Pflichtenheft - Software für Arbeitereinsatzplanung

Alexander Förster, Erik Dubrov, Meric Kaynak, Raphael Bieniek, Tim Hönings

June 2024

Inhaltsverzeichnis

1 Einleitung **3**
1.1 Einleitung . . . 3
1.2 Rahmenbedingungen . . . 3

2 Anforderungen **3**
2.1 Funktionale Anforderung . . . 4
2.2 Nichtfunktionale Anforderungen . . . 5

3 Use Case (Anwendungsfälle) **5**
3.1 Use Case Diagramm . . . 5
3.1.1 Details . . . 6
3.1.2 Use Case 1 . . . 6
3.1.3 Use Case 2 . . . 6
3.1.4 Use Case 3 . . . 6

4 Architektur **7**
4.1 High-Level System Design . . . 7
4.2 Komponenten- & Konnektoransicht . . . 8
4.3 Model View Controller (MVC) - Details der Implementierung . . . 8

5 Geschäftsprozesse **9**
5.1 Geschäftsprozess „Krankmeldung einer Lehrperson“ . . . 9
5.1.1 Geschäftsprozess „Terminplanerstellung“ . . . 9

6 Datenmodell **10**
6.1 ER-Diagramm . . . 10
6.2 Form der Datenhaltung . . . 10

7 GUI-Design **11**

8 Klassendiagramm **13**

9 Implementierung **14**
9.1 Implementierung - Funktionale Anforderungen 14
9.2 Implementierung - Nichtfunktionale Anforderungen 16
9.3 Implementierung - Abweichungen 17

10 Test **17**
10.1 Unit Test 17
10.2 Eingabefeld Test 18

1 Einleitung

1.1 Einleitung

Die Hochschule ”Wissen für Alle - die Wissensakademie”hat uns kontaktiert, um eine Softwarelösung zu entwickeln, die die automatisierte Planung und Verwaltung ihrer Lehrveranstaltungen und Kurse ermöglicht. Die manuelle Planung ist aufgrund des starken Wachstums der Hochschule und der steigenden Anzahl von Studierenden und Lehrpersonen nicht mehr praktikabel und fehleranfällig geworden. Das Ziel dieses Projekts ist es, eine effiziente und zuverlässige Software zu entwickeln, die den Planungsprozess automatisiert und vereinfacht. Die Software soll es Lehrpersonen und Studierenden ermöglichen, jederzeit ihre aktuellen Einsatzpläne und Veranstaltungszeiten zu überprüfen und auf kurzfristige Änderungen flexibel zu reagieren.

1.2 Rahmenbedingungen

- Die Hochschule „Wissen für Alle - die Wissensakademie”verfügt über zwei Standorte, ein Hauptcampus und ein neu eröffneter Campus in der Nachbarstadt.
- Die Hochschule beschäftigt fast 100 Lehrpersonen.
- Die Lehrpersonen arbeiten in der Regel 18 Stunden pro Woche, verteilt auf verschiedene Lehrveranstaltungen.
- Die Hochschule hat über 1000 Studierende, die an verschiedenen Studiengängen und Weiterbildungsprogrammen teilnehmen.
- Eine Lehrveranstaltung dauert in der Regel zwei Stunden.
- Die Planung der Lehrveranstaltungen muss Räume, Lehrpersonen und Zeiten berücksichtigen.
- Sonderveranstaltungen oder spezielle Kurse für Schülergruppen müssen ebenfalls eingeplant werden können.
- Automatisierte Benachrichtigungen bei Planänderungen oder Ausfällen von Veranstaltungen sollen integriert werden.

2 Anforderungen

Priorität Keywords:

- **MUSS** - Muss Anforderungen: Notwendige Anforderungen die erfüllt werden müssen.
- **SOLL** - Soll Anforderung: Anforderungen bezüglich der Mindesterfüllung.
- **KANN** - Kann Anforderung: Wunschanforderungen, die unter Umständen ausgelassen werden können.

2.1 Funktionale Anforderung

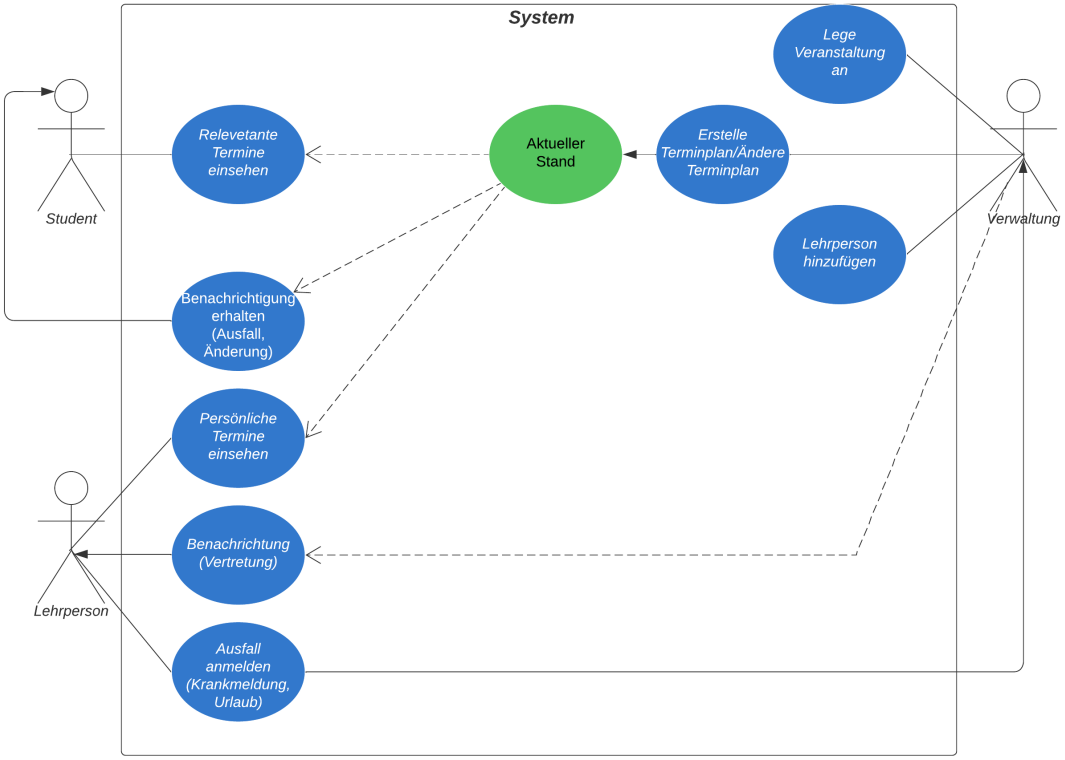
ID	Anforderung	Beschreibung	Prio.
FA_1.0	**Terminplanung**	Die Software muss es ermöglichen, einen Einsatzplan für die Lehrpersonen zu erstellen, der zeigt, welche Lehrveranstaltungen sie betreuen sollen.	MUSS
FA_1.1	**Änderungsmanagement**	Die Software muss Änderungen im Plan berücksichtigen können, z.B. durch Krankmeldungen, Fortbildungen oder Tagungen der Mitarbeiter.	MUSS
FA_1.2	**Benachrichtigungen**	Lehrpersonen und Studierende sollen über Änderungen im Plan informiert werden.	SOLL
FA_1.3	**Benutzerzugriff**	Lehrpersonen und Studierende sollen auf ihre individuellen Pläne zugreifen können.	SOLL
FA_1.4	**Ausfall & Vetretung**	Bei Ausfall eines Lehrbeauftragten soll ein anderer eine Benachrichtung erhalten um diese veranstaltung zu vertreten.	SOLL
FA_1.5	**Sonderveranstaltungen**	Die Software soll Sonderveranstaltungen, wie z.B. Kurse für Firmen oder Schülergruppen, verwalten können.	KANN
FA_1.6	**Automatische Aktualisierung**	Die Software soll sich automatisch aktualisieren, wenn neue Informationen oder Änderungen vorliegen.	SOLL
FA_1.7	**KI-Unterstützung**	Mögliche Integration einer KI, um Sprachnachrichten automatisch zu transkribieren und in Textform an die Sekretärin weiterzuleiten.	KANN

2.2 Nichtfunktionale Anforderungen

ID	Anforderung	Beschreibung
NFA_1.0	**Benutzerfreundlichkeit**	Die Software soll eine intuitive Benutzeroberfläche haben.
NFA_2.0	**Zuverlässigkeit**	Die Software muss zuverlässig arbeiten und darf keine Ausfälle verursachen.
NFA_3.0	**Performance**	Die Software muss schnell reagieren und Pläne zügig aktualisieren können.
NFA_4.0	**Skalierbarkeit**	Die Software soll auch bei weiterem Wachstum der Hochschule problemlos funktionieren.
NFA_5.0	**Datensicherheit**	Die Software muss sicherstellen, dass alle Daten geschützt sind und nur autorisierte Benutzer Zugriff haben.

3 Use Case (Anwendungsfälle)

3.1 Use Case Diagramm



3.1.1 Details

3.1.2 Use Case 1

- **primärer Akteur:** Verwaltung
- **Voraussetzung:** Verwaltung hat sich angemeldet
- **Haupterfolgsszenario**:
  1. Die Verwaltung legt Veranstaltungen an
  2. System aktualisiert die gegebenen Veranstaltungen
  3. Sie fügt Lehrpersonen hinzu
  4. System aktualisiert die gegebenen Lehrpersonen
  5. Verwaltung lässt den Terminplan erstellen
  6. System erstellt einen Terminplan
  7. Sie fügt neue Veranstaltungen an
  8. System aktualisiert die gegebenen Veranstaltungen
  9. Sie fügt neue Lehrpersonen hinzu
  10. System aktualisiert die gegebenen Lehrpersonen
  11. Verwaltung lässt den Terminplan ändern
  12. System aktualisiert den Terminplan

3.1.3 Use Case 2

- **primärer Akteur:** Student
- **Voraussetzung:** Student hat sich angemeldet
- **Haupterfolgsszenario**:
  1. Das System zeigt die Termine des Benutzers
  2. Der Student kann relevante Termine einsehen
  3. Das System leitet den Student eine Benachrichtigung weiter
  4. Falls es einen Ausfall oder eine Änderung gibt, erhält der Student eine Benachrichtigung

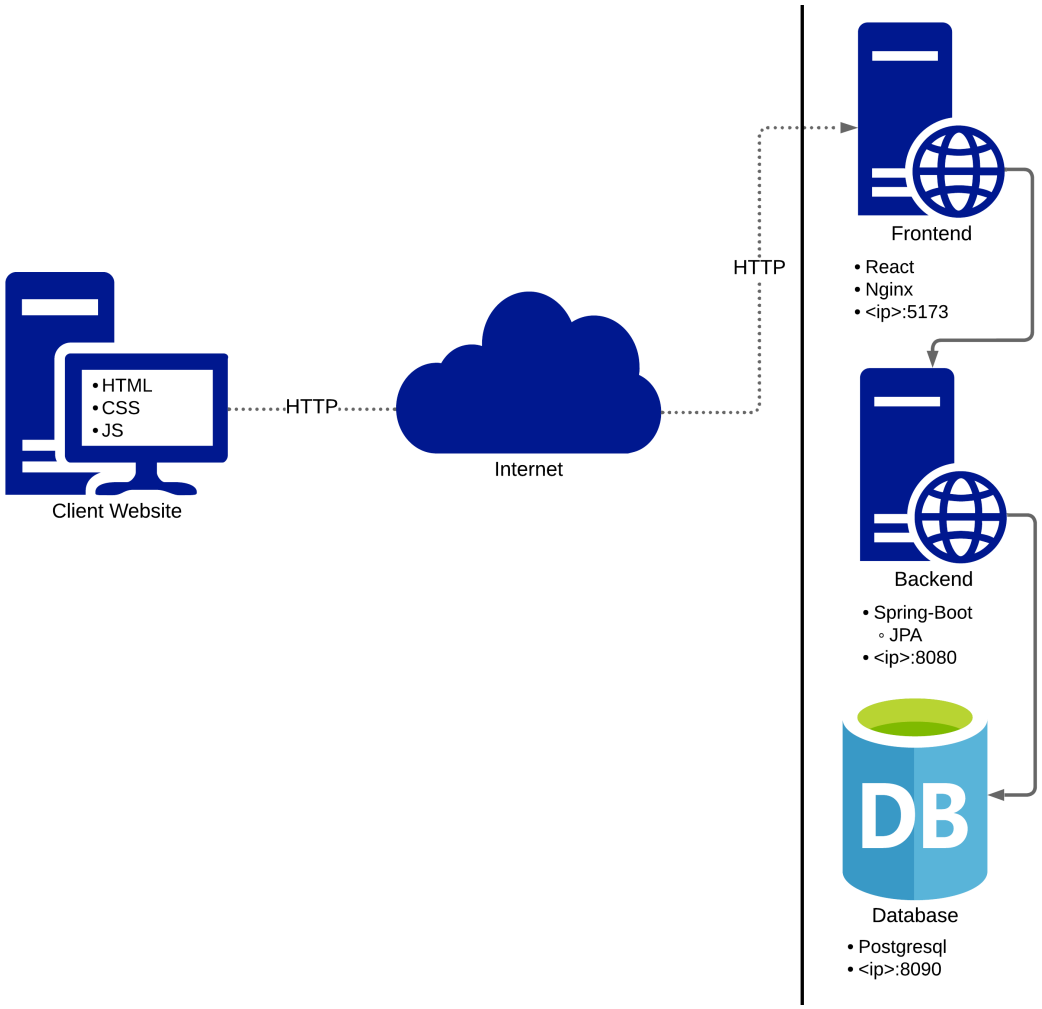
3.1.4 Use Case 3

- **primärer Akteur:** Lehrperson
- **Voraussetzung:** Lehrperson hat sich angemeldet
- **Haupterfolgsszenario**:
  1. Das System zeigt die Termine des Benutzers
  2. Die Lehrperson kann persönliche Termine einsehen
  3. Das Verwaltung benachrichtigt eine Lehrperson bezüglich einer Vertretung
  4. Die Lehrperson meldet bei der Verwaltung einen Ausfall

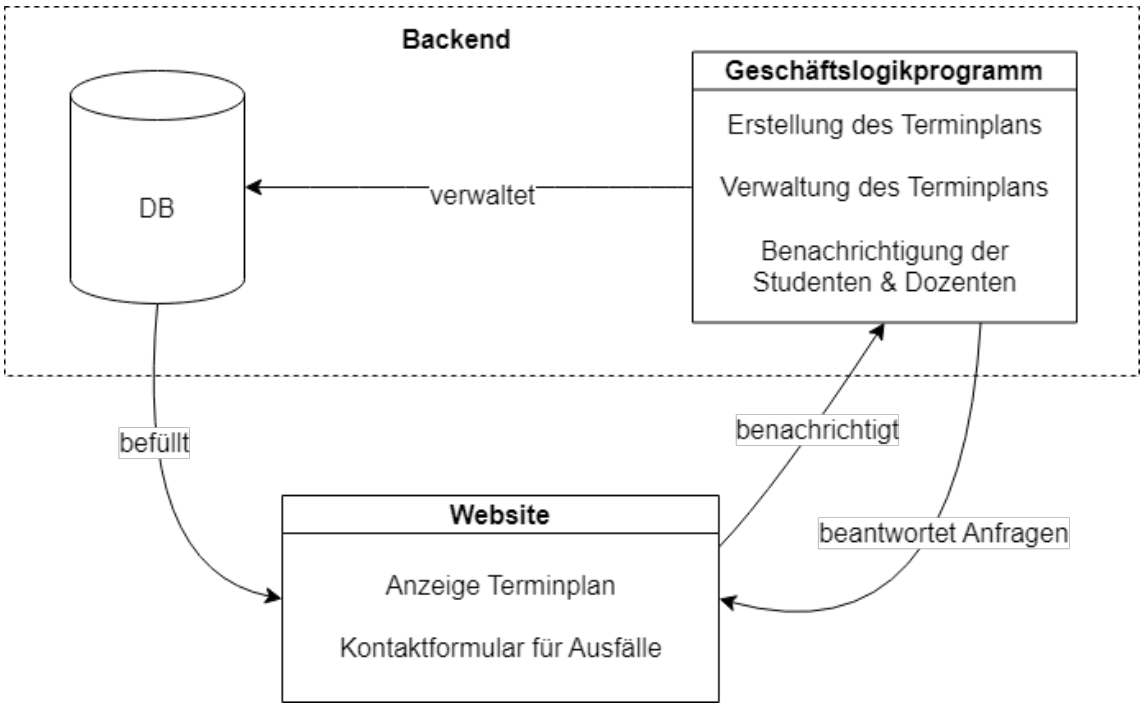
4 Architektur

Es wird ein Model-View-Controller entwickelt, d.h., eine Kombination aus Datenbankserver, Geschäftslogikprogramm und Benutzeroberfläche (Webseite). Der Datenbankserver (Model) enthält Daten zu Lehrpersonen, Studenten, Lehrveranstaltungen, Räumen und dem Terminplan. Das Geschäftslogikprogramm implementiert Funktionen zur Erstellung und Verwaltung des Terminplans sowie zur Kommunikation zwischen Model und View. Die Webseite fungiert als Benutzeroberfläche für Mitarbeiter und Studenten. Sie ermöglicht es diesen, den Terminplan einzusehen und Benachrichtigungen über Änderungen zu erhalten oder selbst zu erstellen.

4.1 High-Level System Design



4.2 Komponenten- & Konnektoransicht

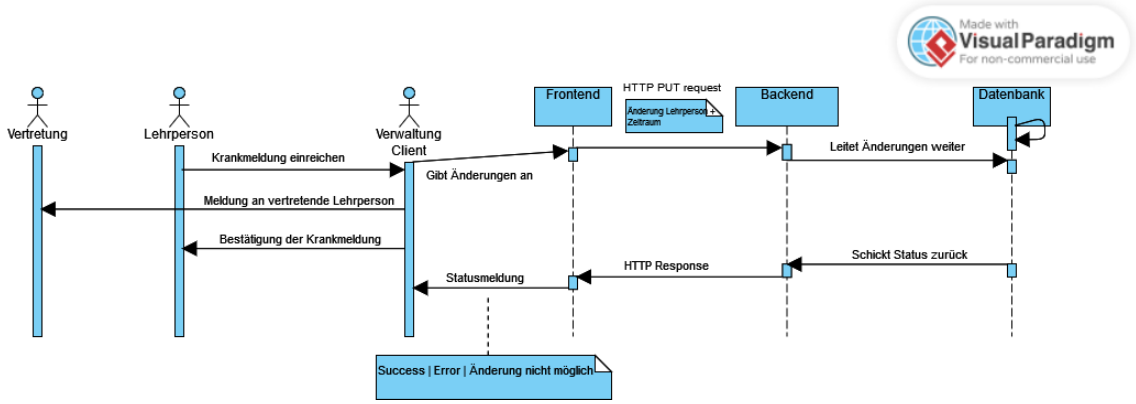


4.3 Model View Controller (MVC) - Details der Implementierung

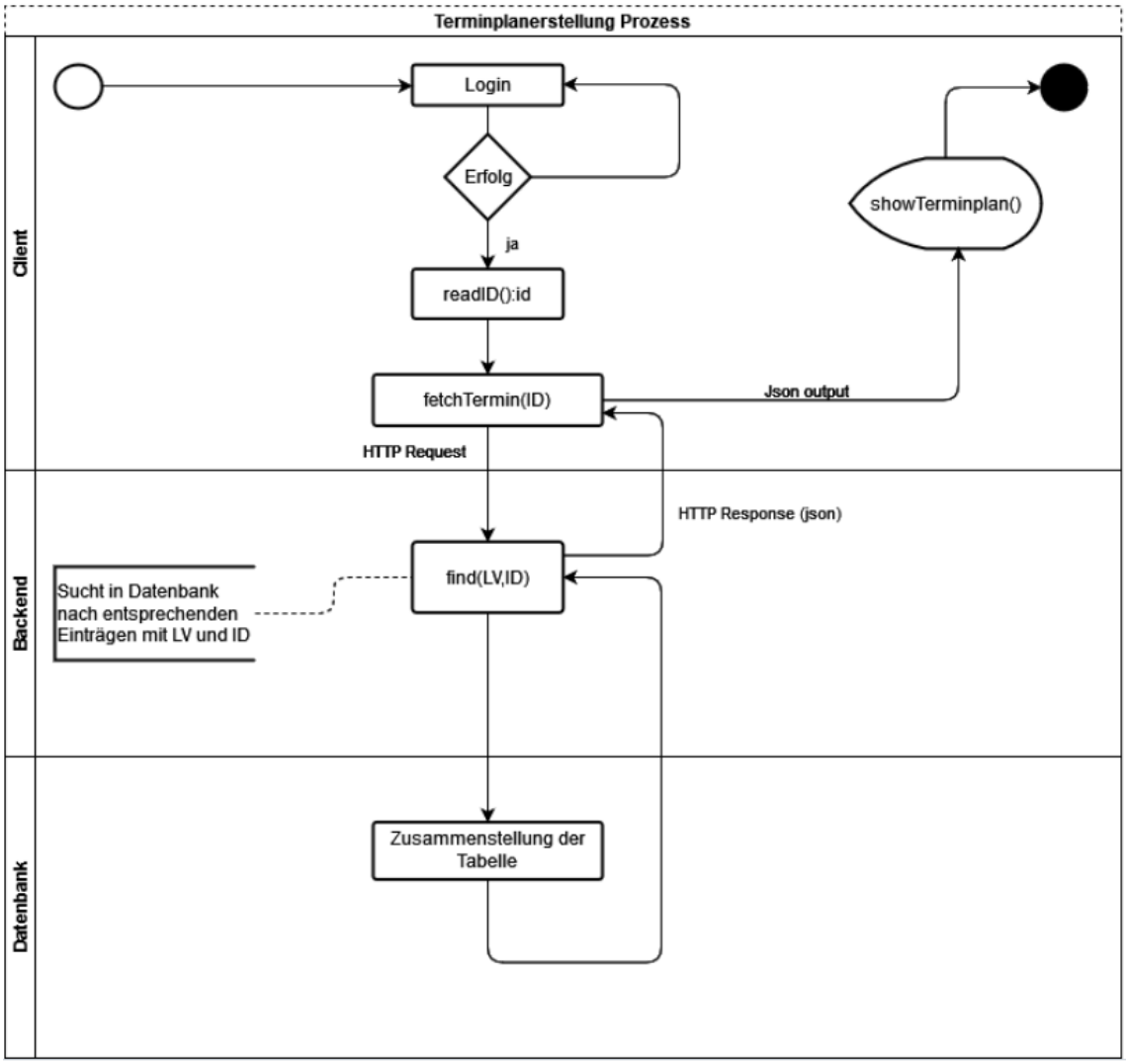
- PostgreSQL Database
  - Containerized in Docker
- Adminer GUI
  - Monitoring der DB View: Darstellung der Daten für die Anwender (Präsentation)
- React Frontend
  - Kalenderansicht
  - REST Methoden zum
    - Anfordern von Daten
    - Abschicken von Daten **Controller**: Vermittlung zwischen View und Model (Programmlogik)
- REST Controller
  - Spring Boot REST Controller
  - Spring Data JPA
  - Flyway SQL Version Control
- Maven Dependency Management

5 Geschäftsprozesse

5.1 Geschäftsprozess „Krankmeldung einer Lehrperson“

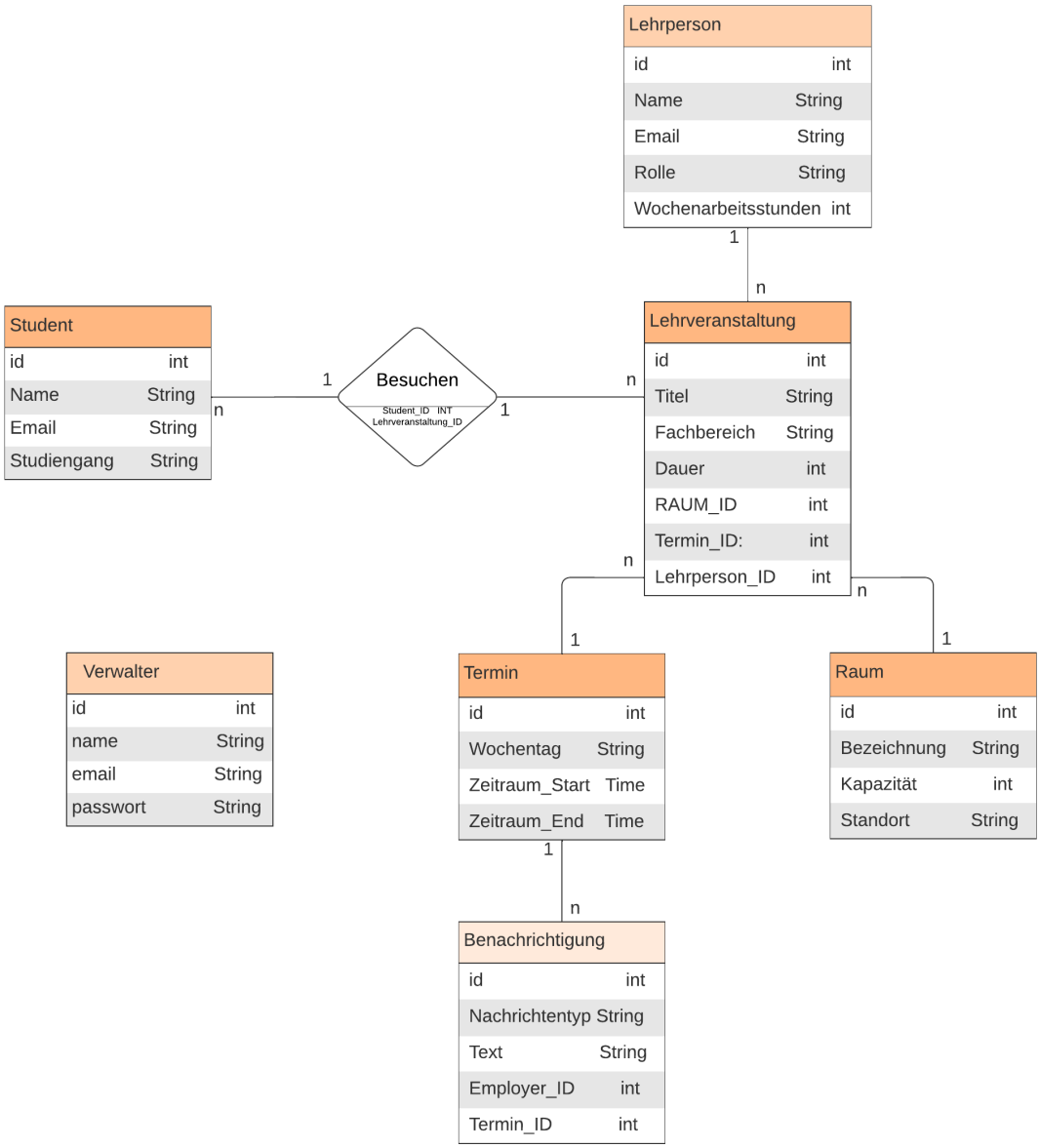


5.1.1 Geschäftsprozess „Terminplanerstellung“



6 Datenmodell

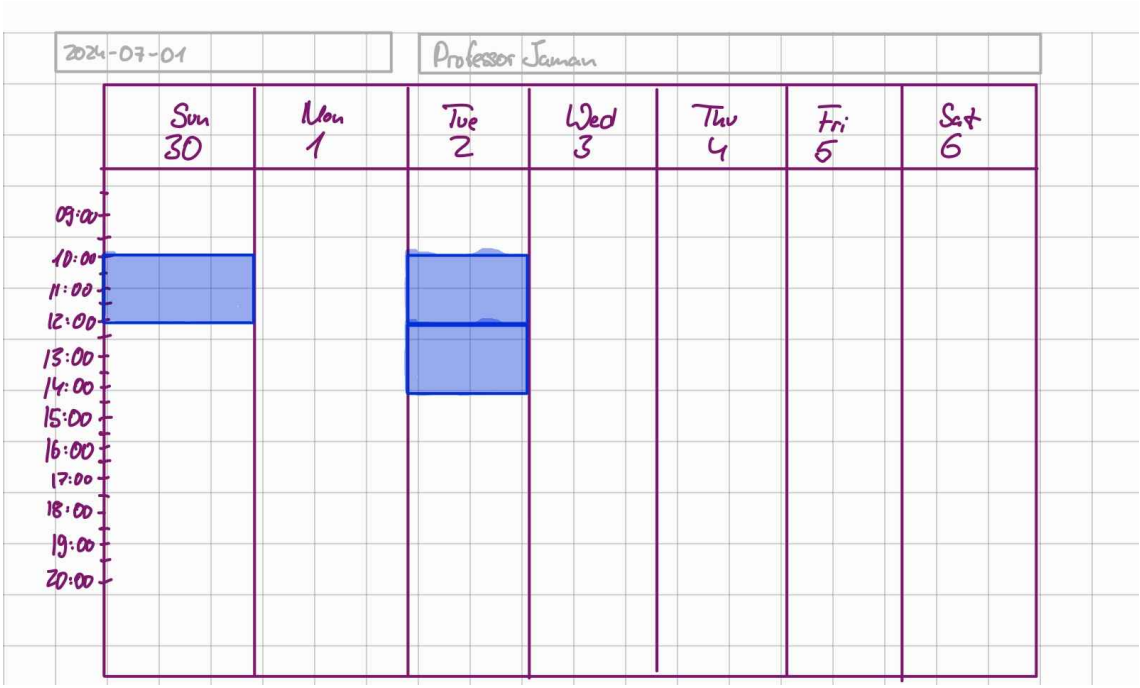
6.1 ER-Diagramm

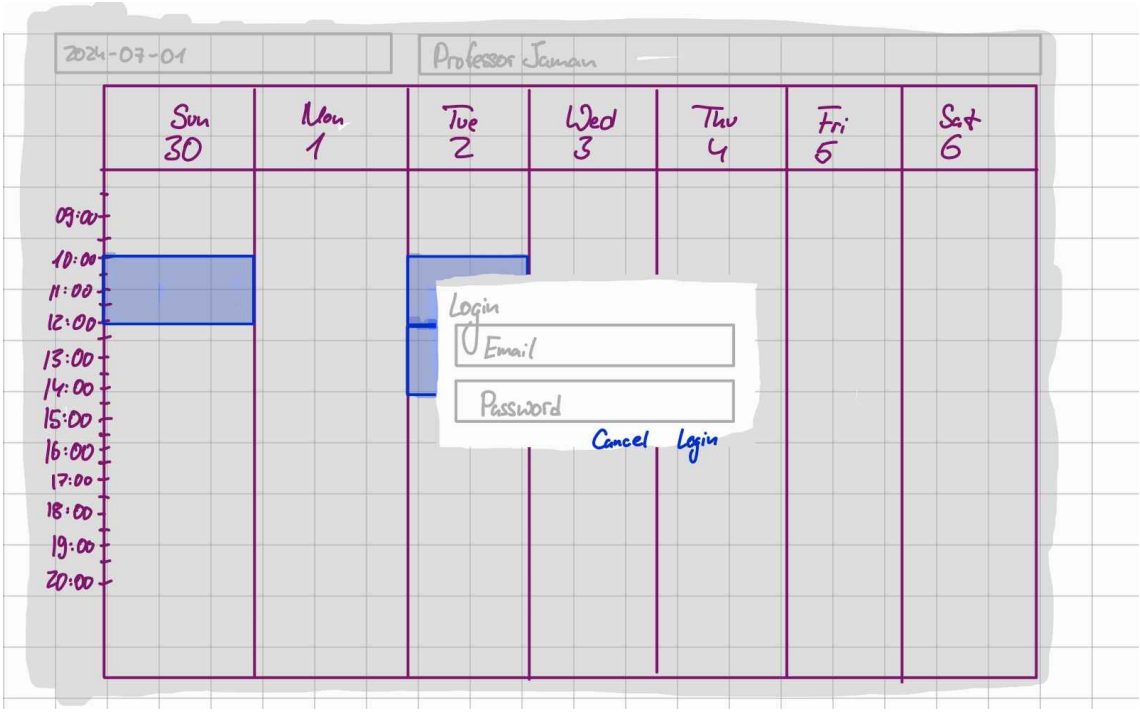


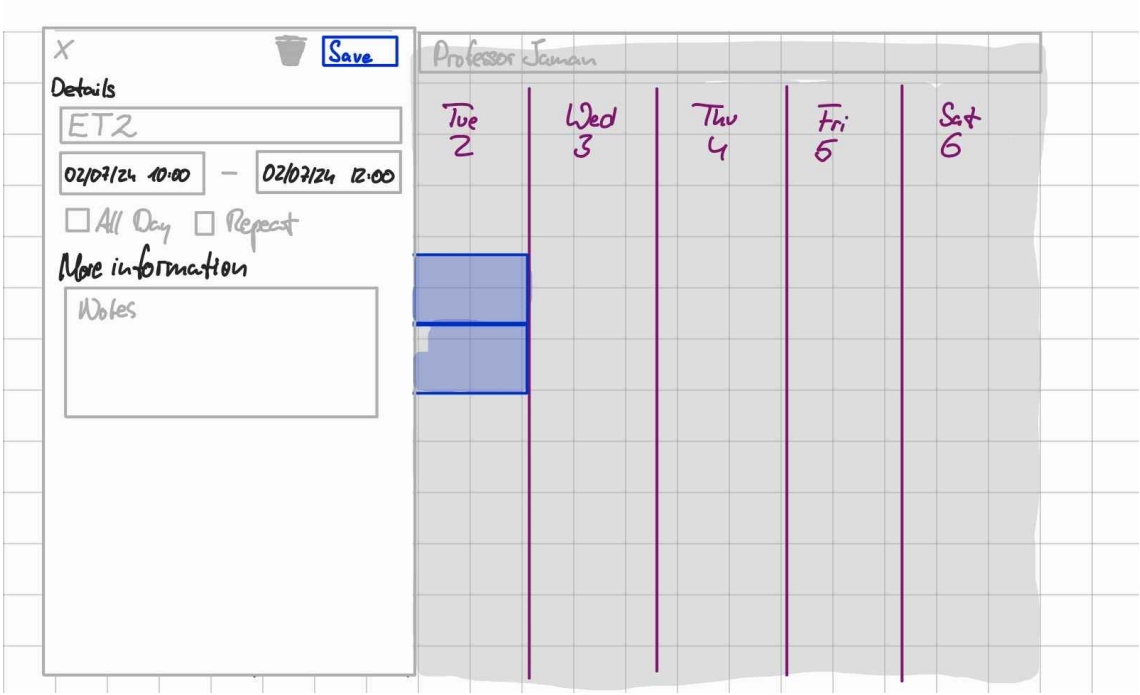
6.2 Form der Datenhaltung

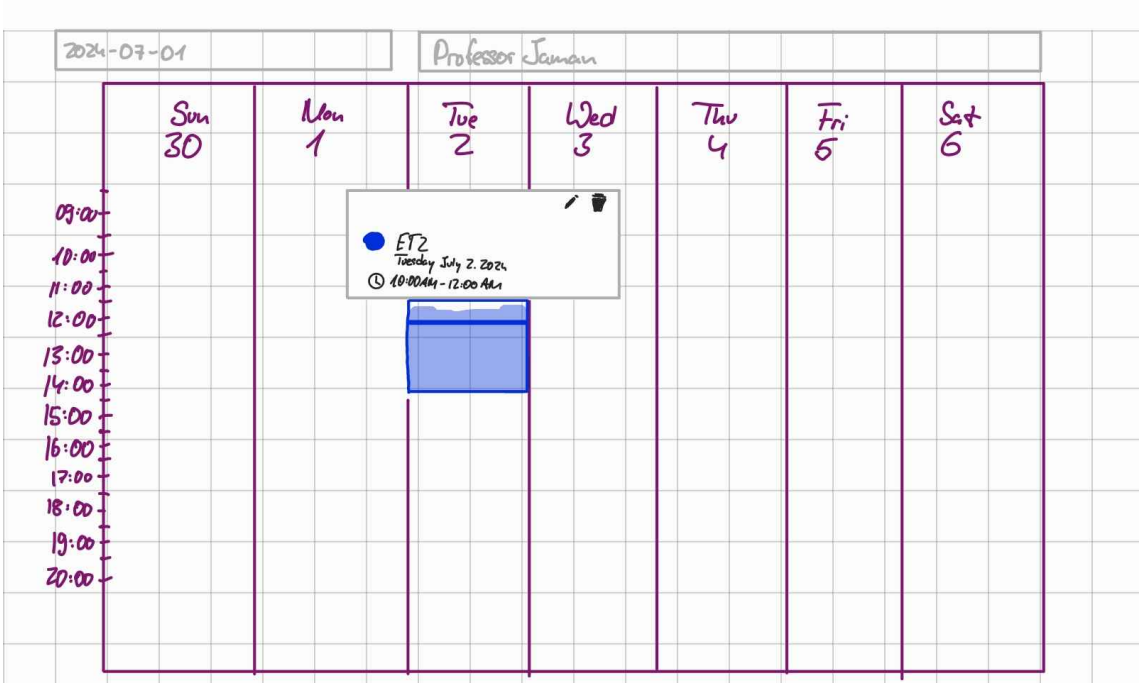
Wir haben uns für PostgreSQL entschieden, da das Projektteam mit diesem SQL-Dialekt gut vertraut ist und bereits erfolgreich Projekte damit abgeschlossen hat. Zur Verwaltung der Datenbank nutzen wir Adminer. Die Wahl fiel auf dieses Tool aufgrund der einfachen Installation, Konfiguration sowie seiner starken Performance.

7 GUI-Design

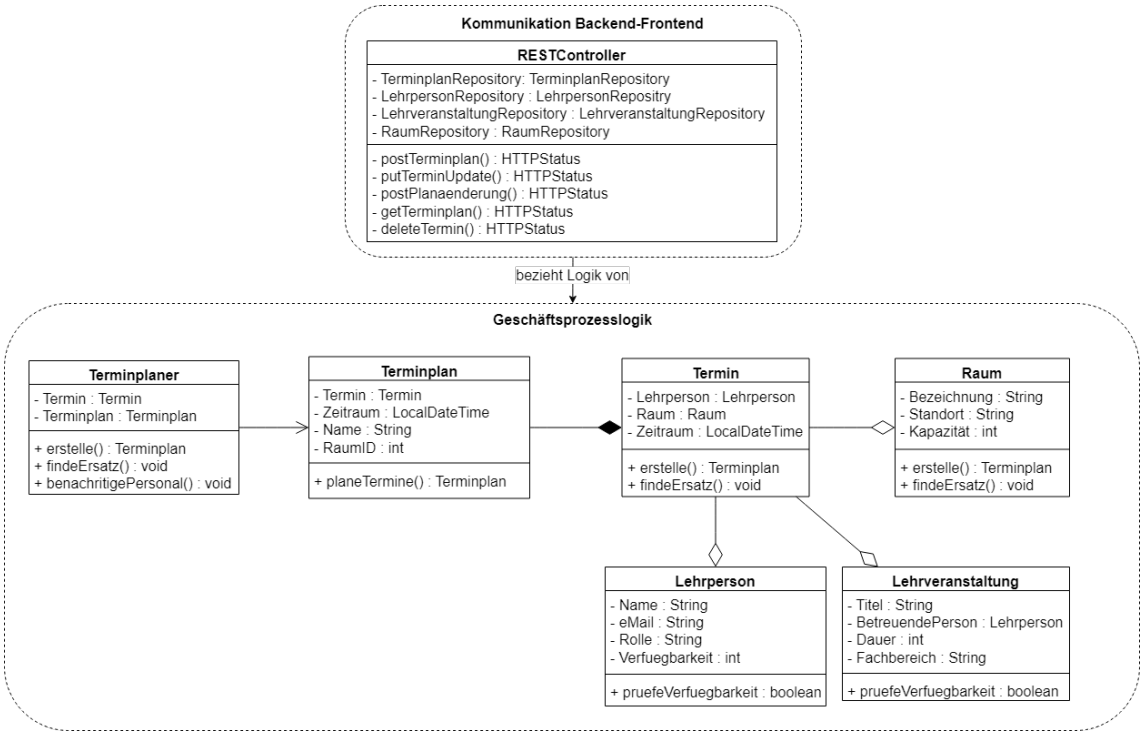

2024-07-01
Professor Jaman
Sun 30
Mon 1
Tue 2
Wed 3
Thu 4
Fri 5
Sat 6
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00


2024-07-01
Professor Jaman
Sun 30
Mon 1
Tue 2
Wed 3
Thu 4
Fri 5
Sat 6
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
Login
Email
Password
Cancel
Login


X
Save
Details
ET2
02/07/24 10:00
–
02/07/24 12:00
All Day
Repeat
More information
Notes
Professor Jaman
Tue 2
Wed 3
Thu 4
Fri 5
Sat 6


2024-07-01
Professor Jaman
Sun 30
Mon 1
Tue 2
Wed 3
Thu 4
Fri 5
Sat 6
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
ET2
Tuesday July 2. 2024
10:00 AM - 12:00 AM

8 Klassendiagramm



9 Implementierung

9.1 Implementierung - Funktionale Anforderungen

ID	Anforderung	Beschreibung	Prio.	Status
FA_1.0	**Terminplanung**	Die Software muss es ermöglichen, einen Einsatzplan für die Lehrpersonen zu erstellen, der zeigt, welche Lehrveranstaltungen sie betreuen sollen.	MUSS	Implementiert
FA_1.1	**Änderungsmanagement**	Die Software muss Änderungen im Plan berücksichtigen können, z.B. durch Krankmeldungen, Fortbildungen oder Tagungen der Mitarbeiter.	MUSS	Implementiert
FA_1.2	**Benachrichtigungen**	Lehrpersonen und Studierende sollen über Änderungen im Plan informiert werden.	SOLL	unvollendet
FA_1.3	**Benutzerzugriff**	Lehrpersonen und Studierende sollen auf ihre individuellen Pläne zugreifen können.	SOLL	Implementiert
FA_1.4	**Ausfall & Vetretung**	Bei Ausfall eines Lehrbeauftragten soll ein anderer eine Benachrichtung erhalten um diese veranstaltung zu vertreten.	SOLL	unvollendet
FA_1.5	**Sonderveranstaltungen**	Die Software soll Sonderveranstaltungen, wie z.B. Kurse für Firmen oder Schülergruppen, verwalten können.	KANN	Implementiert

FA_1.6	**Automatische Aktualisierung**	Die Software soll sich automatisch aktualisieren, wenn neue Informationen oder Änderungen vorliegen.	SOLL	unvollendet
FA_1.7	**KI-Unterstützung**	Mögliche Integration einer KI, um Sprachnachrichten automatisch zu transkribieren und in Textform an die Sekretärin weiterzuleiten.	KANN	unvollendet

1. **FA_1.0**: Eine Methode im REST-Controller wurde implementiert, die eine POST-Anfrage erwartet. Sobald diese empfangen wird, führt sie ein Mapping von Terminen, Räumen und Lehrpersonen zu Lehrveranstaltungen durch. Die Methode wartet darauf, dass eine entsprechende Anfrage gesendet wird.
2. **FA_1.1**: Eine Methode im REST-Controller wurde implementiert, die eine GET-Anfrage erwartet. Sobald diese empfangen wird, erhält die Verwaltung eine Liste verfügbarer Lehrpersonen. Anschließend kann sie die Termine auf der Website ändern.
3. **FA_1.3**: Auf der Website stehen zwei verschiedene Login-Optionen zur Verfügung: für Studierende und die Verwaltung. Jede Benutzergruppe hat spezifische Zugriffsrechte und individuelle Ansichten.
4. **FA_1.5**: Eine Methode im REST-Controller wurde implementiert, die eine POST-Anfrage erwartet. Sobald diese empfangen wird, wird eine Veranstaltung mit dem zugehörigen Raum, Termin und der Lehrperson hinzugefügt.

9.2 Implementierung - Nichtfunktionale Anforderungen

ID	Anforderung	Beschreibung	Status
NFA_1.0	**Benutzerfreundlichkeit**	Die Software soll eine intuitive Benutzeroberfläche haben.	Implementiert
NFA_2.0	**Zuverlässigkeit**	Die Software muss zuverlässig arbeiten und darf keine Ausfälle verursachen.	Implementiert
NFA_3.0	**Performance**	Die Software muss schnell reagieren und Pläne zügig aktualisieren können.	Implementiert
NFA_4.0	**Skalierbarkeit**	Die Software soll auch bei weiterem Wachstum der Hochschule problemlos funktionieren.	Implementiert
NFA_5.0	**Datensicherheit**	Die Software muss sicherstellen, dass alle Daten geschützt sind und nur autorisierte Benutzer Zugriff haben.	Implementiert

1. **NFA_1.0**: Mit React wurde eine benutzerfreundliche und moderne Oberfläche implementiert.
2. **NFA_2.0**: Es wurden verschiedene Exception-Handler integriert, um Problemfälle abzudecken.
3. **NFA_3.0**: Das Programm ist speichereffizient und benötigt nur wenig RAM. In den Testfällen überschritt die Reaktionszeit nie 5 Sekunden.
4. **NFA_4.0**: Alle relevanten Tabellen sind erweiterbar und basieren auf leicht erweiterbaren Frameworks wie React und Spring Boot.
5. **NFA_5.0**: Die Datenbank ist durch Logins geschützt, wobei nur die Verwaltung Zugriff auf die Daten hat.

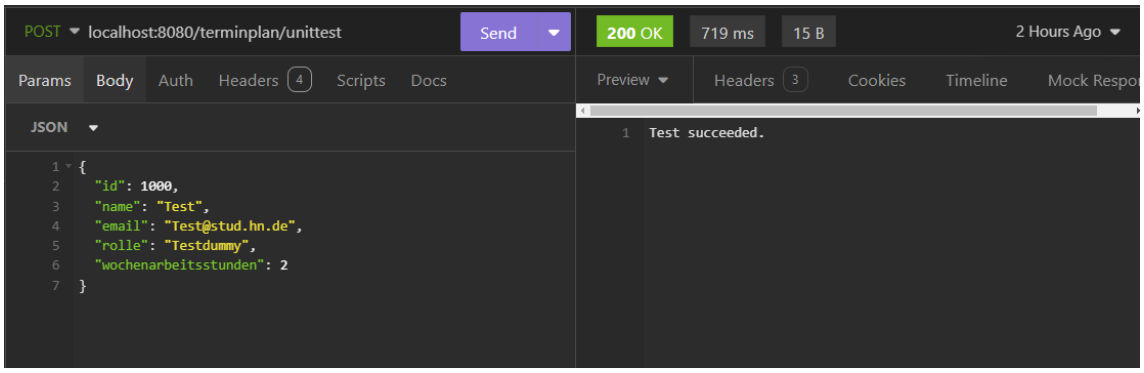
9.3 Implementierung - Abweichungen

1. Es wurde eine neue Relation Vertretung hinzugefügt, um die Ausfälle von Lehrpersonen abzuspeichern.
2. Jede Relation hat im RESTController ein JpaRepository injiziert bekommen.
3. Die Geschäftsprozess-Methoden wurden innerhalb des RESTControllers implementiert (Terminplanerstellung, Ausfallmeldung).
4. Es wurden mehrere Testmethoden für den RESTController erstellt (Reset, Unit-Test).
5. Es wurden Exception-Klassen für Lehrperson und Lehrveranstaltung hinzugefügt.
6. Die CRUD-Methoden sind nur partiell im Front-End implementiert worden.
7. Lehrpersonen können sich nicht einloggen, allerdings kann man über ein Dropdown für spezifische Lehrpersonen einen Terminplan filtern.
8. Es gibt noch weitere Methoden im RESTController, welche die üblichen CRUD Funktionalitäten für die Datenbank ermöglichen. Diese sind für den Verbraucher nicht direkt nutzbar, außer über bestimmte freigegebene Interface Schnittstellen.

10 Test

10.1 Unit Test

Es wurde ein Unit Test geschrieben der die INSERT, SELECT und DELETE Funktionen von JPA für eine Lehrperson testet. Das Backend wird über eine POST Request an die Schnittstelle http://localhost:8080/terminplan/unittest mit einer JSON für eine Lehrperson angesprochen. Eine Antwort im Response Body informiert den Anwender über einen erfolgreichen oder fehlgeschlagenen Test.



10.2 Eingabefeld Test

Beim Log-In für die Verwaltung wird geprüft, ob die E-Mail mit „@hs-niederrhein.deëndet. Falls sie es nicht tut, wird ein Fehler in der Konsole vermerkt.