Studienarbeit

Engineering Project Team Builder

Hochschule für Technik Rapperswil

Abteilung Informatik

Herbstsemester 2020/2021

Autoren Julia Tanner
Severin Amacher

Betreuer Prof. Dr. Farhad D. Mehta

Task Description

Setting

The module "Engineering Project" (EPJ) is central to the software engineering curriculum at the OST Eastern Switzerland University of Applied Sciences. This module requires all participants to form teams of 4 to 5 members in order to work on a project under the guidance of a supervisor. The successful building of a team is a prerequisite for the EPJ module. Over the last five years there has been an increase in the number of different specialisations and timetable variations offered by the degree program. Additionally, a larger percentage of the student population work part-time. Reserving fixed slots in the timetable was deemed infeasible. This has led to a situation where teams have very limited time when they are available to work together. Finding times for their meetings with their supervisors has also become increasingly difficult. Till now, students have been able to make their own teams amongst themselves. The assignment of supervisors to teams has been done by the module organiser manually, based on the mutual availabilities of the team members and the supervisors. In a number of cases, far-reaching last-minute changes in team members and supervisor assignments have been necessary to make sure that all participants were able to take part in the module.

Goals

The main aim of this project is to aid the building of teams and the assignment of teams to supervisors for the EPJ module. This is to be achieved using a software solution "EPJ Team Builder" with the following proposed functionalities:

- Input:
  - Availabilities of all participants and supervisors
  - Student preferences for teams, topics, technologies, etc (if any).
  - Supervisor preferences for teams, topics, technologies, etc (if any).
- Output: A reasonably good assignment of teams and supervisors.
- Quality criteria for solution:
  - The time available for all team members to work together should be maximised.
  - It must be possible to assign each team to at least one supervisor with matching availabilities.
  - If possible, team member preferences for participants and team/topic/technology preferences for supervisors must be taken into account.
- Automated solution search: The software system should incorporate the state of the art in search algorithms in order to find desirable solutions.
- Interactive solution search: It must be possible for the module organiser to view and iteratively modify or refine solutions presented by the system in order to reach a complete solution. This may also include fixing certain points in the solution space and asking the system to show remaining solutions, explain why a solution is currently not possible, suggest constraints that need to be relaxed, etc.

The following tasks are suggested in order to achieve this goal:

1. Look into how team and supervisor assignments for the EPJ module are done currently.
2. Research into the current state of the art (algorithms as well as existing software tools) for solving such problems.

3. The design and implementation of a usable software solution.
4. Evaluation of this solution in December 2020 for building teams for the next EPJ execution (Spring of 2021)

Abstract

Das Modul "Engineering Project" (EPJ) ist ein zentraler Bestandteil des Software-Engineering-Lehrplans der OST - Ostschweizer Fachhochschule. Für das EPJ werden Teams von 4 bis 5 Personen gebildet, welche von einem Betreuer während der Arbeit begleitet werden. Bislang konnten die Studenten die Gruppen selbst bilden. Der EPJ Organisator, Prof. Dr. Farhad D. Mehta, war anschliessend dafür zuständig, für übrig gebliebene Studenten eine Gruppe zu finden und jedem Team einen Betreuer zuzuteilen. Die im ersten Augenblick einfach erscheinende Aufgabe wurde in den letzten Jahren immer komplexer, da es immer mehr Teams gibt und stetig neue Stundenplan-Varianten dazukommen, was zu weniger gemeinsamen Verfügbarkeiten führt.

Bisher wurden für die Planung verschiedene Hilfsmittel verwendet. Einerseits wurden zum Beispiel kleine Programme geschrieben, um eine erste Version für die Zuteilung zu machen und anderseits wurden Dokumente ausgedruckt und auf dem Tisch verteilt, um Gemeinsamkeiten zu erkennen. Diese Arbeit ist sehr aufwändig und erfordert viel Zeit. Durch den Einsatz eines Tools könnte der ganze Prozess digitalisiert werden, wodurch der EPJ Organisator bei seiner Aufgabe entlastet werden würde.

Aus diesem Grund haben wir in unserer Studienarbeit eine massgeschneiderte Web-Applikation ausgearbeitet und entwickelt, welche den ganzen Prozess verwalten und vereinfachen soll. Unser Endprodukt erlaubt das Erstellen, Anzeigen, Editieren und Löschen von Studierenden, Betreuern und EPJ-Gruppen. Um nicht von Hand alle Vorgaben und Wünsche der Studenten bezüglich der Gruppeneinteilung zu sortieren und zu evaluieren gibt es zwei Übersichtsseiten. Auf der einen Seite wird ein Graph mit beliebig vielen Knoten angezeigt, welche die Studenten und Betreuer repräsentieren. Anhand von verschiedenen konfigurierbaren Einstellungen wird evaluiert, ob zwei Knoten eine Verbindung haben oder nicht. Wenn der EPJ Organisator zum Beispiel wissen will, welche Studenten miteinander in einer Gruppe sein wollen, kann er die Berechnung des Graphen so konfigurieren, dass nur Verbindungen zwischen Studenten existieren, die sich gegenseitig als Teampräferenz angegeben haben. Auf der zweiten Seite wird eine Timetable angezeigt. Auf dieser Seite kann der Organisator Studenten, Betreuer und Gruppen auswählen und bekommt eine klare Übersicht, welche Person oder Gruppe was für Verfügbarkeiten angegeben hat. So kann der Organisator bequem den passenden Betreuer für eine Gruppe finden oder einen übriggebliebenen Studenten einer Gruppe zuteilen.

Lay Summary

Ausgangslage

In unserer Studienarbeit hatten wir die Aufgabe, den Organisationsprozess des EPJ Organisator zu digitalisieren und ein Hilfsmittel für das Erstellen von Gruppen zur Verfügung zu stellen. Dabei ging es nicht nur darum eine Applikation aus konkreten Angaben zu programmieren, sondern auch gemeinsam mit Herr Mehta zu evaluieren, welche Funktionalitäten sinnvoll sind und wie die Planung der EPJ Gruppen am besten digitalisiert und optimiert werden kann. Das Ziel war eine Übersicht der vielen Informationen von den Studenten und Betreuern darzustellen. Der Organisator hat dadurch eine bessere Informationsvisualisierung und kann die Gruppen besser und einfacher zusammenstellen als auf dem bisherigen Weg.

Vorgehen

Über die ganze Arbeit bestand eine enge Zusammenarbeit mit Herr Mehta da es schwierig war zu evaluieren, was dem EPJ Organisator schlussendlich helfen wird. Es gab viele Ansätze, die wir verfolgt haben, aber auch wieder fallen liessen, da kein grosser Nutzen daraus gezogen werden konnte. Wir merkten schnell, dass wir realistische Daten benötigen, um die ganze Problematik der Gruppenbildung besser zu verstehen und einschätzen zu können. Also machten wir nach einigen Wochen eine Datenerhebung, bei der die EPJ Betreuer und die SE1 Studenten ihre Verfügbarkeiten für das HS20 und verschiedene Präferenzen erfassten. Mithilfe der Daten aus der Datenerhebung und durch die stetige Kommunikation konnten wir dann die Funktionen ausarbeiten, die wir für sinnvoll empfanden. Diese Funktionen optimierten wir anschliessend, indem wir selbst die Gruppenbildung durchführten und Verbesserungen implementierten. Wir befanden uns also stetig in einem Zyklus aus Funktionalität implementieren, besprechen von Verbesserungen und Verbesserungen umsetzen.

Ergebnisse

Am Ende setzten wir eine Lösung um, bei der die Betreuer und Studenten als Knoten in einer Graphansicht angezeigt werden. Die Kanten zwischen den Knoten stellen die Gemeinsamkeiten dar. Diese Ansicht ermöglicht es dem Organisator gleich von Anfang an die überwältigende Anzahl von Informationen übersichtlich mittels den eingebauten Filtern und Einstellungen zu verarbeiten. Nach dem ersten Durchlauf mit der Graphansicht gibt es immer noch Studenten, die noch keiner Gruppe zugeordnet sind. Zu diesem Zeitpunkt kommt die zweite Übersicht in Form einer Timetable zum Zuge. Die Timetable erlaubt dem Organisator einen besseren direkten Vergleich zwischen Studenten, Gruppen und Betreuer. Zusätzlich beinhaltet die Applikation mehrere Übersichtsseiten, welche spezifischere Auskunft über die Gruppen, Studenten und Betreuer als Individuum geben.

Ausblick

Der Standpunkt unserer Applikation ist ein guter Ansatz, um die Arbeit für den EPJ Organisator zu vereinfachen. Ob alle Funktionalitäten bereits vollständig ausgereift sind, wird sich bei der Benutzung herausstellen. In Zukunft könnte der Vorbereitungsprozess noch digitalisiert werden. Dann würde beim Hinzufügen von Studenten und Betreuern direkt eine E-Mail an die Benutzer mit ihrem Passwort geschickt werden. Weiterhin könnte das Login über einen Microsoft Login Provider gehen, damit sich Studenten und Betreuer gleich mit ihrem OST Account einloggen können. Als letzte Erweiterung könnte man eine Funktion implementieren, die alle erstellten Gruppen analysiert und die Zusammengehörigkeit bewertet. Danach wird ein Durchschnitt der Bewertung aller Gruppen angezeigt. Falls der Durchschnitt schlecht ist, könnte die Applikation gleich Vorschläge geben, um die Bewertungen zu verbessern.

Danksagungen

Wir danken den folgenden Personen für ihre Unterstützung während unserer Studienarbeit:

- Betreuer: Prof. Dr. Farhad D. Mehta
- Technischer Support: Fabian Hauser

Inhaltsverzeichnis

Glossar und Abkürzungsverzeichnis

BCrypt BCrypt ist eine Hashfunktion, welche verwendet wird um Passwörter zu hashen. 11

EPJ Abkürzung für Engineering Project. 2, 48

Frameworks Frameworks sind Platformen, die eingesetzt werden für die Entwicklung von Software Applikationen. 2

JSON JavaScript Object Notation ist ein Datenformat, welches für den Datenaustausch zwischen Anwendungen verwendet wird. Die Daten werden in einer einfachen und lesbaren Textform dargestellt. 10

JUnit JUnit ist ein Framework zum Testen von Java-Programmen. 25

MVP Abkürzung für Minimum Viable Product. 47, 48, 50

SA Abkürzung für Studienarbeit. 47

Spring Boot Spring Boot ist eine Erweiterung des Spring Frameworks, welche das Einrichten einer Spring-Anwendung vereinfacht. 2

Spring Security Spring Security ist ein Teil des Spring Frameworks und stellt Funktionalitäten wie Autorisierung und Authentisierung zur Verfügung. 11

Thymeleaf Thymeleaf ist eine moderne Server-Side Java Template Engine für das Verarbeiten und Erstellen von HTML. 3

Abbildungsverzeichnis

Tabellenverzeichnis

Teil I

Technischer Bericht

Kapitel 1

Technischer Bericht

1.1 Ausgangslage

Für das Engineering Project fand bis jetzt zu Beginn der ersten Woche im Frühlingssemester ein Vorbereitungs- und Kickoff Meeting statt. Das Ziel von dieser Veranstaltung war, die Studenten über das EPJ zu informieren, Gruppen festzulegen und Ideen für den Projektantrag zu sammeln. Bis Mitte Woche musste jede Gruppe einen Projektantrag, in Form einer Markdown Datei, abgeben, welche die Projektbeschreibung und mindestens fünf Angaben für Beratungs- und Review Zeitslots enthält. Nach dieser Abgabe war der EPJ Organisator dafür verantwortlich, möglichst schnell für jede Gruppe, anhand der angegebenen Zeitslots, einen passenden Betreuer zu finden. Oftmals funktionierte dies nur, weil ein Betreuer dazu bereit war, alle Gruppen zu betreuen, für die es sonst keinen passenden Betreuer gab. Ausserdem gab es zahlreiche Fälle, in denen in letzter Minute weitreichende Änderungen bei den Teammitgliedern und den Betreuern notwendig war. Diese Änderungen waren nötig, um sicherzustellen, dass alle für das EPJ angemeldete Studenten auch wirklich an dem Modul teilnehmen konnten.

Um den EPJ Organisator zu entlasten ist das Ziel dieser Arbeit eine Web-Applikation zu implementieren, welche den Organisationsprozess digitalisiert und vereinfacht. Sobald der Stundenplan für das Frühlingssemester bekannt ist, können die Studierenden und Betreuer ihre Wünsche und Verfügbarkeiten erfassen. Dadurch kann Prof. Dr. Farhad D. Mehta die Planung und Zuweisung der Gruppen bereits vor Semesterbeginn durchführen.

Welche Funktionalitäten die Web-Applikation unterstützen soll, waren bei dieser Arbeit nicht von Anfang an bekannt. Es war ein fortlaufender Prozess, um die Anforderungen Schritt für Schritt gemeinsam zu erarbeiten.

1.2 Umsetzung

Um die Web-Applikation zu entwickeln, wollten wir für das Backend entweder C# oder Java verwenden. Die Wahl viel schlussendlich auf Java, weil Prof. Dr. Farhad D. Mehta mehr Erfahrung mit Java anstatt C# hat und er das Produkt nach Ende der Studienarbeit warten wird.

Es gibt verschiedene Frameworks, um eine Java Webanwendung zu implementieren. Wir haben uns für Spring Boot [10] entschieden, da es extrem mächtig und trotzdem einfach zu programmieren ist, man Dependency Injection anwenden kann und aktuell stark verbreitet ist. Ausserdem wollten wir beide etwas Neues lernen, was wir in Zukunft bestimmt wieder einmal antreffen werden.

Für den Aufbau der Views haben wir Thymeleaf [13] verwendet. Thymeleaf ist eine Java Template Engine, mit der Views definiert werden können, welche zugeschickte Daten aus dem Backend auslesen und anzeigen sollen.

Weitere technische Informationen zum Ergebnis, wie zum Beispiel die Softwarearchitektur, alle verwendeten Programmiersprachen sowie genaue Angaben zu den Funktionalitäten, sind im OST Gitlab [4] zu finden.

1.3 Ergebnisse

Das Ergebnis unserer Studienarbeit ist eine Web-Applikation, welche das Verwalten der am EPJ angemeldeten Studenten, die Erfassung der Informationen von Studierenden und Betreuern und die Gruppenbildung in einem Tool abbildet.

Für die Studierenden und die Betreuer gibt es eine separate URL, auf der sie sich anmelden und anschliessend ihre Daten erfassen können. Die Studierenden können angeben, mit welchem Zeitmodell sie studieren und mit wem sie gerne in einem Team wären. Die Betreuer können angeben, wie viele Teams sie mindestens und maximal betreuen möchten. Ausserdem haben beide die Möglichkeit, ihre Interessen anhand der Themenbereiche und Technologien festzulegen. Zusätzlich können sie definieren, an welchen Zeitslots sie in der Woche Zeit haben, um für das EPJ zu arbeiten. Alle diese Angaben können in die Entscheidung des EPJ Organisators für die Gruppenbildung miteinfliessen.

Für den Administrator gibt es eine weitere URL, um die ganze Planung durchzuführen. Auch hier gibt es ein Login, um sicherzustellen, dass nur er Zugriff hat. Da die Funktionalitäten des Administrators sehr umfangreich sind und bereits detailliert im Benutzerhandbuch erklärt werden, möchten wir diese hier nicht genauer ausführen und auf das OST Gitlab [4] verweisen.

1.4 Schlussfolgerungen

Alle Anforderungen, welche sich während des Verlaufs der Studienarbeit herausstellten, konnten wir in der Webanwendung umsetzen. Dies war möglich da wir darauf geachtet haben, alles so zu implementieren, dass die Anwendung einfach erweiterbar ist. Ausserdem konnten wir durch intensives testen, sei es durch manuelle oder Unit Test viele Fehler frühzeitig beheben und sind daher zuversichtlich, dass es keine grösseren Fehler mehr geben sollte.

1.4.1 Probleme

Graphvisualisierung

Der grösste Knackpunkt unserer Studienarbeit stellte die Graph Visualisierung dar. Das Problem lag darin, dass der Graph durch die vielen Knoten und Kanten schnell unübersichtlich wurde. Um dieses Problem etwas zu entschärfen, haben wir als erstes Filter und Konfigurationsoptionen eingebaut, damit der Organisator die Anzeige gut verändern und nach möglichst allen Faktoren ordnen kann. Diese Anpassung verbesserte jedoch nicht den Aspekt, dass dem Organisator bereits von Anfang an einen übersichtlichen Graphen angezeigt werden soll. Wir probierten verschiedenste Anzeigeoptionen mit unterschiedlichen Layouts der Graph Library [5] aus, um beruhend auf den Input Daten festzulegen, wie stark die Verbindung zwischen zwei Knoten angezeigt wird. Je nachdem welche Daten wir verwendet haben und wie viele Kanten es gab, war mal das eine und mal ein anderes Layout besser, um den Graphen übersichtlich darzustellen. Aufgrund dessen kamen wir zum Entschluss, dem Organisator drei Standard Layouts zur Verfügung zu stellen, wie der Graph gezeichnet werden soll. Der

Organisator kann sich jede der Darstellungen anschauen und dann entscheiden, auf welchem Layout er arbeiten möchte.

Docker

Probleme tauchten auch beim Erstellen der Pipeline auf. Da unser Projekt schlussendlich über Portainer gehostet wird, mussten wir von den Applikationen ein Docker Image erstellen und ein Docker Compose bereitstellen, welches auf dem Portainer Host hinterlegt wird. Da wir beide noch nie mit Docker gearbeitet haben, galt dies als eine grössere Herausforderung als es den Anschein machte. Diese Herausforderung bewältigten wir durch die Hilfe von Herr Fabian Hauser. Herr Hauser war unsere Ansprechperson für Portainer und stand uns immer zur Verfügung, wenn wir dockerspeziefische Probleme hatten.

1.4.2 Verbesserungen

Wir sind guter Dinge, das unsere Web-Applikation dem EPJ Organisator bei der Planung helfen wird. Einzig in der Graphansicht können wir uns vorstellen, dass es vermutlich Verbesserungspotential gibt. Wir haben durch die Datenerhebung versucht, möglichst realistische Daten zu erhalten. Leider nahmen an der Datenerhebung nicht so viele Studierende wie erhofft teil oder sie gaben sehr unrealistische Angaben an. Dadurch konnten wir den Graphen und die Darstellung nicht vollständig optimieren, da jede mögliche Verbesserung reine Spekulation gewesen wäre. Wir haben aber versucht, möglichst viele Einstellungen und Filter einzubauen, um den Graphen übersichtlich zu gestalten. Wie ausgereift diese aber sind, und ob noch weitere Funktionen nötig wären, wird sich erst beim Einsatz für die Planung der EPJ Gruppen für das Frühlingssemester 2021 zeigen. Ausserdem kann es gut sein, dass bei der effektiven Anwendung der Applikation dem Organisator weitere Features einfallen, die er noch vermisst.

1.4.3 Ausblick

Für die Zukunft wäre sicher denkbar einen weiteren Algorithmus einzubauen, welcher automatisiert eine erste Version der Gruppenbildung durchführt. So würde dem Organisator einen Vorschlag für die Gruppen angezeigt werden und er müsste anschliessend nur noch Bearbeitungen bei Konflikten oder Studenten, die noch keiner Gruppe zugewiesen sind, vornehmen. Zusätzlich könnte eine Testfunktion implementiert werden, welche die Gruppeneinteilungen des Organisators anschaut und überprüft, ob die Gruppen eine gute Kopplung haben. Falls dies nicht bei allen Gruppen zutrifft, könnte der Organisator darauf aufmerksam gemacht werden und auch gleich Verbesserungsvorschläge angezeigt werden.

Teil II

Projektdokumentation

Kapitel 2

Vision

Nachfolgend wird die Vision des Projekts beschrieben. Es umfasst alle Funktionalitäten, welche die Webapplikation enthalten muss, um nutzbar für den EPJ Organisator zu sein.

- Die Studenten benötigen eine View um Verfügbarkeiten, bevorzugte Teammitglieder, Themen und Technologien erfassen zu können.
- Die Supervisors benötigen eine View um Verfügbarkeiten, Themen, Technologien, minimale und maximale Anzahl der zu betreuenden Teams erfassen zu können.
- Der Administrator hat eine separate View, um alle Verfügbarkeiten von allen Personen anzusehen.
  - Einerseits gibt es eine Timetable Ansicht, bei der verschiedene Personen und Gruppen ausgewählt werden können. Anschliessend werden in der Timetable alle Verfügbarkeiten eingetragen und visuell gekennzeichnet.
  - Ausserdem gibt es eine Smart Data Visualisation. Hier wäre beispielsweise eine Graphansicht vorstellbar. Eine genauere Erläuterung zum Graphen ist im Kapitel Graphansicht zu finden.
- Der Organisator hat die Möglichkeit, neue Gruppen zu erstellen, zu bearbeiten und zu löschen. Die Studenten, welche bereits einer Gruppe hinzugefügt wurden, werden visuell gekennzeichnet.
- Der Administrator kann die Datenbank leeren. Dies ist hilfreich, wenn er mit einer komplett neuen Planung startet und die alten Daten des Vorjahrs nicht mehr benötigt.
- Über ein Input Feld kann der Administrator Benutzer hinzufügen und entfernen. Für neue Benutzer wird direkt ein Passwort angelegt, welches identisch zur E-Mail-Adresse ist. Dieses Passwort kann im Nachhinein von den Benutzern geändert werden. Der Administrator hat zudem die Möglichkeit, Passwörter wieder auf die E-Mail-Adresse zurückzusetzen.
- Das aktuelle Zwischenergebnis kann der Organisator mit einem JSON Export auslesen und zu einem späteren Zeitpunkt wieder importieren. Es werden alle User Inputs und die erstellten Gruppen ausgegeben.
- Der Administrator soll einstellen können, dass Benutzer ihre eingegebenen Daten nicht mehr bearbeiten können. In diesem Fall sind die Daten noch einsehbar aber nur noch im Read-Only Modus. Alle Änderungswünsche laufen dann über den Administrator.
- Es benötigt ein simples Login, damit sich Studenten, Betreuer und der Organisator anmelden können.

2.1 Graphansicht

Die Graphansicht dient zur Visualisierung der Gemeinsamkeiten zwischen den Knoten. Pro Betreuer und pro Student wird ein Knoten erzeugt. Anhand der nachfolgenden Logik werden wir bestimmen, ob eine Kante zwischen zwei Nodes erstellt wird oder nicht.

Presence of Edge

Dieser Abschnitt definiert, ob ein Edge erstellt wird oder nicht. Es muss zwischen zwei Knoten mindestens eine gewisse Anzahl bei der Übereinstimmung der verfügbaren Zeiten geben. Ausserdem muss das berechnete Kantengewicht mindestens einen definierten Wert haben, damit eine Kante erstellt wird.

```
numberSameAvailabilities(N1, N2) = "Anzahl Lektionen, die beide Nodes angegeben haben"
n = "Minimale Anzahl gemeinsamer Lektionen"
t = "Threshold Edge Weight"
EW = "Edge Weight"

=> isEdge(N1, N2) = numberSameAvailabilities (N1, N2) >= n AND EW >= t
```

Edge Weight

Das Kantengewicht definiert, wie stark die Kante gewichtet wird. Dieser Wert wird berechnet, indem pro Property die Anzahl Übereinstimmungen mit dem entsprechenden Gewichtsfaktor multipliziert und das Produkt am Schluss von allen Properties zusammengerechnet wird. Das Kantengewicht kann im Graphen visuell zum Beispiel mit einem dickeren, für viele Übereinstimmungen, oder einem dünneren Edge, für weniger Übereinstimmungen, dargestellt werden.

```
teamMemberPreferences(N1, N2) =
"2 = falls beide Studenten miteinander im Team sein wollen",
"1 = falls nur einer von beiden Studenten gemeinsam im Team sein will",
"0 = falls keine Uebereinstimmung",
sameTopics(N1, N2) = "Anzahl gemeinsamer Themen"
sameTechnologies(N1, N2) = "Anzahl gemeinsamer Technologien"
f1 - f4 = "Gewichtungsfaktor fuer jedes Property"

=> EW(N1, N2) = f1 * numberSameAvailabilities(N1, N2) + f2 * teamMemberPreferences(N1, N2) + f3
    * sameTopics(N1, N2) + f4 * sameTechnologies(N1, N2);
```

Kapitel 3

Anforderungsspezifikation

3.1 Aktoren

3.1.1 Student / Supervisor

Der Student oder Betreuer kann nach erfolgreichem Login seine Daten erfassen und anpassen. Dazu gehören zum Beispiel die Verfügbarkeiten, Themen und Technologien. Zusätzlich hat er die Möglichkeit, das Passwort zu ändern.

3.1.2 Administrator

Der Administrator erfasst die Benutzer, welche am EPJ teilnehmen werden. Er sieht die Eingaben von den Studenten und Betreuern und kann Gruppen erstellen, bearbeiten und löschen. Ausserdem hat er die Möglichkeit, die komplette Datenstruktur zu exportieren, um Zwischenzustände abzuspeichern. Diese kann er zu einem späteren Zeitpunkt wieder importieren.

3.2 Use Cases

Die Use Cases wurden anhand der Meilensteine Core Construction, Alpha Release und Beta Release aufgeteilt und farblich markiert.

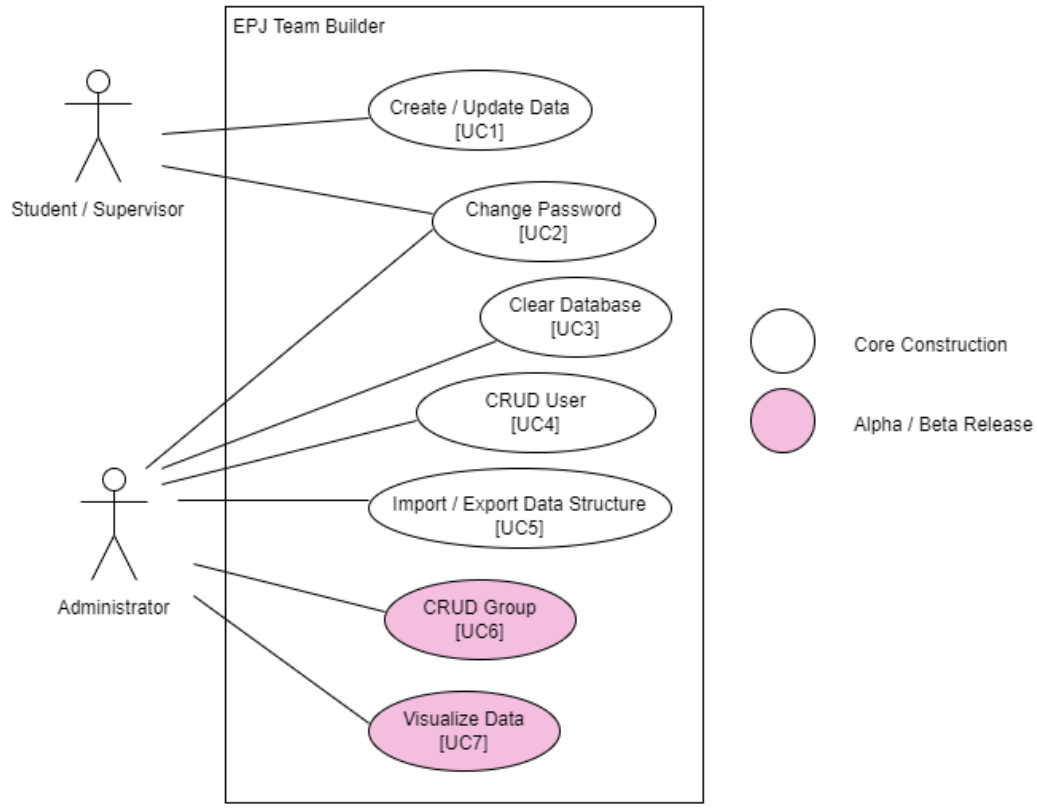

Abbildung 3.1: Use Case Diagramm

3.2.1 Beschreibungen (Brief)

Im Folgenden werden die Use Cases kurz beschrieben.

UC1 - Create / Update Data

Die Studenten und Betreuer haben die Möglichkeit, ihre Daten zu erfassen und zu einem späteren Zeitpunkt zu bearbeiten.

UC2 - Change Passwort

Alle Aktoren können ihr persönliches Passwort ändern.

UC3 - Clear Database

Der Administrator kann alle Records in der Datenbank löschen, bis auf die Technologie und Themen Tags. Diese Funktion benötigt er, wenn er mit der Planung des EPJ Moduls für das nächste Semester beginnt.

UC4 - CRUD User

Über ein Input Feld kann der Administrator Benutzer erfassen und löschen. Dafür kann er mehrere mit Komma getrennte E-Mails eingeben. Für neue Benutzer wird ein Passwort generiert, welches identisch zur E-Mail ist. Ausserdem hat der Administrator die Möglichkeit, dass Passwort eines Users zurückzusetzen und die Benutzerdaten anzupassen.

UC5 - CRUD Group

Der EPJ Organisator kann aus den Studenten und Betreuern Gruppen bilden und diese bearbeiten. Es gibt immer zwei Default Gruppen. Einmal ”keiner Gruppe zugeteilt” und ”Studenten mit mehreren Gruppen”. So kann der Administrator erkennen, ob die Planung gültig ist.

UC6 - Import / Export Data Structure

Zwischenzustände können in eine JSON Datei exportiert werden. Dabei wird die komplette Datenbank exportiert. Die Datei kann zu einem späteren Zeitpunkt wieder importiert werden. Beim Import wird die bestehende Datenbank geleert und die Daten vom JSON werden übernommen.

UC7 - Visualize Data

Der Administrator kann die User Daten visualisieren lassen. Dafür gibt es eine tabellarische Variante und eine Graphansicht. Im Graph gibt es für jeden Studenten und Betreuer einen Knoten, welcher mit einem Kürzel beschriftet ist. Je nach Einstellungen werden zwischen zwei Knoten eine Kante erstellt. Wenn man mit der Maus über die Kante fährt, erhält man zusätzliche Informationen über die Gemeinsamkeiten.

3.3 Nicht funktionale Anforderungen

Nachfolgende Anforderungen werden garantiert und geprüft.

3.3.1 Funktionalität

Die ganze Web-Applikation soll mit den neuesten Versionen von Safari, Chrome und Firefox funktionieren.

Wann wird überprüft: Severin verwendet für das Entwickeln seiner Issues Firefox und Julia verwendet Chrome. So wird die Funktionalität dieser zwei Browser stetig sichergestellt. Nachdem die Construction Phase abgeschlossen ist, werden wir die Checkliste aus Kapitel 7.3 mit diesen drei Browsern durchführen und so die Funktionalität sicherstellen.

3.3.2 Usability

Das Design wird möglichst einfach und selbsterklärend gehalten. Alle Funktionen, welche eine genauere Erklärung benötigen, verfügen über einen Tooltip oder werden im Benutzerhandbuch definiert.

Wann wird geprüft: Die Web-Applikation wird nach jedem Meilenstein präsentiert und für Prof. Dr. Farhad D. Mehta zur Verfügung gestellt. So können wir anhand der Fragen vom Betreuer einschätzen, ob weitere Hilfestellungen im Programm nötig sind und es können jederzeit Verbesserungswünsche angebracht werden. Durch diese Massnahmen können wir die Usability stetig steigern. Ausserdem werden wir im ”Meilenstein 5 - Release” in die Rolle des EPJ Organisators schlüpfen und eine komplette Gruppenbildung durchführen. So erkennen wir komplizierte Konzepte und können diese vereinfachen oder im Benutzerhandbuch genau erläutern.

3.3.3 Sicherheit

Jeder Benutzer, welcher die Web-Applikation verwenden möchte, muss sich gegenüber dem System authentisieren. Das Passwort wird nirgends im Klartext gespeichert, sondern wird mittels BCrypt [2] gehasht und gespeichert. Ausserdem muss eine Authentifizierung stattfinden, damit jeder Benutzer nur auf seine für ihn zugänglichen Seiten zugreifen kann. Der gesamte Sicherheitsmechanismus wird mit dem vom Framework bereitgestellten Spring Security [11] implementiert.

Wenn ein Benutzer also nicht eingeloggt ist oder auf eine Seite zugreifen möchte, für welche er keine Rechte hat, muss der Zugriff unterbunden werden.

Wann wird geprüft: Sobald der Login implementiert ist, muss geprüft werden, ob die Passwörter gehasht gespeichert werden. Ausserdem muss der Zugriff auf unterschiedliche Seiten geprüft werden, um sicherzustellen, dass eine Authentifizierung stattfindet. Die Authentisierung und Authentifizierung wird mit der Checkliste aus Kapitel 7.3 erneut überprüft.

Kapitel 4

Domainanalyse

4.1 Domain Modell

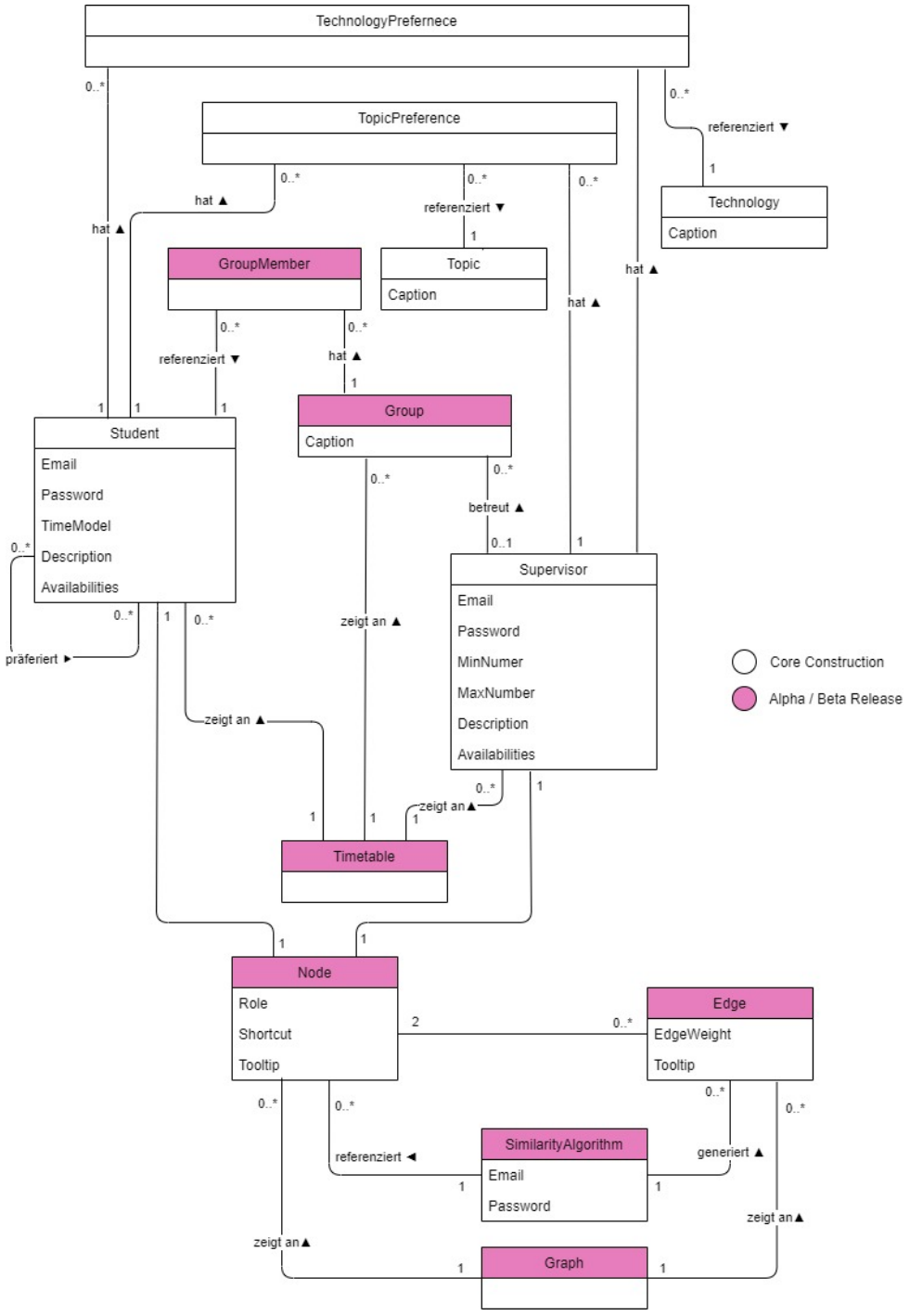

Abbildung 4.1: Domain Modell

Das Domain Modell wurde farblich nach den Meilensteinen aufgeteilt.

4.1.1 Core Construction

Die Konzepte im Domain Modell von der Core Construction sind relativ selbsterklärend. Um das Diagramm etwas kleiner zu gestalten, wurden die Verfügbarkeiten des Studenten und des Betreuers nicht in einer separaten Klasse modelliert. Beide können verschiedene Techno-

logie und Themen Präferenzen angeben. Ausserdem können Studenten andere Studenten als bevorzugte Teammitglieder angeben.

4.1.2 Alpha / Beta Release

Gruppen

Eine Gruppe besteht aus beliebig vielen Gruppenmitgliedern, wobei ein Student zu mehreren Gruppen gehören kann. Ausserdem kann der Gruppe ein Betreuer zugewiesen werden.

Timetable

Über die Timetable können beliebig viele Studenten, Betreuer und Gruppen ausgewählt und angezeigt werden.

Graph

Das Konzept für den Graphen ist etwas komplizierter. Für jeden Studenten und Betreuer wird ein Node erstellt. Über einen Algorithmus wird herausgefunden, ob zwischen den Knoten eine Kante erstellt werden soll. Eine Kante referenziert immer zwei Knoten wobei ein Node zu beliebig vielen Kanten gehören kann. Der Graph zeigt schlussendlich beliebig viele Nodes und Edges an.

Kapitel 5

Software-Architektur

5.1 Datenbankmodell

Das nachfolgende Datenbankmodell wurde für den EPJ Team Builder verwendet. Da vieles selbsterklärend ist, werden hier nur die wichtigsten Punkte genauer erläutert.

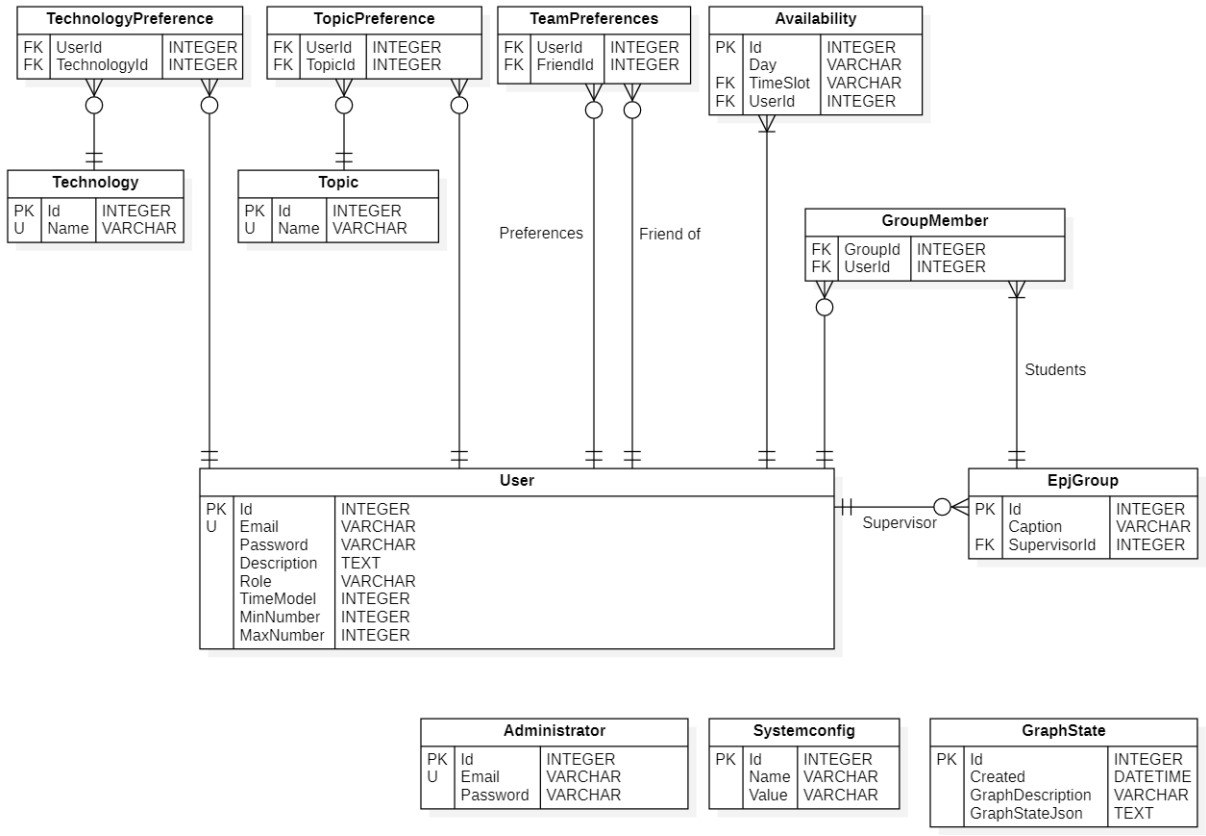

Abbildung 5.1: Datenbankmodell

Studenten und Betreuer werden beide in der `User` Tabelle gespeichert. Sie werden anhand vom `Role` Attribut unterschieden. Für die Rolle verwenden wir einen Enum, welcher in der Datenbank gespeichert wird.

In der `Availability` Tabelle werden alle Verfügbarkeiten einer Person gespeichert. Für die Attribute `Day` und `TimeSlot` haben wir ebenfalls Enums verwendet.

Jede Gruppe besteht aus mehreren Studenten. Dies wird über die Tabelle `GroupMember` abgebildet. Für das Endergebnis darf jeder Student nur einer Gruppe zugewiesen sein jedoch muss der Student für Zwischenresultate mehreren Gruppen zugewiesen werden können. Daher existiert eine `One-to-Many` Verbindung. Der Betreuer einer Gruppe wird direkt über die `EpjGroup` Tabelle gespeichert.

Der Administrator hat die Möglichkeit, eine aktuelle Graphansicht (inklusive Einstellungen und Filter) zu speichern. Die Daten, welche benötigt werden, um beim Laden die Nodes wieder an der gleichen Stelle zu positionieren, werden in der `GraphState` Tabelle gespeichert.

Wenn der Administrator mit der Gruppenbildung Anfangen möchte, kann er die Datenerfassung und Bearbeitung für die Studenten und die Betreuer deaktivieren. Diese Systemkonfiguration wird in der Tabelle `Systemconfig` gespeichert.

5.2 Logische Architektur

5.2.1 Spring - Request Flow

Bevor auf die logische Architektur eingegangen wird, wird in diesem Abschnitt kurz beschrieben, wie der Request Flow in Spring funktioniert. Diese Erklärung soll dabei helfen, dass Schichtenmodell besser zu verstehen.

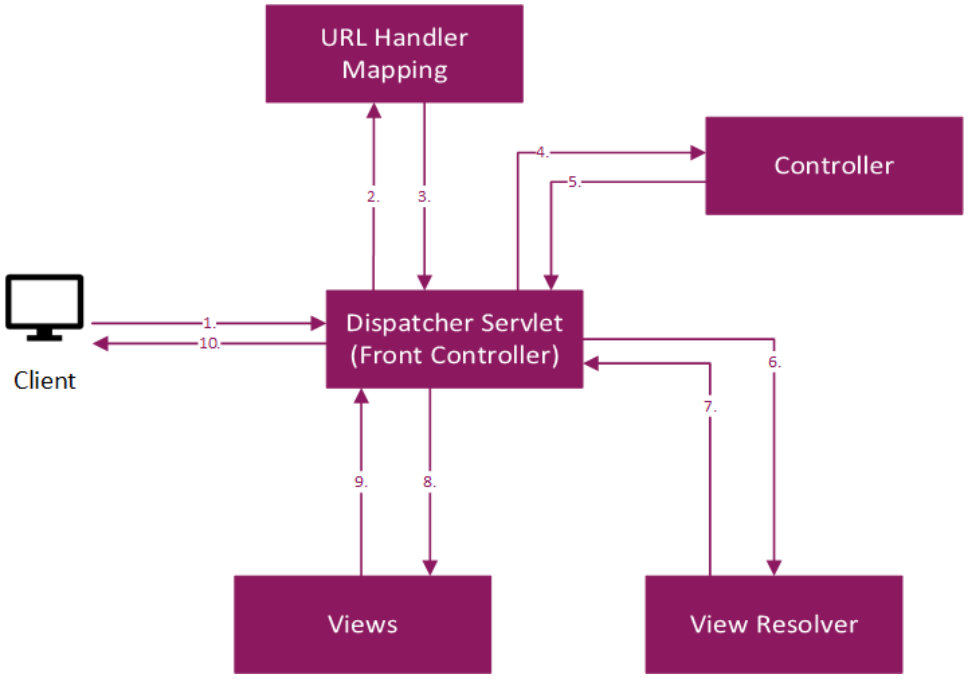

Abbildung 5.2: Spring Request Flow

1. Ein Client tätigt einen Request auf eine URL.
2. Der Request wird vom Dispatcher Servlet, auch Front Controller genannt, empfangen und an den URL-Mapping Handler weitergeleitet. Anhand der URL wird der passende Controller gesucht.
3. Die Informationen zum gefundenen Controller werden an den Front Controller zurückgeschickt.
4. Der Dispatcher Servlet ruft nun den Controller auf. Der Controller ist die Hauptklasse, welche den Ablauf der Logik steuert. Er bereitet die benötigten Daten für die View auf, indem er zum Beispiel einen Aufruf auf einen Service macht.

5. Wenn der Controller mit der Aufbereitung der Daten fertig ist, schickt er `ModelAndView` an den Dispatcher Servlet. `ModelAndView` enthält das Model mit den Daten, welche in der View angezeigt werden sollen, und den View Namen.
6. Der Front Controller sendet nun eine Anfrage an den View Resolver, um die View zu finden.
7. Der View Resolver sucht die View anhand von dem Namen, welcher vom Controller angegeben wurde.
8. Sobald die View gefunden wurde, sendet der Dispatcher Servlet einen Request an die View. Im Request sendet er das Model Objekt mit den Daten, welche in der View angezeigt werden sollen, mit.
9. Die View wird nun mit den Model Daten gerendert.
10. Im letzten Schritt wird die gerenderte View als HTML Response an den Browser geschickt und dem Client angezeigt.

5.2.2 Schichtenmodell

Im folgenden Diagramm ist die grobe Architektur beschrieben.

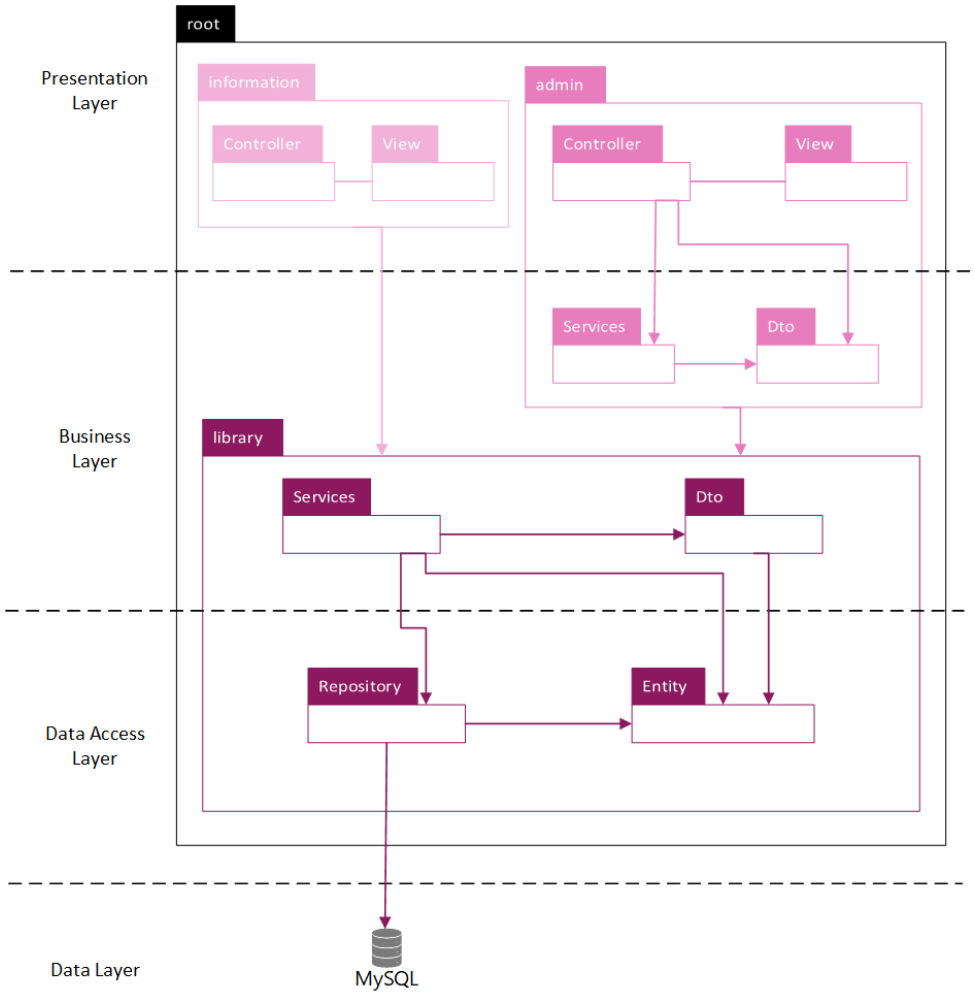

Abbildung 5.3: Schichtenmodell

Die Architektur wurde in drei Module (library, information, admin) aufgeteilt. Um die Leserlichkeit des Diagramms zu steigern, haben wir die einzelnen Verbindungen zwischen den

Modulen nicht genau aufgezeigt. Die Controller verwenden die zentralen Services und die Data Transfer Objects (DTO) im `library` Modul. Die Service Klassen im `admin` Modul haben ausserdem Verbindungen auf die zentralen Services, die Repositories und die Entities.

Die Klassen im Service Package sind dafür verantwortlich, Entities über das JPA Repository aus der MySQL Datenbank auszulesen und in Data Transfer Objects umzuwandeln. Der Controller baut das Model für die Views auf. Dafür liest er die Data Transfer Objects über die Services aus und fügt sie dem Model hinzu. Die View wird anschliessend mit den Daten aus dem Model gerendert. Da der Controller den Namen der anzuzeigenden View definiert und die View die URL angibt, wohin ein Request geschickt werden soll, welcher anschliessend wieder im Controller landet, haben wir eine Verbindung mit einem ungerichteten Pfeil erstellt.

5.2.3 Projekt Aufbau

Unsere Arbeit umfasst zwei Applikationen. Die eine Applikation enthält das Formular für die Studenten und die Betreuer, um ihre Daten zu erfassen. Die andere Applikation beinhaltet das Admin Dashboard. Da das Formular und das Dashboard weitgehend gleiche Funktionalitäten haben, haben wir beschlossen ein Library Projekt zu erstellen, in welchem gemeinsame Services, Entities, Repositories und DTOs implementiert sind. Dies verringert den Maintenance Aufwand, da Änderungen an diesen Stellen ansonsten zwei Mal ausgeführt werden müssten. Diese Aufteilung wurde mittels Maven Modulen gemacht. Das Information (Formular) und Admin (Admin Dashboard) Projekt sind Spring Boot Applikationen und haben jeweils eine Main Methode als Einstiegspunkt in die Applikation. Das Library Projekt hat keine Main Methode da es nicht allein verwendet werden kann. Die drei Maven Module sind in einem Root Projekt vereint, um die Abhängigkeiten gleich beieinander zu haben.

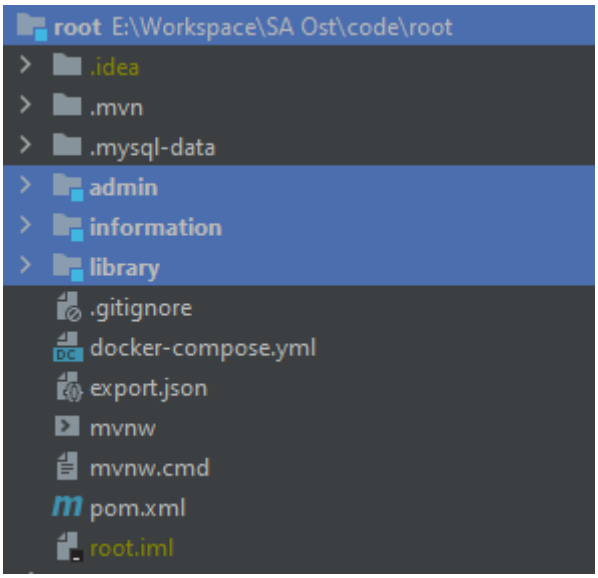

Abbildung 5.4: Projekt Struktur

Pom.xml des Root

Die drei Modules werden im Root Projekt im pom.xml definiert.

```
<modules>
    <module>admin</module>
    <module>information</module>
    <module>library</module>
```

```
</modules>
```

Pom.xml des Library Moduls

Im pom.xml des Library Moduls werden die ArtifactId, die Version und die GroupId gesetzt, welche bei der Angabe der Dependency auf das Library Modul angegeben werden.

```
<groupId>dev.ifs.hsr.ch</groupId>
<artifactId>library</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>library</name>
```

Pom.xml des Information Moduls

Im pom.xml des Information Moduls wird die Dependency auf das Library Modul angegeben.

```
<dependency>
    <groupId>dev.ifs.hsr.ch</groupId>
    <artifactId>library</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
```

5.3 Applikationsarchitektur

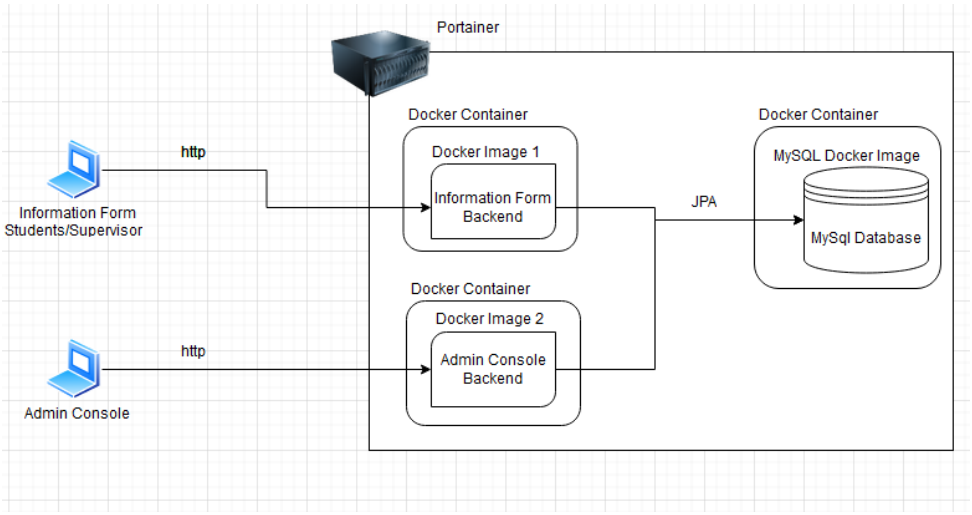

Abbildung 5.5: Applikationsarchitektur

In unserer Arbeit haben wir zwei Applikationen. Die eine Applikation ist für Formular, welches von den Studenten und Betreuern ausgefüllt wird. Die andere ist das Admin Dashboard. Beiden Applikationen sind im gleichen Gitlab Registry, jedoch ist wie bereits erwähnt, jede Applikation ein separates Maven Modul. Obwohl es sich um zwei verschiedene Applikationsinstanzen handelt, greifen beide auf die gleiche DB zu.

5.3.1 Portainer

Das ganze Projekt, also Administrator Dashboard, Studenten / Betreuer Formular und die Datenbank, wird über Portainer gehostet. Pro Applikationen gibt es ein Docker Image, welches in einem Docker Container läuft. Für die Datenbank gibt es einen separaten Docker Container. Das Erstellen der Docker Container wird über Docker-Compose gemacht.

5.3.2 Docker-Compose

Nachfolgend wir die Konfiguration und der Aufbau von Docker-Compose erklärt.

DB Service

```
teambuilder-db:
    image: mysql:latest
    hostname: "{{.Service.Name}}"
    volumes:
      - "db-data:/var/lib/mysql"
    networks:
      - internal-epjteambuilder
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=\${ADMIN_DATABASE_PASSWORD}
```

Für den DB Service wird ein Container aus dem neusten MySQL Image erstellt. Damit der DB Container neu aufgesetzt werden kann ohne, dass die Datenbank alle Daten verliert, wird ein Volume erstellt. Das Volume beschreibt, wo die Daten des Containers auf der Festplatte persistiert werden. In den Environment Variablen wird das Root Passwort für die Datenbank gesetzt. Da die Datenbank nur intern auf dem gleichen Host gebraucht wird, geben wir nur das interne Network an.

Web-Admin / Web-Information Service

```
web-admin:
    image: "docker.io/samacher/admin:master"
    hostname: "{{.Service.Name}}"
    networks:
      - internal-epjteambuilder
      - web
    deploy:
      labels:
        - "traefik.docker.network=web"
        - "traefik.enable=true"
        - "traefik.port=8080"
        - "traefik.frontend.entrypoints=http,https"
        - "traefik.frontend.rule=Host:epjteambuilder-dashboard.dev.ifs.hsr.ch"
        - "traefik.frontend.headers.SSLRedirect=true"
        - "traefik.frontend.headers.SSLProxyHeaders=X-Forwarded-Proto:https"
      mode: replicated
      replicas: 1
      restart_policy:
        condition: on-failure
        delay: 10s
        max_attempts: 10
        window: 60s
    environment:
      - SPRING_DATASOURCE_URL=jdbc:mysql://teambuilder-db/teambuilderdb?
                              createDatabaseIfNotExist=true&
                              serverTimezone=UTC&useLegacyDatetimeCode=false
      - SPRING_DATASOURCE_USERNAME=root
      - SPRING_DATASOURCE_PASSWORD=\${ADMIN_DATABASE_PASSWORD}
      - SPRING_JPA_PROPERTIES_HIBERNATE_DIALECT=org.hibernate.dialect.MySQL5InnoDBDialect
      - SPRING_JPA_HIBERNATE_DDL-AUTO=update
    depends_on:
      - teambuilder-db
    command: bash -c "/usr/wait-for-it.sh --timeout=0 teambuilder-db:3306"
```

Da die Services für die beiden Applikationen fast gleich aussehen, wird hier nur einer gezeigt. Zuerst wird das `image` angegeben, welches sich auf Docker Hub befindet und von unserer Pipeline erstellt und dorthin gepusht wird. Da die Applikationen mit der internen Datenbank kommunizieren und gegen aussen verfügbar sind, geben wir in den Networks das `internal-epjteambuilder` Network an für die interne Kommunikation und das `web` Network für die Kommunikation gegen aussen. Im `deploy` werden mehrere `labels` gesetzt, die für Portainer eine Rolle spielen (z.B. die Url oder den Port). Über `mode: replicated` und `replicas: 1` wird ein Replika des Services erstellt. Zum Schluss werden noch diverse `environment` Variablen angegeben, welche für die Anbindung an die Datenbank notwendig sind.

Networks

```
networks:
  internal-epjteambuilder:
    driver: overlay
    name: internal-epjteambuilder
  web:
    external: true
    name: web
```

In den Services wurden die Networks bereits erwähnt. Insgesamt werden zwei Networks eingesetzt. Das `internal-epjteambuilder` Network, welches für die interne Kommunikation und das `web` Network, welches für die externe Kommunikation zuständig ist. Aus diesem Grund ist beim `web` auch `external : true` gesetzt.

5.4 Gitlab CI/CD

In unserer CI/CD Pipeline haben wir einen Build eingebaut, welcher die Applikationen bildet und auch gleich die Unit Tests ausführt. Ausserdem wird bei einem Push/Merge auf den Master das Docker Image der Applikationen erstellt und auf Docker Hub gepusht. Zum Schluss wird ein WebHook angestossen, um Portainer zu signalisieren, dass eine neue Version der Docker Images zur Verfügung steht. Portainer führt dann das Docker Compose aus und pullt die neuen Images. Beide Applikationen werden in der gleichen Pipeline, jedoch in parallelen Stages, verarbeitet:

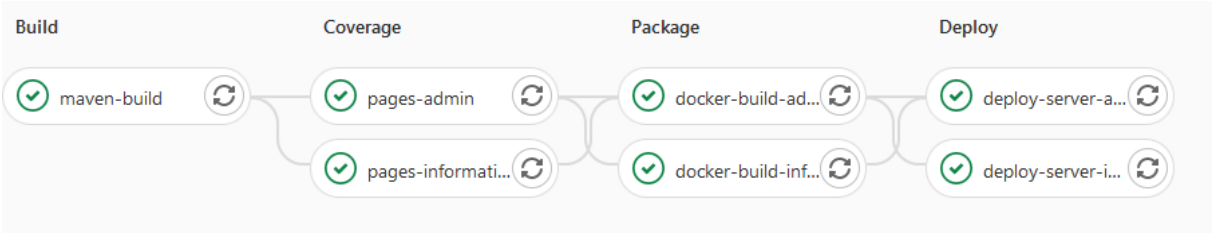

Abbildung 5.6: Gitlab Pipeline

Kapitel 6

Eingesetzte Technologien

In diesem Kapitel werden alle Technologien, welche für den EPJ Team Builder verwendet werden, aufgelistet und beschrieben.

6.1 Frontend

6.1.1 Thymeleaf

Thymeleaf[1] ist eine moderne Server-Side Java Template Engine für das Verarbeiten und Erstellen von HTML. Durch die Verwendung von Thymeleaf können die Models an die View übergeben und direkt über Object Annotation auf die Werte zugegriffen werden. Thymeleaf bietet auch weitere nützliche Features an. Zum Beispiel kann man einen bestimmten Block im HTML abhängig vom Inhalt des Models rendern lassen.

Die View Pages des EPJ Team Builder benutzen also nicht nur reines HTML, sondern auch Thymeleaf, um Daten anzuzeigen.

6.1.2 vis-network

vis-network[2] ist eine JavaScript Library von vis.js[3]. Die Library übernimmt das Zeichnen eines Network Graphen. Diese Library stellt ein Network Objekt bereit, dem man die Nodes und Edges zum Rendern angeben kann. Diese Nodes und Edges kann man zusätzlich über mehrere Optionen selbst konfigurieren. So kann man zum Beispiel Edges schmaler oder breiter zeichnen lassen, die Farbe von Nodes konfigurieren oder einen Tooltip setzen.

Die vis-network[4] Library wird in der Graph Ansicht auf dem Administrator Dashboard verwendet. Ein Algorithmus im Backend berechnet die Nodes und die Kanten und übergibt diese ans Frontend. Das Frontend setzt anschliessend die Nodes und die berechneten Edges auf dem Network Objekt. Über das berechnete Edge Weight kann der Network Graph die Kanten je nach Wert dicker oder dünner darstellen. Ausserdem können die Kanten je nach Einstellung und Filter ausgeblendet werden.

6.1.3 vis-timeline

vis-timeline[5] ist eine JavaScript Library von vis.js[6]. Die Library übernimmt das Anzeigen einer

[1] <https://www.thymeleaf.org/index.html>
[2] <https://visjs.github.io/vis-network/docs/network/>
[3] <https://visjs.org/>
[4] <https://visjs.github.io/vis-network/docs/network/>
[5] <https://visjs.github.io/vis-timeline/docs/timeline/>
[6] <https://visjs.org/>

Timeline. In dieser Timeline können die Zeiten von mehreren Items miteinander verglichen werden.

Die vis-timeline Library für die Timetable Ansicht auf dem Administrator Dashboard eingesetzt. vis-timeline stellt den Zeitraum einer Arbeitswoche täglich von 8:00-19:00 Uhr dar. Die Studenten, Betreuer und Gruppen, welche angezeigt werden sollen, werden als Group Elemente auf das Timeline Objekt gesetzt. Die anzuzeigenden Verfügbarkeiten pro Group Element werden als Items gesetzt.

6.1.4 JavaScript

JavaScript ist eine Skriptsprache, welche im Browser interpretiert wird. So kann man auf einer Webseite Client-Side Logik schreiben, die während der User die Webseite bedient, ausgeführt werden.

Wie bereits erwähnt werden für den EPJ Team Builder einige JavaScript Libraries verwendet. Ausserdem ist die Logik für die Button Interaktionen clientseitig implementiert. Es wird JavaScript verwendet, damit die Interaktionen nicht immer an den Server geschickt werden müssen.

6.1.5 Bootstrap

Für eine schönere Darstellung der Benutzeroberfläche wird Bootstrap[7] verwendet. Bootstrap ist ein Frontend-Framework für die Gestaltung von Webseiten. Durch Bootstrap werden HTML- und CSS Vorlagen bereitgestellt, welche direkt verwendet werden können. Dadurch erspart man sich einiges an Zeit für das Schreiben von CSS.

6.2 Backend

6.2.1 Spring Boot

Spring[8] ist ein Framework, welches für die Entwicklung von Web-Applikationen mit Java verwendet wird. Es bietet viele zusätzliche Features an wie zum Beispiel Dependency Injection, Spring Security und vieles mehr.

Spring Boot[9] ist grundsätzlich eine Erweiterung des Spring Frameworks, welche das Einrichten einer Spring-Anwendung vereinfacht. Mit Spring Boot sind keine XML-Konfigurationen nötig und alle benötigten Klassenbibliotheken werden mitgeliefert.

6.2.2 Spring Security

Spring Security[10] ist ein Teil des Spring Frameworks und stellt Funktionalitäten wie Autorisierung und Authentisierung zur Verfügung.

Über Spring Security werden die Login Daten und die Session eines eingeloggten Users verwaltet. Für diesen Zweck gibt es einen `UserDetails` Service, welcher die Schnittstelle zu den Login Daten auf der Datenbank bietet. Über eine weitere Klasse, die `UserSecurityConfig`, wird festgelegt, wie das Passwort der Benutzer verschlüsselt ist und welche Rollen auf welche Bereiche Zugriff haben. Spring Security verwendet den `UserDetails` Service um zu schauen, ob der angegebene Benutzer überhaupt in der Datenbank persistiert ist. Über die Angaben im

[7] <https://getbootstrap.com/>
[8] <https://spring.io/>
[9] <https://spring.io/projects/spring-boot>
[10] <https://spring.io/projects/spring-security>

`UserSecurityConfig` führt er danach die Authentisierung durch und bestimmt, worauf dieser User zugreifen kann.

6.2.3 JPA

Die Java Persistence API (JPA) ist eine Möglichkeit, um das OR-Mapping in Java umzusetzen. Mit JPA können Daten von der Datenbank auf Java Objekte gemappt, gespeichert, aktualisiert und gelesen werden. Allerdings ist JPA nur die Spezifikation und die effektive Implementation wäre dann Hibernate, EclipseLink, ApacheOpenJPA, etc. Das Mapping zwischen Java Objekten und Datenbank Tabellen wird über Persistence Metadaten definiert, welche von dem JPA Provider benutzt werden, um die korrekten Datenbank Operationen vorzunehmen. Für den EPJ Team Builder wird JPA mit Hibernate verwendet um die MySQL Datenbank zu Mappen.

6.3 Weitere

6.3.1 Maven

Maven[11] ist ein Build-Automatisierungs-Tool, welches mehrheitlich für Java Projekte benutzt wird. Die Projekte werden über Maven gebuildet, wodurch das JAR File für die Docker Images generiert werden. Weiterhin können alle Dependencies wie Thymeleaf, Spring-Boot und JPA über Maven definiert werden. Ausserdem wird Maven im Projekt eingesetzt, um ein Shared Modul zwischen den beiden Applikationen zu halten, welches direkt neu gebuildet wird, wenn die Applikationen gebuildet werden.

6.3.2 Docker

Docker[12] bietet die Möglichkeit, Applikationen in Pakete zu verpacken und diese dann in einer isolierten Umgebung (Container) laufen zu lassen. Dies ermöglicht es die Applikationen von der Infrastruktur zu separieren und so die Software schneller zu deployen.

In der Gitlab Pipeline wird nach dem Maven build das generierte JAR File in ein Docker Image verpackt. Auf der Produktionsumgebung (Portainer) muss dann nur das Docker-Compose File abgelegt sein und über einen WebHook wird Portainer angewiesen, die Docker Images über docker-compose neu zu pullen und anschliessend zu starten. So ist das ganze Deployment der Applikationen automatisiert.

[11] `<https://maven.apache.org/>`
[12] `<https://www.docker.com/>`

Kapitel 7

Testing

7.1 JUnit

Um die Qualität unserer Logik zu sichern, haben wir uns dazu entschieden, das Backend mit JUnit [8] zu testen. Dafür haben wir den Grossteil der Logik in Services ausgelagert. Für jeden Service haben wir eine JUnit Testklasse erstellt und jede Methode getestet. Unser Ziel war es, für jeden Service mindestens eine Testabdeckung von 70% zu erreichen. Die meisten Testklassen sind im "admin" Modul zu finden da wir dort sowohl die Administrator Services als auch die zentralen Services getestet haben. Wir haben hauptsächlich Integrationstests geschrieben, da es für uns mehr Sinn machte, die ganze Funktionalität zu prüfen anstatt alles zu mocken. Für die Tests haben wir eine In-Memory-Datenbank verwendet. Der Context der Datenbank wird nach jeder Test-Methode wieder zurückgesetzt, sodass man immer mit einer leeren Datenbank startet.

Die Controller verwenden die Services und enthalten sehr wenig Logik. Aus diesem Grund wurden die Controller nicht explizit getestet. Auch für die Views haben wir keine automatisierten Tests geschrieben. Die Funktionalität von diesen beiden Komponenten haben wir jedoch mit den Checklisten aus Kapitel 7.3 sichergestellt.

7.1.1 Testabdeckung

Der Coverage Report wurde am 06.12.2020 direkt mit der Entwicklungsumgebung Intellij IDEA [6] generiert. Wenn man die Ergebnisse mit der angezeigten Coverage auf dem OST Gitlab [4] vergleicht erkennt man schnell, dass die Werte sehr unterschiedlich sind. Dies liegt daran, weil auf Gitlab die Codezeilen von der View und dem JavaScript miteinberechnet werden.

Nachfolgend ist eine Coverage Überblick über die Klassen vom Admin und Shared Package zu finden.

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	84.6% (55/ 65)	79.6% (433/ 544)	74.7% (1320/ 1768)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
dev.ifs.hsr.ch.admin	100% (1/ 1)	50% (1/ 2)	33.3% (1/ 3)
dev.ifs.hsr.ch.admin.controller	100% (6/ 6)	16.3% (8/ 49)	20.1% (49/ 244)
dev.ifs.hsr.ch.admin.dto.graph	85.7% (6/ 7)	81.3% (74/ 91)	86% (154/ 179)
dev.ifs.hsr.ch.admin.dto.group	100% (3/ 3)	93.3% (28/ 30)	96.5% (55/ 57)
dev.ifs.hsr.ch.admin.security	100% (2/ 2)	100% (6/ 6)	100% (22/ 22)
dev.ifs.hsr.ch.admin.service	100% (6/ 6)	100% (39/ 39)	90.2% (248/ 275)
dev.ifs.hsr.ch.admin.service.algorithm	100% (3/ 3)	100% (29/ 29)	99.3% (144/ 145)
dev.ifs.hsr.ch.information	0% (0/ 1)	0% (0/ 2)	0% (0/ 2)
dev.ifs.hsr.ch.information.controller	0% (0/ 1)	0% (0/ 14)	0% (0/ 95)
dev.ifs.hsr.ch.information.security	0% (0/ 2)	0% (0/ 7)	0% (0/ 26)
dev.ifs.hsr.ch.information.service	0% (0/ 1)	0% (0/ 2)	0% (0/ 5)
dev.ifs.hsr.ch.shared.modules.dto	76.9% (10/ 13)	78.9% (75/ 95)	81.4% (140/ 172)
dev.ifs.hsr.ch.shared.modules.dto.validation	0% (0/ 1)	0% (0/ 3)	0% (0/ 8)
dev.ifs.hsr.ch.shared.modules.entity	100% (11/ 11)	98% (99/ 101)	95.9% (188/ 196)
dev.ifs.hsr.ch.shared.modules.serializer	100% (1/ 1)	100% (3/ 3)	100% (22/ 22)
dev.ifs.hsr.ch.shared.modules.service	100% (6/ 6)	100% (71/ 71)	93.7% (297/ 317)

generated on 2020-12-06 10:03

Abbildung 7.1: Testabdeckung Admin und Library Package

Wie bereits erwähnt haben wir beim Testen den Fokus auf die Klassen im Service Package gelegt. Deshalb wir die Testabdeckung aller Service Klassen noch genauer aufgelistet.

Coverage Summary for Package: dev.ifs.hsr.ch.shared.modules.service

Package	Class, %	Method, %	Line, %
dev.ifs.hsr.ch.shared.modules.service	100% (6/ 6)	100% (71/ 71)	93.7% (297/ 317)

Class	Class, %	Method, %	Line, %
Database	100% (1/ 1)	100% (13/ 13)	100% (19/ 19)
ServiceConfig	100% (1/ 1)	100% (1/ 1)	100% (1/ 1)
SystemConfigService	100% (1/ 1)	100% (5/ 5)	92.6% (25/ 27)
TagService	100% (1/ 1)	100% (3/ 3)	100% (3/ 3)
UserService	100% (1/ 1)	100% (39/ 39)	92.4% (219/ 237)
UserUtil	100% (1/ 1)	100% (10/ 10)	100% (30/ 30)

generated on 2020-12-06 10:03

Abbildung 7.2: Testabdeckung Shared Services

Coverage Summary for Package: dev.ifs.hsr.ch.admin.service

Package	Class, %	Method, %	Line, %
dev.ifs.hsr.ch.admin.service	100% (6/ 6)	100% (39/ 39)	90.2% (248/ 275)

Class	Class, %	Method, %	Line, %
AdminDetailsServiceImpl	100% (1/ 1)	100% (2/ 2)	100% (5/ 5)
AdminService	100% (1/ 1)	100% (6/ 6)	76.9% (30/ 39)
BulkImporterService	100% (1/ 1)	100% (10/ 10)	88.9% (128/ 144)
GraphService	100% (1/ 1)	100% (6/ 6)	88.2% (15/ 17)
GroupService	100% (1/ 1)	100% (13/ 13)	100% (62/ 62)
MapUtil	100% (1/ 1)	100% (2/ 2)	100% (8/ 8)

generated on 2020-12-06 10:03

Abbildung 7.3: Testabdeckung Admin Services

Coverage Summary for Package: dev.ifs.hsr.ch.information.service

Package	Class, %	Method, %	Line, %
dev.ifs.hsr.ch.information.service	100% (1/ 1)	100% (2/ 2)	100% (5/ 5)

Class	Class, %	Method, %	Line, %
UserDetailsServiceImpl	100% (1/ 1)	100% (2/ 2)	100% (5/ 5)

generated on 2020-12-06 10:17

Abbildung 7.4: Testabdeckung Info Services

7.2 Gruppenbildung

Zu Beginn des Meilensteins ”M5 - Release” hat jeder von uns versucht, mit den Daten aus der Datenerfassung Gruppen zu bilden. Dadurch konnten wir uns besser in die Rolle des EPJ Organisators versetzen und herausfinden, welche Funktionalitäten wirklich sinnvoll sind und welche noch fehlen. Die daraus gewonnenen Erkenntnisse haben wir anschliessend miteinander besprochen und direkt im Meilenstein ”M5 - Release” verbessert.

7.3 Checkliste

Da wir mit JUnit nur die Backend Funktionalität getestet haben, haben wir uns dazu entschieden, verschiedene Checklisten aufzusetzen, um die Qualität und die komplette Funktionalität des Programms zu überprüfen. Diese Checklisten wurden nach Abschluss der Construction Phase durchgeführt und die Ergebnisse festgehalten.

Datum	Betriebssystem	Browser	Version
02.12.2020	Windows	Chrome	87.0.4280.66
02.12.2020	Windows	Firefox	81.0.2
02.12.2020	iOS	Safari	14

Tabelle 7.1: Durchgeführte Überprüfungen

Wir haben die Checklisten mit drei verschiedenen Browsern durchgeführt. Aufgrund von Platzmangel haben wir die Resultate nur für Chrome und Safari angezeigt. Alle Checklisten

waren aber auch mit Firefox erfolgreich.

Folgende URL werden für die verschiedenen Bereiche verwendet:

- Studenten: https://epjteambuilder-form.dev.ifs.hsr.ch/student
- Betreuer: https://epjteambuilder-form.dev.ifs.hsr.ch/supervisor
- Administrator: https://epjteambuilder-dashboard.dev.ifs.hsr.ch/

7.3.1 Information Formular

Mit nachfolgender Checkliste haben wir die An-/Abmeldung und die Verwaltung des Benutzerprofils getestet. Als Voraussetzungen für diese Tests müssen bereits Benutzer vorhanden sein. Diese Checkliste wurde einmal für einen Studenten und einmal für einen Betreuer durchgeführt.

Nr.	**Ablauf**	**Erwartetes Resultat**	**Resultat Chrome**	**Resultat Safari**
1	Benutzer mit ungültiger E-Mail-Adresse anmelden.	Login funktioniert nicht	✓	✓
2	Benutzer mit ungültigem Passwort anmelden.	Login funktioniert nicht	✓	✓
3	Benutzer mit gültigen Daten einloggen. Studenten werden auf den Studenten Bereich [12] und Betreuer auf den Betreuer Bereich [3] weitergeleitet.	Login funktioniert	✓	✓
4	Passwort ändern und ein ungültiges, altes Passwort eingeben.	Passwort wird nicht geändert	✓	✓
5	Passwort ändern und das neue Passwort und die Bestätigung leer lassen.	Passwort wird nicht geändert	✓	✓
6	Passwort ändern und zwei unterschiedliche Passwörter für das neue Passwort und die Bestätigung eingeben.	Passwort wird nicht geändert	✓	✓
7	Passwort ändern mit gültigen Eingaben.	Passwort wird geändert	✓	✓
8	Logout und mit neuem Passwort anmelden.	Login funktioniert	✓	✓
9	Logout und versuchen den Studenten Bereich [12] und den Betreuer Bereich [3] zu öffnen.	Redirect auf Login Page	✓	✓

Tabelle 7.2: Checkliste - Information Formular 1

In der nächsten Checkliste wurde die Datenerfassung getestet. Auch diese Checkliste wurde einmal für einen Studenten und einmal für einen Betreuer durchgeführt.

Nr.	Ablauf	Erwartetes Resultat	Resultat Chrome	Resultat Safari
1	Daten für den Benutzer erfassen und speichern. In jedem Feld soll mindestens 1 Wert eingetragen werden.	Daten wurden gespeichert	✓	✓
2	Jede Auswahlliste öffnen und prüfen, ob die Suche funktioniert. Dafür einfach einen Text eingeben.	Suche funktioniert	✓	✓
3	Suche in der Themen Auswahlliste nach einer Option, welche es noch nicht gibt. Füge die Option hinzu. Die Option wird im Feld und in der Auswahlliste angezeigt. Speichere die Eingabe.	Daten wurden gespeichert und es wurde eine neue Option für die Themen erstellt.	✓	✓
4	Suche in der Technologien Auswahlliste nach einer Option, welche es noch nicht gibt. Füge die Option hinzu. Die Option wird im Feld und in der Auswahlliste angezeigt. Speichere die Eingabe.	Daten wurden gespeichert und es wurde eine neue Option für die Technologien erstellt.	✓	✓
5	Logout und erneut anmelden. Die erfassten Daten werden geladen.	Daten werden geladen	✓	✓
6	Als angemeldeter Student versuchen auf den Studenten Bereich [12] zugreifen.	403 Fehlermeldung	✓	✓
7	Als angemeldeter Betreuer versuchen auf den Betreuer Bereich [3] zugreifen.	403 Fehlermeldung	✓	✓

Tabelle 7.3: Checkliste - Information Formular 2

7.3.2 Administrator Dashboard

Die nachfolgenden Checklisten befassen sich alle mit den Funktionalitäten des Administrators. In einem ersten Schritt wird die An-/Abmeldung und die Verwaltung des Benutzerprofils getestet. Als Voraussetzung muss mindestens ein Administrator in der Datenbank vorhanden sein.

Nr.	Ablauf	Erwartetes Resultat	Resultat Chrome	Resultat Safari
1	Administrator mit ungültiger E-Mail-Adresse anmelden.	Login funktioniert nicht	✓	✓
2	Administrator mit ungültigem Passwort anmelden.	Login funktioniert nicht	✓	✓
3	Administrator mit gültiger E-Mail und Passwort einloggen.	Login funktioniert	✓	✓

4	Passwort ändern und ein ungültiges, altes Passwort eingeben.	Passwort wird nicht geändert	✓	✓
5	Passwort ändern und das neue Passwort und die Bestätigung leer lassen.	Passwort wird nicht geändert	✓	✓
6	Passwort ändern und zwei unterschiedliche Passwörter für das neue Passwort und die Bestätigung eingeben.	Passwort wird nicht geändert	✓	✓
7	Passwort ändern mit gültigen Eingaben.	Passwort wird geändert	✓	✓
8	Logout und mit neuem Passwort anmelden.	Login funktioniert	✓	✓
9	Logout und versuchen den Administrator Bereich [1] zu öffnen.	Redirect auf Login Page	✓	✓

Tabelle 7.4: Checkliste - Administrator Dashboard

Administration

Als nächstes werden alle Funktionalitäten geprüft, welche in der Navigation unter ”Administration” zu finden sind.

Nr.	**Ablauf**	**Erwartetes Resultat**	**Resultat Chrome**	**Resultat Safari**
1	Student hinzufügen mit leerem Input.	Hinzufügen funktioniert nicht	✓	✓
2	Student hinzufügen mit ungültigem Schema (z.B. ”test”).	Hinzufügen funktioniert nicht	✓	✓
3	Student hinzufügen mit gültigem Schema (z.B. ”max.mustermann@ost.ch”).	Hinzufügen funktioniert	✓	✓
4	Gleicher Student nochmals hinzufügen (z.B. ”max.mustermann@ost.ch”).	Hinzufügen funktioniert nicht	✓	✓
5	Mehrere Studenten gleichzeitig hinzufügen (z.B. ”erika.mustermann@ost.ch, paul.richter@ost.ch”).	Hinzufügen funktioniert	✓	✓
6	Den Studenten Bereich [12] öffnen und sich mit einem zuvor erstellten Studenten anmelden (Passwort ist identisch zur E-Mail-Adresse).	Login funktioniert	✓	✓
7	Betreuer hinzufügen mit leerem Input.	Hinzufügen funktioniert nicht	✓	✓

8	Betreuer hinzufügen mit ungültigem Schema (z.B. ”test”).	Hinzufügen funktioniert nicht	✓	✓
9	Betreuer hinzufügen mit gültigem Schema (z.B. ”paul.mahrer@ost.ch”).	Hinzufügen funktioniert	✓	✓
10	Gleicher Betreuer nochmals hinzufügen (z.B. ”paul.mahrer@ost.ch”).	Hinzufügen funktioniert nicht	✓	✓
11	Mehrere Betreuer gleichzeitig hinzufügen (z.B. ”manuel.müller@ost.ch, laura.schneider@ost.ch”).	Hinzufügen funktioniert	✓	✓
12	Den Betreuer Bereich [3] öffnen und sich mit einem zuvor erstellten Betreuer anmelden (Passwort ist identisch zur E-Mail-Adresse).	Login funktioniert	✓	✓
13	Benutzer entfernen mit leerem Input.	Entfernen funktioniert nicht.	✓	✓
14	Einen zuvor erstellten Studenten entfernen. Den Studenten Bereich [12] öffnen und sich mit dem entfernten Studenten versuchen anzumelden.	Student wurde entfernt und Login funktioniert nicht mehr.	✓	✓
15	Einen zuvor erstellten Betreuer entfernen. Den Betreuer Bereich [3] öffnen und sich mit dem entfernten Betreuer versuchen anzumelden.	Betreuer wurde entfernt und Login funktioniert nicht mehr.	✓	✓
16	Anmelden mit einem Studenten. Das Passwort ändern (z.B. ”1111”) und abmelden. Zurück in den Admin Bereich wechseln und das Passwort für den Studenten zurücksetzen. Erneut mit dem Studenten anmelden.	Passwort wurde zurückgesetzt. (Passwort = E-Mail)	✓	✓
17	Anmelden mit einem Betreuer. Das Passwort ändern (z.B. ”1111”) und abmelden. Zurück in den Admin Bereich wechseln und das Passwort für den Betreuer zurücksetzen. Erneut mit dem Betreuer anmelden.	Passwort wurde zurückgesetzt. (Passwort = E-Mail)	✓	✓

18	In die Gruppenübersicht Ansicht wechseln und 3 Gruppen erstellen. Gruppe 1 mit Studenten und einem Betreuer, Gruppe 2 mit Studenten aber ohne Betreuer und Gruppe 3 mit einem Betreuer aber ohne Studenten. Nun wieder in die Administrator Übersicht wechseln und die Datenbank exportieren.	Datenbank wird in eine JSON Datei exportiert.	✓	✓
19	Datenbank importieren, ohne eine Datei auszuwählen.	Import wird nicht ausgeführt.	✓	✓
20	Eine leere JSON Datei auswählen und den Datenbank Import ausführen.	Import funktioniert nicht, bestehende Daten bleiben erhalten.	✓	✓
21	Die zuvor exportierte Datei auswählen und den Datenbank Import ausführen.	Import funktioniert, bestehende Daten werden überschrieben.	✓	✓
22	Datenbank leeren	Alle Daten werden gelöscht bis auf die Technologie, Themen und Administrator Datenbank.	✓	✓
23	Datenerfassung sperren. Anmelden mit einem Studenten [12] und versuchen, Daten zu bearbeiten.	Daten können nicht bearbeitet werden.	✓	✓
24	Datenerfassung sperren. Anmelden mit einem Betreuer [3] und versuchen, Daten zu bearbeiten.	Daten können nicht bearbeitet werden.	✓	✓
25	Datenerfassung sperren. Als Administrator versuchen die Daten von einem Studenten und einem Betreuer zu bearbeiten.	Daten können bearbeitet werden.	✓	✓
26	Datenerfassung wieder aktivieren. Anmelden mit einem Studenten [12] und versuchen, Daten zu bearbeiten.	Daten können bearbeitet werden.	✓	✓
27	Datenerfassung wieder aktivieren. Anmelden mit einem Betreuer [3] und versuchen, Daten zu bearbeiten.	Daten können bearbeitet werden.	✓	✓

Tabelle 7.5: Checkliste - Administration

Als Voraussetzung für alle nachfolgenden Checklisten müssen einige sinnvolle Benutzer vorhanden sein. Das heisst Benutzer, welche Verfügbarkeiten, Team Präferenzen, Technologien und Themen erfasst haben.

Übersicht

In der nächsten Checkliste werden alle Übersichtsseiten getestet.

Nr.	Ablauf	Erwartetes Resultat	Resultat Chrome	Resultat Safari
1	Gruppe in der Gruppenübersicht mit Studenten aber ohne Betreuer erstellen.	Gruppe wird gespeichert und im Bereich "Gruppen ohne Betreuer" in der Gruppenübersicht angezeigt. Wenn man in die Studentenübersicht wechselt, werden die Studenten, welche der Gruppe zugewiesen wurden, nicht mehr im Bereich "Studenten ohne Gruppe" angezeigt.	✓	✓
2	Gespeicherte Gruppe editieren und einen Betreuer hinzufügen.	Gruppe wird gespeichert und ist im Bereich "Gruppen ohne Betreuer" in der Gruppenübersicht nicht mehr aufgelistet. Wenn man in die Betreuerübersicht wechselt, wird die Gruppe beim zugewiesenen Betreuer angezeigt.	✓	✓
3	Gruppe mit verschiedenen Benutzern erstellen und darauf achten, dass es Übereinstimmungen bei den Themen und Technologien gibt. Die erstellte Gruppe erneut öffnen.	Im Gruppendetail Dialog werden die gemeinsamen Themen und Technologien angezeigt.	✓	✓
4	Gruppe mit einem Betreuer und ohne Studenten erstellen. Die erstellte Gruppe öffnen.	Im Gruppendetail Dialog wird in dem Bereich, in dem die Zeiten an denen sowohl Betreuer als auch Studenten verfügbar sind, die Verfügbarkeiten des Betreuers angezeigt. Der Bereich für die Verfügbarkeiten, an denen nur die Studenten Zeit haben, ist leer.	✓	✓
5	Die zuvor erstellte Gruppe editieren und Studenten hinzufügen, welcher eine gemeinsame Verfügbarkeit mit dem Betreuer haben. Die bearbeitete Gruppe erneut öffnen.	Im Gruppendetail Dialog werden die gemeinsamen verfügbaren Zeiten von den Studenten und dem Betreuer angezeigt. Zusätzlich werden die Zeiten angezeigt, an denen nur die Studenten verfügbar sind.	✓	✓

6	Die Gruppe nochmals editieren und den Betreuer entfernen.	Im Gruppendetail Dialog werden nur noch die gemeinsamen Verfügbarkeiten von den Studenten angezeigt. Der Bereich, in dem die gemeinsamen Zeiten von allen Benutzern angezeigt wird, bleibt leer.	✓	✓
7	Eine Gruppe löschen.	Gruppe wir gelöscht und nicht mehr angezeigt.	✓	✓
8	Auf allen Übersichtsseiten die ”Gruppen erstellen” und ”Gruppen editieren” Buttons ausprobieren.	Die Verlinkungen stimmen und der Gruppendetail Dialog wird geöffnet.	✓	✓

Tabelle 7.6: Checkliste - Administrator Übersicht

Timetables

Die nachfolgende Checkliste überprüft die Funktionalitäten in der ”Timtables” Ansicht.

Nr.	**Ablauf**	**Erwartetes Resultat**	**Resultat Chrome**	**Resultat Safari**
1	Stundenplan anzeigen lassen, ohne etwas selektiert zu haben.	Es wird eine leere Timeline angezeigt.	✓	✓
2	Stundenplan für Studenten und Betreuer anzeigen lassen.	Für jeden ausgewählten Benutzer werden die korrekten Verfügbarkeiten angezeigt.	✓	✓
3	Die angezeigten Reihen für die ausgewählten Benutzer mit Drag & Drop verschieben.	Reihen können verschoben werden.	✓	✓
4	Stundenplan für eine Gruppe anzeigen lassen.	Für die Gruppe werden die Verfügbarkeiten angezeigt. Die Zeiten, an denen nur die Studenten verfügbar sind, werden in einer anderen Farbe dargestellt als die Zeiten, an denen sowohl die Studenten als auch der Betreuer verfügbar ist.	✓	✓
5	Eine angezeigte Gruppe editieren und eine Änderung an den Benutzern vornehmen.	Die Studenten, Betreuer und Gruppen bleiben ausgewählt. Die Verfügbarkeiten für die bearbeitete Gruppe werden aktualisiert.	✓	✓

6	Eine neue Gruppe erstellen.	Die Studenten, Betreuer und Gruppen bleiben ausgewählt. Die neue Gruppe wurde als Option bei der Gruppenauswahl hinzugefügt.	✓	✓
7	In den Dropdowns können alle Optionen gleichzeitig aus- oder abgewählt werden.	Alle auswählen / abwählen funktioniert.	✓	✓

Tabelle 7.7: Checkliste - Timetable

Smart Data Visualisation

Als nächstes wird die ”Smart Data Visualisation” geprüft.

Nr.	**Ablauf**	**Erwartetes Resultat**	**Resultat Chrome**	**Resultat Safari**
1	Einstellungen beliebig anpassen und den Graphen neu zeichnen lassen, indem man das Layout auswählt.	Graph wird neu gezeichnet und die Einstellungen werden beachtet.	✓	✓
2	Die Positionierung der Knoten manuell anpassen. Die Einstellungen nochmals ändern und anschliessend die Kanten aktualisieren.	Kanten werden aktualisiert, ohne die Positionierung der Knoten zu verändern.	✓	✓
3	Den Schritt rückgängig machen.	Die letzte Änderung der Einstellungen wird rückgängig gemacht und die Kanten werden aktualisiert.	✓	✓
4	Betreuer ausblenden	Die Betreuer und deren Kanten werden ausgeblendet, ohne den Graph neu zu zeichnen.	✓	✓
5	Studenten, die einer Gruppe zugeordnet sind, ausblenden.	Die Studenten in einer Gruppe werden ausgeblendet, ohne den Graph neu zu zeichnen.	✓	✓
6	Alle Kanten ausblenden die ein tieferes Kante Gewicht als einen beliebigen Wert haben.	Die Kanten mit einem tieferen Kanten Gewicht werden ausgeblendet, ohne den Graph neu zu zeichnen.	✓	✓
7	Einen Knoten mit einer ungültigen E-Mail suchen.	Es passiert nichts	✓	✓
8	Einen Knoten mit einer vorhandenen E-Mail suchen.	Der gefundene Knoten wird selektiert und angezeigt.	✓	✓

9	Beliebige Sachen bei den Einstellungen und Filter definieren. Graph neu anzeigen lassen und Knoten manuell verschieben. Die Positionierung des Graphen und alle Angaben speichern.	Daten zum Graph werden gespeichert.	✓	✓
10	Gespeicherte Graph Daten laden.	Der Graph und alle Angaben werden geladen. Alles sieht so aus wie zu dem Zeitpunkt, als die Daten gespeichert wurden.	✓	✓
11	Beliebige Knoten auswählen und daraus eine neue Gruppe erstellen. Die ausgewählten Daten werden in den Gruppendetail Dialog übernommen. Die Gruppe mit mindestens einem Studenten speichern.	Nach dem Speichern wird die Seite nicht neu geladen. Der Knoten vom Studenten, welcher der Gruppe hinzugefügt wurde, wird grau eingefärbt.	✓	✓

Tabelle 7.8: Checkliste - Smart Data Visualisation

Übersicht Datenerfassung

Als Abschluss werden die Visualisierung und Bearbeitung der Benutzerdaten geprüft.

Nr.	**Ablauf**	**Erwartetes Resultat**	**Resultat Chrome**	**Resultat Safari**
1	Eingegebene Daten von einem Studenten anzeigen lassen.	Alle Daten werden korrekt geladen.	✓	✓
2	Eingegebene Daten von einem Betreuer anzeigen lassen.	Alle Daten werden korrekt geladen.	✓	✓
3	Daten von einem Studenten ändern.	Änderung wird gespeichert.	✓	✓
4	Daten von einem Betreuer ändern.	Änderung wird gespeichert.	✓	✓

Tabelle 7.9: Checkliste - Administrator Übersicht Datenerfassung

Kapitel 8

Handbuch

8.1 Installationsanleitung

8.1.1 Einrichten der Applikation

Die Applikation ist so konstruiert, dass Sie in einem Docker Container läuft. Daher muss das `docker-compose.yml` aus dem Repository auf dem Server abgelegt werden und dann mit dem Befehl `docker compose up` laufen gelassen werden. Das Docker Compose pullt nun die angegebenen Docker Images für das Information Formular und das Administrator Dashboard von der angegebenen URL. Danach wird ein Docker Container für die Applikationen gemeinsam mit dem Datenbank Server erstellt. Die Applikation erstellt beim Startup auch gleich die Datenbank auf dem DB Server.

8.1.2 Datenbank abfüllen

Nachdem die Applikationen und die Datenbank laufen, muss man noch ein initiales SQL Skript[1] auf der Datenbank ausführen. Mit diesem Skript werden die initialen Themen und Technologien sowie auch der Administrator für das Admin Dashboard erstellt.

8.2 Benutzerhandbuch

Das Benutzerhandbuch dient dazu, die Funktionsweise der Web-Applikation detailliert zu beschreiben. Ausserdem soll es eine Hilfestellung bieten, wie man bei der Gruppenbildung vorgehen kann.

8.2.1 Vorbereitung

Starten wir mit den Administrator-Funktionalitäten des EPJ Organisators, welche vor allem bei der Vorbereitung wichtig sind.

[1] <https://gitlab.ost.ch/epj-team-builder/documentation/-/blob/master/software-documentation/04-manual/input.sql>

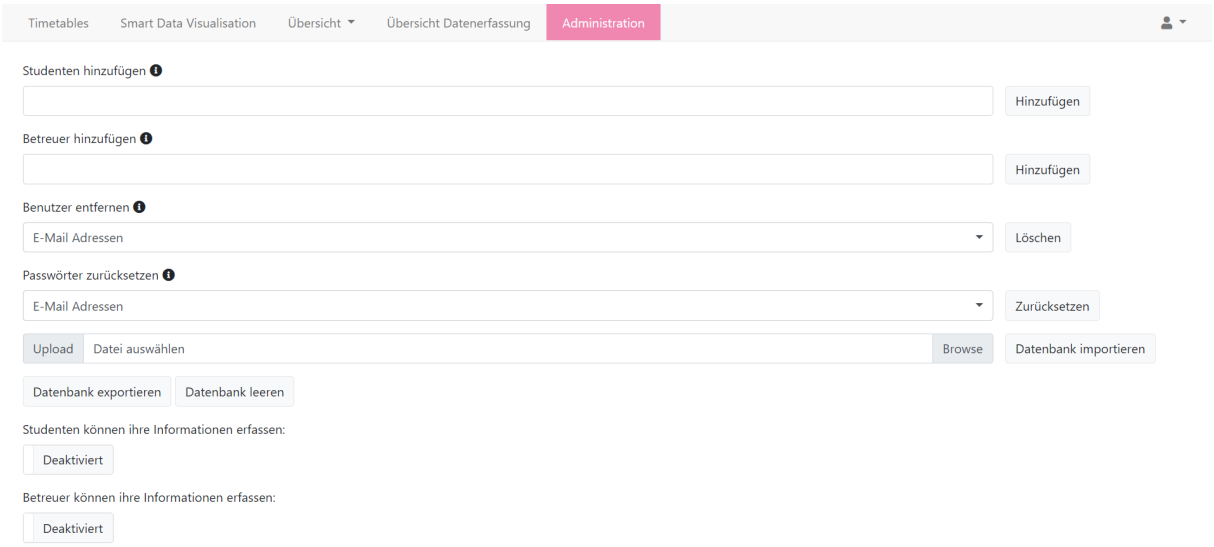

Abbildung 8.1: Administration Funktionalitäten

Datenbank leeren

Zu Beginn jeder Gruppenbildung müssen zuerst die Daten vom Vorjahr zurückgesetzt werden. Mithilfe der Funktion "Datenbank leeren" werden so gut wie alle Datensätze von der Datenbank gelöscht. Die einzigen Ausnahmen sind die "Technologie", "Themen" und "Administrator" Datensätze.

Studenten / Betreuer hinzufügen

Als nächstes werden die Studenten und Betreuer hinzugefügt, welche am Engineering Project teilnehmen oder eine Gruppe betreuen. Dafür können in den Bereichen "Studenten hinzufügen" und "Betreuer hinzufügen" mehrere mit Komma getrennten E-Mail-Adressen eingegeben werden. Für jede E-Mail wird ein neuer Benutzer erstellt, vorausgesetzt es gibt noch keinen mit dieser E-Mail. Das Passwort des neuen Benutzers ist identisch zu seiner E-Mail.

Datenerfassung de-/aktivieren

Sobald diese zwei Schritte erledigt sind, kann der Administrator die Erfassung der Daten für die Studenten und die Betreuer aktivieren. Nun sind die Benutzer berechtigt ihre Daten zu erfassen und zu bearbeiten. Wenn der Organisator mit der Planung der Gruppen beginnt, kann er die Berechtigung für die Erfassung wieder deaktivieren. Die Betreuer und Studenten können dann ihre Daten nur noch im Read-Only Modus anschauen. Falls jemand eine Anpassung seiner Daten wünscht, muss mit dem Administrator Kontakt aufgenommen werden. Dieser kann nach wie vor alle Eingaben der Benutzer anpassen.

Benutzer entfernen / Passwort zurücksetzen

Falls ein Benutzer sich im Nachhinein doch noch für das EPJ abmeldet oder eine E-Mail falsch erfasst wurde, kann dieser über "Benutzer entfernen" von der Datenbank gelöscht werden. Ausserdem gibt es auch die Möglichkeit, ein Passwort auf die E-Mail-Adresse zurückzusetzen, falls jemand sein Passwort vergessen hat. Bei beiden Funktionen können mehrere Benutzer gleichzeitig ausgewählt werden.

Datenbank importieren / exportieren

Falls der Administrator während der Planung einen aktuellen Stand exportieren möchte, kann er dies über "Datenbank exportieren" machen. Es wird ein JSON mit allen Daten exportiert. Zu einem späteren Zeitpunkt kann er die Datenbank wieder importieren. Es ist wichtig, dass die zu importierende JSON Datei die gleiche Struktur wie beim Exportieren aufweist.

Beim Import werden die Datensätze in den Tabellen mittels `TRUNCATE` gelöscht, bevor anschliessend die Werte aus dem JSON eingelesen werden. Sobald ein `TRUNCATE` Befehl ausgeführt wird, kann die Ausführung nicht mehr rückgängig gemacht werden, auch wenn auf der Transaktion ein Rollback gemacht wird. Aus diesem Grund ist es möglich, dass die Datenbank nach dem Import leer ist, falls es beim Hinzufügen der Werte aus dem JSON über den SQL `INSERT` Befehl eine Exception gab. Um das Risiko für dieses Problem zu minimieren ist der Ablauf wie folgt:

1. Das JSON wird zu Beginn geparst. Wenn die Datei eine falsche Struktur aufweist, wird hier bereits ein Error generiert. In diesem Fall wird ein Rollback der Transaktion gemacht und die Daten gehen nicht verloren.
2. Anschliessend wird ein Export von der ganzen Datenbankstruktur ausgeführt und das Ergebnis in eine Variable geschrieben.
3. Nun kommt der eigentliche Import ins Spiel. Zuerst werden die `TRUNCATE` Statements und anschliessend alle `INSERT` Befehle ausgeführt. Wenn es hier irgendwo zu einer Exception kommt, wird diese abgefangen. In diesem Fall wird ein Rollback der Transaktion ausgeführt und es wird versucht, das Ergebnis des zuvor ausgeführten Exports zu importieren, um so den alten Stand der Datenbank wiederherzustellen. Falls die Datenbank zu dieser Zeit aus irgendeinem Grund unavailable wird, kann die Datenbank nicht wiederhergestellt werden.
4. Um das Risiko für den Verlust der Daten möglichst gering zu halten, empfehlen wir zusätzlich vor jedem Import sicherheitshalber einen Export durchzuführen.

8.2.2 Datenerfassung

Nachdem die Benutzer vom Organisator hinzugefügt wurden, können Sie sich mit ihrer E-Mail-Adresse und dem Passwort, welches identisch ist zur E-Mail, einloggen. Es macht Sinn, wenn man am Anfang direkt das Passwort ändert. Sobald der Administrator die Datenerfassung freigegeben hat, können die Daten eingegeben und bearbeitet werden.

Datenerfassung Student

Die Datenerfassung würde für die Studenten wie folgt aussehen.

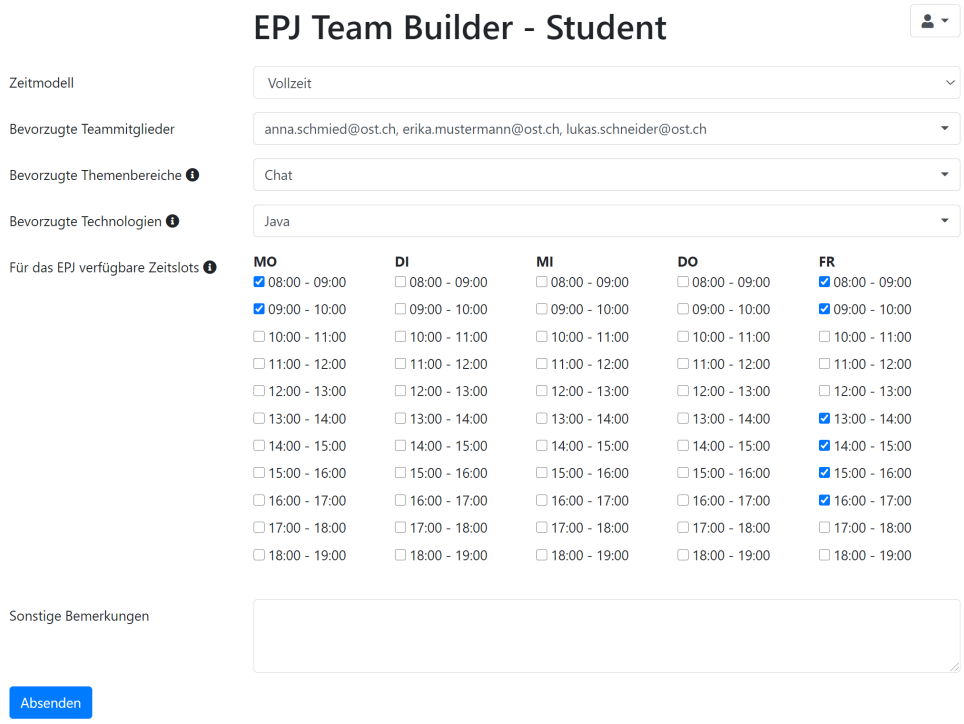

Abbildung 8.2: Datenerfassung Student

Datenerfassung Betreuer

Die Ansicht für die Betreuer unterscheidet sich minimal.

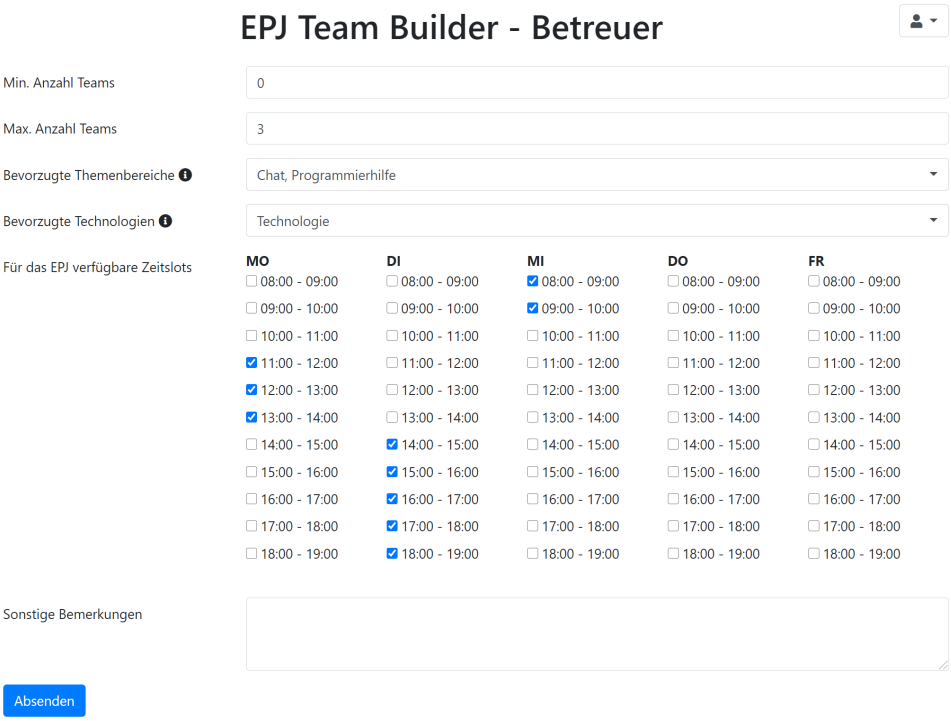

Abbildung 8.3: Datenerfassung Betreuer

Neue Option

Wenn für den Themenbereich oder die Technologien eine Option nicht existiert, kann diese über "Option hinzufügen" hinzugefügt werden. Nachdem die Daten abgesendet wurden, wird ein neuer Eintrag in der Themen oder Technology Tabelle erfasst, sodass diese Option allen anderen Benutzern ebenfalls zur Verfügung steht.

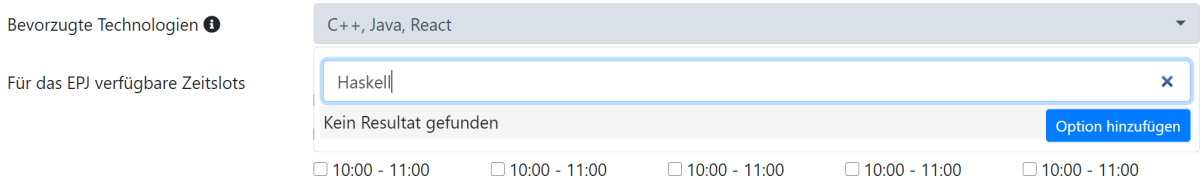

Abbildung 8.4: Option hinzufügen

8.2.3 Planung

Sobald alle Benutzer ihre Daten erfasst haben, kann der Organisator mit der Planung der Gruppen starten. Dafür deaktiviert er am besten die Datenbearbeitung, sodass die Studenten und Betreuer keine Anpassungen mehr vornehmen können.

Smart Data Visualisation

Die ganze Planung startet man am einfachsten in der "Smart Data Visualisation" Ansicht da man hier Gruppenbildungen anhand der Kanten erkennen kann. Für jeden Benutzer gibt es einen Knoten. Darin steht ein Kürzel, welcher aus dem Anfangsbuchstaben vom Vornamen und den ersten zwei Buchstaben vom Nachnamen zusammengesetzt wird.

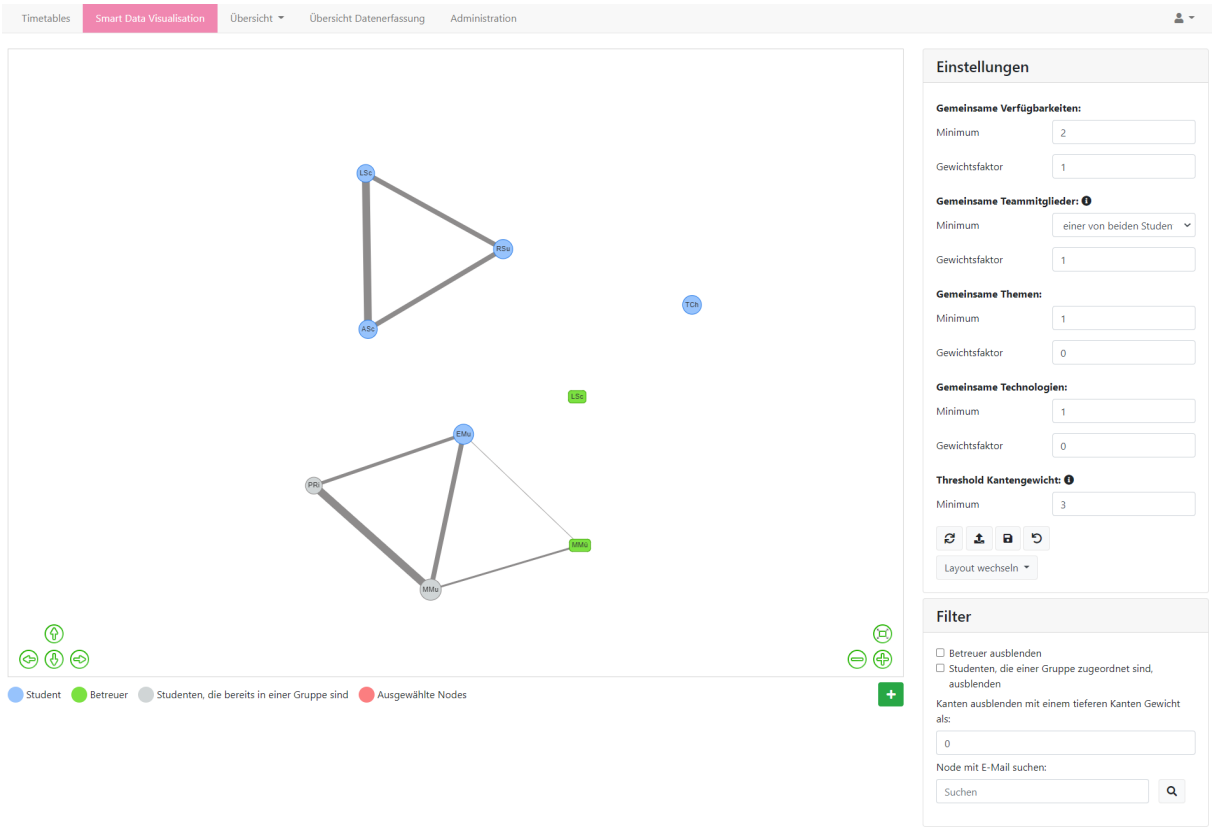

Abbildung 8.5: Smart Data Visualisation

Einstellungen

Über die Einstellungen kann definiert werden, ob zwischen zwei Knoten eine Kante erstellt wird oder nicht. Eine Kante wird nur dann erstellt, wenn das Minimum der Übereinstimmung pro Einstellung erfüllt ist und das Kantengewicht mindestens so gross ist wie der Wert im Feld "Threshold Kantengewicht". Das Kantengewicht berechnet sich aus den Gemeinsamkeiten pro Einstellung und dem dazugehörigen Gewichtsfaktor. Bei der Einstellung für die gemeinsamen Teammitglieder hat "keine Übereinstimmung" den Wert 0, "einer von beiden will gemeinsam im Team sein" den Wert 1 und "beide Studenten wollen miteinander im Team sein" den Wert 2.

Wenn die Einstellungen angepasst wurden, kann über den Button "Kanten aktualisieren" die Kanten neu geladen werden ohne dass die Knoten um positioniert werden. Falls man den Graphen komplett neu zeichnen will, kann man über "Layout wechseln" ein Layout auswählen. Dafür stehen folgende Layouts zur Verfügung:

- barnesHut
- forceAtlas2Based
- repulsion

Die verwendete Graph-Library vis.js[2] bietet diese Layouts an, um die Positionierung der Knoten zu steuern. Die ersten zwei Layouts arbeiten mit Gravitationseinstellungen und das dritte Layout mit Kräftefeldern zwischen den Nodes.

Über den Speicher Button hat man die Möglichkeit, die Positionierung der Nodes und alle definierten Einstellungen und Filter zu speichern. Dies ist vor allem dann sinnvoll, wenn man die Positionierung manuell optimiert hat und zu einem späteren Zeitpunkt damit weiterarbeiten will. Wenn man den Graphstatus wieder laden möchte, kann man dies über den Laden Button machen. Es werden alle Einstellungen und Filter geladen und die Nodes werden dort positioniert, wo sie vor dem Speichern waren.

Falls man mal eine Änderung macht, mit der man nicht zufrieden ist, kann man die letzte Anpassung von der aktuellen Session rückgängig machen. Die Anpassungen von den Einstellungen und Filtern werden erst in die Session geschrieben, wenn man die Kanten aktualisiert oder einen Graphstatus lädt. Ausserdem wird die Knoten Position in die Session geschrieben, sobald ein Node manuell verschoben wird. Wenn die Seite komplett neu geladen wird, wie zum Beispiel bei der "Layout wechseln" Funktion, wird die Session neu aufgesetzt und vorherige Anpassungen können nicht mehr rückgängig gemacht werden.

Filter

Mit den Filtern kann der Graph angepasst werden, ohne dass die Knoten neu positioniert werden. Man kann die Betreuer oder die Studenten, welche bereits einer Gruppe zugeordnet sind, ausblenden. Ausserdem kann man Kanten mit einem kleineren Kantengewicht als angegeben ausblenden.

Falls man einen Knoten suchen möchte, kann man die E-Mail-Adresse eingeben. Wenn ein Knoten mit exakt der eingegebenen E-Mail-Adresse gefunden wurde, wir er ausgewählt, rot markiert und der Graph wird zum Node herangezoomt. Falls kein passender Knoten gefunden wurde passiert nichts.

Neue Gruppe

[2] <https://visjs.org/>

In der Smart Data Visualisation Ansicht hat man ausserdem die Möglichkeit, neue Gruppen über den Plus Button zu erstellen. Im Graph kann man mit Ctrl mehrere Nodes gleichzeitig auswählen, welche dann für die neue Gruppe vorselektiert werden.

Timetables

Nachdem man die ersten Gruppen über die Smart Data Visualisation erstellt hat, können hier gruppenlose Studenten und Betreuer den existierenden Gruppen zugewiesen werden.

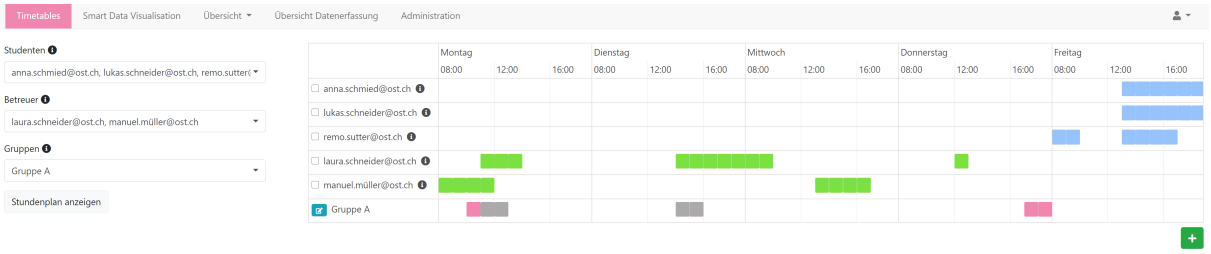

Abbildung 8.6: Timetables

Die verfügbaren Zeitslots der Studenten werden blau und die der Betreuer werden grün angezeigt. Die Zeiten für die Gruppen werden in zwei unterschiedlichen Farben angezeigt. Grau steht für die Zeiten, an denen sowohl der Betreuer als auch die Studenten Zeit haben. Die Zeiten, an denen nur die Studenten Zeit haben, werden pink markiert.

Die einzelnen Reihen können per Drag & Drop verschoben werden.

Jede Gruppe kann direkt über den Edit Button bearbeitet werden. Falls man eine neue Gruppe erstellen möchte, ist dies über den Plus Button möglich. Wenn man bei den Studenten und Betreuern die Checkbox aktiviert hat, werden diese in der neuen Gruppe vorselektiert.

Übersicht

Auf den Übersichtsseiten erhält man eine Übersicht, welche Gruppen es alle gibt und ob die Gruppenbildung gültig ist.

Gruppenübersicht

Auf dieser Übersichtsseite sieht man alle Gruppen und ihre Mitglieder. Ausserdem werden die Gruppen, welche noch keinen Betreuer haben, aufgelistet. Auf dieser Seite können neue Gruppen erstellt, gelöscht oder bearbeitet werden.

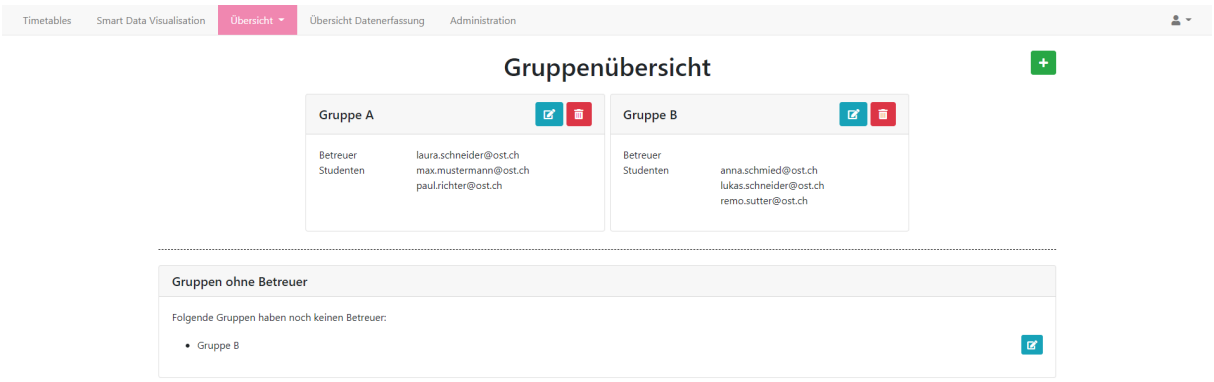

Abbildung 8.7: Gruppenübersicht

Studentenübersicht

Auf der Studentenübersicht sieht man, welche Studenten in keiner und welche in mehreren Gruppen sind. Die Gruppenbildung ist erst gültig, wenn alle Studenten genau einer Gruppe zugewiesen sind. Auch auf dieser Seite können neue Gruppen erstellt werden.

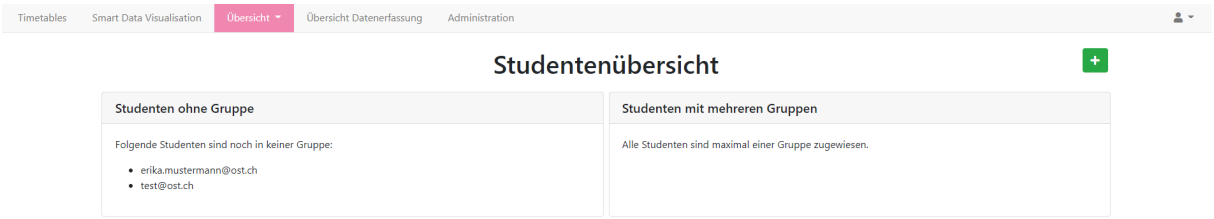

Abbildung 8.8: Studentenübersicht

Betreuerübersicht

Die Betreuerübersicht listet alle Betreuer und deren zugewiesene Gruppen auf. Es ist das Ziel, dass die Anzahl betreuter Gruppen zwischen der minimalen und der maximalen Anzahl liegt. Auch hier können neue Gruppen erstellt oder bestehende bearbeitet werden.

Abbildung 8.9: Betreuerübersicht

Übersicht Datenerfassung

Es ist möglich, dass Studenten oder Betreuer eine Änderung an ihren eingegebenen Daten wünschen. Nachdem der Administrator die Bearbeitung deaktiviert hat, kann nur noch er die Benutzereingaben anpassen. Auf der Übersichtsseite der Datenerfassung werden alle Studenten und Betreuer aufgelistet. Über den Edit Button werden die Benutzerinformationen von einer Person angezeigt und können vom Administrator bearbeitet werden.

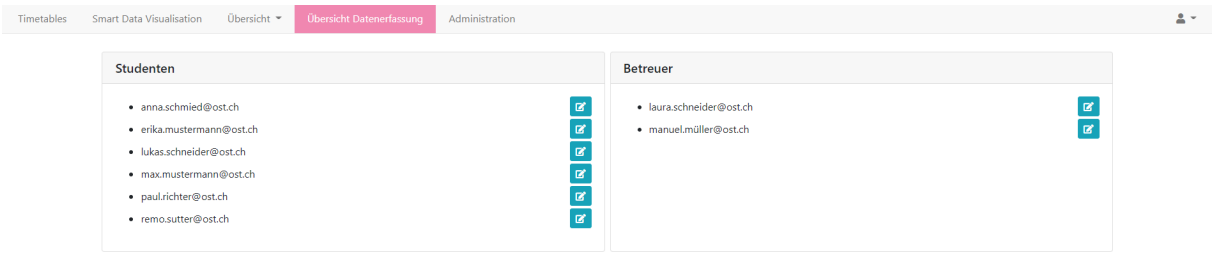

Abbildung 8.10: Übersicht Datenerfassung

Teil III

Administrative Anhänge

Anhang A

Persönliche Berichte

A.1 Julia Tanner

In der ersten Woche unserer Studienarbeit gingen wir davon aus, dass wir einen Algorithmus evaluieren oder ausarbeiten sollen, welcher bereits vorgefertigte Gruppen anhand der eingegebenen Daten liefert. Wir machten uns also relativ schnell viele Gedanken dazu, lehnten ein Buch über Algorithmen in der Bibliothek aus und machten gemeinsam ein Brainstorming über einen ersten Entwurf für einen Algorithmus. Wir erkannten aber schnell, dass wenn wir den besten Algorithmus evaluieren, implementieren und gleichzeitig eine Applikation entwickeln sollen, welche einige Funktionen anbietet, es viel zu viel für eine Studienarbeit wird. Aufgrund dessen war ich in den ersten 1-2 Wochen etwas überfordert und wusste gar nicht, wo ich anfangen soll.

Nachdem wir dann in der zweiten Woche ein Meeting mit Prof. Dr. Farhad D. Mehta hatten, haben wir dieses Problem angesprochen. Er hatte Verständnis für unsere Situation und war ebenfalls der Meinung, dass es zu viel wird. Wir definierten also gemeinsam das MVP um unsere Arbeit etwas einzugrenzen. Dies half mir, die Studienarbeit besser zu verstehen und ich wusste nun, wie ich mit der Arbeit starten soll. Nach diesen Startschwierigkeiten war ich wieder motiviert und konnte endlich richtig in die SA starten. Es gab zwar immer wieder mal kleine Probleme und Unklarheiten jedoch konnten diese durch Sitzungen schnell geklärt werden, sodass wir ziemlich schnell gute Ergebnisse erzielen konnten.

Ein Teil unserer Arbeit bestand daraus, gemeinsam die Funktionalitäten zu definieren und auszuarbeiten. Dadurch hatten wir die Möglichkeit, unsere Ideen und Verbesserungsvorschläge zu teilen. Diese Aufgabe war zum Teil etwas herausfordernd jedoch war es sehr schön Mitspracherecht zu haben und der Kreativität freien lauf zu lassen.

Die Arbeit mit Severin und mir hat meiner Meinung nach gut funktioniert. Wir haben zwar eine etwas unterschiedliche Arbeitsweise jedoch ergänzen wir uns durch unsere Interessen sehr gut. Einzig in der Dokumentation könnten wir uns noch etwas verbessern. Wir haben am Anfang sehr vieles informell festgelegt, ohne zu dokumentieren, sodass wir gegen Ende der SA viel Zeit mit Dokumentation schreiben verbrachten.

Im Allgemeinen bin ich sehr zufrieden mit dem Ergebnis unserer Arbeit und ich hoffe, dass die Web-Applikation die Gruppenbildung und die Zuweisung der Betreuer erleichtern wird. Es war mir eine Freude, zusammen mit Severin ein Projekt für Prof. Dr. Farhad D. Mehta durchzuführen.

A.2 Severin Amacher

Die Arbeit in den ersten paar Wochen der Studienarbeit war etwas Träge. Dies kam davon, dass wir nicht recht wussten was wir genau implementieren sollen und etwas überfordert mit der ursprünglichen Aufgabenstellung waren. Jedoch konnten wir mit Prof. Dr. Farhad D. Mehta schnell ein Meeting halten, um eine genaueres MVP zu vereinbaren, welches sich von der ursprünglichen Aufgabenstellung etwas unterschied.

Nach diesem Meeting konnten wir richtig mit der Studienarbeit loslegen. Es gab noch viele Sachen zu klären, wie wir das Projekt implementieren und genauere Spezifikation der Funktionalitäten. Mit enger Zusammenarbeit zwischen mir und Julia und den fast wöchentlichen Meetings mit Prof. Dr. Farhad D. Mehta konnten wir die Spezifikation wie geplant beenden und pünktlich mit der Construction Phase beginnen.

Die Studienarbeit war für mich sehr interessant, da ich nicht nur etwas Neues implementieren konnte, sondern mich auch mit neuen Technologien wie Docker auseinandersetzen konnte. Ausserdem war es anregend direkt mit dem Kunden (Prof. Dr. Farhad D. Mehta) die Spezifikation auszuarbeiten und nicht wie in meinem Job das Gewünschte zu implementieren. Ich konnte mit meinen Ideen unser Endprodukt gestalten.

Schlussendlich bin ich sehr zufrieden mit unserer Arbeit. Julia und ich haben unser Bestes gegeben und ich denke, wir sind ein gutes und mittlerweile eingespieltes Team. Die Zusammenarbeit mit Prof. Dr. Farhad D. Mehta verlief auch ohne Probleme, da wir bereits mit ihm das EPJ durchgeführt hatten.

Anhang B

Projektmanagement

B.1 Projektorganisation

Severin Amacher wird sich um den Aufbau der Infrastruktur kümmern und Julia Tanner ist dafür verantwortlich, den Projektablauf und die Zeiterfassung im Auge zu behalten. Bei der Programmierung werden beide sowohl Frontend als auch Backend Aufgaben übernehmen. Ausserdem ist es für uns wichtig, dass sich jeder mit allen Features auskennt.

B.2 Projekt Meetings

Jeden Dienstag findet von 10:00 bis 12:00 Uhr ein Teammeeting statt. An diesem Meeting führen wir den Sprint Review und die Sprint Planung durch. Ausserdem besprechen wir offene Fragen und aktuelle Probleme. Jedes Teammitglied präsentiert entweder in der Dokumentation oder im Code, was er in der letzten Woche getan hat. So führen wir direkt zusammen einen Review durch und können Verbesserungen einfach besprechen. Da wir täglich in Kontakt stehen, können wir jederzeit bei Bedarf weitere Meetings planen.

Mit dem Betreuer werden wir je nach Bedarf wöchentlich oder alle zwei Wochen eine Sitzung durchführen. Wir legen also nach jeder Sitzung fest, wann die nächste stattfinden soll.

B.2.1 Protokollierung

Für Sitzungen, welche mit dem Betreuer durchgeführt werden, werden wir immer ein Sitzungsprotokoll schreiben. An den Teammeetings wird nach Bedarf entschieden, ob ein Protokoll erstellt wird oder nicht. Falls nur der aktuelle Stand miteinander besprochen wird und in unseren Augen nichts Wichtiges festgelegt wird, werden wir kein Sitzungsprotokoll schreiben.

B.3 Planung

Unsere Issues werden wir auf dem HSR Gitlab [7] verwalten und planen. Der Code und die Dokumentation ist auf dem OST Gitlab [4] zu finden. Da bei der Migration vom HSR auf das OST Gitlab die Zuweisung der Issues und die rapportierten Zeiten verloren gingen, benötigen wir beide Gitlab Instanzen.

Für unsere Studienarbeit haben wir uns für nachfolgende Zeitplanung entschieden.

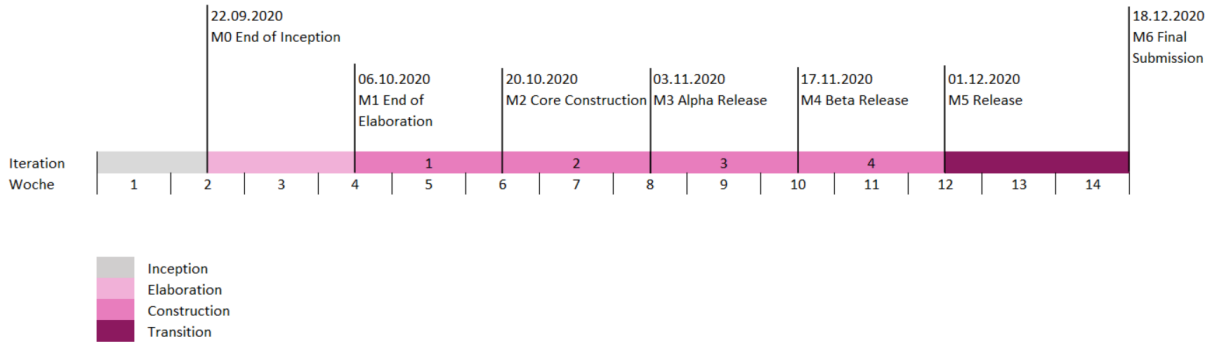

Abbildung B.1: Zeitplanung

Insgesamt haben wir 7 Meilensteine, welche alle 2 Wochen dauern.
Während der Inception, Elaboration und Transition Phase verzichten wir auf klassische Sprints. Wir werden zwar zu Beginn von jedem Meilenstein in diesen Phasen festlegen, welche Aufgaben wir erreichen wollen, jedoch werden wir durch stetige Kommunikation die genaue Planung je nach Bedürfnis anpassen.
Während der Construction Phase arbeiten wir mit 2 Wochen Sprints, welche immer am Dienstag starten und enden. Somit verlaufen die Sprints identisch zu den Meilensteinen.

B.3.1 Construction Phase

Für die Construction Phase haben wir uns folgende Ziele gesetzt.

Iteration 1 Dauer: 06.10.20 - 20.10.20
In der Iteration 1 erstellen wir die Basisfunktionalität für unsere Webapplikation.

- Datenbank wird aufgesetzt und mit der Applikation verknüpft
- Die View für die Datenerfassung der Studenten und Betreuer ist fertig
- Administrator Funktionen wie z.B. ”Datenbank leeren”, ”Benutzer hinzufügen”, ”Benutzer entfernen”, ”Passwort zurücksetzen” werden implementiert
- User können sich einloggen und ihr Passwort ändern
- Import und Export der Datenstruktur

Iteration 2 Dauer: 20.10.20 - 03.11.20
In der Iteration 2 beginnen wir weitere Funktionalitäten für das MVP zu programmieren.

- Gruppen können vom Administrator erstellt, bearbeiten und gelöscht werden
- Eine erste Version der Timetables und Smart Data Visualisation Ansicht wird programmiert

Iteration 3 Dauer: 03.11.20 - 17.11.20
In der Iteration 3 stellen wir die restlichen Funktionalitäten, welche für das MVP nötig sind, fertig.

- Timetables und Smart Data Visualisation Funktionalitäten fertigstellen

Iteration 4 Dauer: 17.11.20 - 01.12.20
In der Iteration 4 werden wir mit den Daten aus der Datenerhebung selbst versuchen Gruppen zu bilden. Anhand der Erkenntnisse von diesem Versuch werden wir Erweiterungen und Verbesserungen einbauen. Ausserdem werden wir das Programm intensiv testen und Fehler beheben.

B.3.2 Iterationsplanung

Zu Beginn jeder Iteration führen wir eine Iterationsplanung durch. Dafür wird zuerst ein kurzer Sprint Review gemacht um herauszufinden, was erreicht wurde, was nicht und wo es Probleme gab. Alle nicht erreichten Issues und die Issues der aktuellen Iteration werden besprochen, geschätzt, priorisiert und verteilt.

B.4 Meilensteine

Bezeichnung	Datum	Beschreibung
M0 - End of Inception	22.09.2020	Projektauftrag verstanden, Ziele und Technologien festgelegt
M1 - End of Elaboration	06.10.2020	Projektplan definiert, Anforderungsspezifikation, Domainmodel und Wireframes erstellt, Software Architektur festgelegt, Prototyp für die Datenerhebung fertig implementiert
M2 - Core Constructio	20.10.2020	Datenbank aufgesetzt, Studenten / Betreuer können sich einloggen, Daten erfassen und ihr Profil verwalten, alle Administrator Funktionen sind implementiert
M3 - Alpha Release	03.11.2020	Gruppen können erstellt und bearbeitet werden, ein erster Entwurf für die Timetable und Smart Data Visualisation ist implementiert
M4 - Beta Release	17.11.2020	Optimierung der Timetable und Smart Data Visualisation
M5 - Release	01.12.2020	Eigene Gruppenbildung durchgeführt, Verbesserungen eingebaut, intensiv getestet
M6 - Final Submission	18.12.2020	Schlussdokumentation erstellt, Go Live der Applikation

Tabelle B.1: Meilensteine

B.5 Gantt-Diagramm

Das nachfolgende Gantt-Diagramm visualisiert die Planung der Tätigkeiten pro Phase und Meilenstein. Die Meilensteine sind immer am Dienstag, also am Sprint Ende, definiert.

EPJ Team Builder	Projekt Start	KW	38	39	40	41	42	43	44	45	46	47	48	49	50	51
	14.09.2020	Sprint Start Datum	15. Sep	22. Sep	29. Sep	6. Okt	13. Okt	20. Okt	27. Okt	03. Nov	10. Nov	17. Nov	24. Nov	1. Dez	08. Dez	18. Dez
Task		**Datum**														
Inception																
Projektauftrag verstanden																
Minimum Viable Product																
Technologien festlegen / kennenlerne																
M0 - End of Inception		**22.09.2020**	★													
Elaboration																
Projektplan																
Prototyp Datenerhebung																
Virtualler Server																
Wireframes																
Use Cases																
Software Architektur																
Datenbankmodell																
Graph Library																
M1 - End of Elaboration		**06.10.2020**			★											
Construction																
Datenbank aufsetzen																
View für Datenerfassung																
Administrator Funktionen																
Login und Passwort ändern																
JSON Export/Import																
M2 - Core Construction		**20.10.2020**					★									
Timetable View																
Smart Data Visualisation																
CRUD Gruppen																
M3 - Alpha Release		**03.11.2020**							★							
Smart Data Visualisation Optimierungen																
Timetable Optimierungen																
M4 - Beta Release		**17.11.2020**									★					
Bug Fixing																
Instensive Testing																
Reserve																
M5 - Release		**01.12.2020**											★			
Transition																
Dokumentation																
M6 - Final Submission		**18.12.2020**														★

Abbildung B.2: Gantt Diagramm

B.6 Zeiterfassung

Da bei der Migration vom HSR auf das OST Gitlab die Zuweisung der Issues und die rapportierten Zeiten verloren ging, wir die Zeiterfassung auf dem HSR Gitlab [7] gemacht. Mit dem Tool GTT (GitLab Time Tracker) können wir die Zeiten auswerten.

B.7 Qualitätsmassnahmen

B.7.1 Dokumentation

Die Markdown Dokumente, welche nach Projektabschluss weiterverwendet und fortlaufend bei Änderungen aktualisiert werden, werden auf dem OST Gitlab [4] verwaltet. Das LaTex Dokument, welches nach Ende des Projekts archiviert wird, wird im Overleaf [9] gespeichert. Nach Abschluss einer Änderung an einem Dokument gibt man dem anderen Mitglied Bescheid. Anschliessend überprüft dieser die Änderungen und korrigiert Schreibfehler. Falls es inhaltliche Fehler gibt, gibt man dem anderen Projektmitglied ein Feedback und es werden nochmals Anpassungen vorgenommen.

B.7.2 Branches

Im Code Repository auf dem OST Gitlab [4] gibt es die zwei Branches "Master" und "Development". Pro Feature wird ein weiterer Branch auf Basis vom Development Branch erstellt. Sobald man mit einem Issue fertig ist, wird über ein Merge Request die Änderung auf den Development Branch geschrieben. Nur wenn ein Release ansteht oder ein Hotfix implementiert wurde, wird ein Merge Request vom Development auf den Master ausgeführt.

B.7.3 Code Review

Wir sind beide keine Fans davon, jeden Merge Request durch eine andere Person überprüfen zu lassen. Es kostet viel Zeit, bringt nur etwas, wenn alle Parteien den Code genau anschauen und man hat unnötige Wartezeiten, wenn ein Merge Request nicht sofort angeschaut wird und man die Änderungen auf einem neuen Branch haben möchte.
Aus diesem Grund haben wir uns dazu entschieden, jeweils einen Code Review am Teammeeting durchzuführen. So kann jede Person wöchentlich vorstellen, welche Features oder Verbesserungen er eingebaut hat und wie dies gemacht wurde. Die Änderung wird sowohl im Programm als auch im Code angeschaut, wodurch automatisch ein Review entsteht.

B.7.4 Unit Testing

Im Service Package vom Backend wird eine Code Coverage von mindestens 70% garantiert. Zu jedem Issue, welcher das Backend betrifft, wird mindestens ein Test erfasst. Die Code Coverage wird mittels der Entwicklungsumgebung Intellij IDEA überprüft.
Bei jedem Commit werden in der CI-Pipeline alle Tests ausgeführt und so Fehler erkannt.

B.7.5 Definition of Done

Die folgenden Kriterien müssen erfüllt sein, bevor ein Issue als abgeschlossen betrachtet wird

- CI-Pipeline ist erfolgreich durchgelaufen
- Alle Unit-Tests laufen fehlerfrei
- Test-Abdeckungsgrad von 70% wurde eingehalten

B.7.6 Code Freeze

Ab dem 01.12.2020, also nach Ende von Meilenstein "M5 - Release", wird ein Code Freeze gemacht. Ab diesem Zeitpunkt werden keine neuen Features mehr implementiert. Es sind nur noch Bug-Fixes und Verbesserungen erlaubt.

Anhang C

Wireframes

Die Wireframes wurden erstellt, um eine ungefähre visuelle Vorstellung für die Web-Applikation zu geben. Sie wurden nur für das Administrator Dashboard erstellt, da zu diesem Zeitpunkt die View für die Datenerfassung der Studenten und Betreuer bereits existierte, da wir diese für die Datenerhebung benötigten.

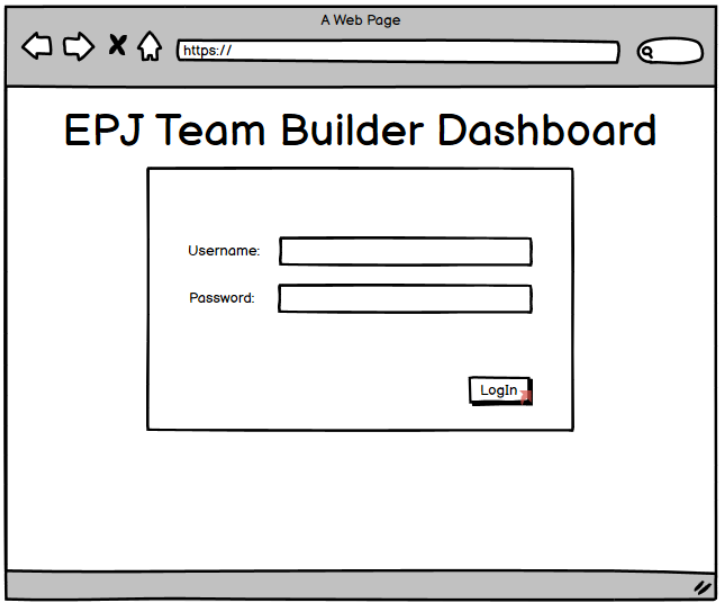

Abbildung C.1: Administrator Login

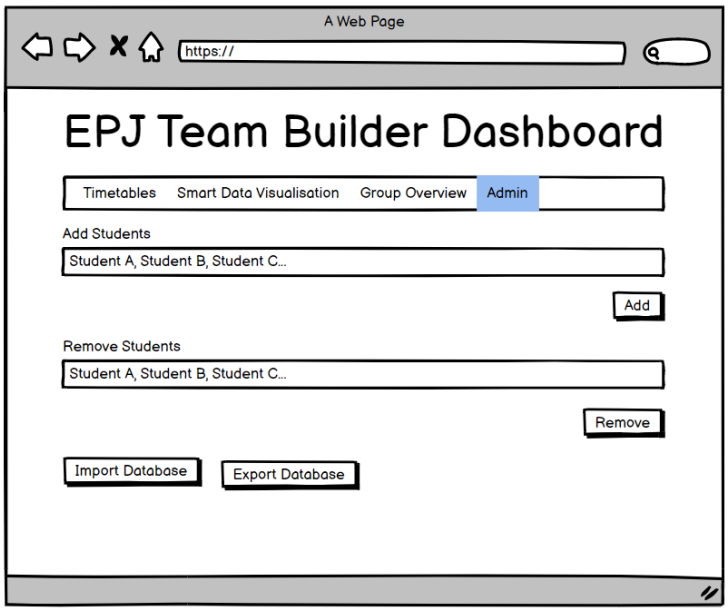

Abbildung C.2: Administrator Dashboard

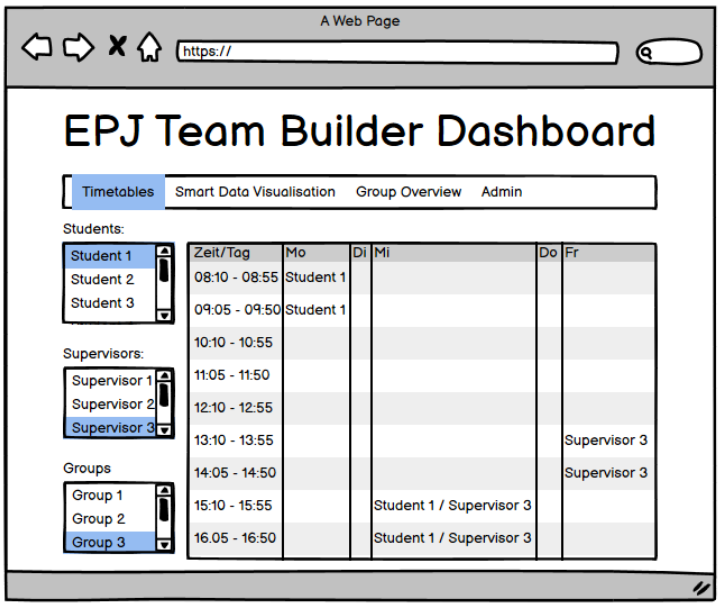

Abbildung C.3: Timetable

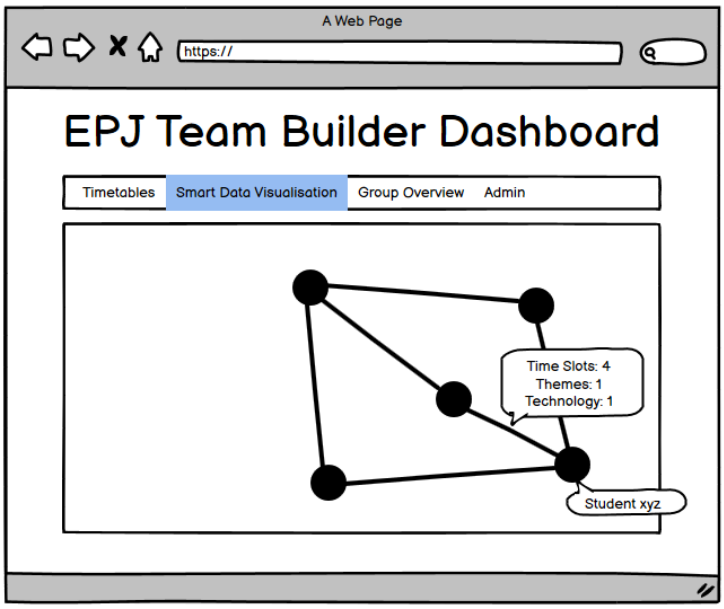

Abbildung C.4: Smart Data Visualisation

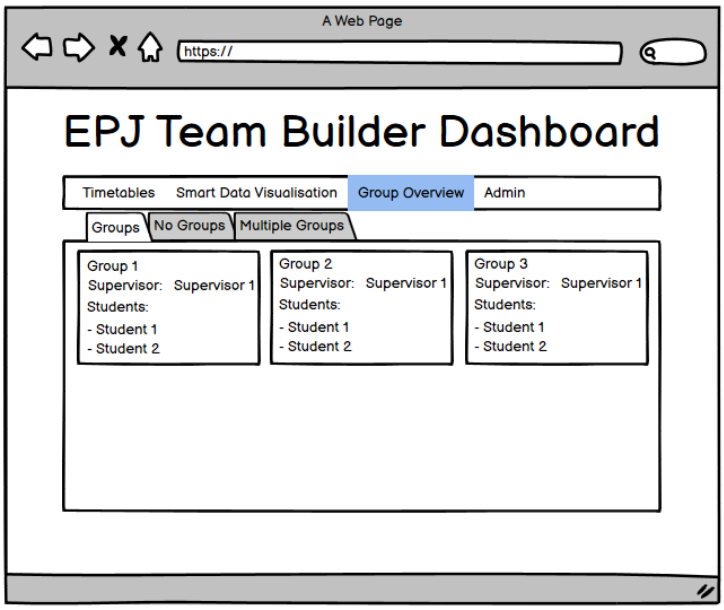

Abbildung C.5: Gruppenübersicht

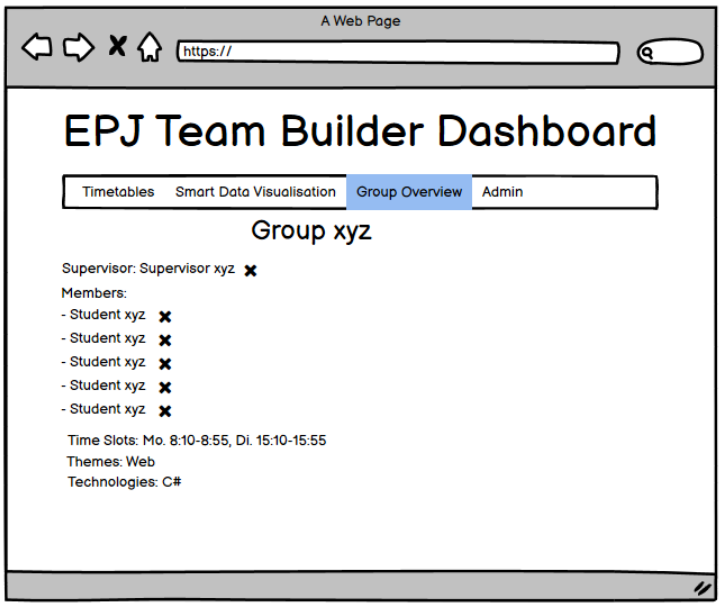

Abbildung C.6: Gruppendetail

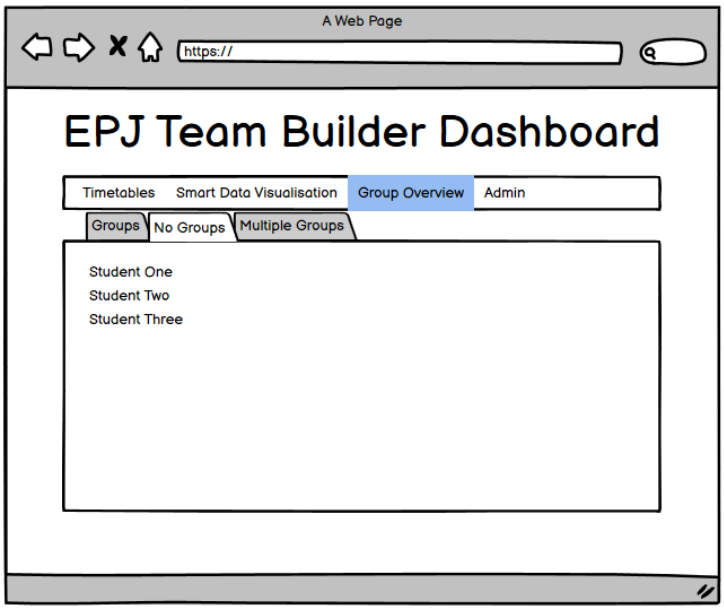

Abbildung C.7: Studenten ohne Gruppe

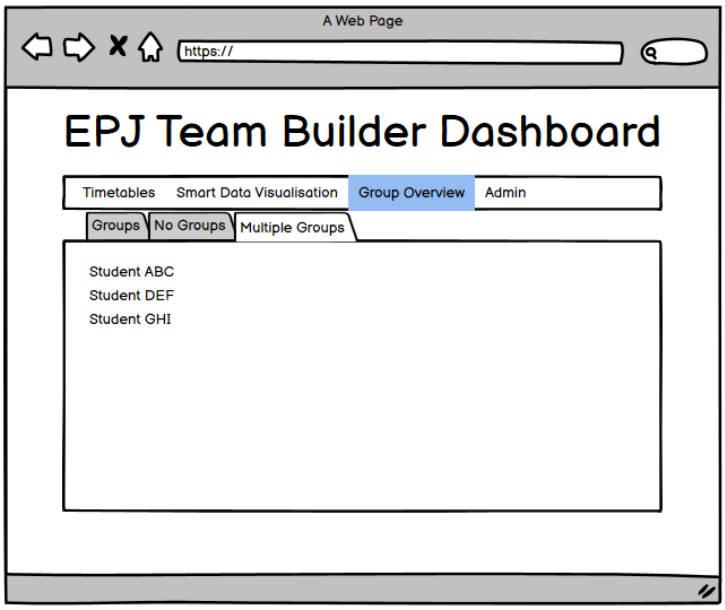

Abbildung C.8: Studenten mit mehreren Gruppe

Anhang D

Risikoanalyse

Während der Arbeit brauchten wir viel Zei,t um den Funktionsumfang zu analysieren und mit Herrn Mehta zu besprechen, was das Endprodukt umfassen soll. Die Risiken wurden regelmässig informell festgehalten jedoch ging es aufgrund des Zeitmangels etwas vergessen sie auch zu dokumentieren. Aus diesem Grund haben wir uns dazu entschieden, die Risiken, welche nach dem Abschluss der Arbeit noch auftreten könnten, zu analysieren.

Nr.	Risiko
R1	Das Produkt ist nicht brauchbar.
R2	Das Projekt ist schwierig zu erweitern.
R3	Unzufriedenheit der Studenten.
R4	Kein Benutzerfreundliches User Interface
R5	Fehler und Bugs im Produkt

Tabelle D.1: Risikoanalyse

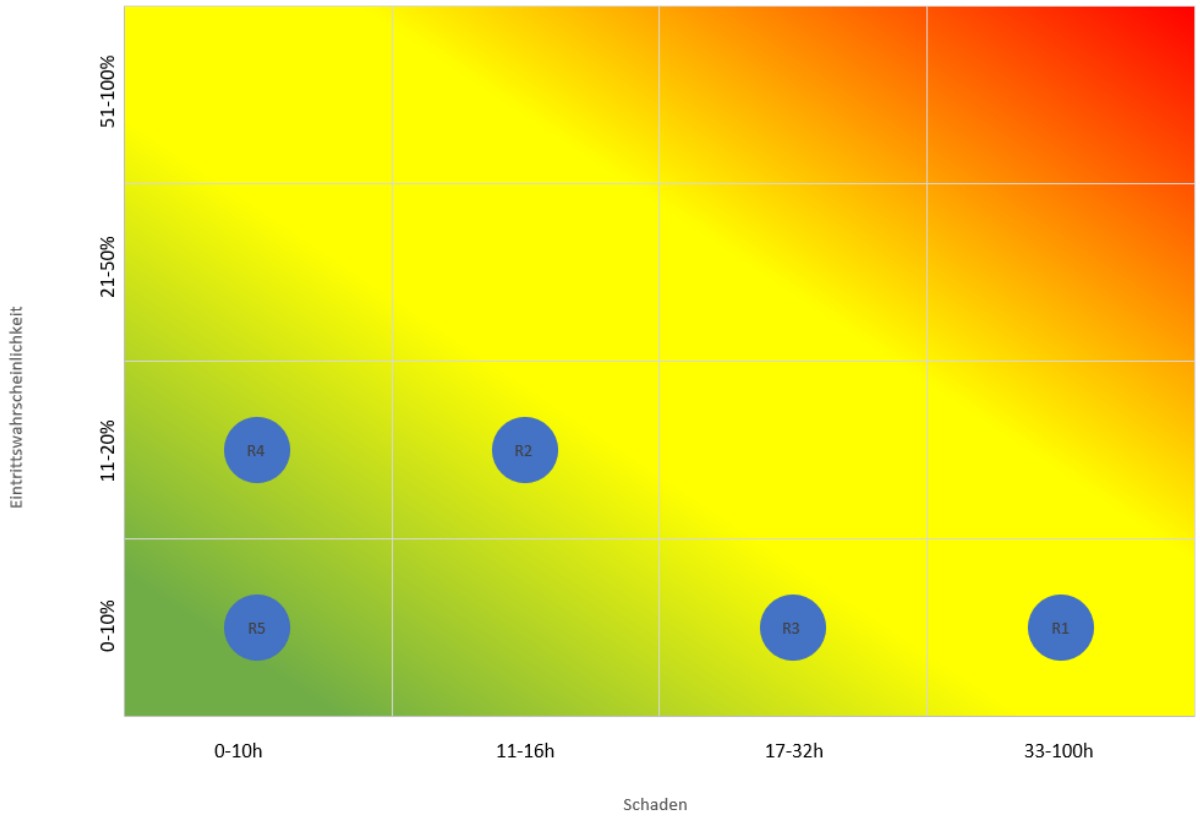

Abbildung D.1: Risikoanalyse

D.1 Risikobeschreibung

D.1.1 R1 Das Produkt ist nicht brauchbar

Beschreibung: Beim Verwenden unseres Produktes stellt sich heraus, dass die Funktionalitäten den EPJ Organisator nicht genügend unterstützen. Es wäre zum Beispiel möglich, das Gruppen nicht gut gebildet werden können. In diesem Fall müsste der EPJ Organisator das EPJ so organisieren wie bisher, da unser Endprodukt nicht anwendbar ist.

Schaden: Der Schaden, der aus diesem Risiko hervorgeht, liegt bei 100 Stunden. Es müsste nochmals ungefähr dieselbe Zeit investiert werden, wie bisher für die Implementierung, um in der Applikation neue Funktionalitäten einzubauen.

Massnahmen: Dieses Risiko haben wir auf unter 10% eingeschätzt. Durch den Testdurchlauf mit den Daten aus der Datenerhebung und die viele Kommunikation mit unserem Betreuer, haben wir das Risiko so gut es geht reduziert. Da die Daten jedoch nicht so gut waren wie gewünscht, besteht immer noch ein Restrisiko.

D.1.2 R2 Das Projekt ist schwierig zu erweitern

Beschreibung: Das Implementieren von Erweiterungen ist schwierig und mühsam. Der Entwickler findet sich nicht im Code zurecht oder die Struktur des Projektes erlaubt es nicht die Erweiterungen einzubauen.

Schaden: Der Schaden wird aus dem Fall berechnet, dass eine interne Struktur Änderung vorgenommen werden muss. Dies sollte ungefähr in einem Bereich von 11-16 Stunden liegen.

Massnahmen: Um das Implementieren von Erweiterungen zu vereinfachen, haben wir das Manual geschrieben. Dort wird anhand des UI erklärt, was bei einer Aktion ausgeführt wird und man kann sich schnell in das Produkt einarbeiten. Zusätzlich ist die Struktur dokumentiert und gewährt einen bequemen Einstieg. Im Code sind die Objekte so gekapselt, dass es keine Schwierigkeiten geben sollte, etwas Neues hinzuzufügen.

D.1.3 R3 Unzufriedenheit der Studenten

Beschreibung: Nach der Gruppeneinteilung gibt es unzufriedene Studenten. Sie sind nicht mit den gewünschten Studenten in einer Gruppe oder die Zeitslots stimmen nicht mit ihren Teammitgliedern überein.

Schaden: Um eine Gruppeneinteilung zu korrigieren, muss der Organisator nochmals mehrere Gruppen vergleichen, um Studenten austauschen zu können. Dies kann je nach Gegebenheit kurz oder lange dauern. Eine längere Umstellung würde ungefähr 20 Stunden dauern.

Massnahmen: Um dieses Risiko zu minimieren, berücksichtigen unsere Visualisierungen sämtliche Angaben der Studenten.

D.1.4 R4 Kein Benutzerfreundliches User Interface

Beschreibung: Die Studenten, Betreuer und der EPJ Organisator finden das UI nicht intuitiv und haben Schwierigkeiten bei der Benutzung.

Schaden: Um das UI etwas intuitiver zu gestalten, muss man höchstens 10 Stunden aufwenden.

Massnahmen: Das Risiko haben wir durch die Datenerhebung, bei der die Studenten und Betreuer die Applikation bereits benutzt haben und ihre Anmerkungen uns mitteilen konnten, minimiert. Ausserdem hat der EPJ Organisator das UI regelmässig gesehen und konnte fortlaufen Änderungswünsche anbringen.

D.1.5 R5 Fehler und Bugs im Produkt

Beschreibung: Im Endprodukt sind mehrere Bugs und Fehler vorhanden, die gefixt werden müssen.

Schaden: Um kleine Bugs und Fehler zu beheben, müssen nicht mehr als 10 Stunden aufgewendet werden.

Massnahmen: Unsere Services haben alle mindestens eine Test Coverage von 70%. Ausserdem haben wir unser UI mit mehreren Browsern anhand einer Checkliste getestet. Dadurch sollten Bugs und Fehler minimiert und die Eintrittswahrscheinlichkeit auf weniger als 10% reduziert worden sein.

Literaturverzeichnis

[1] *Administrator Dashboard*. URL: https://epjteambuilder-dashboard.dev.ifs.hsr.ch/.

[2] *BCryptPasswordEncoder*. URL: https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/crypto/bcrypt/BCryptPasswordEncoder.html.

[3] *Betreuer Information Formular*. URL: https://epjteambuilder-form.dev.ifs.hsr.ch/supervisor.

[4] *Code und Dokumentation Repository*. URL: https://gitlab.ost.ch/epj-team-builder.

[5] *Graph Library vis.js*. URL: https://visjs.github.io/vis-network/docs/network/.

[6] *Intellij IDEA*. URL: https://www.jetbrains.com/de-de/idea.

[7] *Issue Tracking*. URL: https://gitlab.dev.ifs.hsr.ch/sa1/code/epjteambuilder/-/issues.

[8] *JUnit 5*. URL: https://junit.org/junit5/.

[9] *Overleaf*. URL: https://de.overleaf.com/learn.

[10] *Spring Boot*. URL: https://spring.io/projects/spring-boot.

[11] *Spring Security*. URL: https://spring.io/projects/spring-security.

[12] *Studenten Information Formular*. URL: https://epjteambuilder-form.dev.ifs.hsr.ch/student.

[13] *Thymeleaf*. URL: https://www.thymeleaf.org/index.html.