



# **FOM Hochschule für Oekonomie & Management**

Hochschulzentrum Münster

## **Bachelor-Thesis**

im Studiengang Wirtschaftsinformatik

zur Erlangung des Grades eines

**Bachelor of Science (B.Sc.)**

über das Thema

### **Steuerung und Verwaltung von Servicerobotern**

Konzeption und prototypische Implementierung einer Webanwendung

von

**Leopold Pinkernell**

Erstgutachter: Prof. Dr. Jannik Hüls

Matrikelnummer: 564638

Abgabedatum: 19. März 2024

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>V</b>
<b>Tabellenverzeichnis</b>	<b>VI</b>
<b>Abkürzungsverzeichnis</b>	<b>VII</b>
<b>Glossar</b>	<b>VIII</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Hintergrund und Motivation . . . . .	1
1.2 Zielsetzung und Forschungsfrage . . . . .	2
1.3 Methodik . . . . .	2
1.3.1 Design Science Research nach Hevner . . . . .	3
1.3.2 Design Science Research im Rahmen der Arbeit . . . . .	4
<b>2 Grundlagen</b>	<b>5</b>
2.1 Serviceroboter . . . . .	5
2.1.1 Definition . . . . .	5
2.1.2 Einsatzmöglichkeiten . . . . .	5
2.1.3 Pudu Robotics . . . . .	6
2.1.4 Bot Control Backend . . . . .	6
2.2 Webanwendungen . . . . .	7
2.2.1 Technologien und Softwarebibliotheken . . . . .	8
2.2.2 chayns . . . . .	8
2.3 3D Modelle . . . . .	9
2.3.1 Generierung . . . . .	9
2.3.1.1 Fotogrammetrie . . . . .	10
2.3.1.2 LiDAR Scanning . . . . .	10
2.3.1.3 KI gestützte Methoden . . . . .	10
2.3.2 Einbindung im Web . . . . .	11
2.3.2.1 WebGL und WebGPU . . . . .	11
2.3.2.2 deck.gl . . . . .	11
2.4 Softwarequalität . . . . .	12
2.4.1 Benutzerfreundlichkeit . . . . .	13
2.4.1.1 Usability Heuristics . . . . .	14
2.4.1.2 Usability Tests . . . . .	14
2.4.2 Effizienz . . . . .	15

<b>3 Anforderungen an den Prototyp</b>	<b>16</b>
3.1 Anforderungen an Modellerzeugung . . . . .	16
3.2 Funktionale Anforderungen . . . . .	16
3.2.1 Übersicht . . . . .	16
3.2.2 Steuerung . . . . .	17
3.2.3 Verwaltung . . . . .	17
3.2.4 Benutzer . . . . .	17
3.3 Nicht-funktionale Anforderungen . . . . .	18
<b>4 Technische Herausforderungen</b>	<b>19</b>
4.1 Methode zur Modellgenerierung . . . . .	19
4.2 3D Visualisierung im Web . . . . .	19
4.2.1 deck.gl . . . . .	19
4.2.2 Dateiformat der 3D-Modelle . . . . .	20
4.3 Synchronisierung des Gebäudemodells und der Roboterdaten . . . . .	21
<b>5 Umsetzung des Prototyps</b>	<b>22</b>
5.1 Mockup . . . . .	22
5.2 Implementierung . . . . .	23
5.2.1 Übersicht . . . . .	23
5.2.1.1 Gebäudemodelle . . . . .	23
5.2.1.2 Roboterdaten . . . . .	25
5.2.1.3 Echtzeit-Aktualisierung . . . . .	28
5.2.1.4 Interaktion . . . . .	28
5.2.2 Steuerung . . . . .	29
5.2.3 Verwaltung . . . . .	30
5.2.4 Editiermodus . . . . .	32
5.3 Softwaretests . . . . .	34
5.4 Deployment . . . . .	34
<b>6 Evaluierung des Prototyps</b>	<b>36</b>
6.1 Funktionale Anforderungen . . . . .	36
6.2 Benutzerfreundlichkeit . . . . .	37
6.2.1 Usability Entscheidungsregeln . . . . .	37
6.2.2 Usability Tests . . . . .	39
6.2.2.1 Erster Testdurchlauf . . . . .	39
6.2.2.2 Zweiter Testdurchlauf . . . . .	41
6.3 Effizienz . . . . .	43
6.3.1 Effizienzmesswerte . . . . .	44

6.3.2 Bewertung der Effizienz . . . . .	45
<b>7 Diskussion</b>	<b>46</b>
7.1 Erfüllung der Anforderungen . . . . .	46
7.2 Einsatz der Technologien . . . . .	48
7.3 Bewertung der Methodik . . . . .	49
<b>8 Fazit</b>	<b>50</b>
8.1 Nutzung des Prototyps . . . . .	50
8.2 Ausblick . . . . .	50
<b>Anhang</b>	<b>52</b>
<b>Literaturverzeichnis</b>	<b>58</b>

## Abbildungsverzeichnis

Abbildung 1:	Design Science Research nach Hevner . . . . .	3
Abbildung 2:	Kommunikation zwischen Bot Control Backend und Robotern . . . . .	7
Abbildung 3:	IconLayer Beispiel . . . . .	12
Abbildung 4:	Qualitätsmerkmale . . . . .	13
Abbildung 5:	Raummodell ohne und mit Backface Culling . . . . .	25
Abbildung 6:	Kartendarstellung . . . . .	27
Abbildung 7:	Übersicht . . . . .	27
Abbildung 8:	Kommunikationsweg von Statusaktualisierungen der Roboter . . . . .	28
Abbildung 9:	Steuerung und Routenplanung . . . . .	30
Abbildung 10:	Verwaltung . . . . .	31
Abbildung 11:	Editiermodus . . . . .	33
Abbildung 12:	Bestätigungsdialog . . . . .	38
Abbildung 13:	Effizienzmesswerte . . . . .	44
Abbildung 14:	Mockup der Übersicht . . . . .	52
Abbildung 15:	Mockup der Steuerung . . . . .	53
Abbildung 16:	Mockup des Routenplanungs-Popup . . . . .	54
Abbildung 17:	Mockup der Verwaltung . . . . .	55

## Tabellenverzeichnis

Tabelle 1: Gefundene Probleme in erster Usability Testrunde . . . . .	40
Tabelle 2: Bewertung der durchgeführten Aktionen in erster Usability Testrunde . . . . .	41
Tabelle 3: Gefundene Probleme in zweiter Usability Testrunde . . . . .	42
Tabelle 4: Bewertung der durchgeführten Aktionen in zweiter Usability Testrunde . . . . .	43
Tabelle 5: Beschreibung der Probleme in erster Usability Testrunde . . . . .	56
Tabelle 6: Beschreibung der Probleme in zweiter Usability Testrunde . . . . .	57

## Abkürzungsverzeichnis

<b>API</b>	Application Programming Interface
<b>BA</b>	Bundesagentur für Arbeit
<b>BCB</b>	Bot Control Backend
<b>CSS</b>	Cascading Style Sheets
<b>DSR</b>	Design Science Research
<b>FCP</b>	First Contentful Paint
<b>FID</b>	First Input Delay
<b>glTF</b>	Graphics Library Transmission Format
<b>GPU</b>	Grafikkarte
<b>GUI</b>	Graphical User Interface
<b>HTML</b>	HyperText Markup Language
<b>ID</b>	Identifier
<b>IFR</b>	International Federation of Robotics
<b>INP</b>	Interaction to Next Paint
<b>JSX</b>	JavaScript XML
<b>kB</b>	Kilobyte
<b>KI</b>	Künstliche Intelligenz
<b>LCP</b>	Largest Contentful Paint
<b>LiDAR</b>	Light Detection and Ranging
<b>mB</b>	Megabyte
<b>npm</b>	Node Package Manager
<b>OBJ</b>	Wavefront Object
<b>PIL</b>	Primitive Instancing Layering
<b>Sass</b>	Sassy Cascading Style Sheets
<b>SVG</b>	Scalable Vector Graphics
<b>TBT</b>	Total Blocking Time
<b>UI</b>	User Interface
<b>URL</b>	Uniform Resource Locator
<b>VSLAM</b>	Visual Simultaneous Localization and Mapping
<b>WebGL</b>	Web Graphics Library
<b>WebP</b>	Web Picture Format

## Glossar

**Base64** Base64 ist ein Kodierungsverfahren, das Binärdaten in lesbare ASCII-Zeichen umwandelt und dabei 64 verschiedene Zeichen verwendet, um die Übertragung von Daten über textbasierte Protokolle zu ermöglichen. 26

**DOM** Das Document Object Model ist eine Programmierschnittstelle, die die Struktur und Inhalte eines HTML- oder XML-Dokuments repräsentiert und es ermöglicht, auf diese Elemente zuzugreifen, sie zu ändern und zu manipulieren. 8

**HTTP** Das Hypertext Transfer Protocol ist ein grundlegendes Protokoll für den Datenaustausch im World Wide Web, das die Übertragung von Hypertext-Dokumenten zwischen Webbrowsern und -servern ermöglicht. Es ist durch Zustandslosigkeit, verschiedene Methoden – wie GET und POST –, Statuscodes und die Verwendung von URLs gekennzeichnet. 6, 7

**Microservice** Ein Microservice ist eine isolierte und eigenständige Softwarekomponente, die eine spezifische Aufgabe erfüllt und innerhalb einer verteilten Architektur betrieben wird. 6, 7

**Mixins** Sass Mixins sind wiederverwendbare Code-Ausschnitte in Sass, die dazu dienen, CSS-Eigenschaften und -Werte zu kapseln und sie leicht in verschiedenen Stilen oder Elementen anzuwenden. 8

**Mockup** Ein Mockup ist eine visuelle Darstellung oder Modellierung eines Designs, Produkts oder einer Benutzeroberfläche, die dazu dient, das Aussehen und die Funktionalität zu skizzieren und Feedback zu sammeln, bevor die eigentliche Entwicklung beginnt. 22, 23, 27, 30, 31

**MQTT** Message Queuing Telemetry Transport ist ein leichtgewichtiges Netzwerkprotokoll für die effiziente Nachrichtenübertragung zwischen Geräten, besonders in Machine-to-Machine und Internet of Things Anwendungen. 6

**MQTT-Topic** Ein MQTT-Topic ist eine eindeutige Zeichenkette, die verwendet wird, um Nachrichten innerhalb des MQTT-Protokolls zu adressieren und zu organisieren. Clients nutzen MQTT-Topics, um Nachrichten auszutauschen. 6, 7

**MQTT-Broker** Ein MQTT-Broker ist eine Middleware-Komponente, die als Vermittler fungiert, um Nachrichten zwischen verschiedenen Clients über das MQTT-Protokoll auszutauschen und zu verwalten. 6

**Webhook** Ein Webhook ist eine automatisierte Methode zur Übertragung von Echtzeitinformationen zwischen Webdiensten durch das Senden von HTTP-Anfragen an vordefinierte URLs. Webhooks ermöglichen sofortige Ereignisbenachrichtigungen.  
6, 7

**websocket** WebSocket ist ein Kommunikationsprotokoll, das eine bidirektionale, echtzeitfähige Verbindung zwischen einem Webbrowser und Server ermöglicht. 9, 28, 37

## 1 Einleitung

Der Einsatz von Robotern prägt zunehmend den Arbeitsmarkt und beeinflusst die Art und Weise, wie Unternehmen ihre Prozesse gestalten. Diese Entwicklung beschränkt sich nicht nur auf klassische Einsatzgebiete wie die Industrie, in der beispielsweise Montageroboter eingesetzt werden, und den Verbrauchermarkt, in dem Staubsaugerroboter mittlerweile weit verbreitet sind, sondern zunehmend auch auf die Dienstleistungsbranche. Ermöglicht wird diese Entwicklung durch den technischen Fortschritt in den Bereichen Robotik, Künstliche Intelligenz (KI), Big Data, Kameras, Sensorik und Spracherkennung [1, S. 424].

Der Absatz von Servicerobotern für professionelle Anwendungen verzeichnete laut der International Federation of Robotics (IFR) im Jahr 2022 einen Anstieg um 48% [2, S. 1], während der Absatz von Industrierobotern schwächer zunahm [3, S. 9] und bei Service-robotern im Verbrauchermarkt sogar rückläufig war [3, S. 37]. Dieses Wachstum wird laut der IFR durch eine gesteigerte Nachfrage getrieben, die unter anderem auf einen Mangel an Arbeitskräften zurückzuführen ist [3, S. 33-34]. Laut der Bundesagentur für Arbeit (BA) ist die Menge unbesetzter Arbeitsplätze in Deutschland in den letzten 10 Jahren um 43% gestiegen und – trotz des Rückgangs um 17% in den letzten zwei Jahren – weiter auf einem hohen Stand [4]. Insbesondere in der Gastronomie gibt es einen erheblichen Personalmangel, der zum Teil auf die Corona-Pandemie zurückzuführen ist, da in dieser Zeit viele Angestellte in andere Berufsfelder gewechselt sind. Laut dem Institut der deutschen Wirtschaft, das sich auf Daten der BA bezieht, sind während des Pandemiejahres 2020 circa 216.000 Arbeiter aus dem Gastgewerbe in ein anderes Berufsfeld gewechselt [5, S. 1].

Serviceroboter bieten durch eine effiziente Unterstützung des Personals die Möglichkeit den Personalmangel zu mitigieren. So ersetzen Serviceroboter das Personal meistens nicht vollständig, sondern unterstützen es, sodass es mehr Zeit für andere Aufgaben wie die Kundenbetreuung hat [6, S. 271-272]. In der Gastronomie können Lieferroboter beispielsweise bestimmte Kellneraufgaben übernehmen und so das Personal entlasten.

### 1.1 Hintergrund und Motivation

In diesem Kontext hat sich die Firma Tobit Laboratories AG entschieden, die Potenziale von Lieferrobotern für eigene Gastronomiebetriebe zu erkunden. So hat das Unternehmen mehrere Lieferroboter von Pudu Robotics erworben, um diese Technologie zu testen.

Zur Steuerung der Roboter aber auch zur Erweiterung der Funktionen wurde das Bot Control Backend (BCB) entwickelt, das im Abschnitt 2.1.4 näher erläutert wird. Zur Prüfung der Eignung sollen die Roboter zunächst im Firmengebäude für kleinere Botengänge eingesetzt werden. Hierfür müssen die Roboter zum einen auf verschiedene Aspekte, wie die Navigationsfähigkeit und Zuverlässigkeit geprüft werden. Zum anderen muss eine Anwendung entstehen, mit der die Roboter vor allem intuitiv gesteuert und übersichtlich verwaltet werden können. Abhängig von den Ergebnissen könnten die Roboter dann möglicherweise in Gastronomiebetrieben eingesetzt werden.

## 1.2 Zielsetzung und Forschungsfrage

An diesem Punkt wird in dieser Arbeit angesetzt. Es soll eine Anwendung konzipiert und prototypisch implementiert werden, die die Steuerung und Verwaltung von Lieferrobotern ermöglicht und zudem eine Übersicht über deren Positionen bietet. Bei der Anwendung soll es sich um eine Webanwendung handeln, da Webanwendungen im Gegensatz zu native Anwendungen eine plattformunabhängige Nutzung und sofortige Verfügbarkeit ohne Installation bieten [7, Abschnitt 2]. Für eine möglichst übersichtliche Darstellung der Roboterpositionen sollen 3D-Modelle der Räume genutzt werden. Um eine reibungslose Integration der Roboter in Gastronomiebetrieben zu ermöglichen, soll die Erstellung der 3D-Modelle mit minimalem Aufwand verbunden und unkompliziert sein. Hierbei gilt es zu beachten, dass die Erzeugung der 3D-Modelle nicht in dem zu entwickelnden Prototyp integriert werden muss. Trotz des erhöhten Rechenaufwands, der mit der Darstellung von 3D-Modellen verbunden ist, soll die Anwendung außerdem performant sein.

Aus dieser Zielsetzung ergibt sich die folgende Forschungsfrage: Wie kann eine effiziente und benutzerfreundliche Steuerung und Verwaltung von Servicerobotern implementiert werden?

## 1.3 Methodik

Um die Forschungsfrage zu beantworten, wird nach dem Design Science Research (DSR) Ansatz nach Hevner [8] geforscht. Bei diesem handelt es sich um einen iterativen Forschungsansatz, mit dem Lösungen für praktische Probleme durch die Entwicklung von Artefakten gefunden werden. Im Rahmen dieser Arbeit ist der zu entwickelnde Prototyp das Artefakt.

### 1.3.1 Design Science Research nach Hevner

Bei diesem Abschnitt handelt es sich um eine Zusammenfassung des DSR Ansatzes nach Hevner [8, S. 79-81]. Hevnrs Ansatz setzt sich aus drei Schleifen zusammen, wobei die Relevanz- und Strenge-Schleifen nur unterstützende Funktionen für die eigentliche Entwicklung – die Design-Schleife – bieten.

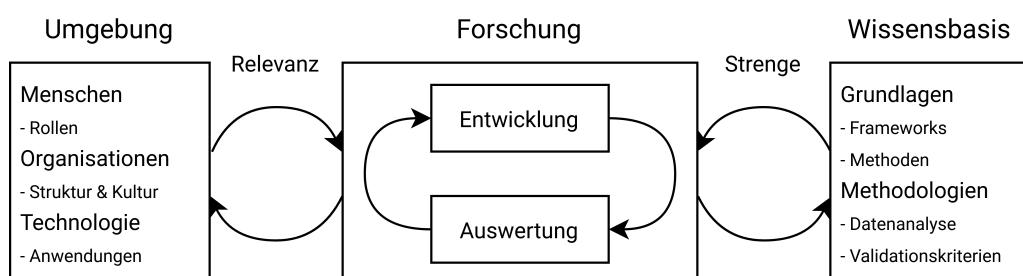
**Relevanz-Schleife:** Ergibt sich aus dem Austausch zwischen der Umgebung und dem Artefakt. So bilden sich die Anforderungen für das Artefakt aus der Umgebung. Das Artefakt wird wiederum innerhalb dieser Umgebung getestet, wodurch es, sowie die Validität der Anforderungen geprüft werden. Sowohl die Menschen und Organisationen, die das Artefakt nutzen sollen, als auch die Technologien, die im Artefakt eingesetzt werden sollen, bilden die Umgebung.

**Strenge-Schleife:** Ergibt sich daraus, dass während der Entwicklung und Auswertung des Artefakts auf die Wissensbasis zurückgegriffen wird. Die Wissensbasis wird durch die Entwicklung und Evaluierung des Artefakts erweitert. Diese Wissensbasis besteht hierbei sowohl aus dem technischen Wissen, dass während der Entwicklung relevant ist, als auch aus dem Wissen über Methodologien, die für die Auswertung relevant sind.

**Design-Schleife:** In dieser wird das Artefakt iterativ entwickelt. So wechselt sich das Entwickeln und Auswerten des Artefakts regelmäßig ab, bis ein befriedigendes Ergebnis erreicht wurde

In der Abbildung 1 werden die beschriebenen Prozesse grafisch dargestellt. So zeigt die Abbildung die Relevanz- und Strenge-Schleifen als Unterstützungsprozesse für die eigentliche Forschung in der Form der Design-Schleife.

**Abbildung 1: Design Science Research nach Hevner**



Quelle: In Anlehnung an Hevner et al. [8, S. 80]

### **1.3.2 Design Science Research im Rahmen der Arbeit**

Die Prozesse des DSR Ansatzes nach Hevner werden durch verschiedene wissenschaftliche Methoden abgebildet. Im ersten Schritt wird eine systematische Literaturrecherche durchgeführt, mit der das grundlegende Verständnis über die Umgebung geschaffen und die bereits vorhandene Wissensbasis erweitert werden soll. So wird unter anderem Wissen zu Servicerobotern, zur Generierung von 3D-Modellen und zur wissenschaftlichen Auswertung von Software-Anwendungen gesammelt. Während der Entwicklung des Prototyps weitet sich diese Literaturrecherche auf die Lösung auftretender Probleme aus. Die Anforderungen, die sich aus der Umgebung ergeben, werden nach der Wissensfindung durch eine Anforderungsanalyse definiert. Diese kann auf die Anforderungen aufgebaut werden, die sich bereits aus der Zielsetzung und der Formulierung der Forschungsfrage ergeben. So stehen die Anforderungen, dass der Prototyp eine Webanwendung sein und eine dreidimensionale Visualisierung bieten soll, bereits fest. Auch steht bereits fest, dass der Prototyp benutzerfreundlich und effizient sein soll. Der Prototyp wird als evolutionärer Prototyp entwickelt. Beim evolutionären Prototyping handelt es sich um einen iterativen Entwicklungsprozess, in dem ein Prototyp schrittweise verbessert wird. Im Gegensatz zum Throwaway Prototyping kann direkt auf den entwickelten Prototyp aufgebaut werden, um ein Produktivsystem umzusetzen.[9, S. 17-18]

## 2 Grundlagen

In diesem Kapitel werden grundlegende Konzepte und Technologien in den Bereichen Serviceroboter, Webanwendungen, 3D-Modelle und Softwarequalität vorgestellt.

### 2.1 Serviceroboter

Dieser Abschnitt gibt einen kurzen Überblick über Serviceroboter im Allgemeinen und eine Einführung in die Funktionen der Roboter, die im Prototyp eingesetzt werden. Außerdem wird das BCB genauer erläutert.

#### 2.1.1 Definition

Nach der ISO Norm 8373:2021 handelt es sich bei Servicerobotern um Roboter, die im privaten oder professionellen Gebrauch für Menschen nützliche Aufgaben erledigen. Serviceroboter werden hierbei von Industrierobotern und Medizinrobotern abgegrenzt. Roboter müssen zudem voll- oder zumindest teilautonom handeln können.[10, S. 1-2] Unter dem professionellen Gebrauch versteht man den kommerziellen Einsatz [11, S. 4], unter anderem im Gesundheitswesen, in der Landwirtschaft oder im Tourismus [11, S. 9].

#### 2.1.2 Einsatzmöglichkeiten

Serviceroboter werden bereits vielfältig professionell eingesetzt. So gibt es verschiedene Beispiele in denen sie in Hotels für den Gästeempfang, den Check-in und die Gepäcklieferung und an Flughäfen für die Beratung von Reisenden, das Scannen von Boardingpassen, den Check-in, die Bodenreinigung und Patrouillengänge genutzt werden [1, S. 425]. In der Pflege können Serviceroboter den Pflegern beim Heben von Patienten und beim Durchführen von Übungen mit Patientengruppen aushelfen [1, S. 427]. Aufgaben mit geringer kognitiver und emotionaler Komplexität können in der Regel vollautonom und ohne Aufsicht eines Menschen durchgeführt werden [1, S. 429]. Unter solche Aufgaben fällt zum Beispiel Staubsaugen, Rasenmähen und Gepäcklieferung. Komplexere Aufgaben erfordern die Aufsicht oder Unterstützung von Menschen, wodurch diese nur teilautonom ausgeführt werden [1, S. 430-431]. Für den Einsatz von Servicerobotern müssen immer die Vor- und Nachteile abgewogen werden. So können zum Beispiel Roboter, die im Kontakt mit Kunden eingesetzt werden, Emotionen vorspielen, die von Kunden allerdings als unauthentisch erkannt werden. Gleichzeitig sind Roboter im Gegensatz zu Menschen dafür ununterbrochen freundlich.[1, S. 427]

### 2.1.3 Pudu Robotics

Der Prototyp wird für den Einsatz von Pudu Robotern konzipiert. Pudu stellt Serviceroboter her, die vor allem in der Gastronomie eingesetzt werden können. Die Modelle sind hierbei auf unterschiedliche Funktionen, wie das Begrüßen von Gästen, das Liefern bestellter Speisen, das Zurückbringen dreckigen Geschirrs und das Putzen des Bodens spezialisiert [12]. Damit die Roboter diese Funktionen ausführen können, müssen sie eigenständig durch komplexe, sich ändernde Umgebungen navigieren. Die eigenständige Navigation lässt sich in die Teilstufen Positionsfindung, Wahrnehmung und Routenplanung aufteilen, wobei die Positionsfindung eine Schlüsselrolle spielt [13, S. 1]. Zur Positionsfindung erstellen sich die Pudu Roboter mit dem sogenannten Visual Simultaneous Localization and Mapping (VSLAM) eine Karte ihrer Umgebung, was bei einer Fläche von 1000 Quadratmetern eine Stunde dauern kann. Während Roboter zur Navigation normalerweise platzierte Markierungen benötigen, können sich Pudu Roboter mithilfe einer nach oben gerichteten Kamera anhand der Zimmerdecke orientieren.[14] Durch weitere Kameras und Sensoren können die Roboter außerdem ihre Umgebung wahrnehmen [13, S. 1].

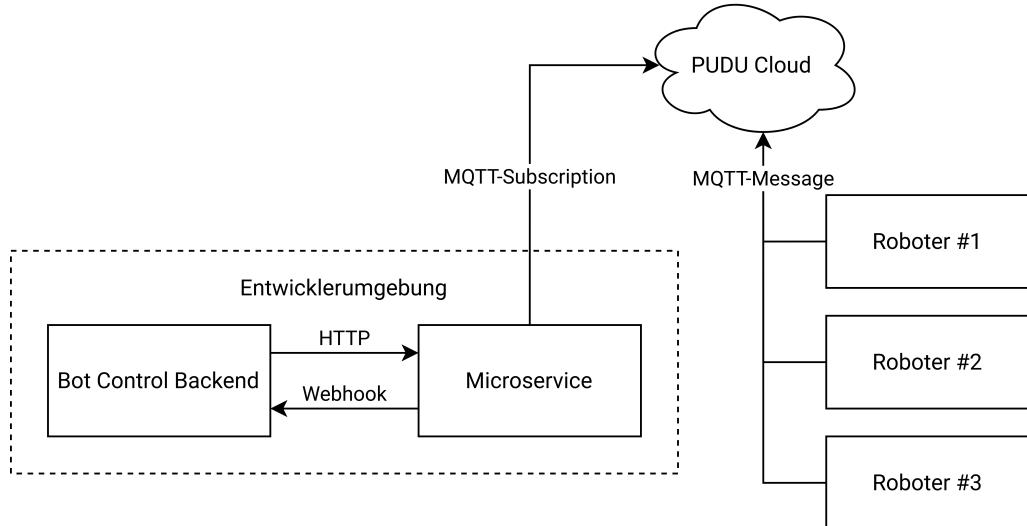
### 2.1.4 Bot Control Backend

Das bereits existierende BCB wird als Schnittstelle zwischen den Pudu Robotern und dem Prototyp genutzt. Über dieses können die Roboter zum Beispiel beauftragt werden. Die folgenden Informationen zum Service Framework der Roboter stammen aus dem SDK Guidance Document [15], das der Tobit Laboratories AG, für die Entwicklung des BCBs, durch Pudu zur Verfügung gestellt wurde.

Die Abbildung 2 veranschaulicht die Kommunikation zwischen dem BCB und den Robotern. Wie man in der Abbildung sieht hat das BCB eine direkte Verbindung zum Microservice, der wiederum über die PUDU Cloud – die als MQTT-Broker agiert – mit den Robotern kommuniziert. Über HTTP-Anfragen an den Microservice können Daten abgefragt und Befehle gegeben werden. Bei der Anfrage von Daten fragt der Microservice die entsprechenden MQTT-Topsics an. Wie die Kommunikation zwischen Microservice und Robotern genau funktioniert, wenn das BCB Befehle an die Roboter verschickt, ist aus dem Dokument nicht ersichtlich. Die Roboter können zusätzlich auch unaufgefordert Ereignisse an das BCB kommunizieren. Hierfür muss das BCB eine Adresse für einzelne Ereignistypen im Microservice als Webhook registrieren. Der Microservice abonniert daraufhin die entsprechende MQTT-Topic. So schicken die Roboter auftretende Ereignisse – wie eine Positionsaktualisierung – an die PUDU Cloud, die diese an den Microservice weiterleitet,

da dieser die MQTT-Topic abonniert hat. Der Microservice schickt das Ereignis daraufhin als HTTP-Nachricht an die, als Webhook registrierte, Adresse im BCB.

**Abbildung 2: Kommunikation zwischen Bot Control Backend und Robotern**



Quelle: In Anlehnung an Pudu [15, S. 4]

Das BCB fungiert nicht nur als Kommunikationsschnittstelle zu den Robotern, sondern abstrahiert auch neue Funktionen. So können Roboter mithilfe des BCBs Fahrstuhl fahren und somit Lieerpunkte in anderen Stockwerken erreichen. Ebenso können sie durch das BCB an geschlossenen Türen halten, diese öffnen und anschließend weiterfahren.

Es gibt verschiedene Daten, die über das BCB abgerufen werden können und für den Prototyp relevant sind, wie die Position der Roboter innerhalb ihrer internen Karte, sowie die Positionen wichtiger Standorte, wie Lieerpunkte und Ladestationen. Außerdem sind auch die Pfade relevant, an denen sich die Roboter während der Fahrt orientieren. Darüber hinaus sind auch noch die Positionen der virtuellen Wände wichtig. Diese müssen manuell platziert werden und agieren aus der Sicht der Roboter wie echte Wände, wodurch sichergestellt wird, dass bestimmte Bereiche nie durchfahren werden.

## 2.2 Webanwendungen

Webanwendungen sind Softwareanwendungen, die auf Webservern gehostet werden und über Webbrower aufgerufen werden können. Sie ermöglichen es Benutzern über das Internet auf Anwendungen zuzugreifen, ohne dass eine Installation dieser erforderlich ist. Da sie auf verschiedenen Betriebssystemen und Gerätetypen verwendet werden können, solange ein kompatibler Webbrower zur Verfügung steht, sind sie plattformunabhängig.

nutzbar.[7, Abschnitt 1-2] Zusammenfassend sind Webanwendungen breit und einfach zugänglich.

### **2.2.1 Technologien und Softwarebibliotheken**

Die reibungslose Entwicklung ansprechender Webanwendungen erfordert den Einsatz verschiedener Technologien.

Die drei zentralen Technologien jeder Webanwendungen sind HyperText Markup Language (HTML), Cascading Style Sheets (CSS) und JavaScript. HTML ist eine Auszeichnungssprache, die die Struktur einer Webseite definiert, indem Elemente wie Überschriften, Absätze und Hyperlinks verwendet werden, um Inhalte zu organisieren und zu strukturieren [16]. Um das Erscheinungsbild einer Webseite zu gestalten wird CSS genutzt. Mit CSS lassen sich Layout, Formatierung und weitere Eigenschaften von HTML-Elementen definieren.[17] Sassy Cascading Style Sheets (Sass) erweitert CSS durch Verschachtelung, Variablen, Mixins und weitere Funktionen [18], wodurch die Wiederverwendbarkeit definierter Styles verbessert und die Entwicklung vereinfacht wird. Die Interaktivität von Webseiten wird durch JavaScript ermöglicht [19]. Bei JavaScript handelt es sich um eine dynamisch typisierte Skriptsprache, die durch TypeScript um eine statische Typisierung erweitert wird [20]. Durch TypeScript wird die Entwicklung einer robusten und fehlerfreien Webanwendung im Vergleich zu JavaScript erleichtert.

Die Entwicklung komplexerer Webanwendungen wird durch die JavaScript-Bibliotheken React und Redux weiter erleichtert. React wird zur Erstellung von Benutzeroberflächen verwendet und ermöglicht die Entwicklung wiederverwendbarer User Interface (UI) Komponenten – im Folgenden auch React-Komponenten genannt –, sowie eine effiziente Aktualisierung der Benutzeroberflächen durch die Verwendung virtueller DOMs. React-Projekte werden in JavaScript XML (JSX) entwickelt, womit sich HTML Elemente im JavaScript-Code integrieren lassen. Damit React Projekte von Webbrowsern ausgelesen werden können, muss der geschriebene JSX-Code zu JavaScript transpiliert werden.[21] Redux ermöglicht eine übersichtliche Verwaltung des Anwendungsstatus [22].

### **2.2.2 chayns**

Bei chayns handelt es sich um eine Digitalisierungsplattform, die durch die Tobit Software GmbH vertrieben wird. Unter anderem bietet chayns einen Cloud-Speicher – den chayns.space – und einen Webseiten-Baukasten, mit dem sich Nutzer eine Webpräsenz erstellen können. Auf den Webseiten können vordefinierte chayns Anwendungen,

wie ein eShop oder ein Bundesliga-Tippspiel, aber auch eigenentwickelte Webanwendungen eingebunden werden.[23] Als Besitzer einer chayns Seite hat man Zugriff auf den Admin-Modus, in dem sich die meisten chayns Anwendungen verwalten lassen. So gibt es auch im entwickelten Prototyp eine Nutzer- und Adminansicht. Tobit bietet verschiedene Softwarebibliotheken, die die Entwicklung von chayns Anwendungen erleichtern. So gibt es den Shell Befehl `create-chayns-app` [24], mit dem chayns basierte React Projekte aufgesetzt werden können; das Node Package Manager (npm) Paket `chayns-toolkit` [25], mit dem chayns basierte React Projekte kompiliert werden können; das npm Paket `chayns-components` [26], das React-Komponenten für verschiedene Graphical User Interface (GUI) Elemente zur Verfügung stellt; und das npm Paket `chayns-api` [27], das verschiedene hilfreiche Funktionen bereitstellt. Für unternehmensinterne Projekte bietet Tobit den WebSocket-Service der eine indirekte WebSocket-Verbindung zwischen Backends und Clients ermöglicht. Über diese können Backends unaufgefordert Nachrichten an verbundene Clients versenden.

## 2.3 3D Modelle

Dieser Abschnitt bietet eine kurze Einführung in das Thema der 3D-Modelle. Daraufhin werden verschiedene Methoden zur Erzeugung von 3D-Gebäudemodellen vorgestellt und es wird erläutert, wie 3D-Modelle in Webanwendungen eingebunden werden können.

3D-Modelle sind digitale Darstellungen von Objekten oder Szenen in drei Dimensionen. Anders als bei 2D-Grafiken, die lediglich eine Breite und Höhe haben, enthalten 3D-Modelle zusätzlich Tiefeninformationen, aus denen sich die dritte Dimension ergibt. Polygonale 3D-Modelle bestehen aus Polygonen, die sich aus Eckpunkten und Kanten – den Verbindungen zwischen Eckpunkten – zusammensetzen. Oberflächeneigenschaften der Polygone, wie Farben, Glanz und Reflexionen lassen sich durch die Anwendung von Texturen und Materialien definieren. Transformationsoperationen wie Skalierung, Rotation und Translation ermöglichen es, 3D-Modelle im Raum zu bewegen und zu manipulieren. Die Kamera, Perspektive und Projektion legen den Standpunkt des Betrachters und die Darstellung des Modells fest.[28, S. 8-16]

### 2.3.1 Generierung

Neben der manuellen Modellierung von 3D-Modellen, mithilfe von Modellierungssoftware wie Blender [29], gibt es verschiedene Methoden, die sich insbesondere zur Generierung von Raummodellen eignen.

### **2.3.1.1 Fotogrammetrie**

Die Fotogrammetrie beschäftigt sich damit, Messungen aus einer Vielzahl an zweidimensionalen Bildern abzuleiten, aus denen sich präzise 3D-Modelle erzeugen lassen [30, S. 19]. Inzwischen erfordert die Fotogrammetrie nicht mehr den Einsatz teurer Kameras, da die Kameras moderner Mobilgeräte eine ausreichende Bildqualität bieten [31]. Bei der Planung und Durchführung der Bildaufnahmen müssen verschiedene Aspekte beachtet werden. Innerhalb der aufzunehmenden Szene sollte es eine gleichmäßige Belichtung, möglichst keine reflektierende oder transparente Flächen und keine sich bewegende Objekte geben. Während der Aufnahme müssen Parameter wie Belichtungszeit und Weißabgleich passend konfiguriert sein und zwischen den einzelnen Aufnahmen unverändert bleiben. Außerdem muss sich der Inhalt aufeinanderfolgender Bilder stets überschneiden.[32] Nach der Durchführung der Aufnahmen erfolgt die Verarbeitung mithilfe spezialisierter Software, deren Bedienung komplex sein kann. Für eine reibungslose Verarbeitung sind eine leistungsstarke Grafikkarte (GPU) und ausreichend Speicherplatz unerlässlich.[33]

### **2.3.1.2 LiDAR Scanning**

Im Gegensatz zur Fotogrammetrie nutzt Light Detection and Ranging (LiDAR) einen aktiven Sensor. Es wird Licht in Form eines pulsierenden Lasers ausgesendet. Die Reflexionen werden mit einem Scanner erfasst, wodurch sich Abstände zwischen Sensor und Gegenständen berechnen lassen. Auf Grundlage dieser Messungen wird ein 3D-Modell erstellt. Wie bei der Fotogrammetrie, muss beim Scannen darauf geachtet werden, dass reflektierende oder transparente Flächen, sowie sich bewegende Objekte vermieden werden. Seit 2020 werden LiDAR-Scanner in iOS-Geräte von Apple integriert, was unter anderem zu einer verbesserten Bildqualität beim Fotografieren beitragen soll [34]. Als glücklicher Nebeneffekt wurden verschiedene Apps, wie Canvas [35], Polycam [36] und Scaniverse [37] entwickelt, die den LiDAR-Scanner nutzen, um 3D-Modelle zu erstellen. Diese Apps versprechen eine schnelle und einfache Erzeugung akkurate 3D-Modelle.

### **2.3.1.3 KI gestützte Methoden**

Es existieren verschiedene KI-Modelle, die darauf trainiert sind, 3D-Gebäudemodelle mit nur wenigen Bildern zu erzeugen. Eines dieser KI-Modelle ist Plan2Scene. Es benötigt als Eingabe einen Grundriss des Gebäudes, sowie Bilder, die den einzelnen Räumen zugeordnet sind. Basierend auf dem Grundriss generiert das KI-Modell ein 3D-Modell mit Möbeln. Basierend auf den Bildern der Räume werden monotone Texturen generiert.[38,

S. 10733] Das Rent3D Modell funktioniert ähnlich, nutzt die Bildaufnahmen aber direkt als Textur, statt sie aus den Bildaufnahmen zu generieren [39, S. 3413].

### **2.3.2 Einbindung im Web**

Das Einbinden von 3D-Modellen wird im Web durch die Web Graphics Library (WebGL) und WebGPU ermöglicht. Während WebGL für lange Zeit der etablierte Standard war, gewinnt WebGPU seit der Veröffentlichung im Jahr 2021 stetig an Popularität.

#### **2.3.2.1 WebGL und WebGPU**

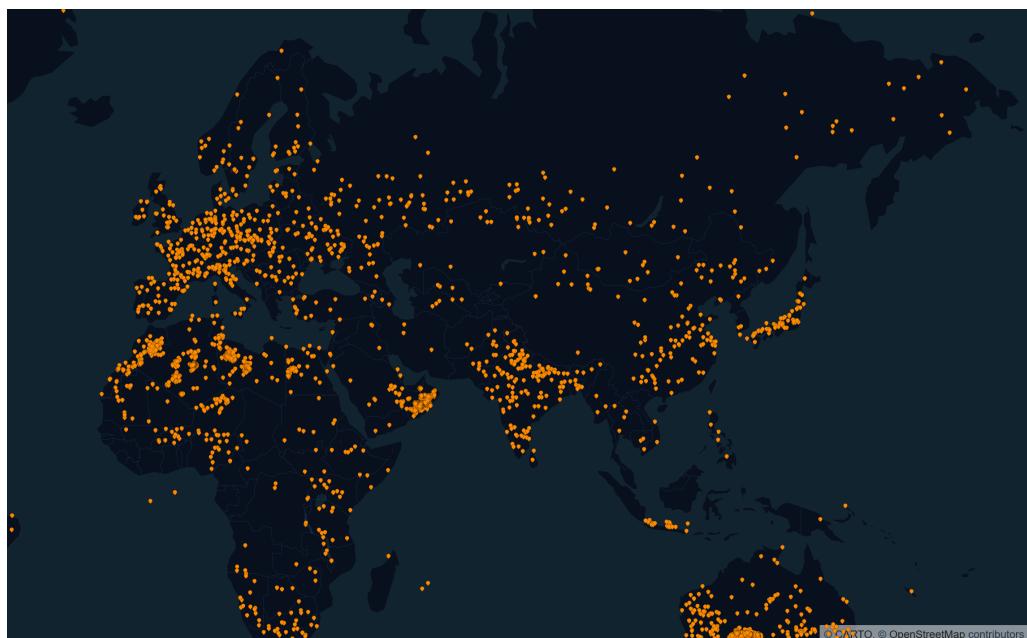
WebGL wurde 2011 von der Khronos Group entwickelt und ist ein JavaScript Application Programming Interface (API) mit dem 3D-Grafiken im Webbrowser ohne den Einsatz zusätzlicher Plugins dargestellt werden können. Die 3D-Grafiken können hierbei hardwarebeschleunigt angezeigt werden, also mithilfe spezialisierter Hardware wie einer GPU. Durch die Integration mit HTML und JavaScript können 3D-Grafiken dynamisch auf Webseiten eingebunden werden. Da WebGL auf offenen Webstandards basiert, ist es in allen Browsern plattformunabhängig nutzbar.[28, S. 17-19] WebGPU bietet wie WebGL das hardwarebeschleunigte Anzeigen von 3D-Grafiken und darüber hinaus eine verbesserte Leistung sowie einen erweiterten Funktionsumfang [40]. Entwicklern steht eine Vielzahl an Frameworks zur Verfügung die auf WebGL oder WebGPU basieren [41] und die Entwicklung vereinfachen und beschleunigen.

#### **2.3.2.2 deck.gl**

Das Framework deck.gl wurde 2016 von Uber als Open Source Projekt veröffentlicht [42]. Das Framework basiert auf WebGL, wobei ab der kommenden Version 9.0.0 stattdessen WebGPU genutzt wird [43]. Mit deck.gl lassen sich hochperformante interaktive Karten und Geovisualisierungen mit tausenden bis Millionen Datenpunkten im Web einbinden. Da das Framework auf React ähnlichen Programmierparadigmen basiert, eignet es sich besonders gut für die Einbindung in React Anwendungen. Das Framework funktioniert nach dem Primitive Instancing Layering (PIL) Prinzip. So gibt es Ebenen, welche grundlegende visuelle Elemente – im Englischen Primitives –, wie Kreise, Rechtecke und Linien, aber auch komplexere teilweise dreidimensionale Elemente nutzen, um Datenpunkte darzustellen. Die Elemente werden in einer Ebene auf Basis der Attribute der Datenpunkte

positioniert, skaliert und gefärbt. Die Ebenen können gestapelt und somit kombiniert werden, was auch die Inspiration für den Namen des Frameworks ist, da das englische Wort deck in etwa Stapel bedeuten kann.[44, S. 2] Das Framework bietet verschiedene vordefinierte Ebenen, wie die IconLayer [45], die Icons als grundlegendes visuelles Element nutzt. Zur Veranschaulichung des PIL-Prinzips zeigt die Abbildung 3 wie die IconLayer – in Kombination mit einer weiteren Ebene zur Darstellung der Weltkarte – genutzt wird, um die Positionen aller bekannten Meteoritenlandungen auf der Erde anzuzeigen.

**Abbildung 3: IconLayer Beispiel**



Quelle: OpenJS Foundation [46]

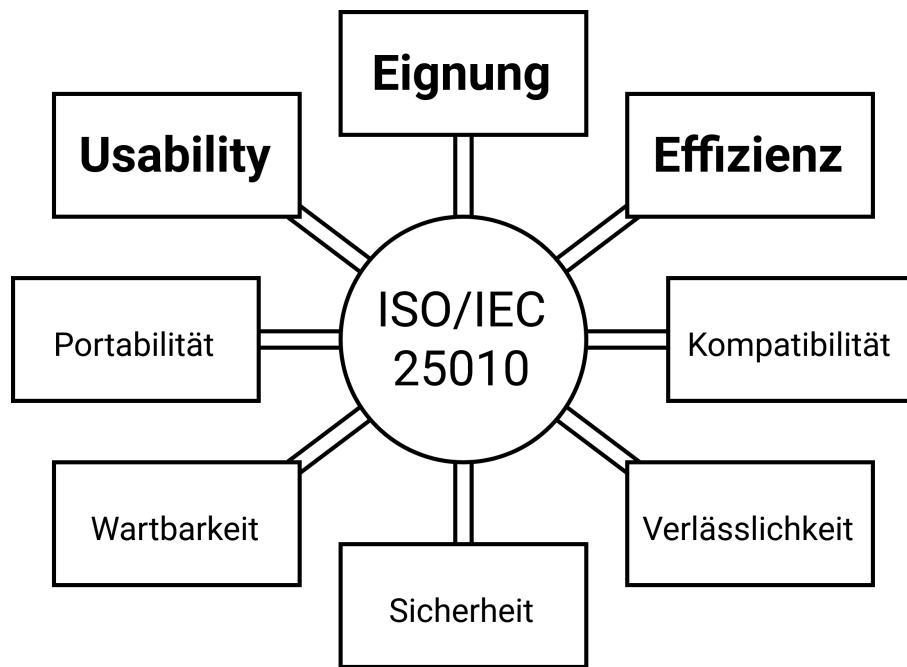
Weitere Ebenen, die für den Prototyp relevant sind, sind die SimpleMeshLayer [47] und ScenegraphLayer [48] für das Anzeigen von 3D-Modellen und die PathLayer für das Anzeigen von Pfaden. deck.gl ermöglicht außerdem den Einsatz eigenentwickelter Ebenen, was für den Prototyp aber nicht gebraucht wird. Weitere wichtige Elemente die deck.gl bietet sind die Controller Klasse [49], mit der die Navigation auf der Karte konfiguriert werden kann und die Viewport Klasse [50], mit der die Navigation direkt kontrolliert werden kann.

## 2.4 Softwarequalität

“Unter Softwarequalität versteht man die Gesamtheit der Merkmale und Merkmalswerte eines Softwareprodukts, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen” [51, S. 257]. So ergibt sich die Softwarequalität aus der

Erfüllung der definierten Anforderungen – aber auch aus der Erfüllung undefinierter Erwartungen. Die Anforderungen an Software können in die acht Produktqualitätsmerkmale nach dem Standard ISO/IEC 25010 aufgeteilt werden [52, S. 3-4]. In Abbildung 4 werden diese Merkmale aufgelistet. Die Merkmale Effizienz und Benutzerfreundlichkeit haben eine besondere Relevanz für die Evaluierung des Prototyps, da sie direkt in der Forschungsfrage gefordert werden. Auch die funktionale Eignung ist relevant, da diese die Erfüllung der funktionalen Anforderungen abdeckt, die im Abschnitt 3.2 definiert werden. Die drei genannten Merkmale sind in der Abbildung entsprechend gekennzeichnet. Im Rahmen dieser Arbeit wird ein Prototyp entwickelt, der nicht die Ansprüche an ein Produktivsystem erfüllen muss. Aus diesem Grund werden die restlichen Merkmale vernachlässigt. Im Folgenden werden die Merkmale Benutzerfreundlichkeit und Effizienz genauer erläutert.

**Abbildung 4: Qualitätsmerkmale**



#### 2.4.1 Benutzerfreundlichkeit

Die Benutzerfreundlichkeit wird nach ISO/IEC 25010 in sechs Kriterien aufgeteilt: angemessene Erkennbarkeit, Erlernbarkeit, Bedienbarkeit, Toleranz gegenüber Anwenderfehlern, Ästhetik der Benutzeroberfläche und Barrierefreiheit [52, S. 4].

#### **2.4.1.1 Usability Heuristics**

Die zehn Usability Heuristics von Nielsen sind grundlegende Richtlinien zur Bewertung der Benutzerfreundlichkeit [53] und decken die ISO/IEC 25010 Kriterien weitestgehend ab. So gibt es zum Beispiel die folgenden Richtlinien:

- Das Design sollte die Sprache der Benutzer sprechen, indem vertraute Wörter, Phrasen und Konzepte, statt interne Fachbegriffe verwendet werden [53, Regel 2].
- Das Design sollte konsistent sein und etablierten Standards folgen, sodass Benutzer nicht darüber nachdenken müssen, ob verschiedene Wörter, Situationen oder Aktionen dasselbe bedeuten [53, Regel 4].

Beide Richtlinien helfen dabei das ISO/IEC 25010 Kriterium der Erlernbarkeit einzuhalten. Während der Entwicklung können die Richtlinien als Orientierungshilfe dienen, um die Benutzerfreundlichkeit zu verbessern. Eine subjektive Einhaltung dieser ist allerdings unzureichend, um die Benutzerfreundlichkeit zu bewerten. Usability Tests eignen sich hierfür besser.

#### **2.4.1.2 Usability Tests**

Usability Tests erfordern die Beobachtung von Testpersonen bei der Nutzung des zu prüfenden Produkts, während sie vorab entwickelte Anwendungsszenarien durchspielen. Die Testpersonen sollten potenzielle Benutzer des Produkts repräsentieren.[54, S. 22] Nach der Durchführung der Tests werden die gesammelten Beobachtungen auf Probleme und Schwachstellen im Produkt ausgewertet. Die Tests können entweder quantitativ oder qualitativ durchgeführt werden. Bei quantitativen Usability Tests werden verschiedene Metriken, wie die Durchführungszeit oder die Rate der erfolgreichen Durchführung von Aufgaben gesammelt. Diese Metriken zeigen im Vergleich zu den Ergebnissen früherer oder zukünftiger Tests, wie sich die Benutzerfreundlichkeit zwischen verschiedenen Versionen entwickelt hat. In qualitativen Usability Tests werden Testpersonen bei der Interaktion mit dem Produkt beobachtet, um Designmerkmale zu identifizieren, die gut oder schlecht zu bedienen sind.[55] Qualitative Tests erfordern einen Moderator, der die Testpersonen durch den Testprozess leitet. Gegebenenfalls gibt es auch weitere Beobachter, die nicht mit den Testpersonen interagieren.[56] Für qualitative Tests reicht eine Auswahl an fünf Testpersonen aus, um einen Großteil der Probleme zu finden. Mit einer zunehmenden Menge an Testpersonen sinkt das Return of Investment maßgeblich dadurch, dass immer weniger neue Fehler pro Testperson entdeckt werden.[57]

## 2.4.2 Effizienz

Die Effizienz einer Anwendung beeinflusst die Benutzerfreundlichkeit und Absprungraten maßgeblich und zeichnet sich unter anderem durch die Dauer der Ladezeiten und die Reaktionsgeschwindigkeit der Benutzeroberfläche aus. Für die Bewertung der Effizienz einer Webanwendung oder Webseite gibt es verschiedene Merkmale. Beim Prototyp wird besonders auf die wahrgenommene Ladezeit, die Lade-Reaktionsfähigkeit und die Smoothness geachtet.

Unter der wahrgenommenen Ladezeit versteht man wie schnell der Ladevorgang dem Nutzer erscheint [58, Abschnitt 4]. Zur Bestimmung dieser Ladezeit wird gemessen, wann bestimmte UI-Elemente angezeigt werden. Der Largest Contentful Paint (LCP) ist hierbei der wichtigste Messwert, da mit diesem gemessen wird, wann das größte oder wichtigste Element angezeigt wird [59]. Der First Contentful Paint (FCP) ist ein weiterer erwähnenswerter Messwert, mit dem gemessen wird, wann das erste Element angezeigt wird [60].

Unter der Lade-Reaktionsfähigkeit versteht man wie flüssig auf Interaktionen des Nutzers reagiert wird, während der Ladevorgang noch stattfindet [58, Abschnitt 4]. Gemessen wird die Reaktionsfähigkeit vor allem durch den First Input Delay (FID). Dieser misst die Verzögerung, die bei der ersten Interaktion des Nutzers auftritt [61]. Zusätzlich kann auch die Total Blocking Time (TBT) gemessen werden. Hier wird gemessen wie lange der Hauptthread so stark beschäftigt ist, dass Interaktionen nur verzögert verarbeitet werden können [62].

Mit der Smoothness ist die insgesamte Geschmeidigkeit des Nutzererlebnisses gemeint, also ob Übergänge und Animationen mit einer konsistenten Bildrate gerendert werden und flüssig aussehen [58, Abschnitt 4]. Die Smoothness lässt sich durch die Interaction to Next Paint (INP) – die Latenz von Interaktionen – messen [63].

## 3 Anforderungen an den Prototyp

In diesem Kapitel werden die funktionalen und nicht-funktionalen Anforderungen an den Prototyp, sowie die Anforderungen an die Modellerzeugungsmethode vorgestellt.

### 3.1 Anforderungen an Modellerzeugung

Für die Relevanz des Prototyps ist es erheblich, dass es eine einfache Methode gibt, mit der 3D-Gebäudemodelle erzeugt werden können. Die Methode soll möglichst geringe Kosten verursachen und kein besonderes technisches Know-how erfordern, um sich von klassischer Modellierungssoftware wie Blender abzuheben. Die erzeugten Modelle müssen nicht hundertprozentig genau, aber genau genug sein, um eine gute Übersicht zu ermöglichen. Eine tiefere Analyse der gewählten Methode sowie ein tieferer Vergleich zwischen verschiedenen Methoden ist unerheblich, da die gewählte Methode nicht die beste sein muss. So soll nur herausgearbeitet werden, dass es eine passende Methode gibt, nicht welche am besten für den Anwendungsfall geeignet ist.

### 3.2 Funktionale Anforderungen

Wie im Abschnitt 1.2 beschrieben, soll der Prototyp als Webanwendung implementiert werden. So wird mit wenig Aufwand eine Plattformunabhängigkeit ermöglicht. Diese ist nötig, da die Anwendung auf beliebigen Geräten nutzbar sein soll. Aus diesem Grund soll der Prototyp außerdem responsiv sein.

Die Anwendung lässt sich in drei zentrale Funktionen aufteilen: Übersicht, Steuerung und Verwaltung.

#### 3.2.1 Übersicht

In der Übersicht sollen die relevanten Roboterdaten zusammen mit dem Gebäudemodell in einer dreidimensionalen Ansicht abgebildet werden. Da die Roboter in der Lage sind Fahrstuhl zu fahren, soll es die Möglichkeit geben zwischen verschiedenen Stockwerken zu navigieren. Es soll außerdem in Echtzeit angezeigt werden, wo sich die Roboter befinden. In der dreidimensionalen Darstellung sollen die Roboter dann so wie in der Realität fahren. Auch soll ersichtlich sein, ob und womit die Roboter beschäftigt sind. Falls ein Roboter einen Lieferauftrag ausführt, soll das Ziel angezeigt werden. Weitere Daten, die von

den Robotern stammen und angezeigt werden sollen, sind die Positionen aller möglichen Ausgabe- und Lieferpunkten, sowie Ladestationen. Zudem haben die Roboter festgelegte Pfade, an denen sie sich beim Fahren orientieren. Diese sollen auch angezeigt werden.

### **3.2.2 Steuerung**

Mit der Steuerung sollen die Roboter beauftragt werden können. So soll es die Möglichkeit geben einen Zielpunkt einzustellen. Auch sollen die Roboter an ihre Ladestation geschickt werden können.

### **3.2.3 Verwaltung**

In der Verwaltung sollen sowohl die Roboter als auch die Übersicht selbst verwaltet werden können. So soll das Ändern der verschiedenen Roboter-Einstellungen möglich sein. Beispielsweise soll die eingestellte Ladestation änderbar sein. Es soll außerdem möglich sein die 3D-Gebäudemodelle zu importieren und diese mit den Roboterdaten zu synchronisieren. So sollte es die Möglichkeit geben das importierte 3D-Gebäudemodell und die mit VSLAM erzeugte interne Karte der Roboter anzulegen, damit die Positionen der Roboter in der Übersicht mit der Realität übereinstimmen.

### **3.2.4 Benutzer**

Für die Anwendung gibt es drei verschiedene Nutzergruppen: Gäste, Mitarbeiter und Administratoren. Der für den Nutzer verfügbare Funktionsumfang erhöht sich in dieser Reihenfolge. So sollen Gäste nur einen eingeschränkten Zugriff auf die Übersicht haben während Administratoren Zugriff auf alle Funktionen der Übersicht, Steuerung und Verwaltung haben.

Der Wert von Servicerobotern muss insbesondere aus der Sicht von Gästen noch bewiesen werden [1, S. 429]. Deshalb sollen Gäste einen eingeschränkten Zugriff auf die Übersicht der Roboter bekommen, in der sie die Positionen und aktuellen Aufträge sehen können. Mithilfe dieser Transparenz sollen ihnen die Vorteile von Servicerobotern anschaulich werden.

Die Mitarbeiter sollen einen vollständigen Zugriff auf die Übersicht und Steuerung bekommen. So werden ihnen alle Funktionen zur Verfügung gestellt, die sie für das Steuern

der Roboter brauchen. Auch sollen ihnen in der Übersicht mehr Informationen angezeigt werden. Einen Zugriff auf die Verwaltung der Roboter brauchen die Mitarbeiter nicht.

Auf die Verwaltung sollen nur Administratoren Zugriff haben. Während die Verwaltung der Roboter kontinuierlich genutzt werden sollte, sollte die Verwaltung der Stockwerke hauptsächlich bei der Einrichtung gebraucht werden. So sollte die Karte nur zu Beginn eingerichtet und danach nie wieder verändert werden müssen.

### **3.3 Nicht-funktionale Anforderungen**

Lange Ladezeiten während der Nutzung der Anwendung verursachen Frust. Kurze Ladezeiten sind deshalb eine zentrale Anforderung an den Prototyp. Normalerweise lassen sich Ladezeiten sowohl auf der Server- als auch auf der Client-Seite verbessern. Der Schwerpunkt des Prototyps liegt allerdings im Frontend, weshalb sich Optimierungen nur auf die Client-Seite beschränken sollen. Das BCB soll unabhängig davon, ob Optimierungspotenzial existiert, nicht verändert werden. Maßnahmen, die sich folglich anbieten sind, das Verzögern des Ladens nicht essenzieller Ressourcen, sowie die Reduktion des Datenverbrauchs. Eine Reduktion des Datenverbrauchs bietet auch den Vorteil, dass sich Kosten für Nutzer reduzieren, die einen teuren oder begrenzten Datenplan verwenden. Unabhängig von Ladezeiten sollte der Prototyp trotz der rechenaufwändigen Darstellung der 3D-Modelle möglichst performant sein. So soll es zum Beispiel beim Navigieren keine Ruckler geben. All diese Anforderungen lassen sich unter dem Begriff der Effizienz zusammenfassen. Die Effizienz soll anhand der wahrgenommenen Ladezeit, der Lade-Reaktionsfähigkeit und der Smoothness bewertet werden. Neben der Effizienz sollte die Anwendung auch benutzerfreundlich sein, damit sichergestellt wird, dass sich die Benutzer problemlos zurechtfinden können. Die Benutzerfreundlichkeit soll anhand von Usability Tests bewertet werden.

## 4 Technische Herausforderungen

Vor und während der Implementierung des Prototyps sind verschiedene größere technische Herausforderungen aufgetreten, die gelöst werden mussten. Die Herausforderungen werden in diesem Kapitel zusammen mit den gewählten Lösungsansätzen vorgestellt.

### 4.1 Methode zur Modellgenerierung

Die 3D-Modelle der Stockwerke wurden mithilfe des LiDAR-Scannens erstellt. Hierfür wurde die Scaniverse App auf einem iPhone 14 Pro verwendet. Da das Gerät zur Verfügung stand und die Scaniverse App kostenlos nutzbar ist, war das Scannen mit keinen Kosten verbunden. Aufgrund des höheren Aufwands bei der Aufnahme und Verarbeitung von Bildern wurde die Fotogrammetrie als Methode verworfen. Zwar würde sich das iPhone auch für Aufnahmen eignen die fotogrammetrisch verarbeitet werden können, diese Verarbeitung erfordert allerdings ein weiteres Gerät mit einer leistungsstarken GPU. Außerdem ist das Verarbeiten der Bilder bei der Fotogrammetrie mit mehr Aufwand als beim LiDAR-Scannen verbunden. Die KI-gestützten Methoden wurden nicht eingesetzt, da sie aufgrund der geringen Menge an Publikationen noch unausgereift erscheinen.

### 4.2 3D Visualisierung im Web

Für das Einbinden von 3D-Visualisierungen im Web gibt es verschiedene Ansätze und Technologien. In diesem Abschnitt wird die Auswahl von deck.gl als Visualisierungs-Framework, sowie die Wahl des Dateiformats der 3D-Modelle erläutert.

#### 4.2.1 deck.gl

Für die Umsetzung der 3D-Visualisierung im Prototyp wurde das Framework deck.gl gewählt, da die Anwendung als Karte genutzt werden soll und deck.gl für das Entwickeln dieser ausgelegt ist. So ist die Navigation und das Verhalten der Kamera bereits passend konfiguriert, sodass bei der Entwicklung dieser Features Zeit gespart werden kann. Vor der Implementierung des Prototyps konnte bestätigt werden, dass die Anforderungen an den Prototyp mit dem Framework eingehalten werden können.

#### 4.2.2 Dateiformat der 3D-Modelle

Für die Darstellung eines 3D-Modells gibt es in deck.gl zwei Möglichkeiten: das Einbinden des Wavefront Object (OBJ) Dateiformats in der SimpleMeshLayer [47] und das Einbinden des Graphics Library Transmission Format (glTF) Dateiformats in der ScenegraphLayer [48]. Beide Dateiformate werden im Dateiexport der Scaniverse App angeboten.

Das OBJ Format besteht aus einer Datei mit der Endung .obj, in der die dreidimensionalen geometrischen Formen kodiert sind [64] und einer Datei mit der Endung .mtl, in der die optischen Materialeigenschaften und Texturierung kodiert sind [65]. Für das Einbinden des OBJ Dateiformats wird die loaders.gl Programmbibliothek benötigt, die allerdings nur die .obj Datei und nicht die .mtl Datei parsen kann [66]. So können die 3D-Modelle in der SimpleMeshLayer nur ohne Textur angezeigt werden.

Das glTF Format bietet zwei verschiedene Dateiformate, wobei hier nur die binäre Variante relevant ist. Diese besteht aus einer Datei mit der Endung .glb, welche neben den geometrischen Formen auch die Materialeigenschaften und Texturierung enthält. Das glTF Format verspricht eine geringere Dateigröße als vergleichbare Dateiformate wie OBJ.[67, Abschnitt 2] Mithilfe der loaders.gl Programmbibliothek lassen sich texturierte 3D-Modelle des Formats ohne großen Aufwand in der ScenegraphLayer von deck.gl einbinden [48]. Da ein 3D-Modell mit passender Texturierung eine bessere Übersichtlichkeit bietet und glTF Dateien eine geringere Dateigröße haben, wird die ScenegraphLayer mit 3D-Modellen im glTF Format für die Darstellung der Raummodelle genutzt.

Die glTF Dateien, die für den Prototyp aus der Scaniverse App exportiert werden, sind mit einer Dateigröße von 14 bis 21 Megabyte (mB) für den Einsatz im Prototyp zu groß, da sie – vermutlich durch ihre Komplexität – nicht auf Mobilgeräten angezeigt werden können. Außerdem beeinflusst die Dateigröße die Ladezeit negativ – sowohl beim Herunterladen der Daten vom Webserver als auch beim Anzeigen der 3D-Modelle. Aus diesem Grund müssen die Dateien komprimiert werden. Hierfür sind die 3D-Modelle, die im Prototyp eingesetzt werden mit dem OptimizeGLB Online Konverter [68] manuell komprimiert worden. Insbesondere durch den Einsatz des Web Picture Format (WebP) Bildformats für Texturen werden die Dateien effektiv komprimiert. Die komprimierten Dateien sind zwischen 330 und 550 Kilobyte (kB) groß, was einer Kompression von über 95% entspricht. Die Qualität der komprimierten 3D-Modelle ist erkennbar geringer. Für den Zweck der Anwendung reicht die Qualität trotzdem aus, da größere Merkmale weiter gut erkennbar sind und somit die Übersichtlichkeit weiter garantiert wird.

### 4.3 Synchronisierung des Gebäudemodells und der Roboterdaten

Im Prototyp sollen die Roboterdaten in den 3D-Modellen integriert dargestellt werden. Die Roboterdaten und 3D-Modelle haben den gleichen Maßstab, die Positionen und Rotationen der Datensätze stimmen allerdings nicht miteinander überein. Diese Unterschiede sind eine Konsequenz daraus, dass zur Generierung unterschiedliche Scanning-Methoden eingesetzt werden. Außerdem stimmen die Positionen der 3D-Modelle nicht untereinander überein, da beim Scannen an verschiedenen Ausgangspunkten angefangen wird.

Aus diesem Grund müssen die Positionen der Roboterdaten mit den 3D-Modellen synchronisiert werden. Auch müssen die Positionen der 3D-Modelle untereinander synchronisiert werden. Eine automatische Synchronisierung ist aus verschiedene Gründen zu komplex. Zum einen sind die Formate der Daten zu verschieden, denn während die 3D-Modelle aus komplexen dreidimensionalen Formen bestehen, setzen sich die Roboterdaten aus zweidimensionalen Linien und Punkten zusammen. Zum anderen gibt es in beiden Datensätzen unterschiedliche Ungenauigkeiten in Bezug auf die Realität. Sowohl VSLAM, mit dem die Roboterdaten untereinander positioniert werden, als auch das LiDAR-Scanning, mit dem die 3D-Modelle generiert werden, sind fehlerbehaftet. Da sich die Scanning-Methoden unterscheiden, unterscheiden sich auch die Ungenauigkeiten.

Da eine automatische Synchronisierung der Datensätze somit ausgeschlossen ist, muss diese manuell durch den Nutzer vorgenommen werden. Hierfür wurde ein Editiermodus implementiert, mit dem der Administrator die 3D-Modelle und Roboterdaten durch Verschieben und Rotieren der 3D-Modelle synchronisieren kann. Die Implementierung wird im Abschnitt 5.2.4 beschrieben.

## 5 Umsetzung des Prototyps

Im Folgenden wird das Mockup, die Implementierung und das Deployment des Prototyps beschrieben. Während der Skizzierung der Mockups und der Implementierung wurde auf eine gute Benutzerfreundlichkeit geachtet. Hierfür wurde sich unter anderem an Nielsens Usability Heuristics [53] – im Folgenden auch Usability Entscheidungsregeln – orientiert. Im Abschnitt 6.2.1 wird genauer geprüft, wie gut diese eingehalten wurden.

### 5.1 Mockup

Für den Prototyp wurden Mockups skizziert die während der Implementierung als Orientierungshilfe genutzt wurden. Diese Mockups wurden zunächst auf Papier niedergeschrieben und für die folgende Beschreibung mithilfe von Figma [69] digitalisiert. Es gibt jeweils ein Mockup für die Übersicht, Steuerung und Verwaltung, sowie für das Routenplanungs-Popup in der Steuerung. Für den Editiermodus wurde kein Mockup entworfen, da das Design und Verhalten zu stark von den Möglichkeiten in deck.gl abhängt, die noch nicht vollständig bekannt waren, als die Mockups entworfen wurden. Stattdessen wurde das Design und Verhalten des Editiermodus während der Implementierung festgelegt.

Die Abbildung 14 im Anhang zeigt das Mockup für die Übersicht. In der Übersicht werden die Raummodelle zusammen mit den verschiedenen Roboterdaten angezeigt. So werden die Standorte mithilfe von Icons, die Roboterpfade mithilfe von Linien und die Roboterpositionen mithilfe von 3D-Modellen der Roboter dargestellt. Es gibt außerdem Buttons mit denen zwischen den verschiedenen Stockwerken navigiert werden kann und mit denen die Roboter ausgewählt werden können. Zusätzlich gibt es einen Button, der es ermöglicht den ausgewählten Roboter automatisch zu verfolgen und einen anderen Button, mit dem die Kameraposition wieder zur Ausgangsposition zurückgesetzt werden kann. Die Buttons sind abhängig ihrer Funktionen in den vier Ecken der Anwendung gruppiert.

Die Steuerung erweitert die Übersicht nur um weitere Funktionen und unterscheidet sich daher kaum von dieser. Abbildung 15 im Anhang zeigt das Mockup der Steuerung. Der einzige Unterschied zur Übersicht sind die zusätzlichen Buttons mit denen eine Lieferoute bestimmt, der Roboter zum Aufladen geschickt und der aktuelle Lieferauftrag abgebrochen werden kann. Die Abbildung 16 im Anhang zeigt das Mockup für das Routenplanungs-Popup, das sich öffnet, wenn der Lieferoute-Button ausgewählt wird. In dem Popup muss ein Ziel und ein Roboter ausgewählt werden, bevor der Auftrag gestartet werden kann. Um die Routenerstellung zu vereinfachen, sollen sowohl Roboter als auch Standorte zusätzlich über das Anklicken auf der Karte auswählbar sein.

Die Abbildung 17 zeigt das Mockup für die Verwaltung. Hier gibt es jeweils eine Liste für die Roboter und die Stockwerke. In der Liste der Roboter können verschiedene Informationen ebendieser, wie beispielsweise der systeminterne Identifier (ID) und die tägliche Neustartzeit ausgelesen werden. Auch können Einstellungen der Roboter, wie Name und Ausgabepunkt geändert werden. Zudem gibt es für jeden Roboter eine Vorschau der Übersicht, in der die Position des Roboters gezeigt wird. Sowohl in der Roboterliste als auch in der Stockwerkliste gibt es eine Auflistung der Standorte. In der Stockwerkliste gibt es außerdem eine Vorschau der Übersicht und zusätzlich ein Kontextmenü, über das in den Editiermodus gewechselt werden kann.

## 5.2 Implementierung

Es wurde ein Frontend entwickelt, das auf das BCB zugreift, um die Roboterdaten anzufragen. Wie das BCB hierbei mit den Robotern kommuniziert, wird im Abschnitt 2.1.4 erklärt. Das Frontend nutzt das Web-Framework React mit HTML, Sass und TypeScript. Die Vorteile von TypeScript und Sass gegenüber JavaScript und CSS, wie auch das Web-Framework React werden im Abschnitt 2.2.1 genauer erläutert. Zustandsinformationen die innerhalb mehrerer React-Komponenten genutzt werden, werden zentral durch Redux gespeichert. Die Grundstruktur des Frontend-Projekts wurde mithilfe des Shell Befehls `create-chayns-app` [24] aufgesetzt. Die Kompilierung ist mithilfe des npm-Pakets `chayns-toolkit` [25] konfiguriert. GUI-Elemente wie Buttons und Aufklapper werden durch die Komponentenbibliothek `chayns-components` [26] bereitgestellt. Außerdem werden die npm-Pakete `clsx` [70] und `fontawesome` [71] genutzt. Mithilfe von `clsx` lassen sich HTML-Klassennamen dynamisch anwenden. Das `fontawesome` npm-Paket wird genutzt, um Scalable Vector Graphics (SVG) Icons als Zeichenkette zu importieren.

### 5.2.1 Übersicht

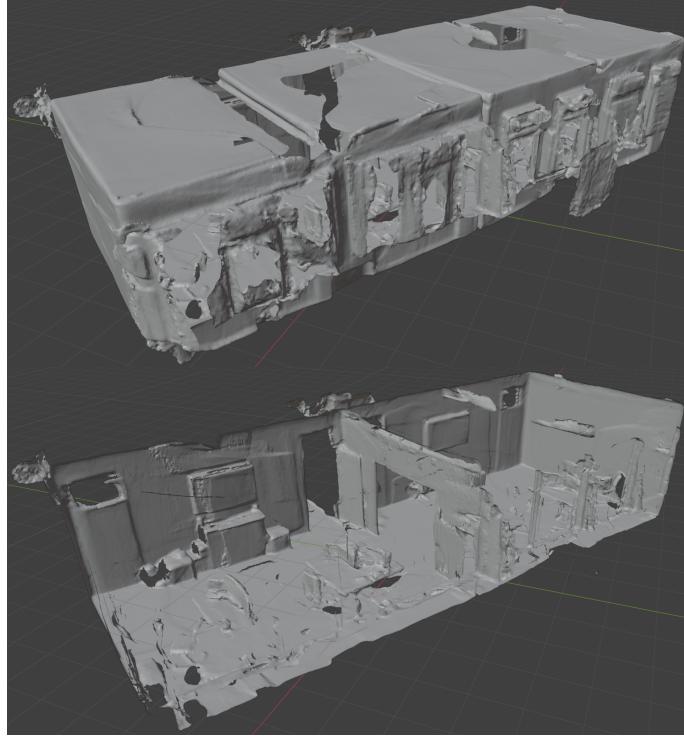
In der Übersicht werden die Roboterdaten in Kombination mit den Gebäudemodellen mithilfe von `deck.gl` angezeigt. Hierfür werden verschiedene Ebenen eingesetzt. Auch bietet die Übersicht eine Echtzeitaktualisierung der Roboter und Interaktivität für den Nutzer.

#### 5.2.1.1 Gebäudemodelle

Für den Prototyp sind die Uniform Resource Locator (URL)-Verweise der 3D-Modelle hart-kodiert, oder in anderen Worten in den Quelltext der Webanwendung eingebettet. Für ein

potenzielles Produktivsystem müssten die URL-Verweise in der Datenbank des BCBs abgespeichert werden. Die Modelle sind im chayns.space gespeichert, nutzen – aus Gründen, die im Abschnitt 4.2.2 erläutert werden – das glTF-Format und werden über die ScenegraphLayer angezeigt. Diese Ebene ist dafür ausgelegt, ein bestimmtes 3D-Modell beliebig oft an verschiedenen Positionen anzuzeigen [48], was auf die Hauptfunktion von deck.gl – die Visualisierung riesiger Geodaten Mengen [44, S. 3] – zurückzuführen ist. Unterschiedliche 3D-Modelle lassen sich nicht in einer ScenegraphLayer-Instanz einbinden, weshalb für jedes 3D-Modell eine eigene Instanz der ScenegraphLayer erzeugt werden muss. deck.gl ist für das Anzeigen von über 100 Ebenen gleichzeitig ausgelegt, wobei wahrscheinlich sogar bis zu 1000 Ebenen ohne große Performance Einbußen angezeigt werden können [72]. Im Prototyp besteht ein Stockwerk aus bis zu sechs 3D-Modellen und somit auch aus bis zu sechs ScenegraphLayers, weshalb davon auszugehen ist, dass nie mehr als 100 Modelle für die Darstellung eines Stockwerks erforderlich sind. In Anbetracht dessen, dass nie mehrere Stockwerke gleichzeitig angezeigt werden, ist es unwahrscheinlich, dass die beschriebene mehrfache Verwendung der Ebene negative Auswirkungen auf die Performance hat. Für die Ebenen ist das Backface Culling aktiviert. Mithilfe vom Backface Culling werden die von der Kamera weg gerichteten Polygone ausgeblendet [73]. Meist wird Backface Culling genutzt, um Polygone auszublenden, die sowieso hinter anderen Polygonen versteckt sind, wodurch die Darstellungsgeschwindigkeit erhöht wird. Im Fall der Gebäudemodelle sind die Polygone in den Raum hineingerichtet. Somit bewirkt das Backface Culling, dass die Wände und Decken, die dem Nutzer die Sicht in den Raum verdecken, ausgeblendet werden. In der Abbildung 5 wird dieser Effekt deutlich. So sieht man oben, dass ohne das Backface Culling nicht von außen in den Raum hineingesehen werden kann, während es unten mit aktiviertem Backface Culling möglich ist.

**Abbildung 5: Raummodell ohne und mit Backface Culling**



### 5.2.1.2 Roboterdaten

Die Roboterdaten werden über verschiedene Endpunkte des BCBs abgerufen. So gibt es einen Endpunkt zum Abrufen der Bezeichnungen der verschiedenen Standorte [74, Endpunkt 3] und einen Endpunkt zum Abrufen des Roboterstatus, Lieferauftrags und der Roboterposition [74, Endpunkt 29]. Für das Abrufen der Positionen aller Standorte und Roboterpfade gibt es keinen Endpunkt. Stattdessen können die Standorte und Pfade nur von Stockwerken angefragt werden, in denen sich zu dem Zeitpunkt Roboter befinden. Damit immer alle Daten aus allen Stockwerken abgerufen werden können, sind diese Daten im Prototyp für alle Stockwerke hartkodiert. Für ein potenzielles Produktivsystem müssten diese Daten in der Datenbank des BCBs gespeichert und regelmäßig mit den Robotern synchronisiert werden.

Im Gegensatz zu den Gebäudemodellen werden die Robotermodelle im OBJ-Format mit der SimpleMeshLayer [47] angezeigt. Mit der ScenegraphLayer gibt es das Problem, dass die Positionsänderungen der Modelle nicht animiert werden können, wodurch die SimpleMeshLayer brauchbarer – aber auch nicht ideal – ist. Laut der Dokumentation von deck.gl sollte das Animieren über die transitions Property in allen Ebenen möglich sein [75]. Deshalb handelt es sich bei dem Problem mit der ScenegraphLayer vermutlich um einen Bug. Wie im Abschnitt 4.2.2 erwähnt, lässt sich die Materialdatei des OBJ Formats nicht in der

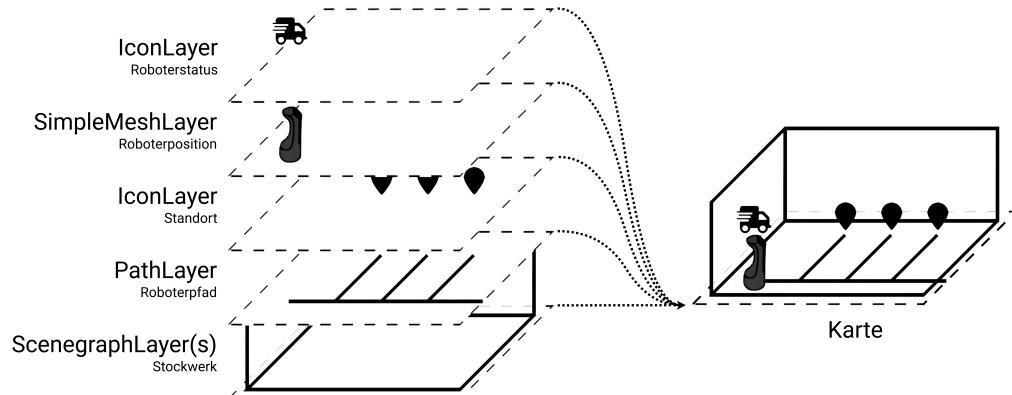
SimpleMeshLayer einbinden, weshalb die Roboter einfärbig angezeigt werden müssen. Das ist nicht unbedingt ein Nachteil, denn so können die Roboter durch eine herausstechende Farbe für den Nutzer besser sichtbar gemacht werden. Die Positionsänderungen der Roboter werden über die transitions Property animiert [75]. Allerdings lassen sich die Rotationsänderungen nicht animieren, was vermutlich auch auf einen Bug in deck.gl zurückzuführen ist.

Über den Robotermodellen wird mithilfe der IconLayer der aktuelle Status der Roboter angezeigt. Während in einer ScenegraphLayer-Instanz nur ein bestimmtes 3D-Modell angezeigt werden kann, können in einer IconLayer-Instanz mithilfe der getIcon Zugriffsfunktion verschiedene Bilder angezeigt werden [45]. So reicht im Gegensatz zur ScenegraphLayer eine IconLayer-Instanz aus, um alle Icons anzuzeigen. Die genutzten Icons stammen aus der Icon-Bibliothek Fontawesome und werden aus dem fortawesome npm-Paket als SVG-Zeichenkette importiert. Da die IconLayer nicht SVG-Zeichenketten unterstützt, werden diese in das Data-URL Format umgewandelt. Das Data-URL Format kann Daten als Base64-Zeichenkette innerhalb einer URL einbetten [76, S. 1], was von der SimpleMeshLayer ausgelesen werden kann [45].

Die verschiedenen Standorte werden über eine weitere IconLayer-Instanz angezeigt. Wie bei den Roboter-Zuständen werden hierfür verschiedene Fontawesome-Icons genutzt. Wurde ein Roboter ausgewählt und hat dieser einen Lieferauftrag, dann wird der Zielstandort farbig markiert. Die Pfade und virtuellen Wände werden über eine Instanz der PathLayer [77] dargestellt. Die virtuellen Wände werden gestrichelt und in einer anderen Farbe angezeigt, damit diese von den Roboterpfaden unterschieden werden können. Für die gestrichelte Darstellung wird die PathStyleExtension [78] genutzt. Der Boden der 3D-Modelle liegt relativ konstant auf der z-Koordinate – der vertikalen Position – 0. Aufgrund der Ungenauigkeiten die durch den LiDAR-Scan entstehen ist der Boden der 3D-Modelle nicht vollständig eben. Da die Positionen der Roboterdaten zweidimensional sind, und somit keine vertikale Position haben, besteht die Gefahr, dass diese an manchen Stellen unter dem Boden der 3D-Modelle verschwinden, wenn sie auf der z-Koordinate 0 angezeigt werden. Aus diesem Grund werden die Icons und Pfade an einer leicht erhöhten z-Koordinate positioniert.

Zusammenfassend wird für die Kartendarstellung die ScenegraphLayer, PathLayer, IconLayer und SimpleMeshLayer genutzt. In Abbildung 6 wird dargestellt wie sich die Karten darstellung aus den einzelnen Ebenen zusammensetzt. Bei der ScenegraphLayer muss beachtet werden, dass für jedes 3D-Modell eine eigene Instanz erstellt wird. Ansonsten stimmt die Menge der Ebenen-Instanzen mit denen im Schaubild überein.

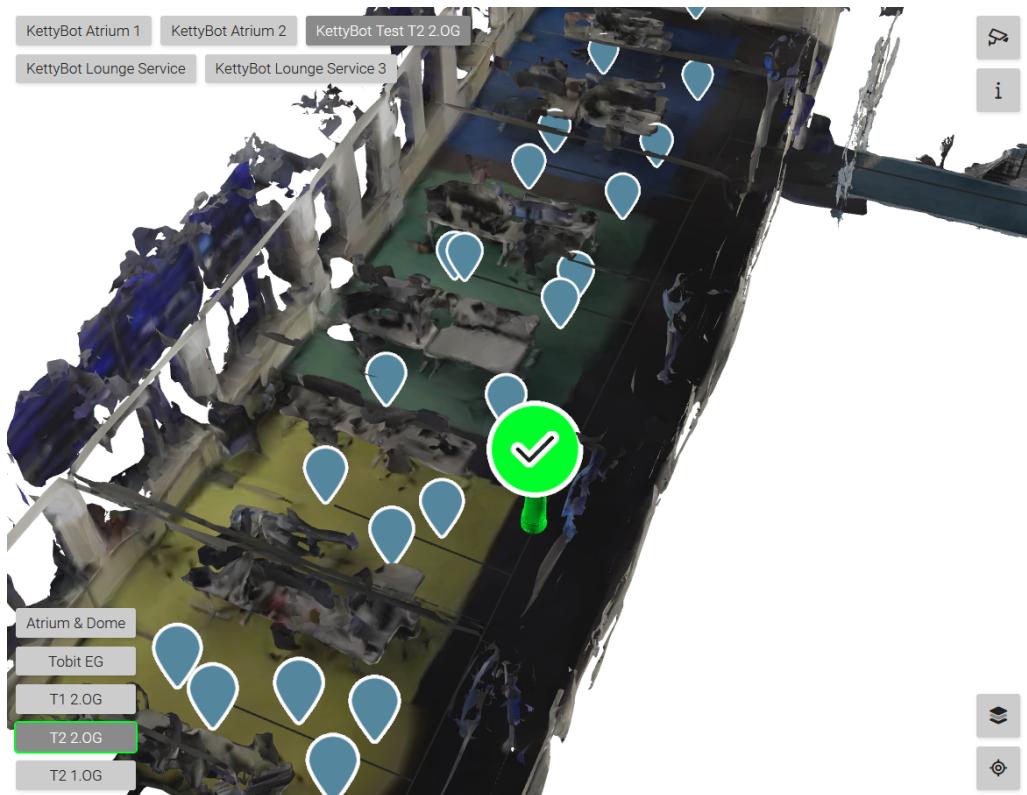
**Abbildung 6: Kartendarstellung**



Quelle: In Anlehnung an OpenJS Foundation [79]

Bei der Abbildung 7 handelt es sich um einen Screenshot aus der Übersicht im Prototyp. Man sieht alle erwähnten Ebenen, sowie die verschiedenen Buttons. Basierend auf dem Feedback aus den Usability Tests, die im Abschnitt 6.2.2 genauer beschrieben werden, wurden Buttons ergänzt, die es nicht im Mockup gibt.

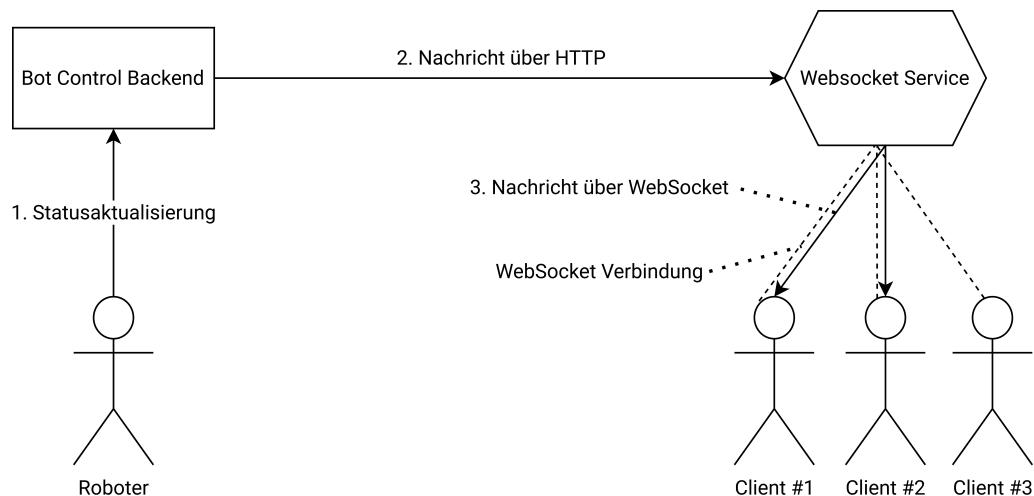
**Abbildung 7: Übersicht**



### 5.2.1.3 Echtzeit-Aktualisierung

Die Positionen sowie weitere Statusinformationen der Roboter, wie beispielsweise der aktuelle Auftrag und Akkuladung, werden mithilfe einer indirekten Verbindung zwischen dem Prototyp und dem BCB regelmäßig aktualisiert. Hierfür wird der im Abschnitt 2.2.2 erwähnte WebSocket-Service genutzt. In der Abbildung 8 ist der Ablauf einer Statusaktualisierung vereinfacht dargestellt. Aktualisiert ein Roboter seine Position, wird die entsprechende Information an das BCB gesendet. Wie die Kommunikation zwischen Robotern und BCB funktioniert wird im Abschnitt 2.1.4 erklärt. Das BCB sendet daraufhin eine Nachricht an den WebSocket-Service, der diese Information wiederum an alle verbundenen Clients schickt, die Nachrichten des BCBs erwarten. So sieht man in der Abbildung auch, dass der dritte Client keine Nachricht empfängt, da er Nachrichten eines anderen Systems erwartet.

**Abbildung 8: Kommunikationsweg von Statusaktualisierungen der Roboter**



Die über den WebSocket-Service empfangene Statusaktualisierung wird zentral mit Redux gespeichert, sodass dem Nutzer direkt die aktualisierten Informationen angezeigt werden können.

### 5.2.1.4 Interaktion

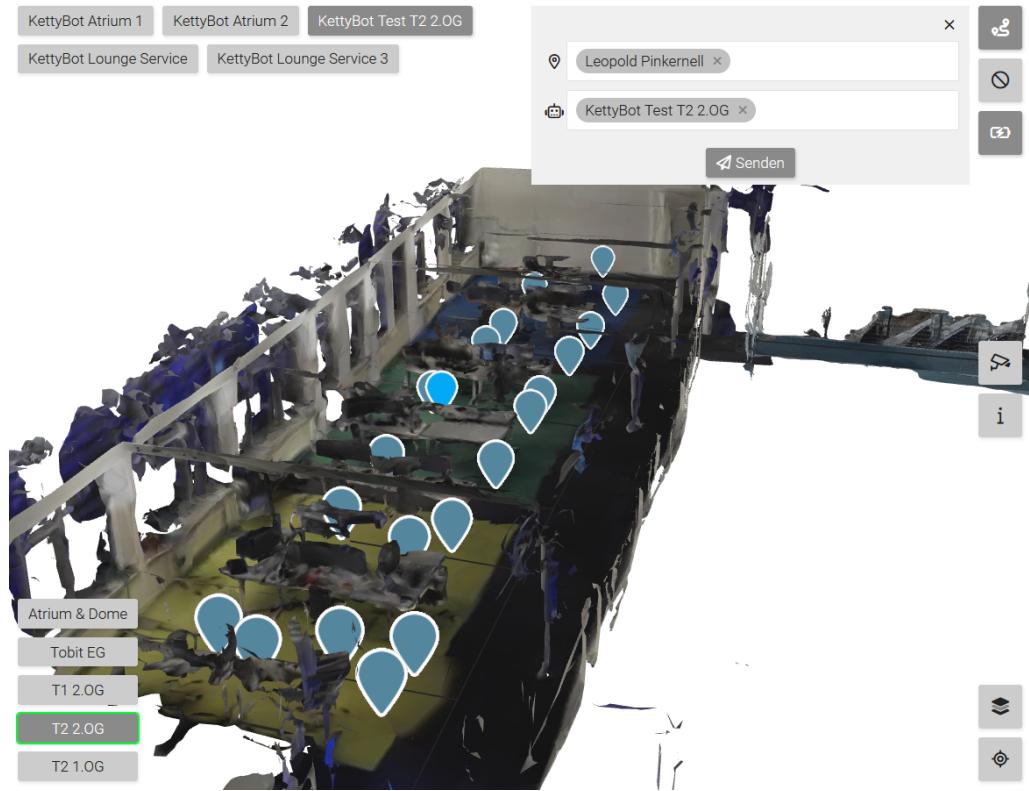
Die Roboter, sowie die Standorte sind mithilfe der `onClick` Property der entsprechenden Ebene [80] auswählbar. Ausgewählte Standorte und Roboter werden verfärbt angezeigt. Außerdem erscheint beim Auswählen eines Roboters ein Button über den sich weitere Informationen wie die Akkuladung anzeigen lassen. Schwebt die Maus über einem Standort oder Roboter, dann wird mithilfe der `getTooltip` Property [81] ein Tooltip angezeigt, in

dem der Name des entsprechenden Objekts und weitere wichtige Informationen stehen. Es gibt zudem einen Button, über den einem Roboter gefolgt werden kann. Die Kamera wird hierfür durch den FlyToInterpolator [82] zum ausgewählten Roboter bewegt. Beim Folgen eines Roboters wird die Kameraposition mithilfe der transitionDuration Property [83] animiert.

### 5.2.2 Steuerung

Wie im Abschnitt 5.1 beschrieben, gibt es drei Aktionen die zum Steuern der Roboter ausgeführt werden können: Lieferauftrag, Laden und Abbrechen. Mit dem Laden und Abbrechen wird der aktuelle Lieferauftrag abgebrochen, worauf der Nutzer auch über einen Bestätigungsdialog hingewiesen wird. Für das Starten eines Lieferauftrags muss ein Ziel und ein Roboter angegeben werden. Hierfür gibt es Inputs, mit denen nach Standorten und Robotern gesucht werden kann. Bei den Inputs handelt es sich um die PersonFinder React-Komponente [84] der chayns-components, die eigentlich für die Suche nach chayns Nutzern genutzt wird, im Prototyp aber für die Suche nach Robotern und Standorten konfiguriert ist. Das Ziel und der Roboter lassen sich außerdem über das Anklicken auf der Karte auswählen. Die Roboter können zudem über ihre Buttons ausgewählt werden. Bestimmte Standorte wie Türen oder Fahrstühle können nicht als Zielstandorte ausgewählt werden. Aus diesem Grund sind diese weder im Input, noch auf der Karte auswählbar. Zum endgültigen Ausführen der drei Aktionen werden die entsprechenden Endpunkte Robot/Call [74, Endpunkt 38], Robot/Charge [74, Endpunkt 40] und Robot/Cancel [74, Endpunkt 41] im BCB aufgerufen. Die Abbildung 9 zeigt die Steuerung und das Routenplanungs-Popup. Im Popup sind bereits Ziel und Roboter eingestellt. Der ausgewählte Standort ist auf der Karte farblich markiert.

**Abbildung 9: Steuerung und Routenplanung**

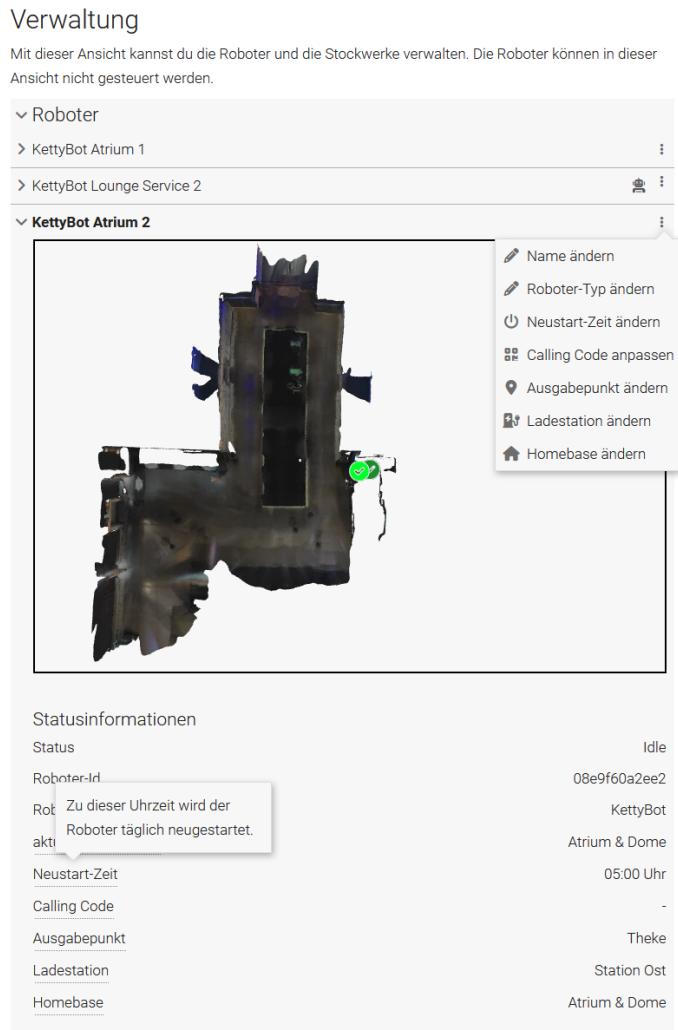


### 5.2.3 Verwaltung

Die Verwaltung ist nicht besonders komplex, da die Daten der Roboter und Stockwerke in jeweils einem Array strukturiert sind und somit leicht mithilfe von React gemappt werden können. Hiermit ist gemeint, dass die Arrays mithilfe der map Funktion zu Arrays an React-Komponenten umgewandelt und daraufhin angezeigt werden können [85, S. 35-36].

In der Roboterliste werden im Gegensatz zum Mockup mehr Statusinformationen angezeigt. Auch können mehr Einstellungen der Roboter geändert werden. Da die Übersicht der verschiedenen Standorte auch über die Liste der Stockwerke ersichtlich ist und dadurch redundant ist, wurde diese aus der Roboterliste entfernt. Bei der Abbildung 10 handelt es sich um einen Screenshot der Verwaltung im Prototyp. Man sieht einen geöffneten Roboter-Eintrag, das Kontextmenü, über das Einstellungen geändert werden können und einen Tooltip, in dem eine Statusinformation erläutert wird.

**Abbildung 10: Verwaltung**



Die Stockwerkliste unterscheidet sich nur geringfügig vom Mockup. So werden die Standorte im Gegensatz zum Mockup gruppiert nach der Art des Standorts aufgelistet. Diese Gruppierung ist hilfreich, da die Art des Standortes nicht unbedingt aus dem Namen ersichtlich ist.

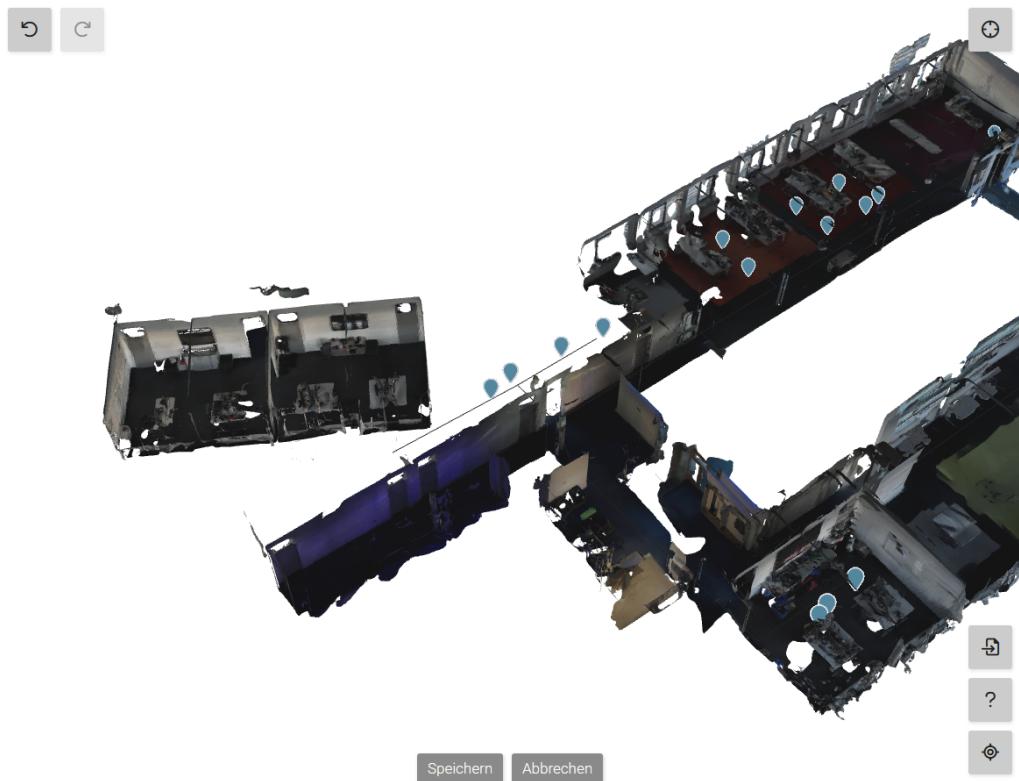
Sowohl in der Roboter- als auch in der Stockwerkliste gibt es eine Vorschau des entsprechenden Stockwerks, um die Position des Roboters oder die Positionen der Standorte zu zeigen. Hierbei handelt es sich um die Übersichtskarte mit reduzierten Funktionen. Die Vorschau ist im Mockup automatisch geöffnet, was ein Problem ist, da die entsprechende deck.gl Karteninstanz – aufgrund des Verhaltens der genutzten Aufklapper-Komponente – beim Öffnen des Aufklappers initialisiert wird. Die Initialisierung der Karteninstanz ist rechnerisch aufwändig und verursacht beim ersten Öffnen des Aufklappers starke Ruckler in der Aufklapper-Animation. Um diese Ruckler zu verhindern wird ein Button angezeigt,

über den die Karte manuell initialisiert werden kann. Dadurch gibt es die Rückler beim Klicken des Buttons und nicht beim Öffnen des Aufklappers, was weniger störend ist.

#### 5.2.4 Editiermodus

In der Verwaltung, sowie in der Übersicht gibt es die Möglichkeit in den Editiermodus eines Stockwerks zu wechseln. In diesem können die Roboterdaten und 3D-Modelle manuell synchronisiert werden. Der Editiermodus ähnelt der Übersicht und unterscheidet sich nur durch andere Buttons und die Editermöglichkeiten. Da der Einsatz der Steuerungs- und Umschalttaste nötig ist, kann der Editiermodus nicht an Mobilgeräten genutzt werden. Im Editiermodus hat der Nutzer die Möglichkeit neue 3D-Modelle zu importieren. Hierfür gibt es einen einfachen Dateiinput der nur Dateien im .glb Format akzeptiert. Wie bereits erwähnt sind die 3D-Modelle im Prototyp hartkodiert. Entsprechend werden Änderungen sowie neu importierte 3D-Modelle nur für die aktuelle Sitzung gespeichert und gehen verloren, wenn die Anwendung erneut geöffnet und somit eine neue Sitzung gestartet wird. Mithilfe der in deck.gl integrierten Events onDragStart, onDrag und onDragEnd [80] kann das angeklickte 3D-Modell per Ziehen der Maus verschoben und rotiert werden. So wird das ausgewählte Modell beim Ziehen entweder verschoben oder rotiert, je nachdem ob die Steuerungs- oder Umschalttaste gedrückt wurde. Das Verschieben und Rotieren kann mithilfe der Tastenkombination Strg + Z rückgängig gemacht und mit Strg + Y wiederholt werden. Hierfür existieren zwei Stapspeicher in denen die vergangenen und rückgängig gemachten Aktionen per push hinzugefügt und per pop wieder herausgenommen werden. Bei Abbildung 11 handelt es sich um einen Screenshot des Editiermodus. Oben Links sind die Buttons zum rückgängig machen und wiederholen. Mit dem Button oben rechts kann die initiale Kameraposition geändert werden. Unten rechts sind Buttons zum Importieren neuer Modelle – wobei das Importieren nicht implementiert ist –, zum Anzeigen verschiedener Tastenkombinationen und zum Zurücksetzen der Kameraposition. Außerdem gibt es unten Buttons zum Speichern und Abbrechen des Editierens.

**Abbildung 11: Editiermodus**



Da der Boden bei allen 3D-Modellen ungefähr auf derselben z-Koordinate positioniert ist, müssen diese durch den Nutzer nicht weiter an der z-Achse verschoben werden. Im Vergleich zu den Roboterdaten sind die 3D-Modelle immer um  $90^{\circ}$  an der z-Achse und einen beliebigen Wert an der y-Achse rotiert, während die Rotation der x-Achse zwischen 3D-Modellen und Roboterdaten bereits übereinstimmt. Die 3D-Modelle werden im Editiermodus automatisch um  $-90^{\circ}$  an der z-Achse rotiert und müssen vom Nutzer somit nur noch an der y-Achse rotiert werden. Die Roboterdaten und 3D-Modelle teilen sich bereits denselben Maßstab, weshalb der Nutzer nicht die Möglichkeit braucht, die Modelle zu skalieren. Es gilt zu beachten, dass im Prototyp 3D-Modelle erwartet werden, die mit der Scaniverse App erzeugt wurden. In einem Produktivsystem sollten auch andere Quellen genutzt werden können. Die genannten Annahmen, dass die Modelle nicht um die z-Koordinate verschoben, nicht um die x- und z-Achse rotiert und nicht skaliert werden müssen, gelten dann nicht mehr. Somit bräuchte der Nutzer in einem Produktivsystem die Möglichkeit Modelle an allen Achsen zu verschieben und um alle Achsen zu rotieren. Auch müsste der Nutzer die Modelle skalieren können.

### 5.3 Softwaretests

Da der Prototyp als evolutionärer Prototyp entwickelt wurde und somit dafür entwickelt wurde, dass ein Produktivsystem auf diesem aufbauen kann, ergibt es Sinn, den Code automatisiert zu testen. Mit dem Test-Framework Jest [86] wurden Modultests geschrieben. Das Testen der deck.gl Ansicht und Ebenen konnte nicht umgesetzt werden. So gibt es zwar den SnapshotTestRunner [87], mit dem das Framework testbar ist, dieser ist aber unzureichend dokumentiert, sodass die Tests nicht erfolgreich implementiert werden konnten. Da die deck.gl Ansicht das Herzstück des Prototyps ist, konnten keine Integrations-tests implementiert werden. Die Testabdeckung ist insgesamt unzureichend, aber auch nicht signifikant ausbaubar. Es wurden Modultests für verschiedene Utility-Funktionen, für die komplexeren Redux-Zugriffsfunktionen und für React-Komponenten der Übersicht und Steuerung geschrieben. Ein Großteil der Redux Implementierung wird bewusst nicht getestet, da es sich hierbei um Implementierungsdetails handelt, die nach Dodds nicht getestet werden sollten [88]. Diese Vorgehensweise wird auch vom Redux Maintainer Erikson empfohlen [89]. Die Modultests sind in die, im Folgenden erklärte, Deployment-Pipeline integriert.

### 5.4 Deployment

Für das Veröffentlichen des Prototyps wird GitHub Actions in Kombination mit GitHub Pages genutzt. Mit GitHub Actions lässt sich die Build-, Test- und Deployment-Pipeline eines Projekts automatisieren [90]. Bei GitHub Pages handelt es sich um einen Hosting-Dienst, der in GitHub integriert ist und aus Repositories statische Websites erstellen kann [91]. So wird mithilfe der actions-gh-pages Github Action [92] bei der Aktualisierung des Haupt-Banches automatisch die Deployment-Pipeline ausgelöst. In dieser werden die Modultests durchgeführt. Falls diese erfolgreich waren, wird der Build erstellt. Das GitHub Repository ist so konfiguriert, dass der erstellte Build automatisch mit GitHub Pages veröffentlicht wird.

Die Anwendung verwendet verschiedene Funktionen der chayns-api [27], wie etwa das Anfordern des Zugangstokens eines angemeldeten Nutzers, ohne den Funktionen des BCBs nicht aufgerufen werden können. Die Funktionen der chayns-api sind nur innerhalb der chayns Umgebung nutzbar, weshalb die Anwendung nur funktioniert, wenn sie – wie in der Dokumentation des create-chayns-app Befehls beschrieben [24] – als Custom Page auf einer chayns Seite eingebunden ist. Der Zugriff auf die meisten Endpunkte des BCBs ist so eingeschränkt, dass diese nur von unternehmensinternen chayns Seiten aus

aufgerufen werden können. Auf anderen chayns Seiten können die Steuerungs- und Verwaltungsfunktionen deshalb nicht oder nur eingeschränkt genutzt werden.

## 6 Evaluierung des Prototyps

Im Folgenden wird gezeigt welche funktionalen Anforderungen erfüllt werden. Auch wird ausgewertet, inwieweit die nicht funktionalen Anforderungen an die Benutzerfreundlichkeit und Effizienz erfüllt werden.

### 6.1 Funktionale Anforderungen

Im Abschnitt 3.2 werden die funktionalen Anforderungen erläutert. Die meisten dieser Anforderungen werden durch den Prototyp erfüllt, weshalb hier nur die nicht erfüllten Anforderungen erwähnt werden. Bei dem Prototyp handelt es sich zwar – wie in den Anforderungen definiert – um eine responsive Webanwendung, sie funktioniert allerdings nicht auf allen Geräten vollständig. So können die für die Gebäudemodelle genutzten glTF-Modelle nicht im Safari-Browser angezeigt werden, weil die Modelle das WebP Bildformat für die Texturierung nutzen und dieses Format noch nicht vollständig von Safari unterstützt wird [93]. Da es sich hierbei um eine Beschränkung des Safari-Browsers handelt, die in Zukunft von Apple behoben werden sollte und weil alle anderen Funktionen des Prototyps auch im Safari-Browser funktionieren, wurde hierfür kein aufwändiger Workaround entwickelt. Während die Positionsänderungen der Roboter animiert werden, ist das bei den Rotationsänderungen aufgrund eines Bugs in deck.gl nicht der Fall. Aus diesem Grund ist die Anforderung, dass der Roboter in der Übersicht fährt, nicht vollständig, aber ausreichend erfüllt.

Die Anforderungen an die Methode zur Gebäudemodell-Generierung konnten mit dem Einsatz des LiDAR-Scannens weitestgehend erfüllt werden. Zum Scannen wird ein neueres iPhone benötigt, welches ein Nutzer unter Umständen bereits besitzt. Das Generieren der Modelle erfordert wenig Aufwand, wobei dieser davon abhängig ist, wie gründlich gescannt wird. Insgesamt ist für das Scannen nur wenig Know-how nötig. Die entstandenen Modelle müssen für den Prototyp manuell komprimiert werden, wofür beispielsweise das im Abschnitt 4.2.2 erwähnte Webtool OptimizeGLB infrage kommt. In einem Produktivsystem könnten die Modelle aber auch mithilfe des glTF-Transform npm-Pakets [94] automatisch beim Import in den Editiermodus komprimiert werden. Zum einen variiert die Qualität der erzeugten Modelle je nachdem, wie gründlich die Scans durchgeführt wurden. Zum anderen wird sie auch durch die eingesetzte Komprimierungsmethode beeinflusst. Insbesondere kleinere Ungenauigkeiten können in den erzeugten Modellen ignoriert werden, solange diese Ungenauigkeiten keinen Einfluss auf die Übersichtlichkeit des Modells haben.

## 6.2 Benutzerfreundlichkeit

Die Benutzerfreundlichkeit wird durch die Einhaltung der Usability Entscheidungsregeln gefördert und durch die Auswertung der Usability Tests bewertet. Beides wird im Folgenden genauer erörtert.

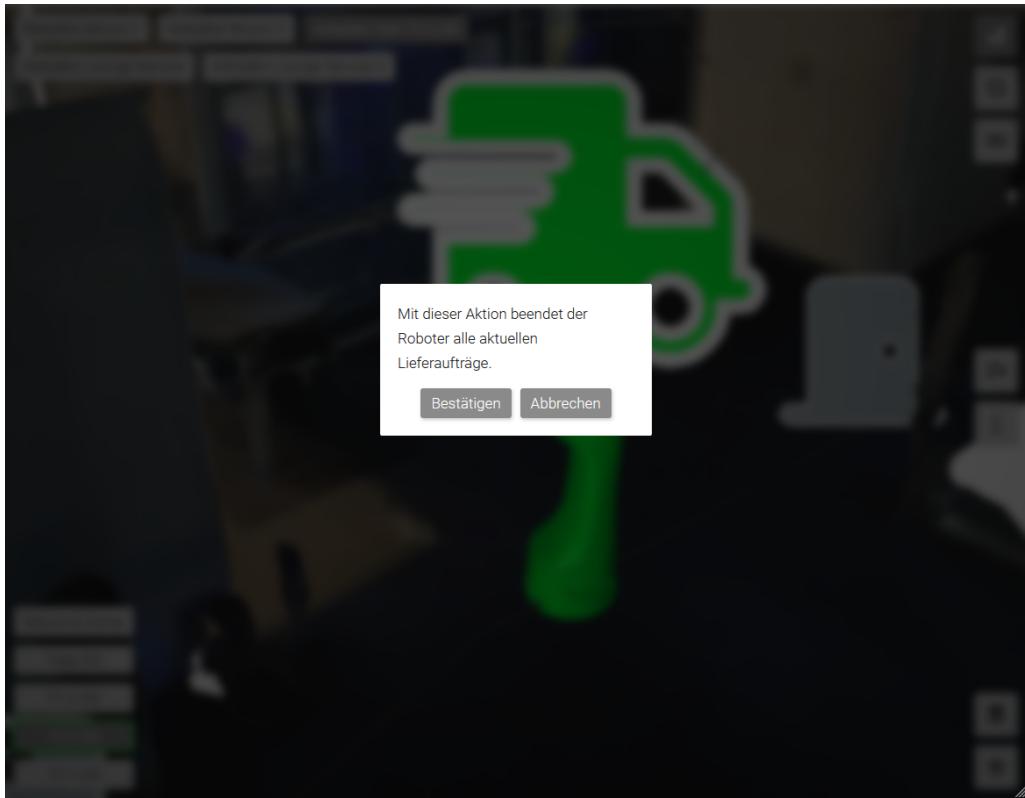
### 6.2.1 Usability Entscheidungsregeln

Die Usability Entscheidungsregeln konnten weitestgehend eingehalten werden. Im Folgenden wird aufgezeigt wie ein paar Entscheidungsregeln konkret eingehalten werden.

Die erste Regel besagt, dass der Nutzer immer innerhalb einer angemessenen Zeitspanne durch geeignete Rückmeldungen mitbekommen sollte, was gerade passiert [53, Regel 1]. Diese Regel wird durch verschiedene Features in der Übersicht und Steuerung erfüllt. Zum einen gibt es die Echtzeit-Aktualisierungen des Roboterstandorts und -Status über die WebSocket-Verbindung mit dem BCB. Zum anderen werden Statusänderungen auch über die Farben der Buttons signalisiert. Außerdem gibt es einen Wait-Cursor, um Ladevorgänge anzuzeigen.

Da Benutzer oft versehentlich Aktionen ausführen, gibt es die dritte Regel, die klar gekennzeichnete Abbruchoptionen fordert, damit unerwünschte Aktionen abgebrochen oder rückgängig gemacht werden können [53, Regel 3]. Um die Kamera wieder in die Ausgangsposition zu bringen, wenn diese versehentlich an eine ungewünschte Position bewegt wurde, gibt es in der Übersicht einen entsprechenden Button. In der Steuerung gibt es außerdem einen Button, mit dem der aktuelle Lieferauftrag des Roboters abgebrochen werden kann. Auch gibt es in der Steuerung Bestätigungsdialoge, mit denen neue Aktionen bestätigt werden müssen. Im Editiermodus gibt es die Möglichkeit, das Verschieben und Rotieren von Objekten rückgängig zu machen oder zu wiederholen. Auch gibt es im Editiermodus einen Button, mit dem das Editieren ohne Speichern abgebrochen werden kann. In Abbildung 12 wird der Bestätigungsdialog gezeigt, der beim Abbrechen der Roboteraktionen geöffnet wird.

**Abbildung 12: Bestätigungsdialog**



In der fünften Regel geht es darum, dass Aktionen, die Fehler auslösen, verhindert werden sollten [53, Regel 5]. In der Steuerung ist es zum Beispiel nicht möglich, ungültige Standorte in der Routenplanung einzugeben. Währenddessen wird diese Regel in der Verwaltung durch die bereits erwähnten Bestätigungsdialoge erfüllt. Im Editiermodus gibt es die bereits erwähnte Rückgängigmachen- und Wiederholen-Funktion.

In der siebten Regel geht es darum, dass bestimmte, häufig genutzte Aktionen durch Shortcuts schneller ausführbar sein sollten [53, Regel 7]. So können Zielstandort und Roboter für einen Lieferauftrag in der Steuerung sowohl über die Inputs als auch über die Karte ausgewählt werden. Im Editiermodus gibt es außerdem Tastenkombinationen für das Rückgängigmachen und Wiederholen.

Auch die anderen Usability Entscheidungsregeln wurden während der Entwicklung beachtet und weitestgehend erfüllt. Das bedeutet nicht automatisch, dass der Prototyp auch wirklich benutzerfreundlich ist. Um das zu bewerten, folgen die Usability Tests.

### 6.2.2 Usability Tests

Während eines Großteils der Implementierung wurde die Benutzerfreundlichkeit nur oberflächlich durch den Entwickler bewertet, wodurch viel Zeit gespart wurde. Da sich die Entwickler von Systemen nicht zum Testen ebendieser eignen, konnten viele Probleme allerdings nicht identifiziert werden. Aus diesem Grund wurden Usability Tests mit ausgewählten Testpersonen durchgeführt, nachdem alle Funktionen des Prototyps erfolgreich implementiert wurden. Da die Benutzerfreundlichkeit des fertigen Prototyps bewertet werden soll und ein Vergleich zu vorherigen Versionen unwichtig ist, wurden qualitative statt quantitative Usability Tests durchgeführt.

Der Aufbau der Usability Tests basiert maßgeblich auf verschiedenen Artikeln der Nielsen Norman Group. So wurden die Aufgaben nach dem Stepped-User-Tasks-System [95] formuliert und es wurde während der Durchführung der Tests darauf geachtet, dass die Testpersonen die Thinking-Aloud-Methode [96] einsetzen. Die Usability Tests wurden in zwei Runden mit jeweils fünf Personen durchgeführt. So konnten die gefundenen Probleme nach der ersten Runde behoben werden, bevor die zweite Runde durchgeführt wurde. Für die Usability Tests wurden drei Aktivitäten vorbereitet, die die Testpersonen nacheinander durchführen sollten. Die Aktivitäten decken direkt oder indirekt einen großen Teil der Funktionen des Prototyps ab. Konkret beschäftigen sich die Aktivitäten mit der Übersicht, der Steuerung und dem Editiermodus, aber nicht mit der Verwaltung. Dieser Umstand wird im Abschnitt 6.2.2.2 genauer erläutert. In der ersten Aktivität müssen Position und Akkustand eines bestimmten Roboters gefunden werden. Daraufhin muss der Roboter in der zweiten Aktivität zu einem oder mehreren Standorten und dann zurück zur Ladestation geschickt werden. In der dritten Aktivität muss ein 3D-Modell im Editiermodus korrekt positioniert werden. In der dritten Aktivität wird nicht nur die Benutzerfreundlichkeit des Editierens überprüft, sondern auch die Zugänglichkeit des Editiermodus bewertet.

Um die geplanten Aktivitäten zu prüfen, wurde zunächst ein Pilottest durchgeführt. Mithilfe von Pilottests können Probleme im Design von Tests gefunden werden, sodass diese vor der Durchführung der richtigen Tests korrigiert werden können [97]. Mithilfe des Pilottests konnten die Aktivitäten optimiert werden. Außerdem konnten die Ergebnisse des Pilottests bezüglich der Benutzerfreundlichkeit des Prototyps ausgewertet werden, sodass viele Probleme behoben wurden, bevor die anderen Tests durchgeführt wurden.

#### 6.2.2.1 Erster Testdurchlauf

In den beiden folgenden Tabellen werden die Ergebnisse des ersten Usability Testdurchlaufs zusammengefasst. Der Pilottest wird zu dieser Runde dazugezählt und ist in den Ta-

bellen als T0 gekennzeichnet. In Tabelle 1 wird dargestellt, welche Probleme bei welcher Testperson aufgefallen sind. So sieht man, dass die meisten Probleme, die im Pilottest aufgetreten sind, danach nicht mehr auftraten. Dies lässt sich darauf zurückführen, dass sie vor der Durchführung der restlichen Tests behoben wurden. Man kann zudem sehen, dass den meisten Testpersonen mindestens ein Problem aufgefallen ist, das von keiner anderen Testperson bemerkt wurde. Hierdurch zeigt sich, dass durch weniger Testpersonen weniger Probleme gefunden worden wären. In Tabelle 5 im Anhang werden die gefundenen Probleme genauer beschrieben. Bei der Fehlerbehebung wurden die Probleme priorisiert, die besonders vielen Testpersonen aufgefallen sind. Im Rahmen der Usability Tests gab es außerdem zusätzlich Feedback der Testpersonen, das in der Implementierung berücksichtigt wurde.

**Tabelle 1: Gefundene Probleme in erster Usability Testrunde**

Problem Nr.	T0	T1	T2	T3	T4	T5
1	X			X		
2	X					
3	X					
4	X					
5	X					
6	X		X			
7	X					
8	X				X	
9		X				
10		X				
11		X	X	X	X	
12	X	X	X			
13			X			
14			X	X		
15	X		X			
16				X		
17				X		
18						X
19						X
20						X

In Tabelle 2 sind die Aktivitäten in verschiedene Aktionen aufgeteilt, die bei der Ausführung der Aktivität durchgeführt werden können, aber nicht unbedingt durchgeführt werden müssen. Die Werte zeigen, wie gut eine Aktion von einer Testperson durchgeführt werden konnte. Je niedriger der Wert, desto besser konnten die Aktionen durchgeführt werden. Kein Wert bedeutet, dass die Testperson die Aktion nicht durchgeführt hat, da sie nicht für den Erfolg der Aktivität benötigt wurde. Aus der Tabelle wird somit die Benutzerfreundlichkeit

keit in den entsprechenden Teilen der Anwendung ersichtlich. Man sieht, dass die meisten Aktionen nach dem Pilottest deutlich besser durchgeführt werden konnten, was auf die erwähnten Anpassungen am Prototyp zurückzuführen ist. Die Tabelle zeigt, dass die meisten Aktionen zuverlässig durchgeführt werden konnten, sie zeigt aber auch, dass die Benutzerfreundlichkeit an einigen Stellen noch ausbaufähig ist. Insbesondere der Editiermodus hat Mängel, aber auch in der Übersicht und Steuerung gibt es kleinere Probleme.

**Tabelle 2: Bewertung der durchgeführten Aktionen in erster Usability Testrunde**

Aktion	T0	T1	T2	T3	T4	T5
<b>Aktivität 1 (Übersicht)</b>						
Roboter mit Button ausgewählt	1	1	1	1	1	1
Akkustand gefunden	2	1	1	2	2	1
Roboter mit Folgen-Button gefunden	2	1	1	-	-	-
Roboter mit Karte gefunden	-	-	-	2	1	1
<b>Aktivität 2 (Steuerung)</b>						
Lieferauftrag-Button gefunden	1	1	1	1	1	1
Standort mit Personfinder ausgewählt	2	-	1	1	1	1
Standort mit Karte ausgewählt	-	1	-	-	1	1
Lieferauftrag gestartet	3	2	1	1	1	1
Roboter zur Ladestation geschickt	-	1	1	1	1	1
<b>Aktivität 3 (Editiermodus)</b>						
Editiermodus über Adminansicht	3	-	-	-	-	-
Editiermodus über Nutzeransicht	-	1	1	1	1	1
Steuerung verstanden	-	1	3	1	2	1
Undo/Redo genutzt	-	-	-	-	-	-
Modell positioniert	2	1	2	1	1	1

### 6.2.2.2 Zweiter Testdurchlauf

Die Ergebnisse der ersten Usability Tests Runde wurden in der darauffolgenden Implementierungsphase einbezogen. Verschiedene Probleme wurden behoben und das Feedback wurde umgesetzt. Daraufhin wurde eine neue Runde an Usability Tests mit fünf neuen Testpersonen durchgeführt. Die Ergebnisse sind in den folgenden zwei Tabellen abgebildet. Die Tabellen folgen der Struktur der vorherigen Tabellen. So zeigt Tabelle 3, welche Testperson welche Probleme hatte, und Tabelle 4, wie gut Aktionen durchgeführt werden konnten. In Tabelle 6 im Anhang werden die gefundenen Probleme beschrieben.

Man sieht in Tabelle 3, dass deutlich weniger Probleme aufgefallen sind. Neben den Problemen gab es auch noch weiteres Feedback, dass sich aber vor allem auf Rechtschreibung, Zeichensetzung und Benennung beschränkt. Außerdem ist aus dem Feedback erkennlich, dass das erste Auffinden von bestimmten Funktion etwas dauern kann, die Be-

dienung dieser Funktionen dann aber einwandfrei funktioniert. Es ist zu erwarten, dass die Bedienung des Prototyps bei einer erneuten Nutzung deutlich leichter ist, da die Funktionen dann nicht mehr lange gesucht werden müssen.

**Tabelle 3: Gefundene Probleme in zweiter Usability Testrunde**

Problem Nr.	T1	T2	T3	T4	T5
1	X				
2	X				
3		X			
4			X		
5				X	
6					X
7					X

Auch in Tabelle 4 fällt auf, dass deutlich weniger Probleme aufgetreten sind. So gab es nur bei der zweiten und fünften Testperson geringfügige bis erhebliche Probleme. Die zweite Testperson hatte erst versucht den Roboter direkt über eine Auswahl auf der Karte zur Ladestation zu schicken, während die fünfte Testperson Probleme damit hatte, den Editiermodus zu finden, wobei hierfür ohne Erfolg in der Verwaltung gesucht wurde, bevor der Editiermodus in der Übersicht gefunden wurde. So handelt es sich um Probleme, die bei einer erneuten Nutzung der Anwendung nicht mehr auftreten würden. Nachdem die Tests ausgewertet wurden, wurde der Prototyp erneut angepasst, wodurch die beiden genannten Probleme nicht mehr auftreten sollten. Auch die in Tabelle 3 aufgelisteten Probleme wurden behoben.

**Tabelle 4: Bewertung der durchgeführten Aktionen in zweiter Usability Testrunde**

Aktion	T1	T2	T3	T4	T5
<b>Aktivität 1 (Übersicht)</b>					
Roboter mit Button ausgewählt	1	1	1	1	1
Akkustand gefunden	1	1	1	1	1
Roboter mit Folgen-Button gefunden	-	-	-	-	-
Roboter mit Karte gefunden	1	1	1	1	1
<b>Aktivität 2 (Steuerung)</b>					
Lieferauftrag-Button gefunden	1	-	-	-	1
Standort mit Personfinder ausgewählt	1	-	-	-	1
Standort mit Karte ausgewählt	-	1	1	1	-
Lieferauftrag gestartet	1	1	1	1	1
Roboter zur Ladestation geschickt	1	2	1	1	1
<b>Aktivität 3 (Editiermodus)</b>					
Editiermodus über Adminansicht	-	-	-	-	3
Editiermodus über Nutzeransicht	1	1	1	1	2
Steuerung verstanden	1	1	1	1	1
Undo/Redo genutzt	-	-	1	-	1
Modell positioniert	1	1	1	1	1

Da die Menge der gefundenen Probleme mit dem zweiten Durchlauf der Tests stark abgenommen hat und da die Aktivitäten fast ohne Probleme durchgeführt wurden, wurde auf einen dritten Durchlauf verzichtet. Es ist nicht zu erwarten, dass ein dritter Durchlauf bedeutende neue Erkenntnisse liefern würde, da aufgrund der Ergebnisse der vorherigen Tests angenommen werden kann, dass nur noch wenige Probleme bestehen, die auch mit einem erneuten Durchlauf nicht unbedingt gefunden werden können. Es ist wichtig zu beachten, dass mit den Usability Tests nicht alle Funktionen des Prototyps getestet wurden. Stattdessen wurden nur die wichtigsten Funktionen getestet die mit deck.gl in Verbindung stehen und somit für die Beantwortung der Forschungsfrage größere Relevanz haben. Ob die Liste der Roboter und Stockwerke in der Verwaltung besonders Benutzerfreundlich ist, ist für die Forschungsfrage nicht so relevant, da die Liste sehr simpel ist und keine besonderen Technologien nutzt. In Kombination mit der Einhaltung der Usability Entscheidungsregeln ist davon auszugehen, dass das Ziel der Benutzerfreundlichkeit ausreichend erfüllt wurde.

### 6.3 Effizienz

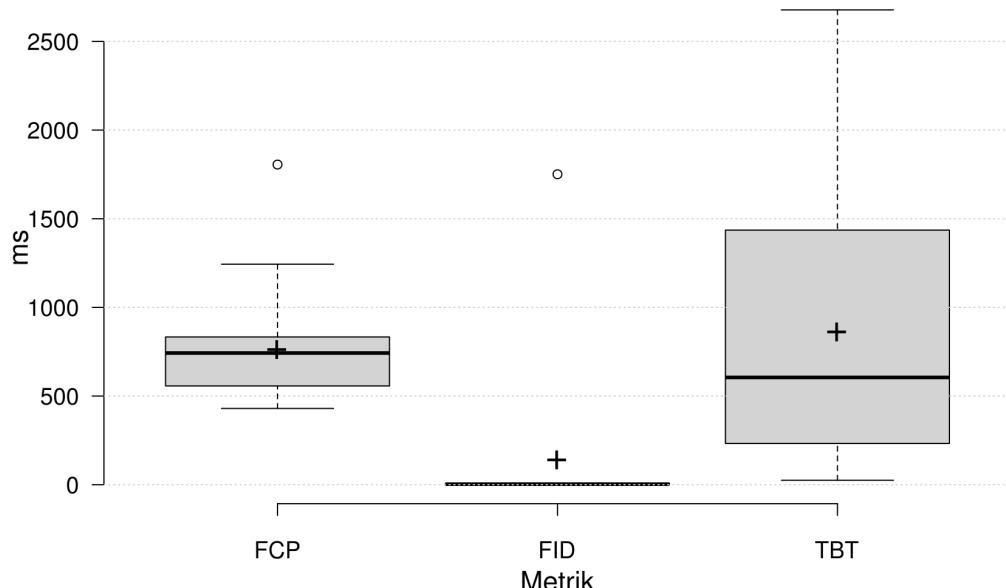
Wie im Abschnitt 2.4.2 beschrieben, werden wahrgenommene Ladezeit, Lade-Reaktionsfähigkeit und Smoothness bewertet. Für die Bestimmung der wahrgenommenen Ladezeit wird der FCP gemessen. Normalerweise wird hierfür der LCP gemessen,

was aber nicht möglich ist, da die Ladezeit des wichtigsten Elements – der deck.gl Karte – nicht gemessen werden kann. Der Grund hierfür ist, dass deck.gl nicht mitteilt, wenn eine Ebene ihren Inhalt darstellt. Die Lade-Reaktionsfähigkeit wird über den FID und die TBT gemessen. Die Messwerte wurden über die – in Webbrowsersn integrierte – PerformanceObserver-API [98] gemessen. Da auch die INP, aufgrund von Beschränkungen durch deck.gl, nicht gemessen werden kann, wird die Smoothness auf der Basis subjektiver Beobachtungen bewertet.

### 6.3.1 Effizienzmesswerte

Über einen Zeitraum von 4 Wochen haben 13 unterschiedliche Personen mindestens einmal mit dem Prototyp interagiert. Hierbei wurden die genannten Messwerte gemessen. In Abbildung 13 werden diese in mehreren Boxplot-Diagrammen dargestellt. Es wurden insgesamt 24 FCP-, 13 FID- und 23 TBT-Werte gemessen. Die Differenz der Menge ist darauf zurückzuführen, dass der FID nicht in allen Webbrowsersn gemessen werden kann. Außerdem werden FCP und TBT nicht zum selben Zeitpunkt bestimmt, sodass der Prototyp geschlossen werden kann, nachdem der FCP, aber bevor der TBT gemessen wurde. Wie man sieht, liegt der Mittelwert der FCP-Werte bei 760 ms und der Median bei 743 ms. Außerdem gibt es einen Ausreißer der bei 1806 ms liegt. Die FID-Werte liegen alle – bis auf einen Ausreißer von 1751 ms – bei unter 10 ms. Durch den Ausreißer gibt es eine große Differenz zwischen Mittelwert und Median. So ist der Mittelwert 138 ms während der Median 3 ms ist. Die TBT-Werte liegen breit verteilt zwischen 25 ms und 2678 ms. Der Mittelwert ist 860 ms und der Median ist 605 ms.

**Abbildung 13: Effizienzmesswerte**



### 6.3.2 Bewertung der Effizienz

Solange der FCP unter 1800 ms liegt, kann dieser als gut bewertet werden [60]. Die gemessenen FCP-Werte liegen bis auf den Ausreißer unter diesem Richtwert. Für eine gründliche Bewertung der wahrgenommenen Ladezeit reicht dieser Messwert nicht aus, da die Ladezeit der Kartenansicht nicht gemessen wird. Trotzdem kann zumindest die Ladezeit der GUI-Elemente, die nicht mit deck.gl in Verbindung stehen, als gut bewertet werden.

Bis auf einen Ausreißer liegen alle gemessenen FID-Werte unter dem Richtwert von 100 ms [61], was bedeutet, dass die erste Interaktion in der Regel eine geringe Latenz hat. Gleichzeitig liegt die TBT über dem Maximalwert von 600 ms [62], was bedeutet, dass der Hauptthread lange blockiert wird, sodass Interaktionen während des Ladens der Kartenansicht eine hohe Latenz haben. Diese Messwerte erscheinen auf den ersten Blick widersprüchlich, lassen sich aber dadurch erklären, dass die Karte während des Ladevorgangs keine Daten anzeigt, wodurch es keinen Anreiz gibt mit der Anwendung zu interagieren. Erst wenn die Kartenansicht vollständig geladen wurde und der Hauptthread nicht mehr blockiert wird, wird der Inhalt angezeigt. So interagieren die meisten Nutzer erst mit der Anwendung, wenn der Hauptthread nicht mehr blockiert wird, wodurch die Latenz der Interaktionen gering ist. Die Lade-Reaktionsfähigkeit ist somit schlecht, was sich allerdings nicht negativ für den Nutzer auswirkt, da dieser sowieso erst mit der Anwendung interagiert, nachdem die Kartenansicht vollständig geladen wurde.

Während der Usability Tests konnten keine Ruckler beobachtet werden und auch außerhalb der Usability Tests sind bei der Nutzung des Prototyps keine Ruckler aufgefallen. Insgesamt erscheint die Bedienung der Kartenansicht sowohl am Desktop als auch an Mobilgeräten flüssig. Daher kann die Smoothness als gut bewertet werden. Es gilt natürlich zu beachten, dass diese Bewertung nicht auf objektiven Messwerten, sondern nur auf subjektiven Beobachtungen basiert.

## 7 Diskussion

Es sollte die Frage beantwortet werden, wie eine effiziente und benutzerfreundliche Steuerung und Verwaltung von Servicerobotern implementiert werden kann. Hierfür wurde ein Prototyp entwickelt, der diese Anforderungen erfüllt. Außerdem sollte zusätzlich eine Methode zur einfachen Generierung von 3D-Modellen identifiziert und eingesetzt werden. Hierfür wurde eine passende Methode gefunden und erfolgreich eingesetzt. Im Folgenden werden diese Ergebnisse genauer zusammengefasst und interpretiert.

### 7.1 Erfüllung der Anforderungen

Wie im Abschnitt 6.1 beschrieben, konnten die funktionalen Anforderungen erfüllt werden, wodurch auch der entsprechende Teil der Forschungsfrage, nämlich die Implementierung einer Steuerung und Verwaltung von Servicerobotern, erfüllt wurde. Wie im Abschnitt 1.1 erwähnt, sollen die Roboter zunächst für kürzere Botengänge eingesetzt werden, wobei der Einsatz in Gastronomiebetrieben im Raum steht. Die Anforderungen wurden für die Durchführung von Botengängen definiert. Für den Einsatz in der Gastronomie müsste der Prototyp somit um weitere Funktionen, wie eine erweiterte Routenplanung erweitert werden.

Das Einhalten der Usability Entscheidungsregeln nach Nielsen [53] hat maßgeblich dazu beigetragen, dass der Prototyp benutzerfreundlich ist. Hierbei sollte auch beachtet werden, dass der Einsatz von deck.gl eine maßgebliche Rolle in der Einhaltung der Entscheidungsregeln spielt, da das Framework unter anderem die Navigation innerhalb der Karte mitliefert. Diese orientiert sich in der Handhabung an anderen gängigen Kartenanwendungen wie Google Maps. Das Framework trägt so dazu bei, dass die vierte Regel der Usability Entscheidungsregeln [53, Regel 4] eingehalten wird. Wie die Usability Tests gezeigt haben, reicht eine subjektive Einhaltung der Entscheidungsregeln nicht aus, um eine gute Benutzerfreundlichkeit zu garantieren. So sind im Pilottest und in der ersten Usability Testrunde eine Vielzahl an Problemen aufgetreten, obwohl die Entscheidungsregeln weitestgehend eingehalten wurden. Durch die Ergebnisse der Tests und das zusätzlich gesammelte Feedback konnten Änderungen vorgenommen werden, durch die die Benutzerfreundlichkeit verbessert wurde. Basierend auf den Ergebnissen der zweiten Usability Testrunde wird davon ausgegangen, dass die Anforderungen an die Benutzerfreundlichkeit eingehalten werden. Diese Annahme ist zwar schlüssig, aber rückblickend nicht unbedingt ausreichend belegt. So hätten sich hier noch weitere Usability Testrunden angeboten, um zu Prüfen, ob noch weitere Probleme auftreten. Weitere Usability Tests hätten auch mit

anderen Aktivitäten durchgeführt werden können, da es möglich ist, dass die wenigen nicht getesteten Funktionen weitere gravierende Probleme aufweisen. Gleichzeitig muss aber auch immer der Aufwand mit den potenziellen Erkenntnissen abgewogen werden. In dieser Hinsicht ist die Entscheidung, keine weiteren Tests durchzuführen nachvollziehbar.

Die Anforderungen an die Effizienz konnten vor allem durch die Reduktion der Ladezeiten eingehalten werden. So wurden die Ladezeiten durch die Komprimierung der 3D-Modelle zweifach reduziert: Zum einen werden die Modelle durch die kleinere Dateigröße schneller heruntergeladen und zum anderen werden die Modelle durch eine geringere Komplexität schneller gerendert. Die Effizienz wurde anhand der Messwerte FCP, FID und TBT, sowie durch die subjektiv beobachtete Smoothness bewertet. Es muss beachtet werden, dass die Bewertung der Ladezeit durch den FCP nicht die Ladezeit der Kartenansicht abdeckt, da diese durch Beschränkungen in deck.gl nicht gemessen werden kann. Hierdurch sind die Messwerte nicht besonders aussagekräftig. Die subjektiv beobachtete Smoothness dient hier als Ausgleich, damit die Bewertung der Effizienz fundierter ist. Bei der Prüfung der Smoothness sind keine Ruckler aufgefallen, was ein positives Signal ist. Die gemessene TBT ist schlecht, was aber wegen des guten FIDs kein Problem ist. Insgesamt scheint die Effizienz des Prototyps gut zu sein, wobei zu beachten ist, dass diese Aussage auf einer dünnen Datenbasis beruht. Für eine bessere Prüfung der Effizienz müsste deck.gl um ein Event erweitert werden, das ausgelöst wird, wenn eine Ebene erstmals angezeigt wird. Mit diesem Event wäre die Ladezeit der Kartenansicht messbar.

Es wurde vorweg erwartet, dass die Anforderungen an den Prototyp mithilfe von deck.gl erfüllt werden können, da das Framework sowie dessen Möglichkeiten bereits bekannt waren. Trotzdem war vorweg nicht klar, wie bestimmte Funktionalitäten – wie zum Beispiel das Synchronisieren der Roboterdaten und Gebäudemodelle – umgesetzt werden können. So war das Verschieben und Rotieren im Editiermodus als Plan B eingeplant, auf den letztendlich auch zurückgegriffen werden musste. Verschiedene Eigenarten von deck.gl, wie dass jedes 3D-Modell über eine eigene ScenegraphLayer-Instanz dargestellt werden muss, waren zum Teil irritierend, aber gleichzeitig meist auch nachvollziehbar. So muss immer beachtet werden, dass das Framework grundsätzlich für die Visualisierung riesiger Geodatensätze und nicht direkt für die Zwecke des Prototyps ausgelegt ist.

Neben der Entwicklung des Prototyps sollte eine Methode identifiziert und eingesetzt werden, die sich für das einfache Generieren von 3D-Modellen eignet. Es wurden verschiedene Methoden oberflächlich miteinander verglichen, wobei letztendlich das LiDAR-Scannen per iPhone mit der Scaniverse App gewählt wurde. Trotz verschiedener Schwächen, wie dem schlechten Scannen transparenter oder reflektierender Flächen, eignet sich die gewählte Methode gut. Es ist erwähnenswert, dass während der Usability Tests mehrere

Testpersonen meinten, dass die 3D-Modelle kaputt aussehen, da die Qualität an vielen Stellen nicht besonders gut ist. Das Scannen ist mit vergleichsweise wenig Aufwand verbunden, wobei manche Scans mehrmals durchgeführt werden mussten, da die generierten Modelle zu große Mängel aufzeigten. Außerdem erfordert das LiDAR-Scannen kein gesondertes Know-how und ist mit keinen Anschaffungskosten verbunden, solange ein LiDAR fähiges iPhone zur Verfügung steht. Es wurde explizit nur nach einer gut geeigneten Methode gesucht, da das Finden der am besten geeigneten Methode für die Beantwortung der Forschungsfrage unerheblich ist. Die Frage, welche Methode am besten geeignet ist, könnte über einen tieferen Vergleich der vorgestellten Methoden beantwortet werden.

## 7.2 Einsatz der Technologien

Der Prototyp wurde basierend auf dem Framework deck.gl implementiert, das eigentlich für die Visualisierung riesiger Geodatensätze ausgelegt ist. So wird gezeigt, dass das ebenenbasierte Visualisieren von Daten mit deck.gl nach dem PIL-Prinzip nicht nur für Geodaten, sondern auch für raumbezogene Daten, wie die Roboterdaten und die 3D-Modelle geeignet ist. Der umfangreiche Katalog an vordefinierten Ebenen bietet eine ausreichend vielseitige Auswahl an Visualisierungsmöglichkeiten, um die visuellen Anforderungen komplexer Anwendungen zu erfüllen. So konnte die Visualisierung der Daten im Prototyp mithilfe der SimpleMeshLayer, ScenegraphLayer, IconLayer und PathLayer umgesetzt werden. Auch bieten die Ebenen-, Controller- und View-Klassen ausreichend viele Schnittstellen, um komplexe Interaktionsmöglichkeiten umzusetzen. So ließ sich das Verschieben und Rotieren der Modelle im Editiermodus über verschiedene Events der Layer-Klasse implementieren und das Zurücksetzen der Kameraposition konnte über die View-Klasse umgesetzt werden.

Da der Prototyp größtenteils auf deck.gl basiert, bestand vor und während der Entwicklung die Hoffnung, dass die Umsetzung mit diesem Framework möglichst unkompliziert sein würde. Diese Hoffnung wurde größtenteils erfüllt. So ist das PIL-Prinzip im Framework intuitiv und gut umgesetzt, was auch die Entwicklung erleichtert. Außerdem ist die Dokumentation des Frameworks, bis auf wenige Stellen ausführlich und hilfreich. Zusätzlich gibt es eine aktive Community an Maintainern des Frameworks und Entwicklern, die es nutzen, wodurch die Lösungsfindung bei Problemen vereinfacht wird. Da das Framework regelmäßig Updates bekommt und ab der kommenden Version 9.0.0 auf WebGPU basiert [43], ist es außerdem zukunftssicher. Gleichzeitig muss auch erwähnt werden, dass während der Entwicklung verschiedene Bugs aufgefallen sind, für die teilweise keine Alternativlösung gefunden werden konnten. So werden zum Beispiel zwar Positionsänderungen,

aber nicht Rotationsänderungen der Roboter animiert, obwohl das laut der Dokumentation möglich sein sollte. Auch ist die Dokumentation des SnapshotTestRunners nicht ausführlich genug, wodurch für große Teile des Prototyps keine Tests implementiert werden konnten.

Der Einsatz der Technologien Sass, TypeScript, React und Redux hat sich als angemessen erwiesen. So konnten mit Sass übersichtliche und wiederverwendbare Styles definiert werden. Der Einsatz von TypeScript war im Nachhinein betrachtet sogar unerlässlich, da die vom BCB empfangenen Objekte ohne definierte Typen für eine effiziente Entwicklung zu komplex sind. Da deck.gl besonders für den Einsatz mit React geeignet ist, konnten hier Synergien genutzt werden, die sich möglicherweise auch positiv auf die Effizienz des Prototyps auswirken. Die angefragten Roboterdaten werden durch Redux zentral gespeichert und können aus allen React-Komponenten abgerufen werden, was die Entwicklung deutlich vereinfacht hat.

### 7.3 Bewertung der Methodik

Zur Beantwortung der Forschungsfrage wurde nach dem DSR Ansatz nach Hevner [8] gearbeitet. Rückblickend erwies sich die gewählte Vorgehensweise als passend, da die Hauptaspekte des Forschungsansatzes – die Relevanz-, Strenge- und Design-Schleife – gut abgebildet werden konnten. So wurde ein fertiges Artefakt – der Prototyp – entwickelt. Die Relevanz-Schleife wurde dadurch abgebildet, dass die Anforderungen basierend auf der Umgebung definiert wurden und der Prototyp, im Rahmen der Usability und Performance Tests, innerhalb der Umgebung getestet wurde. Die Strenge Schleife wurde dadurch abgebildet, dass das zur Entwicklung und Auswertung des Prototyps relevante Wissen aus der Wissensbasis entnommen wurde und diese durch die Entwicklung und Auswertung des Prototyps auch erweitert wurde. Zuletzt wurde die Design-Schleife durch die iterative Entwicklung und Auswertung des Prototyps nach dem Verfahren des evolutionären Prototypings abgebildet.

## 8 Fazit

Im Rahmen dieser Arbeit wurde die Forschungsfrage, wie eine effiziente und benutzerfreundliche Steuerung und Verwaltung von Servicerobotern implementiert werden kann, beantwortet. Hierfür wurde erfolgreich ein Prototyp implementiert, der iterativ entwickelt und auf die verschiedenen Anforderungen geprüft wurde. Die aus der Zielsetzung, Forschungsfrage und Umgebung herausgearbeiteten Anforderungen wurden weitestgehend erfüllt. So handelt es sich bei dem Prototyp um eine benutzerfreundliche und effiziente Webanwendung.

### 8.1 Nutzung des Prototyps

Der lauffähige Prototyp ist mithilfe der Anleitung – die sich in der Datei Prototyp.txt in den Zusatzdokumenten befindet – erreichbar und nutzbar. Es gilt zu beachten, dass die meisten Verwaltungs- und Steuerungsfunktionen aus Sicherheitsgründen nicht genutzt werden können.

### 8.2 Ausblick

Da die Anforderungen an den Prototyp erfüllt werden konnten, ist der nächste logische Schritt die Implementierung als Produktivsystem. Wie bereits erwähnt, sind hierfür verschiedene Anpassungen im Prototyp und im BCB nötig, die allerdings nicht sonderlich groß ausfallen. So müssen vor allem neue Endpunkte hinzugefügt und die Datenbank erweitert werden, damit die 3D-Modelle gespeichert, verändert und abgerufen werden können. Auch müssen die Standorte und Roboterpfade permanent im BCB abgespeichert werden, damit nicht nur Daten von Stockwerken angefragt werden können, in denen sich Roboter befinden. Zusätzlich müssen die Produktqualitätsmerkmale untersucht werden, die im Rahmen dieser Arbeit vernachlässigt wurden. Hierbei handelt es sich um: Portabilität, Wartbarkeit, Sicherheit, Verlässlichkeit und Kompatibilität. Zuletzt sollte ein Produktivsystem auch den Import von 3D-Modellen aus anderen Quellen ermöglichen. Im gleichen Schritt könnte auch eine automatische Kompression der importierten Modelle implementiert werden.

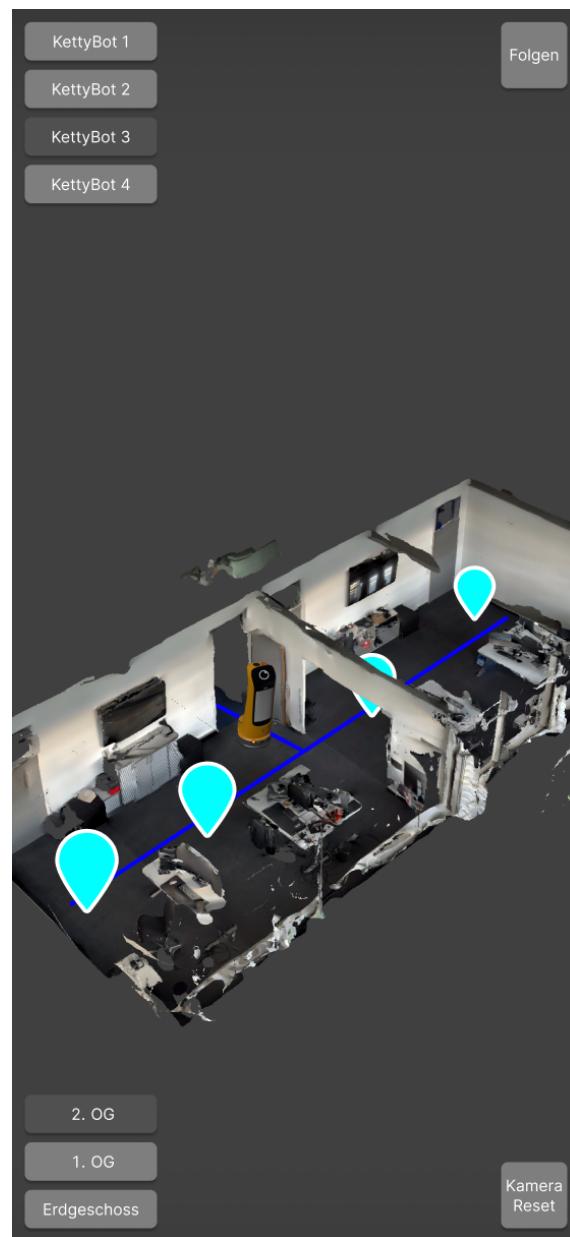
Zusätzlich können auch weitere Aspekte erforscht werden, die im Rahmen dieser Arbeit nur oberflächlich betrachtet wurden. Wie bereits demonstriert, kann deck.gl für mehr als

nur für Geodatenvisualisierungen eingesetzt werden. So könnte genauer erforscht werden für welche weiteren Anwendungsszenarien sich das Framework noch eignet. Wie bereits erwähnt, sollen die Roboter auf ihre Zuverlässigkeit und Navigationsfähigkeit geprüft werden. Hierfür eignet sich eine wissenschaftliche Untersuchung anhand ausgewählter Kriterien. Zuletzt könnte noch untersucht werden, wie der konzipierte Editiermodus zum Verschieben und Rotieren auch für die Nutzung an Smartphones implementiert werden könnte. So ist der Editiermodus im Prototyp nur an Desktop Computern nutzbar, da die Steuerungs- und Umschalttasten benötigt werden.

## Anhang

### Anhang 1: Bilder

Abbildung 14: Mockup der Übersicht



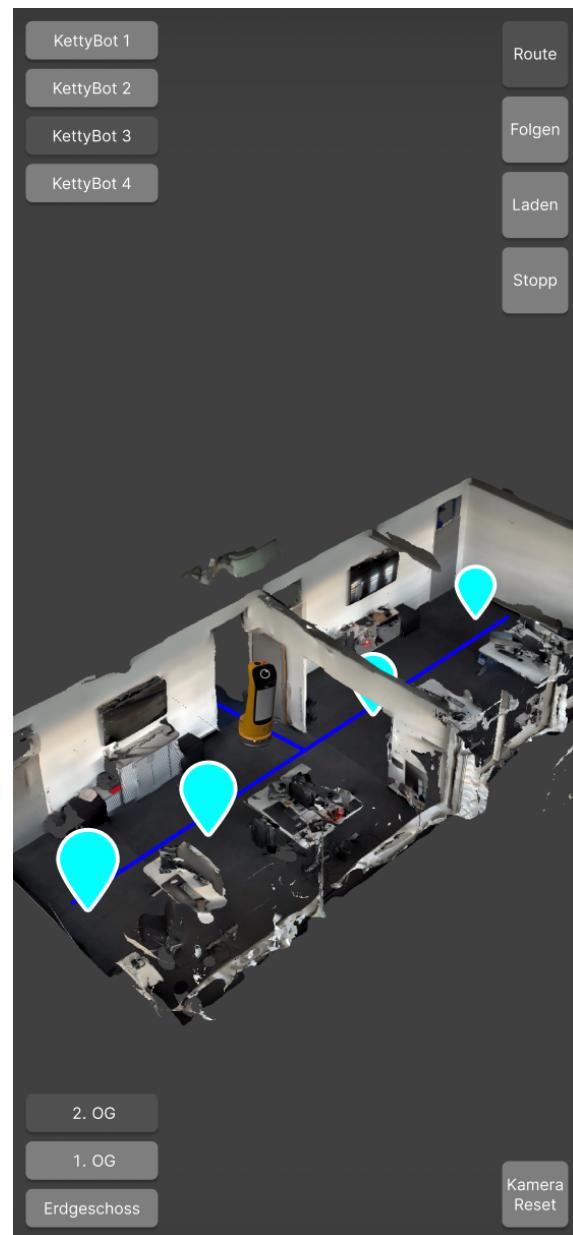
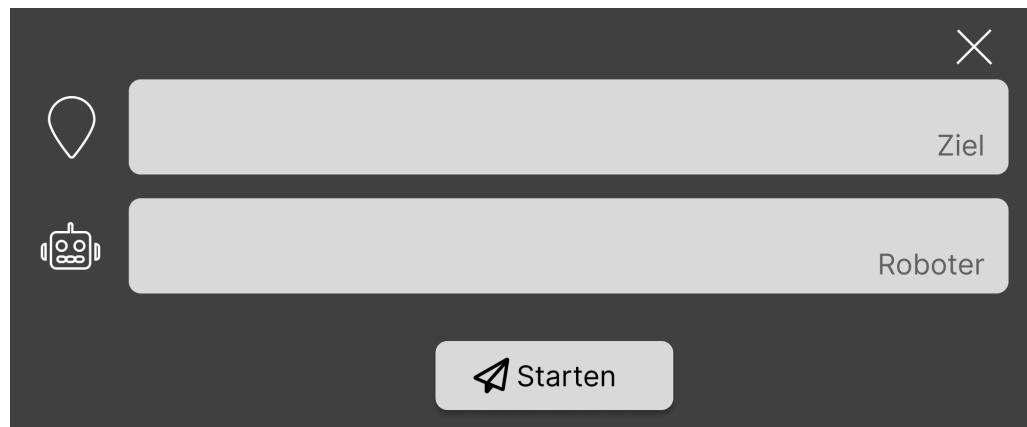
**Abbildung 15: Mockup der Steuerung**

Abbildung 16: Mockup des Routenplanungs-Popup



**Abbildung 17: Mockup der Verwaltung**



## Anhang 2: Tabellen

**Tabelle 5: Beschreibung der Probleme in erster Usability Testrunde**

Problem Nr.	Beschreibung
1	Der Ladestations-Button wurde mit dem Akkustand verwechselt.
2	Die Icons der Buttons sind nicht selbsterklärend. Ohne Anklicken der Buttons kennt man die Funktionen nicht.
3	Die Anordnung der Ergebnisse im Input Dropdown suggeriert, dass man nur nach Nummern und nicht nach Namen suchen kann.
4	Die Tooltips auf der Karte sind durch die Farbwahl schlecht lesbar.
5	Die Fehlermeldungen erklären das Problem nicht.
6	Es wurde versucht, die Modelle in der Übersicht zu verschieben, statt in den Editiermodus zu wechseln.
7	Die Erklärung der Steuerung im Editiermodus ist nicht eindeutig genug.
8	Ohne Loslassen der Maus, kann nicht zwischen Verschieben und Rotieren des Modells gewechselt werden.
9	Es wurde erwartet, dass man in der Routenplanung mehrere Ziele einstellen kann.
10	Es wurde erwartet, dass die Ladestation in der Routenplanung eingesetzt werden kann.
11	Der Toast Dialog, der die Steuerung im Editiermodus erklärt, wurde nicht angezeigt.
12	Es gab Unklarheit darüber, wohin das 3D-Modell verschoben werden muss.
13	Die Darstellung der 3D-Modelle sieht kaputt aus.
14	Ein Stockwerk-Button wird durch das Beta Modus Tag von chayns verdeckt.
15	Beim Positionieren des Modells gab es Schwierigkeiten.
16	Der Roboter wurde aufgrund des Icons – das über diesem angezeigt wird – nicht erkannt.
17	Die Transparenz der Lieferauftrag-Fläche macht Texte schlecht lesbar und sieht insgesamt nicht gut aus.
18	Die Robotersteuerungs-Buttons werden nicht angezeigt, wenn die Routenplanung geöffnet ist.
19	Es wurde erwartet, dass man nach dem Schließen des Editiermodus wieder im Nutzermodus landet.
20	Der Button zum Zurücksetzen der Kameraposition wurde im Editiermodus nicht gefunden, da dieser an einer anderen Position als in der Übersicht angezeigt wird.

**Tabelle 6: Beschreibung der Probleme in zweiter Usability Testrunde**

Problem Nr.	Beschreibung
1	Es gab Unklarheit darüber wie man die Karte rotiert.
2	Beim Entfernen einer Roboterauswahl wird in das Stockwerk des Roboters gewechselt, obwohl das nur bei der Auswahl eines Roboters passieren sollte.
3	Es wurde erwartet, dass man den Roboter über das Ladestations-Icon auf der Karte zur Ladestation schicken kann.
4	Die Suchfunktion zum Auswählen eines Standorts wurde nicht gefunden, da das Routenplanungs-Popup nicht geöffnet wurde.
5	Es gab Irritation darüber, dass die Steuerung über einen Tooltip erklärt wird.
6	Der Editiermodus wurde nicht im Nutzermodus gefunden, da das Icon des Buttons falsch verstanden wurde.
7	Der Editiermodus wurde nicht im Admin-Modus gefunden, da das Kontextmenü übersehen wurde.

## Literaturverzeichnis

- [1] Paluch, Stefanie; Wirtz, Jochen; Kunz, Werner H.: Service Robots and the Future of Services, In: *Marketing Weiterdenken*, 2020, ISBN: 978-3-658-31562-7.
- [2] International Federation of Robotics: Service-Roboter-Absatz steigt weltweit um 48 Prozent – Personalmangel treibt die Nachfrage, 2023, Adresse: [https://ifr.org/downloads/press2018/DE-2023-10-11-IFR-Pressemeldung\\_World\\_Robotics\\_Service\\_Robots\\_2023.pdf](https://ifr.org/downloads/press2018/DE-2023-10-11-IFR-Pressemeldung_World_Robotics_Service_Robots_2023.pdf), Zugriff am: 03.01.2024.
- [3] Bill, Marina; Müller, Christopher; Kraus, Werner; Bieller, Susanne: World Robotics Report 2023 – Press Conference, 2023, Adresse: <https://www.youtube.com/watch?v=h-3ndnde8dY>, Zugriff am: 03.01.2024.
- [4] Bundesagentur für Arbeit: Bestand an offenen Arbeitsstellen im Jahresdurchschnitt bis 2024, 2024, Adresse: <https://de.statista.com/statistik/daten/studie/2903/umfrage/jahresdurchschnittswerte-des-bestands-an-offenen-arbeitsstellen>, Zugriff am: 13.03.2024.
- [5] Jansen, Anika; Risius, Paula: Sorgenkind Gastro? Berufswechsel in der Corona-Pandemie, 2022, Adresse: <https://www.iwkoeln.de/studien/anika-jansen-paula-risius-berufswechsel-in-der-corona-pandemie.html>, Zugriff am: 04.03.2024.
- [6] Sprenger, Michaela; Mettler, Tobias: Service Robots, In: *Business & Information Systems Engineering*, 2015.
- [7] Amazon Web Services: Was ist eine Webanwendung?, Adresse: <https://aws.amazon.com/de/what-is/web-application>, Zugriff am: 07.03.2024.
- [8] Hevner, Alan R.; March, Salvatore T.; Park, Jinsoo; Ram, Sudha: Design Science in Information Systems Research, In: *MIS Quarterly*, 2004.
- [9] Crinnion, John: Evolutionary systems development: A practical guide to the use of prototyping within a structured systems methodology, 1992, Pitman Publishing, London, ISBN: 978-0273032601.
- [10] ISO: ISO 8373:2021 – Robotics – Vocabulary, 2021, Adresse: <https://www.iso.org/standard/75539.html>, Zugriff am: 05.01.2024.
- [11] Gonzalez-Aguirre, Juan A.; Osorio-Oliveros, Ricardo; Rodríguez-Hernández, Karen L. et al.: Service Robots - Trends and Technology, In: *Applied Sciences*, 2021.
- [12] Pudu Robotics: Smart Catering, Adresse: <https://www.pudurobotics.com/de/solutions/Intelligente%20Gastronomie>, Zugriff am: 05.01.2024.
- [13] Nature Research Custom: Intelligent robots offer service with a smile, 2022, Adresse: <https://www.nature.com/articles/d42473-022-00395-5>, Zugriff am: 15.01.2024.
- [14] Pudu Robotics: Pudu Robotics to Expand Service Scenarios with Its Proprietary Upgraded PUDU VSLAM+, 2023, Adresse: <https://www.pudurobotics.com/de/news/784>, Zugriff am: 15.01.2024.
- [15] Pudu Robotics: SDK Guidance Document for PuduTech Open Platform for Robotic Services (Nodejs Microservice).

- [16] MDN Web Docs: HTML basics, Adresse: [https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/HTML\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics), Zugriff am: 07.03.2024.
- [17] MDN Web Docs: CSS basics, Adresse: [https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/CSS\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics), Zugriff am: 07.03.2024.
- [18] Sass: Sass Basics, Adresse: <https://sass-lang.com/guide>, Zugriff am: 07.03.2024.
- [19] MDN Web Docs: JavaScript basics, Adresse: [https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/JavaScript\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics), Zugriff am: 07.03.2024.
- [20] Microsoft: The TypeScript Handbook, Adresse: <https://www.typescriptlang.org/docs/handbook/intro.html>, Zugriff am: 07.03.2024.
- [21] MDN Web Docs: Getting started with React, Adresse: [https://developer.mozilla.org/en-US/docs/Learn/Tools\\_and\\_testing/Client-side\\_JavaScript\\_frameworks/React\\_getting\\_started](https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started), Zugriff am: 07.03.2024.
- [22] Abramov, Dan: Motivation | Redux, Adresse: <https://redux.js.org/understanding/thinking-in-redux/motivation>, Zugriff am: 07.03.2024.
- [23] Tobit Software: chayns.site, Adresse: <https://chayns.site>, Zugriff am: 07.03.2024.
- [24] Tobit.Software: create-chayns-app, Version: 1.1.10, Verfügbar unter: <https://github.com/TobitSoftware/create-chayns-app>, Zugriff am: 20.02.2024.
- [25] Tobit.Software: chayns-toolkit, Version: 2.0.8, Verfügbar unter: <https://github.com/TobitSoftware/chayns-toolkit>, Zugriff am: 13.03.2024.
- [26] Tobit.Software: chayns-components, Version: 4.31.4, Verfügbar unter: <https://github.com/TobitSoftware/chayns-components>, Zugriff am: 20.02.2024.
- [27] Tobit.Software: chayns-api, Version: 1.0.20, Verfügbar unter: <https://github.com/TobitSoftware/chayns-api>, Zugriff am: 13.03.2024.
- [28] Parisi, Tony: Programming 3D applications with HTML5 and WebGL - 3D animation and visualization for web pages, 2014, 1st edition, O'Reilly Media, Sebastopol, Kalifornien, ISBN: 978-1449362966.
- [29] Blender Foundation: Blender, Version: 4.0.2, Verfügbar unter: <https://www.blender.org>, Zugriff am: 18.03.2024.
- [30] Aber, James S.; Marzolff, Irene; Ries, Johannes B.: Small-Format Aerial Photography, 2010, Elsevier, Amsterdam, ISBN: 978-0444532602.
- [31] Cohrs, Jonathan; Boonyapanachoti, Mint; Aneja, Sukanya; Köerner, Willa; Kim, Minkyung: An End-to-End Guide to Photogrammetry with Mobile Devices, 2021, Adresse: <https://rd.nytimes.com/projects/an-end-to-end-guide-to-photogrammetry-with-mobile-devices>, Zugriff am: 16.01.2024.
- [32] Cohrs, Jonathan; Boonyapanachoti, Mint; Aneja, Sukanya; Köerner, Willa; Kim, Minkyung: Capturing Images for Photogrammetry, 2021, Adresse: <https://rd.nytimes.com/projects/capturing-images-for-photogrammetry>, Zugriff am: 16.01.2024.
- [33] Cohrs, Jonathan; Boonyapanachoti, Mint; Aneja, Sukanya; Köerner, Willa; Kim, Minkyung: Processing and Aligning 3D Scenes, 2021, Adresse: <https://rd.nytimes.com/projects/processing-and-aligning-3d-scenes>, Zugriff am: 16.01.2024.

- 
- [34] Fenstermaker: What Cell Phones Have LiDAR?, 2022, Adresse: <https://blog.fenstermaker.com/what-cell-phones-have-lidar>, Zugriff am: 16.01.2024.
- [35] Occipital: Canvas, Version: 3.48, Verfügbar unter: <https://apps.apple.com/us/app/canvas-lidar-3d-measurements/id1169235377>, Zugriff am: 18.03.2024.
- [36] Polycam: Polycam 3D Scanner, Version: 3.3.20, Verfügbar unter: <https://apps.apple.com/us/app/polycam-3d-scanner-lidar-360/id1532482376>, Zugriff am: 18.03.2024.
- [37] Toolbox AI: Scaniverse, Version: 2.1.9, Verfügbar unter: <https://apps.apple.com/us/app/scaniverse-3d-scanner/id1541433223>, Zugriff am: 18.03.2024.
- [38] Vidanapathirana, Madhawa; Wu, Qirui; Furukawa, Yasutaka; Chang, Angel X.; Savva, Manolis: Plan2Scene - Converting Floorplans to 3D Scenes, In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, ISBN: 978-1-6654-4509-2.
- [39] Liu, Chenxi; Schwing, Alexander G.; Kundu, Kaustav; Urtasun, Raquel; Fidler, Sanja: Rent3D - Floor-plan priors for monocular layout estimation, In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, ISBN: 978-1-4673-6964-0.
- [40] Surma: WebGPU — All of the cores none of the canvas, 2022, Adresse: <https://surma.dev/things/webgpu>, Zugriff am: 29.02.2024.
- [41] Seguin, Damien: A collection of WebGL and WebGPU frameworks and libraries, Adresse: <https://gist.github.com/dmnsgn/76878ba6903cf15789b712464875cfdc>, Zugriff am: 27.02.2024.
- [42] OpenJS Foundation: Large scale geospatial data visualization, Adresse: <https://vis.gl>, Zugriff am: 27.02.2024.
- [43] Green, Ib; Palmer, Felix; McCurdy, Don: v9 Tracker, 2022, Adresse: <https://github.com/visgl/deck.gl/issues/7457>, Zugriff am: 27.02.2024.
- [44] Wang, Jang: Deck.gl - Large-scale Web-based Visual Analytics Made Easy, 2019.
- [45] OpenJS Foundation: IconLayer | deck.gl, Adresse: <https://deck.gl/docs/api-reference/layers/icon-layer>, Zugriff am: 20.02.2024.
- [46] OpenJS Foundation: IconLayer – Meteorites Landings, Adresse: <https://deck.gl/examples/icon-layer>, Zugriff am: 27.02.2024.
- [47] OpenJS Foundation: SimpleMeshLayer | deck.gl, Adresse: <https://deck.gl/docs/api-reference/mesh-layers/simple-mesh-layer>, Zugriff am: 20.02.2024.
- [48] OpenJS Foundation: ScenegraphLayer | deck.gl, Adresse: <https://deck.gl/docs/api-reference/mesh-layers/scenegraph-layer>, Zugriff am: 20.02.2024.
- [49] OpenJS Foundation: Controller | deck.gl, Adresse: <https://deck.gl/docs/api-reference/core/controller>, Zugriff am: 20.02.2024.
- [50] OpenJS Foundation: Viewport | deck.gl, Adresse: <https://deck.gl/docs/api-reference/core/viewport>, Zugriff am: 27.02.2024.

- 
- [51] Balzert, Helmut: Software-Management Software-Qualitätssicherung Unternehmensmodellierung, 1998, Spektrum Akademischer Verlag, Heidelberg, ISBN: 978-3827400659.
- [52] ISO: ISO/IEC 25010:2011 – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, 2011, Adresse: <https://www.iso.org/standard/35733.html>, Zugriff am: 27.02.2024.
- [53] Nielsen, Jakob: 10 Usability Heuristics for User Interface Design, 1994, Adresse: <https://www.nngroup.com/articles/ten-usability-heuristics>, Zugriff am: 20.02.2024.
- [54] Dumas, Joseph S.; Redish, Janice C.: A practical guide to usability testing, 1999, Revised edition, Intellect Books, Exeter, ISBN: 978-1841500201.
- [55] Budiu, Raluca: Quantitative vs. Qualitative Usability Testing, 2017, Adresse: <https://www.nngroup.com/articles/quant-vs-qual>, Zugriff am: 15.02.2024.
- [56] Moran, Kate: Usability Testing 101, 2019, Adresse: <https://www.nngroup.com/articles/usability-testing-101>, Zugriff am: 15.02.2024.
- [57] Nielsen, Jakob: How Many Test Users in a Usability Study?, 2012, Adresse: <https://www.nngroup.com/articles/how-many-test-users>, Zugriff am: 15.02.2024.
- [58] Walton, Philip: User-centric performance metrics, Adresse: <https://web.dev/articles/user-centric-performance-metrics>, Zugriff am: 07.03.2024.
- [59] Walton, Philip; Pollard, Barry: Largest Contentful Paint (LCP), Adresse: <https://web.dev/articles/lcp>, Zugriff am: 07.03.2024.
- [60] Walton, Philip: First Contentful Paint (FCP), Adresse: <https://web.dev/articles/fcp>, Zugriff am: 07.03.2024.
- [61] Walton, Philip: First Input Delay (FID), Adresse: <https://web.dev/articles/fid>, Zugriff am: 13.03.2024.
- [62] Walton, Philip: Total Blocking Time (TBT), Adresse: <https://web.dev/articles/tbt>, Zugriff am: 07.03.2024.
- [63] Wagner, Jeremy: Interaction to Next Paint (INP), Adresse: <https://web.dev/articles/inp>, Zugriff am: 11.03.2024.
- [64] Internet Assigned Numbers Authority: obj, 2020, Adresse: <https://www.iana.org/assignments/media-types/model/obj>, Zugriff am: 27.02.2024.
- [65] Internet Assigned Numbers Authority: mtl, 2020, Adresse: <https://www.iana.org/assignments/media-types/model/mtl>, Zugriff am: 27.02.2024.
- [66] OpenJS Foundation: OBJLoader | loaders.gl, Adresse: <https://loaders.gl/docs/modules/obj/api-reference/obj-loader>, Zugriff am: 27.02.2024.
- [67] The Khronos® 3D Formats Working Group: glTF™ 2.0 Specification, 2021, Adresse: <https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html>, Zugriff am: 27.02.2024.
- [68] Adhiban, Antony: OptimizeGLB, Version: 0.1.3, Verfügbar unter: <https://optimize-glb.com>, Zugriff am: 28.02.2024.
- [69] Figma: What is Figma?, Adresse: <https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma>, Zugriff am: 20.02.2024.

- [70] Edwards, Luke: clsx, Version: 2.1.0, Verfügbar unter: <https://github.com/lukeed/clsx>, Zugriff am: 13.03.2023.
- [71] Fonticons: SVG Core | Font Awesome Docs, Adresse: <https://fontawesome.com/docs/web/dig-deeper/svg-core>, Zugriff am: 20.02.2024.
- [72] OpenJS Foundation: Performance Optimization | deck.gl, Adresse: <https://deck.gl/docs/developer-guide/performance>, Zugriff am: 20.02.2024.
- [73] Computer Hope: What is Back-face Culling?, 2022, Adresse: <https://www.computerhope.com/jargon/b/backface-culling.htm>, Zugriff am: 20.02.2024.
- [74] Tobit.Software: Web.API | Swagger UI, Adresse: [https://cube.tobit.cloud/pudu-api/v2/\\_docs/index.html](https://cube.tobit.cloud/pudu-api/v2/_docs/index.html), Zugriff am: 18.03.2024.
- [75] OpenJS Foundation: Layer Class | deck.gl, Adresse: <https://deck.gl/docs/api-reference/core/layer>, Zugriff am: 20.02.2024.
- [76] Masinter, Larry: RFC 2397 – The data URL scheme, 1998, Adresse: <https://datatracker.ietf.org/doc/html/rfc2397>, Zugriff am: 28.02.2024.
- [77] OpenJS Foundation: PathLayer | deck.gl, Adresse: <https://deck.gl/docs/api-reference/layers/path-layer>, Zugriff am: 20.02.2024.
- [78] OpenJS Foundation: PathStyleExtension | deck.gl, Adresse: <https://deck.gl/docs/api-reference/extensions/path-style-extension>, Zugriff am: 20.02.2024.
- [79] OpenJS Foundation: Base Maps | deck.gl, Adresse: <https://deck.gl/docs/get-started/using-with-map>, Zugriff am: 13.03.2024.
- [80] OpenJS Foundation: Adding Interactivity | deck.gl, Adresse: <https://deck.gl/docs/developer-guide/interactivity>, Zugriff am: 20.02.2024.
- [81] OpenJS Foundation: Deck | deck.gl, Adresse: <https://deck.gl/docs/api-reference/core/deck>, Zugriff am: 20.02.2024.
- [82] OpenJS Foundation: FlyToInterpolator | deck.gl, Adresse: <https://deck.gl/docs/api-reference/core/fly-to-interpolator>, Zugriff am: 20.02.2024.
- [83] OpenJS Foundation: Animations and Transitions | deck.gl, Adresse: <https://deck.gl/docs/developer-guide/animations-and-transitions>, Zugriff am: 20.02.2024.
- [84] Tobit.Software: PersonFinder | chayns-components, Adresse: <https://github.com/TobitSoftware/chayns-components/blob/master/docs/components/person-finder.md>, Zugriff am: 13.03.2024.
- [85] Boduch, Adam; Derks, Roy: React and React Native - A complete hands-on guide to modern web and mobile development with React.js, 2020, Third edition, Packt Publishing, Birmingham, ISBN: 978-1839211140.
- [86] Meta Open Source: jest, Version: 29.7.0, Verfügbar unter: <https://jestjs.io/>, Zugriff am: 13.03.2024.
- [87] OpenJS Foundation: SnapshotTestRunner | deck.gl, Adresse: <https://deck.gl/docs/api-reference/test-utils/snapshot-test-runner>, Zugriff am: 07.03.2024.
- [88] Dodds, Kent C.: Testing Implementation Details, 2020, Adresse: <https://kentcdodds.com/blog/testing-implementation-details>, Zugriff am: 07.03.2024.

- [89] Erikson, Mark: The Evolution of Redux Testing Approaches, 2021, Adresse: <https://blog.isquaredsoftware.com/2021/06/the-evolution-of-redux-testing-approaches>, Zugriff am: 07.03.2024.
- [90] GitHub: Informationen zu GitHub Pages | GitHub-Dokumentation, Adresse: <https://docs.github.com/de/pages/getting-started-with-github-pages/about-github-pages>, Zugriff am: 20.02.2024.
- [91] GitHub: Grundlegendes zu GitHub Actions | GitHub-Dokumentation, Adresse: <https://docs.github.com/de/actions/learn-github-actions/understanding-github-actions>, Zugriff am: 20.02.2024.
- [92] Ueda, Shohei: actions-gh-pages, Version: 3.9.3, Verfügbar unter: <https://github.com/peaceiris/actions-gh-pages>, Zugriff am: 13.03.2024.
- [93] Deveria, Alexis: Can I use webp?, Adresse: <https://caniuse.com/?search=webp>, Zugriff am: 20.02.2024.
- [94] McCurdy, Don: glTF-Transform, Version: 3.10.0, Verfügbar unter: <https://github.com/donmccurdy/glTF-Transform>, Zugriff am: 13.03.2024.
- [95] Pernice, Kara: How to Maximize Insights in User Testing – Stepped User Tasks, 2020, Adresse: <https://www.nngroup.com/articles/user-testing-stepped-tasks>, Zugriff am: 15.02.2024.
- [96] Nielsen, Jakob: Thinking Aloud – The #1 Usability Tool, 2012, Adresse: <https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool>, Zugriff am: 15.02.2024.
- [97] Schade, Amy: Pilot Testing – Getting It Right (Before) the First Time, 2015, Adresse: <https://www.nngroup.com/articles/pilot-testing>, Zugriff am: 15.02.2024.
- [98] MDN Web Docs: PerformanceObserver, Adresse: <https://developer.mozilla.org/en-US/docs/Web/API/PerformanceObserver>, Zugriff am: 13.03.2024.

---

## **Ehrenwörtliche Erklärung**

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Ich erkläre mich damit einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Coesfeld, 19.3.2024

---

(Ort, Datum)

---

(Eigenhändige Unterschrift)