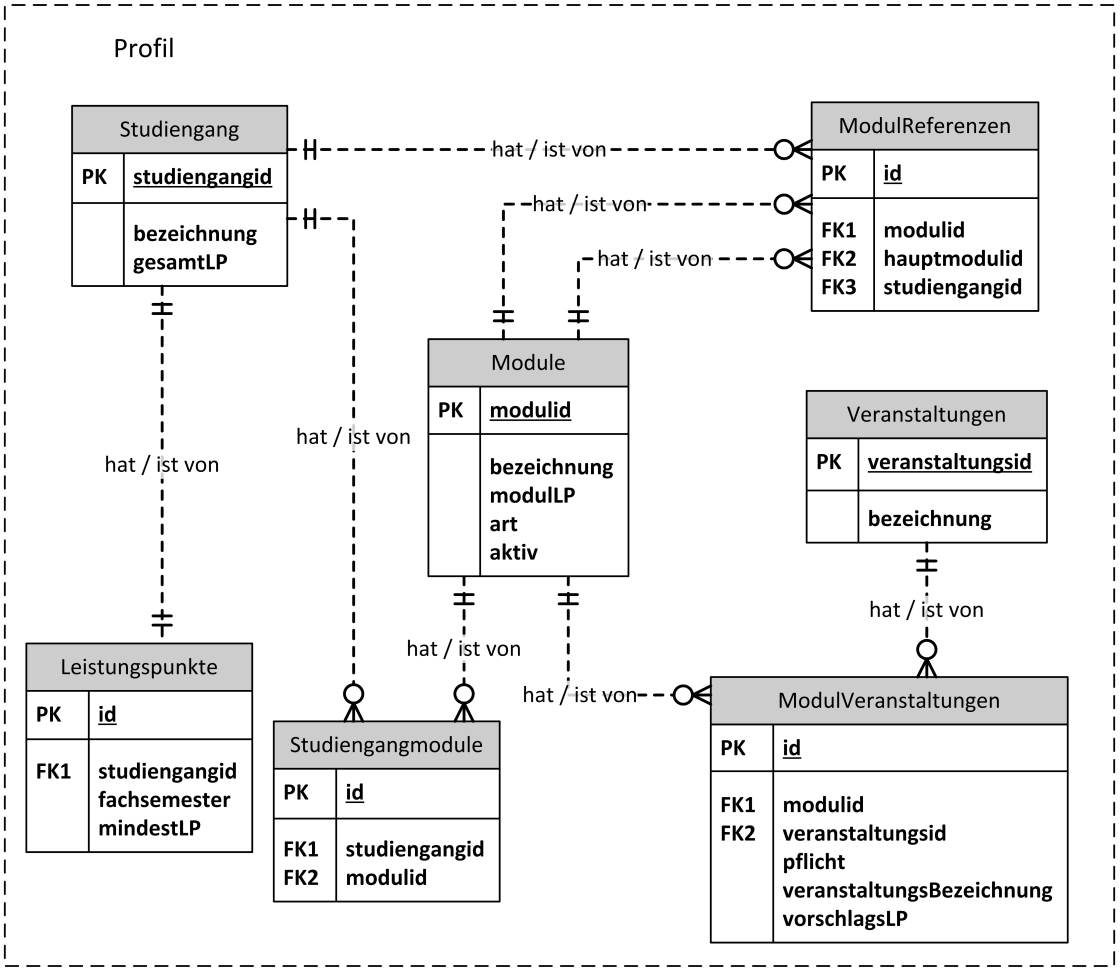


Abbildung 4.4: Datenmodell der Profilkomponente

4.2.1.2 Lokale Datenhaltung von Veranstaltungen:

Diese Komponente des Datenmodells hängt sehr stark von der Profilkomponente ab, da aus der Profilkomponente hervorgeht, welchen Modulen entsprechende Veranstaltungen zugeordnet werden können. Wie in Abbildung 4.5 zu sehen ist, werden in der Komponente zur lokalen Datenhaltung von Veranstaltungen die Veranstaltungen des Benutzers mit allen relevanten Daten gespeichert. Aufgrund der Tatsache, dass verwandte Studiengänge bis zu einem gewissen Grad identische Module in ihren Modulhandbüchern besitzen, können bei einem Wechsel des Profils zu einem anderem Profil mit

gleichwertigem Abschluss, zumindest jene Veranstaltungen in den identischen Modulen überführt werden. Alle anderen Veranstaltungen werden in das Modul Zusatzfachveranstaltung überschrieben. Der Benutzer kann zu jeder Zeit diese Veranstaltungen in andere Module überführen. Jedoch muss hierbei beachtet werden, dass dies nur für Studiengänge mit gleichwertigem Abschluss (z.B. Bachelor) gilt. Bei einem Wechsel zu einem Studiengang mit einem anderem Abschluss (z.B. Bachelor zu Master) werden nur die Veranstaltungen aus dem Modul Zusatzfachveranstaltung übernommen, die der Benutzer daraufhin in mögliche Module eintragen lassen kann. Das hat den Vorteil, dass Zusatzveranstaltungen, die im Bachelor für den Master gehört wurden, einfacher übernommen werden können.

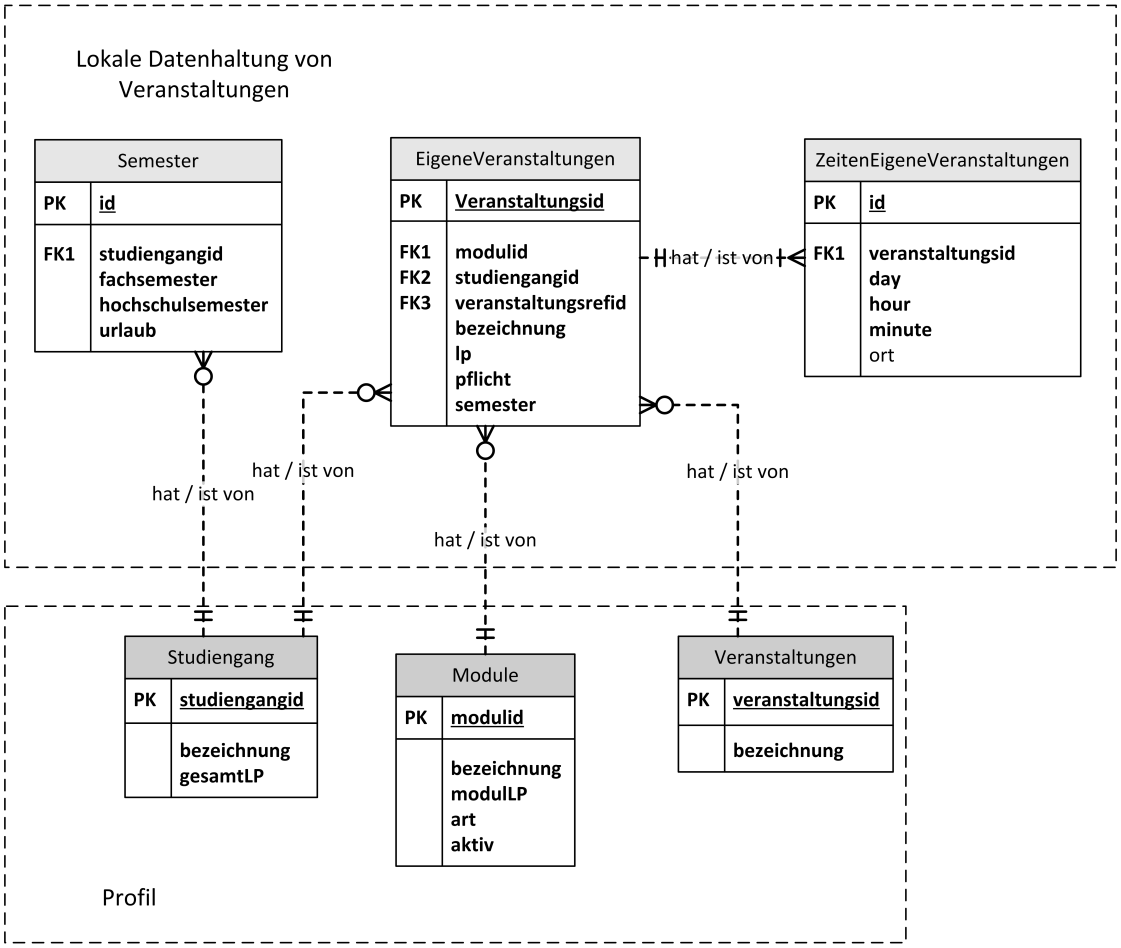


Abbildung 4.5: Komponente zur lokalen Datenhaltung von Veranstaltungen im Datenmodell

4.2.1.3 Externe Datenhaltung von Veranstaltungen:

Abbildung 4.6 stellt die Komponente für die externe Datenhaltung von Veranstaltungen im Datenmodell dar. Wie auch in der lokalen Datenhaltung von Veranstaltungen hängt diese Komponente sehr stark von der Profilkomponente ab. In ihr sind Veranstaltungen der letzten Semester gespeichert. Diese Veranstaltungsdaten kann der Benutzer dabei von einem Server beziehen. Sie wurden von anderen Benutzern eingereicht und enthalten bereits Daten zu den Veranstaltungen, die der Benutzer übernehmen kann.

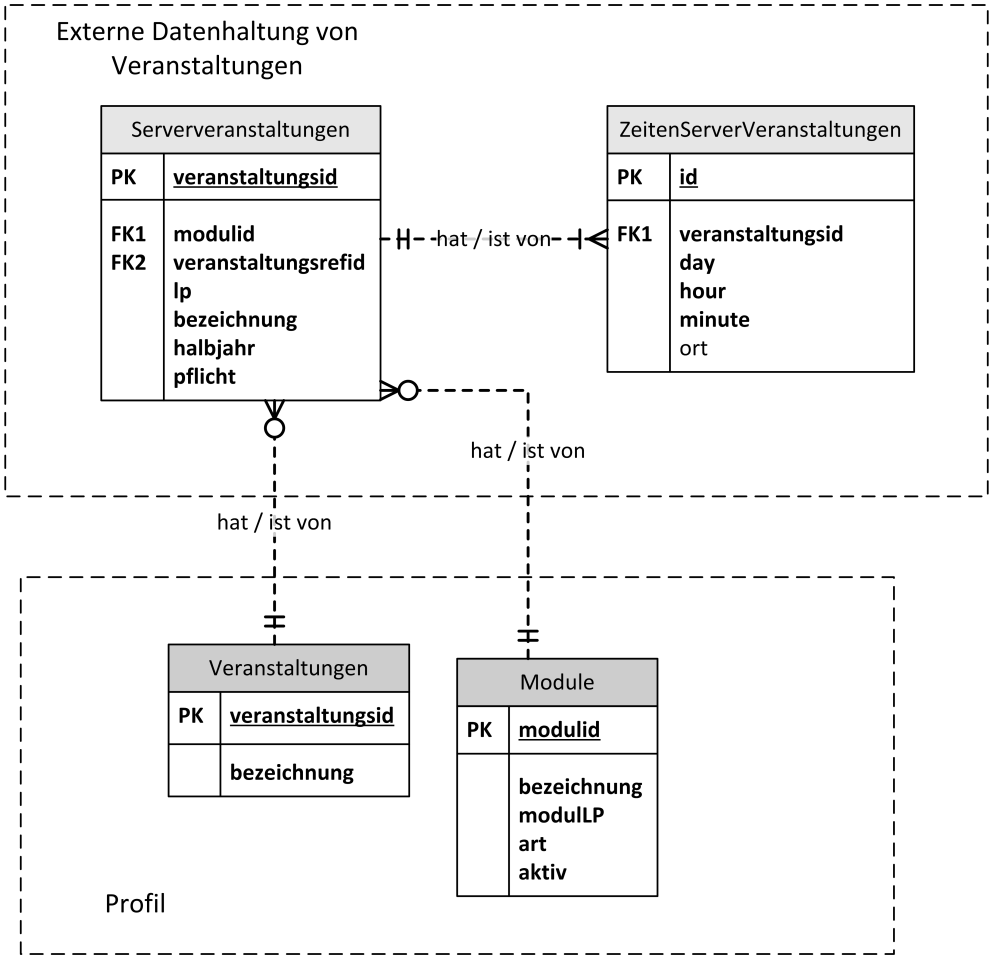


Abbildung 4.6: Komponente zur externen Datenhaltung von Veranstaltungen im Datenmodell

4.2.1.4 FAQ-Komponente:

Die FAQ-Komponente im Datenmodell ist unabhängig von allen anderen Komponenten. Sie ist, wie der Name schon sagt, für die Speicherung der FAQ-Daten zuständig und in Abbildung 4.7 dargestellt.

Abbildung 4.7: FAQ-Komponente im Datenmodell

4.2.2 Datenmodell auf Seite des Servers

Das Datenmodell auf Seite des Servers besteht aus vier Komponenten: Dem Profil, der FAQ-Komponente, der externen Datenhaltung von Veranstaltungen, und der Komponente Versionsverwaltung. Der Aufbau der Profil- und der FAQ-Komponente stimmen dabei mit dem Aufbau des Datenmodells auf der Client-Seite überein. Bevor eingereichte Veranstaltungen von Benutzern anderen zur Verfügung gestellt werden können, müssen die Daten auf dem Server erst überprüft werden. Dazu enthält die Komponente zur externen Datenhaltung von Veranstaltungen noch ein Feld, mit der die Freigabe von Veranstaltungen gesteuert werden kann. Ansonsten stimmt diese Komponente mit dem Datenmodell auf Seite des Clients überein. Die Komponente Versionsverwaltung wird im Folgenden erklärt.

4.2.2.1 Versionsverwaltung:

Die Versionsverwaltung ist in Abbildung 4.8 dargestellt. Eine Versionierung dient dazu, dass ein Austausch von Daten nur möglich ist, wenn die Versionen auf beiden Seiten übereinstimmen. Damit werden Profilinformationen synchron gehalten, womit keine Fehler auftreten, die durch asynchrone Datenhaltung entstehen. Mit der Versionsverwaltung können dabei nicht nur die Profile synchron gehalten werden, sondern auch nicht mehr aktuelle externe Veranstaltungen oder FAQ-Einträge auf der Client-Seite verändert werden.

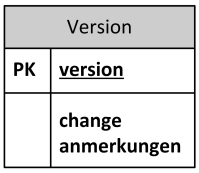


Abbildung 4.8: Komponente Versionsverwaltung im Datenmodell auf Seite des Servers

4.3 Mockups

In diesem Kapitel werden Mockups vorgestellt, die die zukünftige Benutzeroberfläche aufzeigen, mit der die funktionalen Anforderungen aus Kapitel 3.3 unterstützt werden. Hierbei werden auch einige designtechnische Aspekte erläutert, mithilfe derer eine benutzerfreundliche Bedienung erreicht wird und dadurch viele Anforderungen aus Kapitel 3.4 umgesetzt werden. Viele dieser Aspekte beruhen dabei auf dem Buch *Mobile Design Pattern Gallery* [Nei12].

4.3.1 Initialisierung

Abbildung 4.9 stellt ein Mockup für die Startseite, wie sie beim erstmaligen Ausführen der App zu sehen ist, dar. Hier kann der Benutzer sein Profil einrichten und wird daraufhin zur Semesterübersicht weitergeleitet. Die Startseite wird ebenfalls fast identisch bei einem Profilwechsel verwendet, wobei in diesem Fall zur Optionsseite weitergeleitet wird (siehe Kapitel 4.3.9).

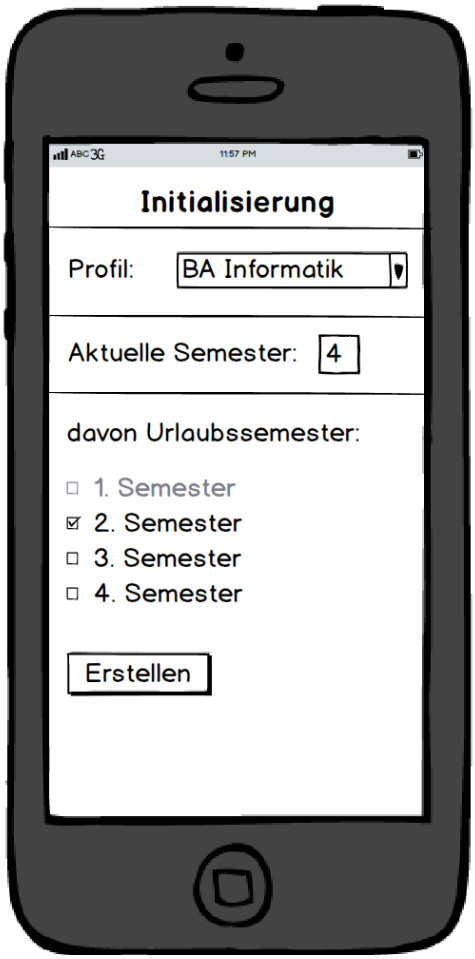


Abbildung 4.9: Mockup für die Einstellungen beim erstmaligen Start

4.3.2 Semesterübersicht

In der Abbildung 4.10 wird das Mockup für die Semesterübersicht dargestellt, welche die Startseite von ULMUS nach der Initialisierung zeigt. Dabei ist erkennbar, dass standardmäßig das höchste Semester aufgeklappt ist und somit Informationen zum aktuellen Semester sofort ins Auge springen. Weitere Informationen zu anderen Semestern können dabei sofort nachgeschlagen werden. Neben den einzelnen Veranstaltungen existiert zudem ein Änderungsfeld in Form eines Stiftes, mit dem die Informationen von Veranstaltungen verändert werden können.

Für die Navigation wurde das Tab-Menü Pattern ausgewählt, das auch in der App für die Uni Tübingen (siehe Kapitel 3.1.1, Abbildung 3.4) verwendet wird.
Die Navigation ist im Fenster ganz unten eingeblendet, um eine daumenfreundliche Navigierung zu den Hauptinhaltsbereichen zu ermöglichen. Durch die farbliche Hervorhebung des aktuell ausgewählten Navigationsfeldes ist sofort erkennbar, in welchem Bereich der App der Benutzer sich befindet. Oben rechts im Kopfbereich werden weitere mögliche Aktionen angezeigt, die zu weiteren Funktion der App führen [Leh12]. Im dargestellten Fall kann der Benutzer zu den Optionen gelangen.

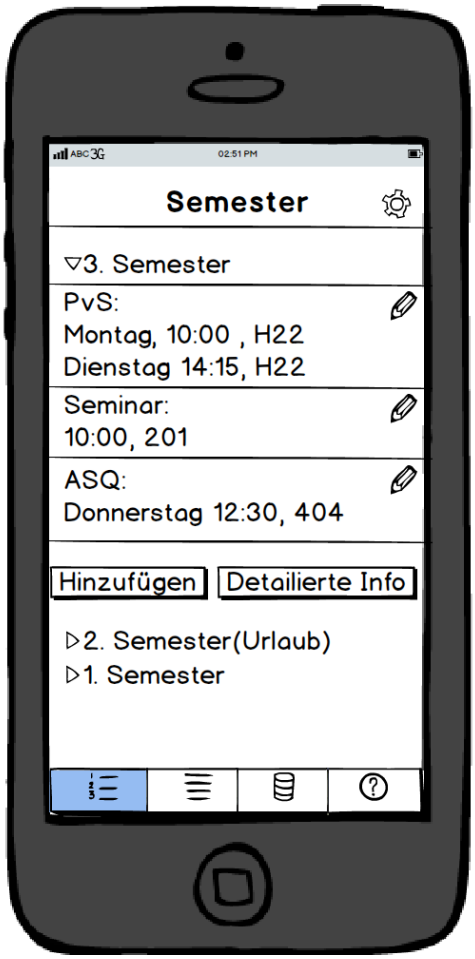


Abbildung 4.10: Mockup für die Semesterübersicht

4.3.3 Semester detailiert

Zu diesem Fenster wird über das Hauptfenster Semesterübersicht navigiert und es bietet Zusatzinformationen zu einem bestimmten Semester an. Wie im Mockup in Abbildung 4.11 zu sehen ist, existiert dazu auch eine Zurücktaste oben links im Bildschirm, mit der wieder zum Hauptbereich navigiert werden kann. Dies ist ein typisches Designelement von iOS und Android Apps [Men13b].

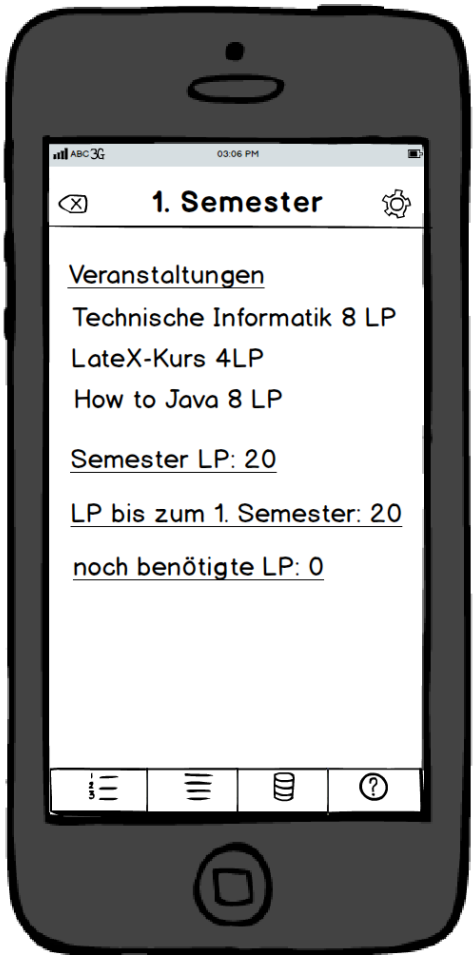


Abbildung 4.11: Mockup für die detailierte Sicht eines bestimmten Semesters

4.3.4 Gesamtübersicht

In der Gesamtübersicht wird der gesamte Verlauf des ausgewählten Profils dargestellt. In Abbildung 4.12 ist dabei im Kopfbereich des Mockups ein weiterer Aktionsbutton in Form einer Kamera abgebildet, mit dem der Benutzer die Gesamtübersicht als PDF exportieren kann.

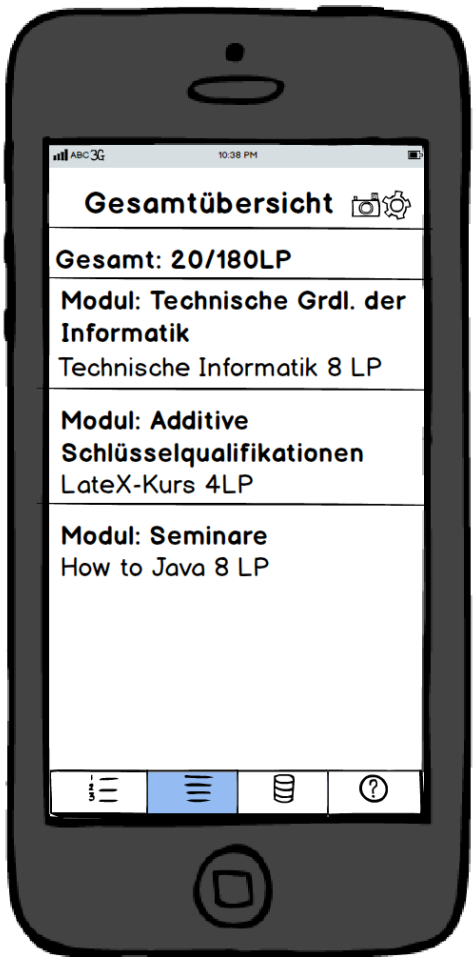


Abbildung 4.12: Mockup für die Gesamtübersicht

4.3.5 Veranstaltungen

In Abbildung 4.13 ist die Oberfläche für die Veranstaltungen abgebildet. Im Kopfbereich hat der Benutzer die Möglichkeit, über einen Aktionsbutton in Form eines Plus-Zeichens neue Veranstaltungen zu erstellen.
Für die Veranstaltungen vom Server existiert dabei eine Suchfunktion, die schon ab Eingabe eines Zeichens die Veranstaltungen filtert und andere Veranstaltungen vom Server lädt. Im Suchfeld existiert zusätzlich eine Löschfunktion, mit der Eingaben mit nur einer Berührung gelöscht werden können.

Abbildung 4.13: Mockups für die Veranstaltungsübersicht

4.3.6 Veranstaltungen erstellen, ändern, löschen

In Abbildung 4.14 ist links ein Mockup für die Erstellung einer Veranstaltung abgebildet und rechts eine beispielhafte Abbildung für eine Änderung einer Veranstaltung.
Hier wird berücksichtigt, so viele Felder wie möglich schon automatisch auszufüllen, so dass der Benutzer nur relevante Felder bearbeiten muss. Durch die Automatisierung von solchen Aufgabenschritten verbessert sich die Benutzerbedienung. Im Bezeichnungsfeld werden ab Eingabe des ersten Zeichens passende Veranstaltungen dynamisch angezeigt und je nach Auswahl das Modulfeld automatisch angepasst. Außerdem wird das Bezeichnungsfeld automatisch ausgefüllt, falls ein Modul im Modulfeld ausgewählt wurde und das Bezeichnungsfeld davor nicht ausgefüllt war. Zudem kann der Benutzer im Kopfbereich zur Quickadd-Funktion (siehe Kapitel 4.3.7) navigieren.

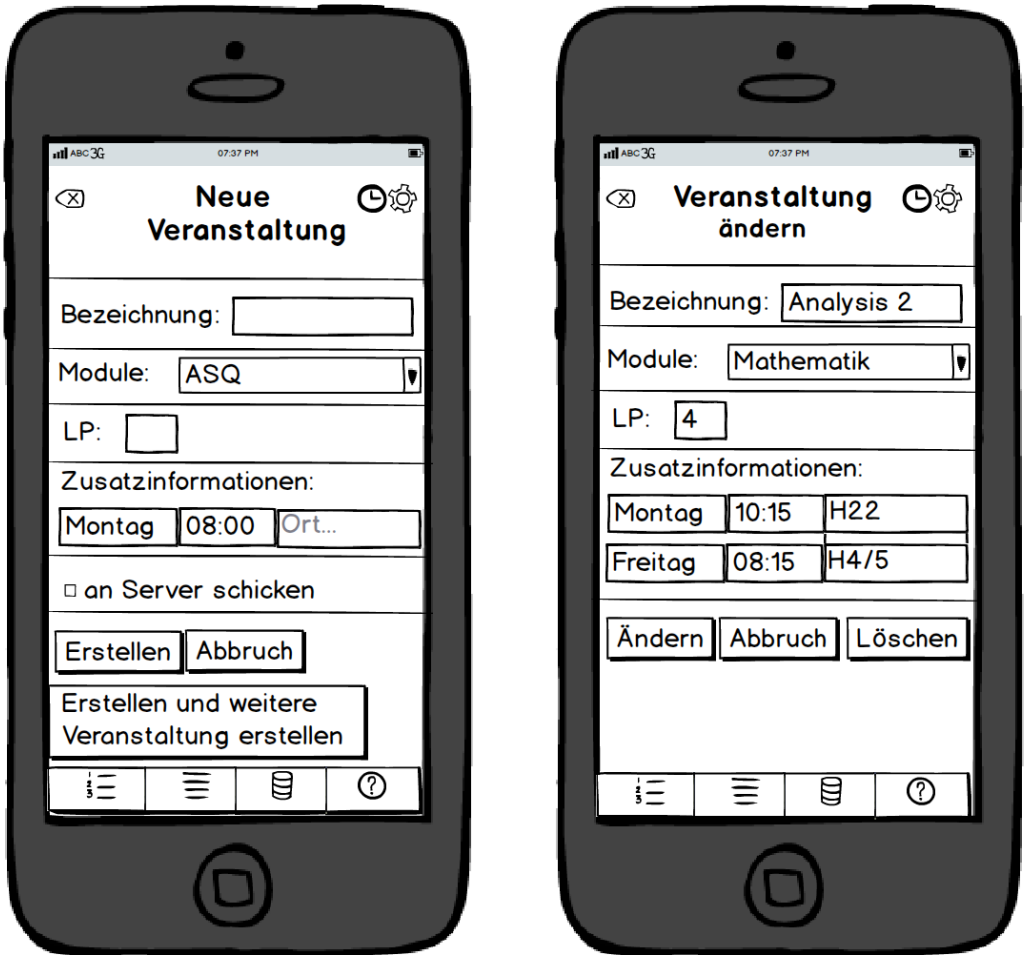


Abbildung 4.14: Mockups für die Erstellung und Änderung von Veranstaltungen

4.3.7 Quickadd

In Abbildung 4.15 ist das Mockup der Quickadd-Funktion dargestellt.

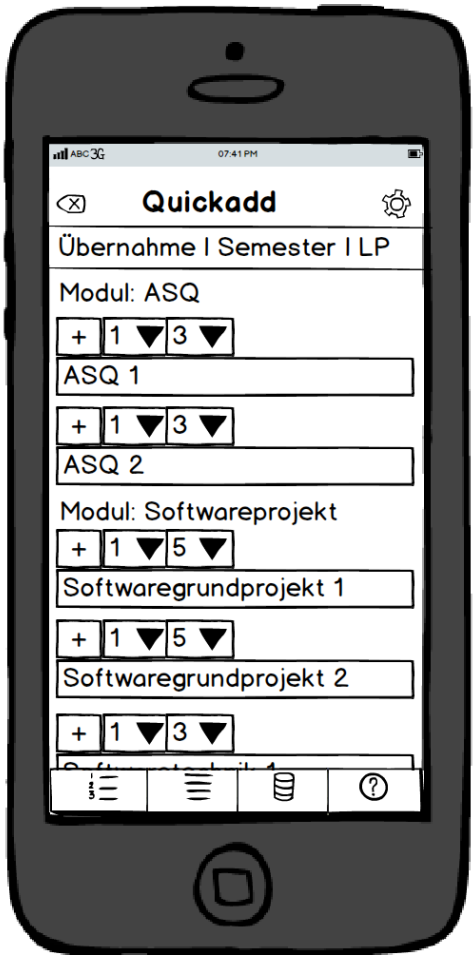


Abbildung 4.15: Mockup für die Quickadd-Funktion

Der Benutzer kann hier mehrere Veranstaltungen auswählen, die er übernehmen möchte. Zu jeder Veranstaltung kann er das Semester angeben, die Leistungspunkte auswählen und auch den Veranstaltungsnamen eintragen. Um die Erstellung zu beschleunigen, sind zu jeder Veranstaltung die Leistungspunkte automatisch an den zu erwartendem Aufwand eingestellt und die Veranstaltungsbezeichnung ebenfalls eingetragen. Es müssen nur noch die bereits besuchten Veranstaltungen und das Semester ausgewählt werden, wodurch sich der Zeitaufwand für die Einrichtung der App vor allem für Studenten aus höheren Semestern verkürzt.

4.3.8 FAQ

In Abbildung 4.16 ist das Mockup für die FAQs zu sehen. Hier existiert die gleiche Suchfilterfunktion wie in Kapitel 4.3.5 erläutert, wobei der Benutzer noch die Möglichkeit hat, die abgebildeten Stichworte zu berühren, woraufhin nur FAQs mit dem berührtem Stichwort angezeigt werden. Durch das Aufzeigen der Stichworte in den Antworten der Fragen wird dem Benutzer die Suche nach ähnlichen Fragen erleichtert.

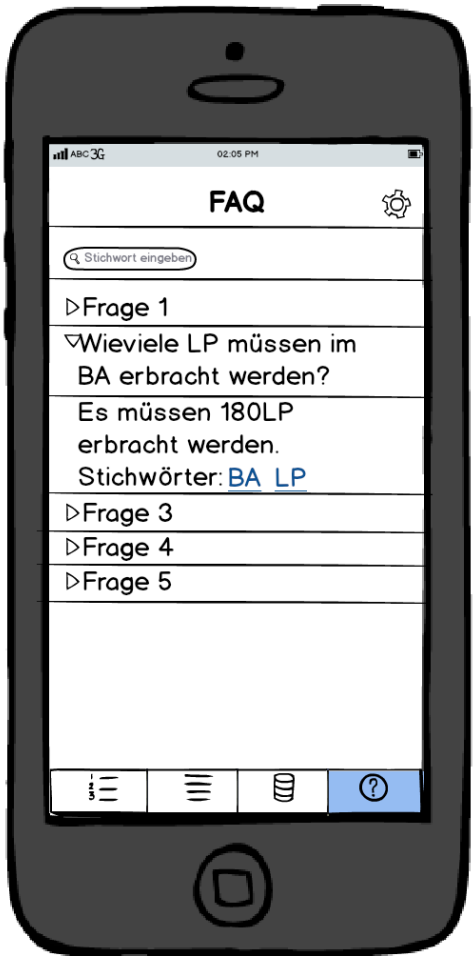


Abbildung 4.16: Mockup für die FAQ-Anzeige

4.3.9 Optionen

In Abbildung 4.17 sind die Mockups für die Optionen von ULMUS abgebildet. Erwähnenswert hierbei ist, dass bei einem Profilwechsel zu einer neuen Seite navigiert wird, in dem der Benutzer sein Profil wechseln kann. Diese ist mit der Startseite aus Kapitel 4.3.1 beinahe identisch. Im Vergleich zur Startseite fehlen lediglich die Angaben der Semesteranzahl und der Urlaubssemester, die bei einem fließenden Studiengangwechsel nicht relevant erscheinen. Zudem können nach einem Wechsel neue Semester und Urlaubssemester schnell erstellt werden.

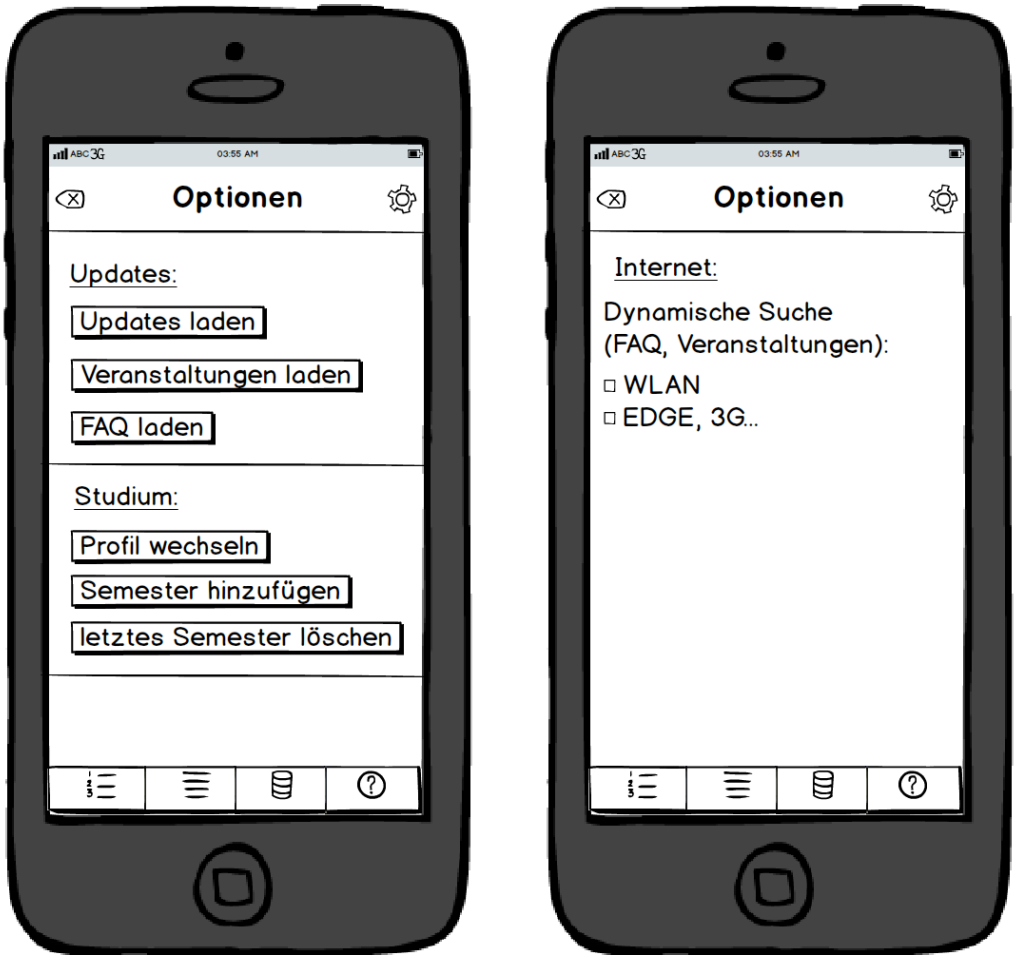


Abbildung 4.17: Mockups für die Optionen

4.3.10 Bestätigungs- und Fehlermeldungen

Das Mockup in Abbildung 4.18 zeigt eine beispielhafte Anzeige einer Meldung, die der Benutzer nach der erfolgreichen Erstellung einer Veranstaltung erhält. Durch die Angabe des Resultates erhält der Benutzer einen Eindruck davon, ob die durchgeführte Aktion erfolgreich war.

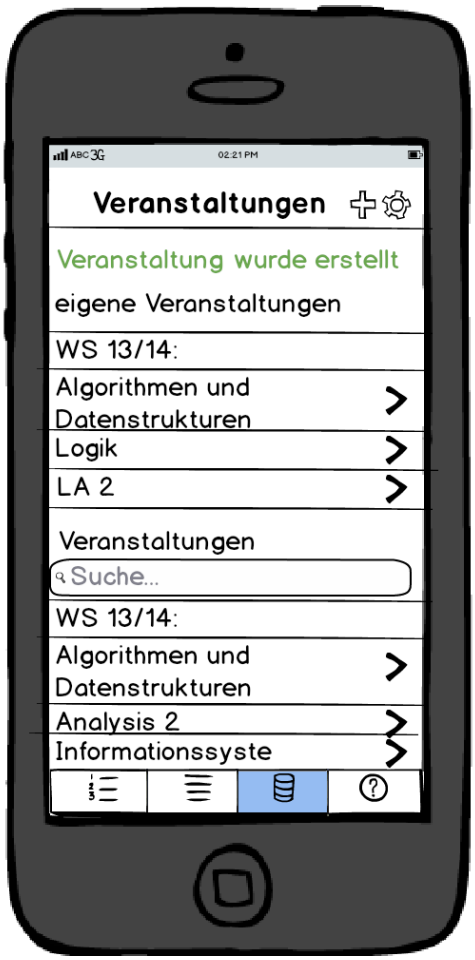


Abbildung 4.18: Mockup für eine Bestätigungsmeldung nach einer erfolgreichen Erstellung einer Veranstaltung

4.3.11 System Status

In Abbildung 4.19 stellt das Mockup das Verhalten des Systems bei einer Aktion dar, die zeitlich länger dauert. Dadurch steigt das Vertrauen des Benutzers beim Umgang mit der App, da er während einer zeitlich längeren Aktion immer noch Informationen über den aktuellen Zustand der App erhält.

Abbildung 4.19: Beispielhaftes Mockup für einen Systemstatus

5

Implementierung

In diesem Kapitel wird die Umsetzung der App anhand der erhobenen Anforderungen (siehe Kapitel 3) und unter Berücksichtigung des Entwurfs (siehe Kapitel 4) erläutert. Zunächst wird in Kapitel 5.1 beschrieben, welche Möglichkeiten existieren, um ULMUS zu entwickeln. Basierend auf der ausgewählten Möglichkeit, werden daraufhin in den Kapiteln 5.2 und 5.3 weitere benötigte Technologien für die Client-Seite diskutiert. Kapitel 5.4 beschreibt die verwendete Technologie für die Umsetzung der serverseitigen Funktionen. In Kapitel 5.5 werden weitere Aspekte der Implementierung erklärt, die mit den Technologien der vorherigen Kapitel zu verwenden sind. Abschließend wird in Kapitel 5.6 die konkrete Umsetzung der App an ausgewählten Beispielen erläutert.

5.1 Arten von mobilen Anwendungen

In den folgenden Kapiteln wird kurz auf die verschiedenen Arten von mobilen Applikationen eingegangen. Ihre jeweiligen Vor- und Nachteile werden aufgezeigt und anschließend erläutert, welche Art in dieser Arbeit eingesetzt wird.

5.1.1 Native App

Eine native App wird speziell für ein bestimmtes Betriebssystem entwickelt. Für die Entwicklung kommen dabei jene Technologien zum Einsatz, die vom ausgewählten Betriebssystem unterstützt werden. Dies ist zum Beispiel bei Android die Programmiersprache Java [And14] oder bei iOS die Programmiersprache Objective-C [Obj12]. Native Apps können dabei dem standardisiertem Design des jeweiligen Betriebssystems, dem sogenannten „Look and Feel“, optimal angepasst werden. Dadurch, dass sie auf die native API direkt zugreifen können, sind sie sehr perfomant und es können zum Beispiel aufwendige 3D-Apps entwickelt werden. Zudem haben sie vollen Zugriff auf die verschiedenen Hardwareressourcen wie die Kamera oder Lautsprecher [Sta10]. Diese Art der App ist dabei für alle drei Möglichkeiten der verschiedenen Client-Server Architekturen (siehe Kapitel 4.1) geeignet.

5.1.2 Web-App

Eine Web-App basiert heute typischerweise auf HTML5, ist im Browser ausführbar und muss auch nicht auf dem Gerät installiert werden [Sta10]. Für die Entwicklung wird dabei HTML, CSS und JavaScript verwendet. Es müssen daher keine besonderen Kenntnisse über das Zielsystem bekannt sein. Das hat den großen Vorteil, dass Web-Apps im Vergleich zu nativen Apps nur einmal entwickelt werden müssen und auf verschiedenen Betriebssystemen lauffähig sind. Im Vergleich zu den nativen Apps sind sie jedoch nicht perfomant genug, um aufwendige Apps zu entwickeln. Da Web-Apps nur im Browser laufen, ist außerdem die Anpassung des Designs der App an das standardisierte Design des jeweiligen Betriebssystems schwierig umzusetzen. Ebenfalls

besitzen die Web-Apps einen limitierten Zugriff auf die Hardwareressourcen des Gerätes. Zum Beispiel kann unter iOS nicht auf die Kamera zugegriffen werden [Pre12]. Web-Apps sind typischerweise der Thin-Client Architektur aus Kapitel 4.1 zuzuordnen, da die meisten Funktionen nur mit einer bestehenden Internetverbindung funktionieren.

5.1.3 Hybride App

Eine hybride App kann zwischen einer nativen App und einer Web-App eingeordnet werden. Für die Entwicklung der App kommen wie in der Web-App HTML, CSS und JavaScript zum Einsatz. Dies macht die Entwicklung für verschiedene Betriebssysteme einfacher. Im Vergleich zur Web-App wird diese App aber auf dem Gerät installiert (siehe Kapitel 5.2). Ihr Look and Feel für ein bestimmtes Betriebssystem ist dabei immer noch schwierig umzusetzen, jedoch kommt man mit entsprechenden Plugins dem Design einer nativen App nahe. Dadurch, dass sie als eigenständige App auf dem Gerät installiert ist, kann wie mit einer nativen App auf die Hardwareressourcen zugegriffen werden. Sie ist wie die native App für alle drei Arten der Client-Server Architekturen geeignet.

Für die Umsetzung von ULMUS wird eine hybride App entwickelt. Sie besitzt die Vorteile der Web-App, einfacher auf verschiedene Betriebssysteme portierbar zu sein und gleichzeitig auch den Vorteil von nativen Apps, als eigenständige App auf dem Gerät installierbar zu sein. Für die Umsetzung wird das Framework Phonegap verwendet, dass in Kapitel 5.2 vorgestellt wird.

5.2 Phonegap

Dieses Kapitel behandelt *Phonegap*, das ein Open-Source Framework ist und die Entwicklung hybrider Apps für verschiedene Betriebssysteme ermöglicht; ohne dass nähere Kenntnisse über diese erforderlich sind [Pho12]. Eine Phonegap-App kann wie eine Web-App mit den Web-Technologien HTML, CSS und JavaScript entwickelt werden. Phonegap portiert den Webcode dabei in eine eigenständige native App. Die Architek-

tur einer Phonegap-App repräsentiert dabei die Client-Seite aus Kapitel 4.1 und ist in Abbildung 5.1 dargestellt.

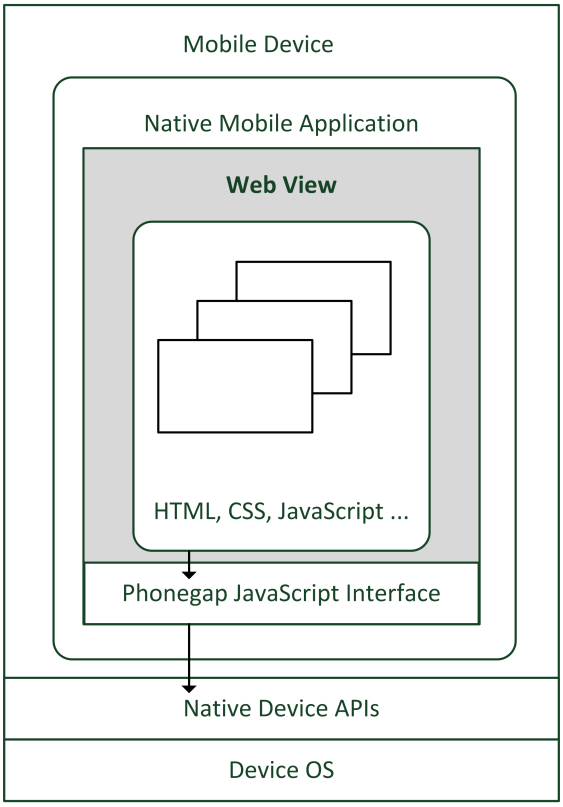


Abbildung 5.1: Phonegap Architektur nach [War12]

Die portierte Web-App läuft dabei in einer sogenannten *Web-View*. Eine Web-View funktioniert ähnlich wie ein Webbrowser und kann somit HTML-Seiten darstellen [Web14]. Für die Logik wird JavaScript verwendet, mit der zum Beispiel neue Seiten aufgebaut werden können oder auf Benutzerinteraktionen reagiert werden kann. Für Zugriffe auf Hardwarekomponenten wie es mit nativen Apps möglich ist, werden Plugins verwendet. Die Plugins bestehen aus zwei Komponenten, dem *JavaScript Interface* und der *plattformabhängigen nativen Implementierung* [Pho14a]. Aus der Web-App können die Funktionen des JavaScript Interfaces aufgerufen werden, woraufhin die Funktion je nach Betriebssystem das passende native Plugin aufruft. Phonegap selbst bietet bereits eine

Reihe von Plugins an, mit der unter anderem auf das Dateisystem zugegriffen werden kann [Pho14b].

5.3 jQuery Mobile

jQuery Mobile ist ein für mobile Geräte optimiertes Web Framework [Bro11]. Das Framwork ist unter anderem mit den mobilen Betriebssystemen Android und iOS und mit dem Framework Phonegap kompatibel. Mit ihm kann die Benutzeroberfläche, wie sie in Kapitel 4.3 in Form von Mockups dargetellt wurde, beschleunigt und standardisiert umgesetzt werden. Die Erstellung der Oberfläche wird vor allem mit der Unterstützung durch zahlreiche Widgets, die individuell einstellbar sind, erleichtert [Jmo14]. Die Benutzeroberfläche passt sich dabei automatisch der Bildschirmgröße des verwendeten mobilen Gerätes an. Unter anderem unterstützt es dabei Operationen auf dem DOM-Baum des grundlegenden HTML-Dokuments, wie zum Beispiel das Einbinden, Ändern oder Löschen von Elementen. Außerdem unterstützt es die Registrierung von Events und die Navigierung zwischen Seiten erfolgt mithilfe von AJAX. Durch seine Fähigkeiten ist jQuery Mobile daher ein wesentlicher Bestandteil der Controller-Komponente (siehe Kapitel 4.1.1) und zuständig für die Änderung und Erstellung der View sowie dem Einlesen von Daten aus der View.

5.4 Node.js

Für die Implementierung der serverseitigen Dienste wird das Framework *Node.js* verwendet. Das Framework Node.js basiert auf der JavaScript Engine von Chrome und mit ihr können Netzwerkanwendungen in JavaScript erstellt werden. In Abbildung 5.2 sind die verschiedenen Komponenten von Node.js dargestellt.

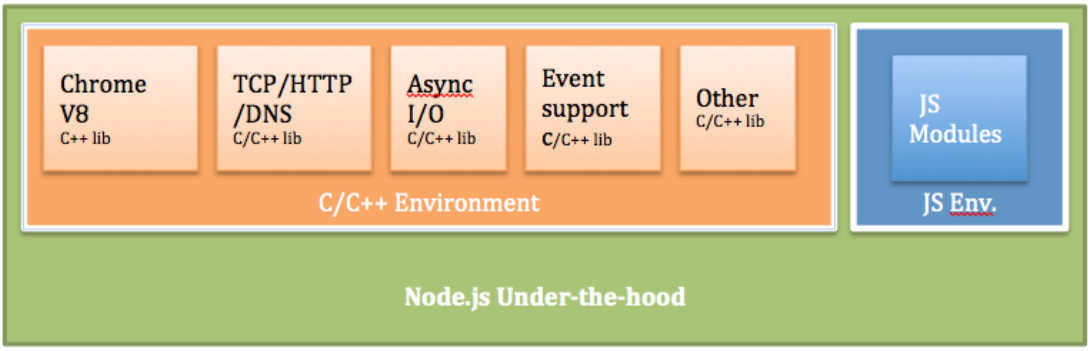


Abbildung 5.2: Node.js Architektur aus [Nod12].

Für die Ausführung von JavaScript wird die *Chrome V8 Engine* verwendet [v8e13a]. Sie basiert dabei auf C++ und übersetzt den auszuführenden JavaScript Code mittels Just-in-time Kompilierung in nativen Maschinencode, wodurch im Vergleich zur Übersetzung in Bytecode oder Verwendung eines Interpreters eine sehr hohe Ausführungsgeschwindigkeit erreicht wird [v8e13b].

Eine Besonderheit von Node.js ist das ereignisgesteuerte Prozessmodell, dass es ermöglicht, dass für die Codeausführung nur ein Thread notwendig ist. Dadurch können Codeabschnitte nicht parallel ausgeführt werden. Das Prozessmodell von Node.js ist dabei in Abbildung 5.3 dargestellt.

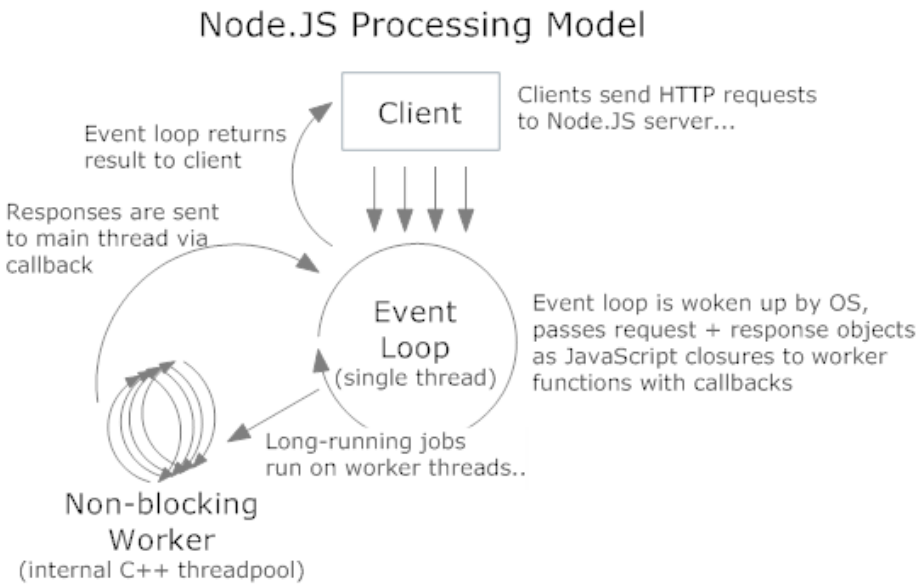


Abbildung 5.3: Prozessmodell von Node.js aus [Pro11].

In einer Ereignisschleife wird auf Ereignisse wie zum Beispiel Anfragen von Nutzern gewartet und bei einem Ereignis wird der entsprechende Code ausgeführt. Dabei werden länger andauernde Operation wie zum Beispiel I/O-Operationen asynchron bearbeitet. Damit die Codeausführung nicht bis zum Ergebnis der Operation blockiert wird, werden derartige Operationen an *Worker-Threads* weitergeleitet. Dies lässt sich am Beispiel vom Quelltext 5.1 verdeutlichen, in der eine Datenbankoperation bearbeitet wird.

```
1 db.all("Select bezeichnung from module", function(err, rows) {
2 //Diese Funktion wird nach der Datenbankoperation ausgeführt
3 });
```

Quelltext 5.1: Callback nach Bearbeitung der Datenbankanfrage

Hier wird ein Callback registriert, welches nach Bearbeitung der Operationen aufgerufen wird. Dies ist eine Methode, welche das Ergebnis der Operation beinhaltet. Nachdem ein Worker-Thread die Datenbankoperation bearbeitet hat, wird dieses Ereignis an die Ereignisschleife gesendet und daraufhin wird der Code des registrierten Callbacks ausgeführt.

Während der Bearbeitung der Operation können somit andere ankommende Ereignisse bearbeitet werden. Im Vergleich zu herkömmlichen Thread-Architekturen, die je Anfrage einen neuen Thread starten, werden bei Node.js durch die Verwendung von nur einem einzigem Thread für die Codeausführung und einigen weiteren Threads für länger andauernde Operationen weniger Kontextwechsel benötigt. Dadurch ist die CPU-Auslastung sowie der Speicherverbrauch geringer.
Mit dem Modulsystem von Node.js können verschiedene Funktionen einfach zum System hinzugefügt werden, wie es in den nicht-funktionalen Anforderungen aus Kapitel 3.4 gefordert wurde. Dabei werden einige Module wie zum Beispiel die Unterstützung für das HTTP-Protokoll standardmäßig mitgeliefert [Nod14].

Die Umsetzung des Servers basiert auf der beschriebenen Architektur aus Kapitel 4.1.2. In Kapitel 5.5.3 wird die verwendete Technik beschrieben.

5.5 Weitere Grundlagen zur Implementierung

In diesem Kapitel werden weitere Technologien vorgestellt, die in den Kapiteln 5.2, 5.3 und 5.4 vorgestellten Technologien zum Einsatz kommen.

5.5.1 Vergleich von Technologien zur Datenhaltung

Phonegap bietet für die Datenhaltung verschiedene Technologien an, nämlich *Web Storage* [Web13b], *IndexedDB* [Cur13] und *WebSQL DB* [Web13a]. Durch die Speicherung der Daten auf der Client-Seite ergibt sich der Vorteil, dass viele Funktionen ohne eine Internetverbindung funktionieren (siehe Kapitel 3.4). Eine kurze Übersicht der Technologien ist in Tabelle 5.1 abgebildet, auf die im Folgenden näher eingegangen wird.

Mit dem Web Storage können einfache *Schlüssel-Werte Paare* im Cache gespeichert werden. Dabei kann zwischen *Sessiondaten*, die nach dem Schließen der App nicht

Technologie	Web Storage	IndexedDB	WebSQL DB
Query	Key-Value	Key Ranges und Cursors	SQL
Transaktionen	-	expliziter Start, Rollback im Fehlerfall	expliziter Start, Rollback bei Fehler möglich
Vorteile	sehr simpel	Speicherung von Objekten	guter Lese- und Schreibzugriff
Nachteile	nur Strings	schwache Browserunterstützung	kein Standard laut W3C

Tabelle 5.1: Mögliche Arten der Datenspeicherung

mehr im Speicher existieren und dem *localStorage*, bei dem die Daten permanent gespeichert werden, unterschieden werden. Der Vorteil des Web Storage ist die einfache Speicherung und der Zugriff auf die Daten. Nachteilig hierbei ist jedoch, dass nur Strings als Werte gespeichert werden können und daher für komplexe Datentypen eine vorherige Serialisierung beziehungsweise anschließende Deserialisierung notwendig ist. Zudem sind komplexe Datenzugriffe durch die einfachere Datenspeicherung nicht möglich.
IndexedDB verwendet als Speicher einen Objektspeicher, in dem Javascript-Objekte mit einem eindeutigem Schlüssel gespeichert werden. Dabei können Datenbankzugriffe mithilfe des Keys oder mit *Indexes* (*Cursor Api* und *Key Range Api*) realisiert werden [Cur13]. Transaktionen werden explizit gestartet und bei einem Fehler wird standardmäßig ein Rollback durchgeführt. IndexedDB bietet in Verwendung mit Phonegap insbesondere den Vorteil, dass die Datenverwaltung durch die Verwendung von Javascript-Objekten erleichtert wird. Ein weiterer Vorteil ist, dass das W3C-Konsortium IndexedDB zurzeit als Standard spezifiziert und somit in Zukunft von den meisten Browsern unterstützt werden sollte [Cur13]. Jedoch ist zurzeit die Unterstützung von IndexedDB für den mobilen Bereich sehr eingeschränkt [ind14]. Erst ab Android 4.4 wird die Technologie im Android Browser unterstützt, welcher bis zur Android 4.3 Version der Standard Browser war. Zudem wird IndexedDB vom Safaribrowser in iOS nicht unterstützt. Somit ist eine volle Unterstützung für die beiden größten mobilen Betriebssysteme nicht gegeben (siehe Kapitel 3.4, Abbildung 3.24).
In WebSQL werden die Daten in einer relationalen Datenbank gespeichert. Queries

können mit SQL-Statements durchgeführt werden. Die Transaktionen werden explizit gestartet und im Falle eines Fehlers wird ein Rollback durchgeführt. Der Vorteil von WebSQL ist eine gute Datenspeicherung und ein einfacher Datenzugriff auf der Client-Seite. Außerdem ist die Technik bereits in zahlreichen Browsern von mobilen Betriebssystemen implementiert, wie zum Beispiel in Safari für iOS und dem Android Browser beziehungsweise Chrome für Android. Nachteilig ist jedoch, dass das W3C-Konsortium die Arbeit an WebSQL als Standard eingestellt hat und es somit möglich ist, dass in zukünftigen Browserversionen diese Technologie nicht mehr unterstützt wird.

Für die Datenhaltung wird in der Implementierung auf der Client-Seite primär WebSQL verwendet. Damit lässt sich der Datenbankentwurf sehr einfach umsetzen. Außerdem wird diese Technologie von den größten mobilen Betriebssystemen Android und iOS unterstützt.

5.5.2 JavaScript Object Notation

JavaScript Object Notation (JSON) ist ein leichtgewichtiges Datenformat für die Übermittlung von Daten, welches für Menschen und Maschinen leicht lesbar ist [CAA+13]. JSON wird für die Übertragung von Anwendungsdaten zwischen Client- und Server-Seite verwendet. Dabei kann eine JSON-Datei aus mehreren Schlüssel-Werte Paaren bestehen. Der Schlüssel besteht dabei aus einem String, mit dem man auf den verknüpften Wert zugreifen kann. Werte können dabei aus Strings, Zahlen, Booleschen Werten, Arrays oder JSON-Dateien bestehen. Im Quelltext 5.2 ist eine JSON-Datei dargestellt, die in dieser App verwendet wird und Daten einer Veranstaltung enthält.

```
1 {
2    "bezeichnung":"Programmierung von Systemen",
3    "modulid":"3",
4    "veranstaltungsrefid":"4",
5    "pflicht":"2",
6    "lp":"8",
7    "zeiten":[
8       {
```

```
9             "day":"Freitag",
10            "hour":"10",
11            "minute":"45",
12            "ort":"H22"
13        },
14        {
15            "day":"Mittwoch",
16            "hour":"10",
17            "minute":"30",
18            "ort":"H3"
19        }
20    ]
21 }
```

Quelltext 5.2: JSON-Datei für eine Veranstaltung

5.5.3 Remote Procedure Call

Für die Delegierung der Aufgaben auf der Seite des Servers (siehe Kapitel 4.1.2) wird die Technik *Remote Procedure Call* (RPC) verwendet. Bei RPC handelt es sich um eine Technik für die Realisierung von entfernten Methodenaufrufen. Dabei sendet der Client eine Anfrage an den Server und übergibt die zu verwendete Methode und die entsprechenden Parameter. Der Server verarbeitet die Anfrage und sendet an den Client eine Antwort mit dem Ergebnis [Mar99].

In der Implementierung von ULMUS wird für die Versendung der Anfragen HTTP verwendet. Dabei wird mit der angegeben URL im Pfad definiert, welche Methode auf dem Server aufgerufen werden soll. In Abbildung 5.4 ist ein möglicher Methodenaufruf dargestellt.

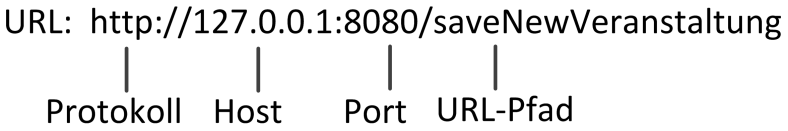


Abbildung 5.4: HTTP-Anfrage mit angegebener Methode im URL-Pfad

Die Parameter können dabei auf zwei verschiedene Arten verschickt werden. Mit der GET-Methode werden die Parameter nach dem URL-Pfad an die URL angehängt. Die GET-Methode wird dabei verwendet, um Daten vom Server anzufordern und keine Änderungen bisheriger Daten durchzuführen. Bei der POST-Methode werden die Parameter dagegen im Rumpf der HTTP-Nachricht angehängt. Parameter, die mit der POST-Methode verschickt werden, erstellen neue Daten auf dem Server.

5.5.4 Asynchronous JavaScript and XML

Für die Übertragung der Daten zwischen Client und Server wird *Asynchronous JavaScript and XML* (AJAX) verwendet. Mit AJAX ist es möglich, Daten asynchron zu übertragen. Dies hat den Vorteil, dass nach dem Absenden einer Anfrage an den Server die App nicht auf die Antwort vom Server wartet, sondern weiteren Programmcode ausführen kann (siehe Kapitel 3.4). Sobald die Antwort vom Server angekommen ist, wird wieder zum AJAX-Aufruf zurückgesprungen und daraufhin werden die Daten verarbeitet.

5.6 Ausgewählte Aspekte der Implementierung

In diesem Kapitel werden die vorgestellten Aspekte der Implementierungen aus Kapitel 5.5 anhand von Beispielen erläutert.

Da die Anforderungen an die FAQ-Funktion im Vergleich zu den anderen Anforderungen einfach umzusetzen sind, werden die verschiedenen Aspekte der Implementierung für eine gute Verständlichkeit anhand ihrer Funktionen erklärt. Die Umsetzungen der anderen Funktionen verläuft dabei in großen Teilen ähnlich, sodass aus den nachfolgenden Beispielen ersichtlich wird, wie ULMUS funktioniert.
Die folgenden Kapitel bauen dabei aufeinander auf. Der Ablauf ist in Abbildung 5.5 visuell dargestellt.

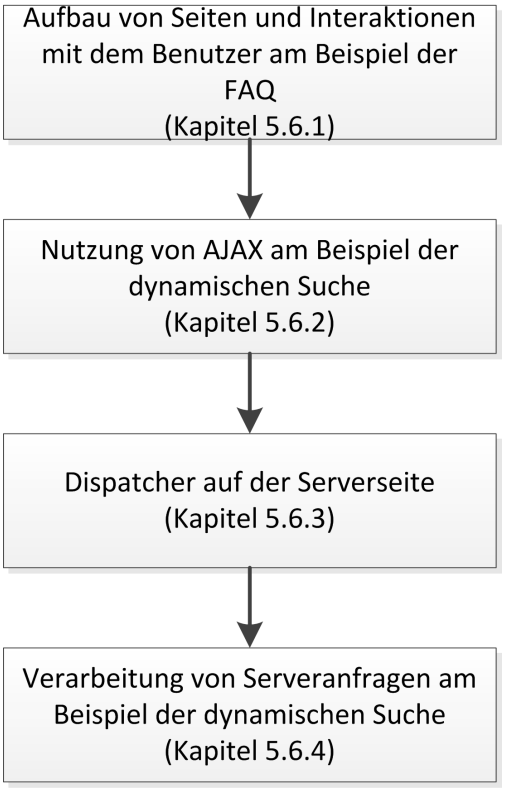


Abbildung 5.5: Logischer Aufbau der nachfolgenden Kapitel 5.6.1-5.6.4

Zunächst wird in Kapitel 5.6.1 beschrieben, wie die FAQ-Seite aufgebaut ist. Zusätzlich wird erläutert, wie auf mögliche Benutzerinteraktionen reagiert werden kann. Daraufhin wird in Kapitel 5.6.2 am Beispiel der dynamischen Suche die Kommunikation mit dem Server mithilfe von AJAX beschrieben. In Kapitel 5.6.3 wird die Umsetzung des Dispatchers auf der Server-Seite beschrieben, der auf Basis des RPC-Prinzips (siehe Kapitel 5.5.3) für die Verteilung der Anfragen auf die zuständigen Methoden zuständig ist. Abschließend wird die Verarbeitung einer Anfrage, in diesem Fall die Verarbeitung einer dynamischen Suche (siehe Kapitel 5.6.2), auf der Server-Seite erläutert.

5.6.1 Aufbau von Seiten und Interaktion mit dem Benutzer am Beispiel der FAQ

In der View-Komponente von ULMUS werden HTML-Seiten als Basis für Ansichten genutzt (siehe Kapitel 4.1.1). Die HTML-Seiten enthalten dabei den groben Aufbau der jeweiligen Seite, die anhand von JavaScript-Programmcode im Controller und des Cores gefüllt werden. Der Quelltext 5.3 zeigt hierbei die wesentlichen Bestandteile der FAQ-Seite in HTML.

```
1 <div data-role="collapsible-set" id='faqcollapse'>
2   <div data-role="listview" id="faqlist" data-inset="true"
3   data-filter="true" data-filter-placeholder="Stichwort eingeben... " >
4   </div>
5 </div>
```

Quelltext 5.3: Hauptbestandteil der FAQ-View

Die Seite besteht anfangs lediglich aus einem sogenannten *Collapsible-Set*, das eine sogenannte *Listview* enthält. Diese Container[1] enthalten die später eingefügten Fragen und Antworten.
Beim Laden der FAQ-Seite wird die Methode *init_page_faq()* im Controller angestoßen, mit der die HTML-Seite mit dynamischen Inhalten befüllt wird und Ereignisse für die Filtersuche registriert werden. Diese ist im Quelltext 5.4 dargestellt.

```
1 function init_page_faq() {
2   //Einbetten der FAQ's in die View
3   getAllFAQ();
4   //Eventhandler für dynamische Suche registrieren
5   $("#faqlist").on("filterablebeforefilter", faqEvent());
6 }
```

Quelltext 5.4: FAQ-Seite initialisieren

Anhand der Methode *getAllFAQ()* im Core werden die benötigten Daten aus der Datenbank geholt. Wie im Quelltext 5.5 dargestellt, führt sie dabei eine SQL-Query innerhalb

[1] zu dt. „Behälter“

einer Transaktion aus. Nach einer erfolgreichen Transaktion wird mit einem Callback die Methode *instertFAQtoView()* aufgerufen.

```
1  function getAllFAQ(){
2      db = window.openDatabase("Database", "1.0", "PhoneGap DB", 200000);
3      db.transaction(function (tx) {
4          var text = "Select frage, antwort, stichworte from faq";
5          tx.executeSql(text, [], insertFAQtoView);
6
7      }, transaction_error);
8  }
9
10 function insertFAQtoView (tx, results){
11   var len = results.rows.length;
12   var html = "";
13
14   // Erstellen des HTML-Codes für die die Fragen und Antworten
15   for (var i = 0; i < len; i++) {
16     var faq = results.rows.item(i);
17     var stichworte = faq.stichworte.split(" ");
18     var html_stichworte = "";
19     for (var j = 0; j < stichworte.length; j++) {
20     html_stichworte += '<a href=# onclick=changeFilterFaq("' + stichworte
           [j] + '")>' + stichworte[j] + '</a> ';
21     }
22     html +=
23     '<div data-filtertext="' + faq.stichworte + '" data-role="collapsible
           ">' +
24     '<h1><div style="white-space : normal;">' + faq.frage + '</div></h1>'
            +
25     '<div>' + faq.antwort + '</div> <div> Stichwörter: ' +
           html_stichworte + '</div> </div>';
26   }
27
28   // Einbetten in die View
29   $("#faqlist").append(html);
30   $("#faqlist").listview("refresh");
31   $("#faqlist").trigger("updatelayout");
32   $("#faqlist").trigger("create");
```

```
33      faqContent = html;
34  }
35
36  function changeFilterFaq(stichwort){
37      $('input[data-type="search"]').val(stichwort);
38      $('input[data-type="search"]').trigger("change");
39  }
```

Quelltext 5.5: FAQ-Einträge aus der DB holen und Seite befüllen

Anhand der Informationen aus der Datenbank wird zuerst der benötigte HTML-Code erstellt und daraufhin mit jQuery Mobile dieser in die HTML-Seite eingebettet.
Innerhalb der Antworten wird im Quelltext 5.5 Zeile 20 für jedes Stichwort noch ein sogenanntes *onclick-Event* hinzugefügt. Falls der Benutzer ein Stichwort berührt, wird dabei die registrierte Methode *changeFilterFaq(stichwort)* aufgerufen. Diese Methode bekommt das berührte Stichwort als Parameter übergeben und fügt es in den Suchfilter ein. Damit werden die Fragen automatisch nach den vorgegebenem Stichwort gefiltert.

Nach dem Aufruf der Methode *getAllFAQ()* wird ein *Eventhandler* für die Filtersuche der FAQ registriert, der für die dynamische Suche zuständig ist. Dieser wird genau dann aufgerufen, wenn die Eingabe im Suchfilter sich ändert. Der Zusammenhang mit AJAX wird in Kapitel 5.6.2 näher erläutert.

Die vollständige Seite der FAQ ist in Abbildung 5.6 abgebildet. Die Oberfläche wurde nach der Vorlage des Mockups aus Kapitel 4.3.8 erstellt.

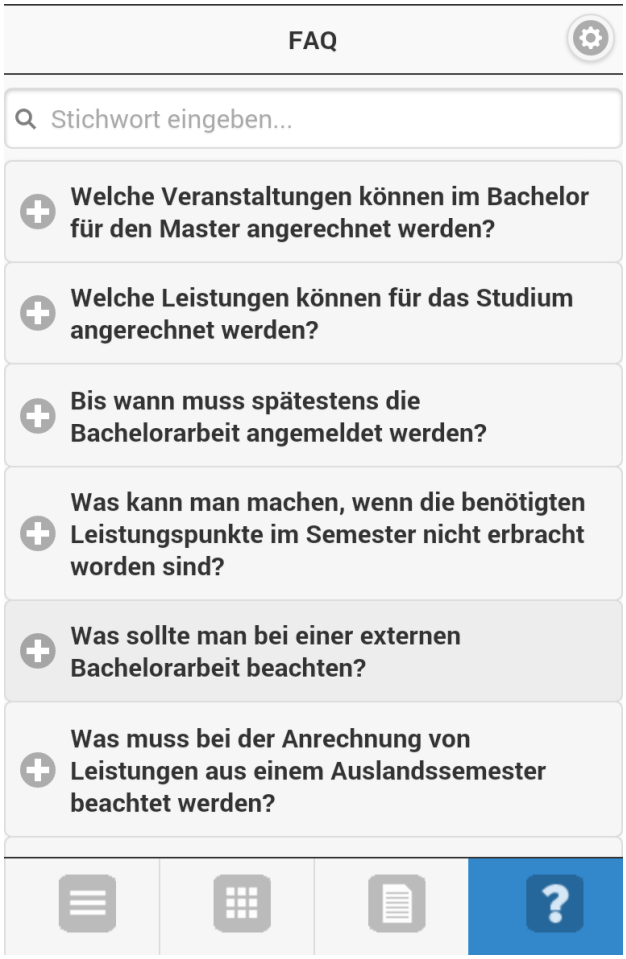


Abbildung 5.6: Darstellung der FAQ-Seite in der ULMUS-App

5.6.2 Nutzung von AJAX am Beispiel der dynamischen Suche

In diesem Kapitel wird die Nutzung von AJAX am Beispiel der dynamischen Suche von FAQ-Einträgen erklärt. Im vorherigen Kapitel 5.6.1 wurde ein Ereignis registriert, das eine Methode aufruft, sobald der Inhalt im Suchfilter verändert wird. Der Quelltext 5.6 stellt dabei den essentiellen Code für die Umsetzung dar.

```
1 // Event, dass bei Eingabe von Zeichen aufgerufen wird
2 function faqEvent(e, data){
3   /*
4   Code für die Herstellung der eingebettenen FAQ's aus der DB
```

```
5     ...
6     */
7     var $input = $(data.input),
8     db = window.openDatabase("Database", "1.0", "PhoneGap DB", 200000);
9     db.transaction(function (tx) {
10      var text = "select faqid from faq where stichworte like '%" +
            $input.val() + "%'";
11      tx.executeSql(text, [], function (tx, results) {
12        var faqids = new Array();
13        // Array befüllen mit den erhaltenen faqids aus der Datenbank
14        dynamicAjaxFAQ(faqids);
15      });
16    }, transaction_error);
17  }
```

Quelltext 5.6: FAQ-Einträge bei einem Event aus der DB holen

Zunächst wird dabei die HTML-Seite auf den Zustand nach der Initialisierung zurückgesetzt, um Duplikate bei wiederholtem Aufruf der Funktion in der Anzeige zu vermeiden. Im nächsten Schritt wird eine Datenbanktransaktion gestartet, die alle FAQ-Einträge aus der Datenbank lädt, in der die eingegebenen Daten vom Benutzer mit den Stichwörtern übereinstimmen. Diese werden mit einem AJAX-Aufruf an den Server gesendet (siehe Quelltext 5.8).

```
1   function dynamicAjaxFAQ(faqids){
2     $.ajax({
3       url: "http://10.0.2.2:8080/getFAQdynamic",
4       type: "GET",
5       timeout: 10000,
6       dataType: "json",
7       crossDomain: true,
8       error: function (xhr, textStatus, errorThrown) {
9         xhr.abort;
10        console.log(textStatus);
11        console.log(errorThrown);
12      },
13      data: {
14        input_data: $input.val(),
```

```
15        faqid_data: faqids,
16        version_data: version
17      }
18
19    }) // end ajax go .then
20    .then(function (response, textStatus) {
21      var html = "";
22      var obj = response;
23      var obj_data = response;
24      /*
25      Code für die Erstellung von HTML Inhalten aus den erhaltenen FAQ'S
26      */
27    });
28  }
```

Quelltext 5.7: AJAX-Aufruf im Core

Am Anfang werden die Einstellungen für die Kommunikation zum Server im AJAX-Aufruf festgelegt (siehe ab Zeile 3). Der URL-Pfad enthält die aufzurufende Methode. Da mit den zu übermittelnden Daten keine neuen Ressourcen auf dem Server erstellt werden, wird die Anfrage per HTTP mit der GET-Methode versendet. Mit dem Datentyp wird festgelegt, von welchem Typen die zurückgelieferten Daten vom Server sein müssen. Das Feld *data* enthält die zu übermittelnden Daten an den Server. Nach der Übermittlung der Daten an den Server können zwei Ereignisse auftreten. Falls eine Anfrage fehlschlägt, wird die im Feld *error* definierte Methode aufgerufen, die für das Fehlerhandling zuständig ist. Ist die Anfrage jedoch erfolgreich, wird die Methode im registrierten Ereignis *.then* aufgerufen. Diese Methode enthält die vom Server übergebenen Parameter und fügt sie zu den bereits bestehenden Daten in die HTML-Seite ein. In Kapitel 5.6.3 wird die Zuweisung der Anfrage an die zuständige Methode auf der Server-Seite erklärt.

5.6.3 Dispatcher auf der Server-Seite

In diesem Kapitel wird die Zuweisung der Anfragen an die korrekten Methoden für die konkrete Bearbeitung der Anfrage behandelt. Die wesentlichen Elemente sind dabei im Quelltext 5.8 dargestellt.

```
1  //Dispatcher
2  var Router = require('routes');
3  var dispatcher = Router();
4  // Event Registrierung für Anfragen
5  http.createServer(onRequest).listen(8080);
6
7  //aufgerufene Funktion bei Anfragen
8  function onRequest(request, response) {
9      if (request.method === "GET") {
10         dispatcher.addRoute("/getFAQdynamic:query", getFAQdynamic);
11     /*
12     weitere Routen
13     */
14
15         var route = dispatcher.match(request.url);
16         //  prüfen, ob Matching auf vorhandene Router erfolgreich
17         if (typeof (route) === 'undefined') {
18             console.log("wrong request");
19             pageNotFound(response);
20         } else {
21         /*
22         Prüfen der Version, Error bei veralteter Version
23         */
24             var queryObject = url.parse(request.url, true).query;
25             route.fn.apply([response, queryObject, route.splats], error);
26         }
27
28     } else if (request.method === "POST") {
29         // Verarbeitung von POST-Anfragen
30     }
31 }
```

Quelltext 5.8: Verteilung der Anfragen an die zuständigen Methoden

Initial wird für ankommenede HTTP-Anfragen die Methode *onRequest* registriert. Sie wird für jede Anfrage aufgerufen und bekommt die JSON-Objekte *request* und *response* übergeben. Das Objekt *request* enthält unter anderem Informationen zur URL, die übergebenen Parameter und um welche Art von Anfrage es sich handelt. Mit dem

Objekt *response* werden die Daten an den Client zurückgesendet. Da die Architektur des Servers auf dem RPC-Prinzip basiert (siehe Kapitel 5.5.3), werden Anfragen an die zuständigen Methoden weitergeleitet. Hierbei wird im ersten Schritt überprüft, um welche Art von Anfrage es sich handelt. Für das Beispiel der dynamischen Suche aus Kapitel 5.6.2 wird eine GET-Anfrage verwendet, da nur Informationen vom Server angefragt werden. Die weitere Zuweisung ergibt sich aus dem Pfad der URL. Dabei wird das Node.js Modul *routes* verwendet, das für verschiedene Pfade eine Methode registriert [Rou13]. Stimmt der Pfad bei einer Anfrage mit einer registrierten Route überein, werden die GET-Parameter aus der URL extrahiert und zusammen mit dem JSON-Objekt *response* an die zuständige Methode übergeben, mit der die Anfrage dann bearbeitet wird. Im Falle eines fehlgeschlagenen Matchings wird die Methode *pageNotFound* aufgerufen, die dem Client eine Fehlernachricht sendet. Der weitere Verlauf der Verarbeitung einer Anfrage wird in Kapitel 5.6.4 am Beispiel der dynamischen Suche beschrieben.

5.6.4 Verarbeitung von Anfragen am Beispiel der dynamischen Suche

Im vorherigen Kapitel 5.6.3 wurden Anfragen an die zuständigen Methoden weitergeleitet. In diesem Kapitel wird die Verarbeitung einer weitergeleiteten Anfrage am Beispiel der dynamischen Suche erklärt. Der Quelltext 5.9 enthält den dazu passenden Code.

```
1  var getFAQdynamic = function getFAQdynamic() {
2    //Respone-Objekt
3    var response = this[0];
4
5    //Übergebene Daten vom Client an den Server
6      var data = this[1];
7
8      var faqbezeichnung = data['input_data'];
9      var faqids = "";
10     if (data['faqid_data[]'] != undefined) {}
11     faqids = data['faqid_data[]'].toString();
12
13     var faqArray = new Array();
```

```
14      var query = "Select faqid, frage, antwort, stichworte from faq where
            faqid not in (" + faqids + ") and stichworte like '%" +
            faqbezeichnung + "%' ";
15
16      db.all(query, function (err, rows) {
17          var len = rows.length;
18          console.log(len);
19          for (var k = 0; k < len; k++) {
20              var faqObject = {};
21              faqObject.frage = rows[k].frage;
22              faqObject.antwort = rows[k].antwort;
23              faqObject.stichworte = rows[k].stichworte;
24              faqArray.push(faqObject);
25          }
26          answer = (JSON.stringify(faqArray));
27          response.writeHead(200, {
28              "Content-Type": "application/json; charset=UTF-8"
29          });
30          response.write(answer);
31          response.end();
32      });
33  }
```

Quelltext 5.9: Verarbeitung der Anfragen am Server

Aus den übergebenen Daten vom Client wird eine Datenbankanfrage (Quelltext 5.9 Zeile 16) gestartet, über die alle FAQ-Einträge mit ähnlichen Stichwörtern bezogen werden. Die Datenbankanfrage ruft die geschachtelte Methode mit einem Callback nach der Bearbeitung auf. In der geschachtelten Methode werden die geladenen FAQ-Einträge als JSON-Objekte in einem Array gespeichert. Dem Client (d.h. der ULMUS-App), wird daraufhin eine Antwort mit dem HTTP-Statuscode 200[2] gesendet, welche das Array mit den FAQ-Einträgen enthält. In Kapitel 5.6.2 wurde die Entgegennahme der Antwort auf der Client-Seite bereits behandelt.

[2]= erfolgreiche Bearbeitung

6

Evaluation

Um herauszufinden, ob die Anforderungen aus Kapitel 3.3 und Kapitel 3.4 sinnvoll umgesetzt wurden und ob die Funktionen von ULMUS auch benutzerfreundlich zu bedienen sind, ist eine Evaluation notwendig [RC08]. Aufgrund des begrenzten Zeitrahmens dieser Arbeit wurde daher für die Evaluation ein *Usability Test* durchgeführt.
Bei einem Usability Test testen und bewerten Benutzer aus der potenziellen Zielgruppe die ULMUS-App. Die Zielgruppe besteht in diesem Fall aus Studenten der Universität Ulm. Dabei wurde den Studenten zuerst ein sogenannter *Teasertext* vorgelegt, welcher eine kurze Zusammenfassung der angebotenen Funktionen enthält. Zusätzlich wurden noch einige Screenshots von ULMUS beigelegt. Der Teasertext wurde bewusst kurz gehalten, so dass er auch als Beschreibung in einem App Store vorkommen könnte und die Bedingungen der größten App Stores eingehalten werden (siehe beispielsweise die Beschreibung des Play Stores [Cha14]).
Für die Evaluation konnten insgesamt sechs Studenten befragt werden. Eine Befragung

von mehr Probanden wird eine geringe Erkenntnissteigerung bewirken [Nie00]. In Kapitel 6.1 werden die Ergebnisse der Evaluation vorgestellt und in Kapitel 6.2 werden darauf aufbauend Optimierungen an der App erläutert. Mögliche Erweiterungsaspekte von ULMUS, die insbesondere bei zukünftigen Betrachtungen eine Rolle spielen können, werden dagegen in Kapitel 7.2 abschließend vorgestellt.

6.1 Auswertung der Evaluation

Für den bisherigen Ablauf ihres Studienverlaufs haben Studenten aus den höheren Semestern (6. und 7. Semester) im Durchschnitt sieben Minuten gebraucht. Die benötigte Zeit wurde von allen Studenen als angemessen empfunden.

Obwohl die Quickadd-Funktion (siehe Kapitel 3.3.6) explizit im Teasertext auftaucht, wurde diese jedoch ohne Hinweis von allen Studenten während der Abbildung des Studienverlaufs zunächst übersehen. Während der Übernahme der Veranstaltungen in der Quickadd-Funktion hat sich zudem gezeigt, dass die Benutzer teilweise irritiert waren, sobald die Bezeichnung eines Moduls mit der Bezeichnung einer Veranstaltung, die zu dem Modul gehört, übereinstimmte.
Außerdem war laut den Probanden nicht immer ersichtlich, welches Feld für die Semesterauswahl und welches für die Leistungspunkte zuständig war. Diese Kritikpunkte zeigen, dass die Quickadd-Funktion beim erstmaligen Erstellen der Veranstaltungen deutlicher hervorgehoben werden muss und dass das Interface dazu übersichtlicher gestaltet werden muss. Ansonsten wurde die Quickadd-Funktion als sehr nützlich beurteilt, da alte Veranstaltungen dadurch sehr schnell übernommen werden konnten.

Die Eintragungsmöglichkeiten bei der Erstellung einer Veranstaltung ohne Quickadd-Funktion wurden als sehr sinnvoll empfunden. Es wurde empfohlen, dass bei Eingabe des Ortes für die Veranstaltung mögliche Orte aufgelistet werden, wie es bei der Eingabe von Veranstaltungen bereits möglich ist. Ebenfalls wurde von einigen Studenten vor-

geschlagen, optionale Zusatzinformationen zu den Veranstaltungen eintragen zu können.

Die FAQ-Seite wurde als sehr nützlich empfunden und auch die Möglichkeit, die Stichwörter für die Filterung zu verwenden. Von einem Student wurde der Vorschlag eingebracht, dass bei sehr vielen FAQ-Einträgen eine weitere Kategorisierung sinnvoll wäre.

Die Gesamtübersicht, die Semesterseite, die detaillierte Semesterübesicht und die Optionsanzeige wurden als übersichtlich empfunden. Insgesamt wurde die Usability und Geschwindigkeit der App als sehr gut bewertet worden.

In Kapitel 6.2 werden einige Optimierungen vorgestellt, die auf der Auswertung der Evaluation basieren.

6.2 Optimierungen der Quickadd-Funktion

Auf Basis der Ergebnisse der Evaluation aus Kapitel 6.1 werden in diesem Kapitel einige Optimierungen für die Verwendung der Quickadd-Funktion beschrieben. Die Änderungen bezüglich der Quickadd-Funktion sind wichtig, da diese Funktion ein wichtiger Bestandteil der App darstellt. Mit ihr wird die Benutzerfreundlichkeit für Studenten mit vielen besuchten Veranstaltungen deutlich erhöht (siehe Kapitel 3.4).
Beim erstmaligen Erstellen einer Veranstaltung wird deshalb mit einem Popup auf die Quickadd-Funktion hingewiesen, mit der auch auf diese weitergeleitet werden kann.
Das Design der Quickadd-Funktionen wurde entsprechend der Kritikpunkte aus Kapitel 6.1 überarbeitet und ist in Abbildung 6.1 rechts dargestellt.

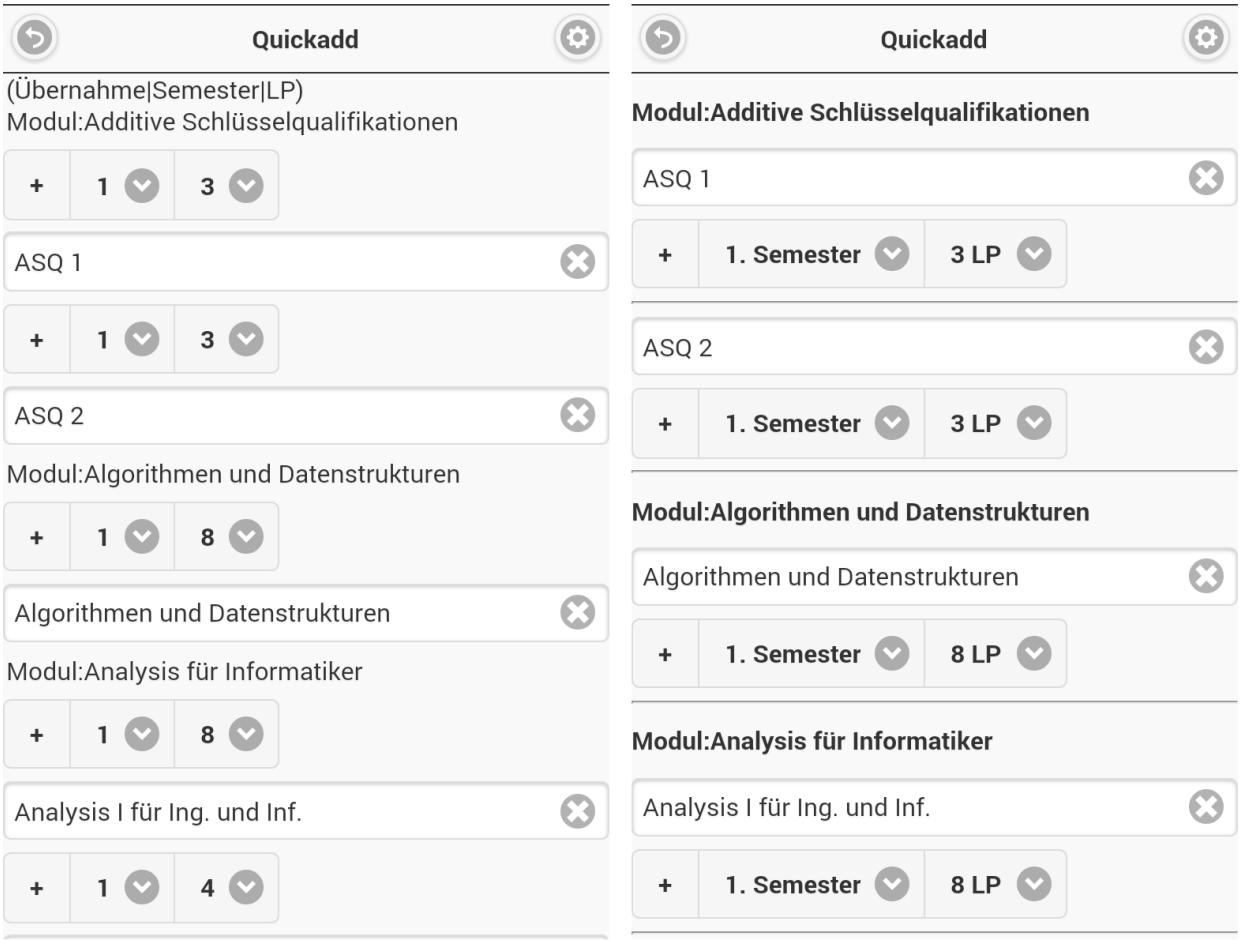


Abbildung 6.1: Links alte Anzeige und rechts neue Anzeige der Quickadd-Funktion

Auf der linken Seite ist die alte Anzeige der Quickadd-Funktion abgebildet. Im Vergleich zur alten Darstellung wurde die Modulzuordnung deutlicher dargestellt und die Veranstaltungen untereinander besser abgetrennt. Die Veranstaltungsbezeichnung steht nun an erster Stelle und die Felder für die Semesterauswahl und die Leistungspunkte sind besser erkennbar. Durch die Anzeige eines Popups wird dem Benutzer die Quickadd-Funktion deutlicher dargestellt und durch die Optimierung der Anzeige wird die Benutzerfreundlichkeit verbessert. Mögliche zukünftige Erweiterungen von ULMUS, die unter anderem auf den Ergebnissen der Evaluation basieren, werden in Kapitel 7.2 erläutert.

7 Fazit

In Kapitel 7.1 werden die Ergebnisse dieser Bachelorarbeit zusammengefasst und im anschließenden Kapitel 7.2 wird ein Ausblick gegeben.

7.1 Zusammenfassung

Im Rahmen dieser Bachelorarbeit wurde ein mobiler Studienführer für die Universität Ulm entwickelt. In Kapitel 2 wurden dazu die Grundlagen rund um das Thema Studium vorgestellt und passend dazu ausgewählte Anwendungsfälle präsentiert.

Kapitel 3 beschreibt den Ist-Stand der bereits vorhandenen Apps für verschiedene Universitäten und analysiert diese. Um die Anforderungen an die App genauer zu definieren, wurden daraufhin Interviews mit Studenten durchgeführt. Auf Basis des analysierten

Ist-Stands und den geführten Interviews wurden die funktionalen und nicht-funktionalen Anforderungen an ULMUS präsentiert. Für die funktionalen Anforderungen wurde unter anderem vorgegeben, dass ein individueller Studienverlauf abgebildet werden kann und und eine zentrale Informationsquelle vorhanden sein muss. Für die nicht-funktionalen Anforderungen wurden die Qualitätseigenschaften der App festgelegt.

Nachdem die Anforderungen festgelegt worden sind, konnte mit dem Entwurf der App in Kapitel 4 gestartet werden. Für die Client-Seite wurde festgelegt, dass ein hybrider Client entwickelt wird, wodurch Funktionen auf der Client-Seite sowie auf der Server-Seite vorhanden sein sollten. Das Datenmodell wurde auf Basis der Anforderungen aus Kapitel 3 erstellt und unterstützt unter anderem die Erstellung eines Studienverlaufs für verschiedene Studiengänge und Prüfungsordnungen. Abschließend wurden Mockups für die Benutzeroberfläche erstellt und die Vorzüge für eine optimale Bedienung der App an ihnen erläutert.

Nach der Erstellung des Entwurfs wurde in Kapitel 5 mit der Implementierung begonnen. Zunächst wurden dazu verschiedene Technologien vorgestellt und bei einigen abgewogen, welche sich für die Umsetzung der App besser eignet. Dabei hat sich herausgestellt, dass eine hybride App auf Basis von Phonegap entwickelt werden sollte. Für die Server-Seite wurde Node.js verwendet. Anhand von ausgewählten Beispielen wurde die Umsetzung der Funktionen in der App und auf der Server-Seite erläutert.

In Kapitel 6 wurden die Ergebnisse einer Evaluation der fertigen App ausgewertet. Aus den Ergebnissen wurden Optimierungen an der Quickadd-Funktion vorgenommen. Zukünftige Erweiterungen wie zum Beispiel die Kategorisierung der FAQ-Einträge werden in Kapitel 7.2 vorgestellt.

7.2 Ausblick

Die Funktionalitäten von ULMUS können in Zukunft erweitert und verbessert werden. Zum Beispiel kann bei der Erstellung von Veranstaltungen ein weiteres Feld für Zusatzinformationen zur Verfügung gestellt werden sowie bei Eingabe des Ortes passende Orte vorgeschlagen werden (vgl. Kapitel 6.1).

Weiterhin können die FAQ-Einträge in verschiedene Kategorien eingeteilt werden. Dazu könnte die Datenbank um zwei Entitäten erweitert werden, konkret um die Entitäten *Kategorien* und *FAQKategorie* (siehe Abbildung 7.1). Die Tabelle FAQKategorie verknüpft dabei FAQ-Einträge mit den Kategorien, sodass FAQ-Einträge in mehreren Kategorien stehen können. Eine Umsetzung wäre von Vorteil, falls die Anzahl der FAQ-Einträge auf einer Seite unüberschaubar wird.

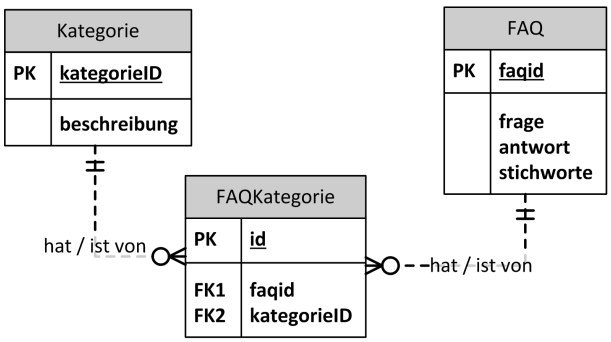


Abbildung 7.1: Erweiterung der FAQ-Einträge um Kategorien

Das derzeitige Design der PDF kann in Zukunft noch verbessert werden. Das liegt im momentanen Zustand daran, dass das verwendete Modul für die Erstellung der PDF zurzeit nur grundlegende Funktionen mit sich bringt und zum Beispiel keine Tabellen unterstützt werden.

Eine Exportfunktion des bisherigen Studienverlaufs auf ein anderes Smartphone wäre zudem wünschenswert. Hier muss zunächst ein passendes Objektmodell erstellt wer-

den, um alle relevanten Daten zu erfassen. Dazu zählen neben den bisher gehörten Veranstaltungen auch die Semester und Zusatzinformationen wie zum Beispiel, welches Anwendungsfach gewählt wurde. Ebenfalls muss darauf geachtet werden, dass die Versionen der Apps aktuell sind, damit die Daten konsistent bleiben. Diese können dann zum Beispiel als JSON-Objekt an einen Server übertragen werden und mit einem neuem Gerät heruntergeladen werden. Aus Datenschutzgründen sollten die Daten nicht ohne ein Kennwort zugänglich sein und nur für einen begrenzten Zeitraum zur Verfügung stehen.

Eine Notenübersicht könnte in Zukunft ebenfalls implementiert werden. Hier können zu Veranstaltungen Noten eingetragen und der Notenschnitt angezeigt werden. Diese sollten aufgrund des Datenschutzes verschlüsselt gespeichert werden, sodass zum Beispiel keine anderen Apps diese Informationen lesen können.

Abschließend können mit dem flexiblem Datenmodell neben den Medien-/Informatik Bachelor und Masterstudiengängen in Zukunft viele weitere Studiengänge unterstützt werden. Die Einpflegung weiterer Studiengänge kann dabei mit der integrierten Updatefunktion in der App durchgeführt werden, da hierbei nur Änderungen in der Datenbank notwendig sind.

A

Anhang

Evaluation zur App Mobiler Studienführer (Usability Testing)

Erst einmal möchte ich mich für deine Teilnahme an dieser Umfrage bedanken.

Das Ziel dieser Umfrage ist es, die angebotenen Funktionen der App und die Usability zu bewerten.

Aus den Ergebnissen soll herausgearbeitet werden, welche Änderungen an der App in Zukunft gemacht werden sollen und welche neue Funktionen eingebaut werden können, die den Studenten der Uni Ulm von Nutzen sein können.

Die Ergebnisse dieser Umfrage werden natürlich vertraulich behandelt und es können keine Rückschlüsse auf einzelne Personen gezogen werden.

Bitte ausfüllen:

Geschlecht: männlich ❑ weiblich ❑

Studiengang:

Prüfungsordnung:

Aktuelles Semester:

Aufgaben:

1. Mache dich mit der App vertraut, indem du den Teasertext durchliest und anschließend durch die App klickst, um einen kurzen Überblick zu erhalten.

Teasertext:

Mit dem Studienführer für die Uni Ulm kannst du bequem dein Studium planen und hast immer alles im Überblick.

HIGHLIGHTS

Semesterübersicht:

Hier kannst du sofort sehen, welche Veranstaltungen du wann und wo im ausgewählten Semester hast.Zudem kannst du detaillierte Informationen zum aktuellen Semester aufrufen.

Gesamtübersicht:

Hier kannst du detaillierte Informationen zum gesamten bisherigem Studiumsverlauf ansehen und hast die Möglichkeit, diese als PDF exportieren.

Übernahme von Veranstaltungen:

Es können Veranstaltungen von anderen Studenten übernommen werden, die Informationen zur Modulzuordnung, Zeiten, LP etc. schon eingetragen haben!

Quickadd-Funktion: Mit der Quickadd-Funktion (unter Veranstaltung erstellen) kannst du bequem mehrere Veranstaltungen gleichzeitig aus früheren Semestern in die App eintragen.

Studiengangwechsel:

Bei einem Studiengangwechsel werden deine bisherigen Veranstaltungen automatisch den neuen Modulen zugeordnet.

FAQ:

Hier kannst du Antworten auf oft gestellte Fragen zum Studium finden.

Dynamische Veranstaltungs- und FAQ-Suche:

Du kannst die Veranstaltungen und Fragen filtern und es werden dir passend zur Suche dynamisch noch nicht auf der App vorhandene Daten vom Server geladen.

2. Bilde deinen bisherigen Studiumsverlauf ab, indem du deine gehörten Veranstaltungen erstellst und sie den Semestern zuordnest.
   Die Zeit für die Erstellung wird gemessen, um herauszufinden, wie „Einstiegsfreundlich" die App vor allem für Studenten in höheren Semestern ist.

3. War die Abbildung des Studiumsverlaufs in angemessener Zeit durchführbar? Falls nicht, welche Information sollten weggelassen werden bzw. was könnte verbessert werden?

4. Wurde die Quickadd-Funktion ohne Hinweis verwendet? ja ❑ nein ❑

5. Wie nützlich war die Funktion und was könnte noch verbessert werden?

6. Erstelle diesmal eine Veranstaltung ohne Quickadd-Funktion.

7. Ändere eine von dir erstellte Veranstaltung oder übernehme eine Veranstaltung vom Server.

8. Wie sinnvoll fandest du die Eintragsmöglichkeiten und was könnte man ggf. noch ändern?

9. Wie übersichtlich ist für dich die FAQ-Seite und was könnte verbessert werden?

10. Wie übersichtlich ist für dich die Gesamtübersicht-Seite und was könnte verbessert werden?

11. Wie übersichtlich sind für dich die Semesterseite und die detaillierte Semesterseite und was könnte verbessert werden?

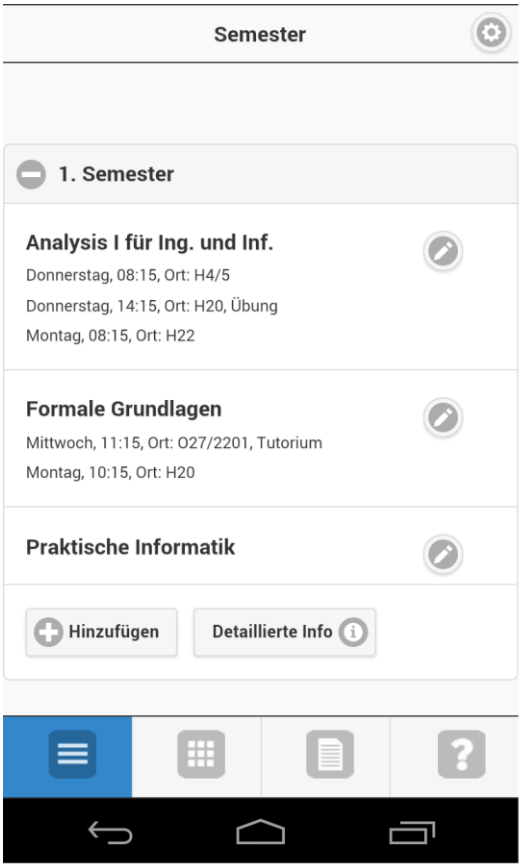
12. Sonstige Anregungen und Feedback (zu Optionen, Usability, Erweiterungen, Geschwindigkeit der App etc.)

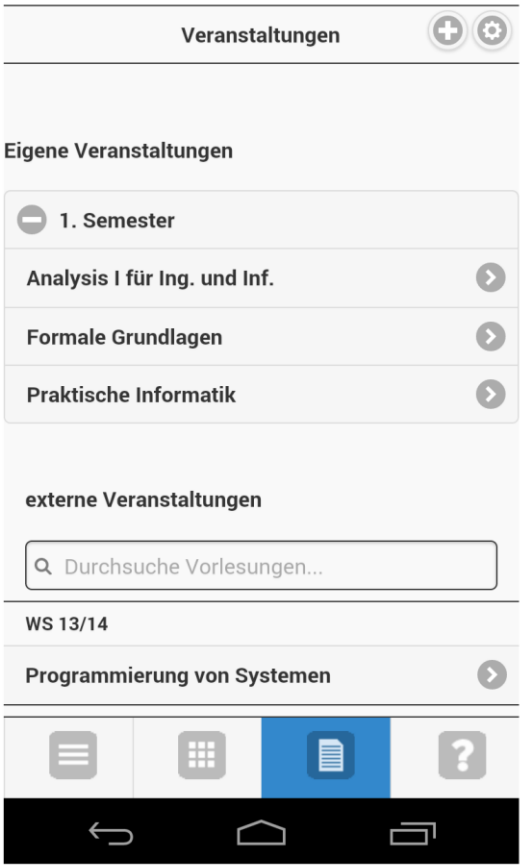
Optionen:

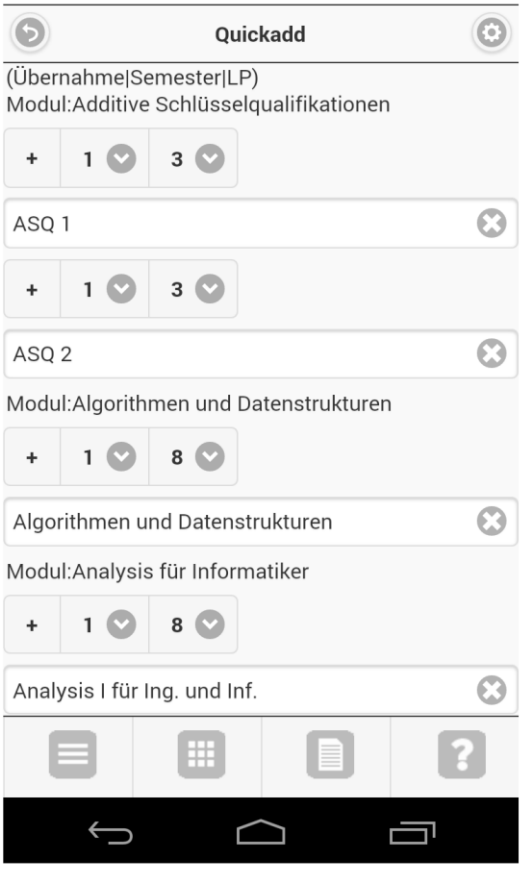
Usability:

Geschwindigkeit:

Sonstiges:







Screenshots zur App

Abbildungsverzeichnis

Quelltextverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[And14] *Application Fundamentals*. http://developer.android.com/guide/components/fundamentals.html, Version: 2014, Abruf: 28.02.2014

[Anz13] *Anzahl der Studierenden an Hochschulen in Deutschland vom Wintersemester 2002/2003 bis 2012/2013*. http://de.statista.com/statistik/daten/studie/221/umfrage/anzahl-der-studenten-an-deutschen-hochschulen/, Version: 2013, Abruf: 08.01.2014

[App14] *Eine Liste verschiedener Apps von und für Universitäten*. http://campusapps.wordpress.com/, Version: 2014, Abruf: 19.02.2014

[BaM10] *Ländergemeinsame Strukturvorgaben für die Akkreditierung von Bachelor und Masterstudiengängen*. http://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/2003/2003_10_10-Laendergemeinsame-Strukturvorgaben.pdf, Version: 2010, Abruf: 08.01.2014

[Beu14] *Beurlaubung, Rückmeldung, Studiengangwechsel und Exmatrikulation*. http://www.uni-ulm.de/studium/studienorganisation/beurlaubung-rueckmeldung-studiengangwechsel-und-exmatrikulation.html, Version: 2014, Abruf: 25.02.2014

[Bro11] BROULIK, Brad: *Pro jQuery Mobile*. New York, NY 10013, USA : Apress, 2011

[CAA+13] CREDLE, R. ; ARMSTRONG, A. ; ATKINSON, C. u. a.: *Implementing IBM CICS JSON Web Services for Mobile Applications*. IBM Hursley Park, Hursley, UK : IBM Redbooks, 2013

[Cha14] *Upload applications*. `https://support.google.com/googleplay/android-developer/answer/113469?hl=en`, Version: 2014, Abruf: 19.02.2014

[Con10] *Connected Worlds: Das Web gehört fest zum Alltag der Menschen*. `http://www.bitkom.org/de/presse/66442_62612.aspx`, Version: 2010, Abruf: 08.01.2014

[Cur13] *Indexed Database API*. `http://www.w3.org/TR/IndexedDB/`, Version: 2013, Abruf: 05.02.2014

[Hev07] HEVNER, Alan R.: A three-cycle view of design science research. In: *Scandinavian Journal of Information Systems* 19 (2007), S. 87–92

[HL95] HÜRSCH, W. L. ; LOPES, C. V.: Separation of Concerns / Northeastern University. 1995. – Technical report NU-CCS-95-03

[Hor07] HORN, Thorsten: *Vorgehensmodelle zum Softwareentwicklungsprozess: Sechs Qualitätsmerkmale für Softwareprodukte nach ISO 9126 (DIN 66272)*. `http://www.torsten-horn.de/techdocs/sw-dev-process.htm`, Version: 2007, Abruf: 18.01.2014

[HSM13] *Hochschule Mittweida App*. `https://www.hs-mittweida.de/webs/hsmwmobil/`, Version: 2013, Abruf: 16.12.2013

[ind14] *Can I use IndexedDB? Compatibility table for support of IndexedDB in desktop and mobile browsers.* `http://caniuse.com/indexeddb`, 2014. – Abruf: 2014-02-05

[Jmo14] *jQuery mobile: Widgets*. `http://api.jquerymobile.com/category/widgets/`, Version: 2014, Abruf: 05.02.2014

[Lau02] LAUESEN, Soren: *Software Requirements: Styles & Techniques*. Harlow, Essex, UK : Pearson Education, 2002

[Leh12] LEHTIMÄKI, Juhani: *Smashing Android Ui*. West Sussex, PO19 8SQ, UK : John Wiley & Sons, 2012

[LR01] LEFF, A. ; RAYFIELD, J.T.: Web-application development using the Model/View-/Controller design pattern. In: *Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing*. Seattle, WA, USA, 2001, S. 118–127

[Mar99] MARSHALL, Dave: *Remote Procedure Calls*. `http://www.cs.cf.ac.uk/Dave/C/node33.html`, Version: 1999, Abruf: 05.02.2014

[Mar13] *Apple Cedes Market Share in Smartphone Operating System Market as Android Surges and Windows Phone Gains, According to IDC*. `http://www.idc.com/getdoc.jsp?containerId=prUS24257413`, Version: 2013, Abruf: 18.01.2014

[Men13a] *Mensaplan App*. `https://play.google.com/store/apps/details?id=de.schwarzfa.android.mensaplan&hl=de`, Version: 2013, Abruf: 16.12.2013

[Men13b] MENDOZA, Adrian: *Mobile User Experience: Patterns to Make Sense of it All*. Waltham, MA 02451, USA : Elsevier Science, 2013

[MKS11] MALY, F. ; KRIZ, P. ; SLABY, A.: Mobile oriented software architecture M client M client server. In: *Proceedings of the ITI 2011 33rd International Conference on Information Technology Interfaces (ITI)*. Cavtat, Croatia, 2011, S. 109–114

[Mob13] *Schub fürs mobile Breitband*. `http://www.bitkom.org/de/presse/8477_76810.aspx`, Version: 2013, Abruf: 08.01.2014

[myU13] *Universität Duisburg-Essen App*. `https://www.uni-due.de/myude/funktionen.shtml`, Version: 2013, Abruf: 16.12.2013

[Nei12] NEIL, Theresa: *Mobile Design Pattern Gallery*. Sebastopol, CA 95472, USA : O`Reilly Media, 2012

[Nie00] NIELSEN, Jakob: *Why You Only Need to Test with 5 Users*. http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/, Version: 2000, Abruf: 19.02.2014

[Nod12] *Future-proofing Your Apps: Cloud Foundry and Node.js*. http://blog.cloudfoundry.com/2012/06/27/future-proofing-your-apps-cloud-foundry-and-node-js/, Version: 2012, Abruf: 29.01.2014

[Nod14] *Node.js v0.10.26 Manual & Documentation*. http://nodejs.org/api/modules.html, Version: 2014, Abruf: 05.02.2014

[Obj12] *Programming with Objective-C*. https://developer.apple.com/library/mac/documentation/cocoa/conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html, Version: 2012, Abruf: 28.02.2014

[OFF11] OFFERGELD, Michael: *Skript zur Vorlesung Usability Engineering*. Universität Ulm, 2011

[Pho12] *PhoneGap Explained Visually*. http://phonegap.com/2012/05/02/phonegap-explained-visually/, Version: 2012, Abruf: 28.02.2014

[Pho14a] *PhoneGap Documentation: Plugin Development Guide*. http://docs.phonegap.com/en/3.0.0/guide_hybrid_plugins_index.md.html#Plugin%20Development%20Guide, Version: 2014, Abruf: 19.02.2014

[Pho14b] *PhoneGap Documentation: Storage*. http://docs.phonegap.com/en/3.0.0/cordova_storage_storage.md.html#Storage, Version: 2014, Abruf: 19.02.2014

[po214a] *Änderungen in der Prüfungsordnung 2010 (Informatikstudiengänge)*. http://www.uni-ulm.de/in/fakultaet/studium/inf-mi/po-2010-inf.html, Version: 2014, Abruf: 19.01.2014

[po214b] *Neue Fachspezifische Prüfungsordnung Informatik, Medieninformatik und Software-Engineering (2012)*. http://www.uni-ulm.de/in/fakultaet/studium/plaene-ordnungen/akkreditierung2012.html, Version: 2014, Abruf: 19.01.2014

[Pre12] PRESTON, Scott: *Learn HTML5 and JavaScript for iOS*. New York, NY 10013, USA : Apress, 2012

[Pro11] *Intro to Node.JS for .NET Developers*. http://www.aaronstannard.com/post/2011/12/14/Intro-to-NodeJS-for-NET-Developers.aspx, Version: 2011, Abruf: 05.02.2014

[RC08] RUBIN, Jeff ; CHISNELL, Dana: *Handbook of Usability Testing*. 2. Auflage. Indianapolis, Indiana, USA : Wiley Publishing, Inc., 2008

[RG11] ROGALL-GROTHE, Cornelia: *Deutschland auf dem Weg in die Informationsgesellschaft*. http://www.bmi.bund.de/SharedDocs/Reden/DE/2011/06/strg_egovernment.html, Version: 2011, Abruf: 19.02.2014

[Rou13] *routes: Minimalist route matching for javascript*. https://www.npmjs.org/package/routes, Version: 2013, Abruf: 01.03.2014

[Rub13] *Ruhr-Universität Bochum App*. http://www.ruhr-uni-bochum.de/mobile/index.php, Version: 2013, Abruf: 16.12.2013

[Rup07] RUPP, Chris: *Requirements-Engineering und -Management: Professionelle, iterative Anforderungsanalyse für die Praxis*. 4. Auflage. Nürnberg : Carl Hanser Verlag GmbH & Co. KG, 2007

[Sta10] STARK, Jonathan: *Building Android Apps with HTML, CSS, and JavaScript*. Sebastopol, CA 95472, USA : O`Reilly Media, 2010

[Stu13] *Study-App*. http://study-app.uni-ulm.de/, Version: 2013, Abruf: 16.12.2013

[Tub13] *Universität Tübingen App*. https://play.google.com/store/apps/details?id=de.uni_tuebingen&hl=de, Version: 2013, Abruf: 16.12.2013

[Uni13] *UniBib App*. https://play.google.com/store/apps/details?id=de.jonaspfeifer.android.unibib&hl=de, Version: 2013, Abruf: 16.12.2013

[v8e13a] *Chrome V8*. https://developers.google.com/v8/, Version: 2013, Abruf: 29.01.2014

[v8e13b] *Chrome V8: Design Elements*. https://developers.google.com/v8/design, Version: 2013, Abruf: 29.01.2014

[War12] WARGO, John M.: *PhoneGap Essentials: Building Cross-Platform Mobile Apps*. Upper Saddle River, New Jersey 07458, USA : Addison-Wesley Professional, 2012

[Web13a] *Web SQL Database*. http://www.w3.org/TR/webdatabase/, Version: 2010, Abruf: 05.02.2014

[Web13b] *Web Storage*. http://www.w3.org/TR/webstorage/, Version: 2013, Abruf: 05.02.2014

[Web14] *WebView*. http://developer.android.com/reference/android/webkit/WebView.html, Version: 2014, Abruf: 19.02.2014

Name: David Damaschk Matrikelnummer: 722867

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

David Damaschk