Dokumentation für das Projekt OCCUPANCY PLANNER

Name	Matrikelnummer	GitHub
Thomas Fetter	763134	codergod1337
David Raupp	766625	darait00
Nico Epp	767294	Niggo2k
Silas Supke	765555	SilaS
Emir Mahmutovic	766200	emir1312

Kunde: Frau Sandra Wickner

Betreuer: Herr Michael Watzko

Inhaltsverzeichnis

Abbildungsverzeichnis

Vorwort

Das Unternehmen, die IT-Designers Gruppe, zieht in ein neues Bürogebäude. Der Gemeinschaftsraum soll den Mitarbeiterinnen und Mitarbeitern eine erholsame Mittagspause ermöglichen. Leider sind die Kapazitäten des Gemeinschaftsraumes sehr begrenzt. Um unnötigen Stress zu vermeiden, zur gewünschten Zeit in die Mittagspause gehen zu können, ermöglicht die Applikation OCCUPANCY PLANNER eine Platzbuchung für den Gemeinschaftsraum.

1 Requirements

1.1 Functional Requirements

1. Jeder Nutzer soll bis zu 4 Zeitslots von 15 min Länge reservieren können.
2. Die Anwendung soll sich merken, welcher Benutzer welchen Zeitslot am häufigsten reserviert hat. Dieser Zeitslot wird dann dem Benutzer durch eine farbliche Markierung angezeigt.
3. Schon vollständig reservierte Zeitslots sollen durch farbliche Markierungen als nicht mehr reservierbar angezeigt werden.
4. Jeder Mitarbeiter, Projektleiter und Teamleiter soll einen Platz reservieren können.
5. Jeder Mitarbeiter, Projektleiter und Teamleiter soll für jede Reservierung Gäste hinzufügen können.
6. Jeder Projektleiter und Teamleiter soll für ein Projektteam mehrere Plätze reservieren können.
7. Jeder Teamleiter soll für mehrere Projektteams Plätze reservieren können.
8. Bei einer Reservierung von mehr als einem Platz sollen diese, sofern möglich, zusammenhängend sein.
9. Jeder Projektleiter und Teamleiter soll bei der Reservierung für ein oder mehrere Projektteams Personen entfernen können.
10. Für eine Person soll nur einmal ein Platz reserviert werden. In allen folgenden Reservierungen des Projektteams soll diese Person farblich markiert werden und ist nicht dabei. Die erste Reservierung soll dabei immer gültig sein.
11. Jeder Projektleiter / Teamleiter soll bei der Buchung für ein / mehrere Projektteams alle Personen mit Namen sehen können.
12. Jeder Nutzer soll über das GUI den genauen reservierten Sitzplatz (auch bei Reservierung von mehrfachen Sitzplätzen) wählen können.
13. Jeder Nutzer soll einen Tag im Voraus reservieren können.

1.2 Non Functional Requirements

1. Das Logo der IT-Designers Gruppe und die Farben sollten überall präsent sein.
2. Die Anwendung muss schnell, einfach und intuitiv von Fachleuten im Bereich der IT bedienbar sein.

1.3 Optionale Requirements

1. Bei mehrfacher Reservierung eines Nutzers durch Vorgesetzte soll sich dieser selbst aussuchen dürfen, zu welcher er geht.
2. Die Anzahl der Tische und Stühle soll in den Einstellungen veränderbar sein.

3. Die Auslastung eines Zeitslots wird beim Reservierung Vorgang farblich markiert.
4. Die Anwendung soll auch am Handy gut bedienbar sein.
5. Eine Reservierung soll auch mehrere Tage im Voraus möglich sein.
6. In der ersten Anzeige der Zeitslots werden nur die vom Nutzer am häufigsten benutzten Zeitslots angezeigt und der am häufigsten benutzte wird vorausgewählt

1.4 Nachträglich ergänzte Requirements

Frage: Was ist, wenn man zweimal pro Tag eingeladen wird? -> erste Buchung gewinnt

Welche Dockerports? -> freie Auswahl

Frage: Wenn der Nutzer über eine GUI die Tische auswählt wo er sitzt, müssen wir die Anordnung der Tische nicht kennen.

Weitere Details sind den Gesprächsprotokollen im Anhang zu entnehmen.

1.5 UML Use Case Diagramm

Um die grundlegenden, von außen sichtbaren, Interaktionen der Akteure (Mitarbeiter, Teamleiter und Projektmanager) mit dem zu entwickelnden System zu visualisieren, haben wir ein Anwendungsfalldiagramm erstellt.

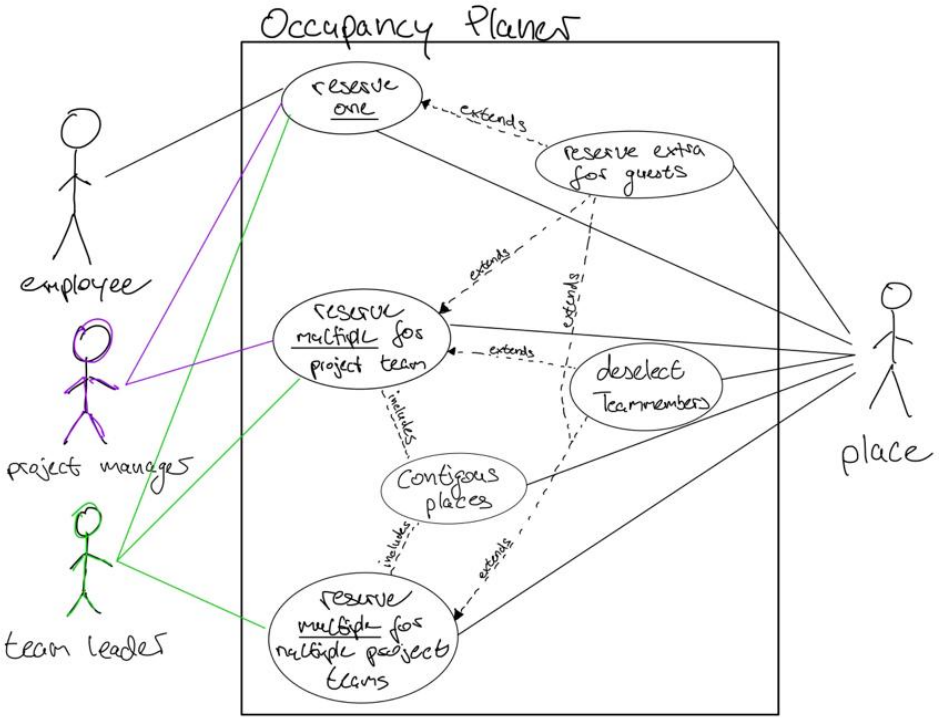

Abbildung 1 Use Case Skizze, um alle Anwendungsfälle darzustellen, erstellt mit Good Notes

2 User Interface

2.1 Vorüberlegungen

Kriterien für die Usability wurden hauptsächlich durch die Non Functional Requirements bestimmt. Generell standen Einfachheit, Effizienz, Konsistenz und Fehlertoleranz im Vordergrund. Design und Farben sind dem Corporate Design von IT-Designer Gruppe entnommen.

2.2 Überprüfung

Die Überprüfung erfolgte durch eine Abnahme der Kundin. Des weiteren werden zu einem späteren Zeitpunkt Benutzertests von realen Benutzern am Prototypen durchgeführt werden. Abschließend führt der Betreuer eine Usability-Inspektion durch.

2.3 Mockups in Figma

Für den generellen Aufbau wurde die Webseite von der IT-Designers Gruppe untersucht und als Inspirationsquelle benutzt. Um die Requirements abbilden zu können, sind für eine Reservierung zwingend die Dauer und der Ort anzugeben. Hierfür wurden zwei mögliche Designs erarbeitet und dem Kunden zur Auswahl freigegeben. Der Kunde hat sich für die Version Abbildung 2 entschieden.

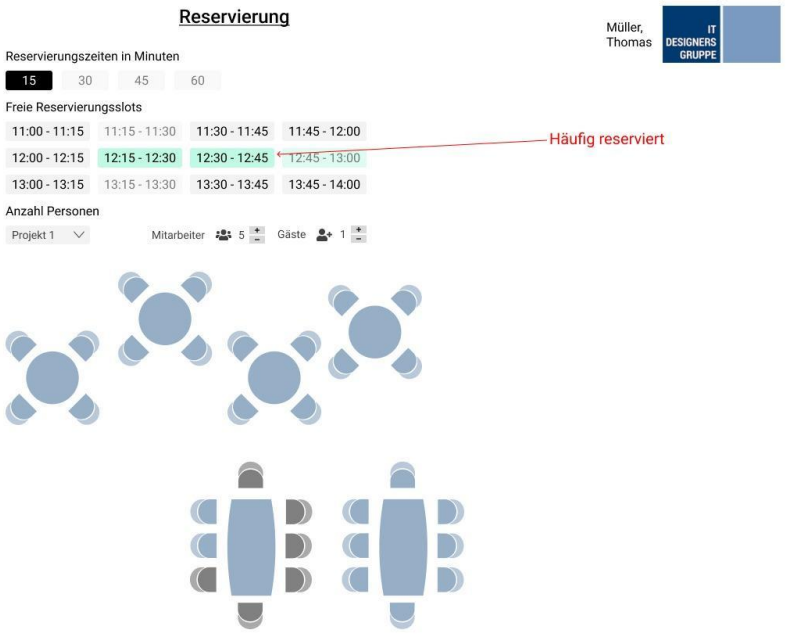

Abbildung 2 Mockup in Figma das eine Reservierung skizziert

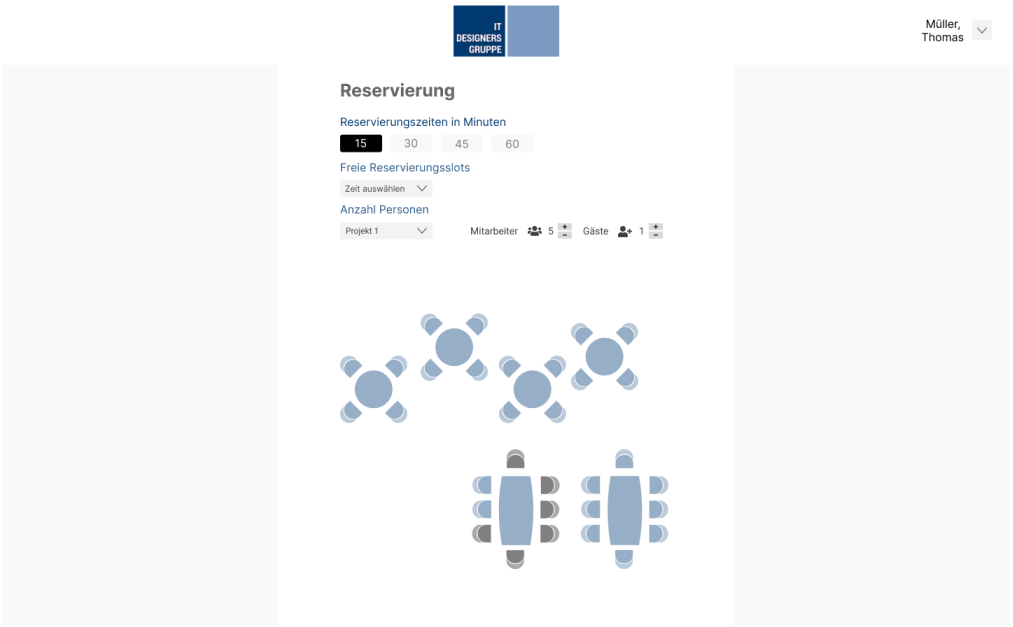

Abbildung 3 Mockup in Figma das eine Reservierung skizziert in einer alternativen Version

Die oben abgebildete alternativ Version (Abbildung 3) wurde vom Kunden abgelehnt. Im Hinblick auf die Bedienung am Smartphone sind Dropdown Menüs zu vermeiden.

Für die Zugriffsrechte ist eine Benutzerverwaltung erforderlich, darum wird

ein Login benötigt. Das Layout orientiert sich am üblichen Standard im Internet.

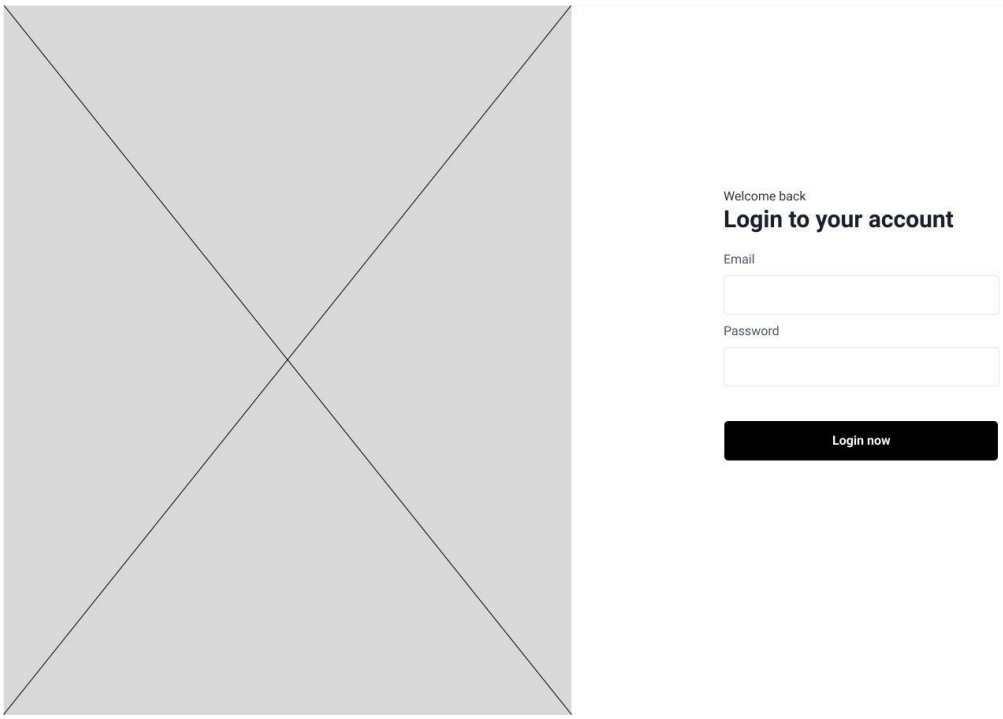

Abbildung 4 Mockup in Figma das einen möglichen Login darstellt

2.4 Personas

Für einen besseren Überblick der Anforderungen und des benötigten Funktionsumfangs wurden noch drei Personas erstellt. Die Bilder sind mit https://this-person-does-not-exist.com/de erstellt.

2.4.1 Lena Schmid

Alter: 35
Beruf: Softwareentwicklerin

Lena mag schnelles Essen, mit Kollegen zusammensitzen, leckeres Mittagessen und orientiert sich bei Buchungen zeitlich an ihren Kolleginnen und Kollegen.

Am liebsten ist ihr eine intuitive und schnelle Lösung, um einen Sitzplatz zu buchen.

Relevante Vorkenntnisse von ihr sind:

- Sie ist Softwareentwicklerin, daher kann man davon ausgehen, dass Sie in aktuellen Betriebssystemen schon einige Erfahrungen hat und somit voreingenommen ist, wie ein Buchungssystem zu bedienen sein sollte.
- Kennt Standard GUI Elemente wie z.B.: Buttons, Sliders, Dropdown-Menüs etc.

2.4.2 Marleen Polt

Alter: 26

Beruf: Softwareentwicklerin

Marleen kennt sich mit den meisten modernen Computern und digitalen Geräten gut aus, ist ungeduldig, toleriert keine übermäßig komplizierten Systeme und Protokolle.

Sie mag einfache, schnelle und vorhersehbare Anwendungen und ist frustriert, wenn Dinge nicht nach Plan laufen oder zu lange dauern.

Marleen ist ein Gewohnheitsmensch und möchte immer um 12.00 pünktlich Essen

2.4.3 Andreas Müller

Alter: 38

Beruf: Softwareentwickler und Teamleiter

Andreas ist verantwortlich für die Leitung von vier Projekten mit unterschiedlichen Anforderungen und Zielsetzungen. Er koordiniert 16 Mitarbeiter, die sowohl intern als auch extern auf verschiedenen Ebenen arbeiten. Er stellt sicher, dass jedes Projektteam über die erforderlichen Ressourcen verfügt und die Projektziele innerhalb des festgelegten Zeitrahmens und Budgets erreicht werden. Er hat jeden Tag sehr viel zu tun und sitzt auch häufiger bis um 13:30 Uhr in Meetings, weshalb er meistens sehr spät Mittagessen geht.

2.4.4 Tom Bauer

Alter: 58
Beruf: Softwareentwickler

Tom versucht seinen Tag durch Routinen zu strukturieren, seitdem der neue Aufenthaltsraum ein Buchungssystem hat. Nach Möglichkeit geht er jeden Tag um 13:00 essen und würde diese Uhrzeit gerne lange im Voraus buchen. Trotzdem braucht er die Möglichkeit, diese Buchung noch kurzfristig zu ändern, falls Meetings länger dauern oder ihm andere wichtige Dinge dazwischenkommen. Seine Reservierung kann Tom sowohl an seinem von der Firma gestellten PC machen oder auch an seinem Smartphone.

2.5 User Stories

Als Mitarbeiter möchte ich per Anwendung einen Platz zum Mittagessen reservieren können, um das Reservieren z. B. mit Jacken zu vermeiden.

Als Mitarbeiter möchte ich mich bei der Anwendung anmelden können, um meine Daten jederzeit abrufen zu können.

Als Mitarbeiter möchte ich auch für Gäste einen Platz reservieren können, um gemeinsam mittags zu essen.

Als Mitarbeiter möchte ich bei der Anwendung zeitlich möglichst genau sehen können, welche Plätze schon reserviert sind und welche nicht, damit bei der Buchung kein Problem auftritt.

Als Mitarbeiter möchte ich Zeitslots basierend auf meinen vorherigen Buchungen vorgeschlagen bekommen, damit der Buchungsvorgang schnell erfolgen kann.

Als Mitarbeiter möchte ich meinen Platz für eine ausreichende Zeit buchen können, damit ich essen kann, ohne unter Zeitdruck zu stehen.

Als Mitarbeiter möchte ich, wenn ich für mehrere Personen buche, möglichst an zusammenhängenden Plätzen sitzen können.

Als Mitarbeiter möchte ich, dass nur meine erste Buchung für einen Zeitslot gilt, damit keine Doppelbuchungen und somit Missverständnisse entstehen.

Als Mitarbeiter möchte ich schon einen Tag vorher buchen können, um mich darauf einzustellen und mir einen passenden Platz zu sichern.

Als Mitarbeiter möchte ich auswählen können, wo ich sitzen werde, um einen für mich passenden Platz zu haben.

Als Teamleiter möchte ich für ein oder mehrere Projektteams reservieren können, damit man sich als Team austauschen kann.

Als Teamleiter möchte ich bei der Anwendung Personen von einem oder mehreren Projektteams von der Reservierung entfernen können.

Als Projektleiter, Teamleiter, Mitarbeiter möchte ich sehen, welche Personen schon anderweitig gebucht sind, damit ich sehe, wer am geplanten Tag zur Verfügung steht.

Als Teamleiter / Projektleiter möchte ich sehen, welche Person welchen Platz reserviert hat.

Als Mitarbeiter möchte ich ein intuitives GUI, welches eine hohe Usability bietet, um schnell und einfach navigieren zu können.

Als Mitarbeiter möchte ich ein intuitives GUI, damit ich schnell und effizient buchen kann.

2.6 Prototyp

In React wurden Prototypen erstellt, die noch nicht mit dem Backend kommunizieren, aber das Design und die Funktionalität sehr anschaulich skizzieren und sich an den Mockups orientieren.

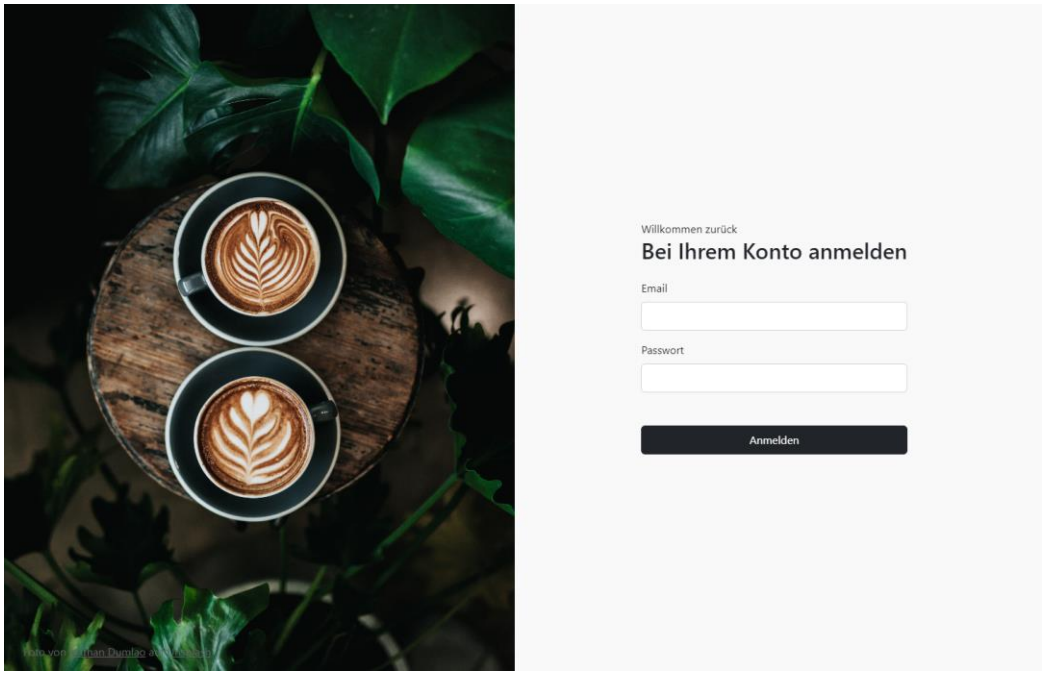

Abbildung 5 Login Prototyp, der im späteren Verlauf durch den Keycloak Login ersetzt wird

In der unteren Abbildung 6 wird die Reservierung mit der berechneten favorisierten Gewohnheit vorgeschlagen, wie es in den Requirements[1] gefordert ist.

1.1. 2. Requirement[1]

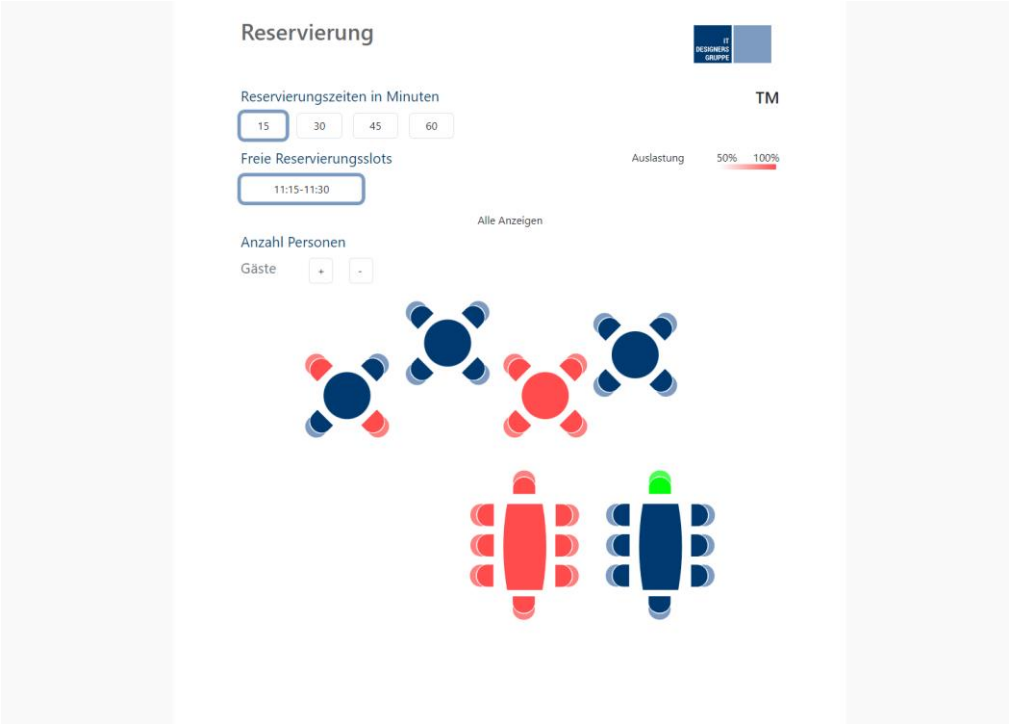

Abbildung 6 zeigt die Reservierungsseite mit Gewohnheitspreset. Darstellung in React gerendert

Um eine freie Reservierung zu ermöglichen, gibt es eine Ansicht (Abbildung 7), bei der alle verfügbaren Timeslots mit jeweiliger Buchungsauslastung angezeigt werden.

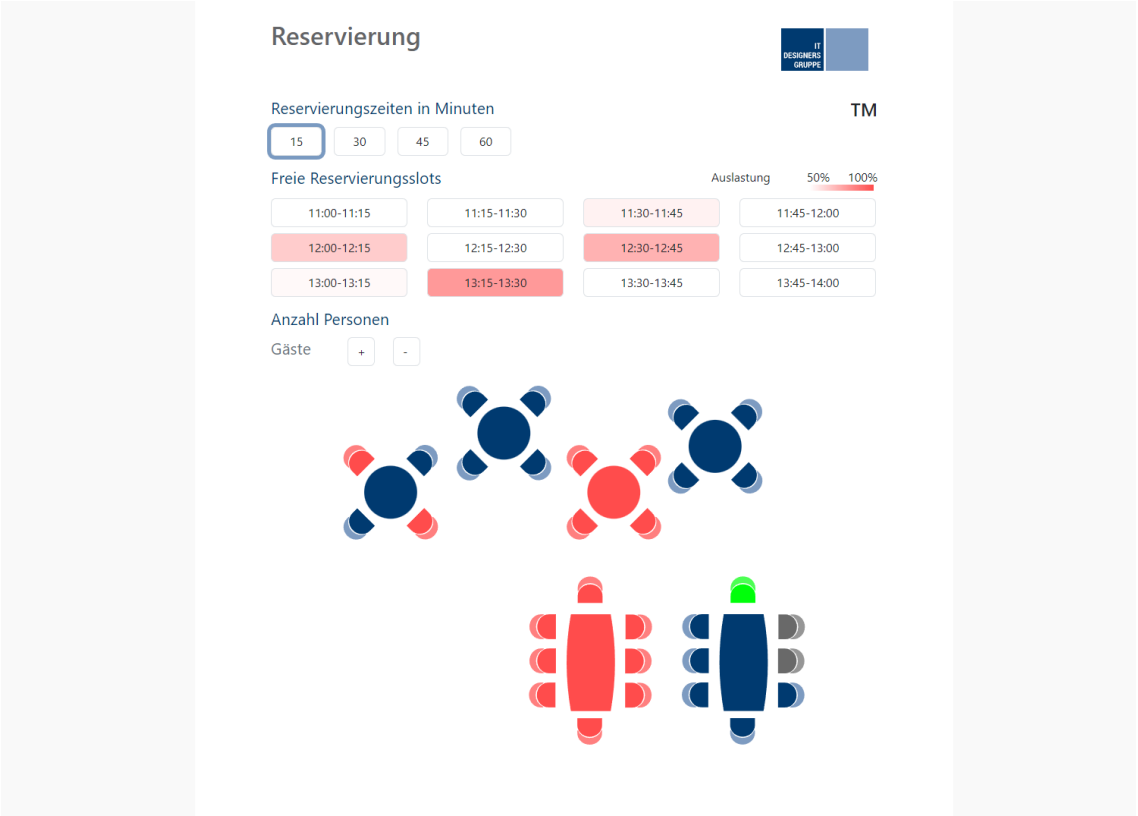

Abbildung 7 Reservierungsseite in React gerendert im erweiterten Modus: mit frei wählbarer Zeit

3 Architektur/ Technologie

3.1 High-Level Architektur Diagramm

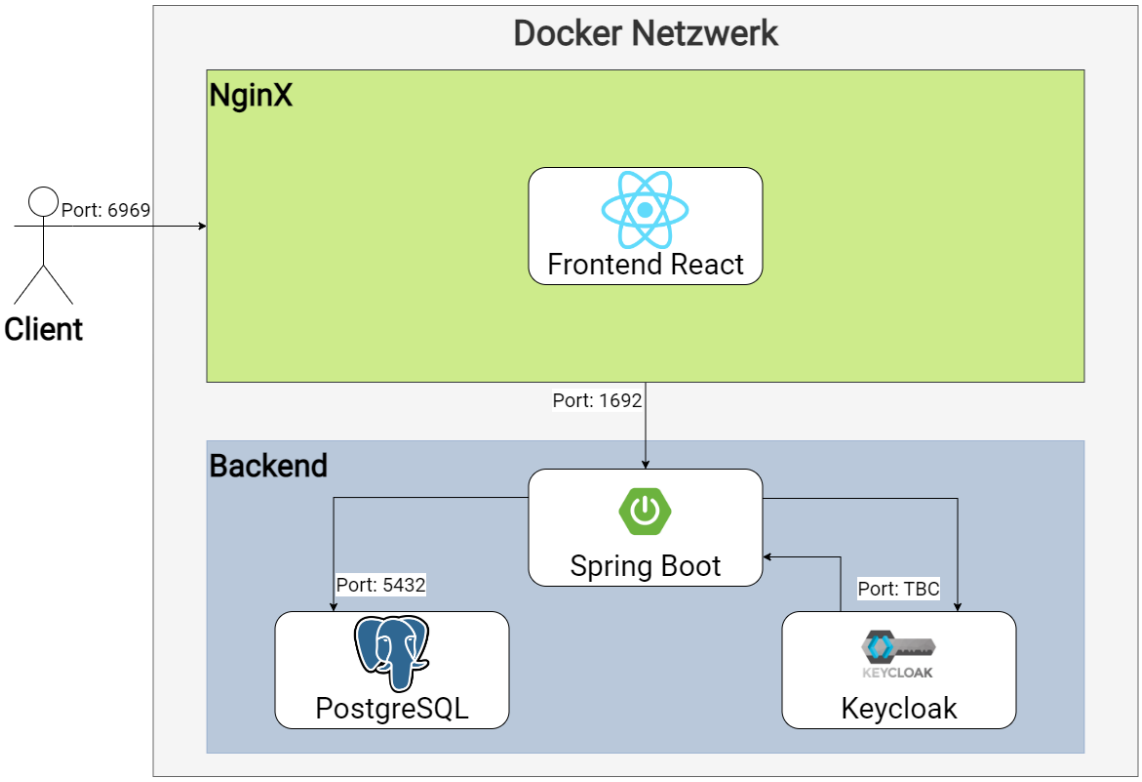

Abbildung 8 zeigt die geplante High-Level Architektur

Der Client verbindet sich zum NGINX Server auf dem Port 6969 und bekommt durch diesen das Frontend angezeigt. Der NGINX Server leitet intern die Anfragen des Clients an Spring Boot weiter, welches je nach Anfrage mit Keycloak und PostgreSQL kommuniziert.

3.2 Softwarearchitektur

3.2.1 UML-Verteilungsdiagramm

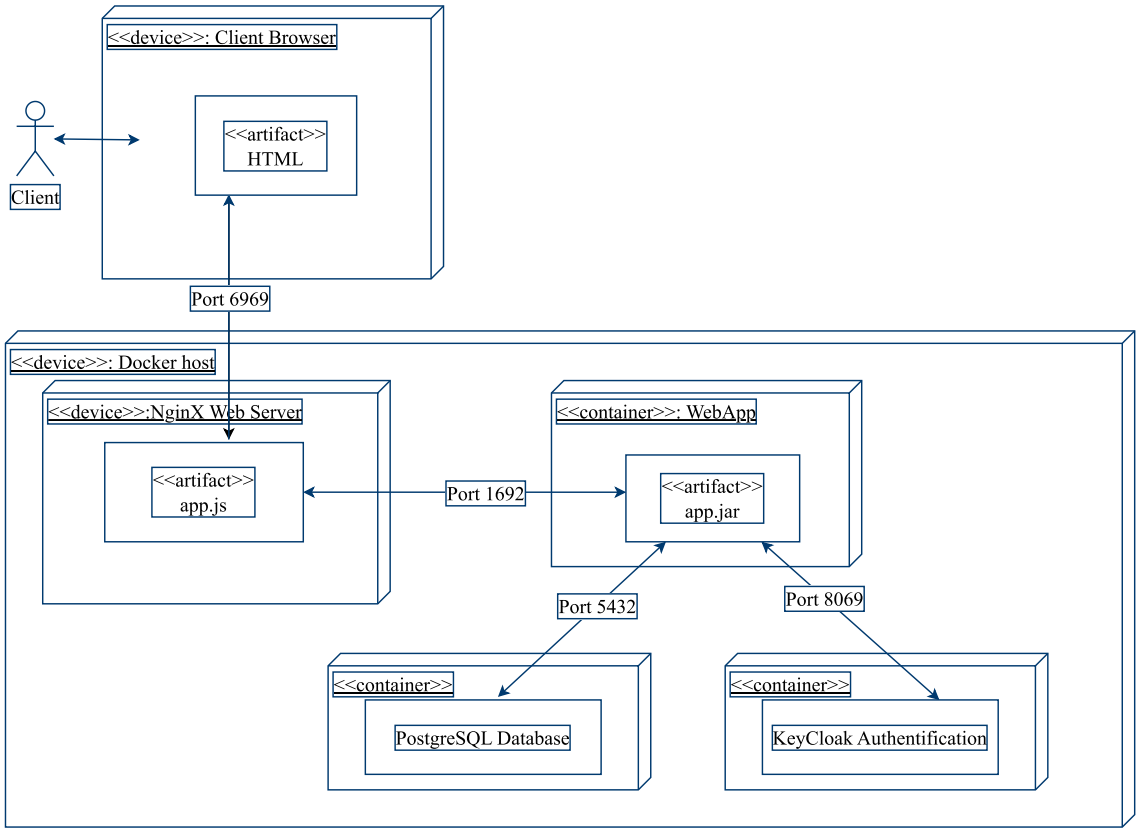

Abbildung 9 Verteilungsdiagramm des Occupanyplanners

In Abbildung 9 werden die verschiedenen Komponenten dargestellt. Jede Komponente soll in einem Docker Container befinden.

3.2.2 UML-Verteilungsdiagramm

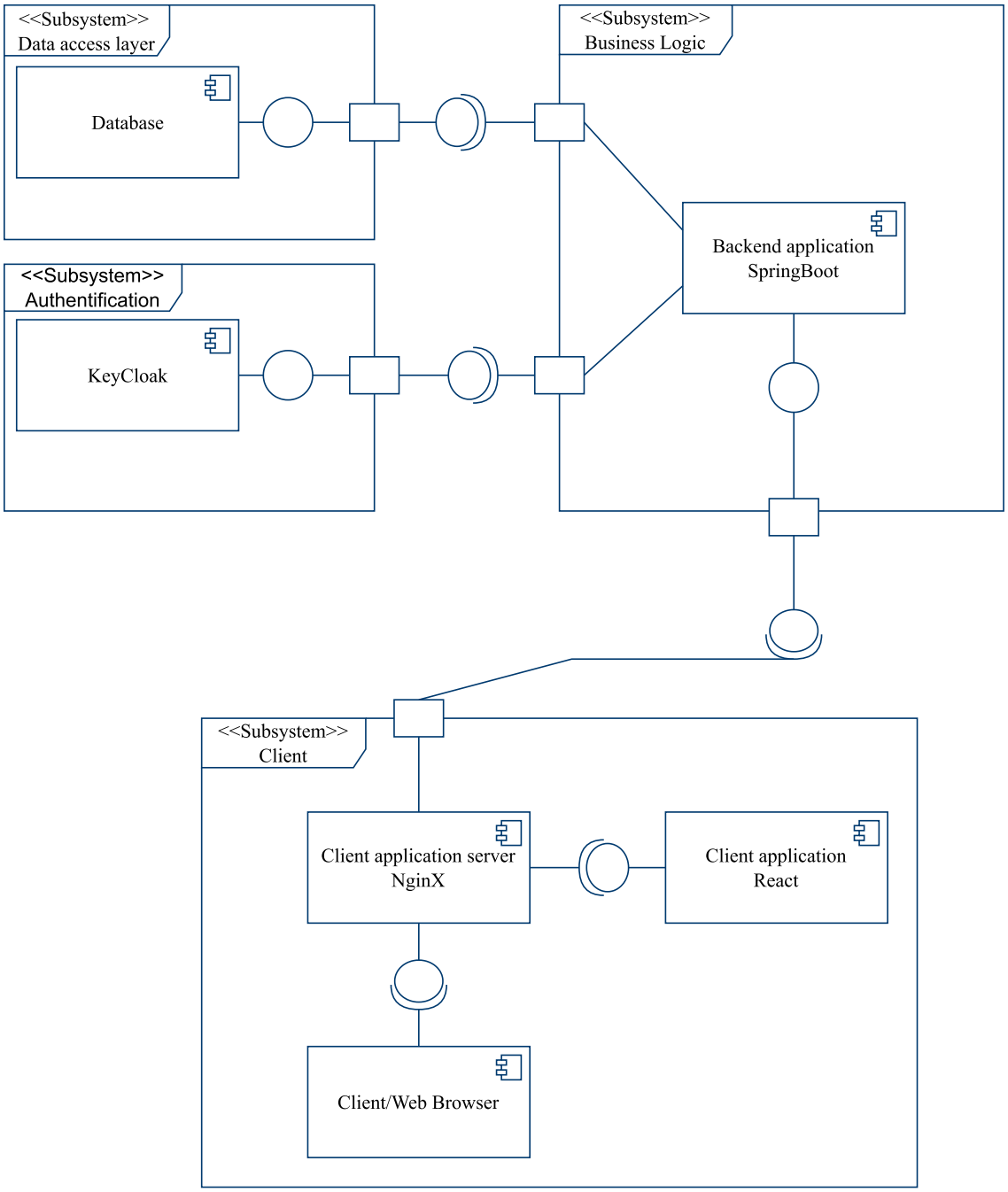

Abbildung 10 Komponentendiagramm des Occupanyplanners

In Abbildung 10 werden visuell die Komponenten des Softwaresystems dargestellt.

3.3 GitHub

Für die Versionsverwaltung wurde GitHub ausgewählt. Ebenso werden Open Issues für einzelne Tasks verwendet. Das Repository ist auf **https://github.com/Occupancy-Planner-Team1** zu finden.

3.4 Verwendete IDEs

- Für das Frontend wird Visual Studio Code benutzt, mit den folgenden Addons: Docker, Github Pull Requests and Issues, React Extension Pack und IntelliCode.
- Für das Backend wird IntelliJ Community benutzt

3.5 Dokumentations Tool Greenshot

Zum Erstellen von Screenshots wurde die Software Greenshot verwendet. So ist ein Referenzieren auf Teilbereiche eines Screenshots einfacher und übersichtlicher

https://getgreenshot.org/downloads

3.6 Postman

Postman wird verwendet, um die Kommunikation mit dem Backend zu testen. So können Frontendanfragen erstellt und die Backendantworten überprüft werden. https://www.postman.com/downloads

3.7 DBeaver

Mit DBeaver wurden die Tabellen der Datenbank erstellt. Ebenso wurde die Verbindung zur Datenbank damit überprüft. Download: https://dbeaver.io/download/

3.8 PostgreSQL (als Docker Container)

Als Datenbank wird die PostgreSQL Datenbank verwendet. Um diese zu installieren sind folgende Schritte zu befolgen:

Im Terminal:

```
docker pull postgres
docker run --name OCCUPANCYP -p 5432:5432 -e POSTGRES_PASSWORD=123 -d post-
gres:latest
```

So wird die Standard-Datenbank Postgres erstellt mit dem Passwort 123 auf dem Standard Port 5432.

Aufbau der Datenbank

```
UNIQUE (datum, member-id, stuhl, timeslot )
reservierung { id, datum, timeslot, creatorid, mitarbeiterid, buchungsid, stuhlid
}
stuhl {id, tisch, posx, posy }
```

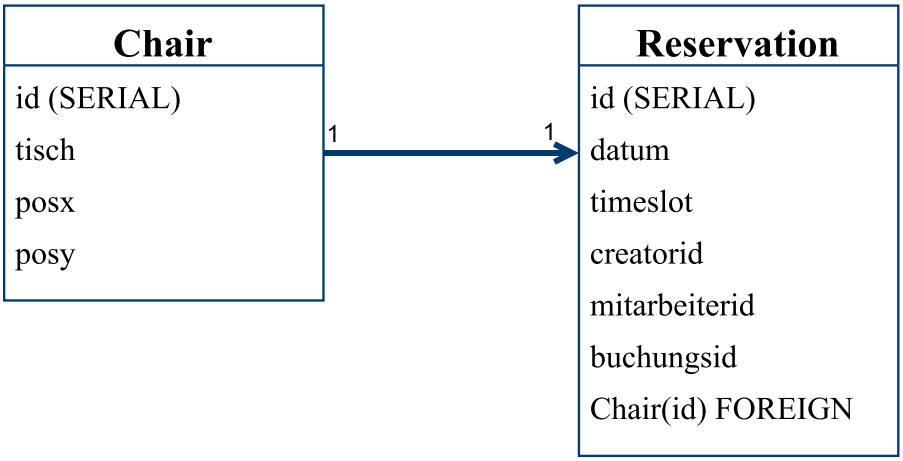

Abbildung 11 ERM verdeutlicht den Aufbau der Datenbank

Pro Reservierung wird für jeden reservierten Stuhl, ein Stuhl aus der *Chair* Tabelle mit einer Zeile in der *Reservation* Tabelle verknüpft. Sollte die Reservierung sich über einen Zeitslot ausdehnen, kommt für jeden weiteren Timeslot ein weiterer Satz an Einträgen hinzu.

3.9 Frontend: ReactJS

Als Frontend wurde, wie auch in der Aufgabenstellung vorgeschlagen ReactJS gewählt. Voraussetzung ist NodeJS (https://nodejs.org/en) LTS Version. Nach der Installation kann im Terminal die Funktionalität getestet werden:

```
node -v
-> v18.15.0
npm -v
-> 9.5.0
```

Für einen Überblick der Funktionalität wurden einige Videos bzw. Anleitungen[2] hergenommen.

Weitere Befehle, die im Terminal auszuführen sind:

```
npm install -g npx
```

Um ein React Projekt zu erstellen: Im Terminal in den übergeordneten Ordner für ein neues Projekt und mit:

```
npx create-react-app projekt123
```

ein neues Projekt erstellen. Danach muss der React Server gestartet werden.

```
cd projekt123
npm start
```

[2] → https://www.youtube.com/watch?v=WMf5UEZLXyM&list=PLNmsVeXQZj7oi_Q4whC28Yp12l1I-hauk&index=3
→ https://www.youtube.com/watch?v=zJxJerQtUdk

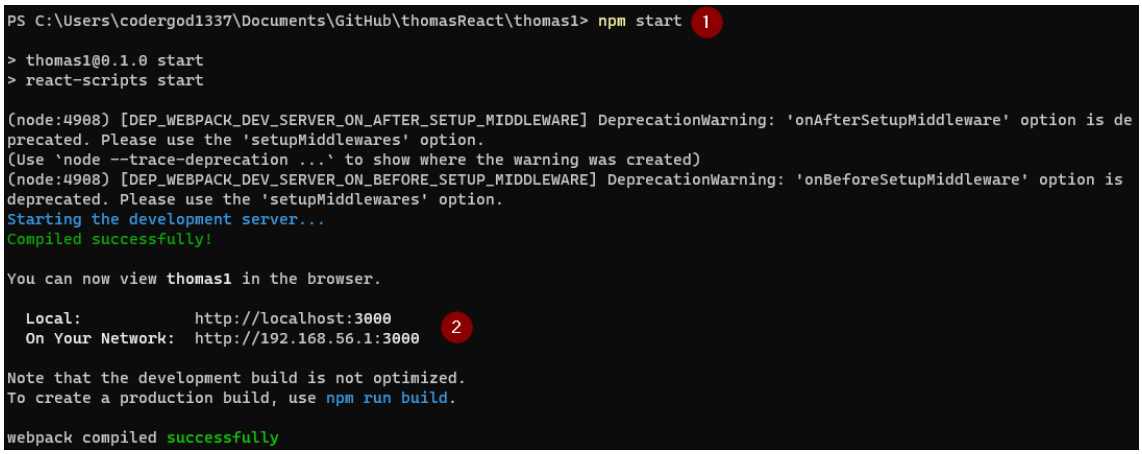

Abbildung 12 zeigt den Status des gestarteten Servers. 1: Kommando, 2: Serveradresse für die Entwicklung

Der Port kann mit einer neuen Datei ".env" mit dem Inhalt:

```
PORT=6969
```

festgelegt werden.

3.10 Backend: Spring Boot

Für das Backend wurde, wie beim Frontend die vorgeschlagene Technologie von IT DESIGNERS GROUP verwendet - Spring Boot. Um ein Startpaket zu erstellen, wurde die URL https://start.spring.io benutzt. Maven, Spring Boot v 3.0.5 und Java Version 17.

Settings im Backend für die Verbindung zu der Datenbank sind in Resources/Application.properties:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
spring.datasource.username=postgres
spring.datasource.password=123
spring.datasource.hikari.connectionTimeout=20000
spring.datasource.hikari.maximumPoolSize=5
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQL95Dialect
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Die vollständige Dokumentation für Maven findet man unter: https://mvnrepository.com

3.11 NGINX Webserver

Die Übergabe der Software erfolgt in einem Docker-Compose File. Um den React Build verwenden zu können wird ein Webserver benötigt.

Anleitung Reverse Proxy:

https://medium.com/bb-tutorials-and-thoughts/react-how-to-proxy-to-backend-server-5588a9e0347

3.12 Authentifikation – KeyCloak und OAuth 2.0

Für die Authentifikation wird KeyCloak als Docker Container genutzt und Spring Boot um die Dependencys:

- spring-boot-starter-oauth2-client
- spring-boot-starter-oauth2-resource-server
- spring-boot-starter-security

erweitert. Finale Anleitung wird zum nächsten Meilenstein nachgereicht.

```
docker run -d -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEY-
CLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:21.0.1 start-dev
```

3.13 REST API – Schnittstellentechnologie

Es werden die URIs definiert:

Methode	Code	Beschreibung
GET	`api/auth/res-day/{date}`	Gibt alle Buchungen an einem Tag zurück
GET	`/api/auth/last-change`	Gibt Id des letzten Buchungseintrags zurück
POST	`/api/auth/res/`	Trägt neue Buchung mit Liste an Reservierungen ein
DELETE	`/api/auth/res/del-booking/{bookingid}`	Löscht existierende Buchung

4 Projektmanagement

Als Arbeitsmodell wird Scrum genutzt. Regelmäßige Treffen ca. zwei Mal pro Woche, in der jeder Teilnehmer seine bearbeiteten Aufgaben des vorangegangenen Treffens vorstellt. Die Aufgaben entsprechen den GitHub Issues (Siehe Kapitel 3.2).

4.1 Rollen/ Hauptverantwortliche

Rolle	Hauptverantwortlicher
Frontend	David
Backend	Emir und Silas
Datenbank	Silas
Protokolle	Nico
Kundenkontakt	Thomas
Dokumentation	Thomas
Authentifikation	Thomas

4.2 Aufwandsschätzungen

Berechnung der Gesamtzeit mit fünf Studenten:
20 Stunden pro Woche * 15 Wochen = 300h pro Student, da jeder einzelne Student in allen Teilbereichen Bescheid wissen muss, wird 50% der Zeit benötigt, die der Hauptverantwortliche benötigt.

Gebiet	Zeit in h
Einarbeitung in die Technologien	60
Frontend Design	30
Frontend Technik/ Logik	80
Backend Datenbankschnittstelle	40
Backend REST	120
Authentifikation	50
Docker-Compose	40
Datenbank Layout	30

Besprechungen/ Regelmeetings	20
Seminare	16
Einführung + Präsentation	10
Testen	10
Gesamt	506

5 Anhang

5.1 Spring Dependencies

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.26</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-oauth2-client</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
    </dependency>
</dependencies>
```

5.2 Kunden Gesprächsprotokolle

5.2.1 Kundenprotokoll vom 30.03.2023

• Mitarbeiter haben verschiedene Projekte, an denen sie arbeiten → Rückmeldung von Sandra über durchschnittliche Teamgröße

• Wenn ein Mitarbeiter schon eine Buchung hatte nicht mehr für spätere Buchungen buchbar

• Bei Doppelbuchung Mitarbeiter darf Buchung aussuchen (Optional)

• Projektleiter sieht Mitwirkende mit Namen

• Mitarbeiter, welche schon gebucht wurden sollen ausgegraut sein und Text „nicht verfügbar“ (Optionale Funktion)

• Es zählt die erste Buchung

• Durchschnittliche Teamgröße noch unbekannt → wird nachgefragt

• Keine festgelegte Docker Ports → wird nochmal nachgefragt

• Mitarbeitergruppen sollen Algorithmus verwenden (wäre ausreichend, wenn man es aussuchen kann)

• Mitarbeiter hat konkrete Platzwahl

• Anzahl der Tische und Anzahl der Stühle einstellbar (Optional)

• Falls Tisch voll ist Nachbartisch nehmen

• Nachbartische berechnen, herausfinden

• Farbliche Darstellung der Auslastung (Optional)

• Responsive Design / Handyoptimierung (Optional)

• Projekte, Rollen & Gruppen im KeyCloak (Optional)

• Teamleiter als Administrator oder extra Administrator

• Mitarbeiter, Projektleiter & Teamleiter kann einen zusätzlichen Gast reservieren

• heutigen (priorisiert) & morgigen Tag im Voraus reservierbar

• Mehrere Tage im Voraus (Optional)

• Dokumentation am Ende an den Kunden senden

5.2.2 Kundenprotokoll vom 06.04.2023

In der ersten Anzeige der Zeitslots werden nur die vom Nutzer am häufigsten benutzten Zeitslot(s) angezeigt und der am häufigsten benutzte wird vorausgewählt.

Team hat meistens zwischen 15-20 Mitarbeitern, im Durchschnitt 18

Im Durchschnitt ist die Projektgröße 4 Mitarbeiter, also 4-5 Projekte pro Teamleiter

Ein Projekt hat 2-10 Mitarbeiter

Docker Ports sind frei wählbar

Personas erstellen bis zum 13.04.

Kundenprotokoll 13.04.2023

Besprechung der Personas

keine konkreten Änderungswünsche an der Dokumentation.

Vorstellung der Implementierung in React

Corporate Design wird uns zur Verfügung gestellt

Fließtext für Bilder in der Dokumentation ergänzt

in der nächsten Version der Dokumentation Überschriften nummerieren

nähere Erklärung zu Mockups

Neues Design für Deckblatt Dokumentation

Requirements in „soll“ umformatieren

5.2.3 Kundenprotokoll 27.04.2023

Platzwahl: den eigenen Platz wählen per Klick auf Stuhl - restlichen Plätze automatisch, aber auch veränderbar

absolute Reservierkettenlimit = ganzer Raum

opt. ganzer Raum nur zwischen 11-12 und 13-14 Uhr reservierbar

Dummydaten in Keycloak: ca. 60 Einträge