

Projekt Softwaretechnik  
**Bewerbungstracker**

Jannika Börner, Marcus Klein, Norman Münzer, Lukas Reinhardt,  
Lara Seline Hippenstiel

Alexander Brendel, Klemens Morbe  
Prof. Dr. Jörg Nitzsche

Studiengang: Softwaretechnik und Medieninformatik  
Fakultät: Informationstechnik  
Hochschule Esslingen  
Wintersemester 2025/26

---

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>IV</b>
<b>1 Einführung und Kontext</b>	<b>1</b>
1.1 Problemstellung . . . . .	1
1.2 Lösungsansatz . . . . .	1
1.3 Zielgruppe . . . . .	1
1.4 Rahmenbedingungen . . . . .	1
1.5 Lizenz . . . . .	1
<b>2 Anforderungen</b>	<b>2</b>
2.1 Geforderter Funktionsumfang . . . . .	2
2.2 User Stories . . . . .	3
2.2.1 Entwickler . . . . .	3
2.2.2 Benutzer . . . . .	3
<b>3 UI-Entwürfe</b>	<b>5</b>
3.0.1 Erster UI-Entwurf . . . . .	5
3.0.2 Verfeinerung des UI-Konzepts . . . . .	5
<b>4 Architektur</b>	<b>10</b>
4.1 Systemarchitektur . . . . .	10
4.2 Komponenten . . . . .	10
4.2.1 Präsentationsschicht . . . . .	10
4.2.2 Identitäts- und Zugriffsmanagement . . . . .	13
4.2.3 Anwendungsschicht . . . . .	14
4.2.4 Datenschicht . . . . .	16
4.3 Verhalten . . . . .	17
4.3.1 Authentifizierung . . . . .	17
4.3.2 Darstellung . . . . .	18
4.3.3 Eingabe . . . . .	19
4.4 Schnittstellenbeschreibung . . . . .	22
4.4.1 Verwendete Schnittstellentechnologie . . . . .	22
4.4.2 Übersicht der Stellenausschreibungen . . . . .	22
4.4.3 Detaillierte Stellenausschreibung ansehen . . . . .	24
4.4.4 Stellenausschreibung hinzufügen und bearbeiten . . . . .	28
4.4.5 Stellenausschreibung löschen . . . . .	31
4.4.6 Übersicht der Termine . . . . .	33
4.5 Entscheidungen . . . . .	34
4.5.1 Verwendete Technologien / Frameworks . . . . .	34
4.5.2 Struktur . . . . .	36
<b>5 Teamorganisation</b>	<b>37</b>
5.1 Projektmanagement . . . . .	37
5.1.1 Meetings . . . . .	37
5.1.2 Kommunikation . . . . .	37

## Inhaltsverzeichnis

5.2	Aufwandsschätzung . . . . .	37
5.3	Aufgabenteilung . . . . .	40
5.3.1	Lukas . . . . .	40
5.3.2	Norman . . . . .	40
5.3.3	Marcus . . . . .	41
5.3.4	Lara . . . . .	41
5.3.5	Jannika . . . . .	41
5.4	Verwendete Hilfsmittel . . . . .	42
<b>6</b>	<b>Retrospektive</b>	<b>43</b>
6.1	Schwierigkeiten . . . . .	43
6.1.1	Fehlende Vorerfahrung . . . . .	43
6.1.2	Ausfälle durch Krankheit . . . . .	43
6.2	Entwicklung . . . . .	43
6.3	Teamorganisation . . . . .	43
6.3.1	Kommunikation . . . . .	43
6.3.2	Board . . . . .	43
<b>7</b>	<b>Installationshandbuch</b>	<b>45</b>
7.1	Einleitung . . . . .	45
7.2	Voraussetzungen . . . . .	45
7.3	Installation . . . . .	45
<b>8</b>	<b>Ausblick</b>	<b>47</b>
8.1	Weitere Funktionalitäten . . . . .	47
<b>9</b>	<b>Appendix</b>	<b>i</b>
9.1	Erstes Datenbankkonzept . . . . .	i
9.2	Linksammlung . . . . .	i
9.3	Protokoll 07. Oktober 2025 . . . . .	ii
9.4	Protokoll 17. Oktober 2025 . . . . .	xiii
9.4.1	Kundengespräch . . . . .	xiii
9.4.2	Betreuergespräch . . . . .	xiii
9.5	Protokoll 24. Oktober 2025 . . . . .	xiv
9.5.1	Kundengespräch . . . . .	xv
9.5.2	Betreuergespräch . . . . .	xvi
9.6	Protokoll 31. Oktober 2025 . . . . .	xvii
9.6.1	Kundengespräch . . . . .	xvii
9.6.2	Betreuergespräch . . . . .	xviii
9.7	Protokoll 07. November 2025 . . . . .	xix
9.7.1	Kundengespräch . . . . .	xx
9.7.2	Betreuergespräch . . . . .	xxi
9.8	Protokoll 14. November 2025 . . . . .	xxii
9.8.1	Kundengespräch . . . . .	xxii
9.8.2	Betreuergespräch . . . . .	xxiii
9.9	Protokoll 21. Monat 2025 . . . . .	xxiv
9.9.1	Kundengespräch . . . . .	xxv
9.9.2	Betreuergespräch . . . . .	xxvi
9.10	Protokoll 28. November 2025 . . . . .	xxvii
9.10.1	Kundengespräch . . . . .	xxvii
9.10.2	Betreuergespräch . . . . .	xxviii

## *Inhaltsverzeichnis*

9.11 Protokoll 05. Dezember 2025 . . . . .	xxix
9.11.1 Kundengespräch . . . . .	xxx
9.11.2 Betreuergespräch . . . . .	xxxi
9.12 Protokoll 12. Dezember 2025 . . . . .	xxxi
9.12.1 Kundengespräch . . . . .	xxxi
9.12.2 Betreuergespräch . . . . .	xxxii
9.13 Protokoll 09. Januar 2026 . . . . .	xxxiii
9.13.1 Kundengespräch . . . . .	xxxiii
9.13.2 Betreuergespräch . . . . .	xxxv
9.14 Protokoll 16. Januar 2026 . . . . .	xxxvi
9.14.1 Kundengespräch . . . . .	xxxvi
9.14.2 Betreuergespräch . . . . .	xxxvii
9.15 Erweiterte Eigenständigkeitserklärung . . . . .	xxxviii

# Abbildungsverzeichnis

3.1	Bevorzugtes 1. Wireframe: Bewerbungsseite . . . . .	6
3.2	Ausgearbeitetes Wireframe Kalender . . . . .	7
3.3	Ausgearbeitetes Wireframe Bewerbungen . . . . .	7
3.4	Ausgearbeitetes Wireframe Unternehmensansicht . . . . .	8
3.5	Ausgearbeitete Wireframe Datenerfassung . . . . .	8
3.6	Ausgearbeitetes Wireframe Dokumente . . . . .	9
4.1	Architektur: Zentrale Komponenten . . . . .	10
4.2	Struktursicht auf Code Ebene des Frontends: Übersicht der Seiten . . . . .	11
4.3	Struktursicht auf Code Ebene des Frontends: Funktionalität Bewerbungen . . . . .	12
4.4	Struktursicht auf Code Ebene des Frontends: Funktionalität Termine . . . . .	13
4.5	Struktursicht auf Code-Ebene (Backend) . . . . .	15
4.6	Entity-Relationship-Modell . . . . .	16
4.7	Anmeldung über Keycloak . . . . .	17
4.8	Darstellung der Stellenübersicht . . . . .	18
4.9	Abruf der Daten für die Stellenübersicht . . . . .	19
4.10	Eingabe der Stelle . . . . .	20
4.11	Speichern der Eingabe . . . . .	21
4.12	Datenmodell der Joboffer Overview Schnittstelle . . . . .	23
4.13	Status der Axios Anfrage . . . . .	23
4.14	Joboffer Details API (1/2) . . . . .	26
4.15	Joboffer Details API (2/2) . . . . .	27
4.16	Add Joboffer API (1/2) . . . . .	30
4.17	Add Joboffer API (2/2) . . . . .	31
4.18	Datenmodell der Appointment Overview Schnittstelle . . . . .	34
5.1	Aufwandsschätzung Gantt . . . . .	38
5.2	Vergleich von Zeitschätzung und tatsächlicher Zeiterfassung Stand 21.01.2026	40
9.1	Erstes ER-Modell . . . . .	i

# 1 Einführung und Kontext

## 1.1 Problemstellung

Bei der Jobsuche werden viele Bewerbungen für unterschiedliche Stellenangebote verfasst und verschickt. Im Laufe des Bewerbungsprozesses müssen Daten, wie Termine, Adressen, Ansprechpartner, Gehalt und mehr, verwaltet werden. Diese Daten befinden sich dabei in der Regel verstreut in Notizen, Mails, Zetteln und Spreadsheets. Dadurch verliert man sehr leicht die Übersicht und die Organisation wird schnell zum Chaos. Die Folge ist ein stressiger und unstrukturierter Bewerbungsprozess.

## 1.2 Lösungsansatz

Mit einer zentralen Anwendung können alle relevanten Informationen an einem Ort verwaltet werden. Durch eine intuitive Benutzeroberfläche wird das Sammeln und Eintragen von Informationen effizient gestaltet und der Nutzer erhält einen gut strukturierten Überblick. Mit Hilfe von persönlichen Bewertungen und Notizen vom Anwender, kann dieser bei der Entscheidungsfindung unterstützt werden.

## 1.3 Zielgruppe

Die Zielgruppe umfasst Schüler, Studierende, Berufseinsteiger und Berufserfahrene, die sich beruflich umorientieren möchten. Diese müssen ihre Bewerbungen oft unter Druck und mit begrenzter Zeit verfassen und verwalten. Einige, wie beispielsweise Studierende, müssen noch anderen Verpflichtungen nachkommen. Insbesondere diese benötigen eine Lösung, die ihnen bei der Strukturierung des Bewerbungsprozesses unter die Arme greift.

## 1.4 Rahmenbedingungen

Das Projekt entsteht im Rahmen eines Hochschulprojekts mit einem Team aus fünf Personen. Die Projektdauer beträgt ein Semester, wodurch die zeitlichen Ressourcen begrenzt sind. Als Richtwert für den Zeitaufwand sind 12 bis 15h pro Person und Woche angesetzt. Das Projekt soll als Open-Source Web-Anwendung entwickelt werden. Die Authentifizierung soll, mit Keycloak als Identity- und Access-Management-System erfolgen. Ansonsten obliegt die Wahl der Technologie dem Team.

## 1.5 Lizenz

Verwendet wurden in diesem Projekt Frameworks mit **MIT** und **Apache 2.0** Lizenzen. Für weitere Verwendung haben wir uns für die **MIT** Lizenz entschieden, da sie inhaltlich kompakter ist.

# 2 Anforderungen

## 2.1 Geforderter Funktionsumfang

In der App soll es möglich sein folgende Informationen bezüglich eines Unternehmens einzutragen und zu verwalten:

- Name und Logo des Unternehmens
- Kontaktperson
- Leistungen/Benefits
- Gehaltsspielraum
- Distanz zum Wohnort
- Mitarbeiteranzahl
- Persönliches Rating
- Eigene Notizen
- Kurze Unternehmensbeschreibung

Zusätzlich soll es möglich sein, diese Basisinformationen von einer KI sammeln und eintragen zu lassen.

Die App soll basierend auf den erfassten Informationen und der Gewichtung unterschiedlicher Schwerpunkte, ausgewählt durch den Nutzer, bei der Entscheidungsfindung helfen. Hierzu soll eine Entscheidungsmatrix benutzt werden können.

Des Weiteren ist es gewünscht, dass Dokumente bezüglich der Bewerbung innerhalb der App hochgeladen und gesammelt werden.

Die Daten sollen in einer Datenbank abgespeichert werden und es soll dem Nutzer möglich sein, diese Daten von jedem beliebigen Gerät aus mithilfe seiner Login-Daten abzurufen. Außerdem sind weitere Optionen für bessere Zugänglichkeit erwünscht:

- ein Dark & Light Theme
- Mehrsprachigkeit

Das gesamte Projekt soll Open-Source sein und inklusive vollständiger Dokumentation veröffentlicht werden.

Aus dem 1. Gesprächstermin (siehe Protokoll vom 07.10.25) ergaben sich zusätzlich folgende Anforderungen:

- Funktion zum Sortieren und Filtern nach z. B. Termin, Gehalt, Bewerbungsdatum, Unternehmensgröße, Entfernung, etc.
- Settings mit Basisinformationen über den Nutzer

## 2.2 User Stories

### 2.2.1 Entwickler

#### **BT-1 – Verwaltung Daten in Datenbank via Spring Boot**

Als Entwickler möchte ich die Daten in der Datenbank über das Backend verwalten können, um den Usern später zu ermöglichen die Daten selbst zu verwalten.

#### **BT-7 – Datenübertragung zwischen den Technologien ermöglichen**

Als Entwickler möchte ich in der Lage sein, Daten zwischen den verschiedenen Technologien zu übertragen, um mit diesen arbeiten zu können.

#### **BT-8 – Datenbank SQL-Skript (Schema & Sample Daten) erstellen**

Als Entwickler möchte ich eine festgelegte und klare Struktur für die Daten in der Datenbank haben, um diese dort ordentlich und effizient verwalten zu können. Zudem möchte ich einen Datensatz mit Testdaten, um mit diesen verschiedene Tests durchführen zu können. Die Datenbank sollte automatisch beim Start des Programms aufgesetzt werden, um mir die Zeit des manuellen Aufsetzens zu sparen.

#### **BT-9 – Keycloak integrieren**

Als Entwickler möchte ich außerdem ein AIM (Keycloak) in die App integrieren, um mir den Aufwand ein eigenes, sicheres Anmeldesystem zu erstellen zu ersparen.

#### **BT-11 – UI Komponenten erstellen**

Als Entwickler möchte ich wiederverwendbare UI-Komponenten zur Verfügung stehen haben, um individuelle Ansichten mit weniger Aufwand zu erstellen.

### 2.2.2 Benutzer

#### **BT-2 – KI-Integration zum Extrahieren der Basisinformationen implementieren**

Als Benutzer möchte ich, dass eine KI grundlegende Informationen zu einer Stellenanzeige in die App überträgt, um mir das manuelle Eingeben zu ersparen.

#### **BT-3 – Englische Übersetzung/Spracheinstellungen erstellen**

Als Benutzer möchte ich, dass die App in mehreren Sprachen verfügbar ist, um die App in meiner bevorzugten Sprache verwenden zu können.

#### **BT-4 – Dark & Light Mode implementieren**

Als Benutzer möchte ich, dass die App sowohl in einem dunklen als auch in einem hellen Design verfügbar ist, um das Design nach meinem bevorzugten Stil einstellen zu können.

#### **BT-5 – Entscheidungsmatrix implementieren**

Als Benutzer möchte ich eine Entscheidungsmatrix zur Verfügung haben, um bessere Entscheidungen bezüglich Jobangeboten treffen zu können.

**BT-6 – UI Komponenten auf der Seite einfügen/anordnen**

Als Benutzer möchte ich eine intuitive Web-App-Oberfläche, um die Web-App problemlos navigieren zu können. Hierfür möchte ich Konsistenz zwischen den einzelnen Ansichten, um mich nicht jedes Mal erneut orientieren zu müssen.

**BT-10 – Dokumentenablage/Dokumentenspeicherlösung erstellen**

Als Benutzer möchte ich eine Dokumentenspeicherlösung, um meine Dokumente zusammen mit Stellenangeboten an einem Ort zu speichern.

# 3 UI-Entwürfe

## 3.0.1 Erster UI-Entwurf

Bei Betrachtung der ersten Mock-Ups (siehe Abbildungen 3.1) wurden einige Punkte beschlossen:

- Name: Bewerbungstracker statt Jobtracker
- Leiste oben mit Widgets
- Neue Bewerbungen hinzufügen durch Pop-Up Menü oder ähnliches mit Eingabemöglichkeiten

## 3.0.2 Verfeinerung des UI-Konzepts

### Kalender

In der überarbeiteten Version dienen der Kalender und die Terminübersicht gleichzeitig zur Übersicht über alle Termine (siehe Abbildung 3.2). Im oberen Bereich befindet sich, wie bei der ersten UI-Version beschlossen, eine Navigationsleiste, über die man zwischen den Hauptbereichen “Termine”, “Bewerbungen” und “Dokumenten” wechseln kann. Darunter befindet sich eine Navigationsleiste mit einer kompakten Terminübersicht in Form einer Liste. Hier können alle Termine eingetragen und tabellarisch mit Informationen zu Unternehmen, Art des Termins, Datum, Uhrzeit und auch möglichen To-Dos angezeigt werden. Termine können über das Plus-Symbol hinzugefügt werden, wobei die Verwaltung einfach über die Bearbeiten und Löschen-Symbole bei jedem Eintrag machbar ist.

Im unteren Bereich ist ein Kalender-Widget, das die Termine des aktuellen Monats visuell darstellt. Termine werden farblich nach Typ unterschieden, wodurch der Nutzer einen schnellen Überblick über alle Fristen und Termine bekommen kann.

### Bewerbungen

Im zweiten Hauptbereich, “Bewerbungen”, (siehe Abbildung 3.3) erhält man eine Übersicht über alle Bewerbungen. Im oberen Bereich befindet sich weiterhin die Navigationsleiste. Darunter folgen drei Bedienfelder mit einer Suchfunktion, einer Sortier- und Filterfunktion und einem Bedienfeld um neue Bewerbungen hinzuzufügen. Unter den Bedienfeldern wird der Seiteninhalt in Kachelansicht angezeigt. Jede Kachel repräsentiert eine Stellenausschreibung, für die eine Bewerbung eingereicht wurde. In der Kachel sind die Jobbezeichnung, das Logo des Unternehmens, sowie der Firmenname und der nächste relevante Termin zu sehen. Durch das Klicken auf eine Bewerbung kommt man in die Übersicht des Unternehmens und des Jobs (siehe Abbildung 3.4). In dieser Ansicht werden Informationen zum Unternehmen und zur beworbenen Position dargestellt. Im oberen Bereich erscheinen das Firmenlogo sowie ein Bewertungssystem in Form von Sternen, das anzeigt, wie gut der Job und das Unternehmen den persönlichen Wünschen entsprechen. Oben links in dem Fenster sieht man einen Pfeil um wieder zurück in die Bewerbungsübersicht zu kommen. Oben rechts sind Buttons zum Bearbeiten und Löschen von der aktuellen Bewerbung zu

finden. Wieder zurück in der Kachelansicht kann über den Hinzufügen-Button ein Formular zum Erfassen aller relevanten Bewerbungsdaten geöffnet werden (siehe Abbildung 3.5).

## Dokumente

Im letzten Hauptbereich der Anwendung, “Dokumente” (siehe Abbildung 3.6), befindet sich die Dokumentenablage. Hier werden alle Dokumente zu den Bewerbungen angezeigt und verwaltet. Über das Bedienfeld im oberen Bereich der Seite kann man neue Dokumente hochladen. Darunter werden alle bereits hochgeladenen Dokumente in Kachelansicht gelistet. In jeder Kachel stehen das Upload-Datum und die Art des Dokuments. Die Dokumente sind über Bearbeitungs- und Löschsymbole in den jeweiligen Kacheln verwaltbar, wodurch eine einfache und intuitive Handhabung ermöglicht wird.

## Startseite

Die Startseite ist eine Kombination aller wichtigsten Elemente der oberen 3 Tabs. Angezeigt werden sollen sowohl die Liste aller bevorstehenden Termine, sowie die Bewerbungs-Kacheln und Dokumenten-Kacheln und einer Vorschau einer Entscheidungsmatrix.

Für die Zwischenpräsentation wurden die Wireframes klickbar gemacht.

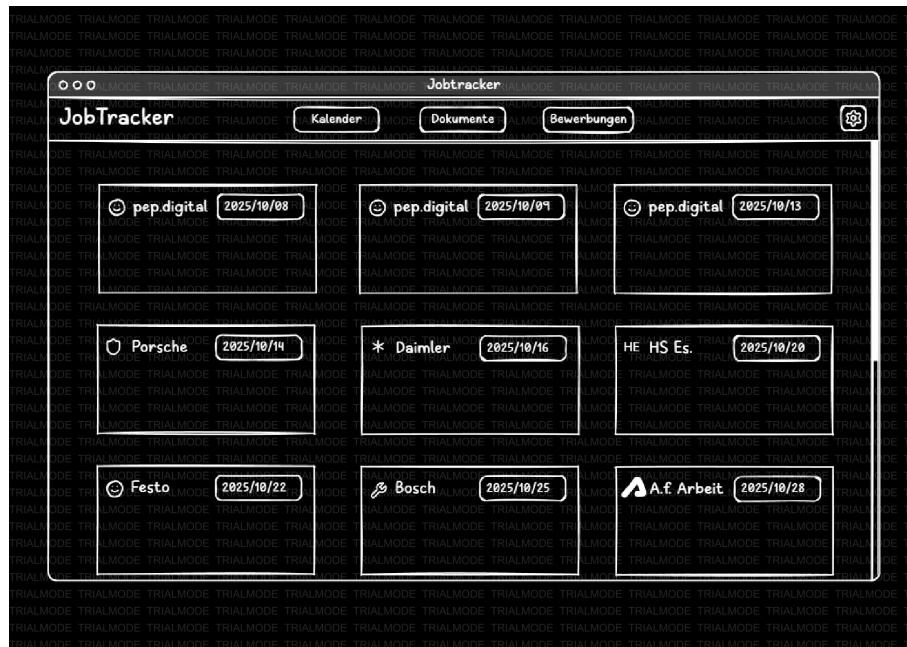


Abbildung 3.1: Bevorzugtes 1. Wireframe: Bewerbungsseite

### 3 UI-Entwürfe

**Bewerbungstracker**

**Termine**      Bewerbungen      Dokumente      ☰

Termine +

Unternehmen	Art des Termins	Datum	Uhrzeit	To Do	
Firma 1	● Bewerbungsgespräch	20.10.2025	14:00	-	<span style="color: #ccc;">/ <span style="font-size: small;">Zeugnis noch hinzufügen</span></span>
Firma 2	● Abgabe Bewerbungsunterlage	20.10.2025	-	Zeugnis noch hinzufügen	<span style="color: #ccc;">/ <span style="font-size: small;">Zeugnis noch hinzufügen</span></span>

Kalender

Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag	Sonntag
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	● Firma 1 Bew ● Firma 2 Abi	21	22	23	24	25
27	28	29	30	31	1	2

Abbildung 3.2: Ausgearbeitetes Wireframe Kalender

**Bewerbungstracker**

**Bewerbungen**      Dokumente      ☰

Unternehmen

Suchen nach ...    Sortieren und Filtern nach ...    + Bewerbung hinzufügen

Stellenausschreibung <small>Firma nächster Termin: dd-mm-yy</small>	Bsp Firma 2 <small>nächster Termin: dd-mm-yy</small>	Bsp Firma 3
Bsp Firma 4	Bsp Firma 5	Bsp Firma 6

Abbildung 3.3: Ausgearbeitetes Wireframe Bewerbungen

### 3 UI-Entwürfe

**Bewerbungstracker**

- Termine
- Bewerbungen**
- Dokumente
- 

---

< Beispiel-Firma

Bearbeiten
 Löschen

☆ ★ ★ ★ ★
nächster Termin dd-mm-yy

„Das Unternehmen ist ein moderner Dienstleister, der sich durch Innovationskraft, Flexibilität und kundenorientiertes Handeln auszeichnet. Mit einem breiten Leistungsspektrum und hoher fachlicher Kompetenz positioniert es sich als verlässlicher Partner in seiner Branche.“

Kontaktperson	Max Mustermann max.mustermann@firma1.bsp Bsp Telefonnummer
Gehaltsspielraum	48.000 € - 58.000 € brutto jährlich – abhängig von Qualifikation und Erfahrung
Distanz	42km bzw. 1,5 Stunden
<ul style="list-style-type: none"> <li>• Flexible Arbeitszeiten</li> <li>• Homeoffice- oder Remote-Work-Möglichkeiten</li> <li>• Mitarbeiter-Events und Teambuilding</li> </ul> <small>Unternehmen beschreibt weitere Vorteile</small>	

Abbildung 3.4: Ausgearbeitetes Wireframe Unternehmensansicht

**Bewerbungstracker**

- Termine
- Bewerbungen**
- Dokumente
- 

---

< Unternehmen hinzufügen

von KI ausfüllen lassen
 Zugehörige Dokumente hinzufügen
 Verwerfen

Name des Unternehmens

+

Persönliches Rating

☆ ★ ★ ★ ★

Unternehmensbeschreibung

Termine

+ weitere Termine hinzufügen

Adresse

Abbildung 3.5: Ausgearbeitete Wireframe Datenerfassung

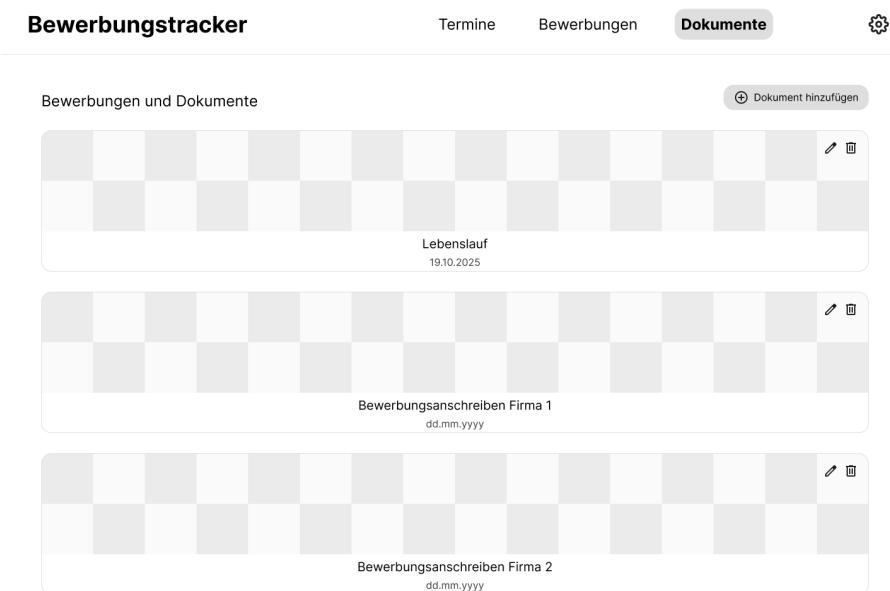


Abbildung 3.6: Ausgearbeitetes Wireframe Dokumente

# 4 Architektur

## 4.1 Systemarchitektur

Die Architektur der Software setzt sich aus vier zentralen Komponenten zusammen, welche in Docker betrieben werden. Im Diagramm (siehe Abbildung 4.1) erkennt man wie das Frontend, das Backend, das Authentifizierungssystem und die Datenbank miteinander interagieren.

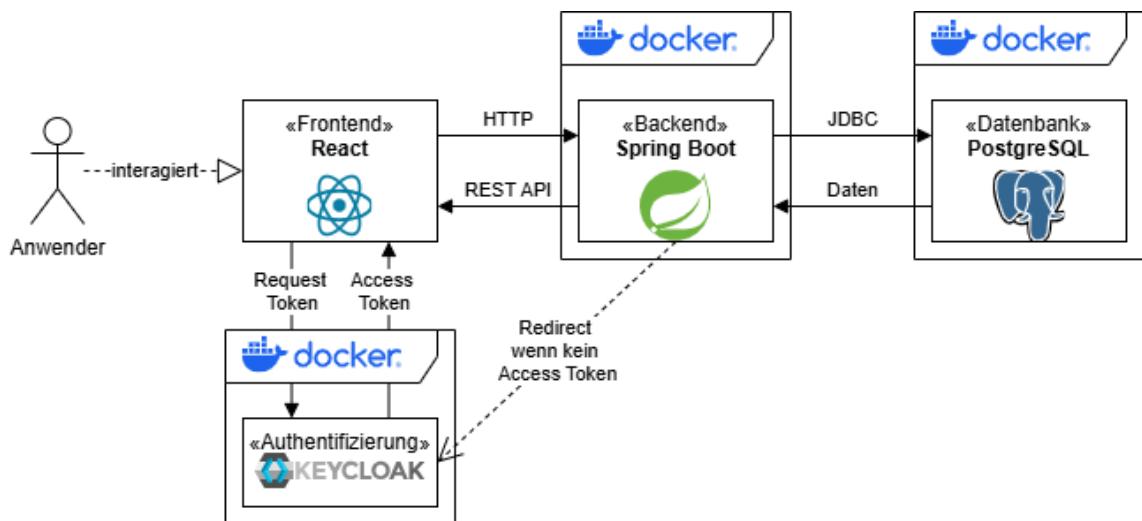


Abbildung 4.1: Architektur: Zentrale Komponenten

## 4.2 Komponenten

### 4.2.1 Präsentationsschicht

- Läuft im Browser des Nutzers, wo dieser mit der Weboberfläche interagiert
- Leitet beim ersten Zugriff zur Authentifizierung (Keycloak) weiter
- Erhält vom IAM (Keycloak) ein Access Token nach erfolgreicher Anwendung  
→ Token wird bis Session-Ende für alle weitere Anfragen an das Backend (Spring Boot) genutzt

## Seitenübersicht

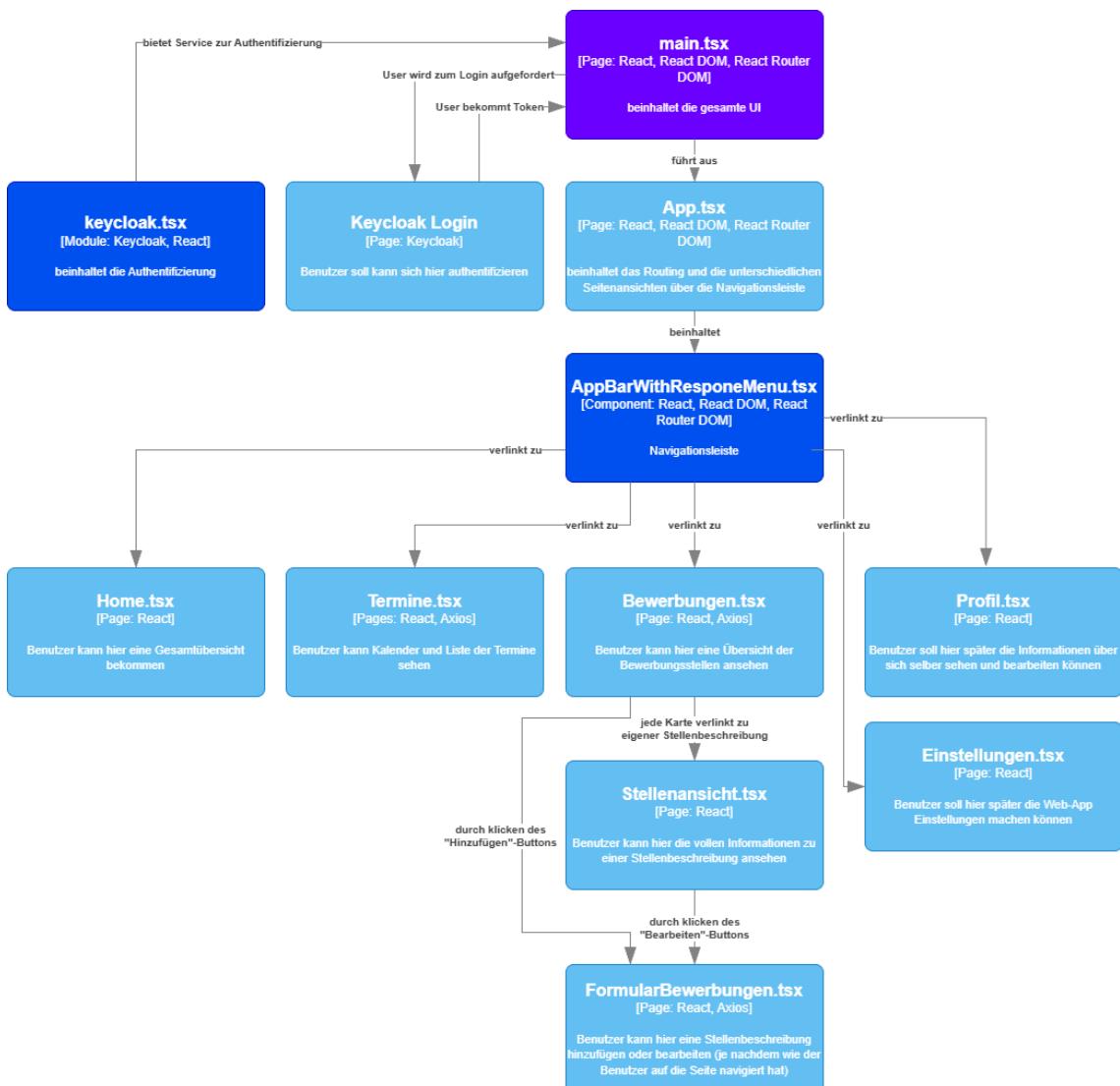


Abbildung 4.2: Struktursicht auf Code Ebene des Frontends: Übersicht der Seiten

Nach dem erfolgreichen Login, wird dem Benutzer die Startseite **Home.tsx** angezeigt, auf der er eine Schnellübersicht aller anstehenden Termine und aller hinterlegten Stellenanzeigen erhält. Mithilfe der Tabs der Navigationsleiste **AppBar.tsx** kann auf die Seiten **Termine** und **Bewerbungen** navigiert werden. Zusätzlich können **Profil** und **Einstellungen** durch das Klicken des **Profil-Icons** als Pop-Up angezeigt werden. Die Seiten **Termine.tsx** und **Bewerbungen.tsx**, die den Großteil der Funktionalitäten enthalten, werden in den folgenden Grafiken näher erklärt.

## Funktionalität Bewerbungen

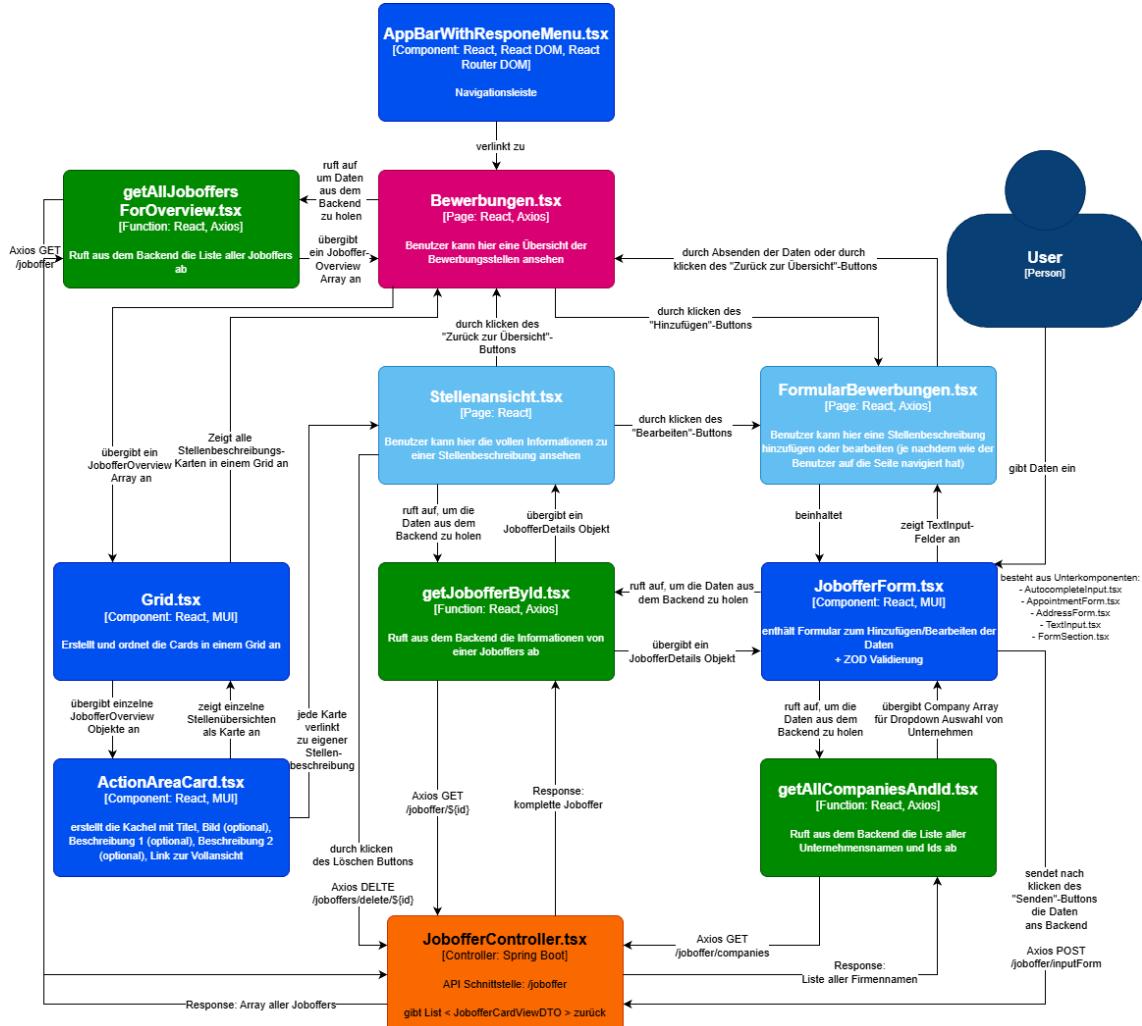


Abbildung 4.3: Struktursicht auf Code Ebene des Frontends: Funktionalität Bewerbungen

Auf der Seite Bewerbungen.tsx wird die Übersicht aller hinterlegten Stellenanzeigen (Joboffers) angezeigt.

Dazu werden mithilfe der Axios-Anfrage in getAllJoboffersForOverview.tsx die Daten aus dem Backend abgerufen. Diese werden dann in ActionAreaCard als Kachel angezeigt, pro Joboffer jeweils eine Kachel. Diese Kacheln werden als Grid angezeigt.

Durch das Anklicken einer Kachel wird dann die Stellenansicht.tsx aufgerufen. Dort können alle Daten, die bezüglich einer Joboffer abgespeichert wurden, angezeigt werden. Die Daten werden mithilfe von getJobofferById.tsx basierend auf der im Link übergebenen Joboffer-Id und dem User-Token aus dem Backend abgerufen.

Von der Stellenansicht.tsx aus, können die Joboffers über Buttons gelöscht oder bearbeitet werden.

Beim Bearbeiten wird die Seite FormularBewerbungen.tsx aufgerufen. Je nach dem, ob man diese Seite über den Bearbeiten-Button auf der Stellenansicht.tsx oder den Hinzufügen-Button auf der Bewerbungen.tsx aufruft, wird das Formular anders verwendet. Die JobofferForm.tsx die auf dieser Seite angezeigt wird besteht aus verschiedenen Untergeordneten Komponenten für spezialisierte Input-Felder.

Die Funktionen aus getCompaniesAndId werden genutzt, um bei dem Text-Input zur Fir-

ma bereits vermerkte Firmen anzuzeigen, damit der Benutzer diese nicht jedes mal neu eingeben muss.

Alle Backend-Anfragen gehen an den JobofferController.tsx im Backend.

## Funktionalität Termine

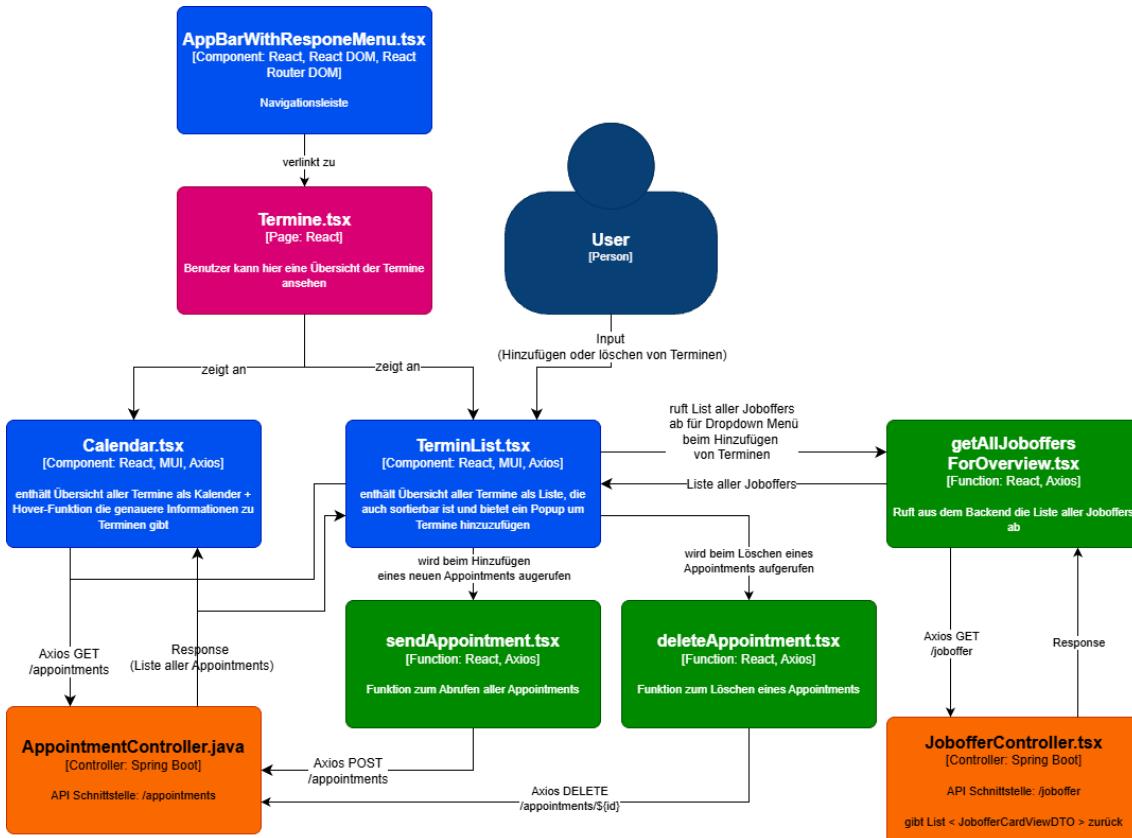


Abbildung 4.4: Struktursicht auf Code Ebene des Frontends: Funktionalität Termine

Auf der Seite Termine.tsx wird sowohl die Kalender als auch die Terminliste aufgerufen und nebeneinander dargestellt.

Mithilfe einer Axios-Anfrage innerhalb der jeweiligen Komponenten werden alle Termine abgefragt und im Kalender angezeigt. Innerhalb der Liste wird jedoch gefiltert und nur zukünftige Termine angezeigt.

Zusätzlich können in der Liste noch Termine gelöscht werden und auf der Seite Termine hinzugefügt.

### 4.2.2 Identitäts- und Zugriffsmanagement

- Das Identity- und Access-Management-System (IAM) Keycloak übernimmt Authentifizierung
- Überprüft Anmelde Daten und gibt bei erfolgreichem Login ein Access Token an das Frontend (React) zurück

### 4.2.3 Anwendungsschicht

#### Beschreibung

- Stellt REST-API-Endpunkte als Schnittstelle bereit.
- Ist zuständig für die Anwendungslogik
- Empfängt Anfragen des Frontends und authentifiziert diese anhand eines JWT-Tokens.
- Ruft Datenbankoperationen über JDBC auf.

#### Struktur

Die Pakete Anwendungsschicht sind nach Features untergliedert. Diese sind Address, Appointment, App-User, Joboffer und Contact, wobei Contact dem Joboffer Paket unterordnet ist.

Der Kern der Anwendungsschicht besteht aus REST-Controllern die HTTP-Anfragen aus dem Frontend entgegen nehmen. Diese rufen dann ihren jeweiligen Service auf, welcher die Logik enthält. Vom Service wird dann das Repository aufgerufen um Daten aus der Datenbank zu lesen oder die neuen oder veränderten Daten zu speichern. Der Datenbankzugriff erfolgt durch JPA und Hibernate.

Die Abbildung 4.5 zeigt die Struktur der einzelnen Komponenten der Anwendung. Zu sehen sind der Controller für Termine und Stellen, sowie die Services und die Repositories. Die Joboffer, also Stelle, steht dabei im Mittelpunkt und ruft die anderen Services auf. Der Grund dafür ist, dass die Logik für die Erstellung der einzelnen Teilentitäten in den jeweiligen Services erfolgen soll um redundanten Code zu vermeiden und klare Zuständigkeiten zu schaffen.

#### 4 Architektur

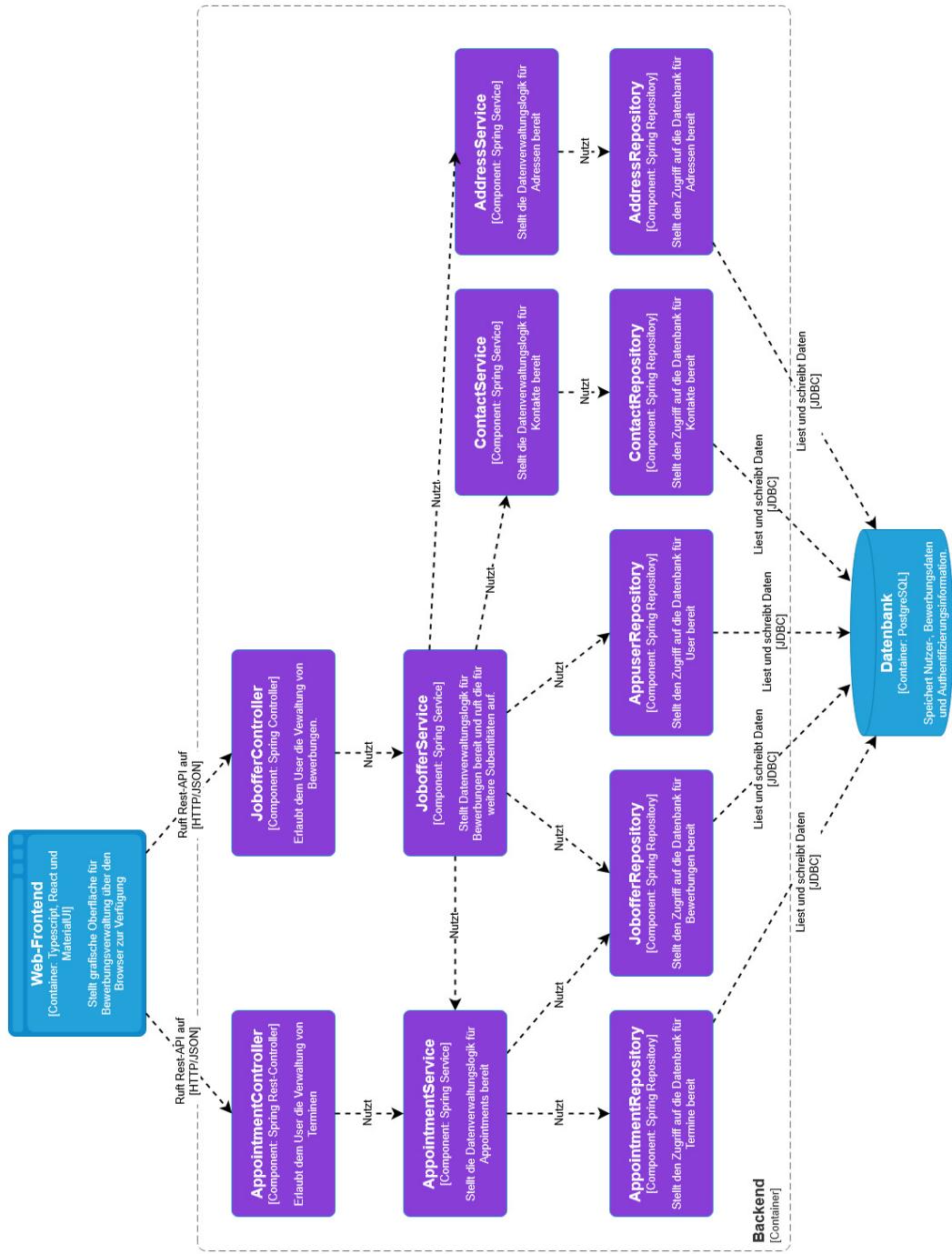


Abbildung 4.5: Struktursicht auf Code-Ebene (Backend)

#### 4.2.4 Datenschicht

- Strukturelles Speichern der Daten
- Kommuniziert über JDBC mit der Anwendungsschicht (Spring Boot)  
→ Ermöglicht das Abrufen, Speichern und Aktualisieren der Nutzerdaten und Anwendungsinformationen

#### Logisches Datenmodell

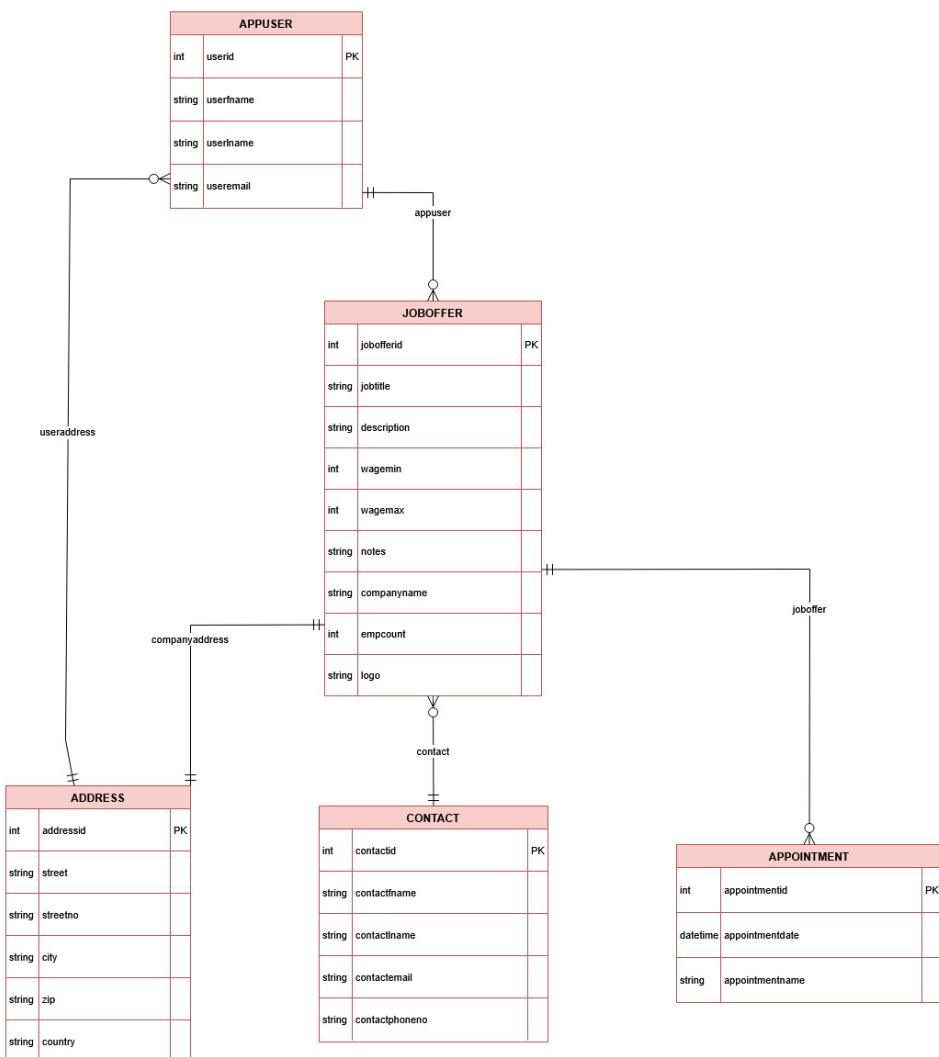


Abbildung 4.6: Entity-Relationship-Modell

## 4.3 Verhalten

Der typische Ablauf einer Interaktion besteht aus den folgenden Schritten:

1. Nutzer öffnet die Webanwendung.
2. Nutzer meldet sich an.
3. Nutzer interagiert mit der Oberfläche.
4. Anfrage mit Access-Token wird an die Anwendungsschicht gesendet.
5. Anwendungsschicht überprüft das Token.
6. Daten werden je nach Operation gelesen, geändert oder gespeichert.
7. Antwort wird an die Präsentationsschicht gesendet.

### 4.3.1 Authentifizierung

Beim Aufrufen der Webanwendung wird der Access-Token überprüft. Ist keiner vorhanden, oder der Token ist veraltet oder falsch, wird der Nutzer auf die Keycloak Login-Page weitergeleitet. Sollte bereits ein Nutzer bestehen kann sich einfach angemeldet werden, wenn jedoch noch kein eigener Nutzer existiert muss eine Registrierung vorgenommen werden. Die daraus gewonnenen Daten werden in der PostgreSQL Datenbank gespeichert und eine Weiterleitung an die eigentliche Webanwendung, zum Ausgangspunkt, geschieht. Im Backend werden nach dem erstellen eines neuen Nutzers die Daten aus der Keycloak Tabelle abgefragt und als Nutzer in die Backend Datenbank übertragen, sollte beim ersten Versuch das nicht gelingen, so wird bei jedem folgenden Anmelden versucht die Daten zu übernehmen.

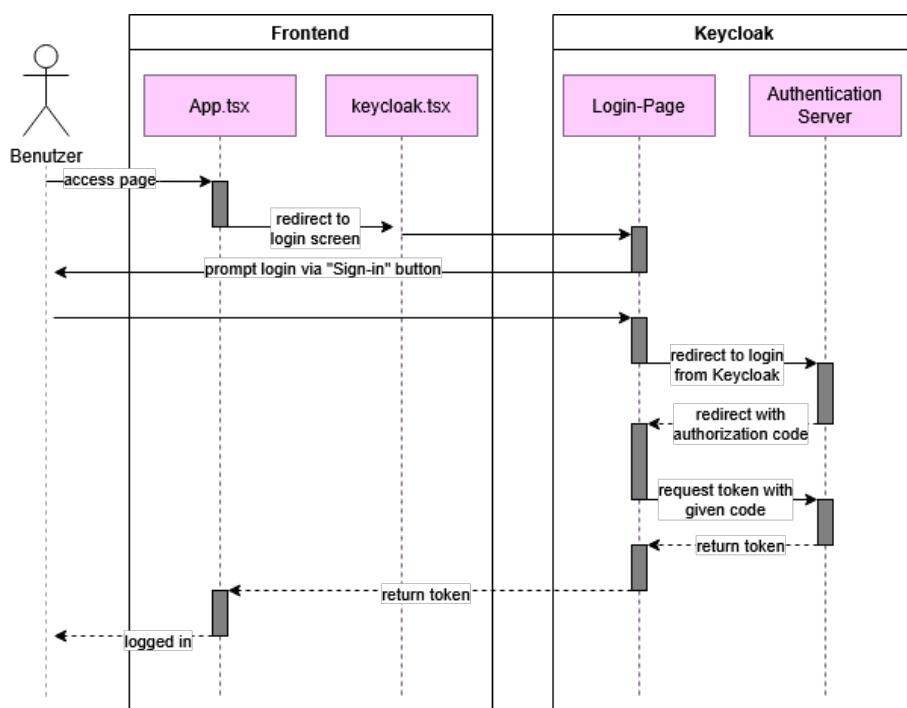


Abbildung 4.7: Anmeldung über Keycloak

In der Abbildung 4.7 wird beschrieben wie ein Nutzer die Webanwendung aufruft und zum Keycloak Login weitergeleitet wird. Hier meldet sich der Nutzer mit einem bereits bestehenden Konto an. Die Daten werden an den Authentication Server weitergeleitet, welcher einen Autorisierungscode zurückgibt und nach Tokenrequest einen autorisierten Token zurückgibt, mit welchem sich der Nutzer auf der Webanwendung frei bewegen kann.

### 4.3.2 Darstellung

Nach einem erfolgreichen Login landet der Nutzer auf einer Startseite, auf der die Bewerbungen und Stellen angezeigt werden. Mit Hilfe einer Navigationsleiste kann man zu einer Terminübersicht oder einer Stellenübersicht wechseln. Mit einem Klick auf einen speziellen Termin oder eine Stelle wird man auf die Detailansicht zur Stelle weitergeleitet. Beim wechseln zwischen den Seiten erfolgt ein Aufruf zur Schnittstelle der Anwendungsschicht über eine GET-Anfrage mit HTTP. In Abbildung 4.8 wird dieser Teil bis zum Aufruf der Anwendungsschicht am Beispiel der Stellenübersicht dargestellt.

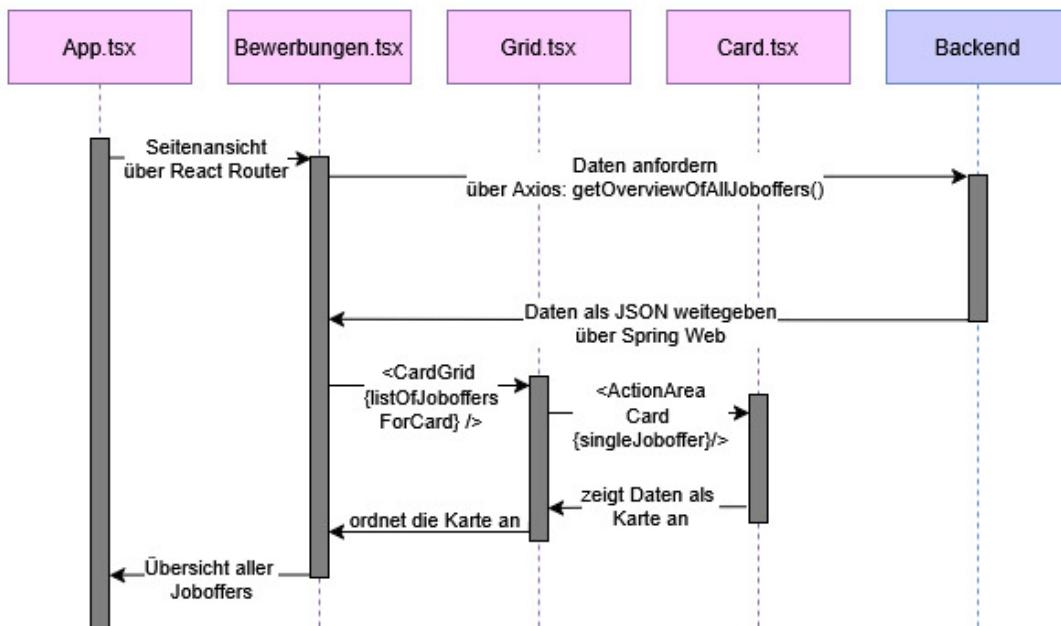


Abbildung 4.8: Darstellung der Stellenübersicht

Die Abbildung 4.9 zeigt, wie in der Anwendungsschicht der entsprechende Controller aufgerufen wird. Dieser ruft die Servicefunktion auf, welche über ein Repository die Daten aus der Datenbank liest und diese an den Controller zurückgibt. Die Daten werden mittels Jackson in ein JSON umgewandelt und an die Präsentationsschicht zurückgesendet.

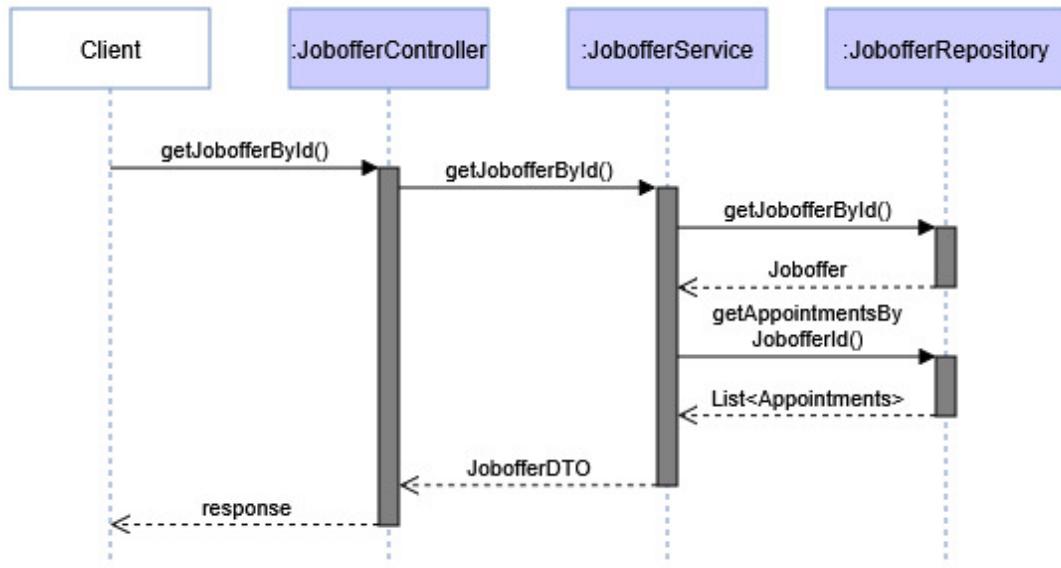


Abbildung 4.9: Abruf der Daten für die Stellenübersicht

#### 4.3.3 Eingabe

Die Eingabe erfolgt über eine weitere Seite oder eine Dialog-Komponente die sich öffnet wenn ein entsprechender Button betätigt wurde. Dann kann der Nutzer Daten in mehrere Felder eingeben. Es wird validiert ob Pflichtfelder ausgefüllt wurden. Wenn alles ausgefüllt ist und der Speichern-Button gedrückt wird werden die Daten als JSON an die Anwendungsschicht gesendet. Dieser Vorgang ist in Abbildung 4.10 dargestellt.

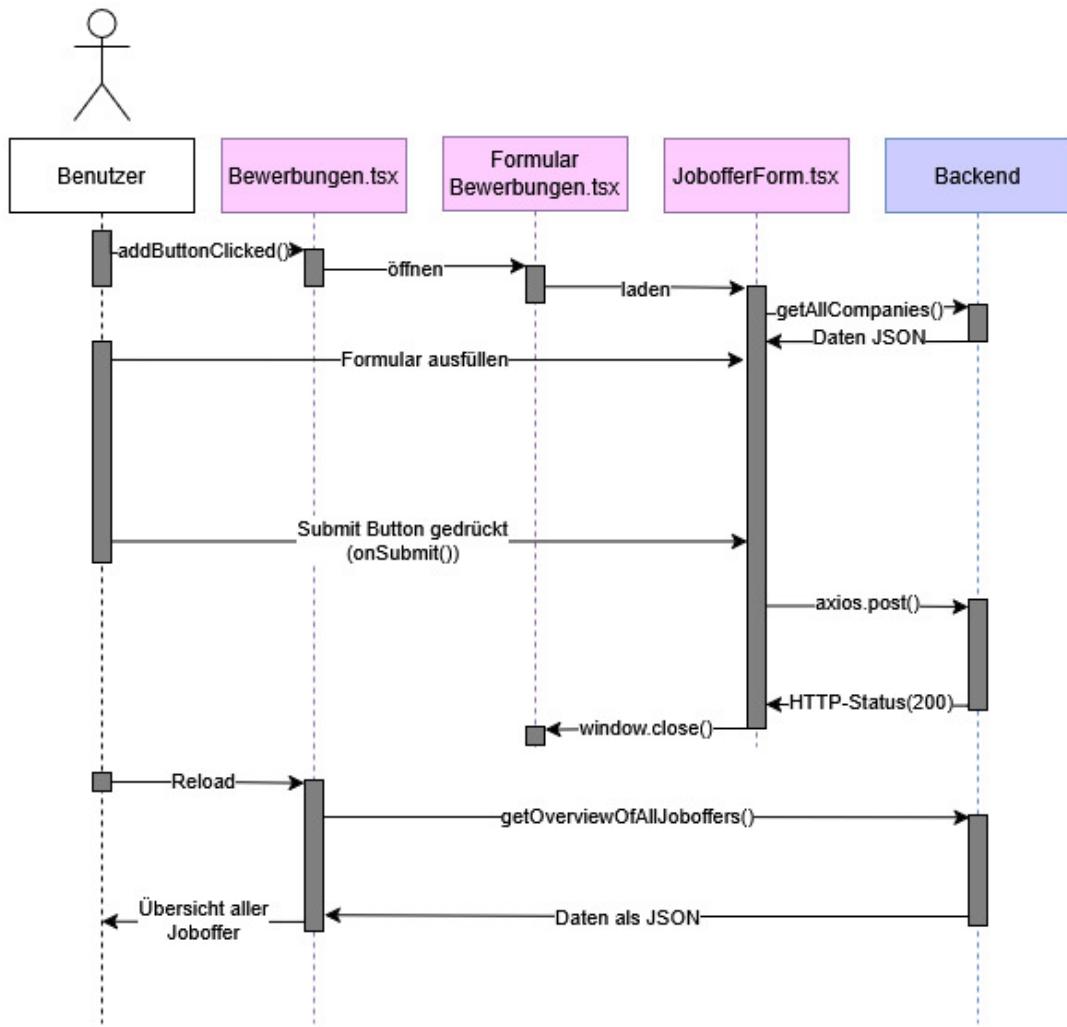


Abbildung 4.10: Eingabe der Stelle

In der Anwendungsschicht wird der Controller für die Stellen aufgerufen. Dieser ruft den Service auf, welcher sich vom App-User-Repository die Entität mit den Nutzerdaten aus der Datenbank lesen lässt. Daraufhin ruft er die Anderen Services auf um die entsprechenden Kindentitäten erstellen und speichern zu lassen. Die neuen Kindentitäten erhält der Joboffer-Service als Rückgabeparameter um diese in der Vaterentität (Joboffer) zu verknüpfen und speichert diese. Abschließend werden die eingetragenen Termine mit Verknüpfung zu der neu erstellten Stelle gespeichert. Dieser Vorgang ist in Abbildung 4.11 modelliert, ohne die detaillierte Aufruffolge innerhalb der Services der Kindentitäten darzustellen. Nach erfolgreichem Speichervorgang gibt der Controller einen HTTP-Status 200 zurück.

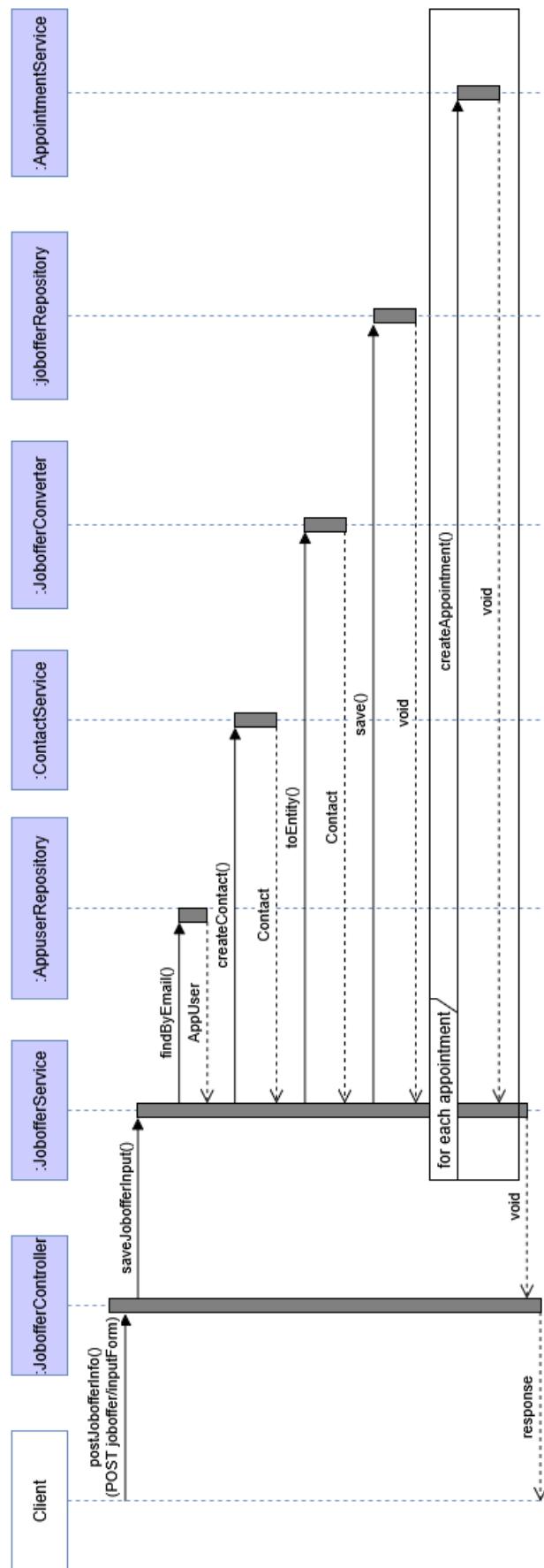


Abbildung 4.11: Speichern der Eingabe

## 4.4 Schnittstellenbeschreibung

### 4.4.1 Verwendete Schnittstellentechnologie

Für den Zugriff vom Frontend auf das Backend wird Axios (HTTP get/post) zur Datenabfrage bzw zum Senden der Daten an das Backend verwendet. Auf Seiten des Backends stellt hier Spring Web die jeweilige Schnittstelle bereit, auf die sich die get/post Anfragen beziehen. Des Weiteren wird der Zugriff auf die Daten im Backend durch Spring Security verifiziert. Daher muss beim Zugriff das Token, das beim Einloggen auf der Backend-Seite entsteht, mitgesendet werden. Das Backend greift dann mit JPA und Hibernate über JDBC auf die Postgres Datenbank zu.

### 4.4.2 Übersicht der Stellenausschreibungen

**Endpoint:** /joboffer

#### Beschreibung

Dieser Endpoint liefert eine Liste aller Stellenausschreibungen (Joboffers) im Überblick. Er wird vom Frontend verwendet, um die Übersicht aller Stellenausschreibungen anzuzeigen.

#### Request

- **Methode:** GET
- **URL (Frontend):** `http://localhost:8080/joboffer`
- **Backend-Controller:** JobofferController.java

#### Response

- **Status 200 OK:** Gibt ein JSON-Array mit allen Stellenausschreibungen zurück.

#### Beispiel-Response

```
[
  {
    "jobofferid": 1,
    "joboffername": "Ad distinctio asperiores omnis autem facilis.",
    "companyid": 1,
    "companynname": "Schäfer KG"
  },
  {
    "jobofferid": 2,
    "joboffername": "Dolores error non blanditiis id dolor.",
    "companyid": 1,
    "companynname": "Schäfer KG"
  }
]
```

## Datenmodell

Datenmodell des erhaltenen JSON Arrays	Datenmodell des verarbeiteten Response = Array von Stellenausschreibungen (Typ: JobofferOverview)	Umwandlung
jobofferid : number = ID der Stellenausschreibung	jobofferId : number = ID der Stellenausschreibung	direkte Zuweisung
joboffername : string = Titel der Stellenausschreibung	jobofferName : string = Titel der Stellenausschreibung	direkte Zuweisung
companyid : number = ID des Unternehmens	companyId : number = ID des Unternehmens	direkte Zuweisung
companyname : string = Name des Unternehmens	companyName : string (optional) = Name des Unternehmens	direkte Zuweisung
	companyImage : string (optional) = Quelle des Unternehmenslogos	setzt leeren String als default da Bilder noch nicht in der Datenbank enthalten sind
nextapptdate : string = nächster anstehender Termin als ISO-Timestamp	nextAppointment : string (optional) = nächster anstehender Termin als String	ISO-Timestamp wird in normale Datumsausgabe als String umgewandelt mit: parseDateToNextAppointmentString (joboffer.nextapptdate)

Abbildung 4.12: Datenmodell der Joboffer Overview Schnittstelle

## Axios GET: Status der Anfrage

Status	Anzeige im Frontend
OK (200)	Zeigt die Kurzübersicht der Stellenausschreibungen als Kacheln an
Loading	Textausgabe, die angibt, dass Daten noch geladen werden
4** oder 5**	Textausgabe: Fehler vom Server: <Fehlerstatus> - <Fehlerbeschreibung>
keine Antwort	Textausgabe: Keine Antwort vom Server erhalten
Fehler in der Axios GET Anfrage	Textausgabe: Fehler bei der Anfrage.

Abbildung 4.13: Status der Axios Anfrage

#### 4.4.3 Detaillierte Stellenausschreibung ansehen

**Endpoint:** /joboffer/\${id}

##### Beschreibung

Dieser Endpoint liefert alle Angaben zu einer einzelnen Stellenausschreibung (je nach im Link übergebener Joboffer-Id). Er wird vom Frontend verwendet, um eine vollständige Stellenausschreibung anzuzeigen.

##### Request

- **Methode:** GET
- **URL (Frontend):** http://localhost:8080/joboffer/\${id}
- **Backend-Controller:** JobofferController.java

##### Response

- **Status 200 OK:** Gibt ein JSON-Objekt mit allen Details zur Stellenausschreibung zurück.

##### Beispiel-Response

```
{
  "joboffer": {
    "id": 3,
    "jobtitle": "Maiores mollitia distinctio dignissimos corporis.",
    "description": "Atque consequatur eligendi sed officiis.",
    "wagemin": 90000,
    "rating": 5,
    "notes": "Vel velit eum veniam est ad magni aut dignissimos.",
    "contact": {
      "id": 2,
      "firstname": "Moritz",
      "lastname": "Erbrecht",
      "email": "moritz.erbrecht@example.net",
      "phoneno": "(05495) 63242"
    },
    "company": {
      "id": 2,
      "companynamne": "Junk AG",
      "empcount": 21,
      "logo": "/8fb17a4dfc36c020015fbc76458492bf.jpg",
      "address": {
        "id": 7,
        "street": "Skyla-Girschner-Platz",
        "streetno": "35",
        "city": "Bremen",
        "zip": "31078",
        "country": ""
      }
    }
}
```

```
},
"appointments": [
  {
    "id": 2,
    "appointmentDate": "2025-12-15T13:20:00",
    "appointmentName": "eveniet"
  },
  {
    "id": 3,
    "appointmentDate": "2025-12-20T09:05:00",
    "appointmentName": "harum"
  }
]
```

**Datenmodell**

Beschreibung	erhaltenes JSON	verarbeitetes Response	optional?	Datentyp
	joboffer.id	jobofferId	Nein	number
Stellentitel	joboffer.jobtitle	jobofferName		string
Beschreibung der Stelle	joboffer.description	jobofferDescription	Ja	string
Persönliche Bewertung der Stelle	joboffer.rating	jobofferRating		number
Minimum der Gehaltsspanne	joboffer.wagemin	jobofferMinimum Wage		number   string
Maximum der Gehaltsspanne	joboffer.wagemax	jobofferMaximum Wage		number   string
Persönliche Notizen zu der Stelle	joboffer.notes	jobofferNotes		string
	joboffer.contact.id	contactId		number
Vorname der Kontaktperson	joboffer.contact.firstname	contactFirstName		string
Nachname der Kontaktperson	joboffer.contact.lastname	contactLastName		string
E-Mail-Adresse der Kontaktperson	joboffer.contact.email	contactEmail		string
Telefonnummer der Kontaktperson	joboffer.contact.phoneno	contactPhone		string
	joboffer.company.id	companyId		string
Firmenname	joboffer.company.companynname	companyName		string
Anzahl der Mitarbeiter der Firma	joboffer.company.empcount	company Employees		string
Adresse des Firmenlogos (z.B. Link)	joboffer.company.logo	companyLogo		string
	joboffer.company.address.id	addressId		number

Abbildung 4.14: Joboffer Details API (1/2)

Beschreibung	erhaltenes JSON	verarbeitetes Response	optional?	Datentyp
Straße des Standorts der Firma/Stelle	joboffer.company .address.street	addressStreet	Ja	string
Hausnummer des Standorts	joboffer.company .address.streetno	addressStreet Number		string   number
Stadt des Standorts	joboffer.company .address.city	addressZipCode		number
Postleitzahl des Standorts	joboffer.company .address.zip	addressCity		string
Land	joboffer.company .address.country	addressCountry		string
Liste der Termine	appointments[]	appointments		Array (Typ: Appointment[])
	appointments[].id	appointmentId		number   string
Datum als ISO-Timestamp-String	appointments[] .appointmentDate	appointmentDate		string
Name des Termins	appointments[] .appointmentName	appointmentName		string

Abbildung 4.15: Joboffer Details API (2/2)

## Zusammenfassung

Die **Joboffer Details API** liefert sämtliche Informationen zu einer einzelnen Stellenausschreibung in einer einzigen strukturierten Payload. Sie wird für die Detailseite einer Stellenausschreibung verwendet und dient als Grundlage für die Anzeige aller relevanten Daten:

- Jobtitel
- Beschreibung
- Gehaltsrange
- Bewertung
- persönliche Notizen
- Kontaktdaten
- Unternehmensinformationen
- Firmenadresse
- Firmenlogo
- alle Termine (Appointments)

### 4.4.4 Stellenausschreibung hinzufügen und bearbeiten

**Endpoint:** /joboffer/inputForm/\${id}

#### Beschreibung

Dieser Endpoint nimmt alle Angaben zu einer einzelnen Stellenausschreibung (Joboffer) entgegen und speichert sie persistent in der Datenbank ab. Bei bereits vorhandener Joboffer-Id werden die Daten geändert, anstatt als neue Joboffer gespeichert. Er wird vom Frontend verwendet, um eine vollständige Stellenausschreibung hinzuzufügen oder zu bearbeiten.

#### Request

- **Methode:** POST bzw PUT
- **URL (Frontend):** `http://localhost:8080/joboffer/inputForm/${id}`
- **Backend-Controller:** JobofferController.java

#### Response

- **Status 200 OK:** Gibt an, dass alles abgespeichert werden konnte.

## Beispiel-Post

```
{  
    "addressCity": "Ortsname",  
    "addressCountry": "Land",  
    "addressId": "",  
    "addressStreet": "Straßennamen",  
    "addressStreetNumber": "0",  
    "addressZipCode": "00000",  
  
    "appointments": [  
        {  
            ...  
        }  
    ],  
  
    "companyEmployees": "0",  
    "companyId": 1,  
    "companyLogo": "",  
    "companyName": "Schäfer KG",  
  
    "contactEmail": "bsp@test.test",  
    "contactFirstName": "Vorname",  
    "contactId": "",  
    "contactLastName": "Nachname",  
    "contactPhoneNumber": "000000000000",  
  
    "distanceLength": "0",  
    "distanceTime": "0",  
  
    "jobofferDescription": "Hier kann eine Kurzbeschreibung stehen",  
    "jobofferId": "",  
    "jobofferName": "Stellentitel",  
    "jobofferRating": "",  
  
    "perks": "Hier können Perks stehen",  
    "personalNotes": "Hier können persönliche Notizen stehen",  
  
    "salaryMaximum": "0",  
    "salaryMinimum": "0"  
}
```

**Datenmodell**

<b>Stellausschreibung</b>				
Beschreibung	Feld (Frontend + Backend)	Typ	Entity	Pflicht
Bei Edit notwendig, bei Create leer	jobofferId	number   string	-	Nein
Titel / Name der Stelle	jobofferName	string	jobtitle	Ja
Persönliche Bewertung	jobofferRating	number   string	-	Nein
Beschreibung der Position	jobofferDescription	string	description	
Eigene Notizen (Frontend-only; Backend: jobofferNotes)	personalNotes	string	notes	
Mindestgehalt	salaryMinimum	number   string	wagemin	
Maximalgehalt	salaryMaximum	number   string	wagemax	
Zusatzleistungen (derzeit noch nicht im Backend verwendet)	perks	string	-	
<b>Unternehmen</b>				
Beschreibung	Feld	Typ	Entity	Pflicht
Wird derzeit ignoriert – Backend erstellt immer neue Company	companyId	number   string	-	Nein
Pflicht laut Backend	companyName	string	companynamen	Ja
Mitarbeiteranzahl	companyEmployees	number   string	empcount	Nein
Noch nicht implementiert	companyLogo	string	-	

Abbildung 4.16: Add Joboffer API (1/2)

<b>Adresse</b>				
<b>Beschreibung</b>	<b>Feld</b>	<b>Typ</b>	<b>Entity</b>	<b>Pflicht</b>
Nicht benötigt – Adressen werden neu angelegt	addressId	number   string	-	alle Felder optional Wenn alle leer -> kein Kontakt wird erstellt
Straße	addressStreet	string	street	
Hausnummer	addressStreetNumber	string	streetno	
PLZ	addressZipCode	number   string	zip	
Stadt	addressCity	string	city	
Land	addressCountry	string	country	
<b>Kontaktperson</b>				
<b>Beschreibung</b>	<b>Feld</b>	<b>Typ</b>	<b>Entity</b>	<b>Pflicht</b>
Nicht benötigt – Contacts werden neu erzeugt	contactId	number   string	-	alle Felder optional Wenn alle leer -> kein Kontakt wird erstellt
Vorname	contactFirstName	string	firstname	
Nachname	contactLastName	string	lastname	
E-Mail	contactEmail	string	email	
Telefonnummer	contactPhoneNumber	string	phoneno	

Abbildung 4.17: Add Joboffer API (2/2)

## Zusammenfassung

JobofferFormData (siehe Felder von oben) dient als Datenmodell zur Erfassung und Bearbeitung einer Stellenanzeige im Frontend.

Die Struktur beinhaltet alle relevanten Informationen zu einer Stellenausschreibung, welche als JSON an das Backend übermittelt werden, wo sie in mehrere Entitäten (Joboffer, Address, Contact, Appointment) konvertiert und persistent gespeichert werden.

Die Schnittstelle wurde so gestaltet, dass sie sowohl neue Einträge unterstützt (leere Entity-Ids) als auch bestehende Einträge lädt und zur Bearbeitung bereitstellt.

Neue Einträge sind an leeren Ids zu erkennen und werden als POST ans backend geschickt, während beim Bearbeiten bereits existierende Ids mitgegeben werden und diese als PUT ans Backend übermittelt werden.

Die Schnittstelle wird von der React-Komponente JobofferForm verwendet.

### 4.4.5 Stellenausschreibung löschen

**Endpoint:** /joboffer/delete/\${id}

#### Beschreibung

Diese Schnittstelle ermöglicht das Löschen einer Joboffer inklusive zugehöriger Entitäten basierend auf der im Link übergebenen Joboffer-Id.

**Request**

- **Methode:** DELETE
- **URL (Frontend):** `http://localhost:8080/joboffer/delete/${id}`
- **Backend-Controller:** JobofferController.java

**Response**

- **Status 204:** Gibt an, dass die Joboffer erfolgreich gelöscht wurde.

#### 4.4.6 Übersicht der Termine

**Endpoint:** /appointment

##### Beschreibung

Dieser Endpunkt liefert eine Liste aller Termine (appointments) im Überblick. Er wird im Frontend verwendet, um die Übersicht aller Termine in einer Liste und einem Kalender zu rendern.

##### Request

- **Methode:** GET
- **URL (Frontend):** `http://localhost:8080/appointment`
- **Backend-Controller:** AppointmentController.java

##### Response

- **Status 200 OK:** Gibt ein JSON-Array an Terminen zurück.

##### Beispiel-Response

```
[
  {
    appointmentID: 1
    appointmentDate: "2024-01-12T14:00:00",
    appointmentName: "Telefonat",
    jobofferid: 1
    joboffername: "Ad distinctio asperiores omnis autem facilis."
    companynname: "Schäfer KG"

  },
  {
    appointmentID: 2
    appointmentDate: "2025-01-12T14:00:00",
    appointmentName: "Bewerbungsgespräch",
    jobofferid: 2
    joboffername: "Dolores error non blanditiis id dolor."
    companynname: "Schäfer KG"

  }
]
```

## Datenmodell

Datenmodell des erhaltenen JSON Arrays		Datenmodell des verarbeiteten Response		Umwandlung
		Joboffer Overview[]:	= Array von Stellenausschreibungen (JobofferOverview)	
Array von:	appointmentID : number = ID des Termins	Joboffer Overview:	JobofferID: number =ID des Termins	direkte Zuweisung
	appointmentdate : string (ISO) = Datum des Termins		jobofferName : string = Titel der Stellenausschreibung	direkte Zuweisung
	appointmentname : string = Terminart		companyName : string (optional) = Name des Unternehmens	direkte Zuweisung
	JobofferID: number =ID des Termins		nextAppointment : string = Datum des nächsten Termins	aus Iso Timestamp formatiert
	joboffername : string = Name der Bewerbung			
	companynamne : string = Name der Firma			

Abbildung 4.18: Datenmodell der Appointment Overview Schnittstelle

## Zusammenfassung

Die **Appointment Overview API** stellt eine kompakte Auflistung aller Termine zur Verfügung. Sie wird hauptsächlich zur Darstellung einer tabellarischen Terminübersicht und eines Kalenders genutzt.

## 4.5 Entscheidungen

### 4.5.1 Verwendete Technologien / Frameworks

Alle Technologien hatten die Grundvoraussetzung, dass sie Open-Source (entsprechend der nicht-funktionalen Anforderungen) und kostenlos sind. Zudem war es und wichtig, dass alle Technologien möglichst viel Dokumentation und Community-Support haben, um sicherzustellen das wir uns in diese möglichst problemlos einarbeiten können und zudem immer eine Anlaufstelle haben, falls wir nicht weiter kommen.

### Frontend – React

Beim Frontend hatten wir zunächst drei Optionen: Angular, Vue.js und React. Bei Vue hätten wir keine Unterstützung durch unseren Betreuer bei Schwierigkeiten erhalten können,

weshalb wir uns dagegen entschieden haben. Bei Angular hätte uns zwar unser Betreuer helfen können, aber laut unserer Recherchen hat dieses eine relativ steile Lernkurve, weshalb wir uns letztendlich für React entschieden haben. Dieses hat außerdem den zusätzlichen Vorteil, dass es sehr schnell lädt, da es eine virtuelle DOM verwendet um die Ansicht des Users nur dort zu verändern wo tatsächlich Änderungen vorliegen.

Innerhalb des React Projekts wurden diese Frameworks und Tools verwendet:

- Vite wurde zum Erstellen des React Projekts verwendet. Es ermöglicht eine Art „SSchnellstart“, indem nicht alles manuell konfiguriert werden muss.
- React Router Dom wurde verwendet, um verschiedene Seitenansichten zu ermöglichen und zwischen diesen Seiten zu navigieren.
- Material UI Komponenten wurden verwendet, um die Erstellung der UI Komponenten zu vereinfachen, indem man die von MUI bereitgestellten Komponenten als Template verwenden kann. Dieses wird dann an den jeweiligen Verwendungszweck angepasst implementiert. Um MUI nutzen zu können, sind außerdem @emotion/react @emotion/styled @fontsource/roboto erforderlich.
- Außerdem wurde date-fns zum Arbeiten von Datumsformaten verwendet.
- Des Weiteren wird Axios genutzt, um Daten über HTTP an das Backend zu senden bzw. vom Backend zu erhalten.
- react-hook-form kommt im Eingabeformular zum Einsatz um Boilerplate-Code zu sparen und eine Performante Eingabe zu ermöglichen.
- Zod wird auch im Eingabeformular zur Validierung verwendet.

Die Installation und Verwaltung dieser Nodejs Packages erfolgt über npm (NodePacket-Manager) und nvm (NodeVersionManager).

### **Backend - Java / Springboot / Maven**

Im Backend fällt die Entscheidung für die verwendeten Technologien auf Springboot mit Java als Programmiersprache und Maven als Buildtool. Die Entscheidung für Java als Programmiersprache ist leicht, da alle im Team die Grundlagen im vorherigen Semester erworben haben. Für das Framework fällt die Entscheidung auf Springboot um von der Erfahrung unseres Betreuers zu profitieren. Ein weiterer Grund ist auch, dass Quarkus ein vergleichsweise neues Framework mit einem wachsenden Ökosystem ist, was die integrierbaren Technologien einschränkt. Springboot hingegen ist schon etabliert und bietet eine Vielzahl an verfügbaren Informationen und Tutorials im Netz für den Einstieg und verschiedene Problemstellungen. Maven als Buildtool wird dann schlussendlich von Springboot als gängiges Buildtool im Java-Umfeld vorgegeben.

### **Datenbankmanagementsystem – PostgreSQL**

Grundsätzlich war klar, dass unsere Datenbank basierend auf unseren Daten eine relationale Datenbank sein sollte. Die Wahl für PostgreSQL als Datenbank war also schnell gefallen, da wir damit ebenfalls in einem unsrer Module arbeiten. PostgreSQL ist außerdem strikt ACID-Konform, d.h. im Gegensatz zu manchen anderen DBMS ist Postgres mit allen verfügbaren Konfigurationen immer noch ACID-Konform und ist allgemein extrem zuverlässig bei Transaktionen. Postgres unterstützt zusätzlich auch komplexere SQL-Abfragen, was uns bei der Entwicklung mehr Flexibilität ermöglicht.

## **Identitäts- und Zugriffsmanagement – Keycloak**

Keycloak als IAM war uns im allgemeinen bereits vorgeschrieben, da es für uns die Login-Logik vollständig übernimmt und dabei auf Standard-Protokollen basiert ist es eine ideale Möglichkeit die Login-Logik sicher zu implementieren. Zusätzlich bietet Keycloak auch sehr viel Flexibilität und Anpassbarkeit. Uns wurde die Möglichkeit gegeben ein anderes IAM mit guter Begründung zu wählen, falls es eins gebe das für unser Projekt besser wäre. Wir haben hierzu aber keins gefunden.

### **4.5.2 Struktur**

#### **Backend Paketstruktur**

Die starke Kohäsion und geringe Kopplung der nach technischen Schichten sortierten Pakete sorgt für einen erhöhten Zeitaufwand beim Entwickeln der Features. Um die Kohäsion zu reduzieren wird die Paketstruktur in eine nach Feature sortierte Struktur umgearbeitet.

#### **Entfernen der Entität Company**

Die separate Company-Entität sorgt für zusätzliche und nicht benötigte Komplexität bei der Entwicklung. Da für die Firmen keine separate Verwaltung geplant ist und der Aufwand einer Umstrukturierung nach Einschätzung weniger aufwändig ist wird die Company-Entität in Rücksprache mit Kunde und Betreuer aufgelöst. Die Funktionalitäten und Attribute werden in die Joboffer Entität übertragen.

Um den Aufwand weiter einzuschränken werden die SQL-Skripte für das Schema und die Testdaten entfernt. Stattdessen wird das Schema durch JPA/Hibernate erstellt.

# 5 Teamorganisation

## 5.1 Projektmanagement

Die Softwareentwicklung verläuft in Anlehnung an Scrum. Es findet ein wöchentliches Meeting statt, das den Beginn des nächsten Sprints markiert. Projektaufgaben werden in einem Kanban-Board zuerst auf GitHub, später auf Jira verwaltet. Zur Versionsverwaltung wird ein privates Repository auf GitHub in einer für das Projekt erstellten Organisation verwendet.

### 5.1.1 Meetings

Das regelmäßige Meeting findet jeden Freitag um 14:00 Uhr über Microsoft Teams statt. Darin präsentiert das Team seine Fortschritte aus dem letzten Sprint und legt zusammen mit dem Kunden die Aufgaben für den nächsten Sprint fest. Zusätzlich bietet das Meeting die Möglichkeit, Anforderungen zu spezifizieren und Rückfragen zu stellen. Der Leiter des Meetings wird vom Team gestellt und rotiert wöchentlich. Ein Protokoll wird von min. 2 Personen gemeinsam in einem Hedge-Doc-Dokument erstellt. Diese Personen rotieren mit dem Gesprächsleiter wöchentlich. Im Anschluss bespricht sich das Team in einem halbstündigen Meeting. Bei weiterem Gesprächsbedarf steht dem Team ein weiteres 30-minütiges Zeitfenster für ein zusätzliches Meeting zur Verfügung.

### 5.1.2 Kommunikation

Für die Kommunikation ist ein Discord-Server eingerichtet. Darin befinden sich eine Reihe von Textkanälen und ein allgemeiner Sprachkanal. Davon sind jeweils ein Textkanal für Dokumente und Protokolle reserviert. Des Weiteren stehen Textkanäle für Fragestellungen zur Verfügung. Für die Teamkommunikation stehen gesonderte Textkanäle bereit, die nur vom Team eingesehen werden können. Einer dient dabei für tägliche Fortschrittsmeldungen der Teammitglieder, um eine Art Daily-Standup zu ermöglichen.

## 5.2 Aufwandsschätzung

Für eine initiale Aufwandsschätzung wurden die Meilensteine als feste Werte in das Gantt-Diagramm, Abbildung 5.1, eingetragen und die Anforderungen der einzelnen Meilensteine als Aufgaben vermerkt und abgeschätzt. Dabei wurden initial die Wochenenden mitgezählt, jedoch von der Software nicht akzeptiert, daher sind manche Aufgaben aufgrund des Wochenendes überdimensioniert. Technische Aufgaben, welche für die Funktionalität des Projekts abgeschlossen werden müssen sind nicht eingetragen. Da es noch kein technisches Konzept und damit keine genaue Aufgaben gibt, sind diese nicht aufgeführt. In die Setup-Schätzungen wurden verschiedene Zeiten der Verfügbarkeit und wahrscheinliche technische Probleme eingerechnet.

## 5 Teamorganisation

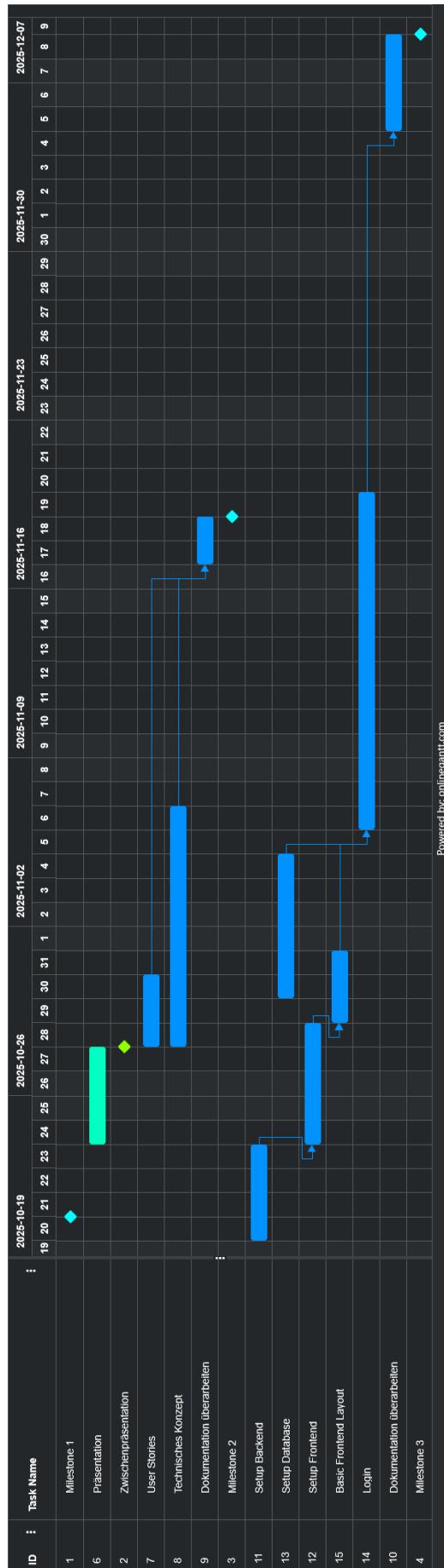


Abbildung 5.1: Aufwandsschätzung Gantt

## 5 Teamorganisation

<b>Task</b>	<b>Zeitschätzung</b>	<b>Zeiterfassung</b>	<b>Abweichung</b>
<b>Grundlagen</b>			
Grundlagen (gesamt)	1w 2d 2h 45m	1w 1d 6h	-4h 45m
Datenübertragung DB Spring Boot Axios	1w	3d 6h 30m	-1d 1h 30m
SQL-Skripte	3h 45m	5h 30m	+1h 45m
Spring Boot Projekt aufsetzen	1h	1h	
React Projekt aufsetzen	1d	4h	-4h
Postgres Projekt aufsetzen	1h	1h 30m	+30m
Docker erstellen und überarbeiten	5h	1d 3h 30m	+6h 30m
<b>Authentifizierung</b>			
Authentifizierung	3w	1w 1h 30m	-1w 4d 6h 30m
<b>Funktionalitäten Bewerbungen</b>			
Funktionalitäten Bewerbungen (gesamt)	2w 2d 6h	3w 1h 55m	+2d 3h 55m
Übersicht Bewerbungen	2d 3h	3d 7h 45m	+1d 4h 45m
Details Bewerbungen ansehen	2d	1d 6h	-2h
Formular Hinzufügen / Bearbeiten	1d	4d 35m	+3d 35m
Umbau Input Formular	1w 7h	4d 5h 20m	-1d 1h 40m
Bearbeiten Bewerbungen (Backend)	4h	7h	+3h
Bewerbungen löschen	1d	7h 15m	-45m
<b>Funktionalität Termine</b>			
Funktionalität Termine (gesamt)	1w 2h 45m	1w 1d 10m	+5h 25m
Terminübersicht	2d 7h	4d	+1d 1h
Termin hinzufügen	1d 6h 45m	1d 1h 10m	-5h 35m
Termin löschen	5h	7h	+2h
Termin bearbeiten	3h	30m	
<b>Allgemeine UI</b>			
Allgemeine UI (gesamt)	2d 5h	1w 4h 30m	+2d 7h 30m
Dark- & Light Mode	1d	3d	+2d
Einstellungen	5h	3h	-2h
UI-Navigationsleiste	1d	2d 1h 30m	+1d 1h 30m
<b>Clean Code</b>			
Clean Code (gesamt)	2d 1h	2d 5h	+4h
Axios Aufrufe modularisieren	1d	6h 30m	-1h 30m
Backend neu strukturieren	1d 1h	1d 6h 30m	+5h 30m
<b>Organisatorisches</b>			
Jira Kanban Board		1d 1h 30m	
<b>Dokumentation</b>			
Dokumentation (gesamt)	1w 3d 2h 45m	2w 2d 3h 30m	+4d 45m
Meilenstein 1	2d 3h	2d 2h 45m	-15m
Meilenstein 2	1d	7h 15m	+5h 15m
Meilenstein 3	1d	3d 30m	+2d 30m
Meilenstein X	2d 4h	2d 4h 30m	+30m
Zwischenpräsentation	3h 45m	3d 3h 30m	+3d
Abschlusspräsentation	1d	1d	

Zeitangaben aus Jira: 1w (Woche) = 5d (Tage) und 1d (Tag) = 8h (Stunden)

## 5 Teamorganisation

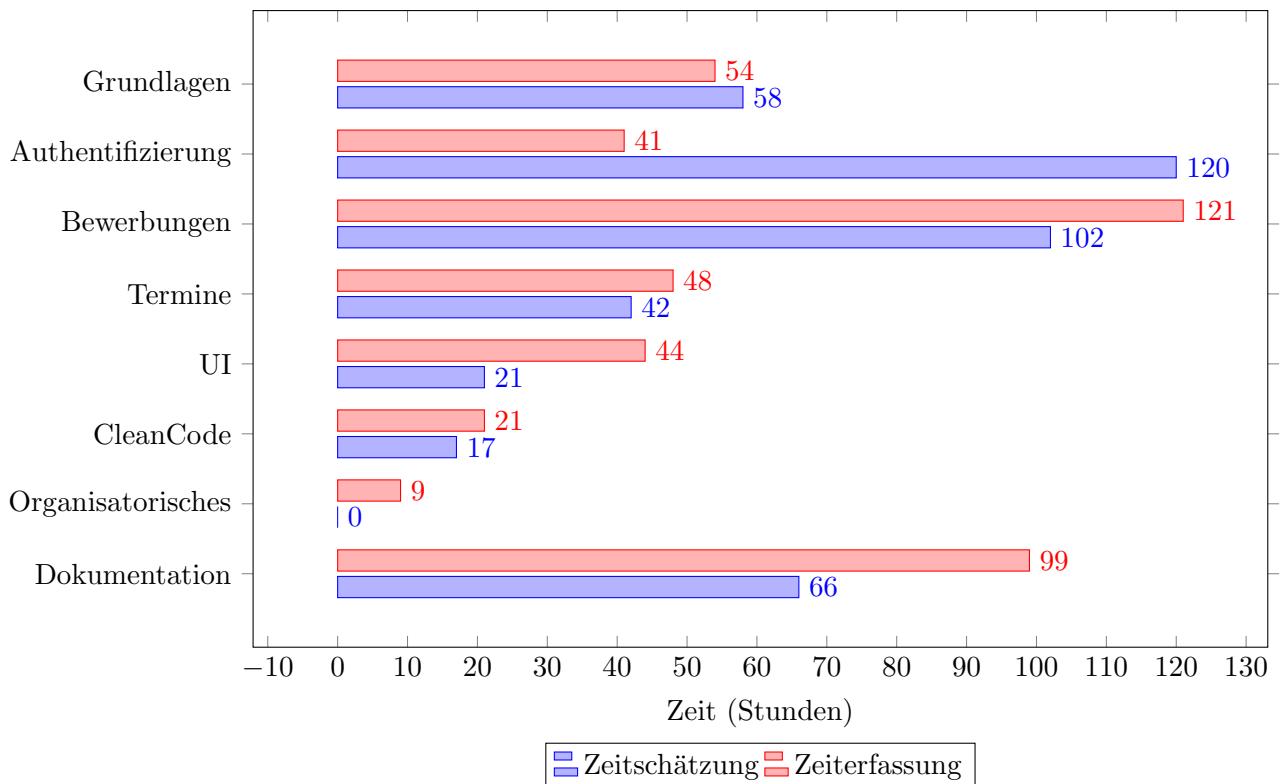


Abbildung 5.2: Vergleich von Zeitschätzung und tatsächlicher Zeiterfassung  
Stand 21.01.2026

## 5.3 Aufgabenteilung

### 5.3.1 Lukas

- Aufsetzen von Docker-Compose mit für Springboot-Backend und Postgres-Datenbank
- REST-Schnittstellen für: Firmenliste, Stellenübersicht, Terminübersicht, Stellenbearbeitung
- ButtonGroup Komponente (Frontend)
- Refactoring Backend nach feature-basierten Packages
- Überarbeitung der Eingabe-Schnittstelle im Zuge des Refactorings
- Einbau von Zod-Validierung und react-hook-form in das Eingabeformular der Stellen.
- Umstrukturierung Datenbank und Backend: Entfernen der separaten Firmenentität
- Dokumentation
- Codeanalyse vom Backend mit SonarQube

### 5.3.2 Norman

- Dokumentation

- Entwicklung eines Kalenders ohne Material UI
- überarbeiten der App Bar und Drawer für Einstellungen
- Einheitliches Styling alle Seiten
- Termine Seite
- Implementierung der Löschfunktion für Termine inklusive notwendiger Backend-Funktion
- Eigenen Kalender durch Material UI ersetzen
- umstellbaren Darkmode (verworfen)
- Eingabefeld für Termine
- Layout für Zeit und Datum auf Systemeinstellungen setzen

### 5.3.3 Marcus

- Einfügen von Keycloak in Docker-Compose
- Einrichten von Realms und Clients
- Backendanbindung von Keycloak, Token-checks, Nutzererstellung
- Grafische Anpassung der Login- und Registrierungsseite
- Dokumentation
- Protokolle (Überarbeitung und Übertragung in Dokumentation)

### 5.3.4 Lara

- Erstellung von Datenbank Schemas und Testdaten
- REST-Schnittstellen für: Stellen-Details-Ansicht, Eingabe
- Dokumentation

### 5.3.5 Jannika

- Erstellung der GitHub-Organisation
- Erstellung eines initialen Kanban-Boards auf GitHub inklusive Erstellung erster Tasks
- Erstellung und iterative Überarbeitung von Wireframes mit Figma
- Aufsetzen des React-Projekts inklusive grundlegender Projektstruktur und Routing
- Entwicklung einer Navigationsleiste ohne Material UI
- Ersetzen der Navigationsleiste durch die Material-UI App Bar und Anpassung an das Projekt
- Erstellen einer Übersichtsseite für Stellenanzeigen in Kachelansicht

- Implementierung einer Detailansicht für einzelne Stellenanzeigen
- Entwicklung von Formularen zur Erstellung und Bearbeitung von Stellenanzeigen
- Modularisierung der Axios-GET-Anfragen im Kontext der Stellenanzeigen
- Implementierung der Löschfunktion für Stellenanzeigen inklusive notwendiger Backend-Funktionalitäten
- Umsetzung eines rudimentären Dark-Modes
- Dokumentation

## 5.4 Verwendete Hilfsmittel

Während der Entwicklung wurden die KI-Modelle ChatGPT 4.0-5.2 und Claude Sonnet verwendet als Tutor für unbekannte Technologien, Sparring Partner für Struktur und Lösungsansätze, Unterstützung bei der Fehlersuche und bei der Analyse von Fehlermeldungen, Generierung von CSS stylings (Keycloak) und vereinzelt zur Generierung von kleinen Code-Snippets.

Des Weiteren wurde SonarQube zur Qualitätssicherung verwendet um eine statische Codeanalyse durchzuführen.

# **6 Retrospektive**

## **6.1 Schwierigkeiten**

### **6.1.1 Fehlende Vorerfahrung**

Durch fehlende Vorerfahrung mit Technologien, Projektstruktur und Organisation wurde das Ergebnis maßgeblich beeinflusst. Es konnten nicht alle verlangten Features in der gegebenen Zeit fertig gestellt werden. Auch das implementieren zusätzlicher Features und die nicht zwingend verlangte Auseinandersetzung und einer Implementierung von Tests konnte daher nicht priorisiert werden.

### **6.1.2 Ausfälle durch Krankheit**

Durch häufige und auch länger anhaltende Ausfälle von Teammitglieder verzögerte sich die Entwicklung erheblich.

## **6.2 Entwicklung**

Die Datenbank einer Software hat großen Einfluss auf die Art und Weise wie diese und ihre Logik entwickelt werden muss. Es stellte sich daher als Fehler heraus, dass Datenbanken Schema vor der Software zu erstellen und diese dann um die Datenbank drum herum zu entwickeln. Das Ergebnis war eine erhöhte Komplexität in der Entwicklung die schließlich das entfernen des Datenbankenschemas zur Folge hatte.

## **6.3 Teamorganisation**

### **6.3.1 Kommunikation**

Der Daily-Scrum Text Kanal hat sich als wichtiger Teil unserer Organisation erwiesen. Er ermöglichte es uns Fortschritte an Tasks miteinander zu teilen, ohne das eine Terminfindung stattfinden musste. Da aber auf Grund des Umfangs im Studium nicht jeder täglich am Projekt arbeiten konnte, blieben Meldungen manchmal aus. Um trotzdem einen kontinuierlichen Austausch sicherzustellen, sollte auch ohne an dem Tag am Projekt gearbeitet zu haben eine Mitteilung gemacht werden. Dies signalisiert auch, dass sich über den Stand des Teams informiert wurde.

### **6.3.2 Board**

Das Kanban-Board für die Organisation der Tasks stellte uns vor eine Herausforderung. Um Sub-Tasks mit ihren Features zu verknüpfen um eine visuelle Zuordnung zu haben sind wir zu einem Jiraboard übergegangen. Eine Kombination aus unterschiedlichen Herangehensweisen an die Granularität und Bezeichnungen von Sub-Task und Features sowie die Menge an Features die Jira zur Verfügung stellt sorgten zunehmend für Verwirrung und einen unnötig hohen Zeitaufwand bei der Organisation der Tasks. Die Folge war, dass das Board nicht mehr regelmäßig gepflegt wurde. Für die Zukunft sollte im Team ein feste

## *6 Retrospektive*

Regelung über Granularität und Struktur für Features und Subtasks erstellt werden. Die Verwendung sollte nicht unnötig mit der Anwendung von unnötigen Features erschwert werden.

# 7 Installationshandbuch

## 7.1 Einleitung

Dieses Installationshandbuch beschreibt die Installation und Inbetriebnahme des Bewerhungstrackers für den lokalen Entwicklungs- und Testbetrieb. Die Anleitung richtet sich an Entwickler. In dieser Anleitung werden alle nötigen Schritte beschrieben, die für die lokale Ausführung benötigt werden.

## 7.2 Voraussetzungen

Die Installation und Inbetriebnahme der Software wurde unter Linux (Mint 22.2) getestet.

### Benötigte Technologien

Für die Installation und Inbetriebnahme werden folgende Technologien benötigt:

- Docker
- Java (Version 21) und JDK
- Maven
- Node.js und npm
- Webbrowser

### Verwendete Ports

Folgende lokale Ports werden von dem Prototypen verwendet:

- Datenbank: 5433
- Backend: 8080
- Authentifizierungssystem: 7080
- Frontend: 5173

## 7.3 Installation

### Systemkonfiguration

Für die Inbetriebnahme des Prototyps wird eine zusätzlich lokal konfigurierte Domain für die auf die Loopback-Adresse benötigt. Dazu ist unter Linux die Datei `/etc/hosts` anzupassen. Der bestehende Eintrag für die Loopback-Adresse ist um den Hostnamen `authentication` zu erweitern:

```
127.0.0.1 localhost authentication
```

## **Backend bauen**

Das Backend wird mit Maven gebaut. Dazu ist in das Backend-Verzeichnis zu wechseln und folgender Befehl auszuführen:

```
SPRING_DATASOURCE_PASSWORD=pass  
SPRING_DATASOURCE_URL=jdbc:postgresql://localhost:5433/testdb  
SPRING_JPA_HIBERNATE_DDL_AUTO=update  
SPRING_DATASOURCE_USERNAME=testU  
mvn clean install -DskipTests
```

## **Datenbank, Backend und Keycloak in Betrieb nehmen**

Die Komponenten Datenbank, Backend und Keycloak werden mit Docker-Compose gestartet. Dafür muss im Projektverzeichnis folgender Befehl ausgeführt werden.

```
docker compose up
```

## **Frontend**

Um die Packages und Dependencies von npm zu installieren muss in den Ordner `.../frontend` des Projektverzeichnisses navigiert werden. Dort wird folgender Befehl ausgeführt:

```
npm install
```

Um das Frontend zu starten muss im selben Verzeichnis folgende Befehl aufgerufen werden:

```
npm run dev
```

Daraufhin kann das Frontend im Browser über die URL `http://localhost:5173` erreicht werden.

# 8 Ausblick

## 8.1 Weitere Funktionalitäten

### Dokumentenverwaltung

Der Benutzer soll die Dokumente (z.B. Lebenslauf, Motivationsschreiben, Emails) die zu einer Stellenanzeige hinzugehören in die App hochladen und verwalten können.

### Unternehmenslogo

Der Benutzer sollte ein Logo des Unternehmens bei dem man sich bewirbt zur Stellenanzeige hinzufügen können. Der Frontend Code dafür existiert bereits und arbeitet mit einem Icon das angezeigt wird, wenn kein Bild-URL mitgegeben wurde. Die Implementierung in der Datenbank und im Backend fehlt jedoch noch.

### Entscheidungsmatrix

Der Benutzer sollte die Stellenanzeigen nach verschiedenen Kriterien sortieren und Filtern können oder idealerweise sich sogar ein ausgewertetes Ranking basierend auf eigenen Kriterien und Schwerpunkten bekommen.

### KI-Integration für Basisinformationen

Um dem Benutzer das Hinzufügen neuer Stellenanzeigen möglichst komfortabel zu gestalten, wäre es sinnvoll die Eingaben mithilfe einer KI vervollständigen zu lassen.

### Caching einbauen

Um die Ladezeit der Seiten zu reduzieren, würde sich das Hinzufügen von Caching im Frontend lohnen.

### E-Mail Client mit Filtern

Um alle Informationen in einem System zu vereinen, kann der Benutzer sein E-Mail Konto mit der App verbinden. Hierbei werden nur E-Mails angezeigt, die entsprechenden Filtern entsprechen und bspw. Bewerbung oder Firmennamen aus den eingegebenen Stellen beinhalten.

### Benachrichtigungen für Termine

Damit der Benutzer nicht regelmäßig die App öffnen und überprüfen muss, wäre eine Benachrichtigung über anstehende Termine (entweder als Push-Benachrichtigung, oder als Email/SMS/...) sinnvoll.

### Mobile Version

Eine mobile Ansicht wäre sinnvoll um die App besser und einfacher zugänglich zu machen. Die meisten Seiten sind bereits auf komplett Flexible Seitenformate ausgerichtet, aber sonst nicht weiter für Mobile Ansicht optimiert.

# 9 Appendix

## 9.1 Erstes Datenbankkonzept

Zu Beginn des Projekts wurde ein Datenbankkonzept erstellt. Das zugehörige Entity-Relationship-Modell ist in Abbildung 9.1 zu sehen. Dieses wurde mit einem SQL-Schema umgesetzt und in das Projekt integriert. Auf Grund von erhöhter Komplexität in der Entwicklung von Features wurde dieses Datenbankenkonzept verworfen und durch ein an die Feature angepasstes Schema ersetzt.

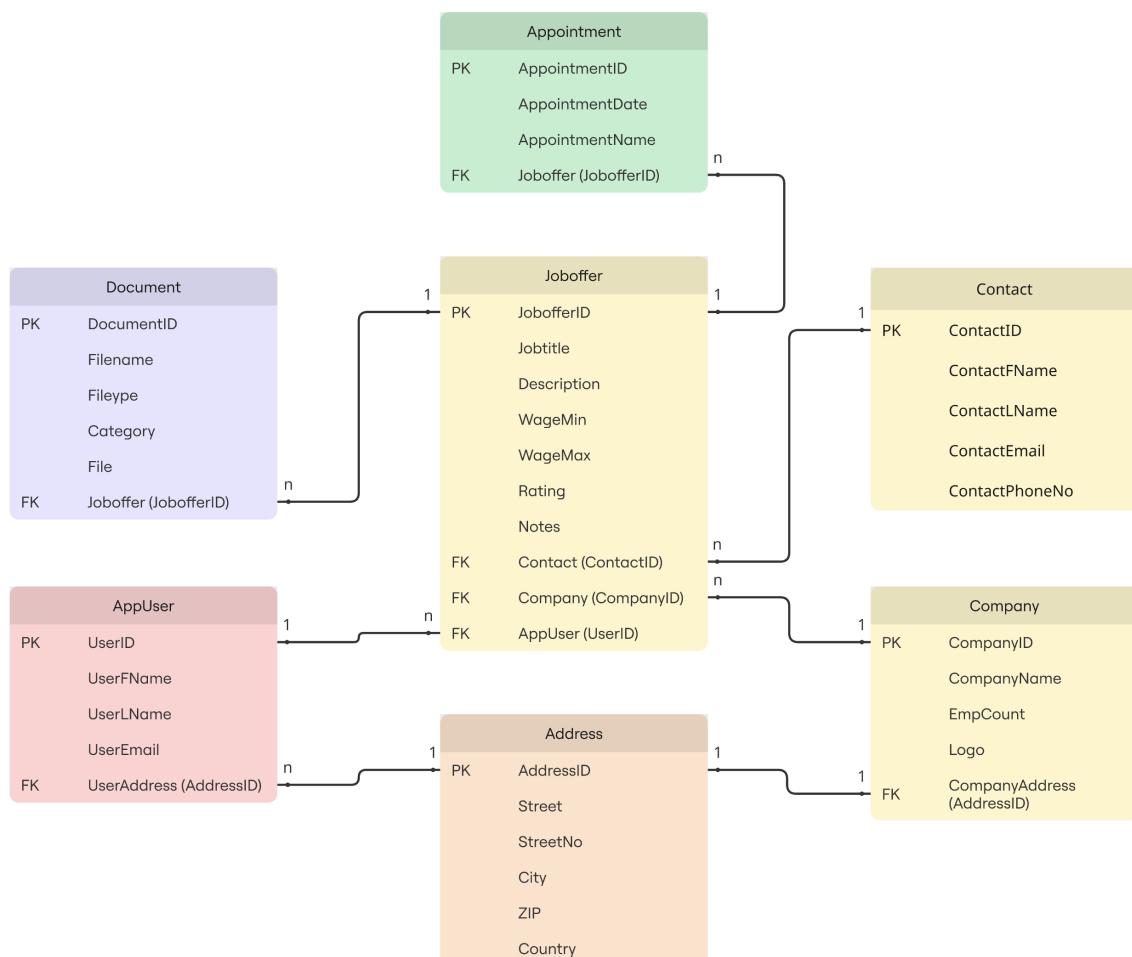


Abbildung 9.1: Erstes ER-Modell

## 9.2 Linkssammlung

Github-Repo <https://github.com/SWB-WS25-Bewerbungstracker/bewerbungstracker>

Verwendete Technologien/Dokus:

- React <https://react.dev/>

## *9 Appendix*

- Material UI <https://mui.com/material-ui/>
- Axios <https://axios-http.com/docs/intro>
- React Hook Form <https://react-hook-form.com/>
- Zod <https://zod.dev/>
- Spring Boot <https://spring.io/>
- Docker <https://docs.docker.com/>
- Keycloak <https://www.keycloak.org/>
- Jira <https://www.atlassian.com/software/jira>
- SonarQube <https://www.sonarsource.com/products/sonarqube/>

### **9.3 Protokoll 07. Oktober 2025**

# Meeting-Protokoll

---

Projekt: Bewerbungstracker

7. Oktober 2025

**Teilnehmer:** Alexander Brendel, Kunde  
Klemens Morbe, Betreuer  
Marcus Klein, Entwickler  
Jannika Nathalie Boerner, Entwicklerin  
Lukas Reinhardt, Entwickler  
Norman Muenzer, Entwickler  
Lara Seline Hippenstiel, Entwicklerin

## 1. Kontakt & Kommunikation

- Klemens: Ansprechpartner für Organisation, Dokumentation und sonstige Unterstützung
- Alex: Ansprechpartner für technische Anforderungen
- Kommunikation: Discord-Server zum Austausch
- Wöchentliche Team-Meetings: Freitag, 14:00 Uhr via [MS Teams](#)

## 2. Anforderungen & Mock-Ups

- Beispiel Mock-Up von jeweils Alex und Team (siehe Anhang)
- Must-Haves:
  - Ansicht mit Jobtitel, Firmenname, Kontakt, Gehalt, Benefits, Rating, Notizen
  - Funktion zum Sortieren & Filtern nach z. B. Termin, Gehalt, Bewerbungsdatum, Unternehmensgröße, Entfernung, etc.
  - Desktop-First Design mit Toolbar oben
  - Settings mit Basisinformationen über den Nutzer
- Nice-To-Haves:
  - Option für Dark/Light Mode und Mehrsprachigkeit
  - Dokumente hochladen und ggf. Bearbeiten
  - Entscheidungsmatrix (wird noch besprochen)
  - KI die automatisch Details zu einer Bewerbung erfasst

### **3. Technische Rahmenbedingungen**

- Frontend: React
- Backend: SpringBoot
- Datenbank: Postgre
- Authentifizierung: Keycloak, ggf. anderes falls einfacher/logischer
- Deployment: Nur lokaler Betrieb

### **4. Projektorganisation**

- Feature-based Branching mit Pull Requests und Code Reviews auf Git
- Aufgaben in kleine, klar definierte Tickets aufteilen (Tool wählen)
  - Jedes Ticket sollte klare Erfüllungskriterien besitzen
  - Priorität überlegen und setzen, wird von Alex ggf. angepasst
  - Dauer einschätzen und eintragen, später tatsächliche Zeit notieren
- Dokumentation laufend pflegen, mindestens grob

### **5. Ziele bis zur nächsten Woche**

- Projekt auf Git aufsetzen und alle Teammitglieder einladen
- Erste Tickets erstellen
- Projektmanagementmethode festlegen
- Nächstes Meeting vorbereiten und Rollen festlegen
- Erste Gedanken zur Zwischenpräsentation
- Discord-Server nützlich strukturieren

### **Sonstige Notizen:**

- Wenn etwas noch nicht definiert ist vorerst nur statische Werte einsetzen
- Bewertungskriterien des Projekts sollten im Blick behalten werden
- Vorzugsweise in Kleingruppen oder allein arbeiten, sofern möglich
- Gesprächsleitung und Protokollant zwischen Meetings durchwechseln

# Anhang

**Neue Bewerbung hinzufügen**

**Unternehmen \***  **Position \***

**Unternehmens-Logo URL**

**Kontaktperson \***

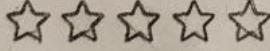
**Beschreibung des Unternehmens**

**Gehaltsspielraum**  **Mitarbeiteranzahl**

**Standort**

**Homeoffice möglich**

**Perks**

**Persönliches Rating** 

**Status**

**Bewerbungsdatum**

**Persönliche Notizen**

**Abbrechen** **Hinzufügen** 

# Bewerbungstracker

+ Neue Bewe-

Verwalten Sie Ihre Demo-Bewerbungen übersichtlich und strukturiert

① Dies sind Demo-Daten. Ihre eigenen Bewerbungen werden lokal in Ihrem Browser gespeichert

1  
Beworben

1  
Interview

1  
Angebot

0  
Angenommen

0  
Abgelehnt

Q < nach Unternehmen, Position oder Kontaktperson >

Alle Status ▾

**I** Frontend Developer  
TechCorp GmbH  
€ 55.000 - 70.000 € **Beworben**  
€ 55.000 - 60.000 €  
**Gemüsekorb** **Firmenhandy**  
**Flexible Arbeitszeiten**

Notizen: interessante Position mit modernem Tech-Stack. Das Team motiviert.

**III** Full Stack Developer  
StartupXYZ **Beworben**  
Thomas Müller  
€ 45.000 - 60.000 € **Nürnberg**  
**Kicker** **Kostenlos Getränke**  
Rating: ★★★★

Notizen: Interessante Startup-Umfeld, aber möglicherweise hoher Stress.

**II** Senior Software Engineer  
Enterprise Solutions AG  
Dr. Andreas Weker **@ Berlin**  
€ 70.000 - 85.000 € **Frankfurt**  
**Familienwagen** **10 Tage Urlaub**  
Rating: ★★★★

Notizen: Schr professionalität aufgestellt, gute Work-Life-Balance erwartet.

## Jobtracker

**JobTracker**

**Kalender**

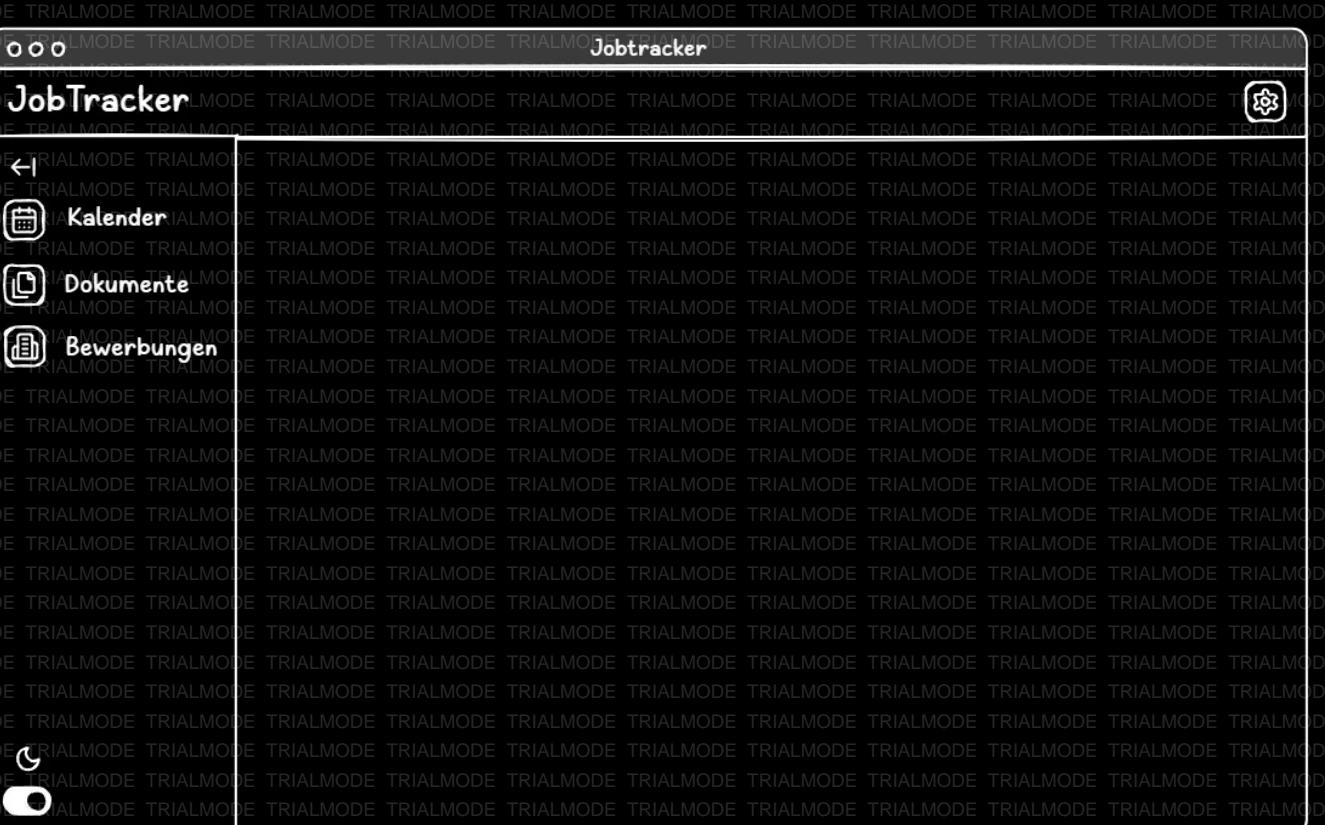
**Dokumente**

**Bewerbungen**



Jobtracker	Kalender	Dokumente	Bewerbungen	
 pep.digital	2025/10/08	 pep.digital	2025/10/09	 pep.digital
 Porsche	2025/10/14	* Daimler	2025/10/16	HE HS Es.
 Festo	2025/10/22	 Bosch	2025/10/25	 A.f.Arbeit
				2025/10/20





**Jobtracker**

**JobTracker**



**Kalender**

**Montag Dienstag Mittwoch Donnerstag Freitag**

**Daimler**

**pep.digital**

**Festo**

**HS Esslingen**

**Bosch**

**Bewerbungen**

**Dokumente**

**Jobtracker**

**Dokumente**

**Kalender**

**Bewerbungen**

Select	Name	Last Modified
<input type="checkbox"/>	Lebenslauf	2025/10/06
<input type="checkbox"/>	Anschreiben pep.digital	2025/10/03
<input type="checkbox"/>	Vertrag pep.digital	2025/10/06
<input type="checkbox"/>	Absage Daimler	2025/10/07

**Upload**



## 9.4 Protokoll 17. Oktober 2025

Teilnehmer:

- Klemens Morbe
- Marcus Klein
- Jannika Börner
- Lukas Reinhardt
- Norman Münzer
- Lara Seline Hippenstiel

### 9.4.1 Kundengespräch

#### Techstack

- React
- Spring-Boot
- PostGreSQL
- Keycloak

### 9.4.2 Betreuergespräch

#### Tickets

Tickets können groß oder klein sein, Mit Pitch (Foliensatz) anfangen, Meilensteine als Abgabe. Alles was gemacht werden muss ist ein Ticket (Daily oder Weeklys sind keine Tickets)

**Erklärung an Scrum** Tickets sind Product Backlog, Freitags sind Sprint Planning, bekommen Prioritäten der Tickets, in Sprint Backlog Tickets für die folgende Woche nehmen Arbeiten alleine/zweit/mehr um die Tickets zu erfüllen, täglich absprechen. Am Ende der Woche (Freitags) im Sprint Review präsentieren was gemacht wurde. Sprint Retrospective erstmal nicht machen, weil Zeitmangel nicht unbedingt Täglich, aber regelmäßig treffen

Aufgaben aus letztem Meeting bearbeiten

#### Github

- Tickets/Backlog angeschaut
- Morbe und Brendel zu Github eingeladen

## Sprintplan

- Github struktur überlegen (alles in einem Repo/Backend, Frontend in separaten Repos)
- empfehlung alles in einem Repo
- Starter erstellen
  - React nach Starter schauen
  - Spring in Gespräch gemacht (spring initializr)
    - \* Spring Web
    - \* Spring Data JPA
    - \* Spring Security
    - \* PostgreSQL Driver
    - \* Lombok
- Docker (Docker Composed) einführungen
  - “Docker Composed” mehrere Container
  - läuft nur auf Linux
  - auf Windows WSL in MSStore
  - Zeilenumbrüche zw. Linux und Windows verschieden in IntelliJ umstellen
  - Docker Hub Postgres
  - Docker Compose
    - \* brauchen kein memory, volume
- Tickets überarbeiten
- Empfehlung IntelliJ (Ultimate) für Spring-Boot
  - bessere Time-to-market
- Dokumentation in Latex via Overleaf (fachschaften.org) aufsetzen

## Meilenstein 1

- Zielgruppe: Studierende, Jobsuchende
- Gantt-Chart

## 9.5 Protokoll 24. Oktober 2025

Teilnehmer:

- Alexander Brendel
- Marcus Klein
- Jannika Börner
- Lukas Reinhardt (Meetingleitung)
- Norman Münzer
- Lara Seline Hippenstiel

### 9.5.1 Kundengespräch

#### Zwischenstand

##### Lukas

- Github Tickets
  - zukünftig vermutlich eher Roadmap anstatt Iterationen verwenden, da manche Tickets sich über mehrere Iterationen ziehen
  - Frage zu Status:
    - \* In-Progress = wird aktuell bearbeitet
    - \* Sprint-Backlog = geplant für den nächsten Sprint
    - \* Product-Backlog = noch zu tun
- Dokumentation
  - Organisation in Scrum, Ticket tracker
  - Meeting:
    - \* Fr 14 Uhr mit Kunden und Betreuer, danach Team-Meeting
    - \* optionaler Team-Meeting Slot: Dienstag Nachmittag (nur bei Bedarf)
  - Meeting-Protokoll wird von 2 Personen pro Meeting geschrieben
  - “Daily-Scrum” asynchron in Discord
    - \* Rückmeldung wie es funktioniert gewünscht

#### Jannika und Norman

- 1. Mockup aus erstem Kundengespräch (Norman)
- 2. Mockup, siehe Anhang
  - pep.digital-Logo nicht benötigt
  - Dokumente vielleicht trennen:
    - \* einmalige Dokumente
    - \* Firmenspezifische Dokumente (auch bei der Firmenansicht hinzufügen)
  - Fehlend: neue Bewerbung erstellen -; Paper-Prototype Meeting 07.10

Hinweis: langsam mit Implementierung anfangen

#### Lara

- Architektur
  - UML-Diagramm aus Meilenstein 1 präsentiert (siehe Abbildung 4.1)
  - für React, Spring Boot, PostgreSQL, Keycloak und Docker entschieden aufgrund Empfehlungen des Betreuer, Vorgaben aus der Vorstellungspräsentation und Verwendung von Spring Boot und PostgreSQL in parallel begleitendem Modul Softwarearchitektur
  - Frage bezüglich KI-Integration: welches Modell?

ab 14:22 Uhr Klemens

## Marcus

- Zeitabschätzung
  - Gantt-Chart
  - User Stories laufen über gesamtes Projekt
  - Funktionaler Prototyp so bald wie möglich gewünscht, aber spätestens bis 9.12 (Meilenstein 3)

## Sprintplan

- Spring-Boot
- Datenbank
- Frontend erste Komponenten
- Docker Container
- Bei jedem Entwickler lokal Grundlagen zum laufen bringen

kritische Nachfrage ob das erreicht werden kann

### 9.5.2 Betreuergespräch

- Web-App, App (Kundenfrage)
  - Web-App, über Browser erreichbar
- Desktop First? (Kundenfrage)
  - Ja, keine mobile Ansicht notwendig
- Npm - Wegen Wurm in VM arbeiten?
  - In VM oder geschachteltem Docker ausführen.

## Fragen zu Meilenstein 2 und Zwischenpräsentation

- Tabellarische Liste an User Stories erstellen:
  - Akteur = Stakeholder (Benutzer, Product Owner, Developer, ...)
  - Bsp: “As a developer i want to secure my environment” -> VM verwenden
- ARC 42 Template als Vorlage für verschiedene Sichten und Designentscheidungen
- Strukturansicht = Front-End, Back-End, ...
- Verhaltensansicht = was passiert wann?
- Verteilungssicht = ? -> ist noch zu klären
- verwendete Technologien/Frameworks = React, Spring Boot, ...
- Schnittstellen = wie kommunizieren die Frameworks usw. miteinander?
- Datenbank
  - Entity-Relationship Model
  - Object-Relationship Model (Klassendiagramm)

- lässt sich in IntelliJ automatisch erstellen
- Feedback zur Verbesserung des Softwarearchitektur-Diagramms:
  - REST ist HTTP, aber nicht alles an HTTP ist REST
    - \* REST benutzt nur CRUD
      - Create = POST
      - Read = GET
      - Update = PATCH bzw PUSH
      - Delete = DELETE
    - REST nutzt also nur 4 Grundlegende Funktionen für die Interaktion zwischen Daten aus der Datenbank und der App, HTTP hat deutlich mehr
    - Bsp: (Bitte Screenshot einfügen)
    - - bis Ende des path = Resource
    - - query = optional, zur Änderung der Darstellung einer Resource
  - Kommunikation Front-End & Back-End = HTTP
  - Kommunikation Back-End & Datenbank = JDBC (TCP)
  - Keycloak = HTTPS
- “Wie werden die potentiell unterschiedlichen Nutzergruppen mit der Applikation interagieren können?” = Zielgruppen (Personas)
  - Primäre Zielgruppe = Junge Erwachsene auf Job-/Praktikumssuche
  - Sekundäre Zielgruppe = mittelalte Erwachsene, denen gekündigt wurde oder die aus anderen Gründen nach einem anderen/neuen Arbeitsplatz suchen

## 9.6 Protokoll 31. Oktober 2025

Teilnehmer:

- Alexander Brendel
- Klemens Morbe
- Marcus Klein
- Jannika Börner
- Lukas Reinhardt
- Norman Münzer
- Lara Seline Hippenstiel (Meetingleitung)

### 9.6.1 Kundengespräch

#### Zwischenstand

##### Lukas

- Docker und Datenbank laufen und kommunizieren

### Jannika

- Vite Projekt gestartet und Routing eingerichtet
- Navigationskomponente erstellt
- Tabs initialisiert

### Fragen

- MaterialUI
  - nicht unverschlüsselt mit Backend kommunizieren
  - Verschlüsselung der Daten von Kunde abhängig (unverschlüsselt)
- Server auf Client Maschine
  - bei Enduser auf Server
  - Entwicklung: lokaler Server auf Client
- Linzenzen?
  - Apache 2/ MIT / GPL möglich
  - immer OpenSource oder irgendwann ClosedSource? → wir entscheiden

### Sprintplan

- Docker Container laufen bei jedem
- Einteilung der Aufgaben bei nächstem Meeting
- Daten von Backend an Frontend
- die verlorenen 16 Arbeitstage aufholen
- Backlog überarbeiten

### 9.6.2 Betreuergespräch

#### Präsentation

- Gliederung am Anfang gut
- Seitenzahlen wichtig
- Gliederungsspalte schlecht lesbar (Kontrast und Größe)
- Codebeispiele immer im Lightmode
- Personas: zu viel Text auf der Folie
- Hinweis: Java Logo darf nicht in öffentlichen Präsentationen verwendet werden
- Präsentation war inhaltlich gut
- mehr Augenkontakt
- freier sprechen
- weniger Zeit für die Vorbereitung reinstecken

## Architektur

- React und Springboot sollten über HTTPS laufen ansonsten unsicher
- eventuell mit Nginx

## IntelliJ Support

- Java 21 empfehlung
- Spring Boot:
  - Springboot Devtools
  - Lombok
  - Spring Web
  - Spring Security
  - OAuth2 Client
  - Spring Data JPA
  - PostgreSQL Driver
  - H2 Database (in Memory Database)
- Demo Application in /root ordner mit folgenden Paketen:
  - controller
  - entity ()
  - model (controller nach außen geben)
  - repository
  - service (interne Logik)

## 9.7 Protokoll 07. November 2025

Teilnehmer:

- Alexander Brendel
- Klemens Morbe
- Marcus Klein
- Jannika Börner
- Lukas Reinhardt
- Norman Münzer (Meetingleitung)
- Lara Seline Hippensiel

## 9.7.1 Kundengespräch

### Tagesordnung

1. Lukas: Fetch
2. Marcus: Keycloak
3. Norman: Frontend (Kalender)
4. Jannika: Frontend (Bewerbungen)
5. Fragen
6. Sprintplan

### Zwischenstand

#### Backend

- Daten aus der Datenbank gefetcht

#### Keycloak

- Grundlegend aufgesetzt und mit Testkonfiguration bereitgestellt

#### Kalender

- Grober Aufbau
- Provisorische Farben
- Noch hardcoded

#### Navigationsbar

- Weiterentwickelt mit Material UI
- Aufbau der Bewerbungen Seite
- Automatischer Umbruch bei verschiedenen Größen
- Eingebauter Light darkmode
  - Kommentar: keine weitere Zeit investieren

#### Fragen

- Dokumenteseite: Wie soll die Oberfläche für Dokumente aussehen?
  - Antwort: erstmal nach hinten Schieben und Bewerbungen fokussieren

#### Sprintplan

- Daten zwischen den Komponenten übertragen
- Übersichtsseiten von Bewerbung und Termin
- Datenbank Prototyp aufbauen
- Darstellung von Daten im Frontend im Testcase (welche Seite ist nicht Relevant)
- Jira Board: Tasks sollen im Board auf Child-Ebene dargestellt werden

### 9.7.2 Betreuergespräch

Frontend Kalender:

- Material UI Standardtheme verwenden um Zeit zu sparen

Fetch mit Axios oder übern Browserfetch?

- Axios ist mächtiger

Docker Container:

- Einfacher halten und das Projekt über die IDE lokal bauen

Qualitätssicherungsprozess:

- Empfehlung einen zu implementieren

Zeitschätzung:

- Ist Herr Nitzsche wichtig
- Für schon vergangene Tickets zumindest die Dauer nachfragen
- Anmerkung schreiben warum es ggf. länger oder kürzer gedauert hat

Meilenstein

- User Stories:
  - Was kann die App am Ende?
  - Umfang der App darstellen
- Verwendete Schnittstellen Technologie
  - Verwendete Protokolle
  - Verwendete Schnittstellen beschreiben
- Datenbank
  - Kann mit IntelliJ generiert werden

Ticketaufbau

- Titel mit Prädikat
- Userstory in Beschreibung

ERM-Diagramm

- Anpassungen am Diagramm
  - ID Anpassungen bei Appointment
  - Dokument: Category und Filetype hinzufügen
- Adresse in separater Tabelle?
  - Kommt drauf an
- Company separat pro User (Adressbuch)
- Ist ein User auch ein Kontakt?

- Anstoß: Minio S3Bucket zur Ablage von Dokumentdateien
- Dokumente ohne UserID aber noch mit Alex abklären wie die Dokumentenablage aussehen soll

CI/CD-Pipeline

- Eventgetriebenes oder Zeitgetriebenes automatische Deployment mit Testingstruktur
- SonarQube (Qualitätssicherung)
- GitHub Actions

Was ist bei dem Prototyp zu erwarten

- Keine spezifischen Anforderung
- Einfach nur das was schon funktioniert
- Es wäre nicht schlecht, wenn es technisch Funktioniert

KI-Nutzung

- Vermerken wo und wie KI-verwendet wurde
- Nicht zwingend jeden Prompt dokumentieren
- Bei regelmäßigen Anwendungen (bspw. Fehlerfindung oder Entscheidungsfindung) als Hinweis in der Doku erwähnen

## 9.8 Protokoll 14. November 2025

Teilnehmer:

- Alexander Brendel bis 14:37
- Klemens Morbe ab 14:20
- Marcus Klein
- Jannika Börner (Meetingleitung)
- Lukas Reinhardt
- Norman Münzer
- Lara Seline Hippenstiel

### 9.8.1 Kundengespräch

#### Zwischenstand

1. Lara: angepasstes Jira Board zeigen
  - Testdaten in Datenbank eingefügt
2. Jannika: Aufrufen der Daten im Backend (via Bewerbungsansicht)
3. Norman: Zeigen der Terminansicht nun in MUI
  - Termine noch Hardcoded

## Terminverschiebung

- Aufteilung von Kunden und Betreuerteil wäre möglich, falls nötig
  - Alex: Montagnachmittag, Dienstag/Mittwoch relativ flexibel
  - Klemens: Montag-, Mittwoch- oder Donnerstagnachmittag
- Vorläufige Entscheidung zum Testen:
  - Meetings von 14-15 Uhr mit hartem Ende
  - Bei Bedarf individuelle Terminfindung mit Klemens

## Sprintplan

Anmerkung der Guppe: Großteil hat kommende Woche wenig Zeit

- Alle: Story Zeiteinschätzungen auf Jira nachholen
- Lara + Lukas: Datenweitergabemöglichkeiten im Backend erweitern
- Jannika: Korrekte Daten im Frontend aufrufen und verarbeiten
- Norman: Einarbeitung von Kalenderdaten in Frontend holen
- Marcus: Keycloakauthentifizierung für Datenübertragung fertig stellen
- Jannika: Dialog zum eintragen der Bewerbungen erstellen (max. 1 Tag)
  - Falls geschafft - Lara: Datenspeicherung und -aktualisierung ermöglichen

## Feedback

- Besseres Vorbereiten von Präsentationen und Technischen geräten
- Fragen auch im Kundenchat stellen

Ende 14:37

### 9.8.2 Betreuergespräch

Probleme der letzten Woche:

- COARS Error im Frontend
  - Verschiedene Ports
  - Zwei Möglichkeiten zum Vermeiden:
    1. Reverse Proxy, d.h. Routing intern übernehmen: localhost leitet weiter mit localhost:port/backend, nginx
    2. SpringSecurityConfig in Backend mit .csrf.disable

Fragen:

- Wie soll die UserID bei get-Anfragen weitergegeben werden (Frontend)?
  - Nicht explizit mitschicken
  - Durch Token von Keycloak (username, vor-nachname, expiration-Date, refreshability) → Token in Header

- Keycloak wird eigene DB haben, man kann einstellen dass sie in Postgres steht via Docker Compose
- Vorrübergehend Workaround wird uns Zeit kosten, User Management vermeiden bis Keycloak steht
- Wie sollen wir die Queries im Backend schreiben?
  - Interface ist JPA (Jakarta Persistence API)
  - JPQL oder HQL in Java (Interface ist JVA, verschiedene Implementationen d.h. Hibernate, etc.)
  - 3 Arten wie man auf etwas zugreifen kann:
    1. Custom Query in Repository, wegen flacher Struktur kann man flache Sachen einfach mit SQL-like Language suchen, Join muss nicht explizit beschrieben werden
    2. Durch Name kann man Query beschreiben (siehe IntelliJ Vorschläge), mit dieser Art macht er einen impliziten Join Empfehlung: nicht verwenden; für einfache Sachen OK aber für komplexe ist es schnell Fehleranfällig, ist außerdem mit normalem SQL genauso lesbar
    3. Braucht transaction, annotieren mit transactional ... was war Methode? LAZY vs EAGER Fetchtypes?
- Zeitplan der Betreuer
  - Meilensteine als Orientierung
  - Alle liegen im Zeitplan zurück, aber Vgl. mit anderen Teams sollte nicht gemacht werden
  - Gleichmäßiges Tempo wäre sinnvoll

Ende 16:20

## 9.9 Protokoll 21. Monat 2025

Teilnehmer:

- Alexander Brendel bis 14:19
- Klemens Morbe
- Marcus Klein (Meetingleitung)
- Jannika Börner
- Lukas Reinhhardt
- Norman Münzer
- Lara Seline Hippensiel

Subway ist besser als Nutellabrot (Klemens)  
Nutellabrot ist besser als Subway (Alexander)

### 9.9.1 Kundengespräch

#### Zwischenstand

Es ist noch nichts fertig, aber Zwischenstand ist einsehbar

- Alle: eingeschränkte Zeitliche Verfügbarkeit
- Alle: Story abschätzungen mit Zeiten nachholen
  - teilweise gemacht
- Jannika+Lara: Korrekte Daten bei Cards
  - Lukas übernommen, jetzt korrekt vorhanden
  - Karten fertig
- Norman: einarbeitung von Kalenderdaten in Frontend holen
  - nicht geschafft
- Lara: Datenaktualisierung in Datenbank ermöglichen
  - sollte auf Frontend warten
- Jannika: Bewerbungen erstellen Seite (max 1 Tag Zeitansatz)
  - aufgeschoben
- Marcus: Keycloak authentifizierung für Datenübertragung
  - nicht geschafft
  - merge des bestehenden mit anderen Branches

#### Fragen

- Was für Funktionalitäten sollen im Prototyp vertreten sein? (Jannika)
  - Bewerbungen anlegen, ansehen und bearbeiten
  - Kalender
  - Login
- Für Prototyp nicht gefordert
  - Sortierung
  - Dokumentenablage
- Repository öffentlich schalten?
  - Zeitpunkt egal
  - Plan: Wenn der Durchstich/Prototyp steht

#### Sprintplan

- Alle: Story abschätzungen mit Zeiten nachholen + Recherche
- Norman+Lukas: einarbeitung von Kalenderdaten in Frontend holen
- Marcus: Keycloak authentifizierung für Datenübertragung (bis nächste Woche geplant)

## 9 Appendix

- Jannika+Lara: Stellenanzeigen mit vollen Backend Daten
- optional: Bewerbungen erstellen Seite (max 1 Tag Zeitansatz)
- Teaminternes Meeting am Wochenende

### Meeting Feedback

- Bitte darum Kamera anzumachen

Ende 14:19

### 9.9.2 Betreuergespräch

#### Fragen

- Such und Filterfunktion:
  - Auf den schon geladenen Daten suchen
- Von Betreuer: Frage zur Testschnittstelle
  - Testschnittstelle provisorisch entstanden aus Fehlkommunikation
  - gibt nicht alle für diese Ansicht nötigen Daten aus
- Dockerfile
  - passt wie es gerade ist
- backend [clean,install] funktioniert nicht richtig
  - entspricht  $\_$  ng build
  - sprinboot:run startet springboot lokal
  - BackendApplication macht in der Regel das gleiche wie sprinboot:run
  - Bei ähnlichen Problemen Stacktrace in Discord schicken
- Codereview: Muss jeder alles erklären können?
  - Jeder muss Alles gesehen und verstanden haben
  - Konkrete Erklärung von allem ist nicht erwartet
  - Jeder sollte grob im Bilde

#### Git

- Pullrequests verwenden um wissen zu verteilen und zu überprüfen, was in Main kommt
  - Test Pullrequest mit Jannika
- Conventional Commits verwenden (schon als Lesezeichen gespeichert)

Ende 14:53

## 9.10 Protokoll 28. November 2025

Teilnehmer:

- Alexander Brendel bis 14:30
- Klemens Morbe
- Marcus Klein
- Lukas Reinhardt (Meetingleitung)
- Lara Seline Hippenstiel
- Norman Münzer
- Jannika Börner

### 9.10.1 Kundengespräch

#### Fortschritt vorgestellt

- Präsentiert: Funktionierenden Keycloak Login + Logout gezeigt
- Präsentiert: Funktionierendes Bewerbungen-Hinzufügen-Formular
- Erwähnung: Terminseite Fortschritt und Planung

#### Zwischenstand

War geplant bis zum 28.11.2025:

- Alle: Story Abschätzungen mit Zeiten nachholen + Recherche
- Norman+Lukas: Einarbeitung wie Kalenderdaten ins Frontend geholt werden
- Marcus: Keycloak Authentifizierung für Datenübertragung (bis nächste Woche geplant)
- Jannika+Lara: Stellenanzeigen mit vollen Backend Daten
- optional: Seite auf der Bewerbungen hinzugefügt werden können erstellen
- Teaminternes Meeting am Wochenende

Erlledigt bis zum 28.11.2025:

- Keycloak Authentifizierung
- Bewerbungen Hinzufügen Button(Komponente) + Bewerbungen-Hinzufügen-Formular mit Backendanbindung
- Terminansicht:
  - Daten aus dem Backend in Terminliste abgerufen und dargestellt
  - Müssen noch in Kalender eingefügt werden

## Fragen

- Soll das Hinzufügenformular in einem Pop-Up ausgelagert werden?
  - Nein, im Tab lassen
- Wie soll im Hinzufügen-Formular mit Firmendaten umgegangen werden, wenn ein User eine bestehende Firma auswählt und abweichende Daten eingibt?
  - Adresse entweder überschreiben lassen oder entkoppelt von Unternehmen abspeichern (erstmal aber nicht so wichtig)
- Welche der Daten beim Hinzufügen einer Bewerbung sind erforderlich und welche sind optional?
  - Jobtitel und Firma sind mandatory, Rest ist optional
  - Adresse kann auch nur teilweise eingegeben werden

## Sprintplan

- Jannika: detaillierte Stellenansicht formatiert im Frontend darstellen
- Lara, Jannika: Bearbeitenfunktion Backend + Frontend, anschließend sobald wie möglich die Löschenfunktion anfangen (eventuell aber erst nächsten Sprint möglich)
- Norman: Termindaten im Kalender anzeigen, danach Beginnen die Termineingabe zu ermöglichen
- Lukas, Marcus: Backendarbindung von Keycloak (Schätzung 2 Wochen, nach Prototyp Abgabe)

## Umfang Prototyp Funktionalität

- Hinzufügen
- Bearbeiten
- Löschen
- Ansehen

Ende 14:30

## 9.10.2 Betreuergespräch

### Fragen und Feedback von Klemens

- Zum Formular:
  - Feldbreite limitieren damit Felder nicht ewig breit und untereinander sind -*i* So groß dass es passend für Inhalt ist
  - Hint und Label tauschen (Text für Label ist sehr lang)
  - Beschriftung links vom Feld, nur 1-2 Wörter
  - Termine ganz am Anfang oder Ende des Formulars, ggf. sogar nur separat im Terminereiter
- Zum Starten des Programms:
  - Run-Config erstellen um nur Start Button in IntelliJ drücken zu müssen

### Fragen von Team

- Wie Programm/Code für Meilenstein abgeben?
  - To Do: Node Modules aus Git raus löschen
  - Es ist die Doku hochzuladen, nicht die Zip der Anwendung
  - Die Doku ins Git Repository hochladen
  - In der Doku die Quelle des Git Repositorys + Commit Stand einfügen
- Bei Dateneingabe in die Datenbank werden Felder meistens mit null aber hin und wieder auch komplett ohne Wert eingespeichert. Ist das ein Problem und wenn ja, wie löst man es?
  - Ggf. durch Skriptgenerierung und Sample-Datenbefüllung
  - Spring Boot befüllt automatisch die Datenbank basierend auf der Entity Deklaration
- Warum wird die Frontend Seite doppelt geladen?
  - Grund ließ sich während Betreuer Gespäch leider nicht rausfinden -; später recherchieren
- Wie arbeitet man mit User ID zwischen Keycloak und Datenbank?
  - User ID von Keycloak in die Backenddatenbank übernehmen?

Ende 15:01

### 9.11 Protokoll 05. Dezember 2025

Teilnehmer:

- Alexander Brendel bis 14:17
- Klemens Morbe
- (Praktikant)
- Marcus Klein (Meetingleitung übernommen)
- Jannika Börner (Meetingleitung angefangen)
- Lukas Reinhardt
- Norman Münzer

Abgemeldet:

- Lara Seline Hippenstiel

### 9.11.1 Kundengespräch

#### Fortschritt

- Stellenansicht Anzeige
- Bearbeitenansicht im Frontend implementiert (Backend noch in Arbeit)
- Termine Ansicht:
  - Termine in Liste und Kalender
  - Hoverfunktion im Kalender
- Keycloak Anbindung im Backend:
  - Userspezifische Anzeige mit Email
  - Hinzufügen in Abhängigkeit vom User

#### Fragen

- Farbcodierung im Kalender
  - Kann (wenn überhaupt eingebaut) bis nach dem Prototypen warten
- Alte Kalenderdaten sollen angezeigt bleiben
  - keinen Filter für Termine im Kalender einbauen
- Bearbeitenfunktion noch vor dem Prototypen fertig machen?
  - Auf Grund der aktuellen Situation muss es nicht fertig werden.

#### Zwischenstand

Geplant:

- Keycloak Authentifizierung
  - Authentifizierung via Email (unique steht noch aus) (Marcus präsentiert)
- Bewerbungen Hinzufügen Button(Komponente) + Formular mit Backendanbindung
  - Button Komponente steht (Jannika präsentiert, Backend fehlt hier)
- Terminansicht:
  - Daten aus dem Backend in Terminliste abgerufen und dargestellt
    - \* funktioniert
  - Müssen noch in Kalender eingefügt werden
    - \* gemacht (Normann präsentiert)

#### Sprintplan

- Dokumentation überarbeiten (nichts aus der Nase ziehen)
- Bearbeiten Funktion (erstmal bei Lara)

Ende 14:17

### 9.11.2 Betreuergespräch

- Termineansicht: Nicht zu viele Farben verwenden, besser mit konstanter Farbpalette
- Nicht zu viel mit Farben verkünsteln
- Auf Lesbarkeit des Inhalts achten

#### Fragen

erstmal in Merge Branch oder direkt in main pullen?

- Main in den eigenen Branch mergen und dann wenn alles läuft Pull Request auf den Main.

Ende 14:25

## 9.12 Protokoll 12. Dezember 2025

Teilnehmer:

- Alexander Brendel
- Klemens Morbe
- Marcus Klein
- Jannika Börner
- Lukas Reinhardt
- Norman Münzer (Meetingleitung)

Abgemeldet:

- Lara Seline Hippenstiel

### 9.12.1 Kundengespräch

#### Zwischenstand

- Dokumentation überarbeiten (nichts aus der Nase ziehen)
  - für Meilenstein 3 gemacht
- Bearbeiten Funktion (erstmal bei Lara)
  - weiterhin krank
- Pop-Up bei Termine hinzufügen steht
  - Backend noch nicht verbunden
- Startseite (noch nicht in main)

Klemens kommt dazu

#### Anmerkungen

- UI auf Startseite noch nicht konsistent
- Label in Termine Pop-Up (durch git probleme nicht gezeigt, schon vorhanden)

## Fragen

- weitere User hinzufügen?
  - ja, soll noch gemacht werden

Jannika möchte unbedingt das Jira board machen

## Sprintplan

- Dokumentation neu strukturieren (Marcus)
- Bearbeiten Funktion (Lara)
- Backendanbindung Bewerbungen löschen (Jannika) + Benutzerprofil anfangen
- Backendrestrukturierung (Norman + Lukas(krank))
- Aufgaben suchen, Teaminterne Absprachen

Ende 14:16

## 9.12.2 Betreuergespräch

### Code-Review

- michael.watzko@uxitra.de
- Termin Freitag 14:00 Uhr über Webex pro Person 15min
- worum es geht:
  - Jeder hat etwas gemacht und nicht nur von anderen präsentiert
  - kein Vibe-Coding
  - max 1h vorbereiten
  - annotations verstehen
  - Grundlagen verstehen, nicht code 1:1 nachvollziehen
  - Team wählt aus was erklärt wird
  - nicht Code vorlesen
  - erklärung auch in Team-trennung möglich, sollte aber mit 1h vorbereitung für alle gehen
  - nicht zu viel sagen
  - nachfragen nach Tests oder Parametern bspw. möglich
  - konstruktiv/positiv auf Fragen reagieren
  - nicht auf andere verweisen, wir sind ein Team
  - kurze präsentation des Projekts
    - \* 1min
    - \* Techstack
    - \* Ziel
    - \* Teammitglieder
- Lara wird nicht da sein
- davor Kamera und technik durchtesten

## Fragen

- präsentieren bei Code-Review von aktuell oder Meilenstein 3
  - wenn große neue Features noch nicht gemerged, dann Featurebranch
  - sonst main
  - keine Fehler
  - für Backup, Videos okay, eigentlich life gewollt

## weiteres

- Code mit sauberer Trennung als Beispiel
  - <https://gist.github.com/kmo-pep/c5c077651997136d4c5ba593f0e7aef9>
- Klassen zusammenlegen?

Ende 15:03

## 9.13 Protokoll 09. Januar 2026

Teilnehmer:

- Alexander Brendel
- Klemens Morbe
- Marcus Klein (Meetingleitung)
- Jannika Börner
- Lukas Reinhardt
- Norman Münzer

Abgemeldet:

- Jannika
- Lara Seline Hippenstiel

### 9.13.1 Kundengespräch

#### Zwischenstand

- Startseite (noch nicht in main)
  - in main
- Dokumentation neu strukturieren (Marcus)
- Bearbeiten Funktion (Lara)
  - in Arbeit, wird von Lukas übernommen
- Backendanbindung Bewerbungen löschen (Jannika) + Benutzerprofil anfangen
  - Appointments löschen geht
  - Bewerbungen geht noch nicht

- Backendrestrukturierung (Norman + Lukas)
  - soweit fertig
- Aufgaben suchen, Teaminterne Absprachen
  - Keycloak update (Marcus)
    - \* neuste Version
    - \* User import
    - \* Login UI
  - User registration (Marcus)
    - \* UI über Keycloak
    - \* Backend verbunden
  - Einstellungen (Norman)
    - \* noch nicht fertig
    - \* Als Seitenleiste
    - \* Theme einstellbar

### Anmerkungen

- Termine erstellen (Norman)
  - Senden rechts orientiert und umbenennen
  - Kalender und ToDo-Liste kompakter für Pfeile
  - Pflichtfelder mit Sternchen ausstatten
- Bewerbung erstellen
  - Firmen Auto complete sollte einfach neu erstellen, passt jetzt auch so
- UI Theme
  - standard MUI wäre schneller gewesen
  - Überschrift nicht überall
  - einheitliches Seitenlayout
  - auf Kontraste für Lesbarkeit achten
- Dokumente komplett entfernen
- Einstellungen komplett verwerfen: Default vom Browser verwenden für dark/light mode
- Firma sollte nicht als eigene Entität in der Datenbank geführt werden:
  - sorgt für zu viel Komplexität
  - Adresse in Firmenentität sorgt dafür dass eine Firma nur eine Adresse haben kann.
  - nicht mehr umänderbar

## Fragen

- Auto-complete, soll der User die Firma in Bewerbungen komplett austauschen können?
  - Namen der Firma kann zu anderer Firma geändert werden, wenn Name nicht mit existierenden Firmen übereinstimmt wird neue Firma erstellt
- soll Darkmode separat einstellbar sein, oder nur Browser standard übernehmen?
  - Broswer Standard und separat

## Sprintplan

- Backendanbindung Bewerbungen löschen (Jannika + Lukas)
- Bearbeiten fertig (Lukas)
- Doku aktualisieren (großteil Lukas wegen Backendrestrukturierung)
- Lizenz festlegen und einfügen (Marcus)
- Frontend in Docker einbinden (Marcus + Lukas) (verzichten)
- CI Pipeline bei Pull-Request einrichten (Marcus) (verzichten)
- Einstellungen verwerfen (Norman)
- Keycloak Login Theme variable, Sternchen in Registrierung hinter Text verschieben, Anmelden Farbe und Registrieren Farbe von Knopf unterschiedlich (Marcus)
- Feature Branches bereinigen, Fortschritt ist nicht zu sehen bei zu vielen Branches

Ende 14:58

## 9.13.2 Betreuergespräch

- Überschneidend mit Kundengespräch
- Teaminterne Kommunikationsprobleme(Lukas, Lara)

## Abschluss-Präsentation

- Termin 21.01.2026 14:00 Uhr
- Vorlage von Hochschule nehmen
- Muss nicht schön aussehen, muss inhaltlich gut sein
- Seitenzahlen!
- frei reden, keine Karteikarten
- nicht so viel Zeit reinstecken
- ohne Lara

Ende 15:02

## 9.14 Protokoll 16. Januar 2026

Teilnehmer:

- Alexander Brendel
- Klemens Morbe
- Marcus Klein
- Jannika Börner
- Lukas Reinhardt (Meetingleitung)
- Norman Münzer
- Jannika

Abgemeldet:

- Lara Seline Hippenstiel

### 9.14.1 Kundengespräch

#### Zwischenstand

- separate Company entität
  - ist umgesetzt
- Backendanbindung Bewerbungen löschen (Jannika + Lukas)
  - in Arbeit
- Bearbeiten fertig (Lukas)
  - läuft in branch, pull wartet noch auf dependency
- Doku aktualisieren (großteil Lukas wegen Backendrestrukturierung)
  - angefangen, einige Grafen stehen
- Lizenz festlegen und einfügen (Marcus)
  - fertig
  - MIT
- Keycloak Login Theme variable, Sternchen in Registrierung hinter Text verschieben, Anmelden Farbe und Registrieren Farbe von Knopf unterschiedlich (Marcus)
  - fertig
  - sternchen fehlt
- Termine überarbeitet zur einheitlichkeit
- Feature Branches bereinigen, Fortschritt ist nicht zu sehen bei zu vielen Branches
  - 2 branches inactive von Lara
  - 1 milestone branch
  - 2 branches active developement

### Anmerkungen

- Keycloak, Sternchen hinter Text, Hyperlinks in Schwarz, Augen abgerundet
- Bewerbungen, Einrückung komisch
- Termine, hätte kleinerer/dünnerer Container sein können, appointmentName zu weniger technischem Namen, termine nach Datum sortiert

Ende 14:39

### 9.14.2 Betreuergespräch

#### Anmerkungen

- Exceptions spezifisch halten
- Kanban Board überarbeiten/aktualisieren

#### Doku

- Quality Gate: SonarCube in Doku aufnehmen
- Nur einzelne Diagramme, nicht alles
- “aus Priorisierung vom Kunden wurde das Frontend nicht containerisiert”
- Bei Reflektion
  - Krankheit hat zu unplanbarkeiten geführt
  - viel Einarbeitungszeit
- Lesson learned:
  - Ticket schreiben hat viel Zeit gefressen und nicht viel geholfen
- KI verwendung vermerken

#### Abschluss-Präsentation

- Vorlage von Hochschule nehmen
- Muss nicht schön aussehen, muss inhaltlich gut sein
- Seitenzahlen!
- frei reden, keine Karteikarten
- nicht so viel Zeit reinstecken
- Als Fallback 1min video durch Anwendung klicken
- Projektplan
  - so war es geplant
  - so sieht es aus
  - da sind wir jetzt
- für Feedback früh genug bei Klemens melden

Ende 16:20

## **9.15 Erweiterte Eigenständigkeitserklärung**

**Erweiterte Eigenständigkeitserklärung zur Verwendung generativer KI-Systeme  
als Hilfsmittel**

Hiermit versichere ich, dass ich diese Prüfungsleistung selbständig verfasst habe.  
Die verwendeten Quellen und Hilfsmittel habe ich angegeben.

Sofern ich generative KI-Systeme über die Prüfung von Grammatik und Rechtschreibung von eigenständig erstelltem Text hinaus genutzt habe, bestätige ich hiermit, dass ich:

- die von generativen KI-Systemen direkt übernommenen Inhalte im inhaltlichen Teil der Arbeit entsprechend markiert bzw. zitiert habe,
- sichergestellt habe, dass die mit Hilfe der genannten generativen KI-Systemen erzeugten und sonstigen von mir verwendeten Inhalte faktisch korrekt sind,
- mir darüber im Klaren bin, dass ich als Verfasserin oder Verfasser dieser Arbeit die Verantwortung für die darin enthaltenen Angaben und Aussagen übernehme,
- sofern ich generative KI-Systeme zur Unterstützung wesentlicher Arbeitsschritte eingesetzt habe, die nicht bereits im inhaltlichen Teil der Arbeit oder den Referenzen thematisiert bzw. zitiert werden, dies tabellarisch erfasst habe. Die Tabelle habe ich mit Abgabe der Arbeit eingereicht (siehe Anhang).

## Anhang der erweiterten Eigenständigkeitserklärung

Arbeitsschritt	Eingesetzte(s) Gen-KI System(e)	Verwendung in Kapitel	Beschreibung der Verwendungsweise
Ausformulierung und Formatierung	ChatGPT 5.2 bis 4.0	komplette Dokumentation	KI wurde verwendet zur Umformatierung und als Formulierungshilfe (Umformulieren, Stichpunkte zu Fließtext ausarbeiten und ähnliches)
Unterstützung beim Code schreiben	siehe Kapitel 5.4 in der Dokumentation		

Lukas Reinhart

Esslingen, 21.01.2026, Lukas Reinhart

Ort, Datum, Unterschrift

Marcus Klein

Esslingen 21.01.2026 M.K.

Ort, Datum, Unterschrift

Jannika Börner

Esslingen 21.01.2026 Jannika Börner

Ort, Datum, Unterschrift

Norman Münzer

Esslingen 21.01.2026 N.M.

Ort, Datum, Unterschrift

Lara Hippenstil

Ort, Datum, Unterschrift

## *9 Appendix*

Anmerkung: Lara Hippenstiel war zu diesem Zeitpunkt seit geraumer Zeit nicht mehr aktiv am Projekt beteiligt und hat sich zu diesem Punkt nicht mehr geäußert. Dementsprechend fehlt ihre Unterschrift.