

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

Abbildungsverzeichnis	III
1 Meilenstein 01	1
1.1 Problemstellung	1
1.2 Lösungsansatz	1
1.3 Zielgruppe	1
1.4 Geforderter Funktionsumfang	1
1.5 UI-Entwürfe	2
1.5.1 Erster UI-Entwurf	2
1.5.2 Verfeinerung des UI-Konzepts	2
1.6 Architektur der Software	3
1.6.1 Übersicht zentraler Komponenten	3
1.6.2 Beschreibung des allgemeinen Ablaufs	4
1.7 Projektmanagement	4
1.7.1 Meetings	4
1.7.2 Kommunikation	5
1.8 Aufwandsschätzung	5
2 Meilenstein 02	10
2.1 User Stories	10
2.1.1 Entwickler	10
2.1.2 Benutzer	10
2.2 Technisches Konzept	12
2.2.1 Softwarearchitektur inklusive unterschiedlicher (logischer) Sichten	12
2.2.2 Verwendete Technologien / Frameworks	17
2.2.3 Verwendete Schnittstellentechnologie	18
2.2.4 Datenbank	19
3 Meilenstein 03	20
3.1 Technisches Konzept	20
3.1.1 Softwarearchitektur	20
3.1.2 Verhalten	23
3.2 Schnittstellenbeschreibung	25
3.2.1 Übersicht der Stellenausschreibungen	25
3.2.2 Detaillierte Stellenausschreibung ansehen	27
3.2.3 Stellenausschreibung hinzufügen	31
3.2.4 Übersicht der Termine	35
3.3 Linkssammlung	38
4 Appendix	i
4.1 Protokoll 07. Oktober 2025	i
4.2 Protokoll 17. Oktober 2025	xii
4.2.1 Kundengespräch	xii
4.2.2 Betreuergespräch	xii

Inhaltsverzeichnis

4.3	Protokoll 24. Oktober 2025	xiii
4.3.1	Kundengespräch	xiv
4.3.2	Betreuergespräch	xv
4.4	Protokoll 31. Oktober 2025	xvi
4.4.1	Kundengespräch	xvi
4.4.2	Betreuergespräch	xvii
4.5	Protokoll 07. November 2025	xviii
4.5.1	Kundengespräch	xix
4.5.2	Betreuergespräch	xx
4.6	Protokoll 14. November 2025	xxi
4.6.1	Kundengespräch	xxi
4.6.2	Betreuergespräch	xxii
4.7	Protokoll 21. Monat 2025	xxiii
4.7.1	Kundengespräch	xxiv
4.7.2	Betreuergespräch	xxv
4.8	Protokoll 28. Monat 2025	xxvi
4.8.1	Kundengespräch	xxvi
4.8.2	Betreuergespräch	xxvii
4.9	Protokoll 05. Dezember 2025	xxviii
4.9.1	Kundengespräch	xxviii
4.9.2	Betreuergespräch	xxix

Abbildungsverzeichnis

1.1	Bevorzugtes 1. Mock-Up: Front Page	5
1.2	Bevorzugtes 1. Mock-Up: Bewerbungsseite	6
1.3	Ausgearbeitetes 2. Mock-Up Kalender	6
1.4	Ausgearbeitetes 2. Mock-Up Bewerbungen	7
1.5	Ausgearbeitetes 2. Mock-Up Unternehmensansicht	7
1.6	Ausgearbeitetes 2. Mock-Up Dokumente	8
1.7	Architektur: Zentrale Komponenten	8
1.8	Aufwandsschätzung: Gantt-Diagramm Stand Meilenstein 1	9
2.1	Struktursicht auf Container-Ebene	12
2.2	Struktursicht auf Komponenten-Ebene	13
2.3	Struktursicht auf Code-Ebene (Frontend)	14
2.4	Struktursicht auf Code-Ebene (Backend)	15
2.5	Verhaltenssicht an Bsp. Bewerbungsübersicht	15
2.6	Verteilungssicht	16
3.1	Struktursicht der Pages auf Code-Ebene	20
3.2	Struktursicht der Bewerbungen auf Code-Ebene (Frontend)	21
3.3	Struktursicht der Termine auf Code-Ebene (Frontend)	21
3.4	Struktursicht auf Code Ebene (Backend)	22
3.5	Sequenz-Diagramm Bewerbungen	23
3.6	Sequenz-Diagramm Termine	24
3.7	Sequenz-Diagramm Login	24
3.8	Datenmodell der Joboffer Overview Schnittstelle	26
3.9	Status der Axios Anfrage	26
3.10	Joboffer Details API (1/2)	29
3.11	Joboffer Details API (2/2)	30
3.12	Status der Axios Anfrage	31
3.13	Add Joboffer API (1/2)	33
3.14	Add Joboffer API (2/2)	34
3.15	Datenmodell der Appointment Overview Schnittstelle	36
3.16	Status der Axios Anfrage	36

1 Meilenstein 01

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 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 zu 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

1.5 UI-Entwürfe

1.5.1 Erster UI-Entwurf

Bei Betrachtung der ersten Mock-Ups (siehe Abbildungen 1.1, 1.2) wurden einige Punkte beschlossen:

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

1.5.2 Verfeinerung des UI-Konzepts

Startseite/Kalender

In der überarbeiteten Version dienen der Kalender und die Termiňübersicht gleichzeitig auch als Startseite (siehe Abbildung 1.3). Im oberen Bereich befindet sich, wie bei der 1. 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. Hier können alle Termine eingetragen und tabellarisch mit Informationen zu Unternehmen, Art des Terms, 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 1.4) 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 zu erfassen und hinzuzufügen. Unter den Bedienfeldern wird der Seiteninhalt in Kachelansicht angezeigt. Jede Kachel repräsentiert eine Firma, bei der eine Bewerbung eingereicht wurde. In der Kachel sind 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 1.5). 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. Darunter folgen in Feldern zuerst der Firmenname und dann ein Feld mit dem nächsten relevanten Termin. Im nächsten Feld wird eine Kurzbeschreibung des Unternehmens eingefügt. Darunter folgen Kontaktpersonen, darunter der Gehaltsspielraum und die Distanz. Als letztes gibt es die Perks zum Arbeitsplatz wie z.B. Homeoffice, Geschäftswagen, usw.

Dokumente

Im letzten Hauptbereich der Anwendung, “Dokumente” (siehe Abbildung 1.6), 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.

1.6 Architektur der Software

Die Architektur der Software setzt sich aus vier zentralen Komponenten zusammen, welche in Docker betrieben werden. Im Diagramm (siehe Abbildung 1.7) erkennt man wie das Frontend, das Backend, das Authentifizierungssystem und die Datenbank miteinander interagieren.

1.6.1 Übersicht zentraler Komponenten

1. Frontend (React)
 - Läuft im Browser des Nutzers, wo dieser mit 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
2. Authentifizierung (Keycloak)
 - Das Identity- und Access-Management-System (IAM) Keycloak übernimmt Authentifizierung

- Überprüft Anmelddaten und gibt bei erfolgreichem Login ein Access Token an das Frontend (React) zurück
3. Backend (Spring Boot)
- Ermöglicht den Zugriff auf REST-API-Endpunkte
 - Empfängt Anfragen des Frontends (React), überprüft die Gültigkeit des Access Tokens
 - Gültig: Verarbeitet die Anfrage weiter und ruft Daten von der Datenbank (PostgreSQL) ab
 - Ungültig: Automatische Weiterleitung zur Authentifizierung (Keycloak)
4. Datenbank (PostgreSQL)
- Strukturelles Speichern der Daten
 - Kommuniziert über JDBC mit dem Backend (Spring Boot)
 - Ermöglicht das Abrufen, Speichern und Aktualisieren der Nutzerdaten und Anwendungsinformationen

1.6.2 Beschreibung des allgemeinen Ablaufs

1. Nutzer öffnet die Webanwendung im Browser und wird bei fehlendem gültigen Access Token an Keycloak-Loginseite weitergeleitet.
2. Access Token wird nach erfolgreicher Anmeldung an das Frontend übergeben, während Nutzer zeitgleich zurück zur Webanwendung geleitet wird.
3. Nutzer interagiert, wodurch eine Anfrage mit Access Token an Spring Boot Backend übersendet wird.
4. Das Backend überprüft die Validität des Tokens und ruft, speichert bzw. ändert ggf. Daten in der PostgreSQL-Datenbank ab.
5. Das Backend übergibt die Daten an das Frontend, welches diese anzeigt.
6. Ablauf wiederholt sich ab Schritt 3. bis die Session des Nutzers beendet wird.

1.7 Projektmanagement

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

1.7.1 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 vom Team gemeinsam in einem Hedge-Doc-Dokument erstellt. 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.

1.7.2 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.

1.8 Aufwandsschätzung

Für eine initiale Aufwandsschätzung wurden die Meilensteine als feste Werte in das Gantt-Diagramm, Abbildung 1.8, 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.



Abbildung 1.1: Bevorzugtes 1. Mock-Up: Front Page

1 Meilenstein 01

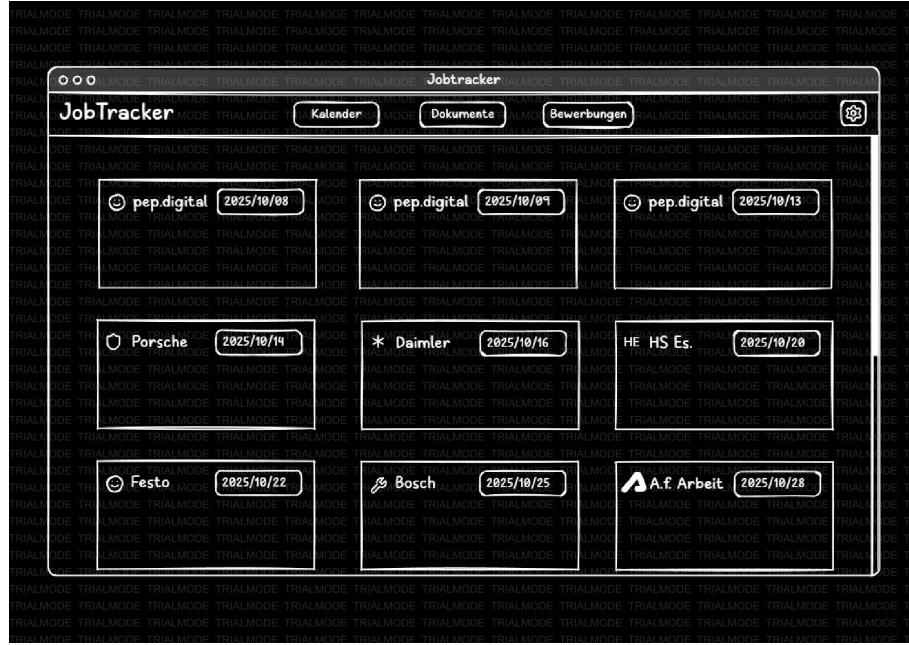


Abbildung 1.2: Bevorzugtes 1. Mock-Up: Bewerbungsseite

The interface includes a header with tabs for 'Termine', 'Bewerbungen', 'Dokumente', and a gear icon. The 'Termine' section shows two entries:

Unternehmen	Art des Termins	Datum	Uhrzeit	To Do
Firma 1	Bewerbungsgespräch	20.10.2025	14:00	-
Firma 2	Abgabe Bewerbungsunterlage	20.10.2025	-	Zeugnis noch hinzufügen

The 'Kalender' section shows a monthly view for October 2025, with specific dates highlighted in blue and red, indicating scheduled events for both companies.

Abbildung 1.3: Ausgearbeitetes 2. Mock-Up Kalender

1 Meilenstein 01

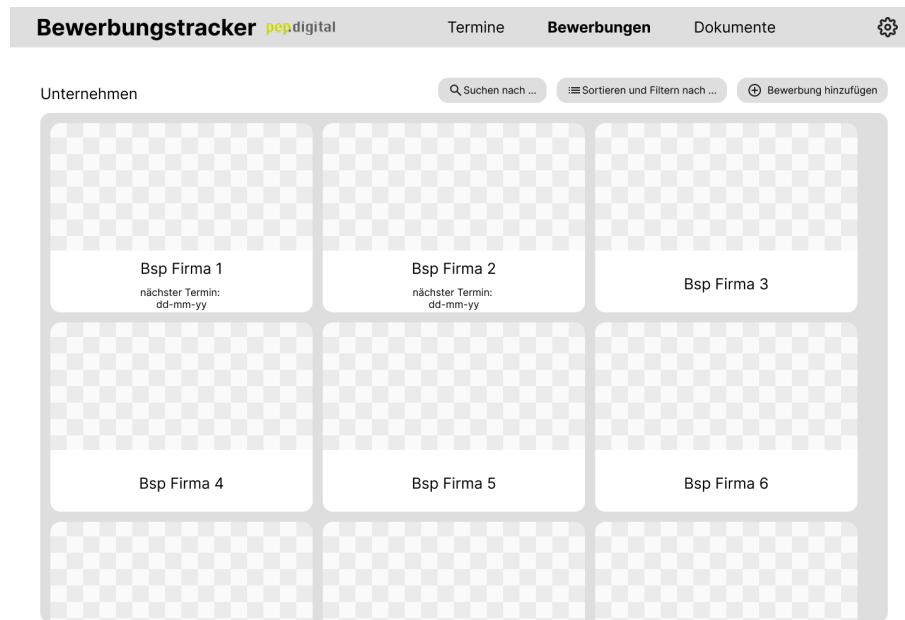


Abbildung 1.4: Ausgearbeitetes 2. Mock-Up Bewerbungen



Abbildung 1.5: Ausgearbeitetes 2. Mock-Up Unternehmensansicht

1 Meilenstein 01



Abbildung 1.6: Ausgearbeitetes 2. Mock-Up Dokumente

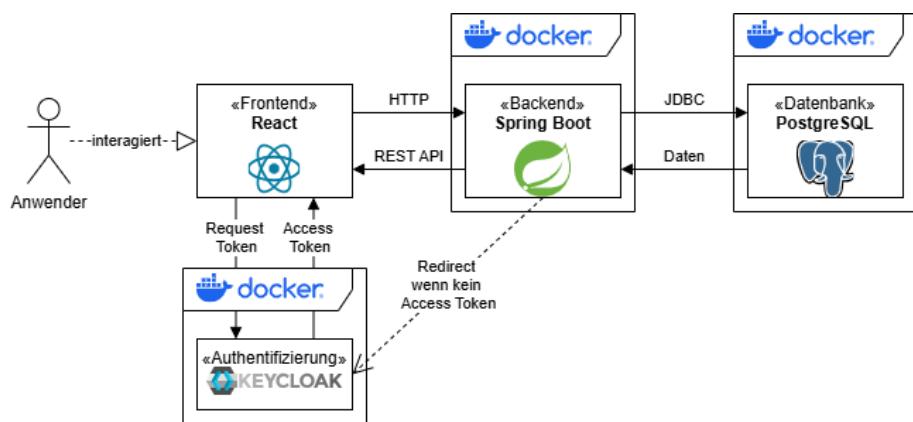


Abbildung 1.7: Architektur: Zentrale Komponenten

1 Meilenstein 01

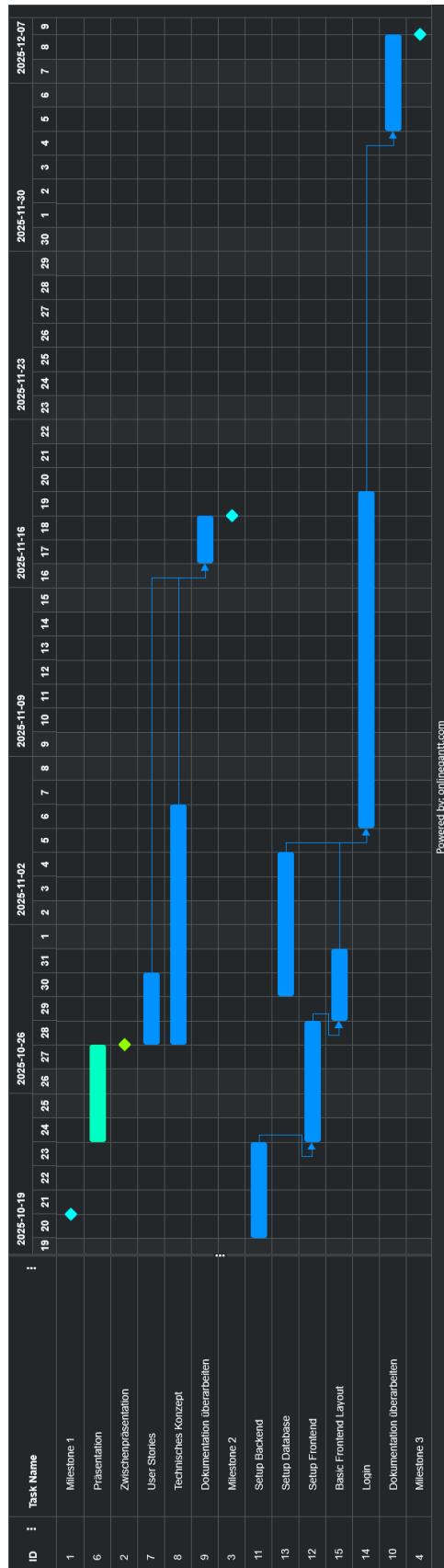


Abbildung 1.8: Aufwandsschätzung: Gantt-Diagramm Stand Meilenstein 1

2 Meilenstein 02

2.1 User Stories

2.1.1 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 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.

2.1.2 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.

2.2 Technisches Konzept

2.2.1 Softwarearchitektur inklusive unterschiedlicher (logischer) Sichten

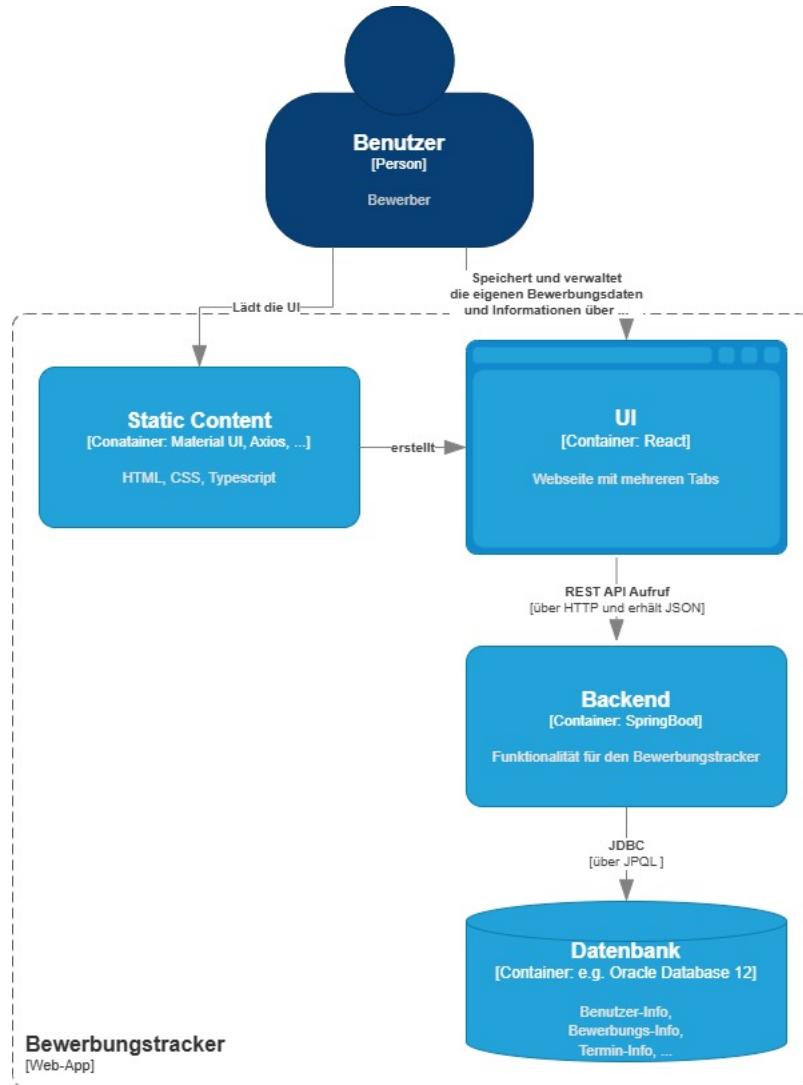


Abbildung 2.1: Struktursicht auf Container-Ebene

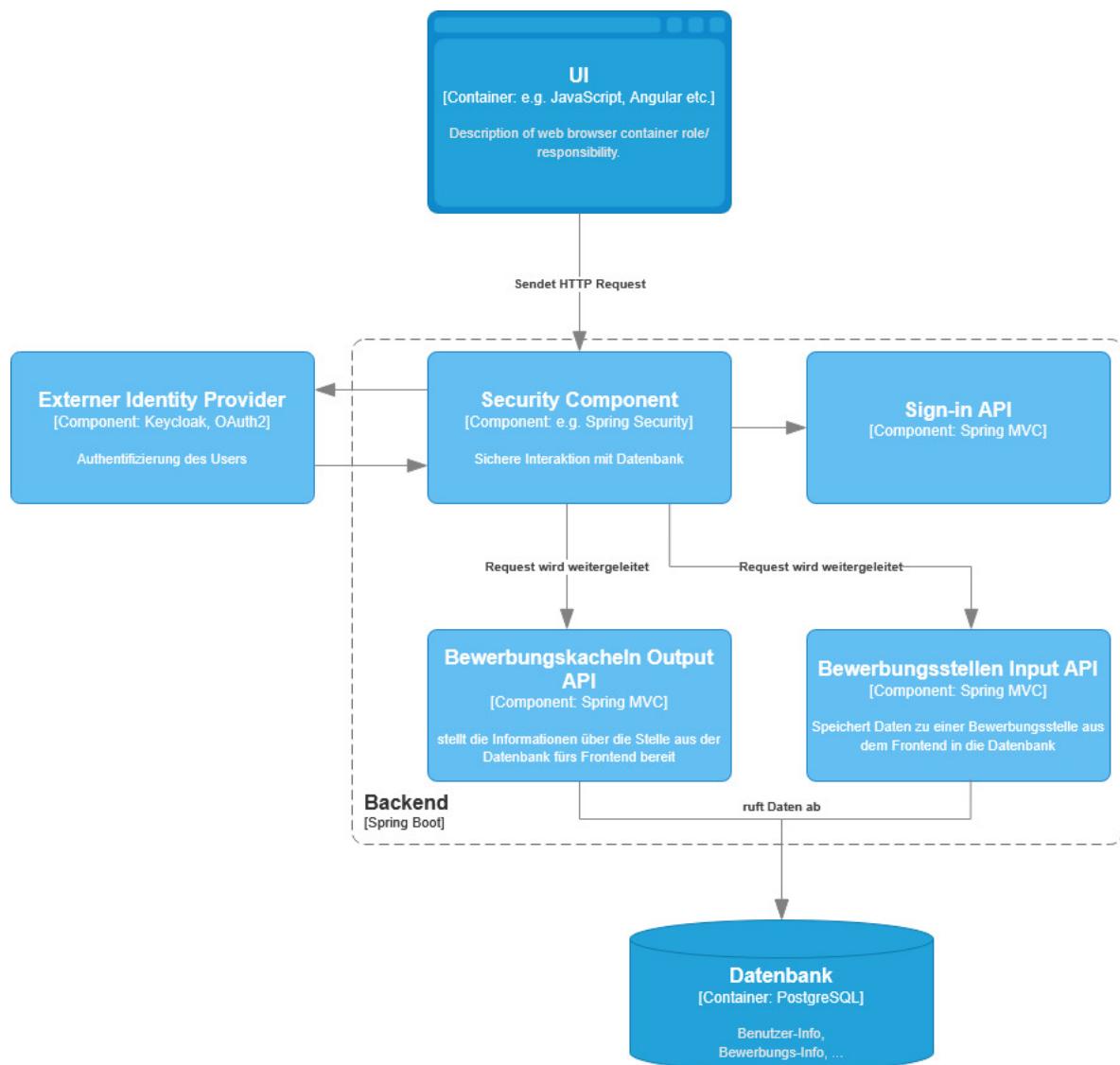


Abbildung 2.2: Struktursicht auf Komponenten-Ebene

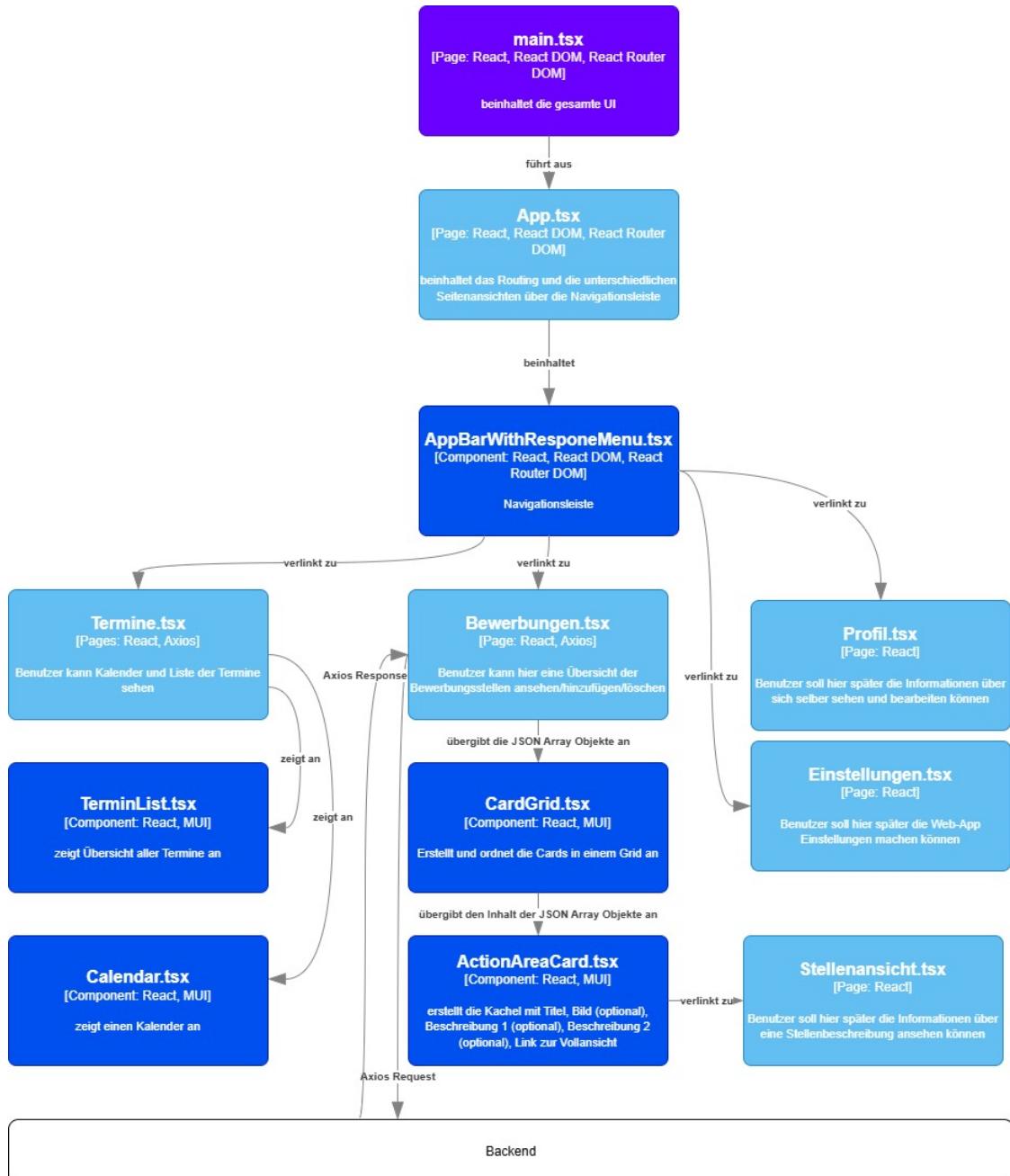


Abbildung 2.3: Struktursicht auf Code-Ebene (Frontend)

2 Meilenstein 02

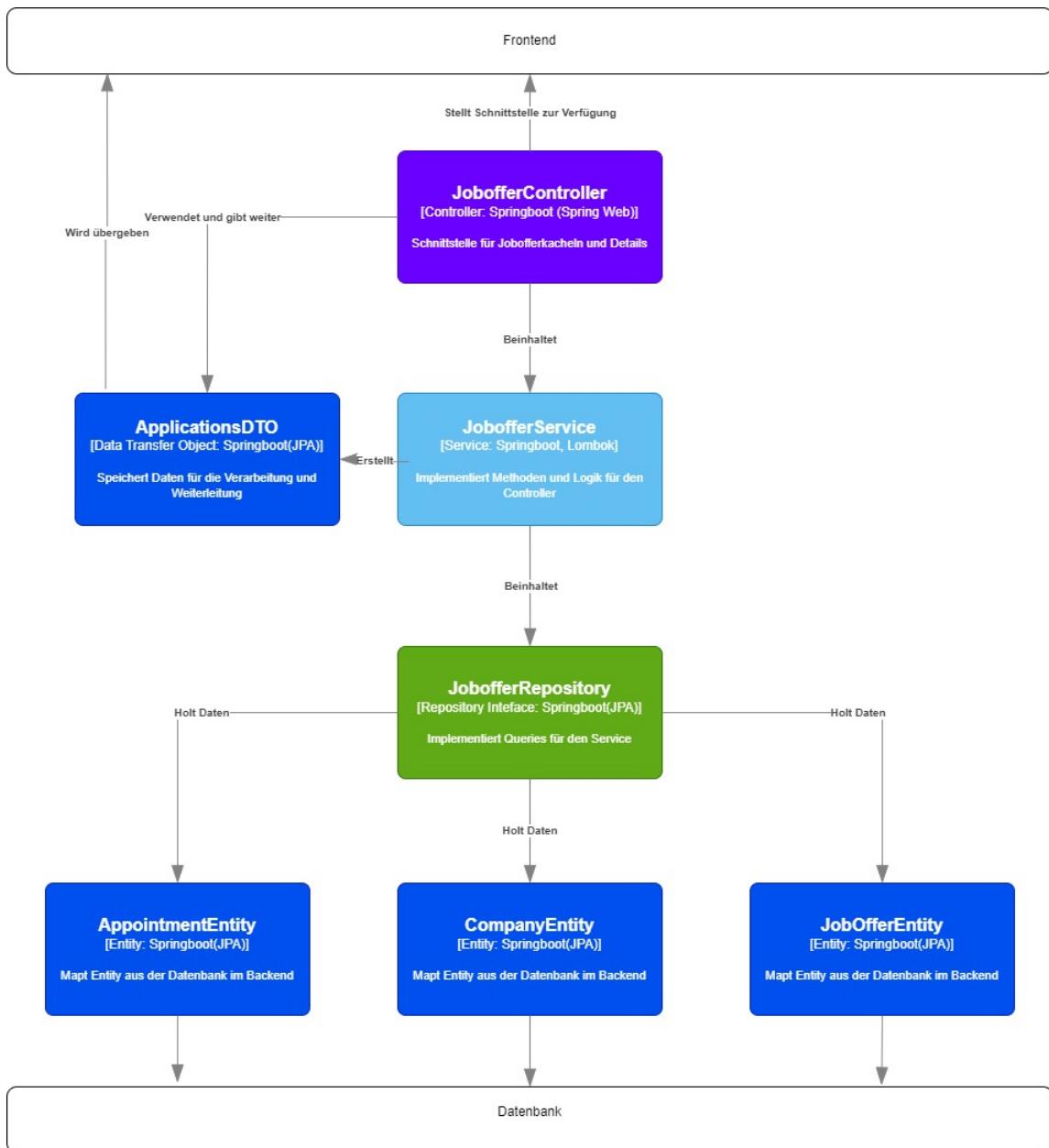


Abbildung 2.4: Struktursicht auf Code-Ebene (Backend)

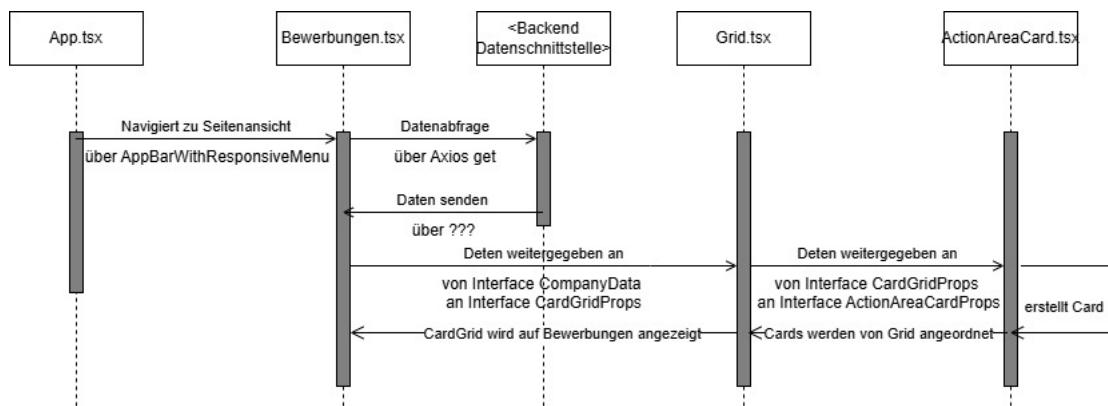


Abbildung 2.5: Verhaltenssicht an Bsp. Bewerbungsübersicht

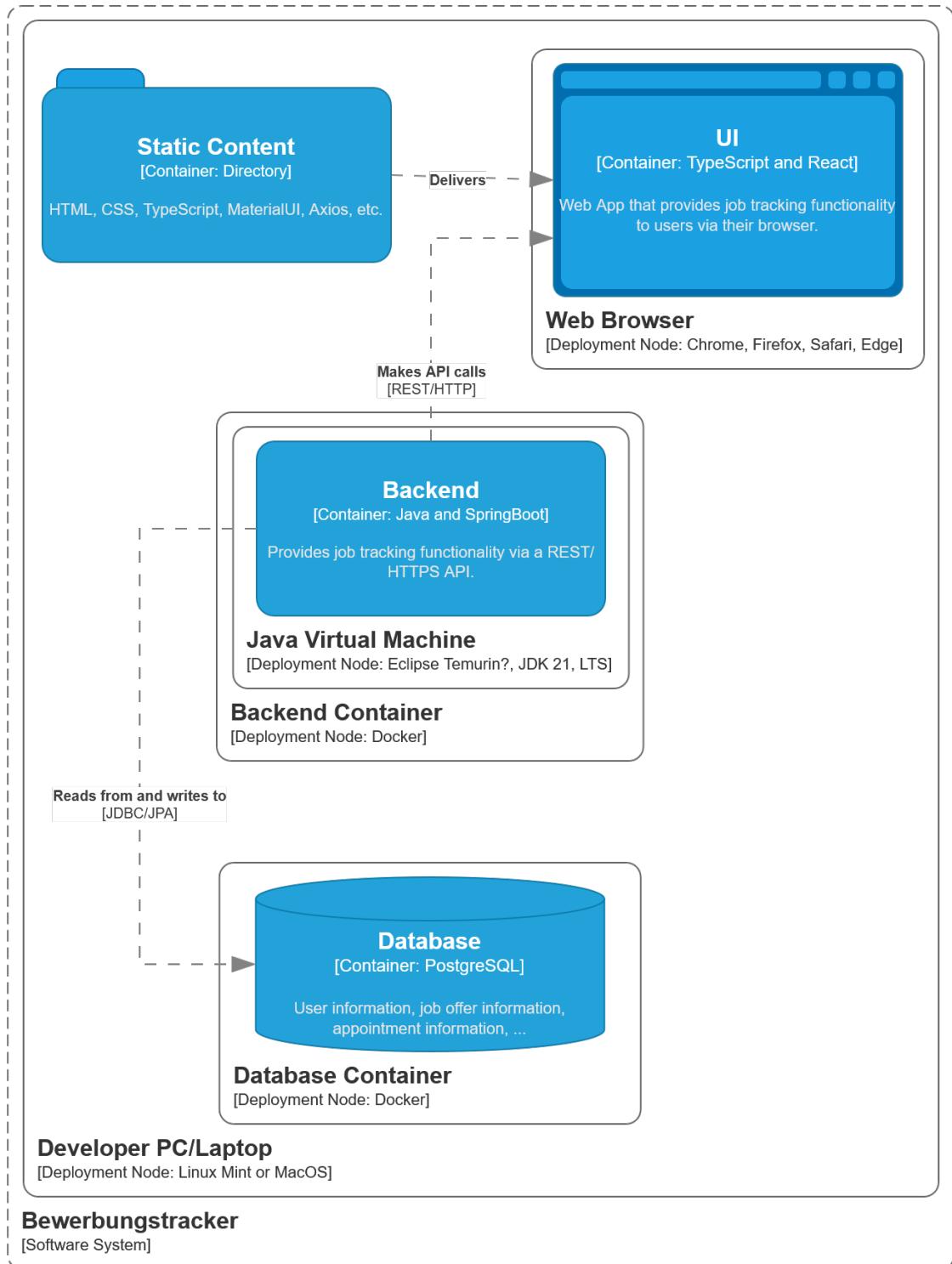


Abbildung 2.6: Verteilungssicht

2.2.2 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 „Schnellstart“, 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.

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

Backend – Java / Maven / Spring Boot

Java als Programmiersprache war für uns bereits vorgegeben. Maven war eine implizierte Entscheidung, da wir alle damit bereits gearbeitet hatten. Zur Auswahl des Java-Frameworks kamen für uns Spring Boot oder Quarkus in Frage. Quarkus ist zwar extrem schnell und light-weight, wenn es um Ressourcen geht, aber hat derzeit noch ein wachsendes Ökosystem, wodurch es nicht so flexibel bei der Integration mit anderen Technologien ist. Spring Boot hat dagegen ein sehr großes und etabliertes Ökosystem, wodurch es sehr leicht ist andere Technologien zu integrieren, was für uns als Laien im Endeffekt wichtiger war als eine extrem schnelle App. Des Weiteren kann uns unser Betreuer auch hier bei Schwierigkeiten weiterhelfen und ggf. auch Tipps geben. Zudem lernen wir Spring Boot parallel in einer unserer Module.

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 Modul 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

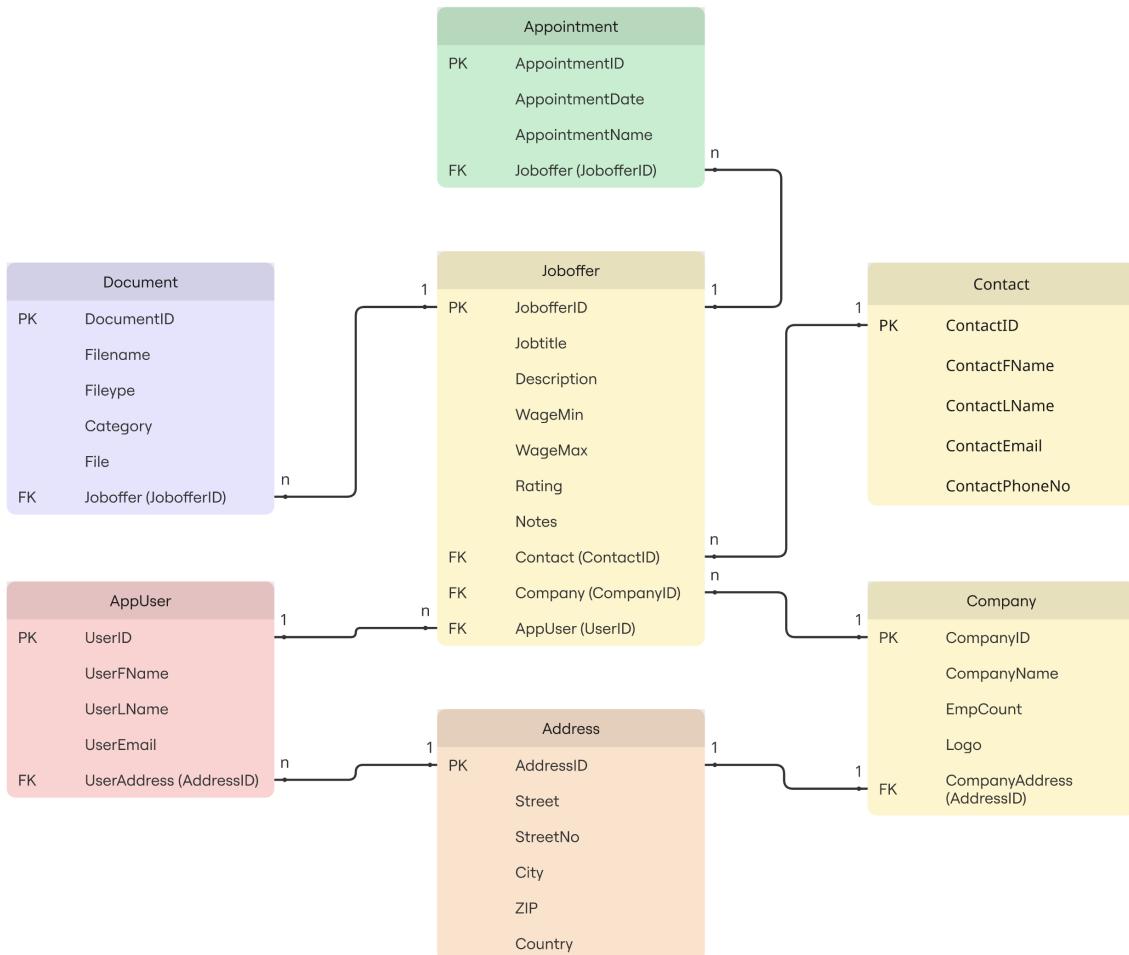
Keycloak als IAM war uns im allgemeinen bereits vorgeschrieben, da es für uns die Login-Logik vollständig übernimmt und dabei auf Standard-Protokollen basiert ist es eine ideale Möglichkeit die Login-Logik sicher zu implementieren. Zusätzlich bietet Keycloak auch sehr viel Flexibilität und Anpassbarkeit. 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.

2.2.3 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 über die Schnittstelle JDBC auf die Postgres Datenbank zu.

2.2.4 Datenbank

Logisches Datenmodell



In der Grafik sind alle gelben Felder relevante Informationen für eine Stellenausschreibung (d.h. die Stellenausschreibung an sich, die entsprechende Kontaktperson und die Firma). In orange dargestellt ist die Tabelle für Adressen, da diese sich sowohl auf eine Firma (gelb) als auch auf einen User (rot) beziehen kann. In lila und grün sind jeweils Dokumente und Termine, da diese ihre eigenen Ansichtsseiten haben.

3 Meilenstein 03

In Meilenstein 3 wurden die Architekturdiagramme überarbeitet und erweitert, entsprechend des aktuellen Stands der Software. Ebenso wurden Schnittstellenbeschreibungen hinzugefügt.

3.1 Technisches Konzept

3.1.1 Softwarearchitektur

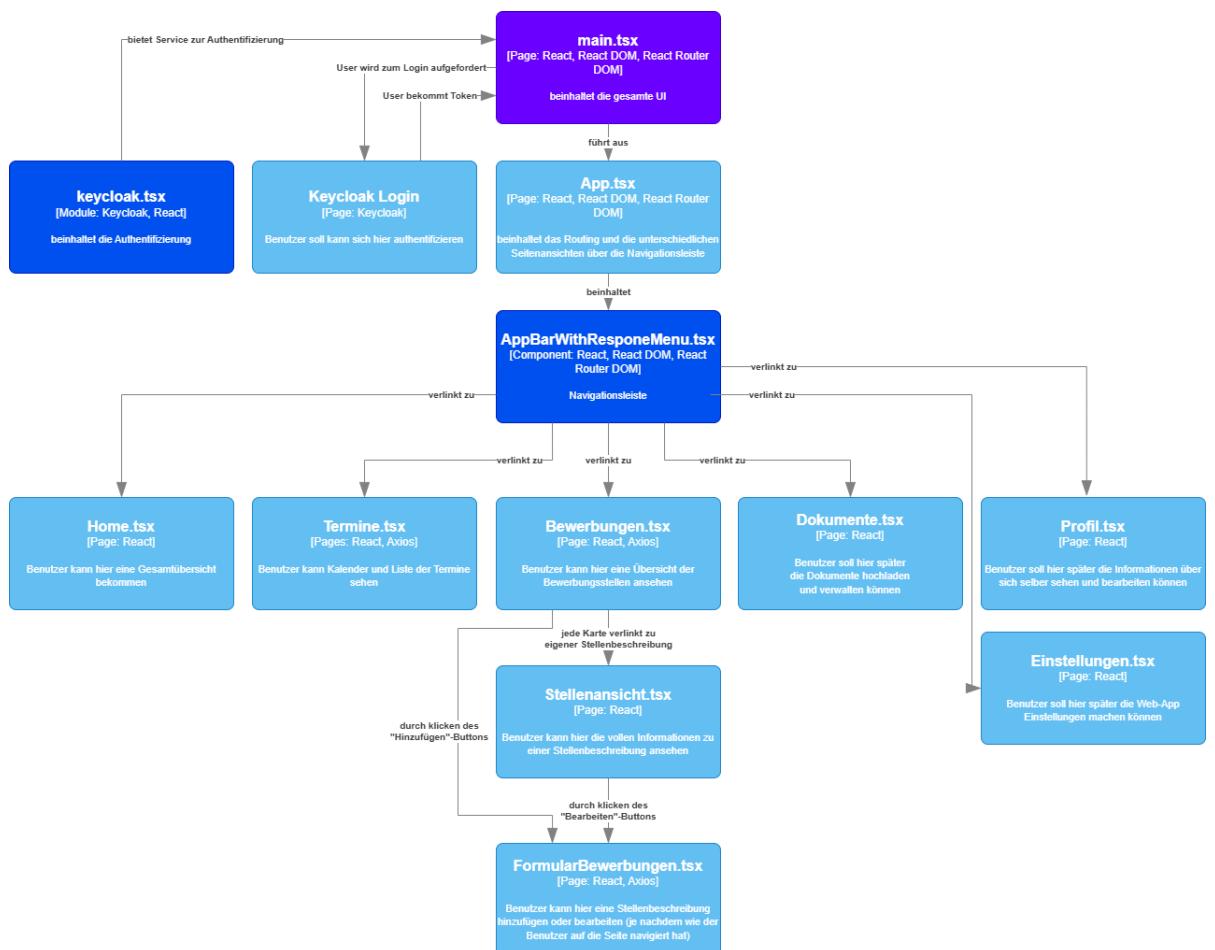


Abbildung 3.1: Struktursicht der Pages auf Code-Ebene

3 Meilenstein 03

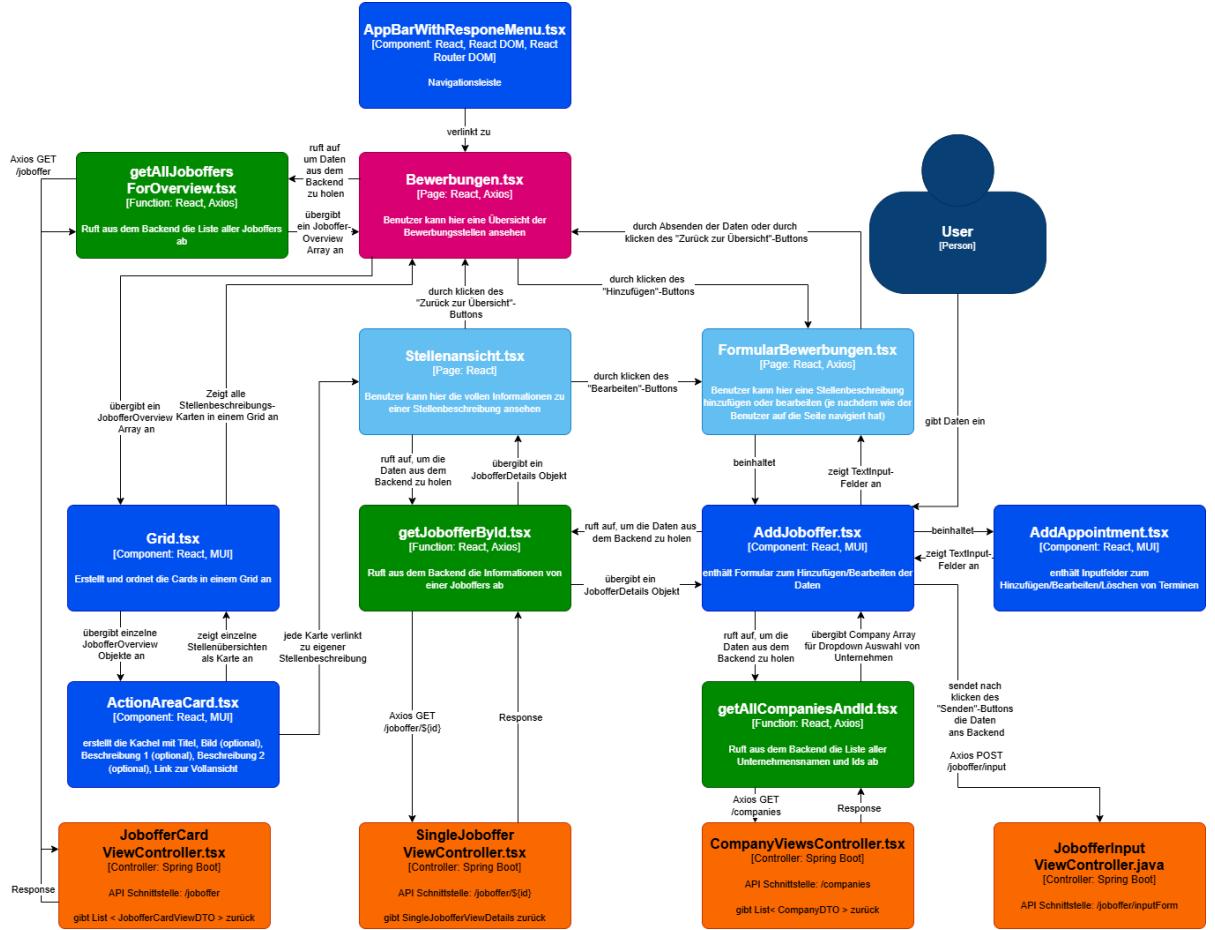


Abbildung 3.2: Struktursicht der Bewerbungen auf Code-Ebene (Frontend)

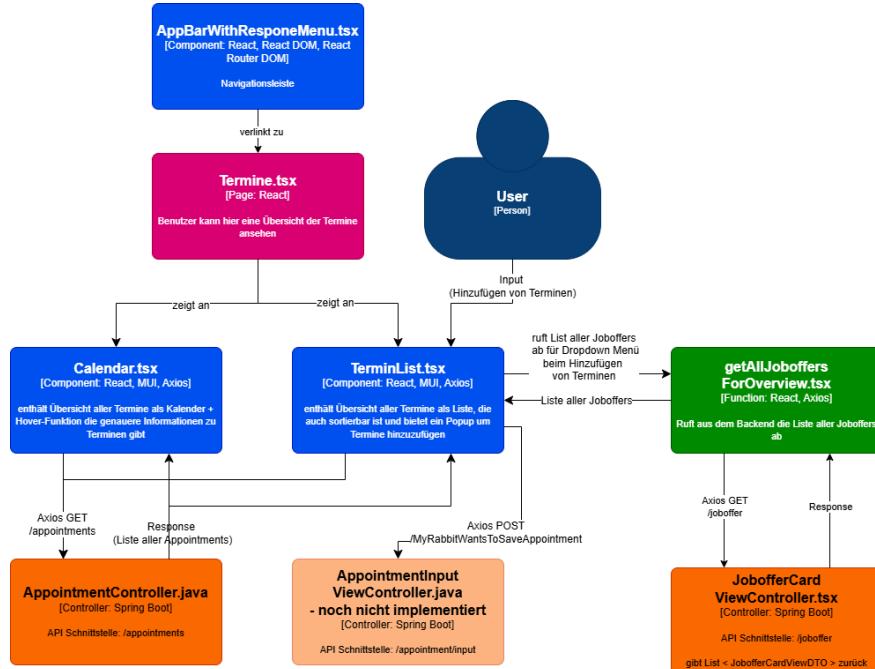


Abbildung 3.3: Struktursicht der Termine auf Code-Ebene (Frontend)

3 Meilenstein 03

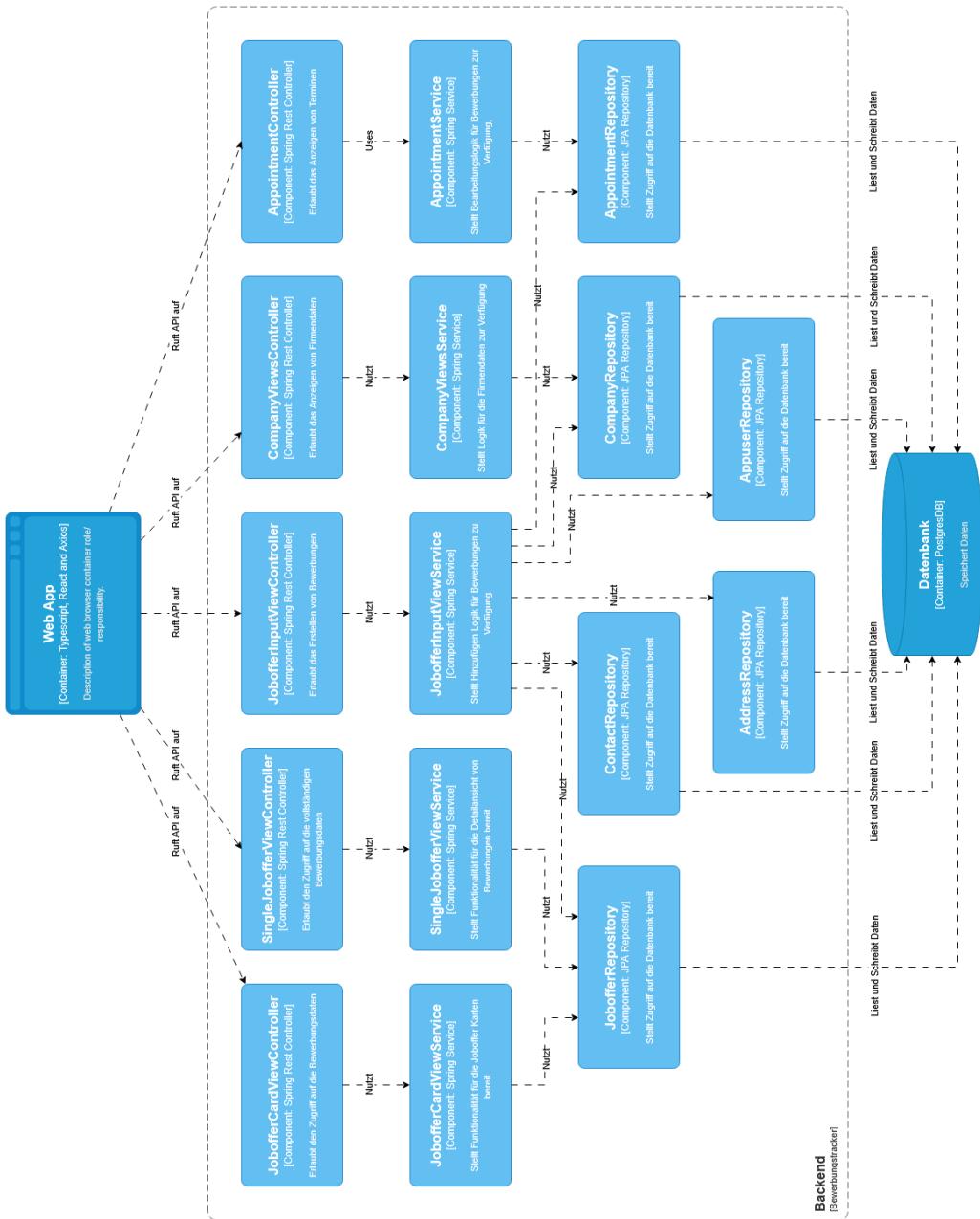


Abbildung 3.4: Struktursicht auf Code Ebene (Backend)

3.1.2 Verhalten

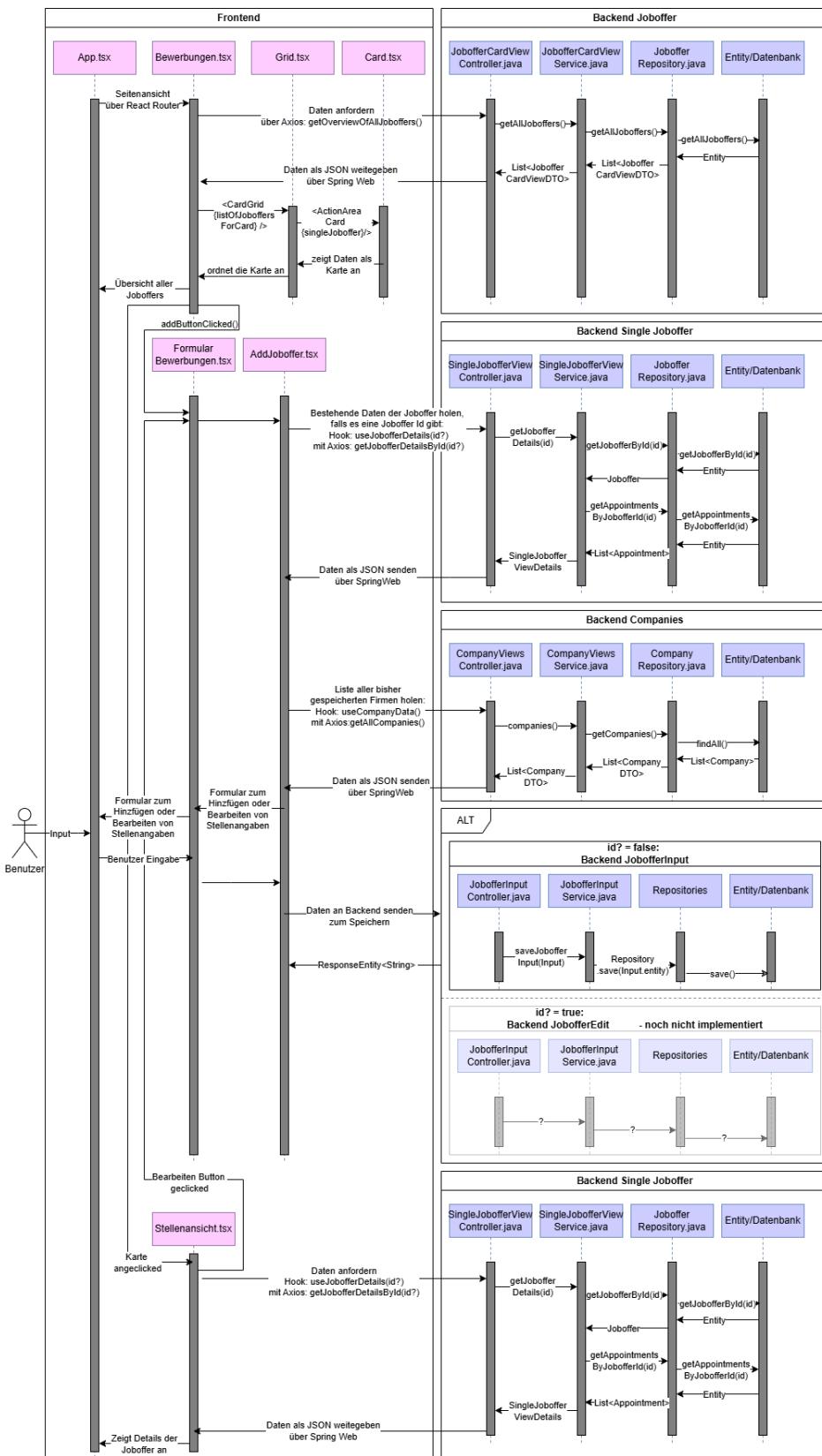


Abbildung 3.5: Sequenz-Diagramm Bewerbungen

3 Meilenstein 03

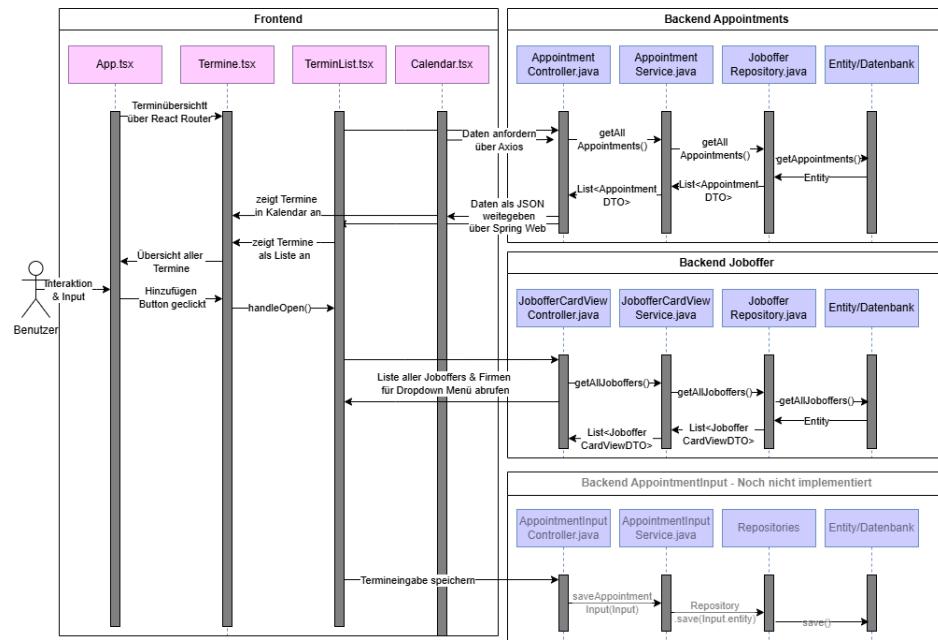


Abbildung 3.6: Sequenz-Diagramm Termine

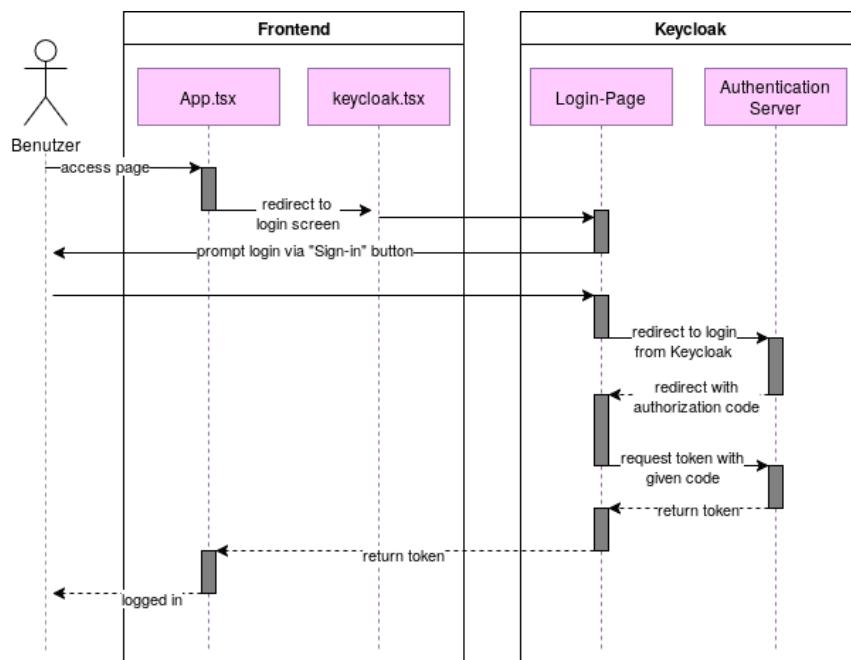


Abbildung 3.7: Sequenz-Diagramm Login

3.2 Schnittstellenbeschreibung

3.2.1 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:** `JobofferCardViewController.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,  
   "companynamen": "Schäfer KG"  
 },  
 {  
   "jobofferid": 2,  
   "joboffername": "Dolores error non blanditiis id dolor.",  
   "companyid": 1,  
   "companynamen": "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: parseDateToNext AppointmentString (joboffer.nextapptdate)

Abbildung 3.8: 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 3.9: Status der Axios Anfrage

Zusammenfassung

Die **Joboffer Overview API** stellt eine kompakte Auflistung aller Stellenausschreibungen zur Verfügung. Sie wird hauptsächlich zur Darstellung einer Kachelansicht genutzt.

3.2.2 Detaillierte Stellenausschreibung ansehen

Endpoint: /joboffer/{id}

Beschreibung

Dieser Endpoint liefert alle Angaben zu einer einzelnen Stellenausschreibung (Joboffer). Er wird vom Frontend verwendet, um eine vollständige Stellenausschreibung anzuzeigen.

Request

- **Methode:** GET
- **URL (Frontend):** `http://localhost:8080/joboffer/{id}`
- **Backend-Controller:** JobofferSingleViewController.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 3.10: 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 3.11: Joboffer Details API (2/2)

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 3.12: Status der Axios Anfrage

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)

3.2.3 Stellenausschreibung hinzufügen

Endpoint: /joboffer/inputForm

Beschreibung

Dieser Endpoint nimmt alle Angaben zu einer einzelnen Stellenausschreibung (Joboffer) entgegen und speichert sie persistent in der Datenbank ab. Er wird vom Frontend verwendet, um eine vollständige Stellenausschreibung hinzuzufügen.

Request

- **Methode:** POST
- **URL (Frontend):** `http://localhost:8080/joboffer/inputForm`
- **Backend-Controller:** `JobofferInputViewController.java`

Response

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

Beispiel-Response

```
{  
    "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": "00000000000000",  
  
    "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

Stellausschreibung				
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	-	
Unternehmen				
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 3.13: Add Joboffer API (1/2)

Adresse				
Beschreibung	Feld	Typ	Entity	Pflicht
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	
Kontaktperson				
Beschreibung	Feld	Typ	Entity	Pflicht
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 3.14: 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 Jobangebot, Unternehmen, Kontaktperson, Anschrift sowie möglichen Terminen.

Die Daten werden als JSON an das Backend übermittelt, wo sie in mehrere Entitäten (Joboffer, Company, Address, Contact, Appointment) konvertiert und persistent gespeichert werden.

Die Schnittstelle wurde so gestaltet, dass sie sowohl neue Einträge unterstützt (leere Standardwerte) als auch bestehende Einträge lädt und zur Bearbeitung vorbereitet.

Neue Termine werden durch leere Strings als IDs als neu markiert und im Backend immer neu gespeichert.

Die Schnittstelle wird von der React-Komponente AddJobofferForm verwendet, welche einerseits bestehende Daten vorlädt (siehe Schnittstelle ‘Detaillierte Stellenausschreibung ansehen’), andererseits neue Eingaben erlaubt und anschließend über POST/PUT-Endpunkte an das Backend übermittelt.

3.2.4 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			
	companyname : string = Name der Firma			

Abbildung 3.15: Datenmodell der Appointment 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 3.16: Status der Axios Anfrage

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.

3.3 Linkssammlung

- Github: <https://github.com/SWB-WS25-Bewerbungstracker/bewerbungstracker>
- Kanban: <https://bewerbungstracker.atlassian.net/jira/software/c/projects/BT/boards/34>
- Hajtex(read-only): <https://tex.fachschaften.org/read/nsrnggsxvjt#65b78d>
- aktueller Commit: <https://github.com/SWB-WS25-Bewerbungstracker/bewerbungstracker/commit/d884d36bca6922afa91e51e4b79b44dc1235ad6c>

4 Appendix

4.1 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 *

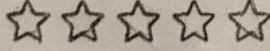
Beschreibung des Unternehmens

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 ▾

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.

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

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

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

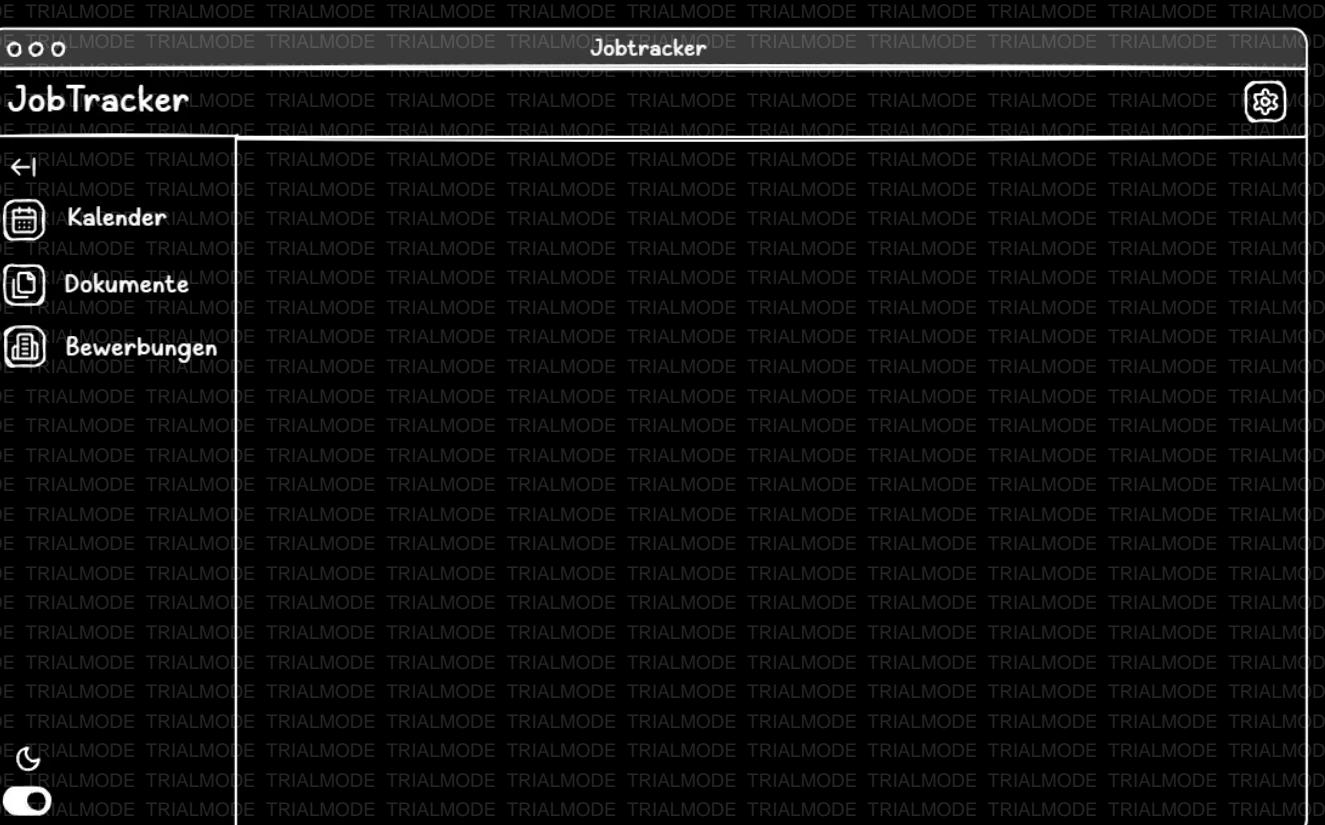
Dokumente

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

Festo

HS Esslingen

Bosch

pep.digital

Porsche

Daimler

Bewerbungen

Dokumente

Jobtracker

Dokumente

Kalender

Bewerbungen

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

Upload

4.2 Protokoll 17. Oktober 2025

Teilnehmer:

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

4.2.1 Kundengespräch

Techstack

- React
- Spring-Boot
- PostGreSQL
- Keycloak

4.2.2 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

4.3 Protokoll 24. Oktober 2025

Teilnehmer:

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

4.3.1 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 1.7)
 - 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

4.3.2 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

4.4 Protokoll 31. Oktober 2025

Teilnehmer:

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

4.4.1 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

4.4.2 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)

4.5 Protokoll 07. November 2025

Teilnehmer:

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

4.5.1 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

4.5.2 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?

4 Appendix

- 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

4.6 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

4.6.1 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

4.6.2 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

4.7 Protokoll 21. Monat 2025

Teilnehmer:

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

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

4.7.1 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)

4 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

4.7.2 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 `springboot: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

4.8 Protokoll 28. Monat 2025

Teilnehmer:

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

4.8.1 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

4.8.2 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

4.9 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

4.9.1 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)

Fragen

Sprintplan

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

Ende 14:17

4.9.2 Betreuergespräch

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

4 Appendix

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