HOCHSCHULE ESSLINGEN

Desk Planner

IT DESIGNERS GRUPPE

Projekt DeskPlanner

Studienprojekt Softwaretechnik und Medieninformatik

Gökcek, Pinar
pigoit00@hs-esslingen.de

Mezger, Kyle
kymeit00@hs-esslingen.de

Imort, Jay
jaimit00@hs-esslingen.de

Hofmann, Rico
rihoit00@hs-esslingen.de

Pachtsinis, Darios
dapait03@hs-esslingen.de

04. April 2022

Inhaltsverzeichnis

0 Zeitplan **4**

1 Zielgruppe, Problemstellung, Eigenschaften, Alleinstellungsmerkmal **7**
1.1 Zielgruppe . . . 7
1.2 Problemstellung . . . 10
1.3 Eigenschaften . . . 11
1.4 Alleinstellungsmerkmal . . . 11

2 User Stories **12**

3 User Interface Entwürfe **14**
3.1 Mitarbeiter User Interface . . . 15
3.1.1 Buchen . . . 15
3.1.2 Buchungen/Chronik . . . 17
3.1.3 Nachrichten . . . 18
3.2 Administrator User Interface . . . 19
3.2.1 Raum-Editor . . . 20
3.2.2 Benutzerverwaltung . . . 22

4 Technisches Konzept **23**
4.1 Verwendete Frameworks . . . 23
4.2 Softwarearchitektur . . . 25
4.3 Datenbanken . . . 27

5 Implementiertes User Interface **29**
5.1 Unterschiede beim generellen Design . . . 29
5.2 Unterschiede des Zeitauswahl Pop-Ups . . . 31
5.3 Räume und Raumeditor . . . 33

6 Finalisierende Ausführungen **34**
6.1 Installations- und Administrationshandbuch . . . 34
6.1.1 Installationshandbuch . . . 34
6.1.2 Administrationshandbuch . . . 37
6.2 Reflektion: Aufteilung des Teams . . . 38
6.3 Reflektion: Projektmanagement . . . 39
6.4 Reflektion: Lernfortschritt . . . 40

6.5 Lizenzen 40
6.6 Ausblick 41

0 Zeitplan

Der Zeitplan hangelt sich an den vorgegebenen Meilensteinen 1-4 entlang.

Meilenstein 1 (12.04.22)
Präsentation:

- Aufgabenverteilung
- Projektmanagement
- Technologien
- Funktionsumfang
- UI-Entwürfe, z.B. Low Fidelity Mock-Ups

Meilenstein 2 (10.05.22)
Dokumentation:

1. Zielgruppe, Problem, Eigenschaften, Alleinstellungsmerkmal
2. User Stories
3. User Interface Entwürfe
4. Technisches Konzept
    - Verwendete Frameworks
    - Softwarearchitektur
        - UML-Verteilungsdiagramm des Gesamtsystems
        - UML-Komponentendiagramm (jeweils für Client und Server)
    - Datenbank
        - Entity Relationship Model
5. UI-Entwürfe, z.B. Low Fidelity Mock-Ups

Meilenstein 3 (24.05.22)
Dokumentation:

- Überarbeiten und ergänzen Sie das Konzept von Meilenstein 2

Technischer Prototyp:

- Entwickeln Sie einen vorführbaren technischen Prototypen inkl. Login.
- Bereit halten für Code Reviews: Jedes Teammitglied muss einen Code-Teil präsentieren können

Meilenstein 4 (28.06.22)
Dokumentation erweitern:

- Installations- und Administrationshandbuch
- Aufteilung des Teams: wer hat was gemacht (mit Namen der Teilnehmer)?
- Reflektion Projektmanagement: vergleichen Sie Ihre erste Planung mit der tatsächlichen Durchführung; vergleichen Sie die Aufwandsschätzungen mit den realen Aufwänden;
- Reflektion Lernfortschritt: was haben Sie gelernt?
- Lizenzen: verwendete Lizenzen (Fremdcode: Frameworks, Libraries), unter welche Lizenz stellen Sie Ihren Codes zur evtl. Weiterverwendung
- Ausblick: was könnte an Ihrem Projekt ergänzt werden?

Abschlusspräsentation

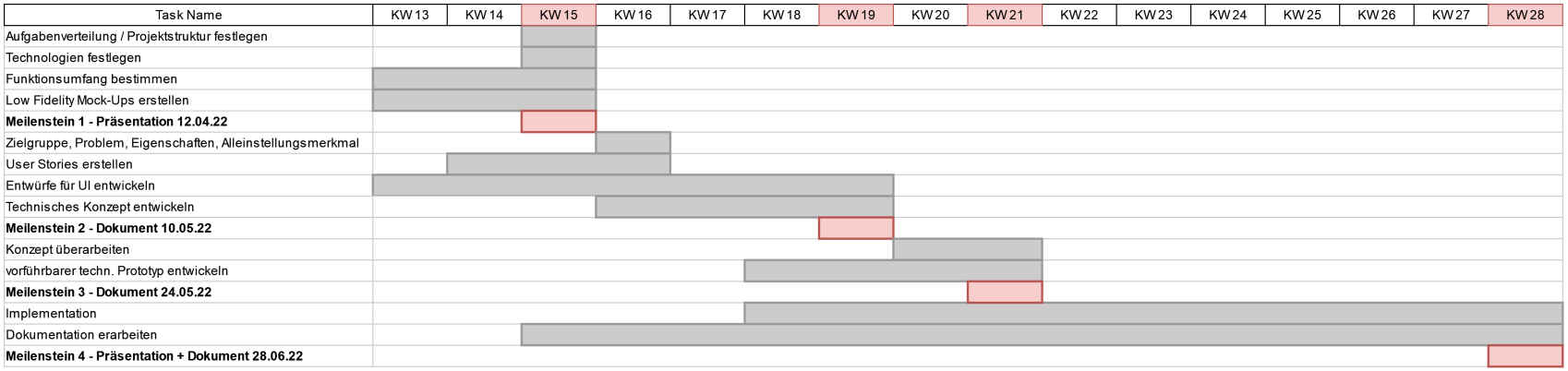

Abbildung 0.1: Zeitplan

1 Zielgruppe, Problemstellung, Eigenschaften, Alleinstellungsmerkmal

Im Folgenden wird der erste Meilenstein des Studienprojekts genauer erläutert. Thematisiert werden die Unterthemen Zielgruppe, Problemstellung, Eigenschaften und Alleinstellungsmerkmal.

1.1 Zielgruppe

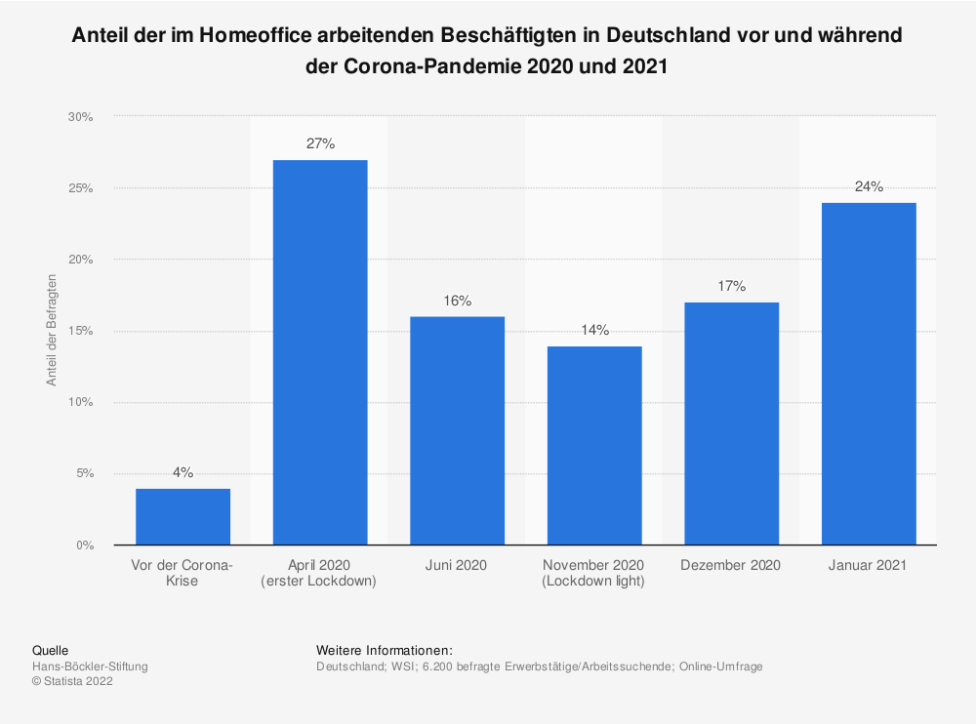

Abbildung 1.1: Home Office

Aufgrund der Coronapandemie wurden Konzepte wie Home-Office in vielen Industrien ausprobiert, in denen es so etwas zuvor wenig gab. Mit diesen neuen Nutzern von dy-

namischen Arbeitsplatzkonzepten ist der potentielle Nutzerstamm für eine Anwendung zum dynamischen Zuweisen von Arbeitsplätzen stark gestiegen. Maximal waren während der Pandemie, laut einer Umfrage der Hans-Böckler-Stiftung (vgl. Abbildung 1.1), 27% (vgl. Abbildung 1.1) der Befragten im Home-Office. Laut dem Ifo-Institut können ca. 56% aller Arbeitnehmer zumindest Teilzeit im Home-Office Arbeiten. Das zeigt für wie viele Arbeitnehmer ein dynamisches Arbeitsplatzkonzept eine Möglichkeit wäre. Dafür müssen aber auch Home-Office skeptische Arbeitgeber von den Vorteilen eines dynamischen Arbeitsplatzsystems, wie z.B. reduzierte Büro-Flächen und so geringere Miet- / Baukosten, überzeugt werden.

Personas:

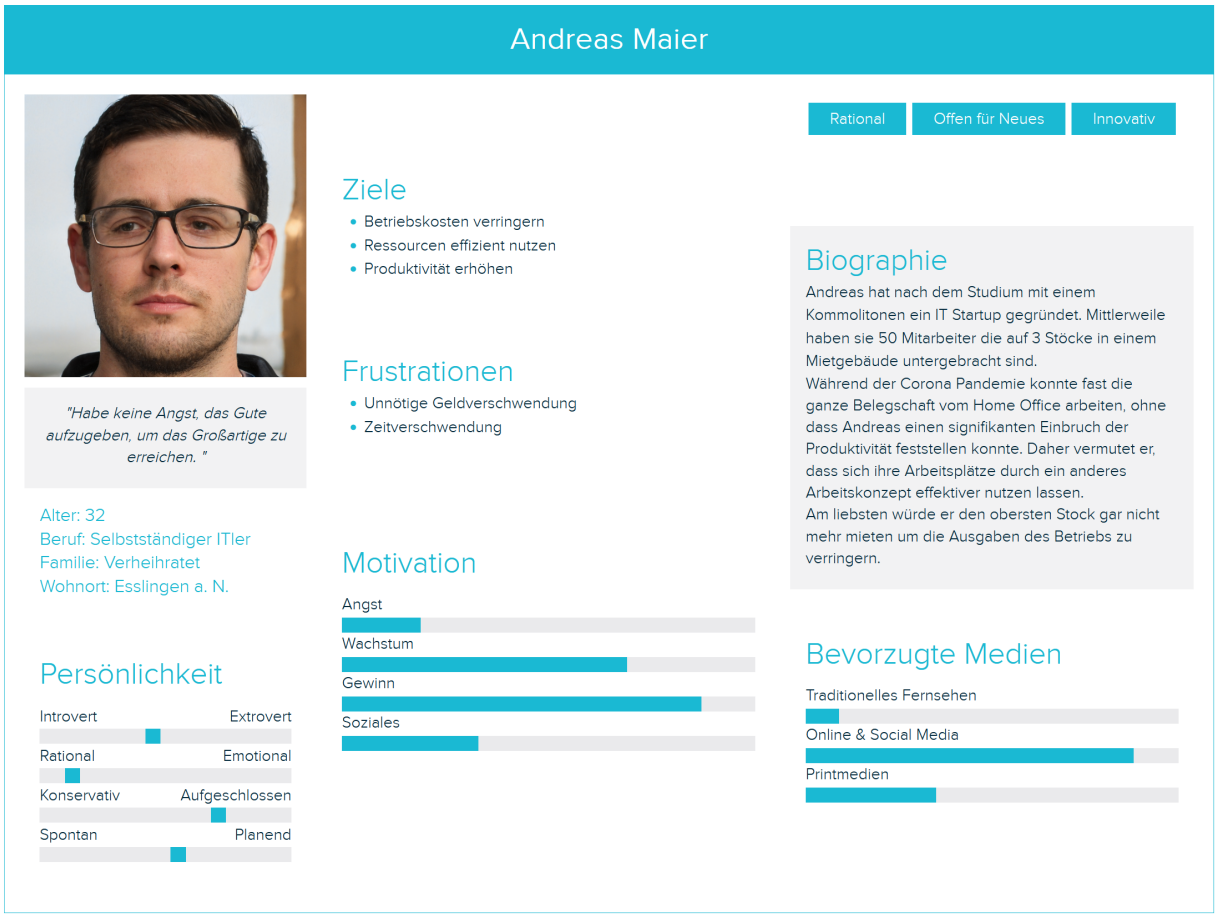

Abbildung 1.2: Persona Andreas Maier

Andreas repräsentiert die offensichtlichste potentielle Nutzergruppe. Er erkennt schon selbst das Potential von nicht traditioneller Arbeitsplatznutzung und kommt schon von sich aus auf die Idee, nach den Erkenntnissen durch die Corona Zeit, die Arbeitsplatznutzung in seiner Firma umzustellen.
Er kann vermutlich mit rein rationalen Argumenten zur Kostenminderung, effizienten Zeit-

nutzung etc. von unserem Konzept überzeugt werden, hat aber auch hohe Ansprüche an Funktionalität und allgemeine Qualität der Anwendung.

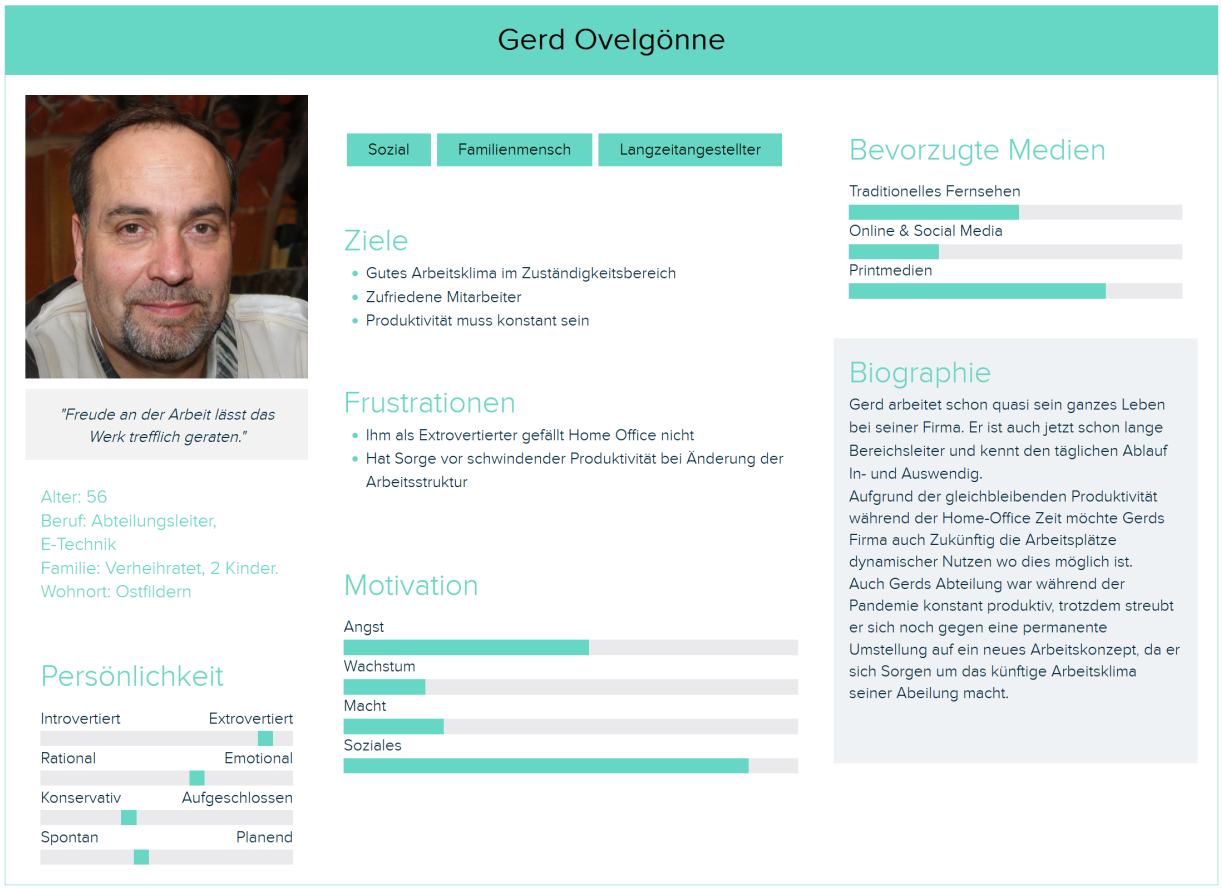

Abbildung 1.3: Persona Gerd Ovelgönne

Gerd ist Repräsentant einer konservativen Nutzergruppe, die noch nicht voll vom Konzept dynamischer Arbeitsplatzbelegung überzeugt ist, aber genug Motivationen haben, um potentiell von den Vorteilen überzeugt zu werden.
Gerd ist schon sehr lange in seinem Beruf. Er weiß genau was wie zu laufen hat und macht daher nur ungern Veränderungen durch.
Durch eine Vorgabe seiner Firma, alternative Arbeitsplatzkonzepte auszuprobieren, hat er jedoch eine externe Motivation, um das Arbeitskonzept seiner Abteilung zumindest ein wenig umzustellen. Auch zufriedene Mitarbeiter als sein persönliches Ziel, könnten für Gerd Motivation zur Umstellung sein. Den Mitarbeitern mehr Flexibilität zu bieten, sowohl in der Wahl wo in der Abteilung sie sitzen als auch in der Entscheidung, ob sie eine Arbeit von Zuhause erledigen können, kann sich positiv auf die Zufriedenheit der Mitarbeiter auswirken. Vor allem wenn flexibleres Arbeiten durch die Vorgabe der Geschäftsführung in anderen Abteilungen möglich wird, sollten den eigenen Mitarbeitern auch ähnliche Möglichkeiten geboten werden, um Frustration zu verhindern.
Ihm sollten, da ihm Home-Office nicht so gefällt, auch Vorteile einer flexiblen Arbeitsplatz-

nutzung ohne Home-Office gezeigt werden, wie bspw. die Einrichtung von Ruhearbeitszonen oder die Wahl der Arbeitsplätze entsprechend der Arbeit, die an einem Tag verrichtet werden muss.

1.2 Problemstellung

Während der Pandemiezeit wird verstärkt von zuhause aus gearbeitet, dadurch schwankt die Anzahl der Arbeiter im Büro stark und die interdisziplinären Kollegen habe keinen Überblick mehr darüber, wer im Büro ist und wer nicht. Zumal grundsätzlich nicht jeder in das Home-Office kann bzw. will. Dies kann diverse Gründe, wie z. B. fehlende Räumlichkeiten oder die geschwächte Konzentrationsfähigkeit in den eigenen vier Wänden sein. Agile Arbeitsplätze haben den Vorteil, dass der Arbeitsplatz je nach momentanem empfinden gewählt werden kann z. B. im sogenannten Ruhebereich, Bereiche die näher an Heizkörpern sind oder hellere Arbeitsplätze an Fenstern. Projektspezifisch kann es vorteilhaft sein neben einem bestimmten Kollegen zu arbeiten, um den Austausch produktiver zu gestalten. Diese Bereiche können frei benutzt werden, sofern diese nicht durch eine andere Kollegin oder einen anderen Kollegen belegt sind. Derzeit ist es gängig, dass die Bereiche nach dem Prinzip “first come first serve“ besetzt werden.

Wenn eine Person am Morgen an einem Arbeitsplatz sitzt, dann aber im Laufe des Tages für einen längeren Zeitraum diesen verlassen muss, wird diese Ressource unnötig blockiert. Raumbelüftungssysteme, Heizungen und Klimaanlagen für Räume bzw. Gebäude werden nicht nach der Personenauslastung betrieben, in Hinblick auf die steigenden Energiekosten, ist dies weder ökonomisch noch nachhaltig. Immungeschwächte Personen die ihre Tätigkeit nur vor Ort im Büro erledigen können, haben pandemiebedingt nicht die Möglichkeit Arbeitsplätze zu Buchen, die weiter weg von den restlichen Arbeitsplätzen bzw. Kollegen sind. Der Arbeitgeber, der nur eine bestimmte Anzahl Arbeitsplätzen hat, kann nicht mehr als diese Anzahl an Arbeiter in dem Bürokomplex beschäftigen. Desweitern gibt es derzeit keine einfache Möglichkeit die Büro Belegschaft, über einen längeren Zeitraum zu analysieren. Diese Analyse kann unter anderem zur dauerhaften Arbeitsplatz Reduktion, bei gleicher Mitarbeiterzahl führen.

Unsere Anwendung ermöglicht es, die oben angesprochenen Fälle zu Gunsten des Arbeitgebers und -nehmers zu lösen. Die Raumplanung ermöglichet eine Buchung in 15 min Rastern auch Dauerbuchungen über mehrere Monate sind möglich. Nicht nur Arbeitsplätze in einem Raum, sondern bestimmte Räume in bestimmten Gebäuden können gebucht werden. Auch ist eine Chatfunktion zwischen Teamleitern und Mitarbeitern bzw. zwischen Mitarbeitern möglich. Dies hat den Vorteil das Buchungen bei Sonderfällen unter Mitarbeitern einfach getauscht werden können. Eine Synchronisation mit dem Outlookkalender wäre auch denkbar.

1.3 Eigenschaften

Die Web-Applikation DeskPlanner wird mindestens folgende Eigenschaften nachweisen können:

- Open Source (Lizenzfrei) als kostenfreie Möglichkeit für vor allem kleine Unternehmen
- Einfache Möglichkeiten, um Komplexität zu vermeiden, aber so viele Möglichkeiten der Raumplanung beizubehalten
- Three-Tier Architecture ähnliche Anwendungsweise der Raumplanung im Sinne von Gebäude → Stockwerk → Raum

Des Weiteren sind kleine Eigenschaften wie Einhaltung der 7 Usability Prinzipien (vgl. DIN EN ISO 9241-110) vorgesehen, um so die Bedienung des Kunden so einfach wie möglich zu halten. Beispiele hierfür sind Hilfstexte, welche die verschiedenen Buttons erklären sollen oder selbsterklärende Icons der jeweiligen Buttons. Außerdem soll der Nutzer nicht mit Information beschüttet werden, sondern eine auf seinen Nutzungskontext angepasste Benutzeroberfläche sorgen für Aufgabenangemessenheit. Durch die individuelle Raumgestaltung erhält der Nutzer ein steuerbares Softwaresystem, welches genau das machen wird, was der Nutzer auch erwartet.

1.4 Alleinstellungsmerkmal

Als herausragendes Leistungsmerkmal des DeskPlanners bezeichnet man die effiziente Arbeitsplatznutzung. In der heutigen Branche, vor allem durch Corona bestätigt, gilt es, die Kosten immer weiter zu reduzieren. Durch die Raumbuchungsmöglichkeiten, kann ein Unternehmen die Arbeitsplätze minimieren, Homeoffice Mitarbeiter perfekt einplanen und dadurch Kosten der Technik und Größe des Raumes einsparen für andere Investitionen. Als Open Source Web-Applikation kann ein Unternehmen sogar lizenzfrei den Arbeitsplatz kosteneffizienter nutzen. Genau hier soll der DeskPlanner vor allem kleinere Unternehmen unterstützen, um in Zeiten der Pandemie, als auch danach, deren Fortbestand zu gewährleisten.

2 User Stories

Mitarbeiter/-in:

- Als Mitarbeiter/-in möchte ich Sitzplätze anschauen, damit ich auf einen Blick sehe welche Plätze verfügbar/vergeben sind.
- Als Mitarbeiter/-in möchte ich Sitzplätze buchen, damit ich für die gebuchte Zeit einen Arbeitsplatz zur Verfügung habe.
- Als Mitarbeiter/-in möchte ich Sitzplätze abonnieren, damit ich einen Sitzplatz über längere Zeit zu einer gewissen Zeit reserviert habe, ohne jeden Tag einzeln buchen zu müssen.
- Als Mitarbeiter/-in möchte ich Sitzplätze stornieren, damit ich bei einer Fehlbuchung oder Krankheit den Platz für andere Mitarbeiter wieder buchbar machen kann.
- Als Mitarbeiter/-in möchte ich mit einem Platz Besitzer chatten können, damit ich anfragen kann, ob der jeweilige Platz verfügbar gemacht werden könnte.
- Als Mitarbeiter/-in möchte ich Nachrichten erhalten können, damit ich bei der Anfrage eines anderen Mitarbeiters meinen gebuchten Platz für diesen stornieren kann.

Administrator/-in:

- Als Administrator/-in möchte ich alle Funktionen des Mitarbeiters haben, da ich mitunter Mitarbeiter/-in bin.
- Als Administrator/-in möchte ich das Layout von Räumen anpassen, damit es dem tatsächlichen Büro entspricht und es somit besser erkennbar ist für Mitarbeiter um welchen Raum und Sitzplatz es sich handelt.
- Als Administrator/-in möchte ich Gruppen verwalten können, damit ich Bestimmte Positionen und Personen in Gruppen einteilen kann, damit ein Geschäftsführer diese in Raumbuchungen beschränken kann.

Geschäftsführer/-in:

- Als Geschäftsführer/-in möchte ich alle Funktionen des Mitarbeiters haben, da ich mitunter Mitarbeiter/-in bin.
- Als Geschäftsführer/-in möchte ich die Buchbarkeit von Räumen auf Gruppen einschränken, damit bestimmte Räume für bestimmte Positionen/Personen ausgelegt sind.

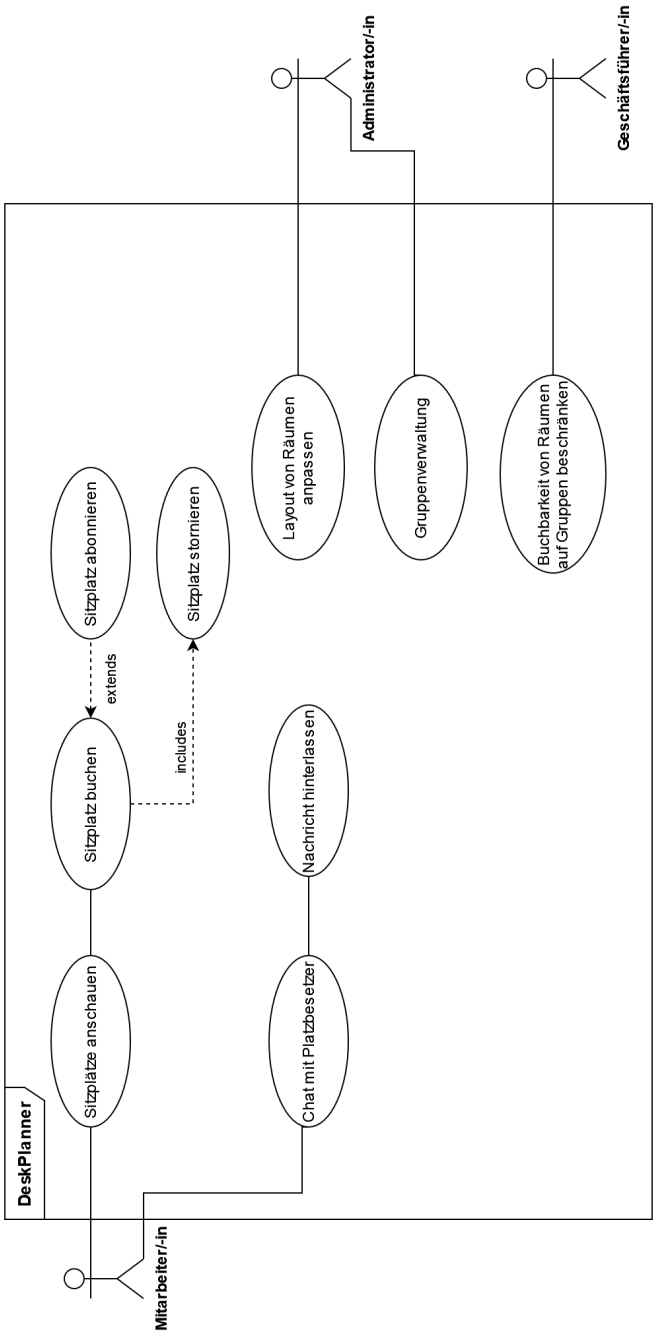

Abbildung 2.1: Usecase-Diagramm

3 User Interface Entwürfe

Die User Interfaces wurden so gestaltet, dass sie die zuvor genannten User Stories, Use Cases und Anforderungen bestmöglich erfüllen.

Dabei werden die Mitarbeiter Use Cases in einer mobile first Anwendung verpackt, weil die Arbeitsplatzbuchungen schnell und leicht von der Hand gehen sollen. Ebenfalls ist mobile first praktisch für die Nachtrichten-Funktion, da die Erreichbarkeit und die Möglichkeit schnell zu antworten, auf mobilen Geräten besser ist als auf Desktop Systemen.

Die Use Cases des Administrators und der Geschäftsführung werden wiederum auf Desktop-Systeme ausgelegt, da dort eine übersichtliche, effiziente und präzise Nutzung wichtiger ist, als die Flexibilität auf das System zuzugreifen zu können.

3.1 Mitarbeiter User Interface

Nachfolgend werden die Funktionen und Seiten des mobile first User Interfaces visualisiert und erläutert.

3.1.1 Buchen

Sobald die Anwendung abgerufen wird, erscheint das Buchen-Fenster.

Ersichtlich ist das aktuell geöffnete Fenster durch die Funktionsleiste am unteren Ende der Anwendung. Dort wird die ausgewählte Funktion, sprich das aktuelle Fenster, durch die dunkle schwarze Färbung und den Balken darunter hervorgehoben.

Im oberen Teil der Anwendung ist es möglich über Dropdown-Menüs das Gebäude, die Etage und den Raum rauszusuchen, in dem die Buchung gewünscht ist. Dabei geschieht die Auswahl von rechts startend aus, spricht zuerst das Gebäude, dann die Etage und zuletzt den Raum. Dies ist wichtig, weil je nach Gebäude andere Etagen bestehen, welche dann wieder nur bestimmte Räume enthalten.

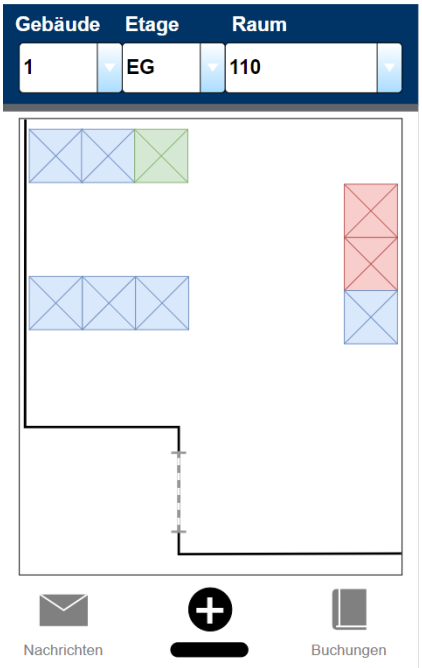

Abbildung 3.1: User Interface: Buchen

Wenn im oberen Teil alle Angaben bis zum Raum hin gegeben wurden, erscheint in der Mitte der Anwendung der ausgewählte Raum. Dieser wird durch ein Rechteck mit dünnen schwarzen Linien in der Anwendung begrenzt. In diesem Rahmen soll es möglich sein den Raum zu verschieben und zu vergrößern und zu verkleinern. Der Raum an sich besteht zum einen aus Wänden, dicke schwarze Striche, und Türen, grau gestrichelte Einsätze in den Wänden. Türen und Wände werden lediglich zur Orientierung und besseren Vorstellung des Raumes genutzt.

In dem Raum befinden sich schließlich farbige Rechtecke, welche Arbeitsplätze darstellen, welche zum Interagieren benutzt werden. Die farbliche Kennzeichnung beschreibt, wie der Platz an dem heutigen Tag genutzt wird. Blau zeigt an, dass der Platz frei verfügbar ist für heute, rot zeigt an, dass der Platz mindestens in einem Zeitfenster von heute eine Buchung eines anderen Arbeitskollegen vorliegen hat und grün zeigt an, dass der Arbeitsplatz für heute selbst in mindestens einem Zeitfenster gebucht ist.

Um weitere und genauere Informationen über die Belegung eines Arbeitsplatzes zu erhalten, wird dieser angeklickt .

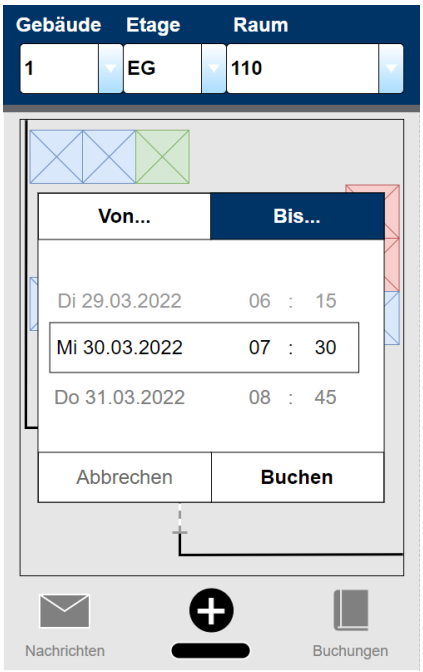

Abbildung 3.2: User Interface: Buchen - Zeitauswahl

Mit dem Anklicken erscheint ein Pop-Up Fenster, für die Zeiteingabe. In der Mitte des Fensters befindet sich die Zeitangabe, die aktuell ausgewählt ist. Durch unabhängiges hoch- und runterwischen des Datums, der Stunden und Minuten verändert sich die Zeitangabe. Die Zeitangaben sind dabei in 15-Minutenschlitze unterteilt.

Zu Beginn startet das Pop-Up mit der „Von“ Zeitangabe, welche den Start des Buchungszeitraumes festlegt. Im oberen Teil des Pop-Ups wird durch blaue Farbe hervorgehoben, ob es sich um die Eingabe der Startzeit oder der Endzeit handelt. Mit einem Klick auf „Von...“ oder „Bis...“ lassen sich somit auch die Zeiteingaben dafür wechseln.

Im unteren Teil des Pop-Ups sind stets die Eingaben verwerfbar durch einen Klick auf „Abbrechen“. Rechts daneben sind die Eingaben zu bestätigen und abzuschicken durch einen Klick auf „Buchen“. Dabei ist der Buchen-Button bei nur einer „Von“ Angabe noch ausgegraut und nicht anklickbar. Erst nach der Auswahl der Startzeit und der Endzeit kann der Buchen-Button geklickt werden.

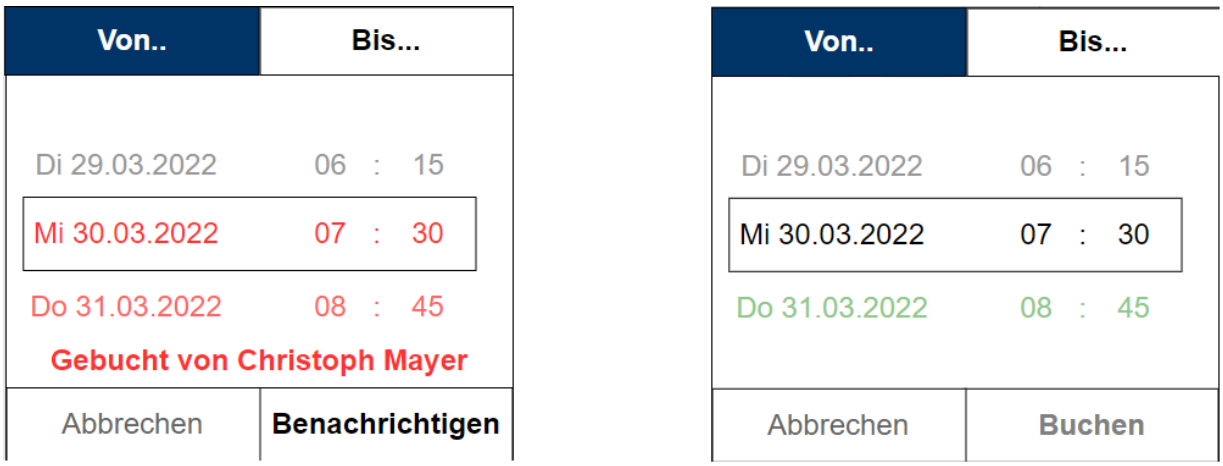

Abbildung 3.3: User Interface: Buchen - Informationen aus der Zeitauswahl

Aus der Zeitauswahl folgen noch weitere Informationen dazu ob ein Arbeitsplatz zu einem Zeitpunkt von jemand Anderen gebucht, frei oder selbst gebucht ist.
Schwarze Zeitangaben sind frei verfügbar und können ohne Weiteres gebucht werden.
Grüne Zeitangaben symbolisieren eigene Buchungen im System.
Diese dienen lediglich zur eigenen Information und besitzen keine weiteren Interaktionsmöglichkeiten.
Rote Zeitangaben sind von einem Kollegen gebucht, dieser Kollege wird auch in dem Pop-Up Fenster angezeigt. Anstatt dem Buchen-Button kann der Kollege nun per Nachricht angeschrieben werden, falls der Arbeitsplatz beispielsweise persönlich benötigt wird.

3.1.2 Buchungen/Chronik

Durch einen Klick auf das Buch in der Funktionsleiste unten, geschieht ein Wechsel in die Übersicht aller getätigten Buchungen. Die Buchungsübersicht wird in Abbildung 9 dargestellt und wird ebenfalls wieder durch die schwarze Färbung des Symbols in der Funktionsleiste und dem Balken darunter hervorgehoben.

In diesem Fenster werden alle Buchungen als Kacheln angezeigt, welche im System getätigt wurden. Dabei umfasst jede Kachel den Raum, in dem die Buchung gemacht wurde, das Datum und den Zeitraum, der gebucht wurde.

In diesem Fenster lässt sich auch jede getätigte Buchung wieder stornieren durch einen Klick auf das Minus-Symbol.

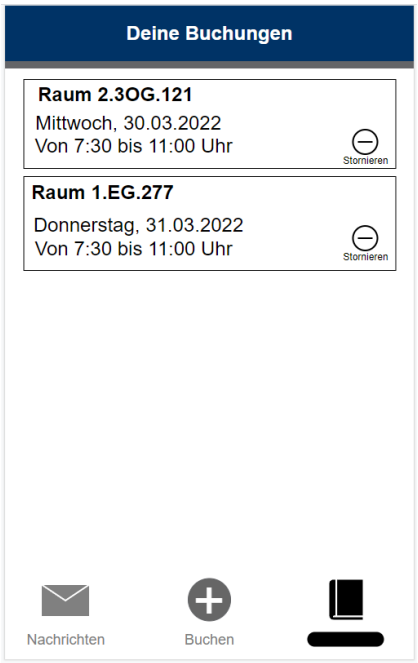

Abbildung 3.4: User Interface: Übersicht getätigte Buchungen

3.1.3 Nachrichten

Mit einem Klick auf das Briefsymbol, links in der Funktionsleiste, wechselt die Anwendung zur Nachrichtenübersicht. Dieses Symbol zeigt auch mit einer kleinen hochgestellten Zahl neben dem Icon an, dass ungelesene Nachrichten eingegangen sind.

Abbildung 3.5: User Interface: Nachrichtenübersicht

Die Nachrichtenübersicht enthält sämtliche Chats, die selbst gestartet wurden oder an einen persönlich gerichtet sind. Denn wie zuvor schon erwähnt, besteht die Möglichkeit, Kollegen eine Nachricht zu schreiben, wenn diese einen Arbeitsplatz gebucht haben, der jedoch selbst benötigt wird. Dies geschieht über die Zeitraumeingabe, beim Klicken des „Benachrichtigen“ Buttons, wenn ein Arbeitsplatz, zur gewünschten Zeit, belegt ist.

Die Nachrichtenübersicht enthält, wie bei der Buchungsübersicht, ebenfalls alle offenen Chats als Kacheln aufgelistet. Diesmal enthalten die Kacheln den Namen des anderen Chatteilnehmers, den Raum in dem die betroffene Buchung ist und den Textanfang der neusten Nachricht im Chat. Chatkacheln mit ungelesenen Nachrichten werden zudem noch hervorgehoben.

Für mehr Übersichtlichkeit in der Nachrichtenübersicht ist der Chat löschbar. Dieser sollte aber lediglich gelöscht werden, wenn das Thema des jeweiligen Chats sich erledigt hat. Dazu lässt sich jeder Chat löschen, mit einem Klick auf das X-Symbol der jeweiligen Kachel. Dies löscht den Chat nur persönlich und nicht für den anderen Chatteilnehmer, dieser kann den Chat so lange einsehen, bis er ihn auch löscht.

Durch das Anklicken einer Kachel in der Nachrichtenübersicht in Abbildung 10, erscheint der Chat inklusive Chatverlauf. Der Chat ist in Abbildung 11 dargestellt.

Im Kopf des Chats sind nochmal die gleichen Informationen, wie in der Übersicht, bis auf den Anfang der neusten Nachricht. Direkt darunter ist der Chatverlauf. Eigene Nachrichten sind hierbei nach rechts verschoben und Nachrichten des Gegenüber nach links.

Oberhalb der Funktionsleiste ist noch die Eingabeleiste, welche durch Anklicken das Verfassen einer Nachricht ermöglicht. Die Eingabeleiste öffnet hierbei schließlich die Texteingabe des Gerätes. Nach fertigem Verfassen der Nachricht kann diese abschickt werden mit dem Symbol rechts neben der Eingabeleiste.

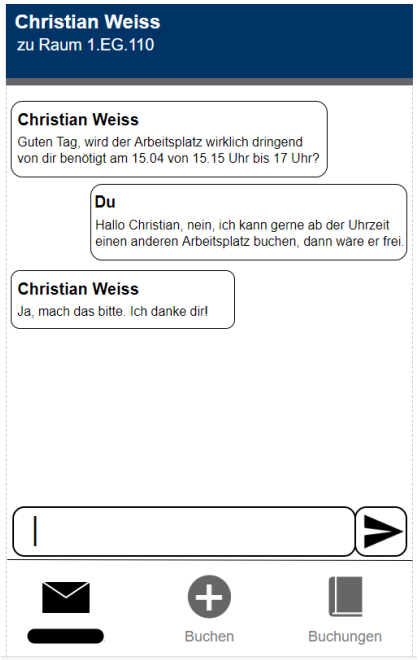

Abbildung 3.6: User Interface: Beispielchat

3.2 Administrator User Interface

Nachfolgend wird auf das Interface für Administratoren und Geschäftsführer eingegangen. Dies wird genutzt um neue Räume im System zu registrieren, zu löschen und zu bearbeiten. Ebenfalls lassen sich dort Nutzergruppen verwalten, die bestimmen, welcher Mitarbeiten in welchem Raum buchen kann.

3.2.1 Raum-Editor

Die genannten Aufgaben des Administrator User Interfaces lassen sich in dem Editor, in Abbildung 12, vornehmen.

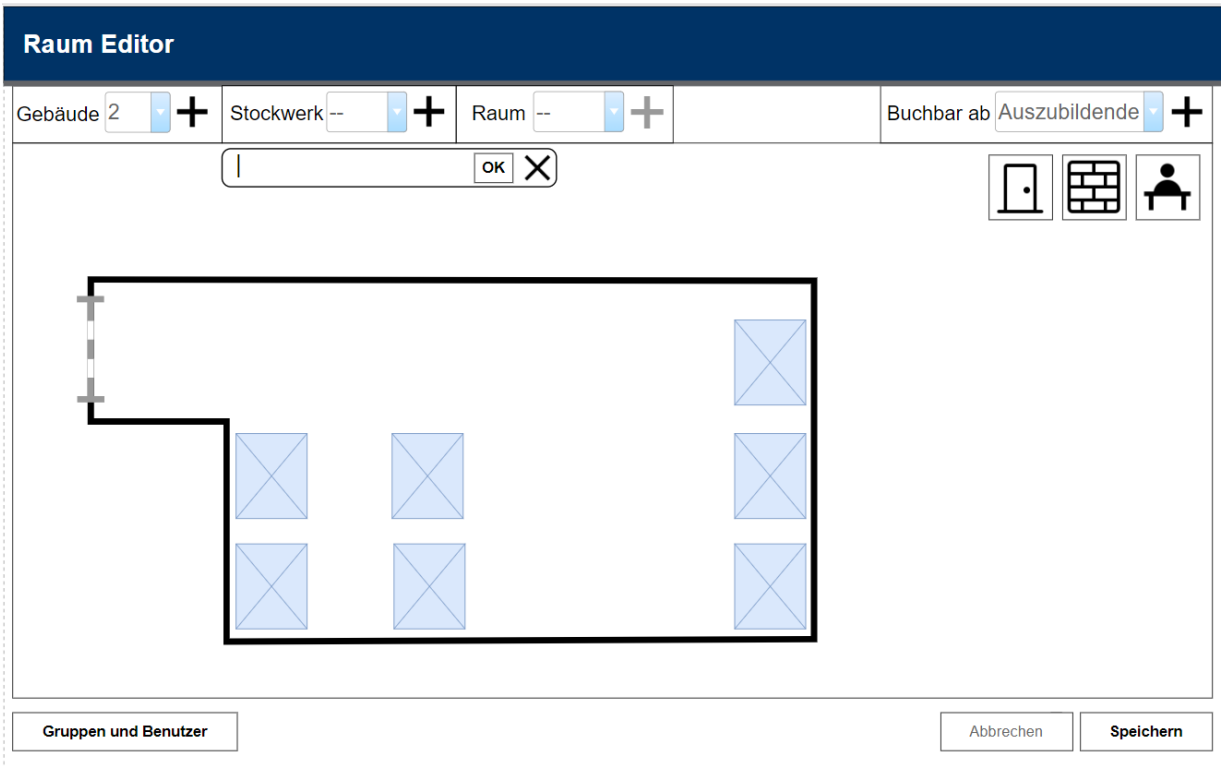

Abbildung 3.7: User Interface: Raum-Editor

Von oben nach unten durchgehend, werden nachfolgend die Funktionen des Editors erklärt.
Oben links besitzt der Editor Auswahlmöglichkeiten für das Gebäude, die Etage und den Raum. Ausgewählt wird über Dropdown Menüs, welche von links nach rechts ausgewählt werden müssen, wie in der normalen Raumwahl im Buchen Fenster. Mit einem Rechtsklick auf Einträge im Dropdown Menü lassen sich Räume löschen. Über das Plus-Symbol rechts neben dem Dropdown Menü kann ein neuer Eintrag jeweils zum System hinzugefügt werden. Dabei ist es so, dass immer ein Gebäude hinzugefügt werden kann, da dies nicht von vorherigen Auswahlen abhängt. Um eine Etage hinzuzufügen, muss ein Gebäude ausgewählt sein, damit die Etage dem ausgewählten Gebäude zugeordnet werden kann. Zuletzt kann ein Raum hinzugefügt werden, indem ein Gebäude und eine Etage ausgewählt werden.
Bei sämtlichem Hinzufügen erscheint eine Eingabeleiste für den Namen des Eintrags. Die Eingabe bestätigt sich schließlich mit Klicken von „Ok“ und verwerft sich mit dem Klicken des X-Symbols.

Ist nun einen Raum ausgewählt oder in der Neuerstellung soweit, dass das Layout angepasst werden kann, kann oben rechts die Gruppe auswählen werden, ab welcher der Raum buchbar ist. Dies geschieht ebenfalls über ein Dropdown Menü und mit dem Plus-Symbol kann eine neue Benutzergruppe hinzugefügt werden.

Mit dem Auswählen oder Erstellen eines Raumes, wird dieser auch unterhalb der oberen Leiste angezeigt. In dem Rechteck mit den dünnen schwarzen Linien, lässt sich der Raum interaktiv bearbeiten.
Oben rechts in der Bearbeitungsfläche sind die drei Objekte, die sich beliebig oft in den Raum ziehen lassen. Nämlich Türen, Wände und Arbeitsplätze. Die Visualisierung von Türen, Wänden und Arbeitsplätzen ist in „3.1.1 Buchen“ beschrieben
Türen lassen sich auf bestehende Wände draufziehen, skalieren und verschieben. Wände lassen sich einfach in die Bearbeitungsfläche ziehen und wie Polygonlinienzüge anpassen. Arbeitsplätze werden in den Raum reingezogen und können dort auch noch verschoben und skaliert werden. Die Elemente in der Bearbeitungsfläche lassen sich generell anklicken und bearbeiten. Mit der Entf. Taste lässt sich das angeklickte Element auch löschen.

Unten rechts sind letztlich die Änderungen an einem Raum komplett verwerfbar durch „Abbrechen“ oder eben speicherbar mit „Speichern“. Nach dem Speichern wird der Raum auch direkt ins System eingefügt.

Unten links ist nun noch ein Button für die Benutzerverwaltung. Mit einem Klick auf diesen öffnet sich ein Pop-Up, in welchem die Mitarbeiter den Berechtigungsgruppen zugewiesen werden können.

3.2.2 Benutzerverwaltung

Das Pop-Up für die Gruppenverwaltung und Nutzerzuweisungen in die Gruppen wird in Abbildung 13 beschrieben.

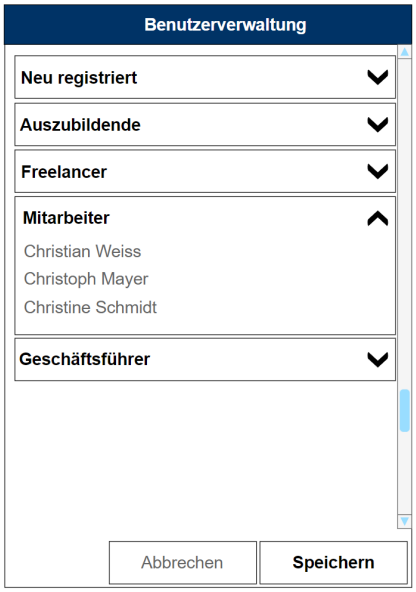

Abbildung 3.8: User Interface: Benutzerverwaltung Pop-Up

Hier sind alle Gruppen ersichtlich, die bisher erstellt wurden. Die Gruppen sind per per Drag and Drop neu sortierbar, damit eine hierarchische Struktur entsteht. Dabei ist die niedrigste Hierarchiestufe oben und die höchste unten. Eine höhere Stufe besitzt dabei jede Zugangsberechtigung der Stufen unterhalb dieser.

Die Gruppen können auch aufgeklappt werden, wodurch die jeweiligen Benutzer in dieser Gruppe angezeigt werden. Diese sind ebenfalls per Drag and Drop in andere Gruppen verschiebbar.

Nach den getätigten Änderungen sind diese speicherbar oder verwerfbar.

4 Technisches Konzept

4.1 Verwendete Frameworks

NestJS	**ExpressJS**
NodeJS Framwork	*NodeJS Framework*
+ Durch vorgegebene Struktur lässt sich das Projekt eher strukturiert halten	- Projekt wird bei großer Projektgröße schnell unstrukturiert
+ Durch die strukturierte Arbeitsweise gut für Teams geeignet	- Durch die Freiheit bei Technologiewahl eher für einzelne Developer besser
- weniger Freiheit bei Implementation	+ Mehr Freiheit bei der Implementation durch weniger Strukturvorgaben
- Aufgrund strengerer Strukturvorgaben schwerer zu lernen	+ Durch einfachen Aufbau leicht zu erlernen
- relativ neue Technologie mit weniger Nutzern und Tutorials	+ Viele Nutzer und dadurch auch viel Dokumentation und Tutorials

NestJS war eine Vorgabe des Product Owners (IT Designers Gruppe) für die Entwicklung einer Back-End API. Es ist ein Framework, das auf der JavaScript Umgebung NodeJS aufbaut und den Code in Module, Controller und Services aufteilt und somit eine klare Struktur zur Entwicklung vorgibt.
Aufgrund diesen strengen Strukturvorgaben, sind NestJS Anwendungen, im Vergleich mit bspw. ExpressJS, sehr gut skalierbar.

React	**Angular**
JS Library	*JS Library*
+ Vorbereitung für das Praxissemester da React sehr weit verbreitet ist	- Etwas weniger verbreitet als React
+ Sehr einfaches Erstellen von GUIs	- Erstellen von Benutzeroberflächen etwas komplizierter
- weniger strukturiert	+ Schon sehr ähnlich zu NestJS strukturiert
+ mehr Freiheit bei Implementation	- weniger Freiheit bei Implementation
- aufgrund weniger Strukturvorgaben schwerer große Projekte sauber zu halten	+ aufgrund der erzwungenen Struktur ergeben sich auch bei großen Projekten wenig Probleme

React ist sehr weit verbreitet und wird von vielen Firmen genutzt. Daher wollte unser Team zunächst diese etwas weiter verbreitete Library lernen. Außerdem ist React im Vergleich

zu Angular noch etwas einfacher zu lernen. Daher fiel auch die Wahl auf React, trotz der sehr NestJS-ähnlichen Struktur die Angular bietet.

MongoDB	**MySQL**
+ No SQL Datenbank	- SQL Datenbank
+ Datenstrukturen sind näher an tatsächlicher Implementation	- Datenstrukturen sind abstrakt, nicht so nah an der Implementation
+ Speichert Daten im JSON Format → Struktur nah an Objekten	- Speichert Daten in Tabellen
- Für uns neue Technologie	+ Schon bekannt aus vorherigen Semestern

Auch MongoDB war eine Vorgabe des ProductOwners. Im Vergleich zu "klassischen" Datenbanken wie MySQL halten Mongo Datenbanken ihre Daten nicht in Tabellen, sondern in JSON Files in einer Baumstruktur, d.h. MongoDB ist eine sog. NoSQL Datenbank. Daher ist MongoDB im Vergleich zu bspw. MySQL Datenbanken schon von der Grundstruktur der Datenhaltung sehr nah an der tatsächlich in der Entwicklung genutzten Strukturen der Implementation und verlangt so weniger Abstraktion als eine SQL Datenbank.

4.2 Softwarearchitektur

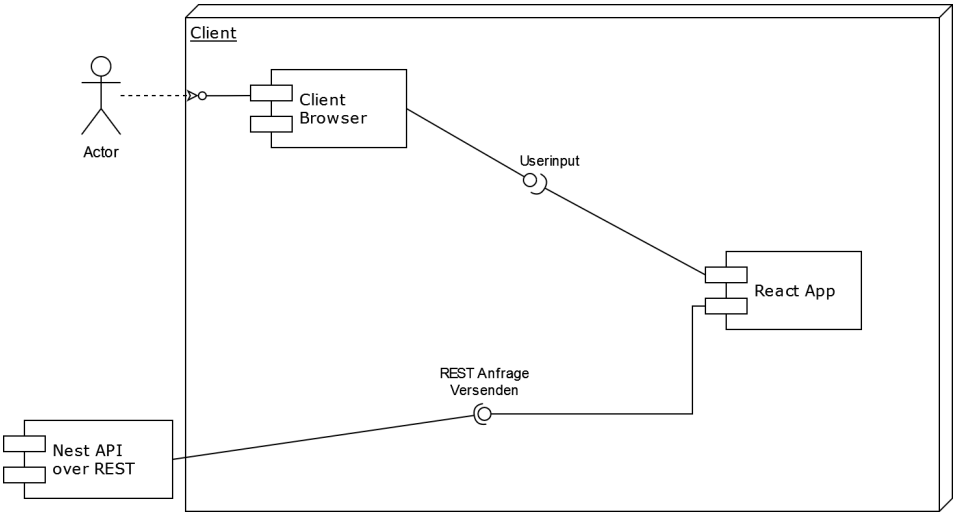

Abbildung 4.1: Component Diagram: Client

In Abbildung 4.1 ist das Komponentendiagramm des Clients zu sehen. Dieses beschreibt die verwendeten Komponenten bei eingegangenem User Input. Der Nutzer macht über den Web-Client eine Anfrage, wie beispielsweise der Aufruf der Web-Applikation oder die Anfrage auf einen bestimmten Raum. Dadurch fordert der Client ein Interface über die React App, welche dann eine REST Anfrage an die Nest API schickt. Im Normalfall sollte dann die REST Anfrage beantwortet werden und die React App kann mit den erhaltenen Informationen ein geeignetes User Interface zur Nutzung auf dem Client bereitstellen.

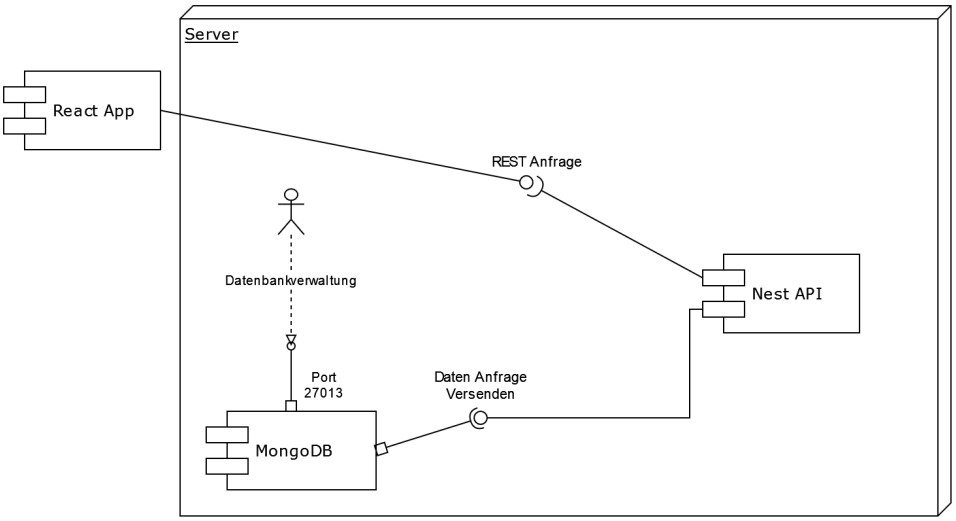

Abbildung 4.2: Component Diagram: Server

In Abbildung 4.2 ist das Komponentendiagramm des Servers zu sehen. Dieses beschreibt die verwendeten Komponenten bei eingegangener REST Anfrage. Die React App stellt eine REST Anfrage an die NEST API, diese fordert dann die verlangten Informationen über ein Interface von der Mongo Datenbank. Die MongoDB ist erreichbar über den Port 27013 durch den Datenbank-Administrator, dieser bearbeitet die Anfrage und sendet die geforderten Daten über das Interface zurück an die Nest API. Zu guter Letzt kann die Nest API dann die bereitgestellten Informationen über ein geeignetes Interface an die React App schicken, welche dadurch weitere Anfragen, wie mithilfe des Komponentendiagramm des Clients beschrieben wurde, beantworten kann.

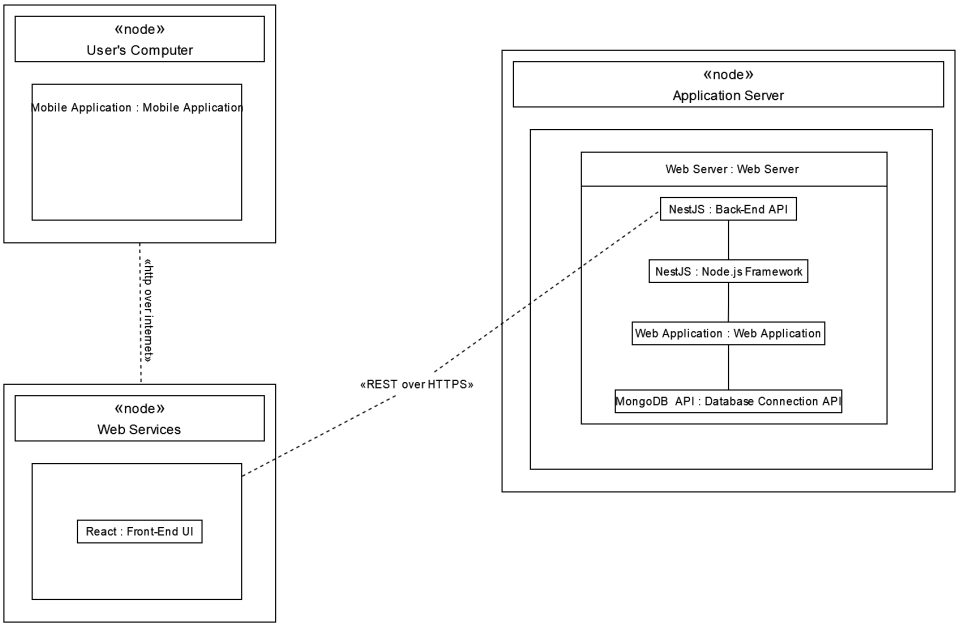

Abbildung 4.3: Verteilungsdiagramm

Das in Abbildung 4.3 gezeigte Verteilungsdiagramm veranschaulicht, wie die Applikation im Browser des Nutzers über das React-Framework an das NestJS backend verbunden ist. Der Nutzer sendet mithilfe der Benutzeroberflächen eine HTTP Anfrage an das React Front-End über das Internet, um zum Beispiel Informationen über Räume anzufordern. Das React Front-End sendet die Anfrage des Nutzers mittels REST Anfrage über HTTPS an das NestJS Backend, damit dies an die Datenbank zugreifen kann. Das NestJS Backend sendet die Anfrage an die Mongoose Datenbank und liefert die geforderten Informationen zurück.

4.3 Datenbanken

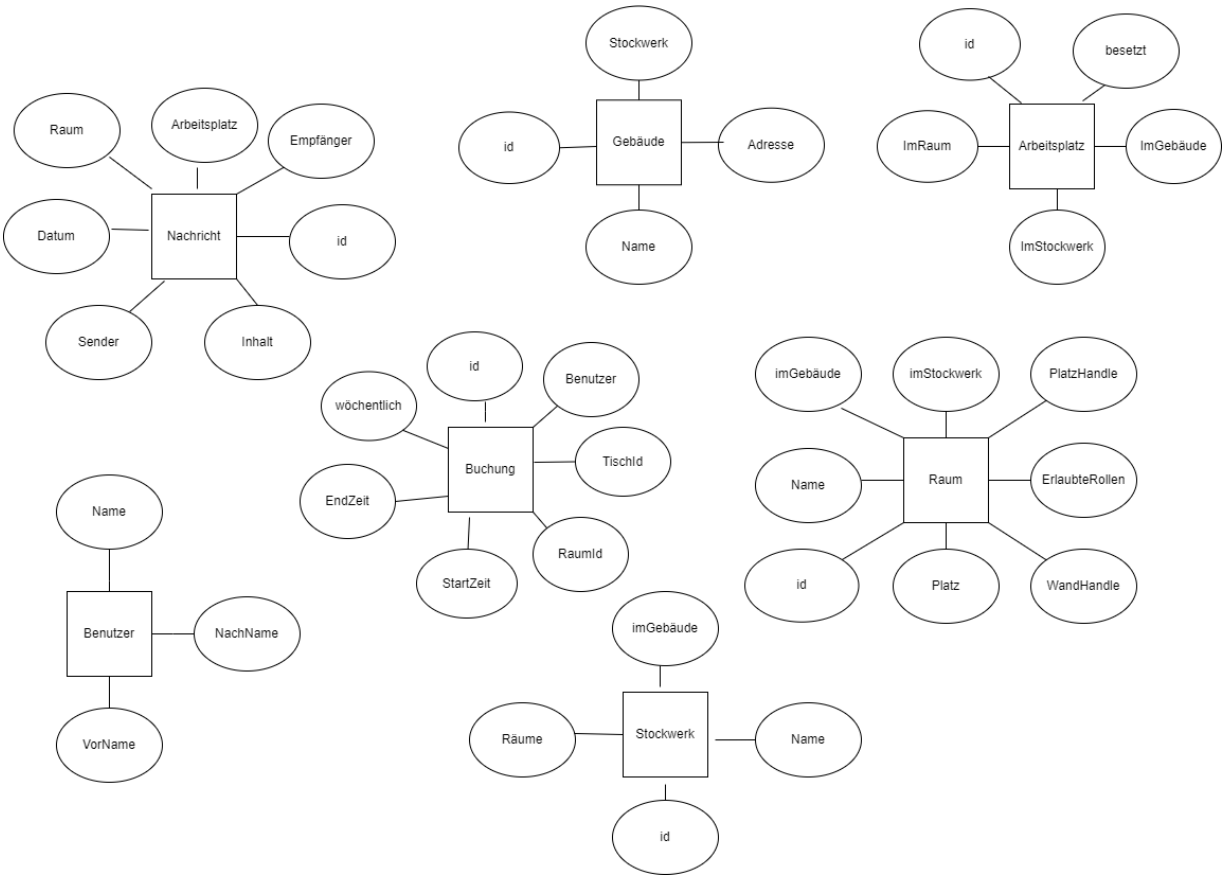

Abbildung 4.4: Skizze-Entity-Relationship-Modell

Das Objekt Benutzer ist die Person die aktiv eine Buchung durchführen will. Die Attribute sind VorName, Name und NachName, die Eigenschaften dieser sind selbsterklärend. Das Objekt Nachricht hat die folgenden Attribute, Platz, Empfänger, Id, Inhalt, Sender, Datum und Raum. Wobei Platz für den Ort der gebucht werden soll, Empfänger für die Person die diese Nachricht erhalten soll, Id die für die Nachrichtenidentifikation, der Inhalt für den Nachrichteninhalt, der Sender für den Verfasser der Nachricht, das Datum für den Zeitpunkt an dem die Nachricht versendet worden ist, Der Raum für die entsprechende Räumlichkeit, verwendet werden sollen.

Das Objekt Gebäude hat die Attribute id, Stockwerk, Adresse und Name. Erstere sind selbsterklärend, das Letzte beschreibt den internen Namen.

Das Objekt Arbeitsplatz, hat die Attribute, Id, besetzt, ImGebäude, ImStockwerk und ImRaum. Mit dem Attribut besetzt ist der Status des Arbeitsplatzes gemeint, ja nach Wert “true“ oder “false“ kann der Momentane Status abgefragt werden. Die restlichen sind selbsterklärend.

Das Objekt Buchung hat die Attribute hat die Attribute, Id, Benutzer, TischId, RaumId, StartZeit, EndZeit und wöchentlich. Mit Benutzer ist die Person, die sich erfolgreich für diesen einen Arbeitsplatz angemeldet hat, gemeint. Mit StartZeit und EndZeit wird der Zeitpunkt und die Dauer der Buchung vermerkt. Das Attribut wöchentlich beschreibt, ob es sich hierbei um eine Dauerbuchung handelt. Die restlichen sind selbsterklärend.

Das Objekt Raum hat die Attribute, imStockwerk, PlatzHandle, ErlaubeRollen, WandHandle, Platz, id, Name und imGebäude. Mit dem Attribut PlatzHandle und WandHandle werden die interaktiv über das UI Setzbaren Parameter definiert. ErlaubeRollen beschreibt welche Rollen in dem Raum erlaubt sind. Für Name siehe oben. Die restlichen sind selbsterklärend.

Das Objekt Stockwerk hat die Attribute, imGebäude, Name, id, Räume. Alle Attribute sind selbsterklärend oder können durch obige Beschreibungen erschlossen werden.

5 Implementiertes User Interface

Nachfolgend wird auf das tatsächlich implementierte User Interface eingegangen. Dabei werden Unterschiede und Abweichungen zum Entwurf aufgezeigt und das generelle Design wird gezeigt. Die Bedienung des User Interfaces bleibt dabei größtenteils gleich, somit ist die Bedienungsanleitung des UI Entwurfs nach wie vor maßgebend. Es wurden hauptsächlich gestalterisch andere, meist bessere Lösungen gefunden.

5.1 Unterschiede beim generellen Design

Abbildung 4.1 zeigt das generelle Design der Buchen Seite.
Dabei sind die Dropdownmenüs aus der oberen Appleiste nach unten gerutscht und wurden visuell ruhiger und simpler gestaltet.
In der oberen Appleiste rechts, wurden noch zwei Buttons hinzugefügt. Der rechte Button zum Auslogen aus dem System und der linke Button um zu dem Raumeditor zu gelangen, falls ein Admin diesen betätigt

In der unteren Auswahlleiste wird die ausgewählte Seite, spricht das Symbol nun nicht wie im Entwurf in schwarz und mit Balken unterhalb hervorgehoben, sondern in einfachen hellblau.

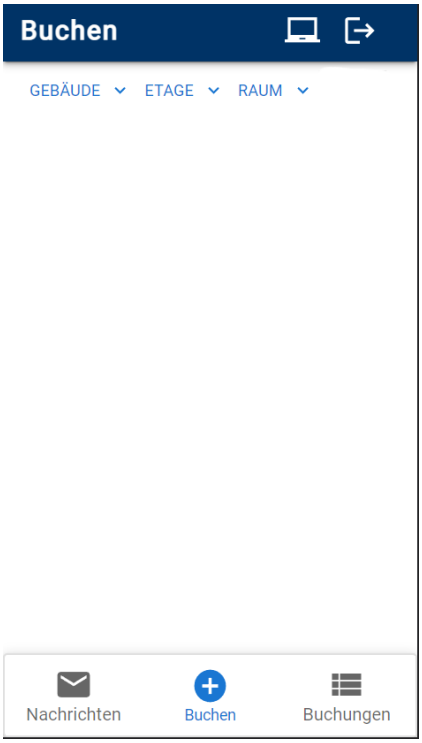

Abbildung 5.1: User Interface: Buchen

In der Chronik der selbst getätigten Buchungen hat sich nicht viel verändert, bis auf das Design.
Die Einträge Kacheln haben einen weniger markanten Rand und es wurde der Hinweis hinzugefügt, ob es sich bei der Buchung um eine wöchentliche Buchung handelt. Wöchentliche Buchungen sind fortlaufende Abonnement Buchungen, die jede Woche sich am selben Tag wiederholen und dabei denselben Zeitraum und Arbeitsplatz haben.

Bei der persönlichen Nachrichtenübersicht hat sich ebenfalls nur das Kacheldesign geändert und ist damit angepasst an die Kacheln der Buchungschronik. Bei den Direktchats blieb alles beim Alten.

Abbildung 5.2: UI: Buchungen

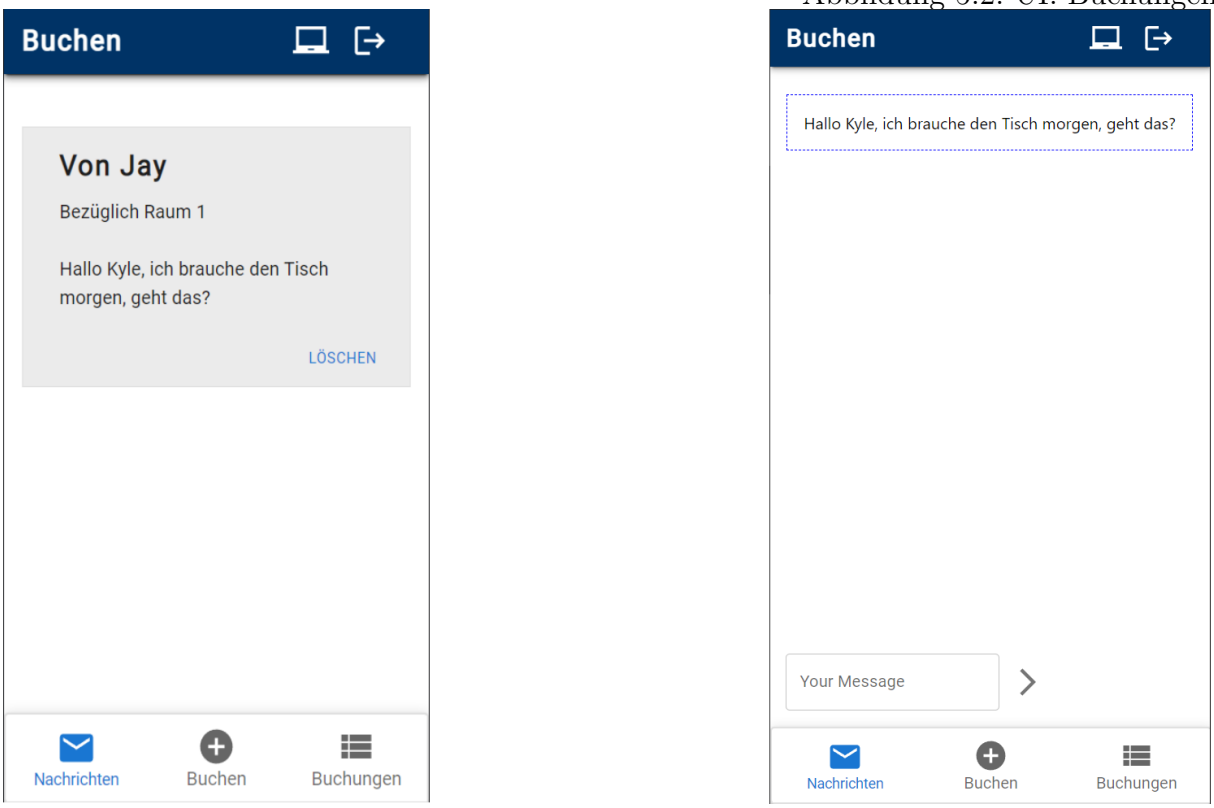

Abbildung 5.3: User Interface: Nachrichten

5.2 Unterschiede des Zeitauswahl Pop-Ups

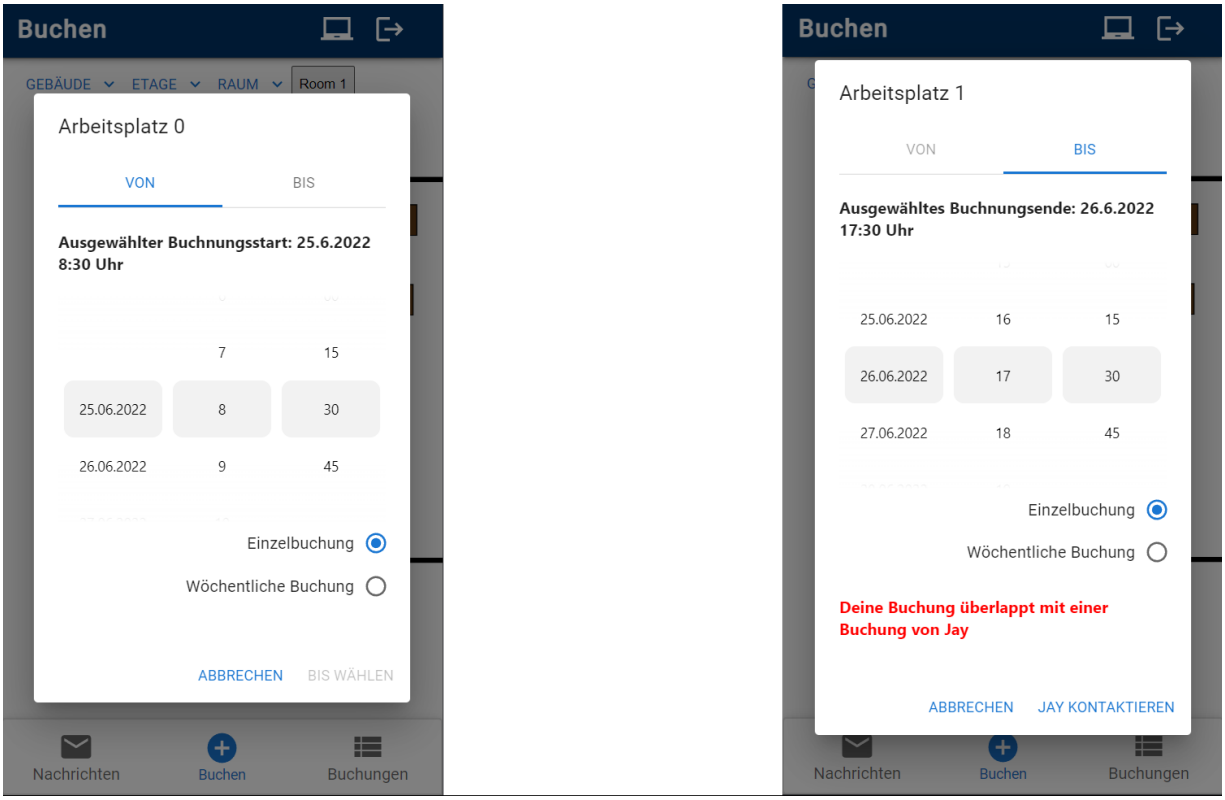

Abbildung 5.4: User Interface: Zeitauswahl Pop-Up

Der Aufbau des Zeitauswahl Pop-Ups ist im Grunde noch gleich, es wurden nur ein paar Regelungen festgelegt beim Eingeben des Zeitraumes und der Überprüfung der Buchung auf Kollisionen mit anderen Buchungen.

Zunächst wurde unterhalb der Tabs, Text hinzugefügt, um verständlicher zu machen, was gerade für eine Zeit eingegeben wird. Dort wird angezeigt, ob gerade der Buchungsstart eingegeben wird oder das Ende, sowie die aktuell gewählte Zeit.

Neu mit dazu kamen auch Optionen, eine Dauerbuchung oder Einzelbuchung zu tätigen mit den Radiobuttons Einzel- oder Wöchentliche Buchung.

Es wurde nun auch beschränkt, dass nach dem Eingeben einer Endzeit, mit Klicken auf den BIS Tab, die Startzeit nicht mehr verändert werden kann. Dies sollte für mehr Linearität und Klarheit beim Buchungsprozess sorgen.

Die Kollisionserkennung von Buchungszeiträumen wurde nun wesentlich anders gelöst, wie im Entwurf.
Nun wird nach Überschneidungen von verschiedenen Buchungen nämlich erst mit Absenden der Buchung gecheckt, nämlich mit dem Klicken auf den Buchen Button. Die im Entwurf gewählte Darstellung mit Färben der Daten und Zeiten, bot letztlich wenig Übersicht über andere Buchungen. Dies kam zustande, weil die Daten oberhalb und unterhalb der ausgewählten mittleren Zeitlinie nicht Anschluss- oder vorangegangene Zeitschlitze sind, sondern immer um einen Tag und eine Stunde vom nächsten Zeitschlitz entfernt sind. Somit könnte nur die mittlere Zeile eingefärbt werden und deshalb wurde dieses Feature eingestellt, weil es zu mehr Verwirrung als Nutzen führen könnte.

Nun wird eben mit Klicken des Buchen Buttons ein Hinweistext angezeigt, welcher darauf hindeutet, dass persönlich eine andere Buchung in dem Buchungszeitraum vorliegt oder dass ein Kollege dort eine Buchung hat. Im Falle der eigenen Buchung bleibt nur die Option Abbrechen, während bei einer Buchung von einem Kollegen man diesen auch anschreiben könnte.

5.3 Räume und Raumeditor

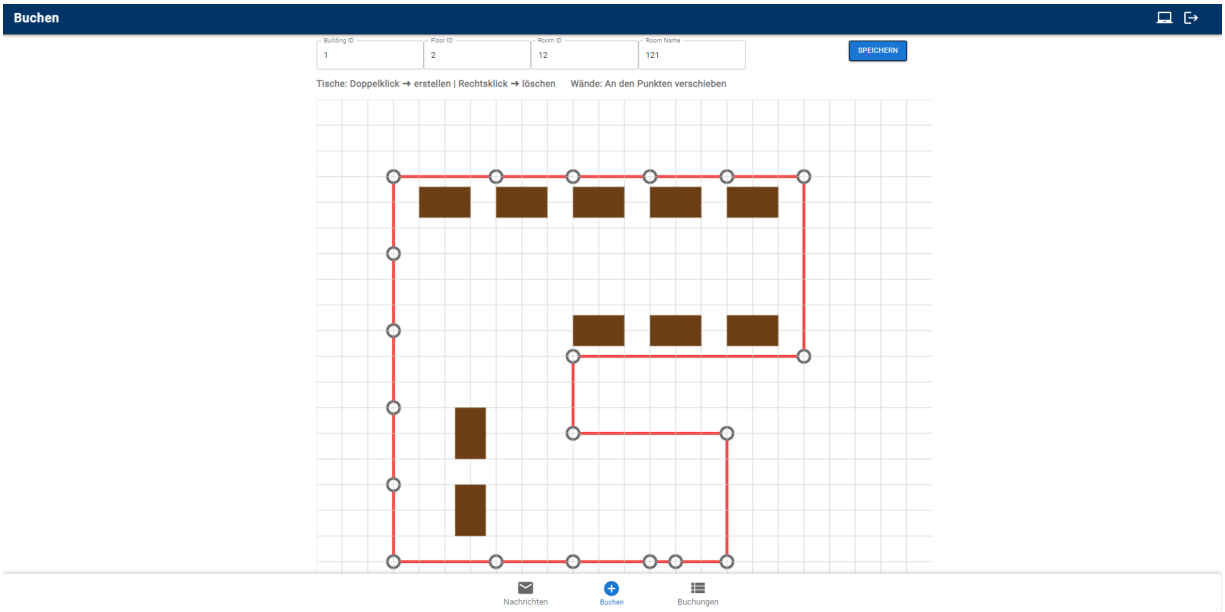

Abbildung 5.5: User Interface: Raum Editor

Abbildung 4.5 zeigt den Stand des Raumeditors.
Der Raumeditor besitzt in der aktuellen Implementierung eine feste Anzahl an Raum-Eckpunkten, die sich verschieben lassen. Mit diesen Raumpunkten sind schon einige Räume nachbildbar und die feste Anzahl erleichterte uns die Implementierung des Editors.
Tische sind nun mit einem Doppelklick auf der Bearbeitungfläche erzeugbar. Diese sind verschiebbar und rotierbar. Diese Erstellung der Tische ermöglich einen schnelleren Workflow, da nicht, wie im Entwurf, immer zuerst die Schaltfläche für einen Tisch gedrückt werden muss. Der Speicher Button ist nun oberhalb des Zeichenfeldes.
Ebenfalls ist es nun möglich zwischen dem Raumeditor und dem restlichen Deskplanner hin und her zu wechseln, durch die Einblendung der oberen Appleiste und unteren Navigationsleiste.

Weitere Features des Raumeditors, die im Entwurf vorhanden sind und in der Implementierung nicht, werden im Ausblick der Dokumentation behandelt.

6 Finalisierende Ausführungen

In diesem letzten Kapitel werden Fazits bezüglich der eigenen Lernfortschritte in Bezug auf das Projekt als auch die Nutzung der Software thematisiert.

6.1 Installations- und Administrationshandbuch

In dieser Sektion werden das Installations- und Administrationshandbuch unterteilt.

6.1.1 Installationshandbuch

In diesem Installationshandbuch werden die einzelnen Schritte aufgeführt, die zur Installation der Software benötigt werden.

- *Docker Desktop* mit WSL Command Line installieren (siehe hierzu Docker Desktop Installationsanleitung)
- *Node.js* installieren und das System neustarten
- Eingabe in eine Kommandozeile: *docker pull mongo*
- In Docker Desktop nun das Image *mongo* starten

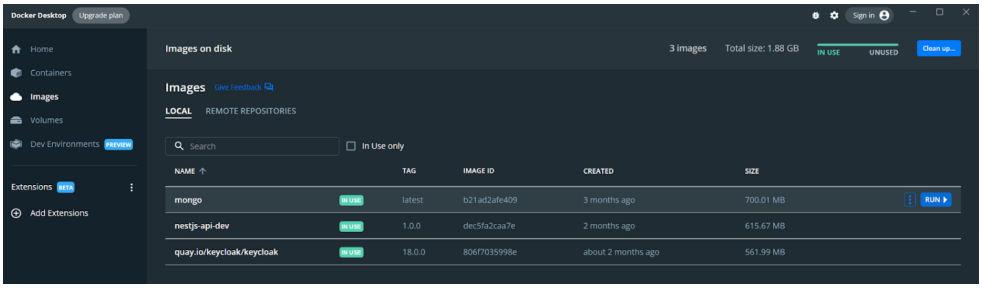

Abbildung 6.1: Das Mongo Image highlighted in Docker Desktop

- Bei der Konfiguration von Mongo in Docker Desktop muss unter dem Punkt *Local Host* in *Ports* folgendes eingegeben werden: *27017*
- Anschließend auf *Run* klicken

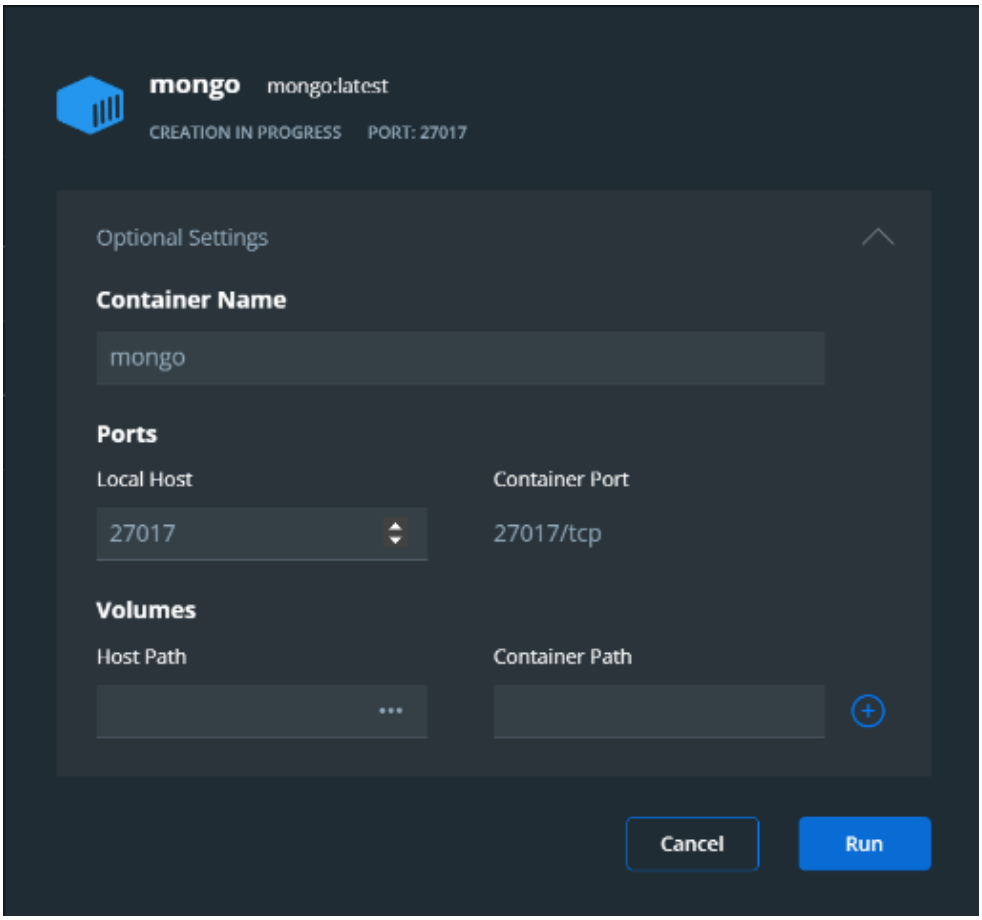

Abbildung 6.2: So sollte die Konfiguration von Mongo aussehen nach dem letzten Schritt

- Anschließend muss Keycloak gestartet werden, dazu muss in eine Kommandozeile folgendes eingegeben werden: *docker run -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:18.0.2 start-dev*
- Nun in Docker Desktop unter dem Punkt *Containers* den *Keycloak* Container (erkennbar am Port 8080) und den *mongo* Container starten
- Im Browser kann dann die Route *localhost:8080* aufgerufen werden und mit dem Nutzernamen *admin* und Passwort *admin* kann sich eingeloggt werden
- Aus dem GitHub Repository dann die Datei *realm-export.json* herunterladen
- Unter *Add Realm* eine Realm namens *DeskPlanner* erstellen
- Auf der linken Seite unter *Import* die zuvor heruntergeladene Datei *realm-export.json* auswählen
- Unter *existierenden Ressourcen Skip* auswählen und importieren

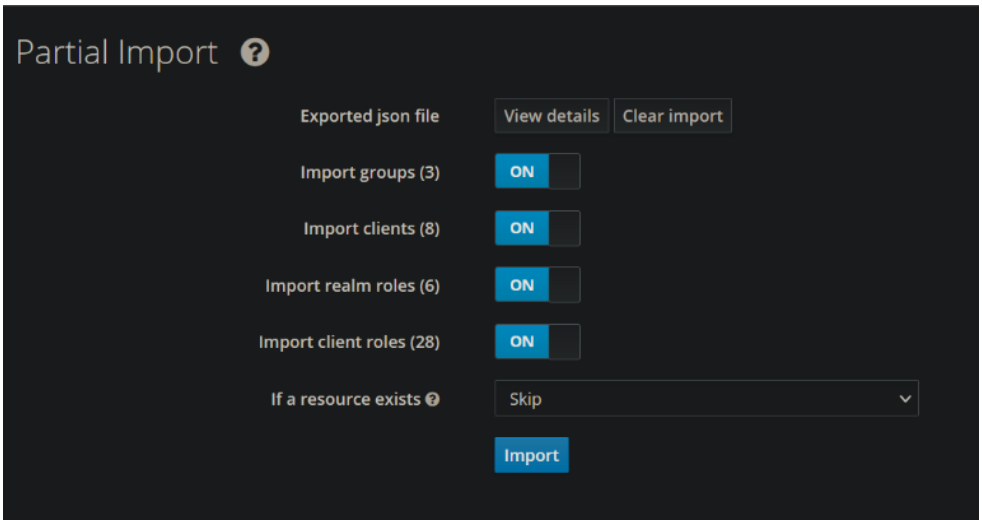

Abbildung 6.3: So sollte der Import aussehen

- Unter *Users* einen neuen User anlegen mit *Add User*
- Anschließend die persönlichen Daten der Nutzer (*Usernamen, Vor- und Nachnamen*) eingeben und speichern
- Nun die User anklicken, dazu muss gegebenenfalls *View all users* angeklickt werden
- Hier kann unter *Credentials* ein Passwort hinzugefügt werden

  Wichtig: Das Passwort muss als *nicht-temporär* gespeichert werden
- In *Role Mappings* im Dropdown *Client Roles* muss *keycloak-reactjs-demo* ausgewählt werden
- Hier können die gewünschten Rollen *(admin, mitarbeiter, ...)* gewählt werden
- Anschließend muss in einem weiteren Konsolenfenster in den Ordner *nestjs-desk-planner* mithilfe von *cd* gewechselt werden
- Nun folgenden Befehle eingeben und ausführen lassen: *npm install*
- Nach einem erfolgreichen ersten Befehl den nächsten: *npm run start*
- In der Umgebungsentwicklung in der Kommandozeile folgenden Befehl eingeben: *cd ./react-user-app/*
- Nun folgende Befehle eingeben und ausführen lassen: *npm install –force* und anschließend *npm start*

Nach erfolreichem Durchführen der Installationsanleitung sollte der DeskPlanner erfolgreich eingerichtet sein und kann anschließend benutzt werden. Im Administrationshandbuch finden Sie weitere Informationen zur Konfiguration und zur Benutzung des RaumEditor für den Administrator.

6.1.2 Administrationshandbuch

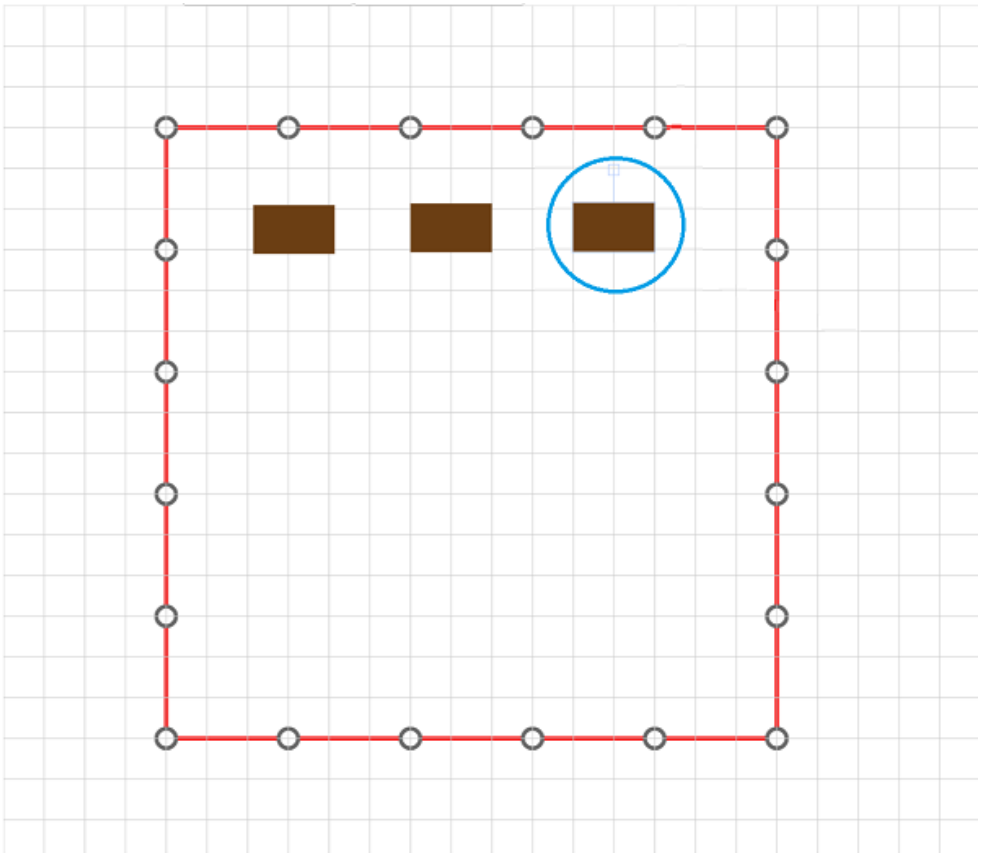

Abbildung 6.4: Layout Designer - Administrator

Als Administrator des IT-Systems ist seine hauptsächliche Aufgabe die Verwaltung von Gebäuden, Stockwerken und Räumen im Layout Designer. In diesem kann der Administrator einen zuvor definierten Bereich anhand der weißen Kugel-Komponenten editieren, um das richtige Raumlayout mit den eingezeichneten roten Linien zu erstellen. Außerdem ist es möglich, mit einem Doppel-Klick eine braune Tisch-Komponente (in der Abbildung blau gekennzeichnet) zu erstellen und diesen beliebig im Raum zu verteilen. Diese Tische können ebenfalls mit einem Rechts-Klick gelöscht werden. Für den Administrator befindet sich ein kleines Tutorial im Layout Designer, welcher die Funktionsweise des Layout Designers erklärt. Hier kann der Administrator den Raum erstellen und angeben, wo dieser sich befindet, um den Arbeitnehmern im Buchen Bereich vorgefertigte Räume zu geben, welche dann gesucht und gebucht werden können.

6.2 Reflektion: Aufteilung des Teams

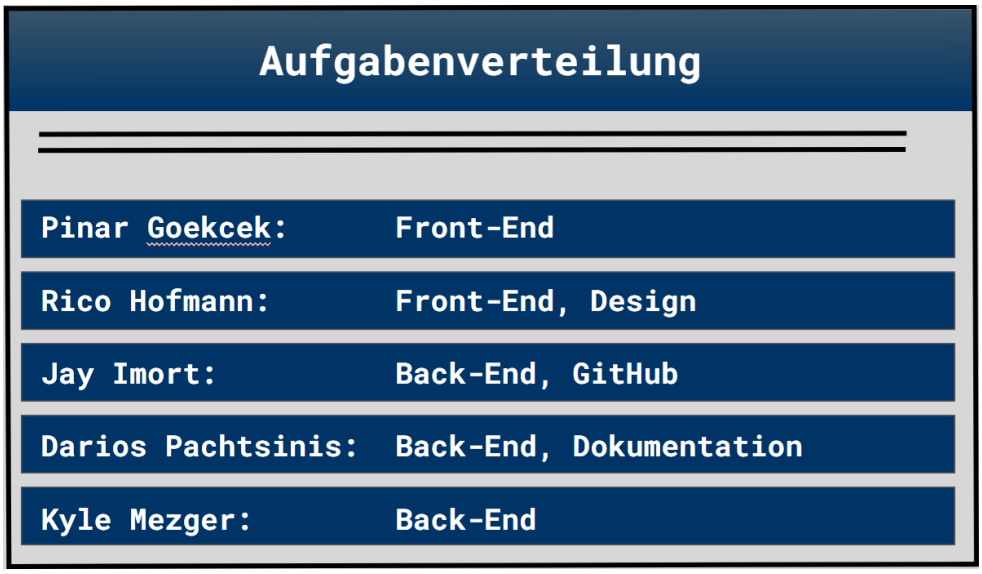

Abbildung 6.5: Aufgabenverteilung - Erste Präsentation

In der Abbildung zu erkennen, ist die erste Aufgabenverteilung im Vorfeld des Projekts. Die Aufteilung fand grob in Front-End und Back-End statt, mit kleinen Aufgabenspezifikationen wie Design, GitHub und Dokumentation. Diese Aufteilung beruht auf vergangene Projekte im Rahmen des Studiums.

Nachdem das Projekt begonnen wurde, wurde die Aufteilung weiter ausgebaut. Im Vorfeld war klar, dass Rico Hofmann als Front-End Entwickler fungiert, da seine vergangenen Aufgaben rund um das Thema Design vorangegangene Projekte prägten. Rico Hofmann ist somit der Lead Designer des Teams, denn er baute das Grundgerüst für die Design-Aufgaben, welche später von Darios Pachtsinis und Pinar Gökcek erweitert und vollendet wurden. Des Weiteren ist Rico der Experte für Material UI und führte außerdem auch das Protokoll für Teambesprechungen mit unserem Betreuer.

Kyle Mezger ist einer der Back-End Entwickler, welcher sich zur Aufgabe machte, die REST API zu erstellen und auszubauen und den Login via Keycloak zu verwalten. Er hatte sich allerdings auch um die Kommunikation mit dem Betreuer als auch für die wöchentliche Aufgabenverteilung innerhalb der Gruppe via Issues in GitHub gekümmert.

Jay Imort hingegen ist der andere Back-End Entwickler, welcher seinen Fokus hauptsächlich auf Nest.js, Routing legt. Des Weiteren beschäftigte er sich mit dem Layout Designer, in welchem der Benutzer einen Raumlayout erstellen, bearbeiten und in die Datenbank laden kann.

Die Aufgaben im Back-End Bereich wurden meist vermischt und deswegen teilten sich Kyle Mezger und Jay Imort die Aufgaben gerecht und deren Fachwissen entsprechend auf.

Darios Pachtsinis wurde ursprünglich dem Back-End Team zugeteilt, allerdings hatte das Team mehr Designer gebraucht, weswegen sein Aufgabenfeld doch zum Front-End zugeteilt worden ist. Er hatte Aufgaben erledigt wie das Finalisieren von Design Entscheidungen, hauptsächlich rund um den Buchungen Bereich (Booking, etc.).

Pinar Gökcek ist ebenfalls im Front-End zuständig gewesen und hatte Design Entscheidungen von Rico Hofmann finalisiert. Außerdem erledigte sie Aufgaben speziell im Nachrichten Bereich (Messages, etc.), wie das Empfangen von Nachrichten, als auch dessen Verknüpfung mit der Datenbank.

Zu guter Letzt jedoch war das Team kein Zusammenschluss von 5 Einzelpersonen mit Fachwissen in besonderen Bereich, sondern das Team unterstützte sich gegenseitig, sodass jeder auch außerhalb des zuvor genannten Aufgabenbereichs Aufgaben erledigen konnte. Dies bedeutet, dass das Team sich immer helfen konnte, sich gegenseitig motiviert hat und auf die Erledigung der Aufgaben anderer achtete und tatkräftig unterstützt hatte. Die Aufgabenverteilung ist somit grob und soll verdeutlichen, dass die Übergänge zu anderen Aufgabenbereichen und Teammitgliedern vermischen.

6.3 Reflektion: Projektmanagement

Als Projektmanagement wurde eine Mischung aus Kanban und Scrum, auch Scrumban, genutzt. Wie im Voraus erwartet, hatte uns das Kanban Board eine große Hilfe geleistet beim Überblicken der Projektaufgaben. Das Kanban Board wurde mit einem von GitHub bereitgestellten Issue-System verbunden, welches wir nutzten, um größere Aufgabenteile zu unterteilen und diese einem Mitglied zuzuweisen.

Mithilfe der wöchentlichen Meetings konnten wir uns gut koordiniert auf die Aufgaben des Boards konzentrieren und diese auch größtenteils bewältigen. Wir setzten unser wöchentliches Meeting auf den Dienstag, um den Montag mit unserem Betreuer zu nutzen. Die Vorstellung unseres jeweiligen Standes, gab uns mindestens eine weitere Expertenmeinung, welche wir für unser Gruppenmeeting nutzten. Um diese Expertenmeinung bestmöglich zu nutzen, haben wir ebenfalls für jedes Treffen ein Protokoll geführt, welches wir direkt in unsere Aufgabenbesprechungen untergebracht hatten.

6.4 Reflektion: Lernfortschritt

Eine elementare Schnittstelle zwischen den Einzelentwicklern sind Softwareentwicklungsprojekte zur Versionsverwaltung.
Im Rahmen unsers Projektes haben wir zum (Merg'en, Pull'en, Push'en, Commit'en etc.) ausschließlich GitHub verwendet. Solche Programme sind unerlässlich, um progressive Fortschritte zu vermerken, dies haben wir besonders im Rahmen dieses Projektes erfahren.

Wir haben durch eigenständige Einarbeitung und gegenseitigen Austausch, unsere Fähigkeiten in den Scriptsprachen JavaScript bzw./und TypeScript ausgebaut. Verstanden was das Front und Backend macht und diese entwickelt und z. B. React und REST API Anbindungen im selbigen Bereich implementiert. Wir haben gelernt welche Schritte notwendig sind, um die Architektur zur Projektumsetzung zu designen. Wir haben gelernt wie eine Benutzeroberfläche mit der Bibliothek Material UI zu entwerfen ist.

Zur Termingerechten Fertigstellung unseres Vorhabens haben wir mit Milestones und Issues gearbeitet. Uns ist nun klar, dass trotz sorgfältiger Planung immer Puffer einzuplanen sind, um realistische Fertigstellungstermine zu prognostizieren. Wir sind uns alle aber dennoch einig, dass das wichtigste ein belastbares, kompetentes und kommunikatives Team ist.

Trotz der hohen Güte des Teams ist ein sauberes Datenbankmodell Elementar, dies haben wir durch ein Negativbeispiel am eigenen Leib erfahren. Wir haben aus diesem Grund ein Freihand Datenmodell nutzen müssen. In Zukunft werden wir bereits im Vorfeld auf die systematische und einheitliche Namensvergabe der Variablen achten. Dadurch kann allein durch den Variablennamen direkt auf Eigenschaft zurückgeschlossen werden. Dies führt zu erheblicher Zeitersparnis bei "Fremd-Codes".

6.5 Lizenzen

Wir haben uns ausschließlich für Open Source Lizenzen entschieden. Der Grund ist zum einen, dass auch unsere Anwendung später als Open Source vermarktet werden soll und zum anderen die folgenden generellen Vorteile dieser Lizensierung.

- Verbreitung kostenlos
- Entwicklungstempo hoch (bei großer Community)
- Bugs im Code werden schneller gefunden (Mehraugen Prinzip)
- Direkte Modifikation des bestehenden Codes (zur optimalen Eigennutzung)
- Hohe Qualität, da hoher Qualitätsdruck für Entwickler (jeder kann jede Änderung auf den Einzelnen Nachverfolgen)

- Meistens gut dokumentiert
- Hohe Innovationspotential (keine Firmenvorgaben etc.)

Wie bei allen Dingen im Leben gibt es natürlich auch hier die Kehrseite der Medaille, im Rahmen unseres Projektes haben die Vorteile aber überhand. Wir haben folgende Bibliotheken mit der entsprechenden Lizenz verwendet:

- MIT Lizenz:
  - Material UI
  - react-native-date-picker
  - Fabric
  - React
  - Nest
- Apache Lizenz 2.0:
  - Keycloak
  - MongoDB

6.6 Ausblick

Um das Gesamtprojekt zu optimieren können noch folgende Features aus den Mockups implementiert werden. Darunter Zählen, die Modifizierbarkeit der Buchungsstartzeit nach dem setzten der Endzeit. Die Darstellung der vorhandenen Räume in der kategorisierten Übersicht. Die Möglichkeit zur Rollenzuweisung der Rechte in einem bestimmten Raum, um gültige Buchungen zu setzen. Ein großer Vorteil würde mit der Live-Chat Funktion einhergehen, da die Kommunikation auf dem direkten Weg immer der schnellste Weg zur Lösungsfindung ist. Durch diese Interaktion wäre die Webapplikation auch attraktiver.