



BWOLF
EINE WEBBASIERTE PLATTFORM ZUR
EINSCHREIBUNG UND VERWALTUNG DES
EMPIRIEPRAKTIKUMS AN DER FSU JENA

Dokumentation

*Christoph Keiner, Matthias Reuse,
Ingo Schäfer, Christoph Staudt*

8. Februar 2018

Eigenständigkeitserklärung

Wir erklären, dass wir die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt haben.

Jena, 8. Februar 2018

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Aufgabenstellung	2
1.3	Aufbau der Arbeit	2
2	Anforderungsanalyse	4
2.1	Anforderungen an das System	4
2.1.1	Ablauf eines Praktikumsmoduls	5
2.1.2	Verschiedene Sichten im Überblick	6
2.2	Der Verteilungsalgorithmus	8
2.3	Technische Details	8
3	Entwurf	11
3.1	Datenmodell	11
3.2	Design	13
3.2.1	Frontend	14
3.2.2	Backend	16
4	Umsetzung	20
4.1	Verwendete Software	20
4.2	Frontend	20
4.3	Backend	20
5	Verteilungsalgorithmus	24
5.1	Aufstellen Zielfunktion	24

5.2	Erweiterung der Zielfunktion und Wahl der Parameter	25
5.2.1	Naheliegende Wahl der Parameter	25
5.2.2	Exponentielle Gewichte	26
5.2.3	Festes Minimum	26
5.2.4	Betrachtung der Varianz	27
6	Test des Systems	29
6.1	Frontend	29
6.1.1	Homepage	29
6.1.2	Registrierung	30
6.1.3	Login	30
6.1.4	Kursübersicht	30
6.1.5	Meine Präferenzliste	31
6.1.6	Ergebnis sehen	31
6.2	Backend	31
6.3	Algorithmus	31
6.3.1	Gleichverteilung	32
6.3.2	Normalverteilung	33
6.4	E-Mail Benachrichtigung	33
7	Zusammenfassung	35
8	Anhang	37

Kapitel 1

Einleitung

1.1 Motivation

Jedes Jahr wird an der Friedrich-Schiller-Universität Jena das Empiriepraktikum vom Institut für Psychologie angeboten. Dabei handelt es sich um eine Pflichtveranstaltung, die jeder Psychologie-Student absolviert haben muss. Innerhalb eines Praktikums werden mehrere Kurse angeboten. An einem dieser Kurse muss der Studierende teilnehmen, um das Modul erfolgreich abzuschließen. Allerdings ist die Teilnehmerzahl eines jeden Kurses begrenzt. So kann nicht jeder der über einhundert Studenten zu seinem Wunschkurs zugelassen werden. Die naheliegende Lösung ist die Zuweisung zu den Kursen nach Geschwindigkeit der Studenten bei der Anmeldung. Allerdings ist diese Verteilungsstrategie mit viel Stress und Streit verbunden. Die Studenten müssen anders auf die Kurse verteilt werden. Hierzu dient dem Institut für Psychologie eine Zuweisung zu den Kursen mittels Präferenzlisten. Jeder Studierende erstellt eine Liste mit Präferenzen, wie gerne er an welchen Kurs teilnehmen möchte. Es liegt auf der Hand, dass diese Listen nun nicht alle per Hand ausgewertet werden können, sondern automatisch erfolgen muss. Für diese Aufgabe existieren bereits Lösungen. So können mit dem Studienverwaltungstool der FSU Jena Friedolin Studenten bereits Präferenzlisten erstellt und ausgewertet werden. Bei der Verteilung mithilfe von Friedolin kommt es jedoch zu Problemen, da beim Erstellen der Präferenzliste **getrickst** werden

kann. Aus diesem Grund hat das Institut für Psychologie bereits eine eigene Plattform geschaffen, die das aufnehmen der Präferenzliste und das Verteilen auf die Kurse übernimmt. Hierbei ist aber die Verteilung auf die verschiedenen Kurse oft nicht zufriedenstellend und es muss manuell das Ergebnis angepasst werden. Das mit dieser Arbeit verbundene Projekt beschäftigt sich nun mit dem Erstellen einer neuen Plattform für das Empiriepraktikum und der Frage, wie sich die Studenten besser auf die Kurse verteilen lassen.

1.2 Aufgabenstellung

Es gilt eine webbasierte Plattform für die Verwaltung des Empiriepraktikums der FSU Jena zu erstellen. Hauptaufgabe dieser Plattform ist, den verantwortlichen Mitarbeitern des Instituts für Psychologie zu ermöglichen, für ein kommendes Semester ein neues Praktikumsmodul mit allen zugehörigen Kursen anlegen zu können. Des Weiteren sollen die Studenten die Möglichkeit erhalten eine Präferenzliste zu erstellen, welche der angebotenen Kurse sie am liebsten besuchen möchten. Nach Ablauf einer vorher festzulegenden Frist sollen die Studenten gemäß ihrer Präferenzliste dann automatisch bestmöglich auf die Kurse verteilt werden. Hierzu soll ein geeigneter Verteilungsalgorithmus entwickelt werden.

1.3 Aufbau der Arbeit

Zunächst werden in dieser Arbeit die Anforderungen an die Plattform, sowie den Verteilungsalgorithmus ausgehend von der Aufgabenstellung aus Abschnitt 1.2 näher spezifiziert. Anschließend wird der Entwurf für die Realisierung der Anforderungen an das zu entwickelnde System dargestellt. Danach folgt eine Beschreibung der tatsächlichen Umsetzung des Systems. Im Anschluss wird der Algorithmus zur Verteilung der Studenten formal beschrieben. Hiernach werden die durchgeführten Tests zu Plattform und Verteilungsalgorithmus vorgestellt. Abschließend wird die Arbeit nochmals kurz zusammengefasst und ein Ausblick auf weitere umzusetzende Funktionalitäten gegeben.

ten gegeben.

Kapitel 2

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an die zu schaffende Plattform dargestellt. Zunächst wird im Abschnitt 2.1 behandelt, wie ein Praktikumsmodul abläuft und welche Erfordernisse sich daraus für die einzelnen Teilnehmergruppen ergeben. Darauf folgt eine Beschreibung der Aufgaben des Verteilungsalgorithmus. Zuletzt werden die technischen Anforderungen an das Projekt beschrieben.

2.1 Anforderungen an das System

Gemäß der allgemeinen Aufgabenstellung aus Abschnitt 1.2 existieren verschiedene Sichtweisen auf die Anforderungen des Systems. Zum einen die Sicht der Verantwortlichen für Praktikum und Kurse, zum anderen die der Studenten. Erstere teilt sich wiederum auf in den Blickwinkel der Dozenten der einzelnen Kurse und der übergeordneten Verantwortlichen für das Empiriepraktikum, im Weiteren Administratoren genannt. Im Folgenden werden zunächst die Anforderungen für Studenten, Dozenten und Administratoren anhand des chronologischen Ablaufs eines Praktikumsmoduls dargestellt. Anschließend wird nochmal für die jeweiligen Sichten eine kurze Übersicht gegeben, sowie weitere detailliertere Anforderungen genannt.

Die Anforderungen sind durch ein Gespräch mit den Verantwortlichen für das

Empiriepraktikum entstanden. Nicht alle Anforderungen sind zwingend für die Funktion des Systems. Einige sind lediglich optionale Funktionalitäten. Es ist zu beachten, dass aus zeitlichen Gründen nicht alle Anforderungen im Rahmen des Projektes umgesetzt werden können. Sie werden der Vollständigkeit halber dennoch aufgeführt.

2.1.1 Ablauf eines Praktikumsmoduls

Der Ablauf beginnt, indem die Administratoren, nachdem sie sich in einer Login-Oberfläche angemeldet haben, ein neues Praktikumsmodul erstellen. Zu diesem Praktikumsmodul gehören neben generellen Informationen wie Name und Semester auch die besonderen Angaben, ab wann Studenten ihre Präferenzliste erstellen können und zu welchem Zeitpunkt die automatische Verteilung vorgenommen werden soll. Im Anschluss können die Dozenten, nachdem auch sie sich in einer entsprechenden Oberfläche angemeldet haben, ihre Kurse zu dem aktuellen Praktikumsmodul hinzufügen. Dabei sollen Kurse Angaben über Titel, Dozent, Teilnehmerzahl, Ort, Zeit, Beschreibung des Kurses und eine Literaturliste besitzen. Zusätzlich soll ein Kurs Informationen über den Lehrstuhl, sowie die Finanzierung erhalten, die jedoch nicht für Studenten einsehbar sein soll. Nachdem alle Dozenten ihre Kurse eingetragen haben, sollen die Administratoren die aktuelle Kursübersicht online stellen können, sodass jeder die Kurse einsehen kann. Besteht Interesse, dass Empiriepraktikum in diesem Semester zu absolvieren, so sollen sich die Studierenden registrieren können. Nach der Registrierung können die Studenten ihre Präferenzliste erstellen und speichern. Dabei soll die Präferenzliste auch jederzeit vom Studierenden noch verändert werden können. Nach jeder gespeicherten Änderung soll der Student eine Email-Benachrichtigung mit seiner aktuellen Präferenzliste erhalten. Nach Ablauf der zuvor von den Administratoren festgelegten Frist sollen die Studenten automatisch anhand ihrer aktuellen Version der Präferenzliste auf die Kurse verteilt werden. Diese Verteilung sollte möglichst gut sein, d.h. in Fall dieses Projekts eine möglichst gleichmäßige Verteilung der Studenten mit wenigen Ausreißern nach Unten. Nach dieser automatischen Verteilung soll eine Nachbearbeitungsphase fol-

gen, in der die Administratoren den Algorithmus bei Bedarf mit vielleicht anderen Parametern neu starten können. Es soll jedoch auch die Möglichkeit geben, das Ergebnis der Verteilung manuell zu ändern. Nachdem das Ergebnis der Verteilung feststeht, sollen alle beteiligten Studenten, Dozenten und Administratoren über das Ergebnis der Verteilung mittels einer Email-Benachrichtigung informiert werden. Ab diesem Moment sollen die Teilnehmer jedes Kurses für jeden angemeldeten Benutzer sichtbar sein. Anschließend an diese Nachbearbeitungsphase, könnte sich optional eine Tauschphase anschließen, in der die Studenten selbstständig über eine geeignete Oberfläche ihre Kurse tauschen können.

2.1.2 Verschiedene Sichten im Überblick

Sicht der Administratoren

Administratoren haben die Möglichkeit ein neues Praktikum zu erstellen. Dabei müssen sie Angaben über Name, Semester, Frist für die Anmeldung und Beginn der automatischen Verteilung festlegen. Sie sollen aktiv die Kursübersicht für das aktuelle Praktikum veröffentlichen können. Des Weiteren können Administratoren den Verteilungsalgorithmus in der Nachbearbeitungsphase erneut mit anderen Parametern starten und die Verteilungsergebnisse auch manuell verändern. Zusätzlich sollen Administratoren die angemeldeten Dozenten verwalten können, um z.B. Dozenten, die keine Kurse mehr anbieten zu entfernen. Auch sollen die Administratoren Kurse als „frei bleiben“ markieren können, sowie die maximale und minimale Teilnehmerzahl für alle Kurse fließend einstellen können, falls nötig. Die Administratoren sollen für Praktika auch die Möglichkeit erhalten Filterfunktionen anzuwenden und sich so z.B. die Kurse nach Lehrstühlen sortiert anzeigen zu lassen.

Sicht der Dozenten

Ein Dozent hat soll die Möglichkeit haben, Kurse für das aktuelle Praktikum zu erstellen und sich eventuell seine Kurse für ältere Praktika nochmals anzusehen, um sie als Vorlage für neue Kurse zu nutzen. Für einen Kurs

müssen sie folgendes angeben: Name, Titel, Dozenten, Zeit und Raum, Teilnehmerzahl (hierbei können die Dozenten nur zwischen 5 oder 10 wählen), eine Kurzbeschreibung des Kurses, eine ausführliche Beschreibung und optional eine Literaturliste. Diese Angaben sind auch für Studenten einsehbar. Nicht für Studenten einsehbar sind die folgenden Angaben über Lehrstuhl, Lehrauftrag und die Information, ob der Dozent das erste Mal ein Empiriepraktikum leitet.

Der Inhalt von längeren Angaben wie der Beschreibung des Praktikums, soll in einer Textumgebung möglich sein, in der geeignete Textformatierung möglich ist. Außerdem sollen auch Bilder in die Beschreibung eingearbeitet werden können. Des Weiteren sollen Dozenten nur ihre eigenen Kurse editieren können. Nachdem die Studenten verteilt worden sind, sollen die Dozenten eine E-Mail mit den Studenten, die in ihrem Kurs sind, erhalten. **Im Verlaufe des Semesters können Dozenten ihren Kurs in zwei Kurse aufteilen.**

Sicht der Studenten

Die Studenten sollen ohne eine Registrierung die Kursübersicht aufrufen können, sobald die Administratoren die Kursübersicht veröffentlicht haben. Nach der Registrierung sollen die Studenten bis zu einer Frist eine Präferenzliste erstellen und späterhin auch bearbeiten können. Bei jeder angenommenen Änderung der Präferenzliste soll der Student über seine aktuelle Wahl per Email informiert werden. Nachdem die Verteilung vom Algorithmus vorgenommen wurde, sollen die Studenten über ihr Ergebnis informiert werden, sowie die Ergebnisse der gesamten Verteilung einsehen können. Anschließend sollen sie eventuell die Möglichkeit erhalten eigenständig die Kurse mit anderen Studenten zu tauschen, sofern beide zustimmen.

Das ganze offen und Modular halten für andere Module als das Empiriepraktikum

Dabei auf Hierarchie von Präferenzen achten

evtl. die Möglichkeit falls mehr Studenten als Kurse, neue Kurse hinzufügen können

evtl. ein Archiv

2.2 Der Verteilungsalgorithmus

Der Verteilungsalgorithmus verteilt alle Studenten auf die Kurse. Dabei ist es wichtig, dass die aufaddierte maximale Teilnehmeranzahl der Kurse größer ist als die Anzahl der zu verteilenden Studenten. Sollte dem nicht der Fall sein, so wird der Administrator informiert. Dieser hat dann die Möglichkeit die maximale Teilnehmeranzahl von Kursen zu erhöhen. Die Eingabe des Algorithmus sind die Gewichte der Präferenzen und (optional) die Auswahl eines Optimierungsalgorithmus.

Startet der Algorithmus, so versucht er, für jeden Studenten die größte, mögliche Präferenz zu den Kursen zu wählen. Dabei ist insbesondere wichtig, dass die Streuung der gewählten Präferenzen möglichst gering ist. Dies erfolgt beispielsweise durch eine Gewichtung der Präferenzen.

Weiterhin platziert der Algorithmus in jeden Kurs mindestens drei Studenten, damit der Kurs sinnvoll angeboten werden kann. Es werden aber nie mehr Teilnehmer einem Kurs zugeordnet als die maximale Teilnehmeranzahl des Kurses vorgibt.

Hat der Algorithmus schließlich eine passende Zuordnung von Studenten zu Kursen gefunden, so wird der Administrator zuerst benachrichtigt. Dieser kann die Ergebnisse bearbeiten, Parameter des Algorithmus neu einstellen, Kurse aktivieren/deaktivieren und ihn erneut starten. Ist der Administrator zufrieden mit der Verteilung, so bestätigt dieser das Ergebnis. Anschließend wird an die Dozenten eine E-Mail mit ihren Teilnehmern geschickt, und Studenten erhalten eine E-Mail mit ihrem Kurs.

2.3 Technische Details

Der Login wird über eine E-Mail/Passwort Authentifizierung realisiert. Dafür werden nur E-Mail Adressen der FSU Jena nutzbar sein. Dies bedeutet, dass E-Mails auf "@uni-jena.de"enden müssen. Man kann sich zu jeder Zeit

ein- und ausloggen.

Bei der Wahl der Präferenzen darf keine Präferenz doppelt belegt werden, d.h. jede Präferenz hat einen eindeutigen Kurs.

Der Verteilungsalgorithmus kann mit verschiedenen Optionen angesteuert werden. Optionen können u.a. die Gewichtung der Varianz, oder welche Kurse belegt werden können, sein. Weiterhin ist der Verteilungsalgorithmus innerhalb von 24 Stunden fertig.

- Man kann sich zu einem beliebigen Zeitpunkt einloggen und ausloggen.
- Der Verteilungsalgorithmus braucht maximal 24 Stunden.
- Die Website muss auch auf mobilen Endgeräten funktionieren.
- Sollte ein Browser älter als IE11 sein, so wird den Nutzer angezeigt, dass sie ihren Browser updaten sollen.
- Die Website wird auf einer docker-compose Umgebung aufgesetzt.
- Die Website wird mit OctoberCMS mit Laravel 5.5 umgesetzt.
- Der Webserver wird mit Nginx umgesetzt.
- Die Datenbank wird mit MySQL umgesetzt.
- Der Cache-Server wird mit Redis umgesetzt.
- Tests erfolgen für die wesentlichsten Bestandteile, insbesondere für den Algorithmus.
- Kursinformationen sind öffentlich einsehbar.
- Kurse können von nicht-Studenten insbesondere nach folgenden Kriterien gefiltert werden:
 - Lehrstuhl

- Finanzierung/Lehrauftrag
- Erstes Mal Empiriepraktikum?

Kapitel 3

Entwurf

Nachdem im vorangegangenen Kapitel die Anforderungen für das System spezifiziert wurden, soll in diesem Kapitel der Umsetzungsentwurf der verschiedenen Sichten dargestellt werden. Zunächst wird das dem System zugrundeliegende Datenmodell vorgestellt. Darauffolgend wird das Design für die webbasierte Plattform erläutert.

3.1 Datenmodell

Der Entwurf für das Datenmodell ist in Abbildung 3.1 zu sehen. Es besteht aus sieben Tabellen. Die Tabelle *bwolfjena_core_modules* beinhaltet die verschiedenen Empiriepraktika. Der Name Module für die Tabelle wurde gewählt, da das System bei Bedarf auch auf andere ausgewählte Module als das Empiriepraktikum erweitert werden kann.

In der Tabelle *bwolfjena_core_courses* sind die verschiedenen Kurse gespeichert. Ein Kurs gehört immer zu einem bestimmten Modul beziehungsweise einem Empiriepraktikum. Um das abzubilden, ist für jeden Kurs mittels eines Fremdschlüssels das Empiriepraktikum, zu dem er gehört, gespeichert.

Ein Kurs wird immer von einem Lehrstuhl angeboten, die in der Tabelle *bwolfjena_core_chairs* abgelegt sind. Auch auf diese Tabelle existiert wieder ein entsprechender Fremdschlüssel in *bwolfjena_core_courses*.

Alle angemeldeten Studenten werden in der Tabelle *users* gespeichert. Jeder

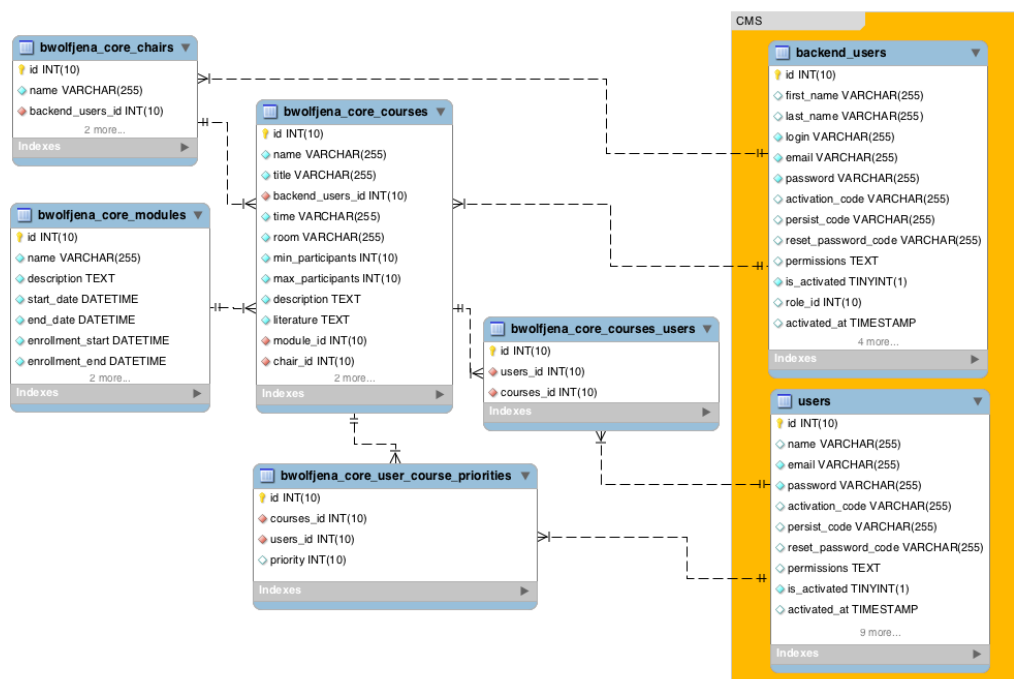


Abbildung 3.1: Entwurf des Datenmodells. Farbcode für Spalten der Tabellen: Gelb Primärschlüssel, Rot Fremdschlüssel, Blau gewöhnliche Variable, Weiß nullable-Variable






	Eingabefeld
	Button
	Drag&Drop-Element
	Textfeld
	Optisches Element

Tabelle 3.1: Farbcode der Mockups

der Studenten hat eine Präferenzliste. Die Einträge der Präferenzliste für jeden Studenten und jeden Kurs werden in Tabelle *bwolfjena_core_priorities* gespeichert. Dementsprechend beinhaltet *bwolfjena_core_priorities* Fremdschlüssel von *users* und *bwolfjena_core_courses*.

In der Tabelle *bwolfjena_core_users* ist abgelegt, welcher Student welchem Kurs zugeteilt wurde. Aus diesem Grund beinhaltet *bwolfjena_core_users* Fremdschlüssel auf *bwolfjena_core_courses* und *users*.

Die letzte Tabelle *backend_users* beinhaltet die Backendbenutzer. Damit sind die Dozenten und Administratoren gemeint. In *bwolfjena_core_courses* ist ein Fremdschlüssel für die Backendbenutzer gespeichert, um den Dozenten anzugeben, der den Kurs leitet. Auch die Tabelle *bwolfjena_core_chairs* hat einen Fremdschlüssel zu der Tabelle *backend_users*, um den Lehrstuhlinhaber anzugeben.

Neben den beschriebenen Fremdschlüsseln existieren für die Einträge in jeder Tabelle IDs als Primärschlüssel. Die weiteren Spalten der Tabellen sind entsprechend den Anforderung aus Kapitel 2 gewählt.

3.2 Design

Für den Desing-Entwurf der Seite wurden Mock-Ups erstellt, die den groben Aufbau der Website mit den entsprechenden Funktionen zeigen. Im folgenden werden diese Mock-Ups für das sogenannte Frontend, also aus Sicht der Studenten, und für das Backend, die Sicht der Dozenten beziehungsweise der Administratoren, vorgestellt. Dabei gilt der in Tabelle 3.1 angegebene Farbcode für die verschiedenen Elemente der Mock-Ups.

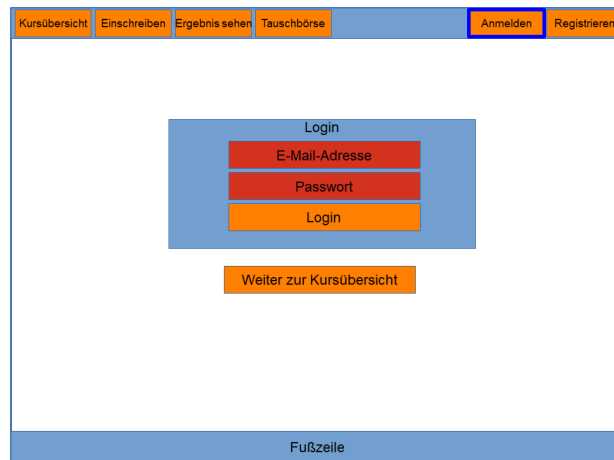


Abbildung 3.2: Entwurf für die Login-Oberfläche

3.2.1 Frontend

Wie in Kapitel 2 bereits ausgeführt, sollen die Studenten zunächst eine Registrierungs- bzw. Login-Oberfläche sehen. Jedoch sollen die verschiedenen Kurse auch ohne eine Anmeldung einsehbar sein.

In Abbildung 3.2 werden beide Anforderungen umgesetzt. Zum einen die Login-Oberfläche, in der in zwei Textfeldern Benutzername und Passwort für einen erfolgreichen Login eingetragen werden müssen, zum anderen die direkte Weiterleitung zur Kursübersicht als einfacher Knopf darunter.

Die Kopfzeile umfasst einige Reiter. So kann mithilfe der Kopfzeile auf die verschiedenen Funktionen des Frontends gewechselt werden. Dazu zählen die Kursübersicht, die Erstellung der Präferenzliste, das Einsehen der Verteilungsergebnisse, sowie das Tauschen von Kursen. Außer der Navigation auf die Kursübersicht sind diese jedoch erst nach einer erfolgreichen Anmeldung funktional. Ohne eine Anmeldung wird der Benutzer wieder auf die Login-Oberfläche verwiesen. In der Kopfzeile soll auch eine Schaltfläche zum erstmaligen Registrieren im System sein. Die Registrierungsseite gleicht strukturell der Login-Oberfläche. Sobald ein Benutzer sich erfolgreich angemeldet hat, soll der Schalter zum Anmelden in der Kopfzeile gegen einen Abmelden-Knopf ausgetauscht werden. Auf allen weiteren Seiten des Frontends soll die Kopfzeile die gleichen Funktionen und das gleiche Aussehen haben, der

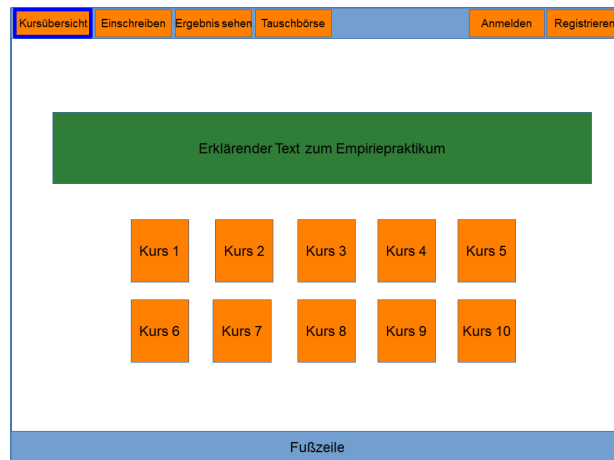


Abbildung 3.3: Entwurf für die Kursübersicht

Registrieren-Knopf entfällt jedoch, sobald die Anmeldung erfolgt ist. Die Fußzeile soll weitere allgemeine Informationen bereitstellen, sofern diese von Nöten sein sollten, aber vor allem als optischer Abschluss der Seite dienen. Auch sie soll auf alle folgenden Seiten übernommen werden.

In Abbildung 3.3 ist die Seite für die Kursübersicht zu sehen. Unter der Kopfzeile folgt eine textuelle Erklärung des Ablaufs des Empiriepraktikums und der für die Studenten relevanten Schritte, um sich erfolgreich für die Kurse einzuschreiben. Die verschiedenen Kurse werden darunter jeweils mit einer kurzen Beschreibung und den wichtigsten Informationen angezeigt. Durch einen Klick auf ein Kursfeld, sollen sich die detaillierten Informationen zu dem Kurs einsehen lassen.

Die Seite für die Einschreibung in die Kurse soll wie in Abbildung 3.4 gestaltet sein. Wieder bilden Kopf- und Fußzeile den Rahmen der Seite. Unter der Kopfzeile befindet sich auch hier eine kurze Erklärung, wie man die Präferenzliste genau erstellt. Das Erstellen soll im Feld *Präferenzliste* über ein „Drag&Drop“-System vorgenommen werden und über den Knopf *Absenden* fixiert werden können.

Nachdem die Verteilung auf die Kurse vorgenommen wurde, sollen die Benutzer unter dem Reiter Ergebnisübersicht die Resultate einsehen können. Der Entwurf hierfür gleicht dem für die Kursübersicht. Statt einer Beschreibung des Kurses, werden allerdings die Teilnehmer der Kurse angezeigt.

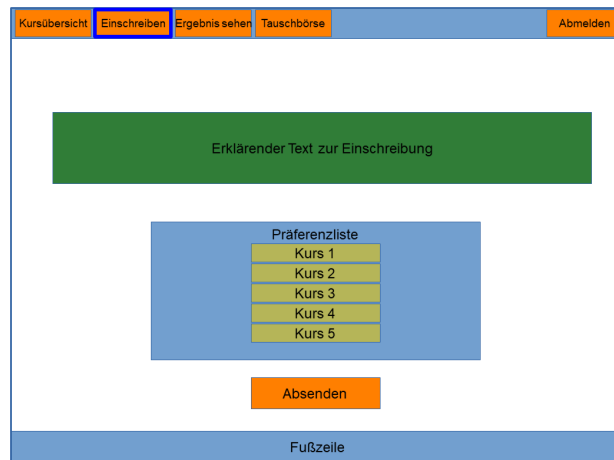


Abbildung 3.4: Entwurf für die Einschreibungs-Oberfläche

Unter dem Reiter *Tauschbörse* soll die in Kapitel 2 genannte Tauschmöglichkeit für Studenten realisiert sein. Die Struktur der Seite gleicht dem der Kursübersicht. Unter einer Information in welchem Kurs man sich befindet und wie die Tauschfunktionen zu benutzen sind, finden sich die verschiedenen Kurse, genauso wie in der Kursübersicht. Der Kurs, dem man zugeteilt ist, erscheint ausgegraut oder auf eine andere Art und Weise markiert.

Durch einen Klick auf den Kurs in den gewechselt werden möchte, öffnet sich die in Abbildung 3.5 gezeigte Seite. Hier kann in dem Textfeld *Begründung* eine Begründung für den Tausch angegeben werden. Mit der Schaltfläche *Absenden* kann die Tauschanfrage abgeschickt werden, mit *Abbrechen* gelangt der Benutzer wieder auf die vorherige Ansicht der Tauschbörse.

3.2.2 Backend

Im Folgenden werden die wichtigsten Mockups für Administratoren und Dozenten vorgestellt. Für beide Sichten gibt es wieder eine Kopf- und eine Fußzeile. Die Kopfzeile für die Administratoren umfasst zunächst einen Reiter *Verwalten*, in dem Kurse, Module und Lehrstühle angelegt werden können. Des Weiteren gibt es den Reiter *Benutzer*, unter dem alle Benutzer des Systems verwaltet werden können. Unter dem Reiter *Verteilung* können die Administratoren den Verteilungsalgorithmus starten und sich die Ergebnisse

Abbildung 3.5: Entwurf für die Tauschanfrage

anzeigen lassen. Zusätzlich gibt es noch eine Seitenleiste, in der unter dem jeweiligen Reiter weitere Optionen zur Verfügung stehen.

In Abbildung 3.6 ist beispielhaft für den Reiter Verwaltung, die der Kursverwaltung zu sehen. Mithilfe der Seitenleiste am linken Rand, kann zwischen dem Verwalten von Kursen, Modulen (Praktika) und Lehrstühlen gewechselt werden. Alle Vorhandenen Kurse (bzw. Module oder Lehrstühle) werden in einer Tabelle angezeigt. Mit den beiden Schaltflächen am oberen Rand der Tabelle kann ein neuer Eintrag in der jeweiligen Tabelle erstellt, bzw. ein vorhandener Eintrag gelöscht werden. Die Einträge der entsprechenden Tabelle können nachträglich durch einen Klick auf den Eintrag in der Tabelle geändert werden.

Abbildung 3.7 zeigt den Entwurf für das Erstellen oder Ändern eines Kurses, Moduls oder Lehrstuhls. Alle notwendigen Informationen sollen in Textfeldern eingegeben und mit geeigneten Textformatierungsoptionen bearbeitet werden können. Mit einem Klick auf die Schaltfläche *Speichern* wird die Änderung übernommen, bzw. die Erstellung bestätigt.

Die Verwaltung der Benutzer erfolgt auf die selbe Weise und gleicht sich im Design. Wieder kann mithilfe der linken Seitenleiste umgeschaltet werden, ob Studenten oder Dozenten verwaltet werden sollen. Die entsprechende Benutzergruppe wird in einer Tabelle angezeigt und kann mittels einer Schaltfläche

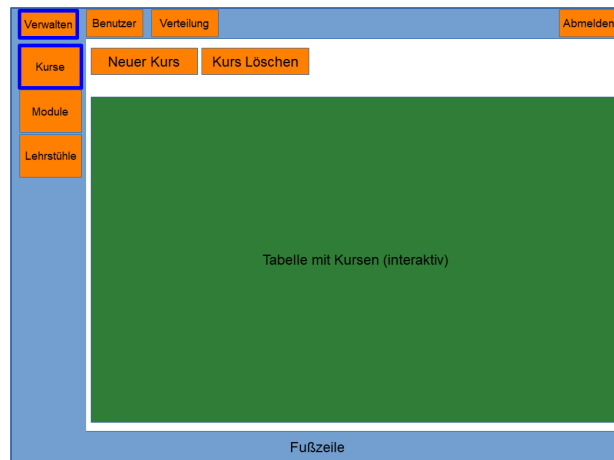


Abbildung 3.6: Entwurf für das Verwalten der Kurse

gelöscht werden. Durch einen Klick auf die Tabellenzeile sollen die Informationen der Benutzer analog zum Bearbeiten der Kurse oder Module geändert werden können.

Für die Steuerung des Verteilungsalgorithmus unter dem Reiter *Verteilung*, muss zunächst aus einer Liste das entsprechende Praktikum ausgewählt werden. Danach wird die Verteilung für dieses Modul mit verschiedenen Parametern ermittelt und anschließend werden die Ergebnisse mithilfe von geeigneten Grafiken angezeigt. Der Entwurf für diese Seite ist in Abbildung 3.8 zu sehen. Durch einen Klick auf den entsprechende Verteilungsvorschlag, soll dieser als Ergebnis übernommen werden.

Die Ansicht der Dozenten unterscheidet sich lediglich im Funktionsumfang. So sehen die Dozenten nur den Reiter *Verwalten* in der Kopfzeile und in der linken Seitenleiste nur die Option zum Verwalten der Kurse. Die Tabelle zeigt jetzt nur die Kurse, die der entsprechende Dozent selbst angelegt hat.

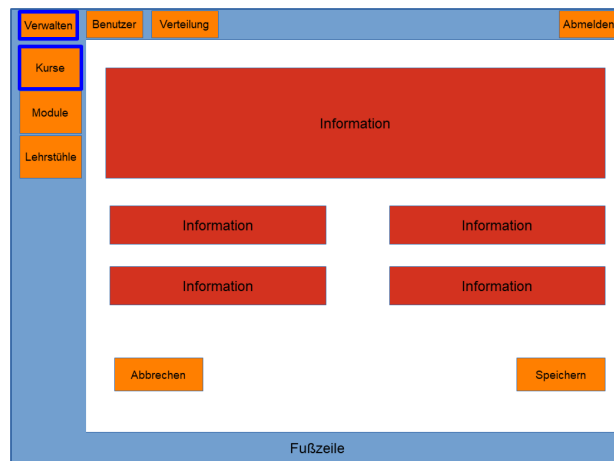


Abbildung 3.7: Entwurf für das editieren eines Kurses

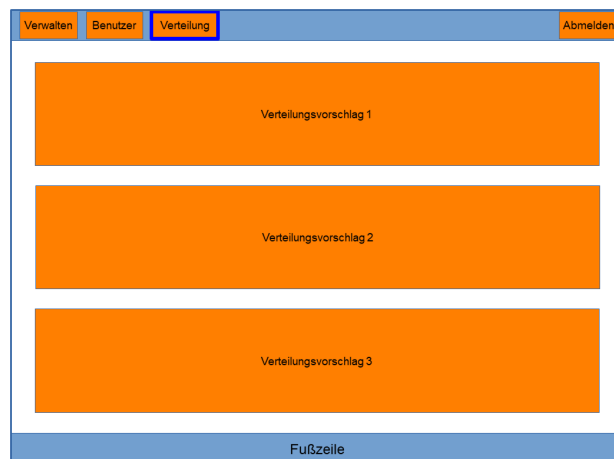


Abbildung 3.8: Entwurf für das Anzeigen der Verteilungsvorschläge

Kapitel 4

Umsetzung

4.1 Verwendete Software

4.2 Frontend

4.3 Backend

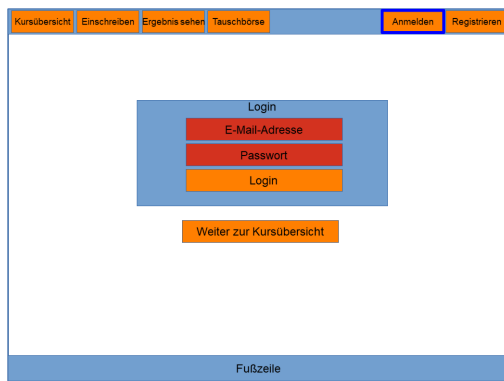


Abbildung 4.1: Gegenüberstellung von Entwurf der Login-Oberfläche (links) und Umsetzung (rechts)

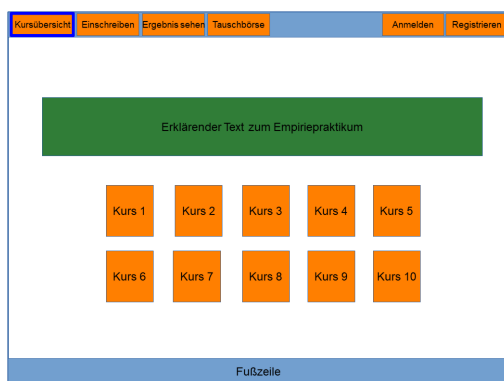


Abbildung 4.2: Gegenüberstellung von Entwurf der Kursübersicht (links) und Umsetzung (rechts)

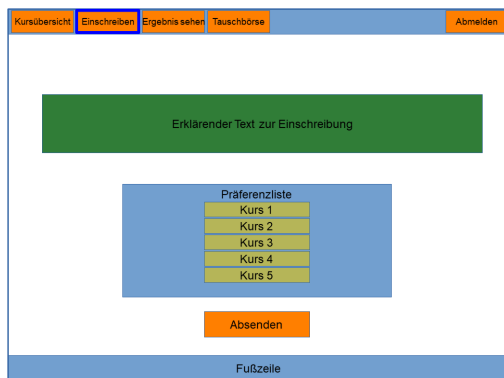


Abbildung 4.3: Gegenüberstellung von Entwurf der Einschreibungs-Oberfläche (links) und Umsetzung (rechts)



Abbildung 4.4: Gegenüberstellung von Entwurf der Kursverwaltung (links) und Umsetzung (rechts)

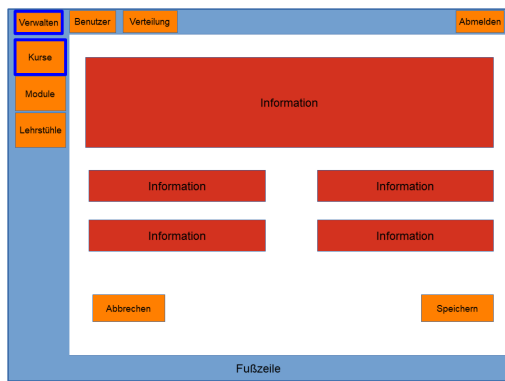


Abbildung 4.5: Gegenüberstellung von Entwurf der Kurserstellung (links) und Umsetzung (rechts)

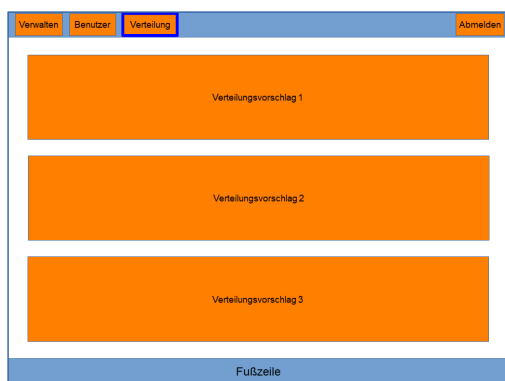


Abbildung 4.6: Gegenüberstellung von Entwurf der Anzeige der Verteilungsvorschläge (links) und Umsetzung (rechts)

Kapitel 5

Verteilungsalgorithmus

Im folgenden Kapitel wird der Algorithmus zum Verteilen der Studenten anhand der Präferenzlisten auf die Kurse vorgestellt. Zunächst wird im ersten Abschnitt eine geeignete Zielfunktion aufgestellt und kurz erklärt. Anschließend werden verschiedene Varianten für die Parameter diskutiert.

5.1 Aufstellen Zielfunktion

Ausgangspunkt für die Zielfunktion des Verteilungsalgorithmus ist eine Matrix $X \in \{0, 1\}^{n \times m}$, wobei n die Anzahl der Studenten und m die Anzahl der Kurse ist. Ein Eintrag in dieser Matrix x_{ij} kodiert, ob ein Student einem Kurs zugeordnet wurde, auf folgende Weise: wurde ein Student i dem Kurs j zugeteilt, dann ist $x_{ij} = 1$, wenn nicht, dann gilt $x_{ij} = 0$. Diese x_{ij} sind die Variablen der Zielfunktion. Das bedeutet, es wird nach einer Belegung der Einträge x_{ij} der Matrix X gesucht, so dass in jeder Zeile der Matrix immer genau ein Eintrag auf 1 gesetzt ist. Das ist gleich Bedeutend mit der Aussage, dass jeder Student genau einem Kurs zugeteilt wurde. Daraus ergibt sich die Zielfunktion zu

$$\max \sum_{i=1}^n \sum_{j=1}^m c(i, j) x_{ij} \quad ,$$

dabei bezeichnet $c(i, j)$ eine Gewichtsfunktion. Die Gewichte bilden die Parameter der Zielfunktion.

Zusätzlich zu dieser linearen Zielfunktion ist es notwendig zwei Nebenbedingungen zu formulieren, um die oben beschriebene Idee der Matrix X zu formalisieren. Zum einen müssen die x_{ij} nur die Werte 0 oder 1 annehmen können:

$$\forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, m\} : x_{ij} \in \{0, 1\} \quad .$$

Zum anderen soll jeder Student nur einem Kurs zugeteilt werden:

$$\forall i \in \{1, \dots, n\} : \sum_{j=1}^m x_{ij} = 1 \quad .$$

Zuletzt ist mit einer dritten Nebenbedingung die Teilnehmerzahl für die Kurse zu begrenzen:

$$\forall j \in \{1, \dots, m\} : t_{\min}(j) \leq \sum_{i=1}^n x_{ij} \leq t_{\max}(j) \quad ,$$

dabei bezeichnet $t_{\min}(j)$ die minimale und $t_{\max}(j)$ die maximale Teilnehmerzahl eines Kurses j .

5.2 Erweiterung der Zielfunktion und Wahl der Parameter

Die Parameter der Zielfunktion können frei gewählt werden. In diesem Abschnitt sollen verschiedene Varianten dargestellt werden, wie die Gewichte gewählt bzw. wie die Zielfunktion weiter angepasst und erweitert werden kann.

5.2.1 Naheliegende Wahl der Parameter

Die nahe liegende Wahl der Gewichte ist, die Präferenzliste der einzelnen Studenten für die Kurse zu verwenden. Da versucht wird, die x_{ij} so zu wählen, dass die Summe maximal wird, wird so eine Zuordnung eines Studenten zu einem Kurs, für den er eine höhere Präferenz angegeben hat, gegenüber einem

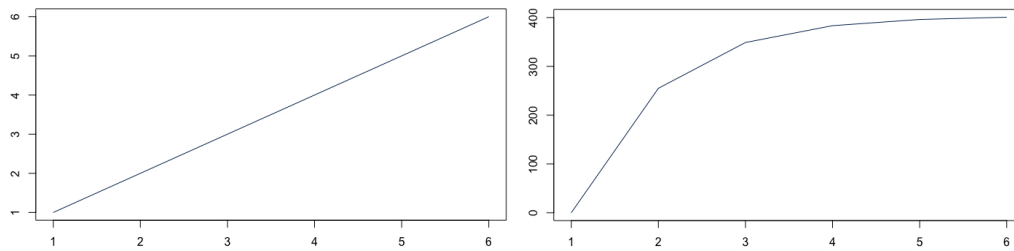


Abbildung 5.1: Gegenüberstellung von linearer Gewichtung (links) und exponentieller Gewichtung (rechts). Die horizontale Achse bezeichnet die Präferenzen, die vertikale den Wert des Gewichts

mit einer niedrigen Präferenz bevorzugt. Der sogenannte *Score* der Funktion, also das Ergebnis der Summe, wird somit höher, je mehr Studenten in von ihnen gewünschte Kurse verteilt werden.

5.2.2 Exponentielle Gewichte

Eine zweite Variante ist die Gewichte nicht nach der Präferenzliste, also linear und in gleichen Abständen, zu wählen, sondern exponentiell. Dadurch wirkt sich das Zuteilen eines Studenten auf einen Kurs, dem er eine hohe Präferenz gegeben hat, deutlich stärker auf den Score aus, als in der vorherigen Variante. Oder andersherum: das Zuteilen zu einem Kurs mit niedriger Präferenz wird stärker mit einem niedrigen Score bestraft.

In Abbildung 5.1 sind die beiden verschiedenen Gewichtungen gegenübergestellt. Wie bereits erwähnt, bestraft die exponentielle Gewichtung die Zuteilung zu Kursen mit niedriger Präferenz deutlich stärker als die lineare Gewichtung.

5.2.3 Festes Minimum

Die Gewichte können bei dieser Erweiterung der Zielfunktion weiterhin frei gewählt werden. So zum Beispiel sind die beiden Ansätze aus dem vorherigen Abschnitt denkbar. Allerdings wird hier zusätzlich ein festes Minimum für die kleinste noch zulässige Präferenz angegeben. Das bedeutet, dass kein

Student einem Kurs zugeteilt wird, dem er eine geringere Präferenz als das Minimum gegeben hat. Dies wird umgesetzt, indem die zweite Nebenbedingungen aufgeteilt wird. Jeder Student muss genau einem Kurs mit ausreichender Präferenz zugeteilt werden:

$$\forall i \in \{1, \dots, n\} : \sum_{\{j \in \{1, \dots, n\} | c(i,j) \geq \text{minPref}\}} x_{ij} = 1 \quad ,$$

und jeder Student darf keinem Kurs mit zu kleiner Präferenz zugeteilt werden:

$$\forall i \in \{1, \dots, n\} : \sum_{\{j \in \{1, \dots, n\} | c(i,j) < \text{minPref}\}} x_{ij} = 0 \quad ,$$

dabei bezeichnet minPref das entsprechend der Gewichtsfunktion transformierte Minimum.

Das Anwenden dieser Methode hat den Vorteil, dass das Minimum fest gewählt werden kann und somit garantiert ist, dass kein Student schlechter als das Minimum eingeteilt wird. Dadurch kann es jedoch dazu kommen, dass einige Studenten in ihrer Präferenz zugunsten des festen Minimums fallen oder auch, dass bei einem zu hohen Minimum keine Lösung für das Problem gefunden werden kann.

5.2.4 Betrachtung der Varianz

Eine weitere Anpassung der Zielfunktion könnte die Erweiterung um einen zusätzlichen Term sein. Es ist so denkbar, die Varianz als Maß für die Streuung mit in die Zielfunktion einzubeziehen. Formal würde das zu folgender Zielfunktion führen:

$$\max \sum_{i=1}^n \sum_{j=1}^m c(i,j)x_{ij} - \frac{\beta}{n} \sum_{i=1}^n \left[\left(\sum_{j=1}^m c(i,j)x_{ij} \right) - \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m c(i,j)x_{ij} \right]^2 \quad ,$$

dabei bezeichnet der hintere Term die empirische Varianz, β ist ein weiteres Gewicht, um den Einfluss der Varianz zu steuern. Mithilfe dieser Zielfunktion wird nun nicht nur versucht jeden Studenten in einen für ihn möglichst guten Kurs zuzuteilen, sondern auch die Varianz minimiert. Damit werden

nach Möglichkeit alle Studenten in einen ähnlich von ihnen präferierten Kurs verteilt, oder mit anderen Worten, es werden Ausreißer verhindert.

Nachteil bei dieser Variante ist jedoch, dass die Zielfunktion nun nicht mehr linear, sondern quadratisch ist. Zum einen erhöht das den Aufwand, um eine Lösung zu finden, enorm. Zum anderen kann mit bekannten Algorithmen zum Lösen solch quadratischer Probleme das Finden einer optimalen Lösung nicht garantiert werden. Aus diesem Grund wurde bei der Umsetzung des Verteilungsalgorithmus die zuvor gezeigte lineare Zielfunktion verwendet.

Kapitel 6

Test des Systems

Das Testen dieses Softwaresystem wurde in drei Aufgabenbereiche unterteilt. Zuerst wurde das Frontend durchgängig getestet. Anschließend wurde das Backend überprüft. Letztendlich wurde der Algorithmus mit seinen Auswirkungen auf Richtigkeit untersucht.

6.1 Frontend

Um das Frontend ordnungsgemäß zu testen, wurden verschiedene Testobjekte betrachtet. Im Folgenden werden diese erläutert.

6.1.1 Homepage

Um die Homepage zu testen, reicht es aus, den Header zu überprüfen. Dies bedeutet, dass sowohl die Textfelder ("Start", "Kursübersicht", "Meine Präferenzliste", "Ergebnis sehen") des Headers, als auch die Reihenfolge derer überprüft wird. Zusätzlich werden die Texte der Buttons "Anmelden" und "Registrieren" und die Reihenfolge dieser geprüft. Dies geschieht in dem Testfall *HomepageTest*.

6.1.2 Registrierung

Der Testfall *RegisterTest* verifiziert, dass ein Nutzer sich ordnungsgemäß registrieren kann und anschließend auch mit diesen Daten anmelden kann. Dafür klickt der automatisierte Nutzer auf den "Registrieren"-Button der Homepage und registriert sich. Anschließend ruft der User die Login-Seite auf und meldet sich an. Ein erfolgreicher Login wird durch die Abwesenheit des "Anmelden"-Buttons sicher gestellt. Zu Ende des Testfalls wird der neu registrierte Nutzer gelöscht.

6.1.3 Login

Der daran anschließende Testfall ist *LoginTest*. Dieser testet, dass ein bereits registrierter Benutzer sich anmelden kann. Dafür wird zuerst ein Nutzer in der Datenbank kreiert. Der automatische User öffnet die Login-Seite und gibt die Daten des kreierten Nutzers ein. Der erfolgreiche Login wird durch die Abwesenheit des "Anmelden"-Buttons sicher gestellt. Zu Ende des Testfalls wird der kreierte Nutzer aus der Datenbank gelöscht.

6.1.4 Kursübersicht

Manuell getestet wurde die Kursübersicht. Um die Kursübersicht zu testen, müssen zuerst Kurse im Backend erstellt werden. Diese werden anschließend im Frontend unter dem Menüpunkt "Kursübersicht" überprüft. Hierbei ist es wichtig, dass in dem Einführungstext das korrekte Jahr steht. Anschließend wird die Vorschau für die Kurse kontrolliert. Dafür wird sichergestellt, dass der Kurstitel, die Kurzbeschreibung, Ort und Zeit und ein Ausschnitt der ausführlichen Beschreibung zu sehen sind. Weiterhin muss jede Vorschau einen "Details"-Button beinhalten, der zu der kompletten Kursbeschreibung führt. Diese muss den Anforderungen entsprechen.

6.1.5 Meine Präferenzliste

Anschließend wurde der Menüpunkt "Meine Präferenzliste" überprüft. Hier müssen alle Kurse angezeigt werden, die in diesem Semester gewählt werden können. Insbesondere ist es wichtig, dass jeder Kurs einzeln per Drag and Drop verschoben werden kann. Die verschobene Liste muss nun gespeichert werden können. Dies wird verifiziert, indem die Cookies und der Cache geleert werden und anschließend die Seite neu aufgerufen wird. Nun muss die Reihenfolge, in welcher die Kurse angeordnet wurden, mit der Reihenfolge der neu geladenen Seite übereinstimmen.

6.1.6 Ergebnis sehen

Der Menüpunkt 'Ergebnis sehen' wird überprüft, indem sicher gestellt wird, dass alle Nutzer, die eine Präferenzliste abgeschickt haben, verteilt wurden.

6.2 Backend

Für das Testen des Backends

6.3 Algorithmus

Um den Algorithmus zu testen, mussten Daten generiert werden und in die Datenbank importiert werden. Dafür wurden zuerst ein R-Skript geschrieben, welches eine Präferenzenmatrix aus beliebig vielen Studenten und Kursen generiert. Mithilfe eines weiteren Skripts wurde dieses Datenset in die Datenbank importiert. Anschließend konnte die Verteilung im Backend ausgelöst werden.

Zur Evaluierung des Ergebnisses wurden die künstlichen Daten des R-Skriptes verwendet und reale Daten des Empiriepraktikums aus dem Wintersemester

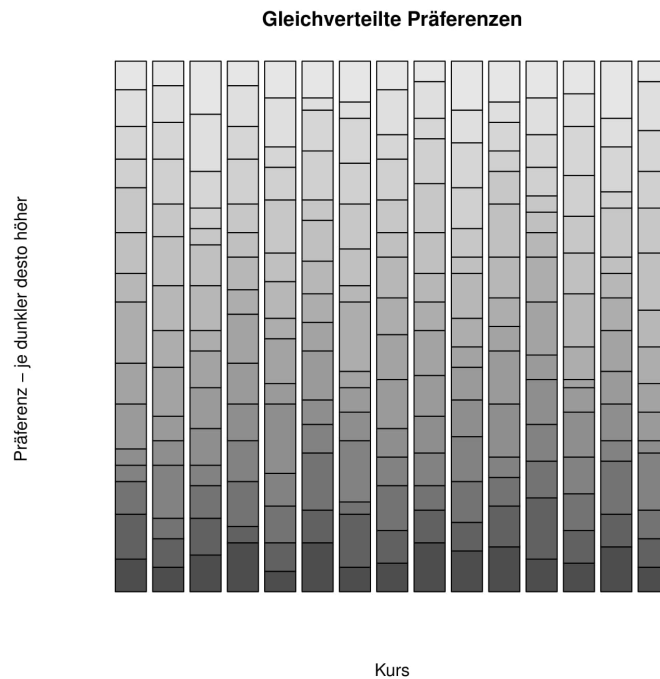


Abbildung 6.1: Gleichverteilte Präferenzen mit 130 Studenten auf 15 Kursen mit maximal 10 Teilnehmern pro Kurs. Die Dunkelheit stellt die Höhe der Präferenz dar

2017/2018. Die R-Skripts verteilen dabei einmal gleichverteilt und einmal normalverteilt Präferenzen.

6.3.1 Gleichverteilung

Zum einen generiert dies eine gleichverteilte Präferenzenmatrix, die in Abbildung 6.1 zu sehen ist.

Die Verteilung ist in dem Präferenzenhistogramm in Abbildung 6.2 dargestellt. Wie man sieht, bekommen die meisten Studenten ihre Erstwahl, das heißt den Kurs mit Präferenz 15. Nur sehr wenige Studenten bekommen ihre Zweitwahl, das heißt den Kurs mit Präferenz 14.

Der Grund hierfür ist, dass die Präferenzen auf die Kurse gleichverteilt werden. Das heißt, die Wahrscheinlichkeit, dass ein beliebiger Student einen be-

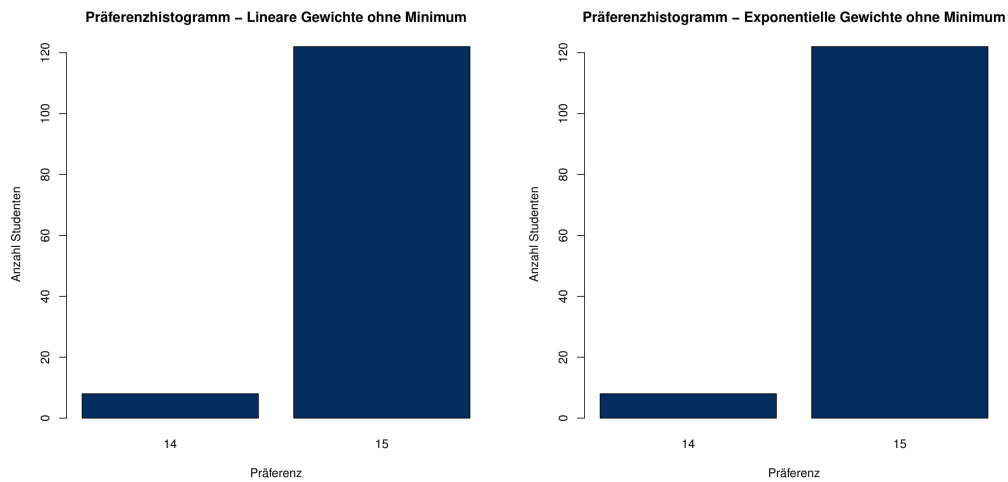


Abbildung 6.2: Gegenüberstellung der Präferenzhistogramm für lineare und exponentielle Gewichtung

liebigen Kurs mit beliebiger Präferenz wählt, ist immer gleich. Allerdings ist eine gewisse Varianz in der Ziehung der Präferenzen zu beobachten. Daher hat jeder Kurs unterschiedlich viele Studenten pro Präferenz. Daher kann man davon ausgehen, dass ein paar Studenten nur ihre Zweitwahl kriegen können.

6.3.2 Normalverteilung

Weiterhin generiert das Skript normalverteilte Präferenzen.

6.4 E-Mail Benachrichtigung

Zuletzt wurde die E-Mail-Benachrichtigung getestet. Dafür wurde der vor-konfigurierte Mailserver genutzt, der alle E-Mails abfängt. Dort konnten alle Mails überprüft werden.

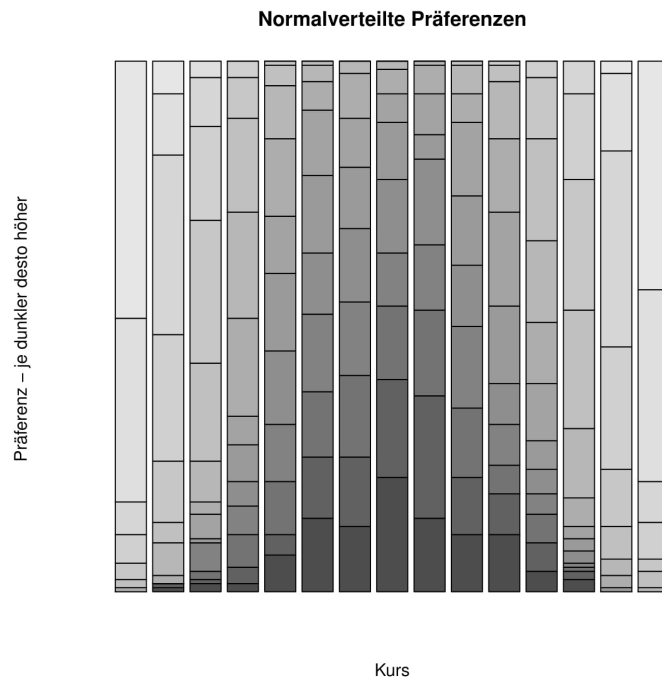


Abbildung 6.3: Normalverteilte Präferenzen mit 130 Studenten auf 15 Kursen mit maximal 10 Teilnehmern pro Kurs. Die Dunkelheit stellt die Höhe der Präferenz dar

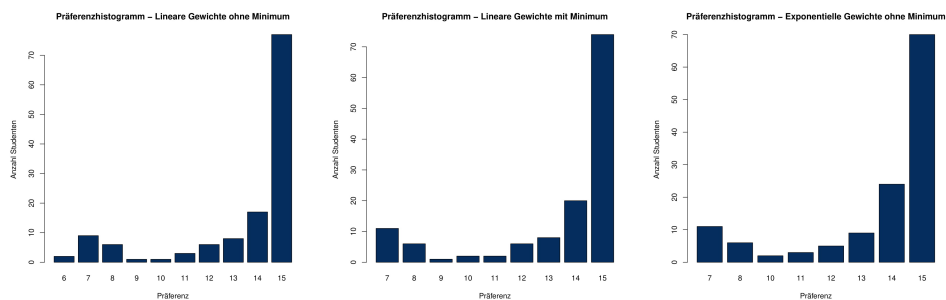


Abbildung 6.4: Gegenüberstellung der Präferenzhistogramm für lineare Gewichtung, lineare Gewichtung mit Minimum und exponentielle Gewichtung

Kapitel 7

Zusammenfassung

Zielsetzung des Projekts war es, eine Plattform zu schaffen, mithilfe der das Empiriepraktikum des Instituts für Psychologie verwaltet werden kann. Hierbei sollte vor allem die Möglichkeit der automatischen Verteilung der Studenten anhand von Präferenzlisten umgesetzt werden. Diese und weitere Anforderungen wurden im Rahmen dieser Arbeit herausgearbeitet. Vor allem die Aufteilung in die verschiedenen Rollen Studenten, Dozenten und Administratoren mit unterschiedlichen Sichten auf das System waren ein wichtiger Anspruch. Basierend auf diesen Anforderungen wurde ein Entwurf für das System entwickelt und vorgestellt. Dieser umfasste neben Mockups für die wichtigsten Seiten der Plattform die Aufteilung in ein Front- und Backend mit dazugehörigem Datenmodell. Anschließend konnte die Umsetzung der Entwürfe erfolgen. Große Teile wurden dabei mit dem Framework *October CMS* realisiert. Nachdem der Funktionsrahmen der Plattform geschaffen war, konnte der Verteilungsalgorithmus entwickelt werden. Eine geeignete lineare Zielfunktion wurde aufgestellt und eine mögliche Wahl der Parameter und weitere Varianten der Zielfunktion diskutiert.

Im Anschluss wurden die Tests der Plattform zur Einschreibung und Verwaltung und des Verteilungsalgorithmus betrachtet. Anhand der Tests konnte gezeigt werden, dass alle Kernfunktionen des Systems wie gewünscht funktionieren. Der Verteilungsalgorithmus erreichte auf künstlich generierten Da-

tensätzen gute Verteilungen. Im Vergleich mit dem Algorithmus der alten Lösung, konnten ein deutlich besseres Ergebnis der Verteilung erzielt werden.

Es konnten jedoch nicht alle Anforderungen an das System umgesetzt werden. So wurde die mögliche Tauschbörse nicht implementiert. Des Weiteren wird die im Datenmodell angelegte Unterstützung für mehrere verschiedene Module zwar im Backend unterstützt, im Frontend ist es jedoch nicht möglich Präferenzlisten für mehre Module parallel zu verwalten. Auch konnten einige der im Entwurf angedachten Elemente nicht umgesetzt werden, da durch *October CMS* bereits Teile vorgegeben waren.

Kapitel 8

Anhang