

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

GitHub Repository:

<https://github.com/Occupancy-Planner-Team1/Occupancy-Planner>

Inhaltsverzeichnis

Inhaltsverzeichnis	2
Zielgruppe / Problemstellung.....	3
Requirements	3
High-Level Architektur Diagramm	5
UI Entwürfe	6
UML Use Case Diagramm.....	6
Mockups in Figma	7
Personas	9
Prototyp.....	11
Architektur/ Technologie	13
GitHub.....	13
Dokumentations Tool Greenshot	13
Genutzte IDEs	13
Postman	14
DBeaver	14
Frontend: React.....	15
PostgreSQL als Docker Container	14
Backend: Spring Boot.....	16
NGINX Webserver.....	17
Authentifikation - KeyCloak (not yet).....	17
Projektmanagement	18
Rollen/ Hauptverantwortlich:.....	18
Aufwandsschätzungen	18
Anhang	19
Noch nicht benutze Parts.....	19
Nginx	19
Dependencies	20
Protokolle	21

Zielgruppe / Problemstellung

Das Unternehmen IT-Designers Gruppe hat derzeit 150 Mitarbeiter. Davon arbeiten ca. 60% im Homeoffice. Die restlichen 40% verbringen ihre Mittagspause im Gemeinschaftsraum, welcher zur Zeit 32 Sitzgelegenheiten bietet. Um einem IT-Unternehmen gerecht zu werden, soll nun eine Raumbuchung implementiert werden, die sicherstellt, dass jeder der rund 60 Mitarbeiter einen Sitzplatz für seine Mittagspause erhält. Der Funktionsumfang ist den Requirements zu entnehmen.

Requirements

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.
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.

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.

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.

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.

High-Level Architektur Diagramm

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.

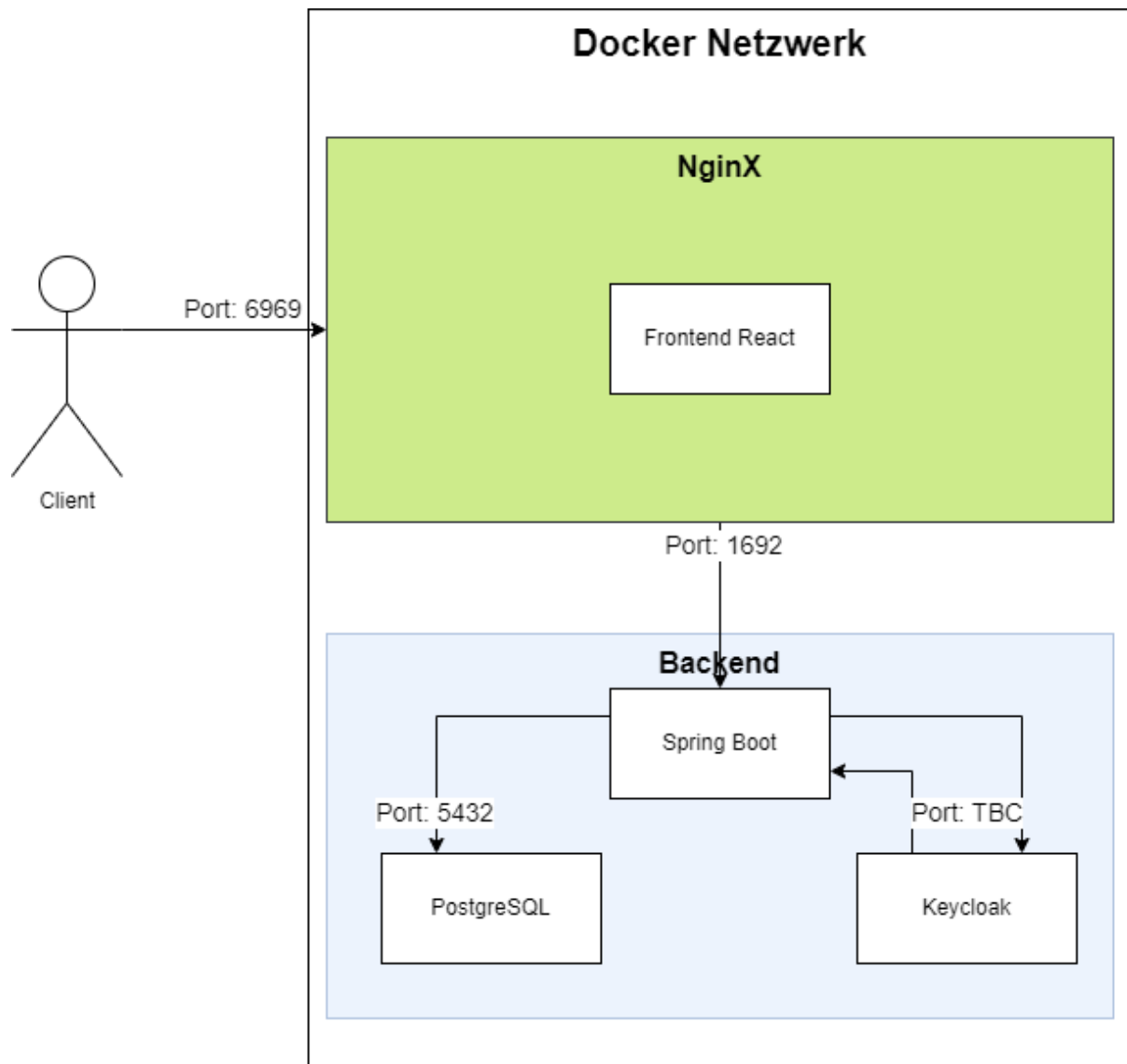


Figure 1: High-level Architektur erstellt mit diagrams.net

UI Entwürfe

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.

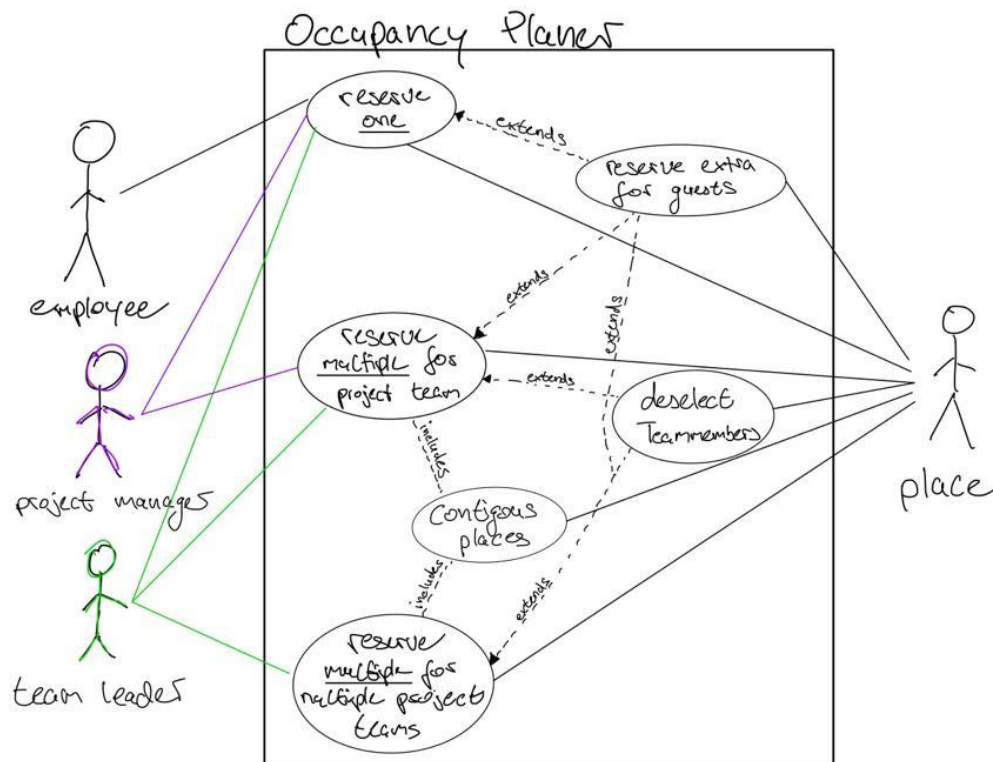


Figure 2: Use Case Diagramm welche Fälle können auftreten

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 Figure 3 entschieden.

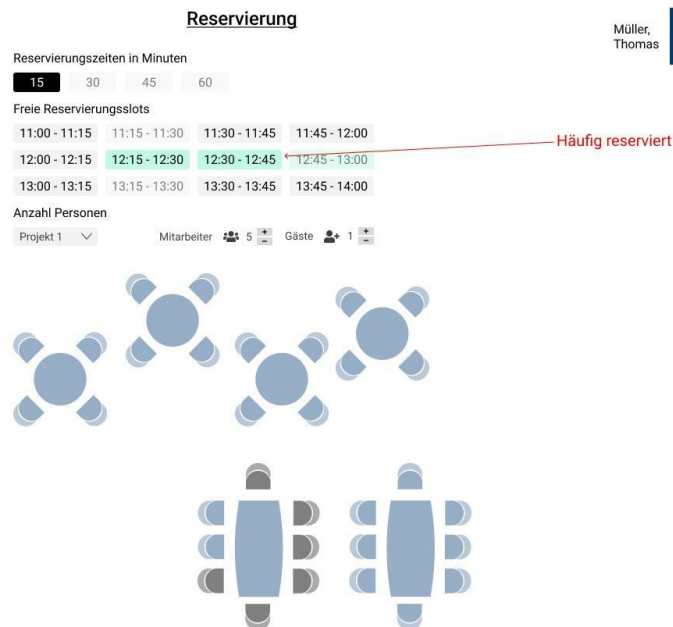


Figure 3: Mockup in Figma das eine Reservierung skizziert

Alternative 1 (Figure 4) wurde vom Kunden abgelehnt, da es eine geringere User Experience bietet und der Dropdown Button nicht so komfortabel ist.

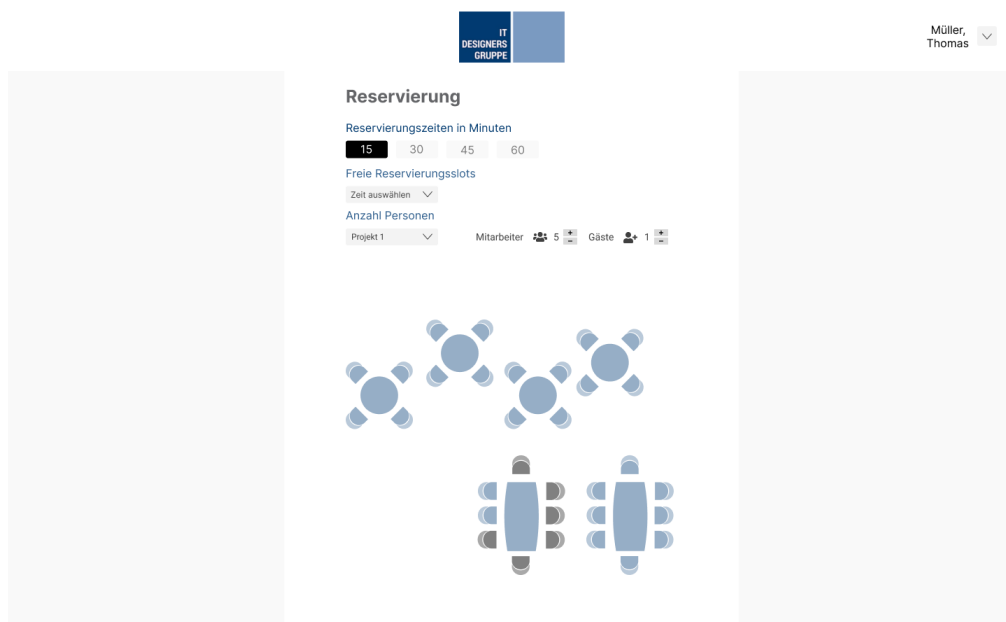
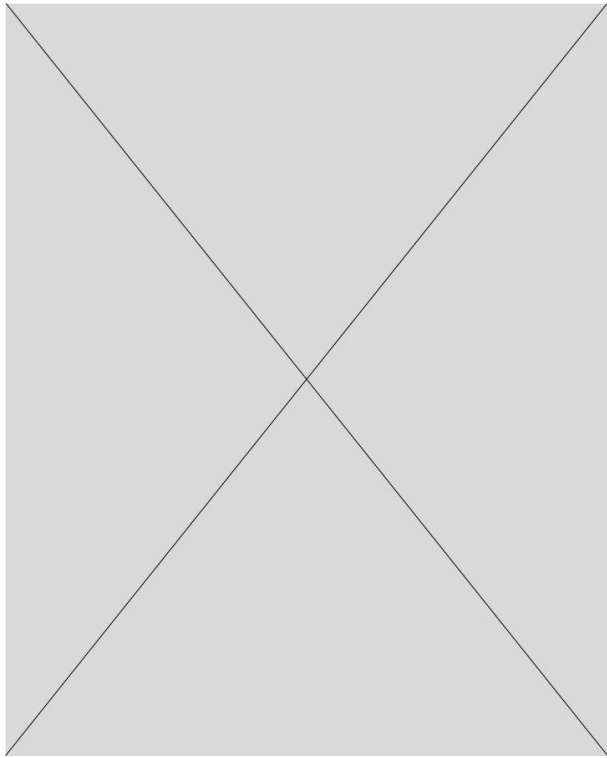


Figure 4: Mockup in Figma das eine Reservierung skizziert Alternative 1

Für die Zugriffsrechte ist eine Benutzerverwaltung erforderlich, darum wird ein Login benötigt. Das Layout orientiert sich am üblichen Standard im Internet.



Welcome back

Login to your account

Email

Password

Login now

Figure 5: Mockup in Figma das den Login verdeutlicht

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.

1. Lena Schmidt:

Alter: 35

Name: Lena Schmidt

Beruf: Softwareentwickler

Probleme:

- Sie findet oft keinen Sitzplatz in der Cafeteria

Bedürfnisse:

Lena möchte...

- schnelles Essen
- mit Kollegen sitzen
- leckeres Mittagessen
- orientiert bei der Buchung zeitlich an ihren Kollegen und Kolleginnen
- eine intuitive Lösung um einen Sitzplatz zu buchen
- eine schnelle Lösung



Für das System relevante Vorkenntnisse:

- Sie ist Softwareentwickler, 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. Marleen Polt

Marleen Polt, 26, Softwareentwicklerin bei IT-Designers Gruppe

- Kennt sich mit den meisten modernen Computern und digitalen Geräten gut aus.
- Ist ungeduldig, toleriert keine übermäßig komplizierten Systeme, Protokolle.
- Mag einfache, schnelle und vorhersehbare Anwendungen.
- Ist frustriert, wenn Dinge nicht nach Plan laufen oder zu lange brauchen.
- Gewohnheitsmensch und möchte immer um 12.00 pünktlich Essen



3. Andreas Müller

28 Jahre, Teamleiter bei der IT Designers Gruppe

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.



4. Tom Bauer

Tom ist ein 48 Jahre alter Softwareentwickler bei der IT Designers Gruppe. Seit der neue Aufenthaltsraum ein Buchungssystem hat, versucht er seinen Tag durch Routinen zu strukturieren. 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 durchführen oder auch an seinem Handy.



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.

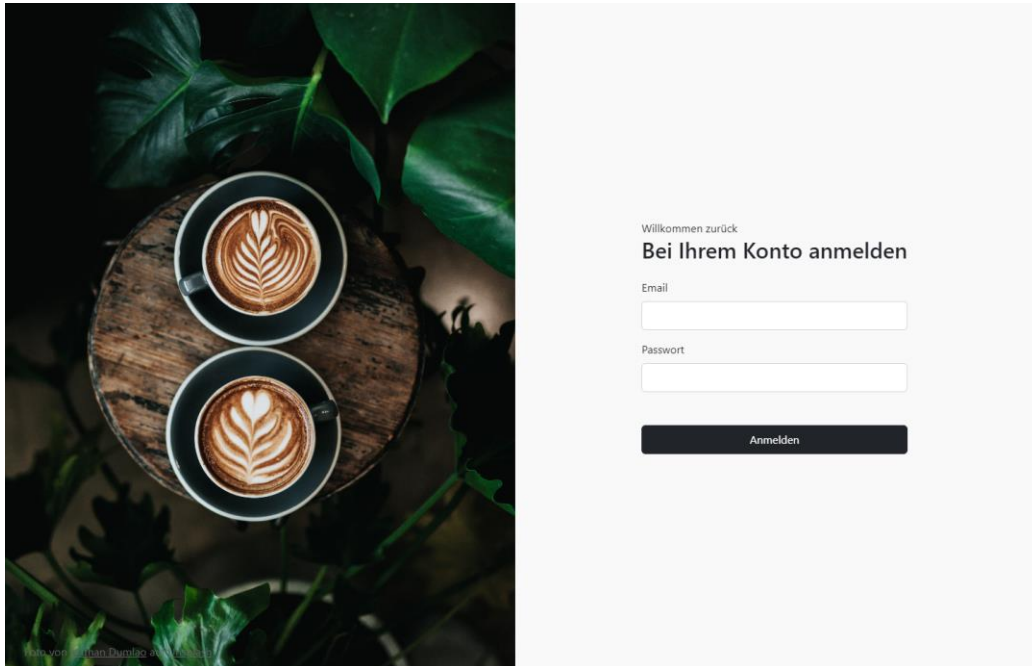


Figure 6: Login Seite in React gerendert

In Figure 7 wird die Reservierung mit den favorisierten Gewohnheiten vorgeschlagen, wie es in den Requirements gefordert ist.

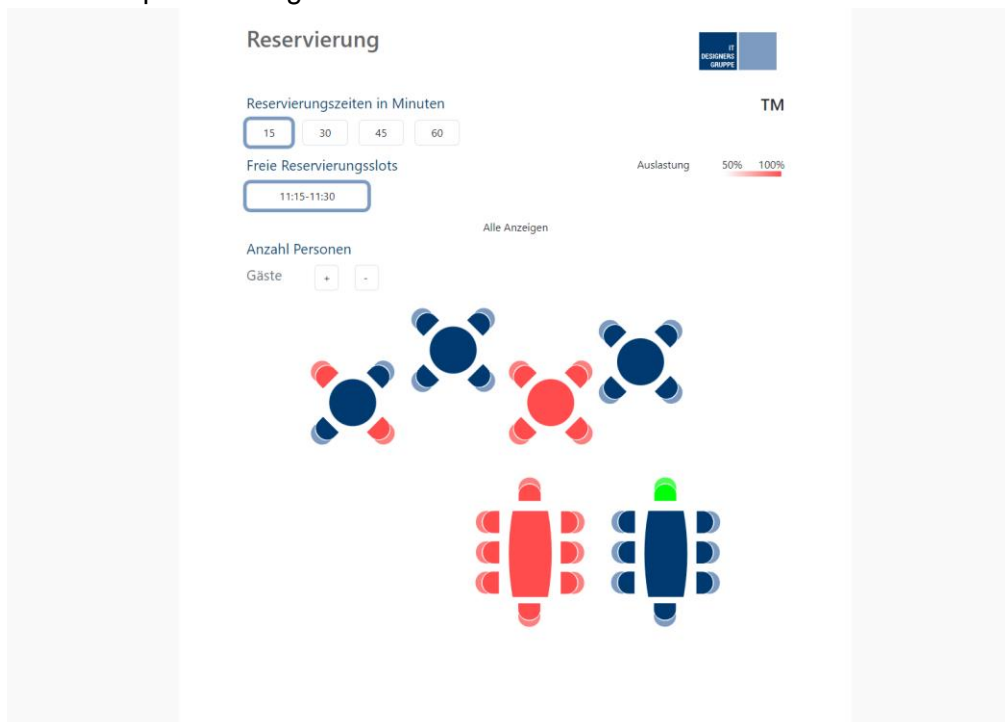



Figure 7: Reservierungsseite mit Gewohnheitspreset in React gerendert mit den Firmenfarben

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

Reservierung 

Reservierungszeiten in Minuten TM

Freie Reservierungsslots Auslastung 50% 100%

11:00-11:15	11:15-11:30	11:30-11:45	11:45-12:00
12:00-12:15	12:15-12:30	12:30-12:45	12:45-13:00
13:00-13:15	13:15-13:30	13:30-13:45	13:45-14:00

Anzahl Personen

Gäste

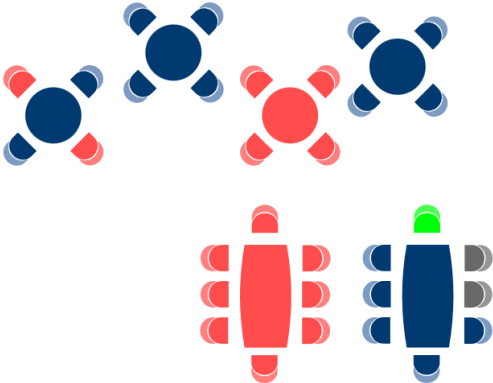


Figure 8: Reservierungsseite in React gerendert im erweiterten Modus mit frei wählbarer Zeit

Architektur/ Technologie

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.

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>

Genutzte IDEs

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

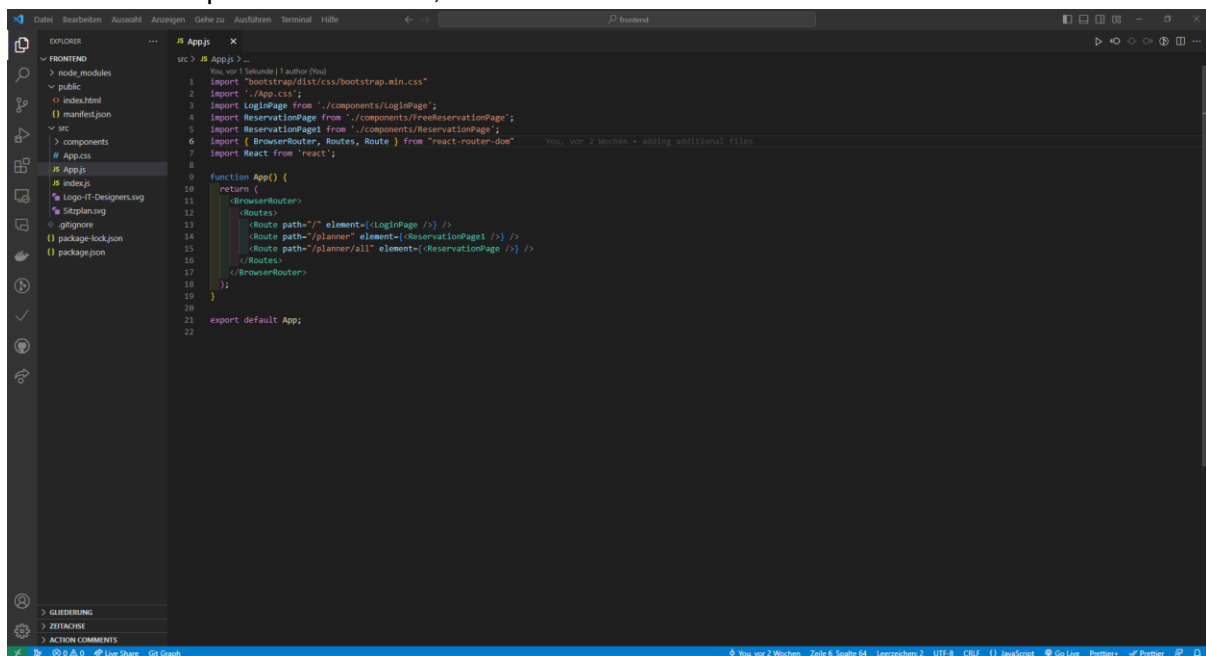


Figure 9: Für das Frontend wird IntelliJ Community verwendet.

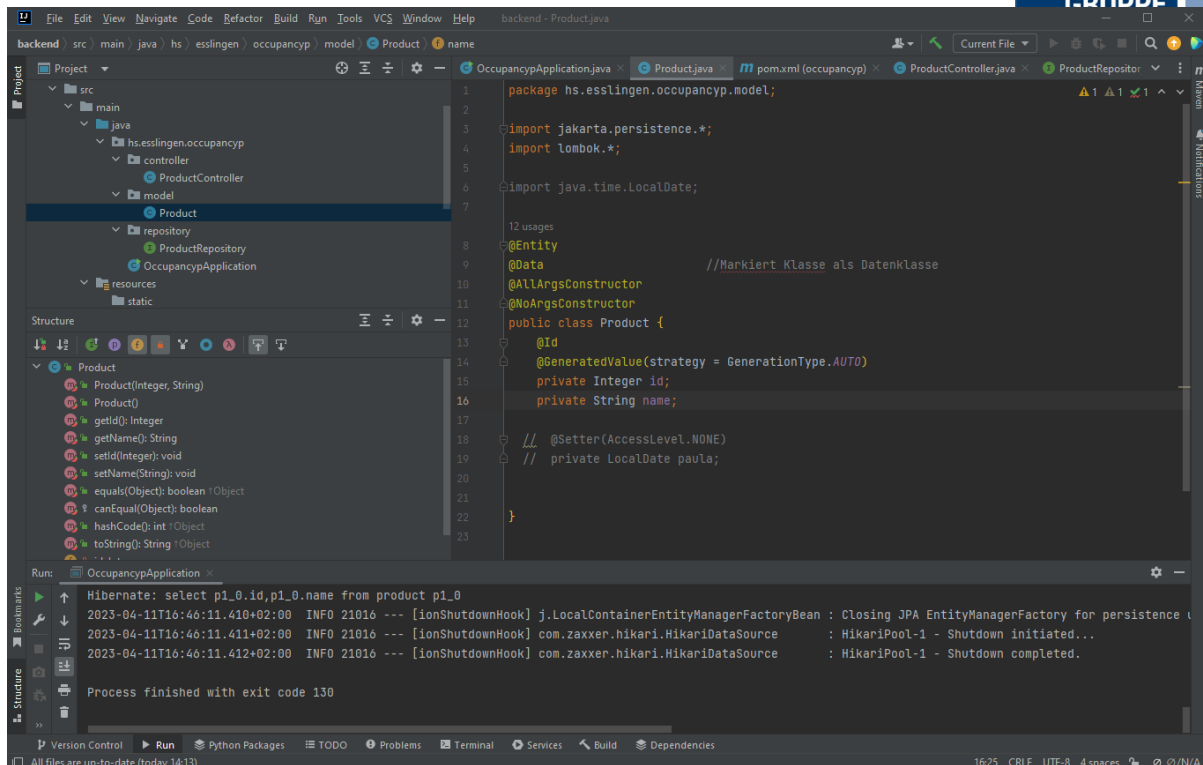


Figure 10: Für das Backend wird IntelliJ Community verwendet.

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/?utm_source=postman-home

DBeaver

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

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 postgres:latest
```

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

Frontend: React

Als Frontend wurde, wie auch in der Aufgabenstellung vorgeschlagen React.js 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 sind folgende Tutorials empfohlen:

https://www.youtube.com/watch?v=WMf5UEZLXyM&list=PLNmsVeXQZj7oi_Q4whC28Yp12l1l-hauk&index=3

tutorial basics only

<https://www.youtube.com/watch?v=zJxJerQtUdk>

Weitere Befehle, die im Terminal auszuführen sind:

```
npm install -g npx
```

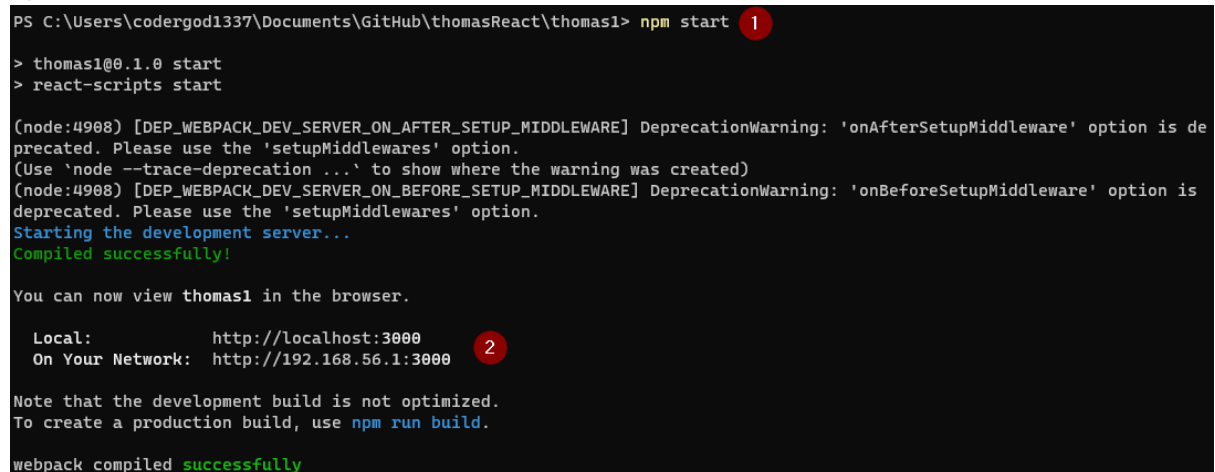
Um ein React Projekt zu erstellen: Im Terminal in den übergeordneten Ordner für ein neues Projekt und mit:

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

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

```
cd projekt123
```

```
npm start
```



```
PS C:\Users\codergod1337\Documents\GitHub\thomasReact\thomas1> npm start 1
> thomas1@0.1.0 start
> react-scripts start

(node:4908) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is de
precated. Please use the 'setupMiddlewares' option.
(Use 'node --trace-deprecation ...' to show where the warning was created)
(node:4908) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is
deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

You can now view thomas1 in the browser.

  Local:            http://localhost:3000 2
  On Your Network:  http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

1: Kommando

2: Serveradresse für die Entwicklung

Der Port kann mit einer neuen Datei ".env" mit dem Inhalt: `PORT=6969` festgelegt werden.

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.

Presettings bis jetzt:

Maven Projekt! Eine vollständige Liste der Dependencies im Anhang:

The screenshot shows the Spring Initializr interface. On the left, under 'Project', 'Gradle - Groovy' is selected. Under 'Language', 'Java' is selected. 'Spring Boot' version '3.0.5' is selected. Under 'Project Metadata', the group is 'HS-Esslingen', artifact is 'OCCUPANCYP', name is 'OCCUPANCYP', description is 'We will rock it', and package name is 'HS-Esslingen.OCCUPANCYP'. Packaging is 'Jar' and Java version is '17'. On the right, under 'Dependencies', 'Spring Web' (WEB), 'Spring Data JPA' (SQL), and 'PostgreSQL Driver' (SQL) are listed with red minus icons to remove them. A button 'ADD DEPENDENCIES... CTRL + B' is at the top right of the dependencies section.

Screenshot mit den benötigten Einstellungen (wird noch aktualisiert, wenn alle Dependencies geklärt sind)

Setting Port:

neue Datei: application.properties mit `PORT=1692`

In den Resources/ Application.properties sind für die Datenbank folgende Einstellungen:

`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`

Doku für Maven

<https://mvnrepository.com/>

NGINX Webserver

Für die Übergabe der Software als Docker Container, wird NGINX ermöglicht. So kann die React.js App mit dem Backend kommunizieren

```
docker pull nginx
```

```
docker run -d -p 6969:80 -v c:/dockerdir/mynginx:/usr/share/nginx/html --name  
OCCUPANCYP nginx
```

-p Portmapping -> localhost:6969 im Browser

-v Volume mounten: auf C:\dockerdir\mynginx liegen die Reactfiles/ HTML

Anleitung Reverse Proxy:

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

Authentifikation - KeyCloak (not yet)

(kommt erst noch)

Für die Authentifikation wird KeyCloak als Docker Container genutzt.

```
docker pull bitnami/keycloak
```

adding "Client" für die App und "User" die sich anmelden müssen

```
docker run -d -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e
```

```
KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:21.0.1 start-dev
```

<https://www.youtube.com/watch?v=5z6gy4WGnUs>

<https://www.youtube.com/watch?v=RupQWmYhrLA>

https://www.youtube.com/watch?v=Pwyl3kXr_IM

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.

Rollen/ Hauptverantwortlich:

Frontend: David
Backend: Emir und Silas
Datenbank: Silas
Protokolle: Nico
Kundenkontakt: Thomas
Doku: Thomas
Authentifikation: Thomas

Aufwandsschätzungen

Gesamte Zeit 5 Studenten: 20 Stunden pro Woche * 15 Wochen

→ 300h pro Student, da jeder einzelne Student in allen Teilbereichen Bescheid wissen muss, werden 50% der Zeit, die ein Verantwortlicher für ein konkretes Thema benötigt für die Einarbeitung der anderen in das jeweilige Thema veranschlagt.

Fett geschriebene Punkte sind nicht aufteilen und Für alle Studenten voll

- **Zeit für Technologierecherche und Ausarbeiten der Anforderungen:**
3 Wochen je Student 60h
- Frontend Design 30h
- Frontend Technik/ Logik 80h
- Backend Datenbankschnittstelle 40h
- Backend REST 120h
- Authentifikation 40h
- Docker Compose für Übergabe 30h
- Datenbank Layout 30h
- **Besprechungen 20h**
- **Seminare 16h**
- **Einführung + Präsentation 10h**
- Testen 10h

Eine nähere Beschreibung der Aufwandseinschätzung folgt in der Präsentation mit Hilfe eines Gantt diagrams.

Anhang

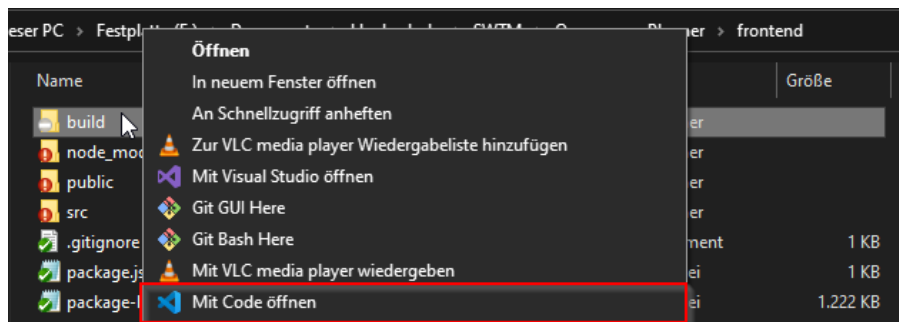
Noch nicht benutzte Parts

Nginx

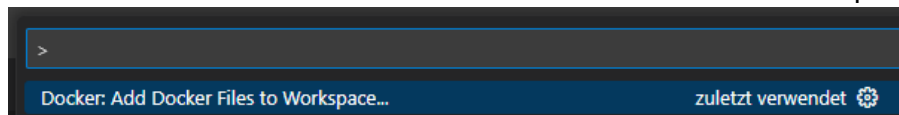
<https://aws.plainenglish.io/how-to-create-custom-nginx-docker-image-cd02242a2478>

`npm run build` → Erstellt erforderliche html,js,css Dateien im “build” Ordner

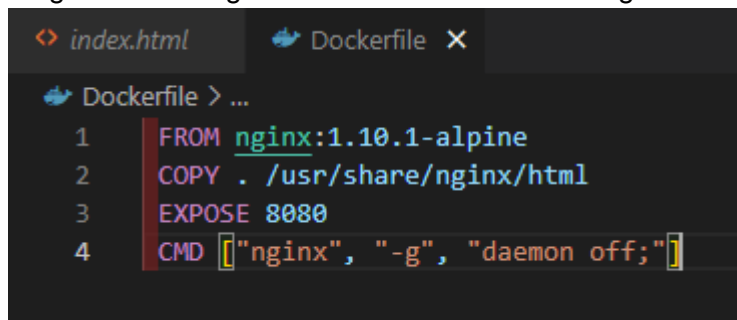
build Ordner in VS Code öffnen



Dockerfile in VS Code mit dem Befehl “Add Docker files to Workspace”



Folgender Eintrag in erstelltes Dockerfile eintragen



Mit dem folgenden Befehl Docker Image erstellen

```
docker build -t <new_image_name> <url or path of context>
```

Mit dem Befehl überprüfen, ob das Image erstellt wurde

```
docker images
```

Docker Container aus Image erstellen und ausführen

```
docker run -d --name <name-container> -p 8080:80 <image_name>
```

Docker Container überprüfen, ob dieser ausgeführt wird

```
docker ps
```

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>
</dependencies>
```

Protokolle

Kundenprotokoll vom 30.03.2023 (Requirements)

- 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

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