



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

Dokumentation

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

9. Februar 2018

Eigenständigkeitserklärung

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

Jena, 9. Februar 2018

Unterschriften einfügen

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	7
2.3	Technische Details	8
3	Entwurf	10
3.1	Datenmodell	10
3.2	Design	12
3.2.1	Frontend	12
3.2.2	Backend	16
4	Verteilungsalgorithmus	20
4.1	Aufstellen Zielfunktion	20
4.2	Erweiterung der Zielfunktion und Wahl der Parameter	21
4.2.1	Lineare Gewichte	21
4.2.2	Exponentielle Gewichte	22
4.2.3	Festes Minimum	22
4.2.4	Betrachtung der Varianz	23

5 Umsetzung	25
5.1 Verwendete Software	25
5.2 Frontend	26
5.3 Backend	29
5.4 Algorithmus	35
6 Test des Systems	36
6.1 Frontend	36
6.1.1 Startseite	36
6.1.2 Registrierung	37
6.1.3 Login	37
6.1.4 Kursübersicht	37
6.1.5 Meine Präferenzliste	38
6.1.6 Ergebnis sehen	38
6.2 Backend	38
6.2.1 Anlegen neuer Dozenten und Administratoren	38
6.2.2 Erstellen von Lehrstühlen, Modulen und Kursen	39
6.3 Verteilungsalgorithmus	39
6.3.1 Gleichverteilung	40
6.3.2 Normalverteilung	41
6.3.3 Realdaten	43
6.3.4 Ergebnisse des Algorithmus	43
7 Zusammenfassung	46

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 100 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 folglich 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 welchem Kurs teilnehmen möchte. Es liegt auf der Hand, dass diese Listen nicht alle manuell ausgewertet werden können. Daher muss die Verteilung automatisch erfolgen. Für diese Aufgabe existieren bereits Lösungen. So können mit Friedolin, dem Studienverwaltungstool der FSU Jena, bereits Präferenzlisten erstellt und ausgewertet werden. Bei der Verteilung mithilfe von Friedolin kommt es jedoch zu Problemen. Zum einen kann Friedolin durch unvoll-

ständige Präferenzvergabe dazu gezwungen werden eine hohe Präferenz zu vergeben. Zum anderen werden bei Friedolin immer den gleichen Studenten eines Jahrgangs niedrige Präferenzen zugewiesen. 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 das Ergebnis muss manuell 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. Die verantwortlichen Mitarbeiter des Instituts für Psychologie sollen dort für ein kommendes Semester ein neues Praktiumsmodul 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 der vorher festzulegenden Einschreibungsperiode, sollen die Studenten mit einem Klick 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. Im Anschluss wird der Algorithmus zur Verteilung der Studenten formal beschrieben. Danach folgt eine Beschreibung der tatsächlichen Umsetzung des Systems. Hiernach werden die durchgeführten Tests zu Plattform und Ver-

teilungsalgorithmus vorgestellt. Abschließend wird die Arbeit nochmals kurz zusammengefasst und ein Ausblick auf weitere umzusetzende Funktionalitäten gegeben.

Kapitel 2

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an die Plattform dargestellt. Zunächst wird in 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 System beschrieben.

2.1 Anforderungen an das System

Gemäß der allgemeinen Aufgabenstellung aus Abschnitt 1.2 existieren verschiedene Sichtweisen auf das Systems. Zum einen die Sicht der Verantwortlichen für Praktikum und Kurse, zum anderen die der Studenten. Erstere teilt sich wiederum in den Blickwinkel der Dozenten der Kurse und der übergeordneten Verantwortlichen für das Empiriepraktikum, im Weiteren Administratoren genannt, auf. Im Folgenden werden zunächst die Anforderungen für Studenten, Dozenten und Administratoren anhand des chronologischen Ablaufs eines Praktikumsmoduls dargestellt. Anschließend wird nochmals 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 optionale Funktionalitäten. Es ist zu beachten, dass aus zeitlichen Gründen nicht alle Anforderungen im Rahmen des Projektes umgesetzt werden konnten. Sie werden der Vollständigkeit halber dennoch aufgeführt.

2.1.1 Ablauf eines Praktikumsmoduls

Ein Praktikum beginnt, indem die Administratoren, nachdem sie sich in einer 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, das 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 Änderung soll die Präferenzliste des Student gespeichert werden. Nach Ablauf der zuvor von den Administratoren festgelegten Frist können die Studenten anhand ihrer aktuellen Version der Präferenzliste auf die Kurse automatisch verteilt werden. Diese Verteilung sollte möglichst gut sein, d.h. in Fall dieses Projekts eine möglichst gleichmäßige Verteilung der Studenten mit vorzugsweise geringer Varianz. Nach dieser Verteilung soll eine Nachbearbeitungsphase folgen, in der die Administratoren das Ergebnis der Verteilung

bei Bedarf manuell ändern können. Nachdem das Ergebnis der Verteilung fixiert wurde, sollen alle beteiligten Studenten, Dozenten und Administratoren über das Ergebnis der Verteilung mittels einer E-Mail-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 soll die Möglichkeit haben, Kurse für das aktuelle Praktikum zu erstellen und sich seine Kurse älterer 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, eine Kurzbeschreibung des Kurses, eine ausführliche Beschreibung und optional

eine Literaturliste. Diese Angaben sind auch für Studenten einsehbar. Dabei kann die Teilnehmerzahl nur auf fünf oder zehn gesetzt werden. Nicht für Studenten einsehbar sind die 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.

Sicht der Studenten

Die Studenten sollen ohne 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 E-Mail 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.

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 nun die Möglichkeit die maximale Teilnehmeranzahl von Kursen zu erhöhen. Die Eingabe des Algorithmus sind die Gewichte der Präferenzen.

Startet der Algorithmus, so versucht er, für jeden Studenten die größte, mög-

liche 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 allerdings nie mehr Teilnehmer einem Kurs zugeordnet als die maximale Teilnehmeranzahl. Hat der Algorithmus schließlich eine passende Zuordnung von Studenten zu Kursen gefunden, so kann der Administrator 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

Besucher der Website können sowohl auf mobilen Endgeräten, als auch per Desktop die Website besuchen. Diese passt sich dynamisch an die Bildschirmauflösung an.

Benutzt ein Besucher jedoch einen Browser, der älter als Internet Explorer 11 ist, so wird ihm mitgeteilt, er solle seinen Browser aktualisieren.

Der Login wird über eine E-Mail/Passwort Authentifizierung realisiert. Dafür sind nur E-Mail-Adressen der Friedrich-Schiller-Universität Jena nutzbar. Dies bedeutet, dass E-Mails auf „@uni-jena.de“ enden müssen. Die An- beziehungsweise Abmeldung ist hierbei zu jeder Zeit möglich.

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 parametrisiert werden. Optionen können unter anderem die Gewichtung der Varianz, oder welche Kurse belegt werden können, sein. Weiterhin muss der Verteilungsal-

gorithmus innerhalb von 24 Stunden fertig sein.

Zu Zwecken der Portabilität wird das Produkt in einer *docker-compose* Umgebung ausgeführt. In dieser Umgebung soll die Website auf *October CMS* mit *Laravel 5.5* aufsetzen. Für den Webserver ist *nginx* zu nutzen. Die Datenbank muss auf *MySQL* basieren. Zuletzt soll ein Cache-Server verwendet werden. Vorzugsweise ist dies *Redis*.

Die wesentlichsten Bestandteile der Website müssen über Tests validiert werden. Insbesondere der Verteilungsalgorithmus muss getestet werden.

Kapitel 3

Entwurf

Nachdem im vorangegangenem 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 von uns modellierten Tabellen entsprechend der Vorgaben von OctoberCMS für Plugins benannt: `author_plugin_table`. In unserem Fall wird dies auf `bwolfjena_core_table` abgebildet. Die beiden Tabellen `backend_users` und `users` werden vom CMS bereitgestellt.

Die Tabelle `bwolfjena_core_modules` beinhaltet die verschiedenen Empiriepraktika. Der Name Module für die Tabelle wurde gewählt, damit das System bei Bedarf auch auf weitere Module ausgeweitet werden kann.

In der Tabelle `bwolfjena_core_courses` sind die verschiedenen Kurse gespeichert. Ein Kurs gehört immer zu einem bestimmten Modul beziehungsweise einem Empiriepraktikum (z. Bsp. Emperiepraktikum 17/18). Daher wird für jeden Kurs mittels eines Fremdschlüssels das Modul, zu dem er gehört, gespeichert.

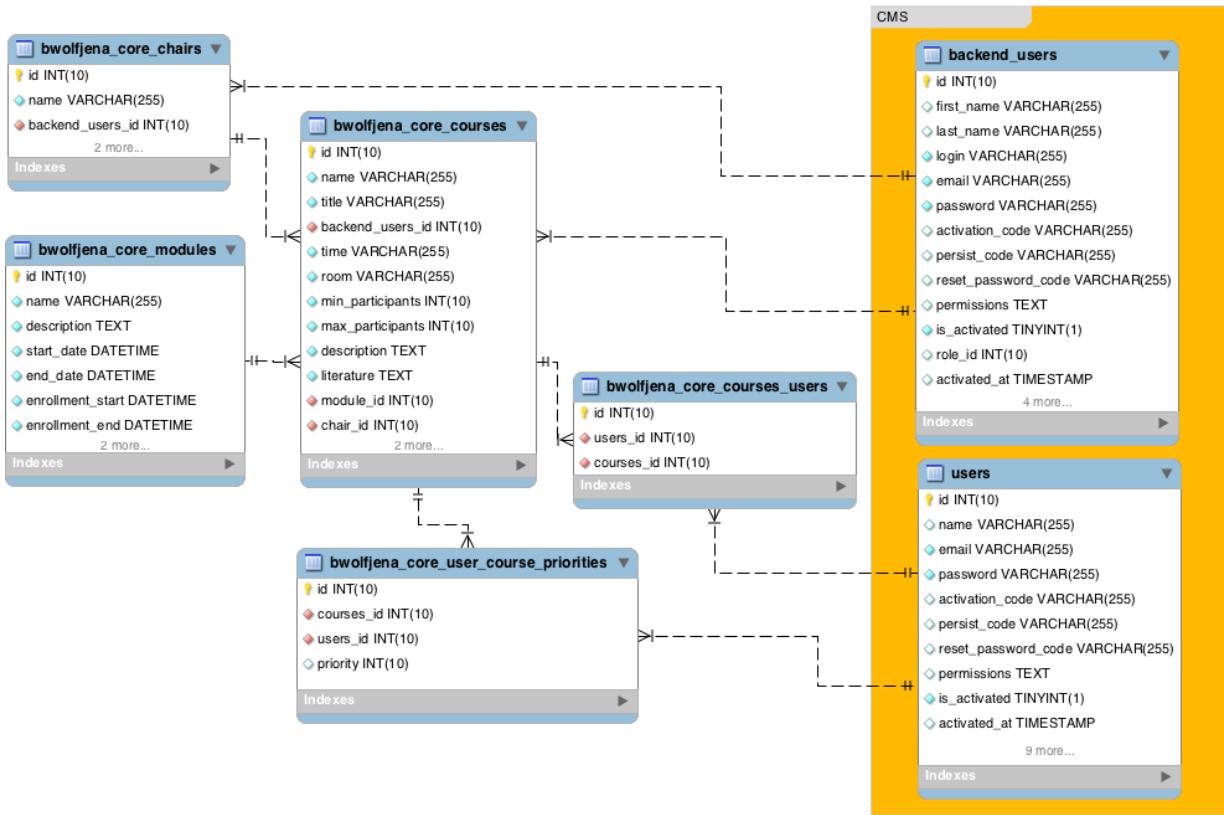


Abbildung 3.1: Entwurf des Datenmodells. Farbcode für Spalten der Tabellen: Gelb Primärschlüssel, Rot Fremdschlüssel, Blau gewöhnliche Variable, Weiß nullable-Variable

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 der Studenten hat eine Präferenzliste. Die Einträge der Präferenzliste für jeden Studenten und Kurs werden in **bwolfjena_core_user_course_priorities** gespeichert. Dementsprechend beinhaltet sie Fremdschlüssel auf **users** und **bwolfjena_core_courses**.

In der Tabelle **bwolfjena_core_courses_users** ist abgelegt, welcher Student welchem Kurs zugeteilt wurde. Aus diesem Grund beinhaltet sie 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 die Spalte `id` als Primärschlüssel und die Zeitstempel `created_at` und `updated_at` zur Protokollierung. 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 Frontend, also aus Sicht der Studenten, und für das Backend, die Sicht der Dozenten und Administratoren, vorgestellt. Dabei gilt der in Tabelle 3.1 angegebene Farbcode für die verschiedenen Elemente der Mock-Ups.

	Eingabefeld
	Button
	Drag&Drop-Element
	Textfeld
	Optisches Element

Tabelle 3.1: Farbcode der Mockups

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

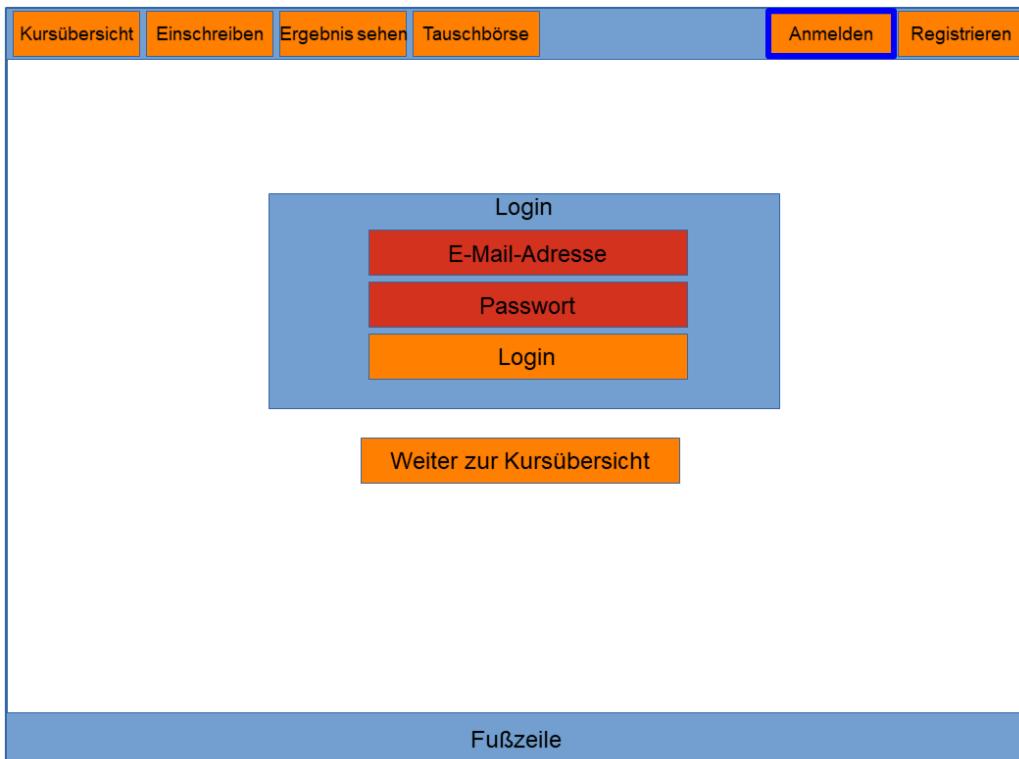


Abbildung 3.2: Entwurf für die Login-Oberfläche

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 erfolgreiche 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. Sobald

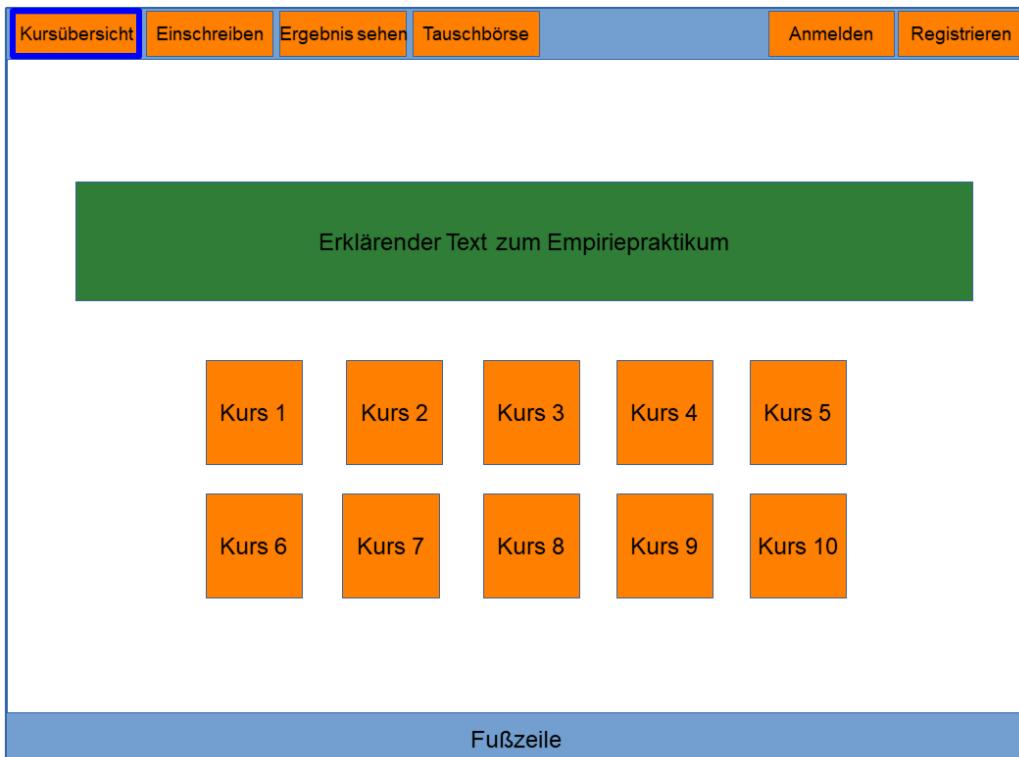


Abbildung 3.3: Entwurf für die Kursübersicht

man sich angemeldet hat, entfällt jedoch der Registrieren-Knopf und Anmelden wird durch Abmelden ersetzt. Die Fußzeile soll weitere allgemeine Informationen bereitstellen, sofern diese von Nöten sein sollten, aber vor allem als optischer Abschluss der Seite dienen. Auch sie wird auf alle folgenden Seiten übernommen.

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

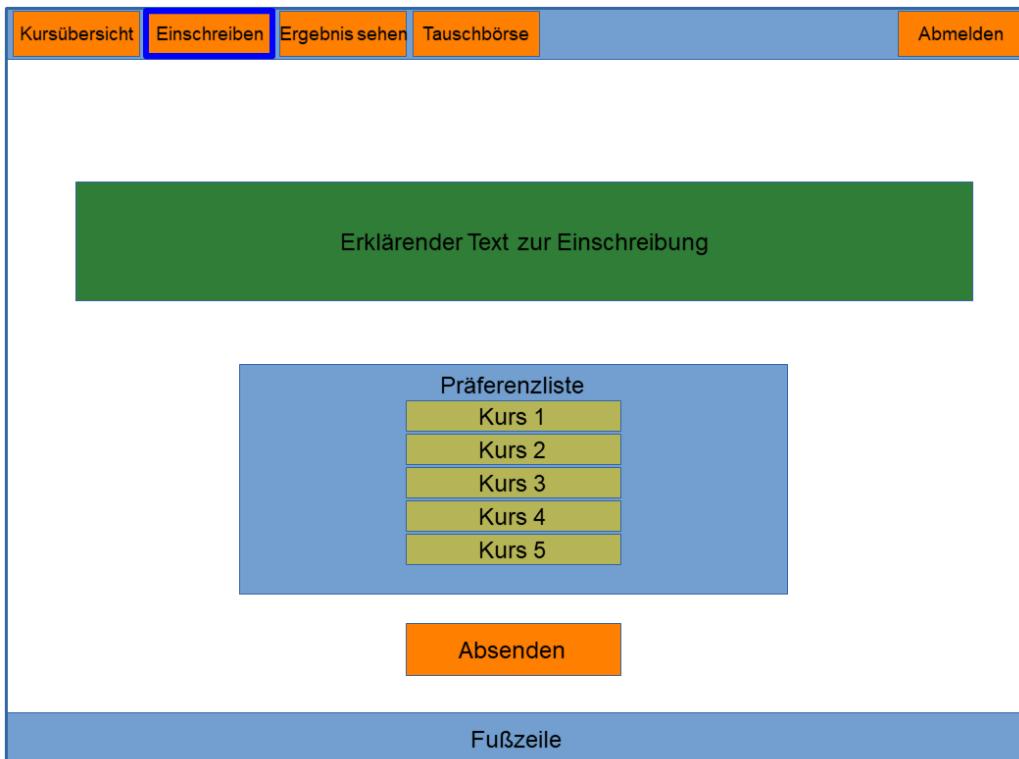


Abbildung 3.4: Entwurf für die Einschreibungs-Oberfläche

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.

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 benutztten 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.

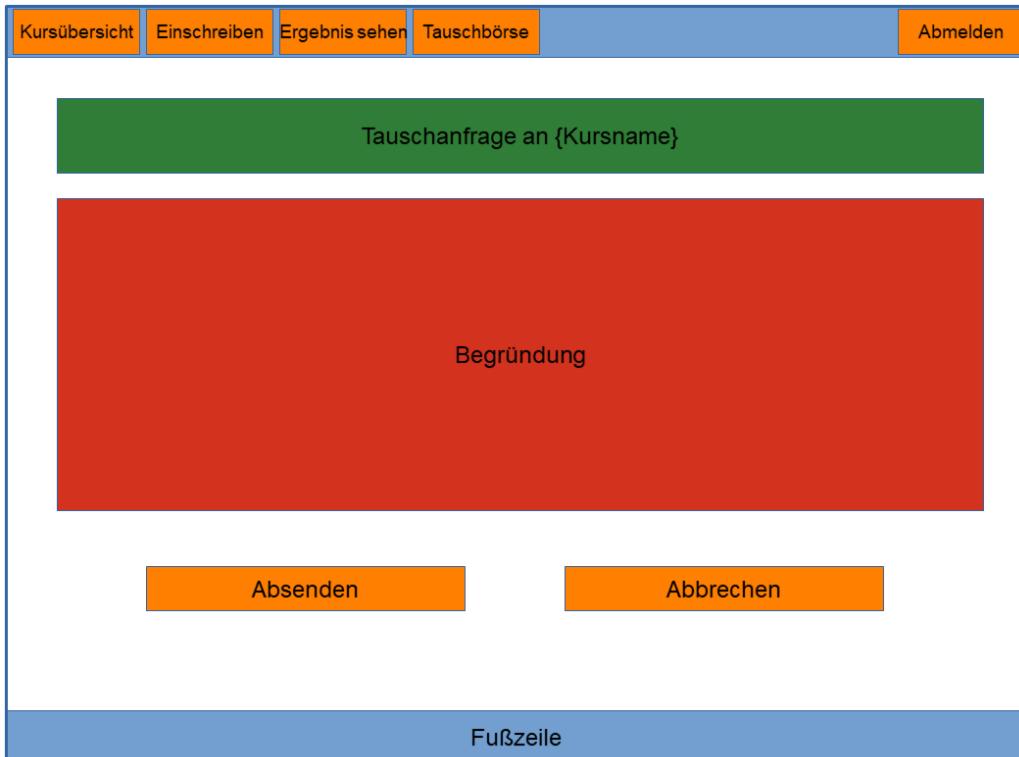


Abbildung 3.5: Entwurf für die Tauschanfrage

Durch einen Klick auf den Kurs in den gewechselt werden möchte, öffnet sich die in Abbildung 3.5 gezeigt 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 Ad-

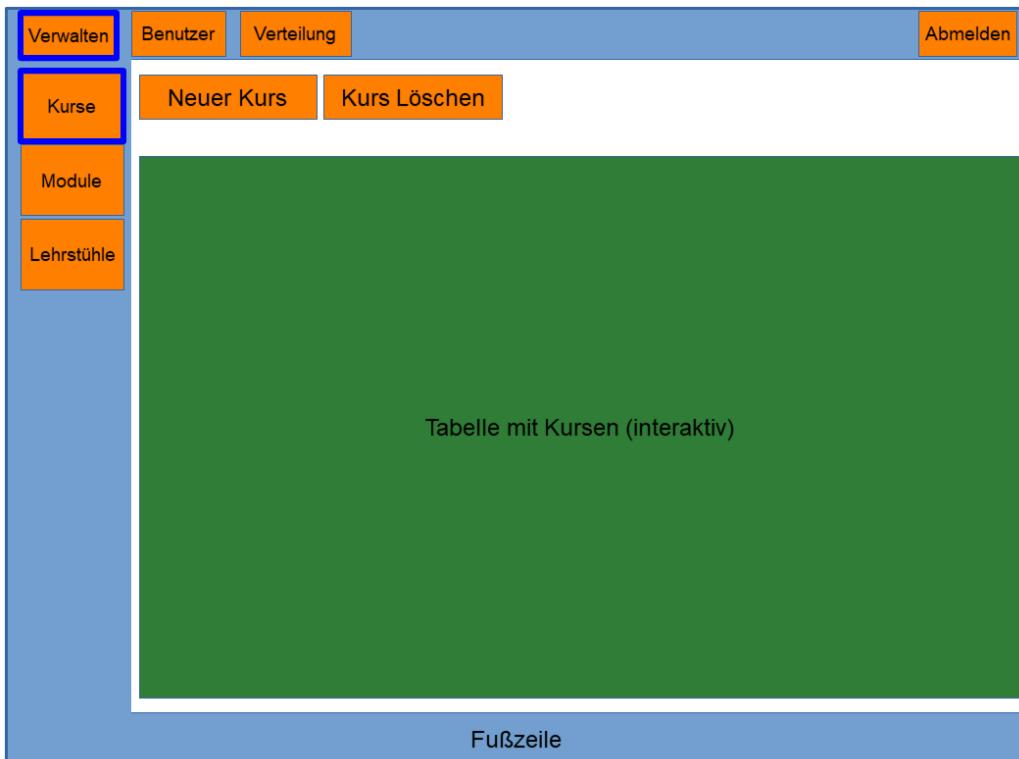


Abbildung 3.6: Entwurf für das Verwalten der Kurse

ministratoren den Verteilungsalgorithmus starten und sich die Ergebnisse 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 Textfel-

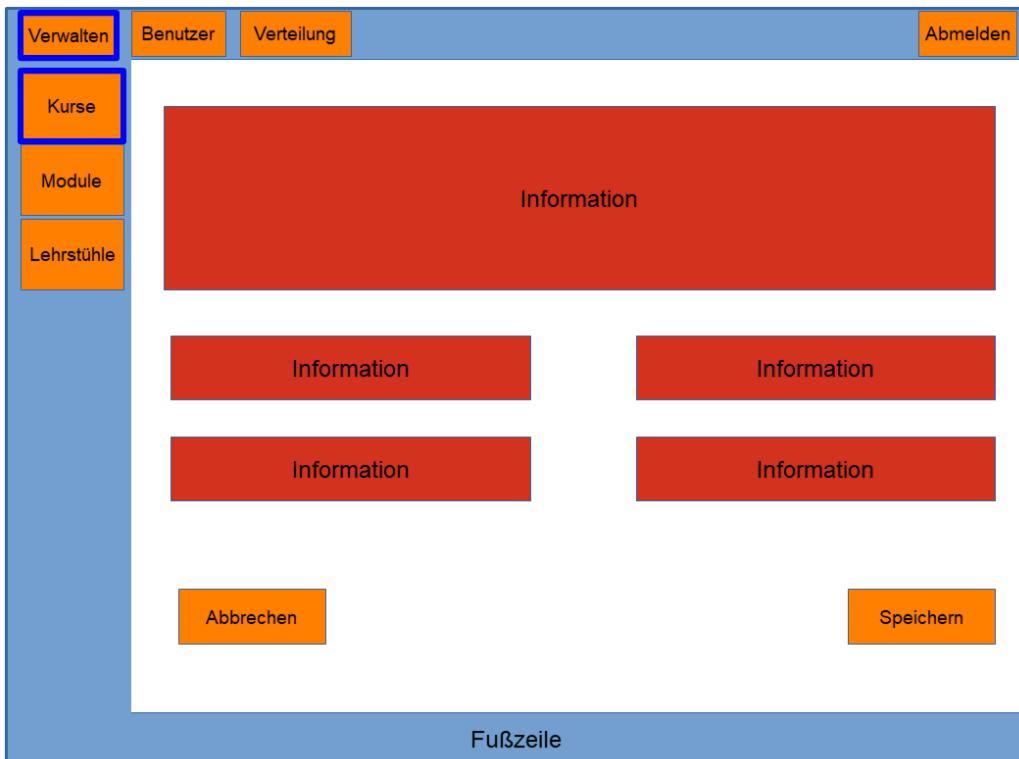


Abbildung 3.7: Entwurf für das editieren eines Kurses

dern 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 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 wer-

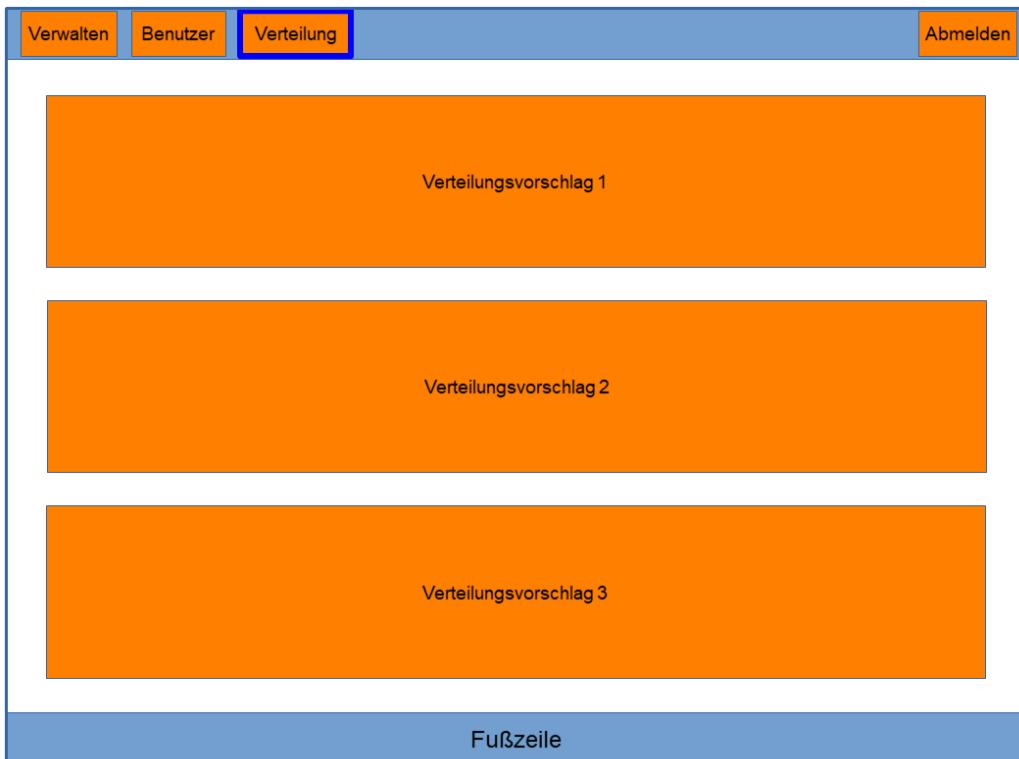


Abbildung 3.8: Entwurf für das Anzeigen der Verteilungsvorschläge

den. 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.

Kapitel 4

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. Darauffolgend wird auf die Einbindung in das Backend eingegangen.

4.1 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 zugewiesen, 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 gleichbedeutend mit der Aussage, dass jeder Student genau einem Kurs zugewiesen wurde. Daraus ergibt sich die Zielfunktion zu

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

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 zugewiesen 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 .

4.2 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 werden können bzw. wie die Zielfunktion weiter angepasst und erweitert werden kann.

4.2.1 Lineare Gewichte

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,

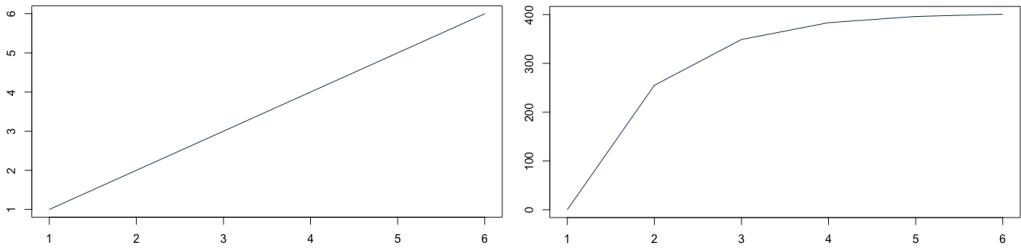


Abbildung 4.1: Gegenüberstellung von linearer Gewichtung (links) und exponentieller Gewichtung (rechts). Die horizontale Achse bezeichnet die Präferenzen, die vertikale den Wert des Gewichts

dass die Summe maximal wird, wird so eine Zuordnung eines Studenten zu einem Kurs, für den er eine höhere Präferenz angegeben hat, gegenüber einem 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.

4.2.2 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 4.1 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.

4.2.3 Festes Minimum

Die Gewichte können bei dieser Erweiterung der Zielfunktion weiterhin frei gewählt werden. So sind zum Beispiel die beiden Ansätze aus dem vorheri-

gen 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.

4.2.4 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 zu minimieren. 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 5

Umsetzung

In diesem Kapitel wird die Umsetzung des System erläutert. Zunächst wird die zur Implementierung verwendete Software näher erläutert. Anschließend werden Front- und Backend kurz vorgestellt und mit dem Entwurf aus Kapitel 3 verglichen. Zur Versionsverwaltung der verwendeten Komponenten wurde die GitHub Organisation BwolfJena¹ erstellt.

5.1 Verwendete Software

Für die lokale Entwicklung wurde als Basis *Laradock*² verwendet und auf die benötigte Software angepasst. Diese modifizierte *Laradock* Variante wurde als *git submodule* in das Hauptrepository *BWolf* eingebunden. Als Webserver kommt lokal *nginx*³ zum Einsatz, wobei die *PHP*⁴ 7.1 Skripte von *php-fpm*⁵ ausgeführt werden. In der Produktionsumgebung wird *Apache* verwendet, weitere Details sind nicht bekannt. Für die Datenbank kommt in allen Umgebungen *MySQL*⁶ zum Einsatz. Um auch lokal E-Mails verschicken und testen zu können, wird *MailDev* als *SMTP* Server verwendet. Für den Ver-

¹<https://github.com/BWolfJena>

²<http://laradock.io/>

³<https://nginx.org/en/>

⁴<http://php.net/>

⁵<https://php-fpm.org/>

⁶<https://www.mysql.com/de/>

teilungsalgorithmus wird *Node.js* ⁷ eingesetzt.

Als Entwicklungsframework für *PHP* wird *October CMS*⁸ (basierend auf *Laravel*⁹ 5.5.) verwendet. Ein Großteil der Funktionalitäten, wie die Front- und Backendbenutzerverwaltung und die Inhaltsverwaltung werden dabei von *October CMS* bereitgestellt. Die Verwaltungsansichten werden basierend auf *yaml* Dateien generiert. Im Frontend kommt *Semantic UI*¹⁰ 2.2 zum Einsatz, wodurch die Webseite ohne größere Anpassungen auf Mobilgeräten komfortabel benutzt werden kann. Schließlich dient das *Express*¹¹ *Framework* als Basis für den Algorithmus Mini-HTTP Server (weitere Details in Kapitel 4).

5.2 Frontend

Die Entwürfe für das Frontend wurden ohne große Änderungen umgesetzt. In Abbildung 5.1 sind der Entwurf für die Login-Oberfläche und die Umsetzung selbiger gegenüber gestellt. Es fällt auf, dass die grobe Struktur der Seite bis auf kleine Änderungen mit dem Entwurf übereinstimmt. Unterschiede lassen sich vor allem in der Kopfzeile erkennen. So wurde der Reiter *Start* hinzugefügt. Anders als zuvor angedacht, fiel die Entscheidung für eine Startseite, in der das Empiriepraktikum kurz vorgestellt wird. Des Weiteren wurde der Reiter *Einschreiben* aus Gründen der Verständlichkeit in *Meine Präferenzliste* umbenannt. Die Seite erfüllt jedoch die selbe Funktion.

Abbildung 5.2 zeigt Entwurf und Umsetzung der Kursübersicht. Die Struktur des Entwurfs wurde direkt umgesetzt. Wie in Kapitel 2 beschrieben, ist es, wie in Abbildung 5.2 zu sehen ist, möglich auch ohne eine Anmeldung auf die Kursübersicht zuzugreifen. Es ist anzumerken, dass die Fußleiste auch in der Kursübersicht 5.2 den Abschluss der Seite bildet und lediglich durch die Menge an Kursen nicht zu sehen ist.

Die Oberfläche zum Erstellen der Präferenzliste wurde ebenfalls wie ange- dacht umgesetzt. Mithilfe von Drag&Drop könne die verschiedenen Kurse

⁷<https://nodejs.org/en/>

⁸<https://octobercms.com/>

⁹<https://laravel.com/>

¹⁰<https://semantic-ui.com/>

¹¹<http://expressjs.com/de/>

The figure shows a comparison between the design of a login interface and its actual implementation.

Design (Top):

- A horizontal navigation bar at the top with buttons: Kursübersicht, Einschreiben, Ergebnis sehen, Tauschbörse, Anmelden (highlighted in blue), and Registrieren.
- The main area contains a "Login" form with fields for E-Mail-Adresse and Passwort, and a Login button.
- A "Weiter zur Kursübersicht" button is located below the form.
- A blue footer bar labeled Fußzeile.

Implementation (Bottom):

- A horizontal navigation bar at the top with buttons: Start, Kursübersicht, Meine Präferenzliste, Ergebnis sehen, Archiv, Anmelden (highlighted in blue), and Registrieren.
- A message box in the center says: Sie haben sich erfolgreich ausgeloggt!
- The login form is identical to the design, with fields for E-Mail and Passwort, and a "Passwort vergessen?" link.
- Buttons for Anmelden and Zur Kursübersicht are present.
- A dark blue footer bar containing links: Kontakt, Impressum, Hilfe, Sitemap, Copyright, and BWolfJena 2017.

Abbildung 5.1: Gegenüberstellung von Entwurf der Login-Oberfläche und Umsetzung

Abbildung 5.2: Gegenüberstellung von Entwurf der Kursübersicht und Umsetzung

in die gewünschte Reihenfolge gebracht werden. Der Knopf *Absenden* wurde in *Speichern* umbenannt. Dadurch soll mehr Klarheit darüber geschaffen werden, dass die Präferenzliste bis zum Ablauf der Frist jederzeit verändert werden kann.

Es ist zu erwähnen, dass das gesamte Frontend auch auf mobilen Geräten unterstützt wird. Die Ansichten für Kursübersicht und auch die Wahl der Präferenzliste werden entsprechend der Größe des Bildschirms angepasst, so dass die Benutzerfreundlichkeit erhalten bleibt.

Die Tauschbörse wurde nicht implementiert. Grund hierfür ist zum einen die zeitliche Beschränkung des Projekts, wodurch einige optionale Funktionalitäten nicht umgesetzt werden konnten. Zum anderen sind bei einem weiteren Gespräch nach dem Erstellen der Anforderungen Zweifel an der Notwendigkeit solch einer Tauschbörse aufgetreten. Eine weitere Funktion, die nicht im Frontend umgesetzt werden konnte, war die in Kapitel 3 geplante Erweiterbarkeit auf andere Module. So kann ein Benutzer im Frontend nicht mehrere Präferenzlisten parallel verwalten.

Im Frontend wurde jedoch auch eine zusätzliche Funktionalität implementiert. So wurde ein Archiv umgesetzt, in dem die Kurse vergangener Empiriepraktika einsehbar sind.

5.3 Backend

Auch im Backend ist die Struktur der Entwürfe weitestgehend übernommen worden.

Abbildung 5.4 zeigt Entwurf und Umsetzung der Kursverwaltung. Die Struktur aus Kopf- und Seitenleiste wurde übernommen. Die Fußzeile wurde jedoch entfernt. Es fällt außerdem auf, dass die Kopfzeile mehr Reiter umfasst, als zunächst geplant. Das zum Erstellen des Backend verwendete Framework *October CMS* stellt einige vordefinierte Seiten zur Verfügung. Da von Beginn an geplant war für die entsprechenden Funktionen ein CMS zu verwenden wur-

The screenshot displays a web-based course registration system. At the top, there is a navigation bar with buttons for 'Kursübersicht' (Course Overview), 'Einschreiben' (Register), 'Ergebnis sehen' (View Results), 'Tauschbörse' (Exchange Board), and 'Abmelden' (Logout). The 'Einschreiben' button is highlighted with a blue border.

The main content area consists of two stacked sections. The top section has a green header bar containing the text 'Erklärender Text zur Einschreibung'. Below this is a blue rectangular box labeled 'Präferenzliste' (Preference List) which contains five yellow boxes, each labeled 'Kurs 1', 'Kurs 2', 'Kurs 3', 'Kurs 4', and 'Kurs 5'. At the bottom of this section is an orange 'Absenden' (Send) button. The bottom section has a blue footer bar labeled 'Fußzeile' (Footnote).

Below the main content area is a dark blue footer bar with a small logo of a stylized animal head on the left. The menu items in the footer are 'Start', 'Kursübersicht', 'Meine Präferenzliste' (which is underlined), 'Ergebnis sehen', 'Archiv', and 'Abmelden'.

The middle section of the screenshot shows a list of courses with their respective point values:

- 15 Kurs Kreysa
- 14 Kurs Koranyi
- 13 Kurs Hechler
- 12 Kurs Urschler
- 11 Kurs Buttelmann
- 10 Kurs Kaufmann 1
- 9 Kurs Ambrus 1
- 8 Kurs Amado 2
- 7 Kurs Sonntag
- 6 Kurs Ambrus 2
- 5 Kurs Schreckenbach
- 4 Kurs Grigutsch
- 3 Kurs Amado 1
- 2 Kurs Kaufmann 2
- 1 Kurs Wulf

At the bottom of this list is a dark blue 'Speichern' (Save) button.

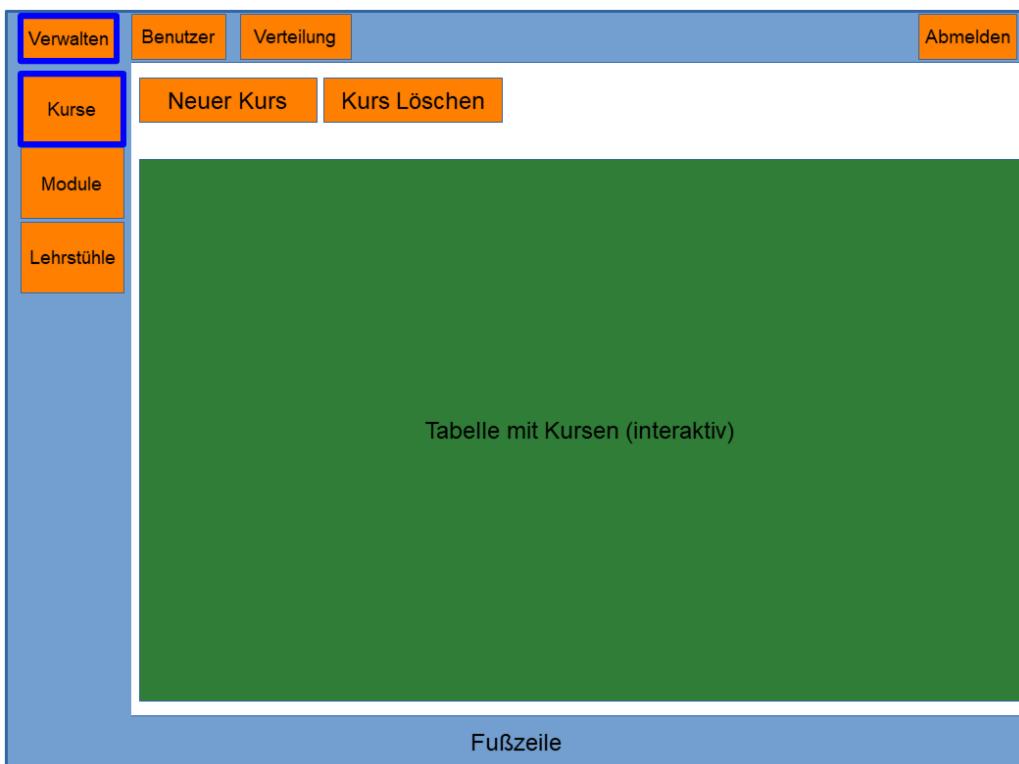
The bottom-most section is a dark blue footer bar containing links to 'Kontakt' (Contact), 'Impressum' (Imprint), 'Hilfe' (Help), 'Sitemap', 'Copyright' (with the text 'BWolfJena 2017'), and 'Abmelden'.

Abbildung 5.3: Gegenüberstellung von Entwurf der Einschreibungs-Oberfläche und Umsetzung 30

den dafür keine Mockups erstellt. Darunter fällt das *Dashboard*, in dem zum Start des Backends einige wichtige Informationen angezeigt werden. Unter dem Reiter *Seiten* können weitere Seiten für das Frontend erstellt und die vorhanden Seiten bearbeitet werden. Der Reiter *Einstellungen* ist ebenfalls von *October CMS* bereitgestellt und erlaubt das Ändern und Verwalten verschiedenster Optionen. Für dieses Projekt relevant sind vor allem die Einstellungen bezüglich der automatisch versendeten E-Mails, sowie das Verwalten der Backendbenutzer. Anders als im Entwurf vorgesehen, werden nicht alle Benutzer über den Reiter *Benutzer* verwaltet, sondern nur die Frontendnutzer. Diese Änderung ist durch *October CMS* bedingt.

Das Erstellen und Bearbeiten von Kursen, Lehrstühlen, Benutzern, usw. ist, wie in Abbildung 5.5 zu sehen, wie im Entwurf umgesetzt. Verschiedene Eingabefelder mit *Richtext*-Bearbeitung und Auswahlfelder werden wie vorgesehen zur Verfügung gestellt.

Auch der Entwurf der Verteilungsansicht wurde umgesetzt. In Abbildung 5.6 sind wieder Mockup und Resultat gegenübergestellt.



Lehrstühle: Alle									
ID	NAME	DOZENT	WEITERE DOZENTEN	ZEIT	ORT	MIN. TEILN.	MAX. TEILN.	MODUL	LEHRSTUHL
15	Kurs Kreysa	Admin Person		Do 8-10	Tiergarten	0	10	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie II
14	Kurs Amado 2	Admin Person		Do 8-10	Tiergarten	0	5	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
13	Kurs Amado 1	Admin Person		Do 8-10	Tiergarten	0	0	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
12	Kurs Ambrus 2	Admin Person		Do 8-10	Tiergarten	0	0	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
11	Kurs Ambrus 1	Admin Person		Do 8-10	Tiergarten	0	0	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
10	Kurs Urschler	Admin Person		Do 8-10	Tiergarten	0	10	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
9	Kurs Hechler	Admin Person		Do 8-10	Tiergarten	0	10	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
8	Kurs Kaufmann 2	Admin Person		Do 8-10	Tiergarten	0	10	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
7	Kurs Kaufmann 1	Admin Person		Do 8-10	Tiergarten	0	10	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
6	Kurs Butzelmann	Admin Person		Do 8-10	Tiergarten	0	10	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
5	Kurs Wulf	Admin Person		Do 8-10	Tiergarten	0	5	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
4	Kurs Sonntag	Admin Person		Do 8-10	Tiergarten	0	5	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
3	Kurs Schredenbach	Admin Person		Mi 12-14	CSZ 3	0	10	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
2	Kurs Koranyi	Dr. Nicolas Koranyi				3	10	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I
1	Kurs Grigutsch	Admin Person		Di 12-10	Bla gebaude	0	10	Empiriepraktikum 2017/2018	Lehrstuhl Psychologie I

Abbildung 5.4: Gegenüberstellung von Entwurf der Kursverwaltung und Umsetzung

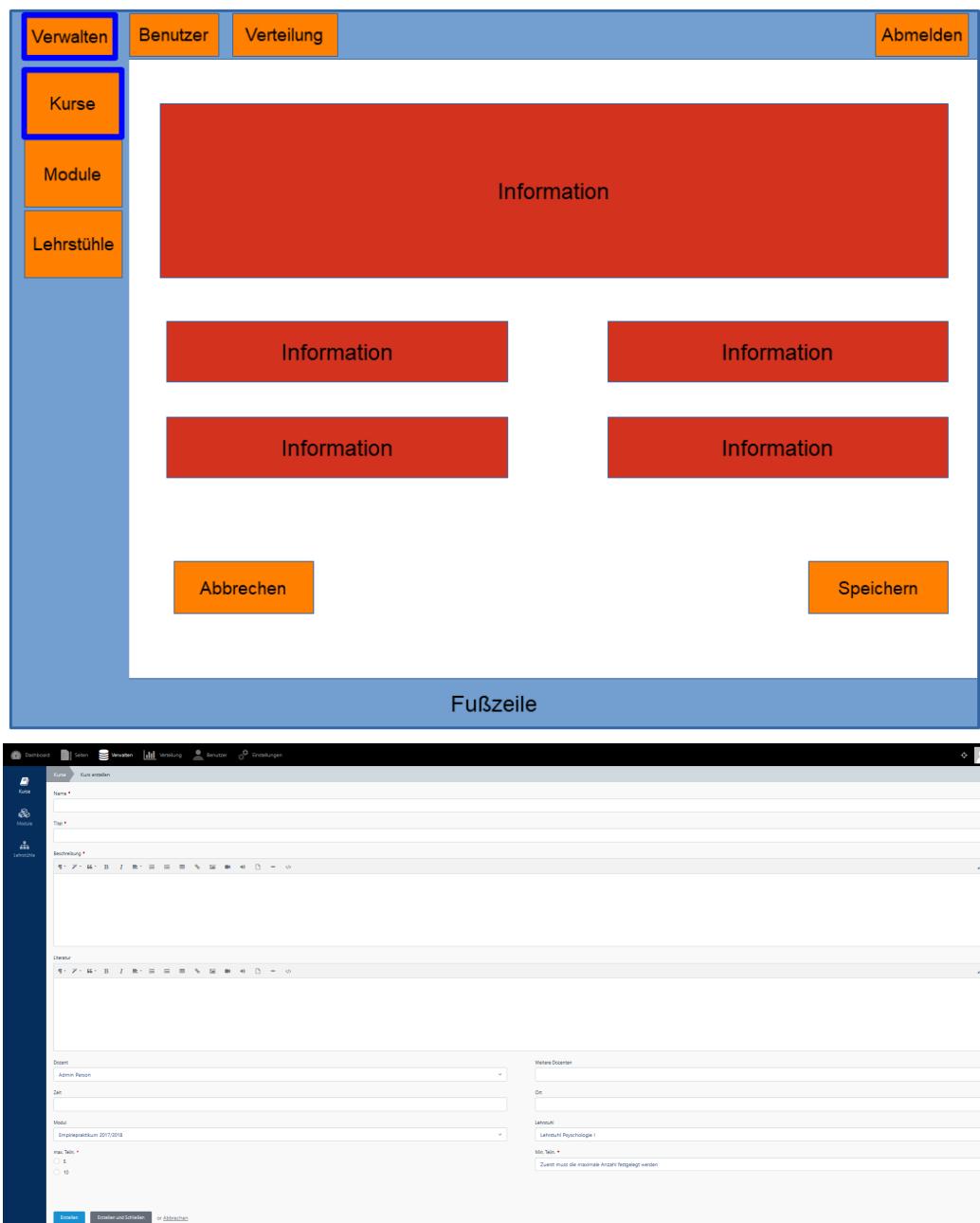


Abbildung 5.5: Gegenüberstellung von Entwurf der Kurserstellung und Umsetzung



Abbildung 5.6: Gegenüberstellung von Entwurf der Anzeige der Verteilungsvorschläge und Umsetzung

5.4 Algorithmus

Da lange unbekannt war, ob die für den Algorithmus verwendete Software *lp_solve*¹² auf dem Server laufen würde, wird der Algorithmus auf einem externen Server ausgeführt. Dazu wurde ein HTTP-Service in *Node.js* implementiert, der eine Liste mit Kursteilnehmerbegrenzungen, die Präferenzlisten (nur IDs), sowie die Parameter für den Algorithmus (minPref, weights) entgegennimmt und mit der Lösung des in Kapitel 4 beschriebenen Optimierungsproblems antwortet. Da es kein Budget für das Projekt gab, wird der Server dauerhaft kostenlos bei *Heroku*¹³ gehostet. Nach 30 Minuten Inaktivität geht er dabei in den Ruhezustand wodurch die nächste Anfrage ein paar Sekunden langsamer beantwortet wird. Falls *Heroku* diesen Service einstellen sollte, kann der Algorithmus auch lokal ausgeführt werden. Eine Anleitung wie die lokale Umgebung installiert wird befindet sich in der Anwenderdokumentation. Die Daten müssen dann mit Hilfe eines *MySQL* Clients vom Server der Universität exportiert und lokal importiert werden. Zum Exportieren und Importieren von Daten befindet sich ebenfalls ein Abschnitt in der Anwenderdokumentation.

¹²<http://lpsolve.sourceforge.net/5.5/index.htm>

¹³<https://www.heroku.com/home>

Kapitel 6

Test des Systems

Das Testen des Softwaresystems wurde in drei Aufgabenbereiche unterteilt. Zuerst wurde das Frontend durchgängig getestet. Anschließend wurde das Backend überprüft. Letztendlich wurden der Algorithmus und seine Auswirkungen auf die Richtigkeit hin untersucht.

6.1 Frontend

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

6.1.1 Startseite

Zum Testen der Startseite steht das Testskript *HomepageTest* zur Verfügung. Das Testskript überprüft dabei nur die Kopfzeile der Website. Dieser Test ist für die Startseite ausreichend, da der Inhalt der Seite von den Verantwortlichen des Empiriepraktikums noch zu erstellen ist. Dies bedeutet, dass nur die Textfelder der Kopfzeile, also *Start*, *Kursübersicht*, *Meine Präferenzliste*, und *Ergebnis sehen*, und deren Reihenfolge verifiziert wird. Gleichermassen werden auch die Schaltflächen in der Kopfzeile, *Anmelden* und *Registrieren*, überprüft.

6.1.2 Registrierung

Der Testfall *RegisterTest* verifiziert, dass ein Nutzer sich ordnungsgemäß registrieren und sich anschließend mit diesen Daten anmelden kann. Dafür ruft der automatisierte Nutzer die Startseite auf und gelangt über den „Registrieren“-Button zum Registrierungsformular. Dort gibt er seine Daten ein. Anschließend ruft der Benutzer die Login-Seite über den „Anmelden“-Button auf und meldet sich mit den Daten aus der Registrierung an. Ein erfolgreicher Login wird durch die Abwesenheit des „Anmelden“-Buttons sicher gestellt. Am Ende des Tests wird der neu registrierte Nutzer gelöscht.

6.1.3 Login

Daran schließt der Testfall *LoginTest* an. Dieser verifiziert, dass ein bereits registrierter Nutzer sich anmelden kann. Dafür wird zuerst ein Benutzer in der Datenbank kreiert. Anschließend wird die Login-Seite geöffnet und die Daten des kreierten Nutzers eingegeben. Der erfolgreiche Login wird durch die Abwesenheit des „Anmelden“-Buttons sicher gestellt. Am Ende des Tests wird der kreierte Nutzer wiederum aus der Datenbank gelöscht.

6.1.4 Kursübersicht

Die Kursübersicht wurde manuell getestet. Dafür wurden zuerst einige Kurse im Backend erstellt. Anschließend wurden sie im Frontend unter dem Menüpunkt *Kursübersicht* auf Richtigkeit überprüft. Hierbei ist es wichtig, dass in dem Einführungstext der Übersicht das korrekte Jahr steht. Daraufhin wurde die Vorschau der Kurse kontrolliert. Dafür wurde sichergestellt, dass der Kurstitel, die Kurzbeschreibung, Ort, Zeit und ein Ausschnitt der ausführlichen Beschreibung dargestellt ist. Weiterhin musste jede Vorschau einen „Details“-Button beinhalten, der zur kompletten Kursbeschreibung führt. Diese detaillierte Ansicht entsprach den in Kapitel 2 aufgeführten Anforderungen.

6.1.5 Meine Präferenzliste

Anschließend wurde der Menüpunkt *Meine Präferenzliste* manuell überprüft. Hierbei müssen alle Kurse sichtbar sein, die in diesem Semester gewählt werden können. Insbesondere ist es wichtig, dass jeder Kurs einzeln per Drag&Drop verschoben werden kann. Nachdem Kurse anders platziert wurden, muss die Liste gespeichert werden können. Das bedeutet, dass bei einem erneuten Aufruf des Menüpunktes die editierte Liste in der zuletzt verschobenen Reihenfolge angezeigt werden muss. Dies wurde verifiziert, indem die Cookies und der Cache geleert wurden und anschließend die Seite neu aufgerufen wurde.

6.1.6 Ergebnis sehen

Der Menüpunkt *Ergebnis sehen* wurde manuell überprüft. Es wurde sicher gestellt, dass alle Nutzer, die eine Präferenzliste abgeschickt haben, in der Übersicht auftauchen.

6.2 Backend

Für das Backend war es wichtig, zu verifizieren, dass sowohl Lehrstühle, Module, und Kurse, als auch neue Dozenten und Administratoren erstellt werden können.

6.2.1 Anlegen neuer Dozenten und Administratoren

Das Anlegen neuer Dozenten und Administratoren erfolgt im Backend unter dem Menüpunkt *Einstellungen* und anschließend unter *Administratoren*. Jeder Benutzer des Backends wird intern als Administrator geführt. Die Rechte eines Backendbenutzer werden über ein Rollensystem verwaltet.

Unabhängig des Rollensystems wurde verifiziert, dass ein Einloggen im Backend einen bereits registrierten Nutzer und die korrekten Anmeldedaten benötigt.

Für jede Rolle, das heißt Dozent und Administrator, wurde weiterhin getestet, dass sie nur die konfigurierten Rechte besitzt.

Zuletzt wurde verifiziert, dass die Rechte der Dozenten und Administratoren den Anforderungen aus 2 genügen.

6.2.2 Erstellen von Lehrstühlen, Modulen und Kursen

Das Erstellen von Lehrstühlen, Modulen und Kurse erfolgt im Backend unter dem Menüpunkt *Verwalten*.

Ein Testskript für das Erstellen von Lehrstühlen wurde unter dem Namen *CreateChair* bereit gestellt. Da das Erstellen von Modulen und Kursen denselben Prinzipien folgt, wurden hierfür keine weiteren Skripte erstellt. Bei diesen wurde daher manuell sicher gestellt, dass die nach den Anforderungen benötigten Felder existieren und die Pflichtfelder ausgefüllt werden müssen. Weiterhin wurde verifiziert, dass Änderungen an bereits erstellten Kursen, Lehrstühle und Module wirksam sind. Das heißt, ändert ein Dozent seinen Kurs, muss diese Änderung sofort im Frontend sichtbar sein.

6.3 Verteilungsalgorithmus

Um den Algorithmus zu testen, mussten zunächst Daten generiert und in die Datenbank importiert werden. Zu diesem Zweck wurde zuerst ein R-Skript geschrieben, das Präferenzlisten für beliebig viele Studenten und Kurse generiert. Mithilfe eines weiteren Skripts wurde dieses Datenset in die Datenbank des Servers importiert. Anschließend konnte der Verteilungsalgorithmus im Backend gestartet werden.

Für das Erzeugen der Präferenzlisten wurden eine Gleichverteilung und eine Normalverteilung genutzt.

Des Weiteren wurden auch reale Präferenzlisten aus dem Wintersemester 2017/2018 bereit gestellt. Dadurch konnte der neue Algorithmus direkt mit dem alten Verteilungsalgorithmus verglichen werden.

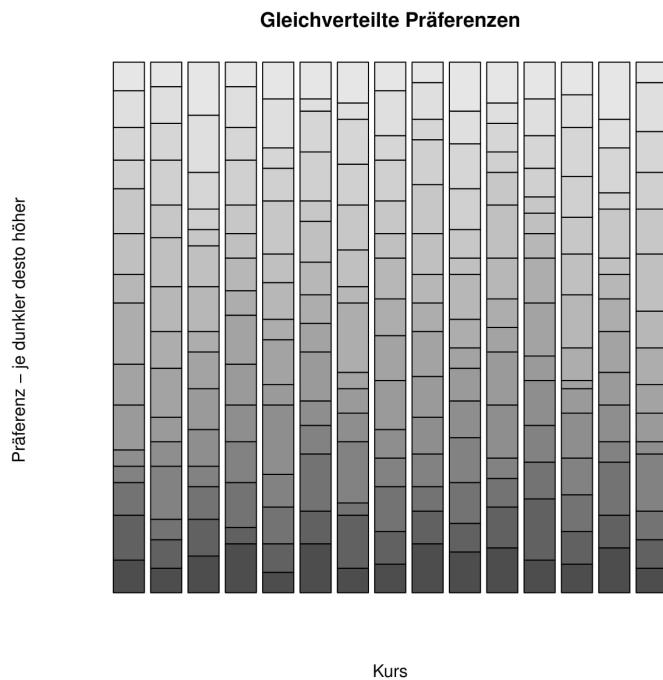


Abbildung 6.1: Gleichverteilte Präferenzen mit 130 Studenten auf 15 Kursen mit maximal 10 Teilnehmern pro Kurs. Die Dunkelheit stellt die Höhe der Präferenz dar

6.3.1 Gleichverteilung

In Abbildung 6.1 sind die gleich verteilten Präferenzen zu sehen. Das heißt, die Wahrscheinlichkeit, dass ein beliebiger Student einen beliebigen Kurs mit beliebiger Präferenz wählt, ist immer gleich. Dennoch ist eine gewisse Streuung in der Ziehung der Präferenzen zu beobachten. Daher hat jeder Kurs leicht unterschiedlich viele Studenten pro Präferenz.

Das Ergebnis des Verteilungsalgorithmus ist in Abbildung 6.2 dargestellt. Es ist zu sehen, dass die meisten Studenten ihrer Erstwahl, das heißt dem Kurs mit Präferenz 15, zugewiesen werden. Nur sehr wenige Studenten werden in ihre Zweitwahl verteilt.

Der Grund für diese derart gut Verteilung ist, dass die Präferenzen auf die Kurse gleich verteilt werden. Wären die Präferenzlisten exakt gleich verteilt, dann könnte jeder Student dem Kurs mit Präferenz 15 zugewiesen werden.

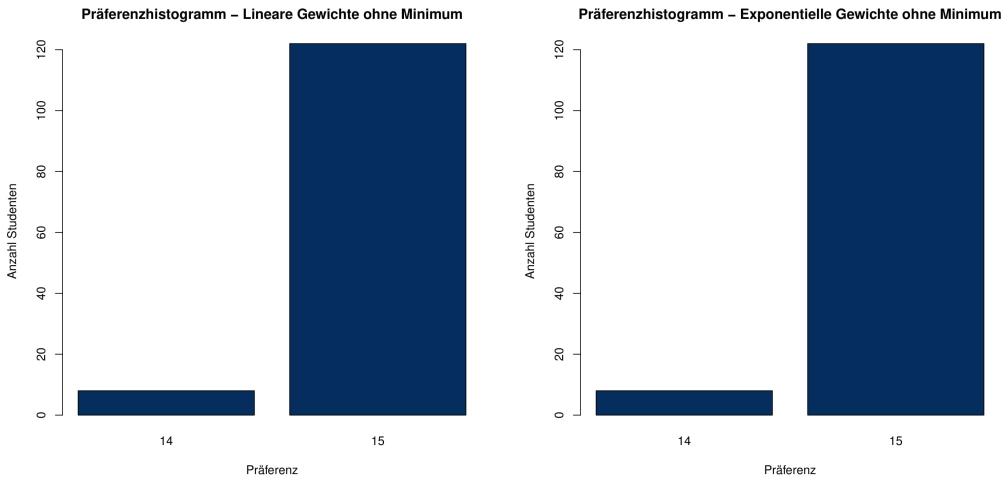


Abbildung 6.2: Gegenüberstellung der Präferenzhistogramm für lineare und exponentielle Gewichtung

Durch die oben erwähnte leichte Streuung, muss es allerdings Studenten geben, die nicht ihrer Erstwahl zugeordnet werden.

Die Variante mit einem festen Minimum wurde nicht getestet, da es nicht möglich ist, ein höheres Minimum zu erreichen.

6.3.2 Normalverteilung

Abbildung 6.3 zeigt die normal verteilten Präferenzlisten. Es ist zu sehen, dass manche Kurse deutlich häufiger als hohen Präferenzen angegeben wurden. Basierend auf diesen Präferenzlisten, ist davon auszugehen, dass nicht alle Studenten ihrer Erst- oder Zweitwahl zugeordnet werden.

Das Ergebnis des Verteilungsalgorithmus ist in Abbildung 6.2 dargestellt.

Wie man sieht, kommen immer noch mehr als die Hälfte der Studenten in ihre Erstwahl. Um die 20 Studenten werden allerdings nur einem Kurs mit einer Präferenz kleiner 11 zugeteilt.

Gut zu sehen ist, dass die Variante „Lineare Gewichte ohne Minimum“ sich von „Lineare Gewichte mit Minimum“ unterscheidet. Die erste Variante hat

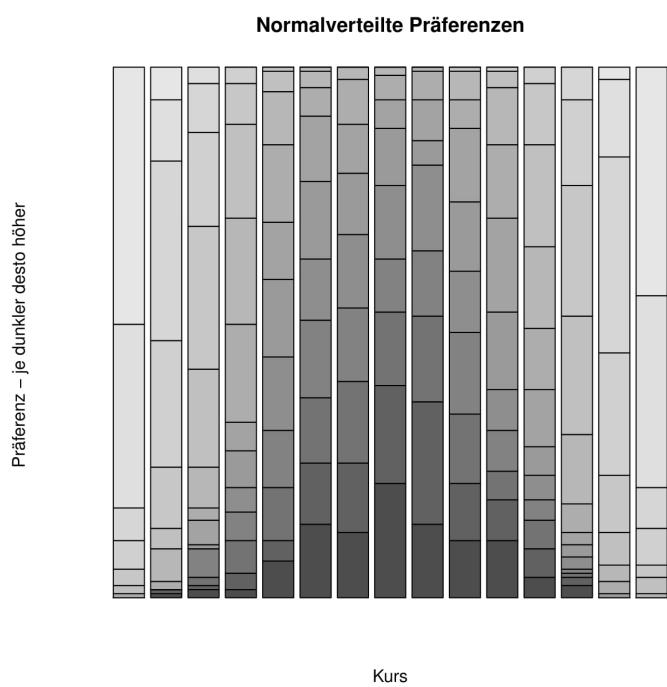


Abbildung 6.3: Normalverteilte Präferenzen mit 130 Studenten auf 15 Kursen mit maximal 10 Teilnehmern pro Kurs. Die Dunkelheit stellt die Höhe der Präferenz dar

die minimale Präferenz 6, doch durch das Erzwingen eines festen Minimums verbessert sich diese auf 7. Nach den Anforderungen ist dies so wünschenswert. Zuletzt ist in der Variante „Exponentielle Gewichte ohne Minimum“ zu sehen, dass weniger Studenten ihre Erstwahl erhalten, dafür aber mehr Studenten die Präferenz 10 oder höher bekommen. Ob dieser Effekt ebenfalls wünschenswert ist, oder aber die Lösung der Variante „Lineare Gewichte mit Minimum“ vorzuziehen ist, kann hier nicht beantwortet werden.

6.3.3 Realdaten

Zuletzt wurde der Algorithmus mit realen Daten aus dem Wintersemester 2017/2018 getestet. Dadurch ist ein direkter Vergleich mit dem alten Verteilungsalgorithmus möglich. Der Datensatz umfasst die Präferenzlisten von 92 Studenten, die auf 15 Kurse verteilt werden sollen.

	Alter Algorithmus	Neuer Algorithmus
Mittelwert	14.2	14.4
Varianz	1,8	0,75
Minimale Präferenz	7	12

Tabelle 6.1: Vergleich des alten Algorithmus mit dem neuen Algorithmus

In Tabelle 6.1 sind die Ergebnisse des alten und des neuen Algorithmus gegenübergestellt. Für diesen Vergleich wurden lineare Gewichte mit einem festen Minimum von 12 verwendet. Es ist zu sehen, dass sich der Mittelwert nur leicht erhöht. Die Varianz hingegen sinkt um 1.05, sinkt also deutlich. Die minimale Präferenz steigt im Vergleich zur vorigen Variante des Verteilungsalgorithmus um 5 an. Daraus lässt sich ableiten, dass der neue Algorithmus zu einem besseren Ergebnis führt.

6.3.4 Ergebnisse des Algorithmus

Wie in Abschnitt 6.3.2 bereits erläutert, führen die verschiedenen Varianten des Algorithmus zum Teil zu unterschiedlichen Ergebnissen. Nach den Anforderungen aus Kapitel 2, soll der Administrator die Wahl haben, welches

Verteilungsergebnis genutzt werden soll. Um die korrekte Funktionalität dieser Ergebnisauswahl zu testen, wurden mehrmals Verteilungen generiert und im Anschluss sicher gestellt, dass die entsprechenden Verteilungen ins System übernommen wurden. Dies wurde realisiert, indem die minimale Präferenz der eingetragenen Nutzer mit dem theoretischem Ergebnis verglichen wurde.

Gleichzeitig wurde verifiziert, dass ein Bestätigen der Verteilung die entsprechenden E-Mail-Benachrichtigung nach sich zieht. Dafür wurde ein Mailserver genutzt, der die E-Mails aller Benutzer erhält. Nach den Anforderungen aus Kapitel 2 sollten hierbei drei Typen von Mails verschickt werden. Studenten erhalten eine E-Mail mit Informationen, welchen Kurs sie zugeteilt wurden. Dozenten erhalten eine E-Mail mit Informationen, welche Studenten in ihren Kursen sind. Administratoren erhalten eine E-Mail mit Informationen, welche Studenten welchen Kurs bekommen haben.

Für die Verifikation wurde jeweils eine E-Mail der Rolle gewählt und die Richtigkeit der Informationen überprüft.

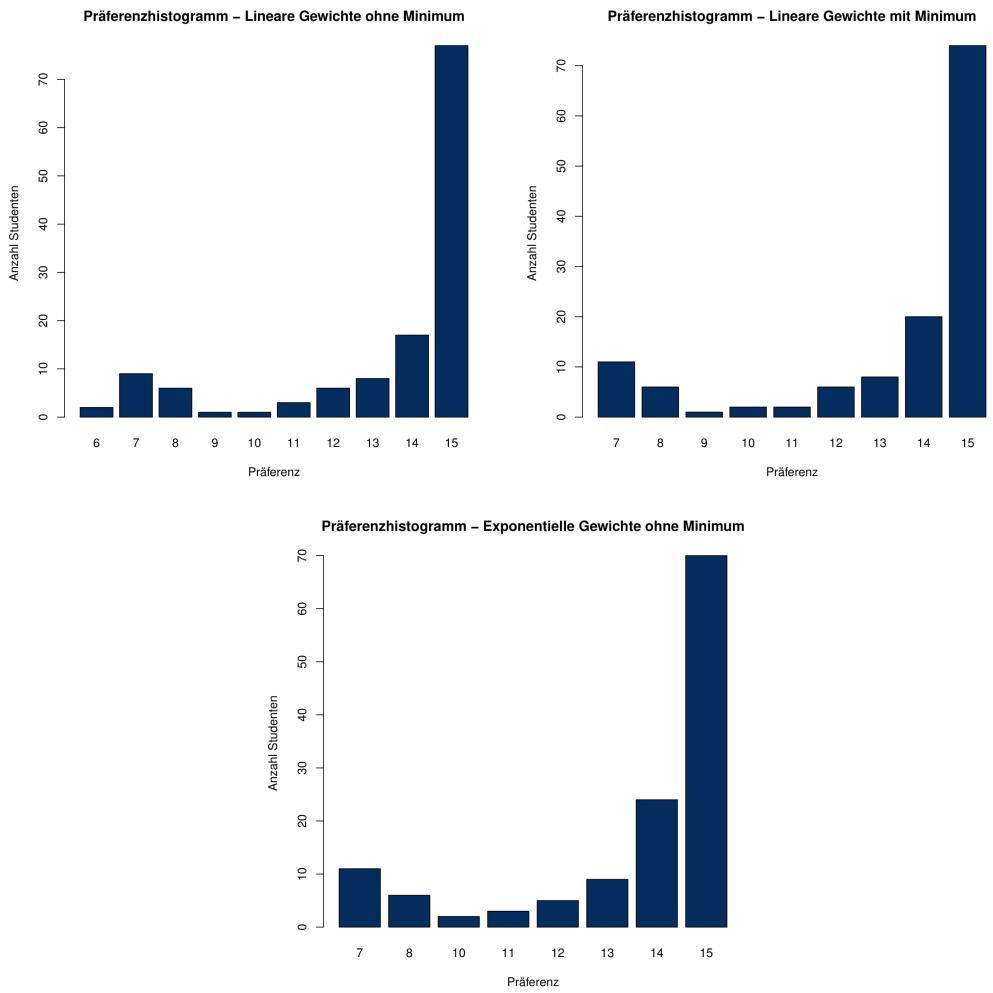


Abbildung 6.4: Gegenüberstellung der Präferenzhistogramm für lineare Gewichtung, lineare Gewichtung mit Minimum und exponentielle Gewichtung

Kapitel 7

Zusammenfassung

Zielsetzung des Projekts war es, eine Plattform zu schaffen, mithilfe dessen 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. Dann konnte der Verteilungsalgorithmus in die bestehende Plattform eingefügt werden.

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 funk-

tionieren. Der Verteilungsalgorithmus erreichte auf künstlich generierten Datensätzen gute Verteilungen. Im Vergleich mit dem Algorithmus der alten Lösung, konnten ein deutlich besseres Ergebnis der Verteilung erzielt werden.

Es konnten nicht alle Anforderungen umgesetzt werden. So wurde zum Beispiel auf die Implementierung einer Tauschbörse verzichtet. Ebenso wurde zwar das Backend zur Verwaltung verschiedener Module ausgelegt, nicht aber das Frontend. Diese beiden Erweiterungen sind eine Weiterführung der Arbeit an der Einschreibungsplattform naheliegend.

noch irgend ein schlauer Schlussatz der mit Zusammenfassend... beginnt