



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: 08. April 2024

Inhaltsverzeichnis

Abbildungsverzeichnis	V
Tabellenverzeichnis	VI
Abkürzungsverzeichnis	VII
Glossar	VIII
1 Einleitung	1
1.1 Hintergrund und Motivation	1
1.2 Zielsetzung und Forschungsfrage	2
1.3 Methodik	2
1.3.1 Design Science Research nach Hevner	2
1.3.2 Design Science Research im Rahmen der Arbeit	3
2 Grundlagen	5
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	8
2.2.1 Technologien und Softwarebibliotheken	8
2.2.2 chayns	9
2.3 3D Modelle	9
2.3.1 Generierung	10
2.3.1.1 Fotogrammetrie	10
2.3.1.2 LiDAR Scanning	10
2.3.1.3 KI gestützte Methoden	11
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	13
2.4.1 Benutzerfreundlichkeit	14
2.4.1.1 Usability Heuristics	14
2.4.1.2 Usability Tests	15
2.4.2 Performance	15

3 Anforderungen an den Prototyp	17
3.1 Anforderungen an Modellerzeugung	17
3.2 Funktionale Anforderungen	17
3.2.1 Übersicht	17
3.2.2 Steuerung	18
3.2.3 Verwaltung	18
3.2.4 Benutzer	18
3.3 Nicht-funktionale Anforderungen	19
4 Technische Herausforderungen	20
4.1 3D Modelle von Gebäuden	20
4.2 3D Visualisierung im Web	20
4.2.1 deck.gl	20
4.2.2 Dateiformat der 3D-Modelle	20
4.3 Synchronisierung des Gebäudemodells und der Roboterdaten	21
5 Umsetzung des Prototyps	23
5.1 Mockup	23
5.2 Implementierung	24
5.2.1 Übersicht	24
5.2.1.1 Gebäudemodelle	24
5.2.1.2 Roboterdaten	26
5.2.1.3 Echtzeit-Aktualisierung	27
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
6 Evaluierung des Prototyps	35
6.1 Funktionale Anforderungen	35
6.2 Benutzerfreundlichkeit	36
6.2.1 Usability Heuristics	36
6.2.2 Usability Tests	38
6.2.2.1 Erste Usability Test Runde	39
6.2.2.2 Zweite Usability Tests Runde	40
6.3 Performance	42

7 Diskussion	44
7.1 Erfüllung der Anforderungen	44
7.2 Einsatz der Technologien	46
7.3 Bewertung der Methodik	47
8 Fazit	48
8.1 Aufgetretene Probleme	48
8.2 Ausblick	49
Anhang	50
Literaturverzeichnis	56

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	14
Abbildung 5:	Raummodell ohne und mit Backface Culling	25
Abbildung 6:	Übersicht	27
Abbildung 8:	Steuerung und Routenplanung	29
Abbildung 9:	Verwaltung	31
Abbildung 10:	Editiermodus	33
Abbildung 11:	Bestätigungsdialog	37

Tabellenverzeichnis

Tabelle 1: Gefundene Probleme in erster Usability Test Runde	39
Tabelle 2: Bewertung der durchgeführten Aktionen in erster Usability Test Runde	40
Tabelle 3: Gefundene Probleme in zweiter Usability Test Runde	41
Tabelle 4: Bewertung der durchgeführten Aktionen in zweiter Usability Test Runde	42
Tabelle 5: Beschreibung der Probleme in erster Usability Test Runde	54
Tabelle 6: Beschreibung der Probleme in zweiter Usability Test Runde	55

Abkürzungsverzeichnis

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

Glossar

Base64 Base64 ist ein Kodierungsverfahren, das Binärdaten in eine Textdarstellung umwandelt, indem 64 verschiedene ASCII-Zeichen verwendet werden, wodurch die Daten für die Übertragung über textbasierte Protokolle geeignet gemacht werden.
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 und durch Zustandslosigkeit, verschiedene Methoden (GET, POST), Statuscodes und die Verwendung von URLs gekennzeichnet ist. 6

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

MQTT Message Queuing Telemetry Transport ist ein leichtgewichtiges Netzwerkprotokoll für die effiziente Nachrichtenübertragung zwischen Geräten, besonders in Machine-to-Machine (M2M) und Internet of Things (IoT)-Anwendungen. 6, 7

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

WebSocket Ein Kommunikationsprotokoll, das eine bidirektionale, echtzeitfähige Verbindung zwischen einem Webbrower und einem Server ermöglicht, wodurch eine kontinuierliche Datenübertragung für interaktive Webanwendungen in Echtzeit ermöglicht wird. 9, 27, 28, 36

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, Sensoren 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], während der Absatz von Industrierobotern schwächer zunahm [3, S. 9] und bei Servicerobotern 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 40% gestiegen und – trotz des Rückgangs um 18% 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, welches sich auf Daten der BA bezieht, sind während des Pandemiejahrs 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 Kellner Aufgaben 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 aber auch eine Anwendung entstehen, mit der die Roboter vor allem intuitiv gesteuert, aber auch verwaltet werden können. Abhängig der Ergebnisse könnten die Roboter dann möglicherweise in verschiedenen Gastronomiebetrieben eingesetzt werden.

1.2 Zielsetzung und Forschungsfrage

Diese Arbeit setzt an diesem Punkt an und zielt darauf ab, eine Anwendung zu konzipieren und prototypisch zu implementieren, 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 nativen Anwendungen eine plattformunabhängige Nutzung und sofortigen Verfügbarkeit ohne Installation bieten. Für eine möglichst übersichtliche Darstellung der Roboterpositionen sollen 3D-Modelle der Räume genutzt werden. Um eine reibungslose Integration der Roboter in Gastronomiebetriebe zu ermöglichen, sollte die Erstellung der 3D-Modelle mit minimalem Aufwand verbunden und möglichst 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

Die Forschungsfrage wird anhand des Design Science Research (DSR) Ansatzes nach Hevner [7] beantwortet. 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

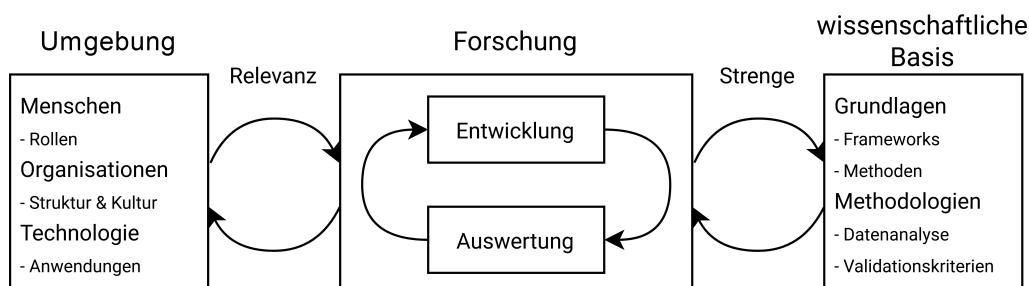
Bei diesem Abschnitt handelt es sich um eine Zusammenfassung des DSR Ansatzes nach Hevner [7, S. 79-81]. Hevnors Ansatz setzt sich aus drei Schleifen zusammen, wobei die

Relevanz- und Strenge-Schleifen nur unterstützende Funktionen für die eigentliche Entwicklung - die Design-Schleife - bieten:

- Die Relevanz-Schleife ergibt sich daraus, dass sich die Anforderungen an das Artefakt aus der Umgebung - für die es entwickelt wird - ergeben und das entwickelte Artefakt daraufhin in der Umgebung getestet wird. Die Umgebung ergibt sich hierbei aus den Menschen und Organisationen, die das Artefakt nutzen sollen und den Technologien, die im Artefakt eingesetzt werden sollen.
- Die Strenge-Schleife ergibt sich daraus, dass während der Entwicklung und Auswertung des Artefakts auf die Wissensbasis zurückgegriffen wird und diese durch die Entwicklung und Evaluierung erweitert wird. Die Wissensbasis besteht hierbei zum einen aus dem technischen Wissen, dass während der Entwicklung relevant ist und Wissen über Methodologien, die während der Auswertung relevant sind.
- In der Design-Schleife 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 wird dieser Prozess grafisch dargestellt. Man sieht 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. [7, S. 80]

1.3.2 Design Science Research im Rahmen der Arbeit

Die Prozesse des DSR Ansatzes nach Hevner werden in dieser Arbeit durch verschiedene wissenschaftliche Methoden abgebildet. Im ersten Schritt wird eine systematische Literaturrecherche durchgeführt, mit der das grundlegende Verständnis über die Umgebung geschaffen werden soll. Auch soll die Literaturrecherche die bereits vorhandene Wissensbasis erweitern. 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 von auftretenden Problemen aus. Die Anforderungen, die sich aus der Umgebung ergeben werden nach der Wissensfindung über eine kurze Anforderungsanalyse definiert. Diese kann auf die funktionalen und nicht funktionalen 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. Die Implementierung des Prototyps soll nach dem Verfahren des Rapid Prototypings durchgeführt werden. So wechseln sich die Entwicklung und Auswertung der Benutzerfreundlichkeit und Effizienz iterativ ab bis ein Prototyp entstanden ist, der die definierten Anforderungen erfüllt.

2 Grundlagen

Im folgenden 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 in dieser Arbeit eingesetzt werden. Außerdem wird das BCB genauer erläutert.

2.1.1 Definition

In wissenschaftlichen Arbeiten wird mit vielen verschiedenen Definitionen für Serviceroboter gearbeitet. In dieser Arbeit wird die Definition aus der ISO Norm 8373:2021 [8, Kap. 3] verwendet. Nach dieser handelt es sich bei Servicerobotern um Roboter, die im privaten oder professionellen Gebrauch für Menschen nützliche Aufgaben erledigen. Serviceroboter werden hierbei von Industrierobotern und Medizinerobotern abgegrenzt. Die IFR [9] ergänzt die Voraussetzung, dass Serviceroboter voll- oder zumindest teilautonom handeln können. Unter dem professionellen Gebrauch versteht man den kommerziellen Einsatz [10, S. 4], unter anderem im Gesundheitswesen, in der Landwirtschaft oder im Tourismus [10, 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 helfen Serviceroboter den Pflegern beim Heben von Patienten und beim Durchführen von Übungen mit Patientengruppen [1, S. 427]. Aufgaben mit geringer kognitiver und emotionaler Komplexität können hierbei vollautonom und ohne Aufsicht eines Menschen durchgeführt werden [1, S. 429]. Unter solche Aufgaben fällt zum Beispiel Staubsaugen, Rasenmähen oder Gepäcklieferung. Komplexere Aufgaben erfordern die Aufsicht oder Unterstützung von Menschen, was bedeutet, dass diese nur teilautonom ausgeführt werden [1, S. 430-431]. Für den Einsatz von Servicerobotern müssen immer die Vor- und

Nachteile verglichen werden. Roboter, die im Kontakt mit Kunden eingesetzt werden, können 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

Diese Arbeit beschäftigt sich mit Robotern von Pudu. 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 [11]. Damit die Roboter diese Funktionen ausführen können, müssen sie eigenständig durch komplexe, sich ändernde Umgebungen navigieren können. Das eigenständige Navigieren lässt sich in die Teilfunktionen Positionsfindung, Wahrnehmung und Routenplanung aufteilen, wobei die Positionsfindung eine Schlüsselrolle spielt [12]. Zur Positionsfindung erstellen sich die Pudu Roboter mit 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 die Roboter mithilfe einer nach oben gerichteten Kamera anhand der Zimmerdecke orientieren.[13] Durch weitere Kameras und Sensoren können die Roboter ihre Umgebung wahrnehmen [12].

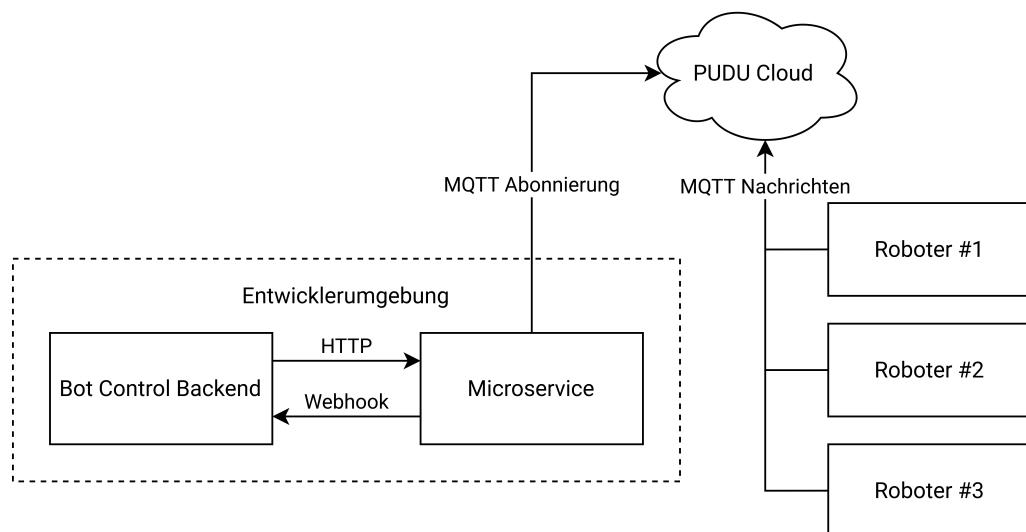
2.1.4 Bot Control Backend

Im Rahmen dieser Arbeit wird das bereits existierende BCB als Schnittstelle zwischen Robotern und Prototyp genutzt, über die die Roboter zum Beispiel beauftragt werden können. Die folgenden Informationen zum Service Framework der Roboter stammen aus dem SDK Guidance Document [14], das der Tobit Laboratories AG 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-Tops an. Wie Befehle versendet werden können, wird in dem SDK Guidance

Document nicht genauer beschrieben. Die Roboter können auch unaufgefordert Ereignisse an das BCB kommunizieren. Hierfür muss das BCB eine Adresse für einzelne Ereignistypen im Microservice als Webhook registrieren. Der Microservice abonniert wiederum die entsprechende MQTT-Topic. Tritt das Ereignis auf, schickt der Roboter eine Nachricht an die PUDU Cloud, die diese wiederum an den Microservice weiterleitet, da dieser die MQTT-Topic abonniert hat. Der Microservice schickt daraufhin eine HTTP-Anfrage an die registrierte Adresse im BCB.

Abbildung 2: Kommunikation zwischen Bot Control Backend und Robotern



Quelle: In Anlehnung an Pudu [14, S. 4]

Das BCB fungiert nicht nur als Schnittstelle zu den Robotern, sondern abstrahiert auch neue Funktionen. So können Roboter mithilfe des BCBs Fahrstuhl fahren und somit Lieferpunkte in anderen Stockwerken erreichen. Ebenso können sie durch das BCB an geschlossenen Tür 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 von wichtigen Standorten wie Lieferpunkten und Ladestationen. Außerdem sind auch die Pfade relevant, an denen sich die Roboter während der Fahrt orientieren. Darüber hinaus sind auch 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 Webservleten gehostet werden und über einen Webbrower aufgerufen werden können. Sie ermöglichen es Benutzern über das Internet auf verschiedene Dienste und Funktionen zuzugreifen, ohne dass eine Installation auf den Geräten erforderlich ist. Da sie auf verschiedenen Betriebssystemen und Gerätetypen verwendet werden können, solange ein kompatibler Webbrower zur Verfügung steht, sind sie außerdem plattformunabhängig. Zusammenfassend ermöglichen Webanwendungen eine breite Zugänglichkeit und viel Flexibilität.

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 strukturieren. CSS wird verwendet, um das Erscheinungsbild einer Webseite zu gestalten, indem es Layout, Formatierung und weitere Eigenschaften von HTML-Elementen definiert. Sassy Cascading Style Sheets (Sass) erweitert CSS durch Variablen, Mixins und weitere Funktionen, wodurch die Wiederverwendbarkeit definierter Styles verbessert und die Entwicklung vereinfacht wird. JavaScript ist eine dynamisch typisierte Skriptsprache, die durch Webbrower ausgeführt wird und die Interaktivität von Webseiten ermöglicht. Die Sprache wird durch TypeScript, um die statische Typisierung erweitert, wodurch die Entwicklung einer robusten und fehlerfreien Anwendung erleichtert wird.

React und React-Redux sind JavaScript-Bibliotheken, die die Entwicklung komplexerer Webanwendungen vereinfachen. 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 über JavaScript XML (JSX) Dateien entwickelt, in denen HTML Elemente im JavaScript-Code integriert werden können. Diese JSX Dateien müssen zu gültigen JavaScript-Dateien kompiliert werden, die von Webbrowsern ausgelesen werden können. React Redux baut auf React auf und dient zur Verwaltung des Anwendungsstatus sowie zur Datenflusssteuerung. Es folgt dem Konzept des zentralisierten Zustandsmanagements, wodurch der Datenfluss komplexer Anwendungen vereinfacht wird.

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 Webseiten-Baukasten, mit dem sich Nutzer eine Webpräsenz erstellen können. Auf den Webseiten können unter anderem vordefinierte chayns Anwendungen, wie ein eShop oder ein Bundesliga-Tippspiel, aber auch eigenentwickelte Webanwendungen eingebunden werden. Tobit bietet verschiedene Softwarebibliotheken, die die Entwicklung eigener chayns Anwendungen erleichtern. So gibt es den Shell Befehl `create-chayns-app` mit dem chayns basierte React Projekte aufgesetzt werden können; das Node Package Manager (npm) Paket `chayns-components`, das React-Komponenten für verschiedene Graphical User Interface (GUI) Elemente zur Verfügung stellt; und das npm Paket `chayns-api`, 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 Clients versenden. 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.

2.3 3D Modelle

Dieser Abschnitt bietet eine kurze Einführung zum Thema 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 manipulieren. Die Kamera, Perspektive und Projektion legen den Standpunkt des Betrachters und die Darstellung des Modells fest.[15, S. 8-16]

2.3.1 Generierung

Neben der manuellen Modellierung von 3D-Modellen mithilfe von Modellierungssoftware wie Blender 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 [16, S. 19]. Inzwischen erfordert die Fotogrammetrie nicht mehr den Einsatz teurer Kameras, da die Kameras moderner Mobilgeräte eine ausreichende Bildqualität bieten [17]. Bei der Planung und Durchführung der Bildaufnahmen müssen verschiedene Aspekte beachtet werden. Innerhalb der aufzunehmenden Szene sollte es eine gleichmäßige Belichtung, möglichst keine reflektierenden 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 Inhalt aufeinanderfolgender Bilder stets überschneiden.[18] Nach der Durchführung der Aufnahmen erfolgt die Verarbeitung mithilfe spezialisierter Software, deren Bedienung komplex sein kann. Für eine reibungslose Verarbeitung sind eine leistungsstarke Grafikkarte und ausreichend Speicherplatz unerlässlich.[19]

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. Seit 2020 werden LiDAR-Scanner in neuere iOS-Geräte von Apple integriert, wodurch sich eine verbesserte Bildqualität erhofft wird [20]. Als Nebeneffekt wurden verschiedene Apps entwickelt, die den LiDAR-Scanner nutzen, um 3D-Modelle zu erstellen, wie zum Beispiel Canvas [21], Polycam [22] und Scaniverse [23]. Diese Apps versprechen eine schnelle und einfache Erzeugung von 3D-Modellen.

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 Modelle ist Plan2Scene. Es benötigt einen Grundriss des Gebäudes, sowie Bilder, die den einzelnen Räumen zugeordnet sind als Eingabe. Basierend auf dem Grundriss generiert das KI-Modell ein 3D-Modell mit Möbeln und monotone Texturen. Die Texturen werden hierbei basierend auf den Bildern der Räume generiert.[24, S. 10733] Das Rent3D Modell funktioniert ähnlich, nutzt aber die Bildaufnahmen direkt als Textur, statt sie aus den Bildaufnahmen zu generieren [25, 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

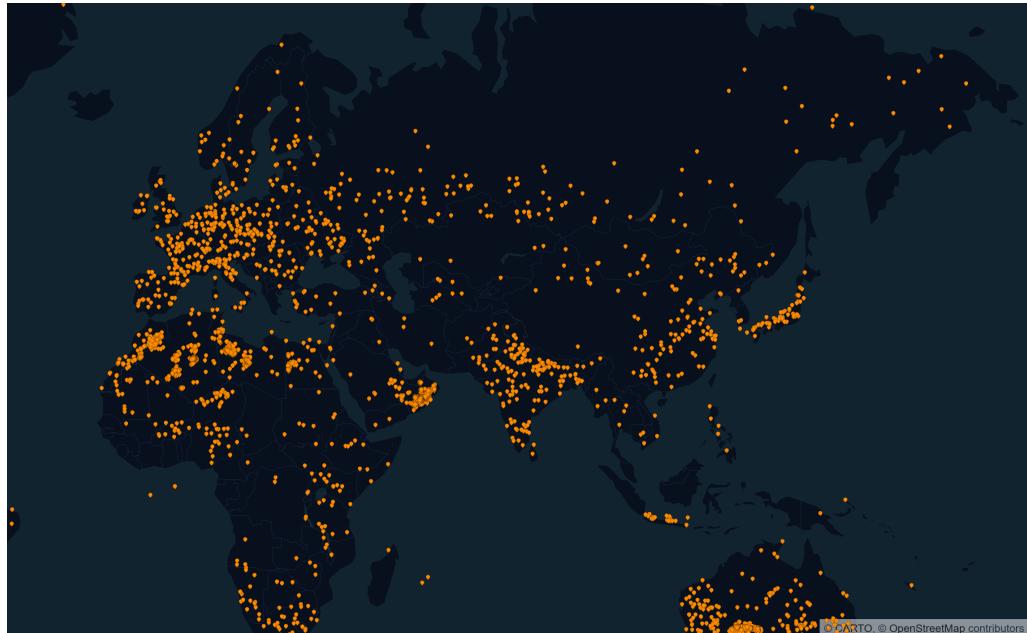
WebGL wurde 2011 von der Khronos Group entwickelt und ist eine JavaScript-Application Programming Interface (API) mit der 3D-Grafiken im Webbrowser ohne den Einsatz zusätzlicher Plugins dargestellt werden können. Die 3D-Grafiken können hierbei hardwarebeschleunigt angezeigt werden, also über den Einsatz spezialisierter Hardware wie einer Graphics Processing Unit (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 sowohl an Desktop- als auch an Mobilgeräten nutzbar.[15, S. 17-19] WebGPU bietet wie WebGL das hardwarebeschleunigte Anzeigen von 3D-Grafiken und darüber hinaus eine verbesserte Leistung sowie erweiterte Funktionen [26]. Entwicklern steht eine Vielzahl an Frameworks zur Verfügung die auf WebGL oder WebGPU basieren [27] und die Entwicklung im Vergleich zu diesen vereinfachen und beschleunigen.

2.3.2.2 deck.gl

Das Framework deck.gl wurde 2016 von Uber als Open Source Projekt veröffentlicht [28]. Das Framework basiert auf WebGL, wobei ab der kommenden Version 9.0.0 stattdessen WebGPU genutzt werden soll [29]. 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 - wie Kreise, Rechtecke und Linien, aber auch komplexe teilweise auch dreidimensionale Elemente - nutzen, um Datenpunkte darzustellen. Die Elemente werden in der Ebene auf Basis der Datenpunkte und deren Attribute 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.[30, S. 2] Das Framework bietet verschiedene vordefinierte Ebenen, wie die IconLayer [31], die Icons als das grundlegende visuelles Element nutzt. Zur Veranschaulichung des PIL-Prinzips zeigt die Abbildung 3 wie die IconLayer - in Kombination mit einer Ebene zur Darstellung der Weltkarte - genutzt wird, um die Positionen aller bekannter Meteoritenlandungen auf der Erde anzuzeigen.

Abbildung 3: IconLayer Beispiel



Quelle: OpenJS Foundation [32]

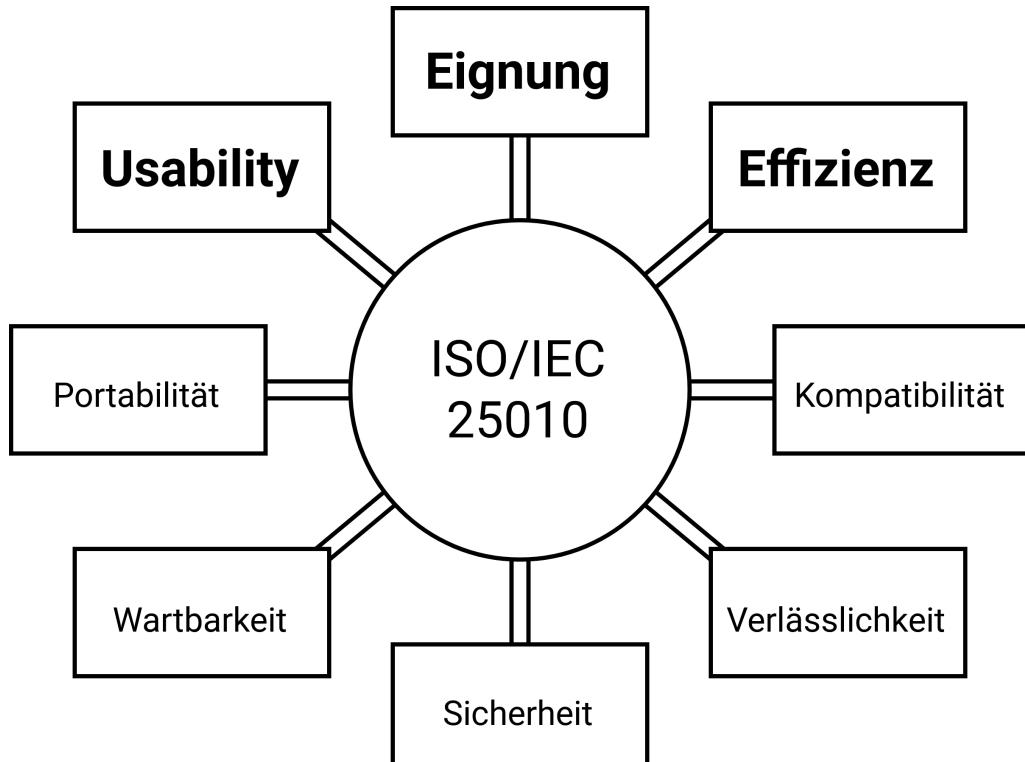
Weitere für diese Arbeit relevante vordefinierte Ebenen sind die SimpleMeshLayer [33] und ScenegraphLayer [34] für das Anzeigen von 3D-Modellen und die PathLayer für das Anzeigen von Pfaden. deck.gl bietet auch die Möglichkeit neue Ebenen zu entwickeln, was für diese Arbeit aber nicht relevant ist, da die Auswahl an vordefinierten Ebenen ausreicht. Weitere wichtige Elemente die deck.gl bietet sind die Controller Klasse [35], mit der die

Navigation auf der Karte konfiguriert werden kann und die Viewport Klasse [36] mit der die Navigation direkt gesteuert 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” [37, S. 257]. So ergibt sich die Softwarequalität aus der Erfüllung der definierten Anforderungen und Erwartungen und zielt darauf ab den Bedürfnissen der Benutzer gerecht zu werden. Diese Anforderungen können in die acht Produktqualitätsmerkmale nach dem ISO-Standard ISO/IEC 25010 eingeteilt werden [38]. In Abbildung 4 werden diese Merkmale aufgelistet. Die Merkmale Effizienz und Benutzerfreundlichkeit haben eine besondere Relevanz für diese Arbeit, 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. Diese drei Merkmale sind in der Abbildung entsprechend gekennzeichnet. Im Rahmen dieser Arbeit wird ein Prototyp entwickelt, der die Ansprüche an ein Produktivsystem nicht erfüllen muss. Aus diesem Grund werden die restlichen Produktqualitätsmerkmale nicht weiter beachtet. Im Folgenden werden die Merkmale Benutzerfreundlichkeit und Effizienz genauer erläutert.

Abbildung 4: Qualitätsmerkmale



Quelle: Eigene Darstellung

2.4.1 Benutzerfreundlichkeit

Die Benutzerfreundlichkeit wird nach ISO 25010 in sechs Kriterien aufgeteilt: angemessene Erkennbarkeit, Erlernbarkeit, Bedienbarkeit, Toleranz gegenüber Anwenderfehlern, Ästhetik der Benutzeroberfläche und Barrierefreiheit [38].

2.4.1.1 Usability Heuristics

Die zehn Usability Heuristics von Nielsen sind grundlegende Richtlinien zur Bewertung der Benutzerfreundlichkeit [39] und decken die ISO 25010 Kriterien weitestgehend ab. Während der Entwicklung können diese Richtlinien als Orientierungshilfe dienen, um die Benutzerfreundlichkeit zu verbessern. Zum Beispiel gibt es die folgenden Richtlinien:

- Die zweite Regel besagt, dass das Design die Sprache der Benutzer sprechen sollte, indem vertraute Wörter, Phrasen und Konzepte, statt interne Fachbegriffe verwendet werden [39, Regel 2].

- Die vierte Regel besagt, dass das Design konsistent sein und etablierten Standards folgen sollte, sodass Benutzer nicht darüber nachdenken müssen, ob verschiedene Wörter, Situationen oder Aktionen dasselbe bedeuten [39, Regel 4].

Beide Richtlinien helfen dabei das ISO 25010 Kriterium der Erlernbarkeit einzuhalten. Eine subjektive Bewertung der Einhaltung dieser Richtlinien durch die Entwickler ist unzureichend, um die Benutzerfreundlichkeit zu bewerten. Stattdessen eignen sich hier Expertenreviews oder Usability Tests 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.[40, 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.[41]. Qualitative Tests erfordern einen Moderator, der die Testpersonen durch den Testprozess leitet. Gegebenenfalls gibt es auch weitere Beobachter, die nicht mit den Testpersonen interagieren.[42] 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.[43]

2.4.2 Performance

Die Performance 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 Performance einer Webanwendung oder Webseite gibt es verschiedene Merkmale. Im Rahmen dieser Arbeit werden der Perceived Load Speed (PLS), die Load Responsiveness (LR) und die Smoothness genauer betrachtet.

Unter dem PLS versteht man wie schnell alle visuellen Elemente angezeigt werden. Er lässt sich mit dem Largest Contentful Paint (LCP) - der Zeit bis das größte oder wichtigste Element angezeigt wird - messen. Auch kann der First Contentful Paint (FCP) gemessen werden, mit dem gemessen wird, wann das erste Element angezeigt wird.

Unter der LR versteht man wie schnell der JavaScript Code geladen wird, damit flüssig auf Eingaben des Nutzers reagiert werden kann. Gemessen wird diese mithilfe des First Input Delay (FID) Messwerts. Mit diesem wird die Verzögerung gemessen, die bei der ersten Eingabe des Nutzers auftritt. Auch kann die Total Blocking Time (TBT) gemessen werden, die die Zeit bestimmt, in der der Hauptthread so stark beschäftigt ist, dass die Eingabe-Reaktionsfähigkeit leidet.

Mit der Smoothness ist die Geschmeidigkeit gemeint, also ob Übergänge und Animationen mit einer konsistenten Bildrate gerendert werden und flüssig aussehen.

3 Anforderungen an den Prototyp

In diesem Kapitel werden die Anforderungen an den Prototyp 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. Hierfür soll eine Methode gefunden werden, die diese Anforderung erfüllt. 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 in dieser Arbeit nur herausgearbeitet werden, dass es eine passende Methode gibt, nicht welche am besten für den Anwendungsfall geeignet ist. Trotzdem soll die Methode möglichst geringe Kosten verursachen. Auch sollte die Methode 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.

3.2 Funktionale Anforderungen

Wie bereits 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. Insbesondere soll die Anwendung sowohl auf Smartphones als auch auf Desktop Computern genutzt werden können, weshalb der Prototyp außerdem responsiv sein soll.

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 Darstellung sollen die Roboter dann so wie in der Realität fahren. Auch soll

ersichtlich sein, ob und womit die Roboter beschäftigt sind. Falls der 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. Auch haben die Roboter festgelegte Pfade, an denen sie sich beim Fahren orientieren. Diese Pfade sollen auch angezeigt werden.

3.2.2 Steuerung

In 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. Das Ändern der verschiedenen Roboter-Einstellungen soll möglich sein. So soll beispielsweise die eingestellte Ladestation geändert werden können.

Es muss außerdem möglich sein die 3D-Gebäudemodelle zu importieren und diese mit den Roboterdaten zu synchronisieren. So muss 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 können schnell Frust verursachen. Die Reduktion der Ladezeiten ist deshalb eine zentrale Anforderung an den Prototyp. Normalerweise lassen sich Ladezeiten sowohl auf der Server- als auch auf der Client-Seite verbessern. Der Schwerpunkt dieser Arbeit liegt allerdings im Frontend, weshalb sich Optimierungen auch nur auf die Client-Seite beschränken sollen. Das BCB soll entsprechend, unabhängig davon, ob Optimierungspotenzial existiert, nicht verändert werden. Maßnahmen, die sich hier 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 aufwändigen Darstellung von 3D-Modellen möglichst performant sein. So soll es beispielsweise beim Navigieren keine Ruckler geben. Hierdurch wird auch die Benutzerfreundlichkeit verbessert. Diese Anforderungen lassen sich unter dem Begriff der Performance zusammenfassen. Die Performance soll anhand des PLS, der LR und der Smoothness bewertet werden.

Neben einer guten Performance sollte die Anwendung auch eine gute Benutzerfreundlichkeit aufweisen, 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 mit den gewählten Lösungsansätzen vorgestellt.

4.1 3D Modelle von Gebäuden

TODO: Erklären warum LiDAR Scanning genutzt wird; Erklären warum die Scaniverse App genutzt wird.

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 Framework für die Umsetzung des Prototyps, sowie die Wahl des Dateiformats für die 3D-Modelle erläutert.

4.2.1 deck.gl

Für die Umsetzung des Prototyps wurde deck.gl gewählt. Das Framework wurde 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, wodurch ein größerer Fokus auf die Darstellung Karte gelegt werden kann. Vor der Implementierung des Prototyps konnte bestätigt werden, dass die Anforderungen an die Anwendung mit dem Framework erfüllt 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 [33] und das Einbinden des Graphics Library Transmission Format (glTF) Dateiformats in der ScenegraphLayer [34]. Beide Dateiformate werden in der Scaniverse App als Dateiexport angeboten.

Das OBJ Format besteht aus einer Datei mit der Endung .obj, in der die dreidimensionalen geometrischen Formen kodiert sind [44] und einer Datei mit der Endung .mtl, in der die

optischen Materialeigenschaften und Texturierung kodiert sind [45]. Für die Einbindung des OBJ Dateiformats wird die loaders.gl Programmbibliothek benötigt, die allerdings nur die .obj Datei und nicht die .mtl Datei parsen kann [46]. 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 Variation relevant ist. Diese hat die Endung .glb und besteht nur aus einer Datei, die neben den geometrischen Formen auch die Materialeigenschaften und Texturierung enthält. Das glTF Format bietet eine geringere Dateigröße als vergleichbare Dateiformate.[47, Abschnitt 2] Mithilfe der loaders.gl Programmbibliothek lassen sich 3D-Modelle des Formats ohne großen Aufwand in der ScenegraphLayer von deck.gl einbinden [34]. Da ein 3D-Modell mit einer passenden Texturierung eine bessere Übersichtlichkeit bietet und glTF Dateien eine geringe 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. 3D-Modelle mit dieser Größe können nicht auf Mobilgeräten angezeigt werden. Außerdem beeinflusst die Dateigröße die Ladezeiten 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 manuell komprimiert worden. Insbesondere durch den Einsatz des Web Picture Format (WebP) Bildformats für Texturen werden die Dateien effektiv komprimiert.[48] 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, reicht aber für den Zweck der Anwendung 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 der Datensätze 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 diese  Scanning-Methoden unterscheiden, unterscheiden sich auch diese Ungenauigkeiten.

Da eine automatische Synchronisierung der Datensätze somit ausgeschlossen ist, muss diese manuell durch den Nutzer vorgenommen werden. Deshalb 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 genauer beschrieben.

5 Umsetzung des Prototyps

Im Folgenden wird das Mockup, die Implementierung, sowie die Evaluierung 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 [39] 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 [49] 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 Aussehen zu stark von den Möglichkeiten in deck.gl abhängt, die noch nicht vollständig bekannt waren, als die Mockups entworfen wurden. Stattdessen wurde das Aussehen des Editiermodus während der Implementierung festgelegt.

Die Abbildung 12 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 von ihren Funktionen in den vier Ecken der Anwendung gruppiert.

Die Steuerung erweitert die Übersicht um weitere Funktionen und sieht daher fast identisch zur Übersicht aus. Abbildung 13 im Anhang zeigt das Mockup der Steuerung. Als einzigen Unterschied zur Übersicht sieht man die zusätzlichen Buttons mit denen eine Lieferoute bestimmt, der Roboter zum Aufladen geschickt und der aktuelle Lieferauftrag abgebrochen werden kann. Die Abbildung 14 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 15 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 der einzelnen Roboter, 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 Liste der Roboter als auch in der Liste der Stockwerke gibt es für jedes Stockwerk eine Auflistung der Standorte. In der Liste der Stockwerke gibt es außerdem eine Vorschau der Übersicht und zusätzlich ein Kontextmenü, über das in den Editiermodus des Stockwerks gewechselt werden kann.

5.2 Implementierung

Es wurde ein Frontend entwickelt, dass 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 Komponentenübergreifend abgerufen werden, werden zentral mithilfe von React-Redux gespeichert. Die Grundstruktur des Frontend-Projekts wurde mithilfe des Shell Befehls `npx create-chayns-app` [50] aufgesetzt. Die Kompilierung ist mithilfe des npm-Pakets `chayns-toolkit` [51] konfiguriert. GUI-Elemente wie Buttons und Aufklapper werden durch die Komponentenbibliothek `chayns-components` [52] bereitgestellt. Außerdem werden die npm-Pakete `clsx` [53] und `fortawesome` [54] genutzt. Mithilfe von `clsx` lassen sich HTML-Klassennamen dynamisch anwenden. Das `fortawesome` 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 `deck.gl` Ebenen eingesetzt.

5.2.1.1 Gebäudemodelle

Für den Prototyp sind die Uniform Resource Locator (URL)-Verweise der 3D-Modelle hartkodiert, 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 `chanys.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 [34], was auf die Hauptfunktion von `deck.gl` - die Visualisierung riesiger Geodaten Mengen [30, 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 den Einsatz von über 100 Ebenen gleichzeitig ausgelegt, wobei wahrscheinlich sogar der Einsatz von bis zu 1000 Ebenen ohne große Performance Einbußen möglich ist [55]. Im Prototyp besteht ein Stockwerk aus bis zu sechs 3D-Modellen und somit auch aus bis zu sechs `ScenegraphLayers`. Es ist davon auszugehen, dass nie mehr als 20 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 `ScenegraphLayers` ist das Backface Culling aktiviert. Mithilfe vom Backface Culling werden die von der Kamera weg gerichteten Polygone ausgeblendet [56]. 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 ausgeblendet werden, die dem Nutzer die Sicht in den Raum verdecken. In der Abbildung 5 wird dieser Effekt deutlich. So sieht man auf der linken Seite, dass ohne das Backface Culling nicht von außen in den Raum hineingesehen werden kann, während es auf der rechten Seite mit aktiviertem Backface Culling möglich ist.

Abbildung 5: Raummodell ohne und mit Backface Culling



Quelle: Eigene Darstellung

5.2.1.2 Roboterdaten

Die Roboterdaten werden über verschiedene Endpunkte des BCBs abgerufen. So gibt es einen Endpunkt für die Bezeichnungen der verschiedenen Standorte und einen Endpunkt für den Roboterstatus, Lieferauftrag und die Roboterposition [57]. Für die Positionen aller Standorte und der Roboterpfade gibt es keinen Endpunkt. Stattdessen können die Standorte und Pfade nur von Stockwerken angefragt werden, in denen sich Roboter zu dem Zeitpunkt 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 [33] angezeigt. Mit der ScenegraphLayer gibt es das Problem, dass die Positionen 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 transition Property in allen Ebenen möglich sein [58]. Deshalb handelt es sich bei dem Problem mit der ScenegraphLayer vermutlich um einen Bug in deck.gl. Wie im Abschnitt 4.2.2 erwähnt, lässt sich die Materialdatei des OBJ Formats nicht in der SimpleMeshLayer einbinden, weshalb die Roboter einfarbig 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 [58]. Allerdings lässt sich die Rotationsänderungen nicht animieren, was vermutlich auch auf einen Bug in deck.gl zurückzuführen ist.

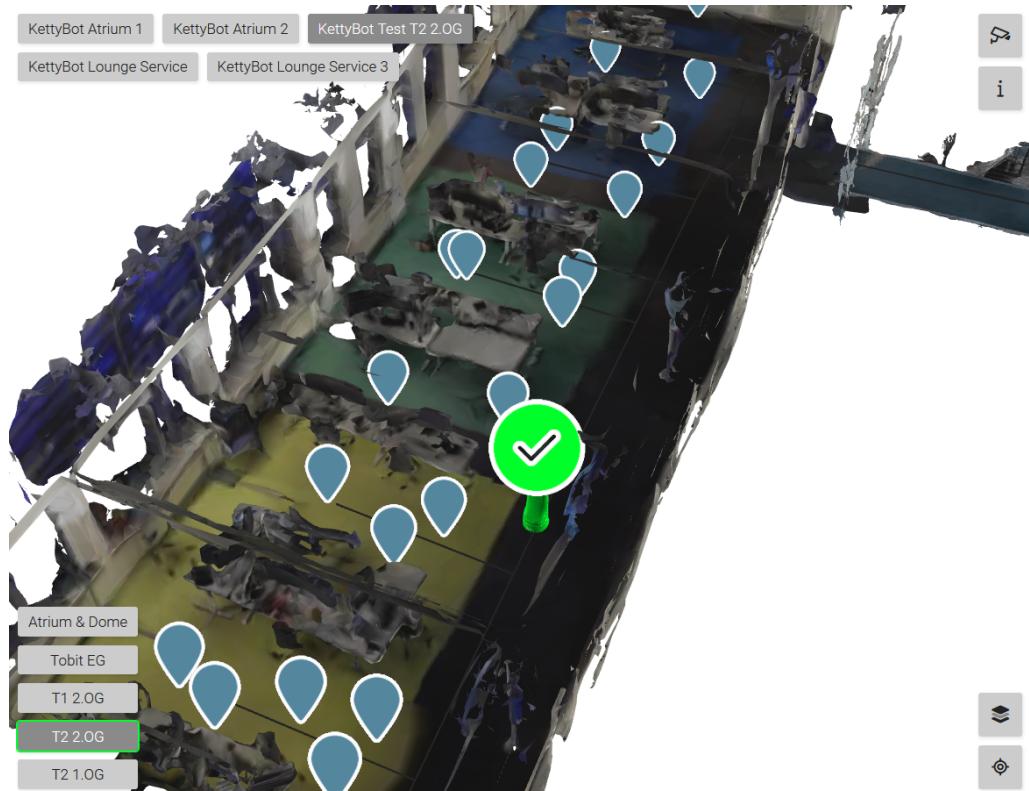
Über den Robotermodellen wird mithilfe der IconLayer der aktuelle Status der Roboter angezeigt. Während in einer ScenegraphLayer-Instanz nur ein bestimmtes 3D-Modell angezeigt werden kann, können in einer IconLayer-Instanz verschiedene Bilder mithilfe der getIcon Zugriffsfunktion angezeigt werden [31]. 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 das SVG Format nicht unterstützt, werden die SVG-Zeichenketten in das Data-URL Format umgewandelt. Das Data-URL Format kann Daten als Base64-Zeichenkette innerhalb einer URL einbetten [59] und kann von der SimpleMeshLayer ausgelesen werden [31].

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 [60] 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 [61] genutzt.

Bei der Abbildung 6 handelt es sich um einen Screenshot 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 6: Übersicht



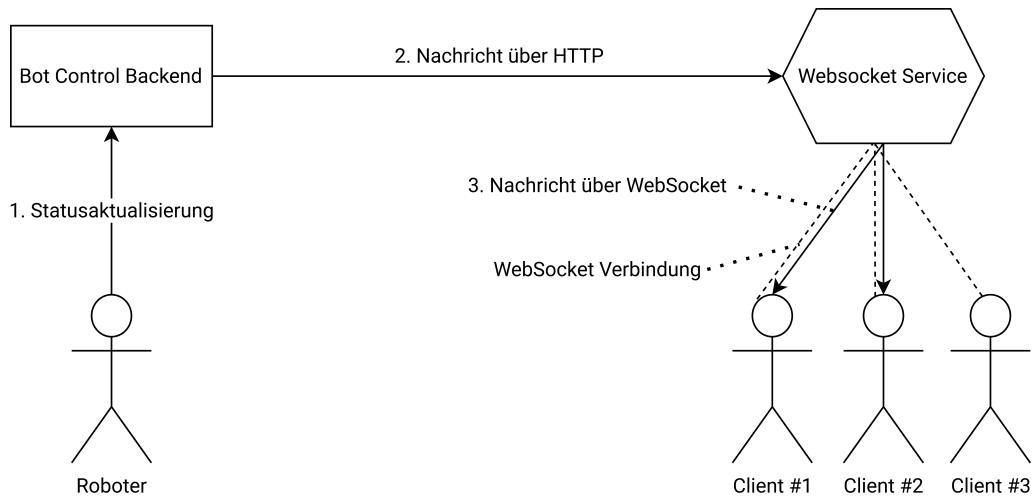
Quelle: Eigene Darstellung

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 der Webanwendung und dem BCB regelmäßig aktualisiert. Hierfür wird der im Abschnitt 2.2.2 erwähnte WebSocket-Service genutzt. In der Abbildung 7 ist der Ablauf einer Statusaktualisierung vereinfacht dargestellt. Aktualisiert ein Roboter seine Position, wird die entspre-

chende Information an das BCB gesendet. Wie die Kommunikation zwischen Robotern und BCB genau funktioniert wird im Abschnitt 2.1.4 genauer 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 7: Kommunikationsweg von Statusaktualisierungen der Roboter



Quelle: Eigene Darstellung

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

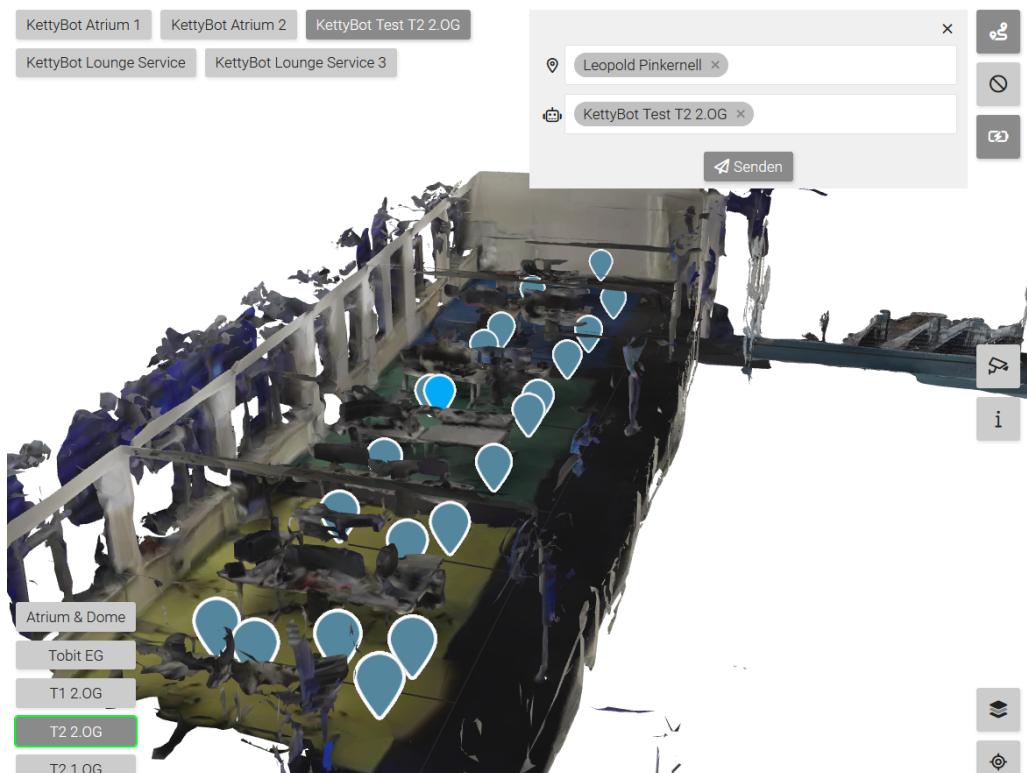
5.2.1.4 Interaktion

Die Roboter, sowie die Standorte sind mithilfe der `onClick` Property der entsprechenden Ebene [62] auswählbar. Ausgewählte Standorte und Roboter werden verfärbt angezeigt. Außerdem erscheint beim Auswählen eines Roboters ein Button über den sich weitere Informationen wie beispielsweise die Akkuladung anzeigen lassen. Schwebt die Maus über einem Standort oder Roboter, dann wird mithilfe der `getTooltip` Property [63] 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 mithilfe des FlyToInterpolators [64] zu dem ausgewählten Roboter bewegt. Beim Folgen eines Roboters wird die Kameraposition mithilfe der `transitionDuration` Property [65] 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-Komponente [66] der chayns-components, die eigentlich für das Suchen nach chayns Nutzern genutzt wird, für den Prototyp aber für die Suche nach Robotern und Standorten konfiguriert ist. Auch lassen sich Ziel und Roboter mit der Auswahl über die Karte festlegen. 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 und noch auf der Karte auswählbar. Zum endgültigen Ausführen der drei Aktionen werden die entsprechenden Endpunkte Robot/Call, Robot/Charge und Robot/Cancel im BCB [57] aufgerufen. Die Abbildung 8 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 8: Steuerung und Routenplanung

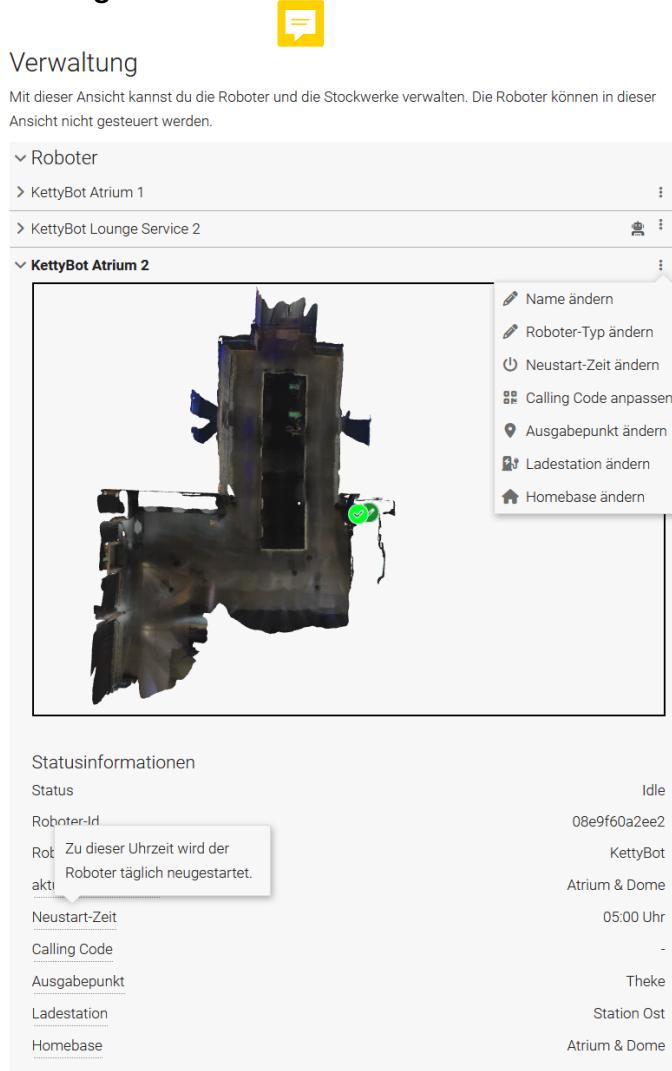


Quelle: Eigene Darstellung

5.2.3 Verwaltung

Die Verwaltung ist nicht besonders komplex, da die Daten der Roboter und Stockwerke sauber strukturiert sind und somit leicht mithilfe von React gemappt werden können. Hiermit ist gemeint, dass die verschiedenen Listen mithilfe der map Funktion zu Listen an React Komponenten umgewandelt werden können, die daraufhin gerendert werden können [67, S. 35-36].

In der Roboterliste werden im Gegensatz zum Mockup mehr Statusinformationen der Roboter angezeigt. Auch können mehr Einstellungen der Roboter geändert werden. Da die Übersicht über die verschiedenen Standorte auch über die Liste der Stockwerke ersichtlich ist und dadurch redundant ist, wurde diese aus der Roboterliste entfernt. Bei der Abbildung 9 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 9: Verwaltung

Quelle: Eigene Darstellung

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.

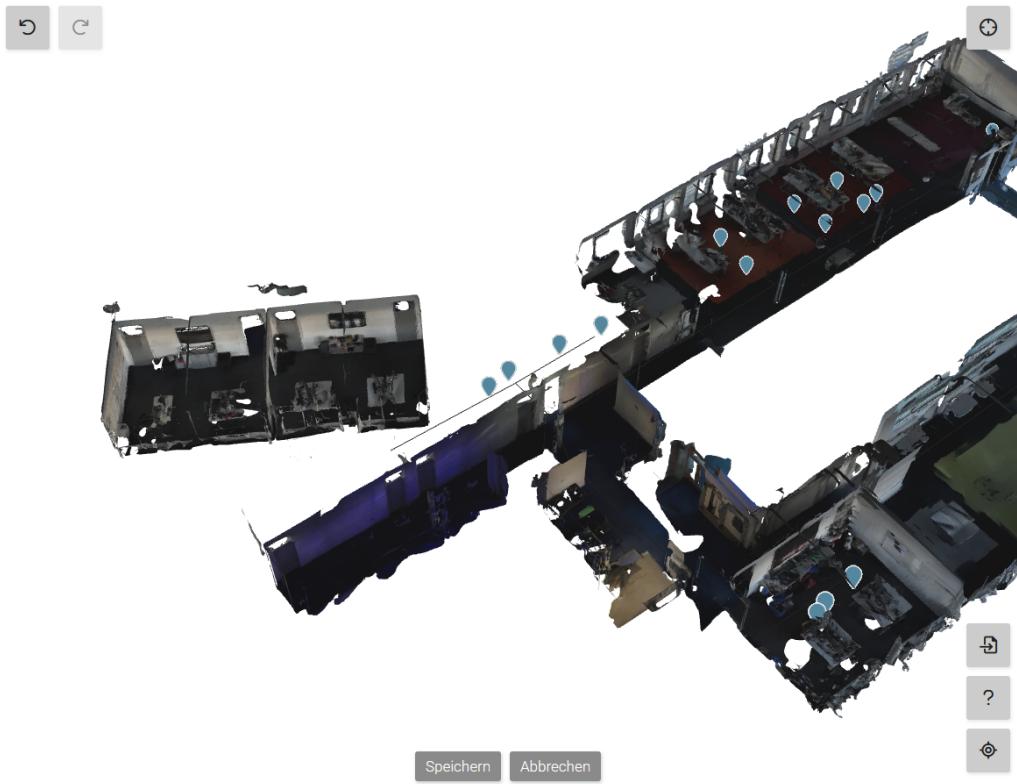
In beiden Listen gibt es eine Vorschau des entsprechenden Stockwerks, um die Position des Roboters oder der Standorte zu zeigen. Hierbei handelt es sich um die Nutzeransicht mit reduzierten Funktionen. Die Vorschau ist im Mockup automatisch geöffnet, was ein Problem ist, da die entsprechende deck.gl-Karteninstanz - aufgrund des Verhaltens der genutzten Aufklapper-Komponente - beim Öffnen des Aufklappers initialisiert wird. Die Initialisierung der deck.gl-Karteninstanz ist rechnerisch aufwändig und verursacht dadurch beim ersten Öffnen des Aufklappers starke Ruckler in der aufklapp-Animation. Um diese Ruckler zu verhindern wird ein Button angezeigt, über den die Karte manuell initialisiert

werden kann. Dadurch gibt es die Ruckler beim Klicken des Buttons und nicht beim Öffnen des Aufklappers, was weniger störend ist.

5.2.4 Editiermodus

In der Verwaltung, sowie in der Nutzeransicht 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 Nutzeransicht und unterscheidet sich nur durch andere Buttons und die Editermöglichkeiten. Da der Einsatz der Steuerungs- und Umschalttaste nötig ist, kann dieser Modus 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 [62] 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 wurden. Das Verschieben und Rotieren kann mithilfe der Tastenkombination Strg + Z rückgängig gemacht und mit Strg + Y wiederholt werden. Hierfür sind zwei Stapselspeicher implementiert in denen die vergangenen und rückgängig gemachten Aktionen per push hinzugefügt und per pop wieder herausgenommen werden. Bei Abbildung 10 handelt es sich um einen Screenshot des Editiermodus. Oben Links sind die Buttons zum rückgängig machen und wiederhohlen. 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, ein Button, über den man die verschiedenen Tastenkombinationen angezeigt bekommt und einen Button zum Zurücksetzen der Kameraposition. Außerdem gibt es unten Buttons zum Speichern und Abbrechen des Editierens.

Abbildung 10: Editiermodus



Quelle: Eigene Darstellung

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 nicht vollständig eben. Da die 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 Roboterdaten an einer leicht erhöhten z-Koordinate positioniert. Da der Boden bei allen 3D-Modellen an 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° 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° an der z-Achse rotiert und müssen vom Nutzer somit nur nach 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

TODO: Softwaretests beschreiben; Erwähnen, dass die deck.gl Darstellungen nicht automatisch getestet werden können und die Testabdeckung deswegen nicht ausreicht.

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 [68] und bei GitHub Pages handelt es sich um einen Hosting-Dienst, der in GitHub integriert ist und aus Repositories statische Websites erstellen kann [69]. So wird mithilfe der actions-gh-pages Github Action [70] bei der Aktualisierung des Haupt-Branches automatisch ein 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 [71], wie beispielsweise das Anfordern eines Zugangstokens, ohne den Funktionen des Backends nicht aufgerufen werden können. Aus diesem Grund funktioniert die Anwendung nur, wenn sie - wie in der Dokumentation des create-chayns-app Befehls beschrieben [50] - als Custom Page auf einer chayns Seite eingebunden ist. Der Zugriff auf die meisten Funktionen des BCBs ist so eingeschränkt, dass diese nur auf unternehmensinternen chayns Seiten 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 Usability und Performance erfüllt werden.

6.1 Funktionale Anforderungen

Im Abschnitt 3.2 werden die funktionalen Anforderungen erläutert. Die meisten dieser Anforderungen werden 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 Texturen nutzen und dieses noch nicht vollständig von Safari unterstützt wird [72]. 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 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, nur teilweise erfüllt.

Die Anforderungen an die Methode zur Gebäudemodell-Generierung konnten mit dem Einsatz des LiDAR-Scannens weitestgehend erfüllt werden. Für die Methode wird ein neueres iPhone benötigt, welches man als Nutzer unter Umständen bereits besitzt. Das Generieren der Modelle erfordert wenig Aufwand, wobei dieser davon abhängig ist, wie gründlich das Scannen durchgeführt wird. Insgesamt ist für das Scannen nur wenig Know-how nötig, da es in der Scaniverse App gut und einfach erklärt wird. Die entstandenen Modelle müssen für den Prototyp manuell komprimiert werden, wofür beispielsweise das im Abschnitt 4.2.2 erwähnte Webtool infrage kommt. In einem Produktivsystem könnten die Modelle aber auch mithilfe des glTF-Transform npm-Pakets [73] automatisch beim Import in den Editiermodus komprimiert werden. Die Qualität der erzeugten Modelle variiert zum einen dadurch wie gründlich die Scans durchgeführt wurden und zum anderen dadurch welche Methode zur Komprimierung des Modells genutzt wird. Insbesondere kleinere Ungenauigkeiten in den erzeugten Modellen können ignoriert werden, solange diese keinen Einfluss auf die Übersichtlichkeit des Modells haben.

6.2 Benutzerfreundlichkeit

Die Benutzerfreundlichkeit wird sowohl anhand der Erfüllung der Usability Entscheidungsregeln als auch durch die Auswertung von Usability Tests beurteilt.

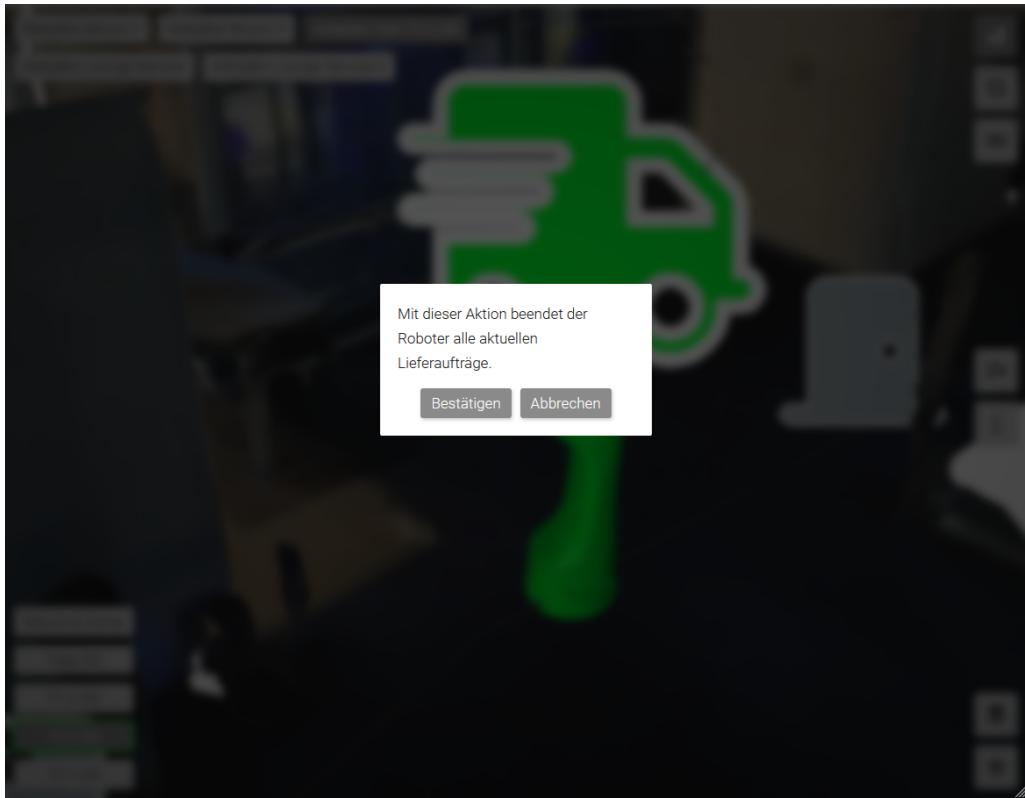
6.2.1 Usability Heuristics

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 [39, Regel 1]. Diese Regel ist 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. Auch 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 [39, 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 sowohl in der Steuerung als auch in der Verwaltung Bestätigungsdialoge mit denen neue Aktionen und Einstellungsänderungen 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 11 wird der Bestätigungsdialog gezeigt, der beim Abbrechen der Roboter Aktionen geöffnet wird.

Abbildung 11: Bestätigungsdialog



Quelle: Eigene Darstellung

In der fünften Regel geht es darum, dass Probleme, die Fehler auslösen, verhindert werden sollten [39, 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 währenddessen die bereits erwähnte Rückgängigmachen und Wiederholen Funktion.

In der siebten Regel geht darum, dass bestimmte, häufig genutzte Aktionen mit Shortcuts schneller ausführbar sein sollten [39, Regel 7]. So können Lieferstandort 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, was aber nicht automatisch bedeutet, dass der Prototyp auch wirklich benutzerfreundlich ist. Um das zu bestimmen, 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 dieser 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 [74] formuliert und während der Durchführung der Tests wurde darauf geachtet, dass die Testpersonen die Thinking-Aloud-Methode [75] einsetzen. Die ausführlicheren 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. 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 mithilfe des Editiermodus richtig positioniert werden. In der dritten Aktivität wird beispielsweise nicht nur geprüft wie Benutzerfreundlich das Editieren ist, sondern auch wie leicht der Editiermodus überhaupt gefunden werden kann.

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 aus dem Weg geschafft werden können [76]. 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 werden konnten, bevor die anderen Tests durchgeführt wurden.

6.2.2.1 Erste Usability Test Runde

In den beiden folgenden Tabellen werden die Ergebnisse des ersten Usability Testdurchlaufs zusammengefasst. Der Pilottest wird zu dieser Runde dazugezählt und ist in den Tabellen 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 aufgefallen sind, danach nicht mehr aufgetreten sind, was darauf zurückzuführen ist, dass diese vor der Durchführung der restlichen Tests behoben wurden. Man kann zudem sehen, dass den meisten Testpersonen mindestens ein Problem aufgefallen ist, das keiner anderen Testperson aufgefallen ist. 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 Korrektur der Probleme wurden die Probleme priorisiert, die besonders vielen Testpersonen aufgefallen sind. Durch die Usability Tests gab es außerdem zusätzlich Feedback der Testpersonen, das in der folgenden Implementierungsphase berücksichtigt wurde.

Tabelle 1: Gefundene Probleme in erster Usability Test Runde

	T0	T1	T2	T3	T4	T5
Problem 1	X			X		
Problem 2	X					
Problem 3	X					
Problem 4	X					
Problem 5	X					
Problem 6	X		X			
Problem 7	X					
Problem 8	X				X	
Problem 9		X				
Problem 10		X				
Problem 11		X	X	X	X	
Problem 12	X	X	X			
Problem 13			X			
Problem 14			X	X		
Problem 15	X		X			
Problem 16				X		
Problem 17				X		
Problem 18						X
Problem 19						X
Problem 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 weniger Probleme sind aufgetreten. Kein Wert bedeutet, dass die Testperson die Aktion nicht durchgeführt hat, da die Aktion für den Erfolg der Aktivität nicht benötigt wurde. Somit wird die Benutzerfreundlichkeit in den entsprechenden Teilen der Anwendung ersichtlich. Man sieht, dass die meisten Aktionen nach dem Pilot-test deutlich besser durchgeführt werden konnten, was auf die erwähnten Anpassungen am Prototyp zurückzuführen ist. Die Tabelle 2 zeigt, dass die meisten Aktionen zuverlässig durchgeführt werden können, 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 Test Runde

Aktion	T0	T1	T2	T3	T4	T5
Aktivität 1 (Übersicht)						
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
Aktivität 2 (Steuerung)						
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
Aktivität 3 (Editiermodus)						
Editormodus über Adminansicht	3	-	-	-	-	-
Editormodus 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 Zweite Usability Tests Runde

Die Ergebnisse der ersten Usability Tests Runde wurden in der darauffolgenden Implementierungsphase einbezogen. Verschiedene Probleme wurden behoben und 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 Bedienung 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 Test Runde

	T1	T2	T3	T4	T5
Problem 1	X				
Problem 2	X				
Problem 3		X			
Problem 4			X		
Problem 5				X	
Problem 6					X
Problem 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. Bei der zweiten Testperson wurde 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 Nutzeransicht gefunden wurde. So handelt es sich hier 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 Test Runde

Aktion	T1	T2	T3	T4	T5
Aktivität 1 (Übersicht)					
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
Aktivität - (Steuerung)					
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
Aktivität 3 (Editiermodus)					
Editormodus über Adminansicht	-	-	-	-	3
Editormodus 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 Forschungsfrage dieser Arbeit größere Relevanz haben. Ob die Liste der Roboter und Stockwerke in der Verwaltung besonders Benutzerfreundlich ist, ist für die Forschungsfrage nicht besonders relevant, da diese 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 Performance

Wie im Abschnitt 2.4.2 beschrieben werden PLS, LR und Smoothness gemessen. Für die Bestimmung des PLS wird der FCP gemessen. Normalerweise wird hierfür der LCP gemessen. Allerdings kann die Ladezeit des wichtigsten Elements - der deck.gl Karte - nicht

gemessen werden. Die LR wird über den FID und die TBT gemessen. Die Smoothness wird nur oberflächig ohne Messwert überprüft.

TODO: Ergebnisse erläutern.

7 Diskussion

Im Rahmen dieser Arbeit sollte die Frage beantwortet werden, wie eine effiziente und benutzerfreundliche Steuerung und Verwaltung von Servicerobotern implementiert werden kann. Es wurde ein Prototyp entwickelt, der diese Anforderungen erfüllt und im Kapitel 5 wird vorgestellt wie dieser im Detail entwickelt wurde. Außerdem sollte zusätzlich eine Methode zur einfachen Generierung von 3D-Modellen identifiziert und eingesetzt werden. Es wurde eine passende Methode gefunden, die erfolgreich eingesetzt wurde. Im Folgenden werden die Ergebnisse dieser Arbeit genauer zusammengefasst und interpretiert.

7.1 Erfüllung der Anforderungen

Wie in Kapitel 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 Kapitel 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 um weitere Funktionen wie eine erweiterte Routenplanung ergänzt werden.

Das Einhalten der Usability Entscheidungsregeln nach Nielsen [39] 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 zum Beispiel die Navigation innerhalb der Karte mitliefert. Diese orientiert sich bei der Handhabung an anderen gängigen Kartenanwendungen wie Google Maps. Das Framework trägt so beispielsweise dazu bei, dass die vierte Regel der Usability Entscheidungsregeln eingehalten wird, nach welcher den Standards ähnlicher Anwendungen gefolgt werden soll. 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 Test Runde 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 Test Runde 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 Test Runden angeboten, um zu Prüfen, ob noch

weitere Probleme auftreten. Weitere 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 konnte 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 ausreicht, der viel wichtigere LCP allerdings durch Beschränkungen in deck.gl nicht gemessen werden kann. Hierdurch sind die Messwerte leider nicht besonders aussagekräftig. Die subjektiv beobachtete Smoothness dient hier als Ausgleich, damit die Bewertung der Effizienz fundierter ist. Die gemessenen Werte sind nicht optimal, aber trotzdem zumindest meistens ausreichend. Vergleichsweise schlechte Messwerte sind durch die 3D-Darstellung der Stockwerke zu erwarten und somit noch im akzeptablen Bereich. Bei der Prüfung der Smoothness sind keine Ruckler aufgefallen, was ein positives Signal ist. Da die Darstellung der 3D-Modelle einen hohen Rechenaufwand fordert, erscheinen die schlechten Messwerte in Kombination mit der guten Smoothness verbesserungsfähig, aber trotzdem ausreichend. Für eine bessere Prüfung der Effizienz müsste deck.gl ein Event erweitert werden, das ausgelöst wird, wenn eine Ebene erstmals angezeigt wird.

Es wurde vorweg erwartet, dass die Anforderungen an den Prototyp mithilfe von deck.gl erfüllt werden können, da das Framework sowie die Möglichkeiten bereits bekannt waren. Trotzdem war vorweg nicht klar, wie bestimmte Funktionen - wie zum Beispiel das Synchronisieren der Roboterdaten und Gebäudemodelle - umgesetzt werden könnten. 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 Raummodell 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 außerdem eine Methode identifiziert und eingesetzt werden, die sich für das einfache Generieren von 3D-Modellen eignet. Es wurden verschiedene Methoden oberflächlich miteinander verglichen, wobei das LiDAR-Scannen

per iPhone mit der Scaniverse App letztendlich ausgewählt wurde. Trotz verschiedener Schwächen, wie das schlechte Scannen transparenter oder reflektierender Flächen, eignet sich die gewählte Methode gut. So ist sie mit vergleichsweise wenig Aufwand beim Scannen verbunden, erfordert kein gesondertes Know-how und ist mit keinen Anschaffungskosten verbunden, falls ein LiDAR fähiges iPhone zur Verfügung steht. In dieser Arbeit wurde explizit nicht nach der am besten geeigneten Funktion, sondern nur nach einer gut geeigneten Funktion gesucht, um den Umfang der Arbeit nicht zu überziehen. 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, dass eigentlich für die Visualisierung riesiger Geodatensätze ausgelegt ist. Diese Arbeit zeigt, dass ebenen basierte Visualisieren von Daten nach dem PIL-Prinzip, nicht nur für Geodaten, sondern auch für Raumbezogene Daten - wie die Roboterdaten und die 3D-Modelle - geeignet sind. 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 zum Großteil auf deck.gl basiert, gab es vor und während der Entwicklung die Hoffnung, dass die Entwicklung mit dem Framework möglichst unkompliziert ist. 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, ist es zukunftssicher.

Der Einsatz der Technologien Sass, TypeScript, React und React-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 wahrscheinlich auch auf die Effizienz des Prototyps auswirken. Der Einsatz von React-Redux hat die Entwicklung mit der Vielzahl an angefragten Roboterdaten vereinfacht, da diese zentral gespeichert und aus allen React-Komponenten abgerufen werden können.

7.3 Bewertung der Methodik

Zur Beantwortung der Forschungsfrage wurde nach dem DSR Ansatz nach Hevner [7] gearbeitet. Rückblickend wurde die gewählte Vorgehensweise passend gewählt, 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 Rapid Prototypings abgebildet.

8 Fazit

Im Rahmen der 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.

Der lauffähige Prototyp ist mithilfe der Anleitung "Prototyp.txt", die in den Zusatzdokumenten zu finden ist, erreichbar und zumindest eingeschränkt nutzbar. So können Verwaltungs- und Steuerungsfunktionen aus Sicherheitsgründen nicht genutzt werden.



8.1 Aufgetretene Probleme

Während der Entwicklung des Prototyps wurden verschiedene Probleme identifiziert, die mit deck.gl oder dem Einbinden von 3D-Modellen im Web in Verbindung stehen. So ist die SimpleMeshLayer des Frameworks dadurch beschränkt das OBJ Dateien nur ohne die .mtl Datei, also ohne Textur eingebunden werden können. Dadurch musste bei der Anzeige der Gebäudemodelle auf die ScenegraphLayer zurückgegriffen werden und die Robotermodelle konnten nur einfarbig ohne Textur dargestellt werden. Insgesamt gibt es beim Animieren von 3D-Modellen verschiedene Probleme: In der ScenegraphLayer funktioniert das Animieren über die transition Property gar nicht, während in der SimpleMeshLayer nur das Animieren der Rotation nicht funktioniert. Da deck.gl WebGL für die Darstellung nutzt, können automatisierte Tests nur über den Vergleich von Screenshots durchgeführt werden. Hierfür existiert zwar der SnapshotTestRunner der diesen Prozess automatisieren kann, die Klasse ist allerdings nicht ausreichend dokumentiert, weshalb deck.gl Funktionen im Prototyp nicht automatisch getestet werden können. Da das Framework ein Open-Source-Projekt ist und aktiv aktualisiert wird, können diese Probleme behoben werden, nachdem diese gemeldet wurden.

Beim Einbinden von 3D-Modellen gibt es weitere Probleme, die unabhängig von deck.gl auftreten. So können die verwendeten glTF Modelle mit WebP Texturen nicht im Safari Browser genutzt werden. Auch gibt es an Mobilgeräten Probleme bei der Darstellung großer und somit rechenaufwändiger 3D-Modelle.

8.2 Ausblick

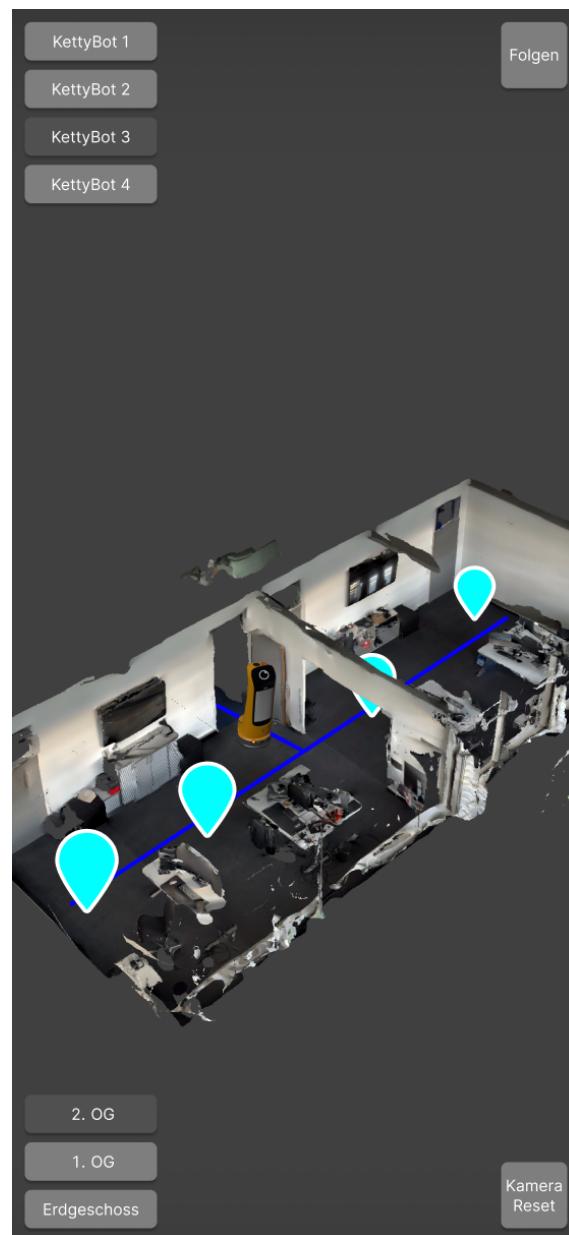
Da die Anforderungen an den Prototyp erfüllt waren konnten ist der nächste logische Schritt die Implementierung als Produktivsystem. Wie bereits erwähnt sind hierfür verschiedene Anpassungen im Prototyp und BCB nötig, die allerdings nicht sonderlich groß ausfallen. Im Backend müssen vor allem neue Endpunkte 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 eingebaut werden.

Zusätzlich können auch weitere Aspekte erforscht werden, die im Rahmen dieser Arbeit nur oberflächlich betrachtet wurden. Wie bereits demonstriert wird 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 Aspekte. 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

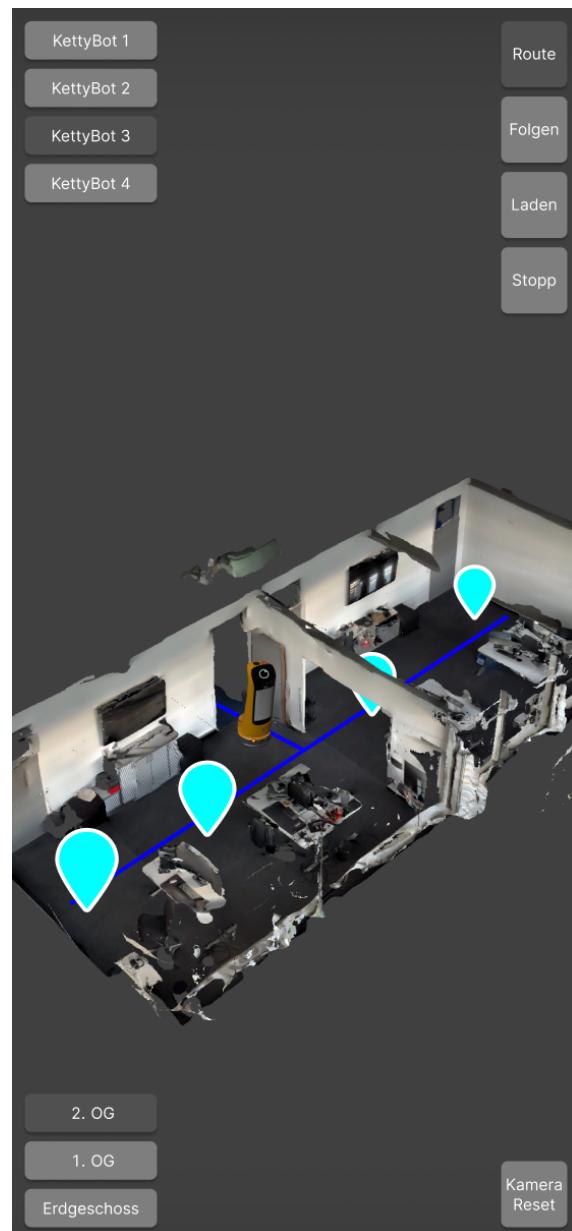
Anhang 1: Bilder

Abbildung 12: Mockup der Übersicht



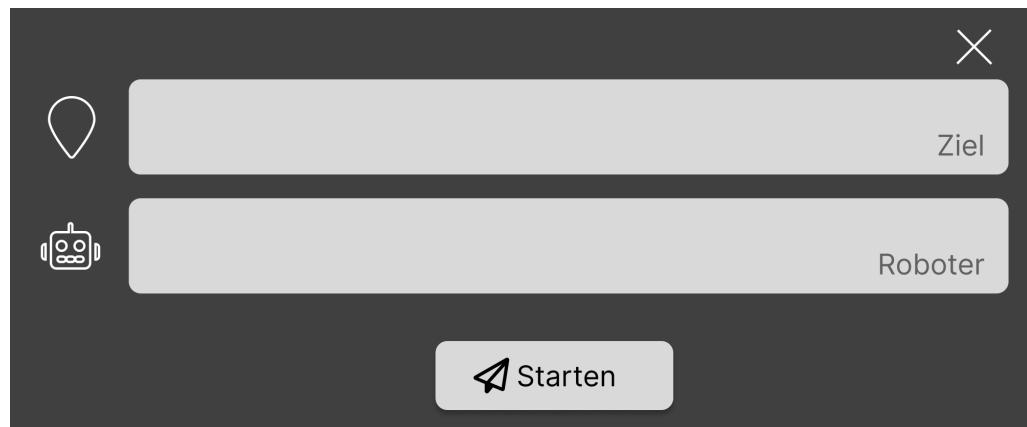
Quelle: Eigene Darstellung

Abbildung 13: Mockup der Steuerung



Quelle: Eigene Darstellung

Abbildung 14: Mockup des Routenplanungs-Popup



Quelle: Eigene Darstellung

Abbildung 15: Mockup der Verwaltung

Quelle: Eigene Darstellung

Anhang 2: Tabellen

Tabelle 5: Beschreibung der Probleme in erster Usability Test Runde

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

Problem	Beschreibung
Problem 1	Unklarheit darüber wie man die Karte rotiert.
Problem 2	Beim Entfernen einer Roboterauswahl, wird in das Stockwerk des Roboters gewechselt, obwohl das nur bei der Auswahl eines Roboters passieren sollte.
Problem 3	Es wurde erwartet, dass man den Roboter über das Ladestations-Icon auf der Karte zur Ladestation schicken kann.
Problem 4	Suchfunktion zum Auswählen eines Standorts nicht gefunden, da das Routenplanungs-Popup nicht geöffnet wurde.
Problem 5	Irritation darüber, dass die Steuerung über einen Tooltip erklärt wird.
Problem 6	Editiermodus nicht im Nutzermodus gefunden, da das Icon des Buttons falsch verstanden wurde.
Problem 7	Editiermodus 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.
- [6] Sprenger, Michaela; Mettler, Tobias: Service Robots, In: *Business & Information Systems Engineering*, 2015.
- [7] Hevner, Alan R.; March, Salvatore T.; Park, Jinsoo; Ram, Sudha: Design Science in Information Systems Research, In: *MIS Quarterly*, 2004.
- [10] Gonzalez-Aguirre, Juan A.; Osorio-Oliveros, Ricardo; Rodríguez-Hernández, Karen L. et al.: Service Robots - Trends and Technology, In: *Applied Sciences*, 2021.
- [14] Robotics, Pudu: SDK Guidance Document for PuduTech Open Platform for Robotic Services (Nodejs Microservice).
- [15] Parisi, Tony: Programming 3D applications with HTML5 and WebGL - 3D animation and visualization for web pages, 1st edition, O'Reilly, Sebastopol, Kalifornien 2014, ISBN: 9781449362966.
- [16] Aber, James S.; Marzolff, Irene; Ries, Johannes B.: Small-Format Aerial Photography, Elsevier, Amsterdam 2010, ISBN: 978-0-444-53260-2.
- [24] 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.
- [25] 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.
- [30] Wang, Jang: Deck.gl - Large-scale Web-based Visual Analytics Made Easy, 2019.
- [37] Balzert, Helmut: Software-Management Software-Qualitätssicherung Unternehmensmodellierung, Spektrum Akademischer Verlag, Heidelberg 1998, ISBN: 3827400651.
- [40] Dumas, Joseph S.; Redish, Janice C.: A practical guide to usability testing, Revised edition, Intellect Books, Exeter 1999, ISBN: 9781841500201.
- [67] Boduch, Adam; Derks, Roy: React and React Native - A complete hands-on guide to modern web and mobile development with React.js, Third edition, Packt Publishing, Birmingham, UK 2020, ISBN: 9781839212437.
- [70] Ueda, Shohei: *actions-gh-pages*, Version 3.9.3, 29.02.2024. Adresse: <https://github.com/peaceiris/actions-gh-pages>.

Internetquellen

- [2] International Federation of Robotics: Service-Roboter-Absatz steigt weltweit um 48 Prozent - Personalmangel treibt die Nachfrage, Adresse: https://ifr.org/downloads/press2018/DE-2023-10-11-IFR-Pressemeldung_World_Robotics_Service_Robots_2023.pdf, 2023, Zugriff am: 03.01.2024.
- [3] Bill, Marina; Müller, Christopher; Kraus, Werner; Bieller, Susanne: World Robotics Report 2023 - Press Conference, Adresse: <https://www.youtube.com/watch?v=h-3ndnde8dY>, 2023, Zugriff am: 03.01.2024.
- [4] Bundesagentur für Arbeit: Bestand an offenen Arbeitsstellen im Jahresdurchschnitt bis 2024, Adresse: <https://de.statista.com/statistik/daten/studie/2903/umfrage/jahresdurchschnittswerte-des-bestands-an-offenen-arbeitsstellen>, 2024, Zugriff am: 28.02.2024.
- [5] Jansen, Anika; Risius, Paula: Sorgenkind Gastro?: Berufswechsel in der Corona-Pandemie, Adresse: <https://www.iwkoeln.de/studien/anika-jansen-paula-risius-berufswechsel-in-der-corona-pandemie.html>, 2022, Zugriff am: 04.03.2024.
- [8] ISO: ISO 8373:2021 - Robotics - Vocabulary, Adresse: <https://www.iso.org/standard/75539.html>, 2021, Zugriff am: 05.01.2024.
- [9] International Federation of Robotics: Service Robots, Adresse: <https://ifr.org/service-robots>, Zugriff am: 05.01.2024.
- [11] Pudu Robotics: Smart Catering, Adresse: <https://www.pudurobotics.com/solutions/Smart%20Catering>, Zugriff am: 05.01.2024.
- [12] Nature Research Custom: Intelligent robots offer service with a smile, Adresse: <https://www.nature.com/articles/d42473-022-00395-5>, 2022, Zugriff am: 15.01.2024.
- [13] Robotics, Pudu: Pudu Robotics to Expand Service Scenarios with Its Proprietary Upgraded PUDU VSLAM+, Adresse: <https://www.pudurobotics.com/de/news/784>, 2023, Zugriff am: 15.01.2024.
- [17] Cohrs, Jonathan; Boonyapanachoti, Mint; Aneja, Sukanya; Köerner, Willa; Kim, Min-kyoung: An End-to-End Guide to Photogrammetry with Mobile Devices, Adresse: <https://rd.nytimes.com/projects/an-end-to-end-guide-to-photogrammetry-with-mobile-devices>, 2021, Zugriff am: 16.01.2024.
- [18] Cohrs, Jonathan; Boonyapanachoti, Mint; Aneja, Sukanya; Köerner, Willa; Kim, Min-kyoung: Capturing Images for Photogrammetry, Adresse: <https://rd.nytimes.com/projects/capturing-images-for-photogrammetry>, 2021, Zugriff am: 16.01.2024.
- [19] Cohrs, Jonathan; Boonyapanachoti, Mint; Aneja, Sukanya; Köerner, Willa; Kim, Min-kyoung: Processing and Aligning 3D Scenes, Adresse: <https://rd.nytimes.com/projects/processing-and-aligning-3d-scenes>, 2021, Zugriff am: 16.01.2024.
- [20] Fenstermaker: What Cell Phones Have LiDAR?, Adresse: <https://blog.fenstermaker.com/what-cell-phones-have-lidar>, 2022, Zugriff am: 16.01.2024.
- [21] Occipital: Canvas, Verfügbar unter: <https://apps.apple.com/us/app/canvas-lidar-3d-measurements/id1169235377>, Version: 3.48, Zugriff am: 28.02.2024.

- [22] Polycam: Polycam 3D Scanner, Verfügbar unter: <https://apps.apple.com/us/app/polycam-3d-scanner-lidar-360/id1532482376>, Version: 3.3.20, Zugriff am: 28.02.2024.
- [23] Toolbox AI: Scaniverse, Verfügbar unter: <https://apps.apple.com/us/app/scaniverse-3d-scanner/id1541433223>, Version: 2.1.9, Zugriff am: 28.02.2024.
- [26] Surma: WebGPU — All of the cores, none of the canvas, Adresse: <https://surma.dev/things/webgpu>, 2022, Zugriff am: 29.02.2024.
- [27] Seguin, Damien: A collection of WebGL and WebGPU frameworks and libraries, Adresse: <https://gist.github.com/dmnsgn/76878ba6903cf15789b712464875cfdc>, Zugriff am: 27.02.2024.
- [28] OpenJS Foundation: Large scale geospatial data visualization, Adresse: <https://vis.gl>, 27.02.2024, Zugriff am: 27.02.2024.
- [29] Green, Ib; Palmer, Felix; McCurdy, Don: v9 Tracker, Adresse: <https://github.com/visgl/deck.gl/issues/7457>, 2022, Zugriff am: 27.02.2024.
- [31] OpenJS Foundation: IconLayer | deck.gl, Adresse: <https://deck.gl/docs/api-reference/layers/icon-layer>, Zugriff am: 20.02.2024.
- [32] OpenJS Foundation: IconLayer - Meteorites Landings, Adresse: <https://deck.gl/examples/icon-layer>, Zugriff am: 27.02.2024.
- [33] OpenJS Foundation: SimpleMeshLayer | deck.gl, Adresse: <https://deck.gl/docs/api-reference/mesh-layers/simple-mesh-layer>, Zugriff am: 20.02.2024.
- [34] OpenJS Foundation: ScenegraphLayer | deck.gl, Adresse: <https://deck.gl/docs/api-reference/mesh-layers/scenegraph-layer>, Zugriff am: 20.02.2024.
- [35] OpenJS Foundation: Controller | deck.gl, Adresse: <https://deck.gl/docs/api-reference/core/controller>, Zugriff am: 20.02.2024.
- [36] OpenJS Foundation: Viewport | deck.gl, Adresse: <https://deck.gl/docs/api-reference/core/viewport>, Zugriff am: 27.02.2024.
- [38] ISO: ISO/IEC 25010:2011 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, Adresse: <https://www.iso.org/standard/35733.html>, 2011, Zugriff am: 27.02.2024.
- [39] Nielsen, Jakob: 10 Usability Heuristics for User Interface Design, Adresse: <https://www.nngroup.com/articles/ten-usability-heuristics>, 1994, Zugriff am: 20.02.2024.
- [41] Budiu, Raluca: Quantitative vs. Qualitative Usability Testing, Adresse: <https://www.nngroup.com/articles/quant-vs-qual>, 2017, Zugriff am: 15.02.2024.
- [42] Moran, Kate: Usability Testing 101, Adresse: <https://www.nngroup.com/articles/usability-testing-101>, 2019, Zugriff am: 15.02.2024.
- [43] Nielsen, Jakob: How Many Test Users in a Usability Study?, Adresse: <https://www.nngroup.com/articles/how-many-test-users>, 2012, Zugriff am: 15.02.2024.
- [44] Internet Assigned Numbers Authority: mtl, Adresse: <https://www.iana.org/assignments/media-types/model/mlt>, 2020, Zugriff am: 27.02.2024.

- [45] Internet Assigned Numbers Authority: obj, Adresse: <https://www.iana.org/assignments/media-types/model/obj>, 2020, Zugriff am: 27.02.2024.
- [46] OpenJS Foundation: OBJLoader | loaders.gl, Adresse: <https://loaders.gl/docs/modules/obj/api-reference/obj-loader>, Zugriff am: 27.02.2024.
- [47] The Khronos® 3D Formats Working Group: glTF™ 2.0 Specification, Adresse: <https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html>, 2021, Zugriff am: 27.02.2024.
- [48] Adhiban, Antony: OptimizeGLB, Verfügbar unter: <https://optimizeglb.com>, Version: 0.1.3, Zugriff am: 28.02.2024.
- [49] Figma: What is Figma?, Adresse: <https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma>, Zugriff am: 20.02.2024.
- [50] Tobit.Software: create-chayns-app, Verfügbar unter: <https://github.com/TobitSoftware/create-chayns-app>, Version: 1.1.10, Zugriff am: 20.02.2024.
- [51] Tobit.Software: chayns-toolkit, Verfügbar unter: <https://github.com/TobitSoftware/chayns-toolkit>, Version: 2.0.8, Zugriff am: 20.02.2024.
- [52] Tobit.Software: chayns-components, Verfügbar unter: <https://github.com/TobitSoftware/chayns-components>, Version: 4.31.4, Zugriff am: 20.02.2024.
- [53] Edwards, Luke: clsx, Verfügbar unter: <https://github.com/lukeed/clsx>, Version: 2.1.0, Zugriff am: 20.02.2023.
- [54] Fonticons: SVG Core | Font Awesome Docs, Adresse: <https://fontawesome.com/docs/web/dig-deeper/svg-core>, Zugriff am: 20.02.2024.
- [55] OpenJS Foundation: Performance Optimization | deck.gl, Adresse: <https://deck.gl/docs/developer-guide/performance>, Zugriff am: 20.02.2024.
- [56] Computer Hope: What is Back-face Culling?, Adresse: <https://www.computerhope.com/jargon/b/backface-culling.htm>, 2022, Zugriff am: 20.02.2024.
- [57] Tobit.Software: Web.API | Swagger UI, Adresse: https://cube.tobit.cloud/pudu-api/v2/_docs/index.html, Zugriff am: 20.02.2024.
- [58] OpenJS Foundation: Layer Class | deck.gl, Adresse: <https://deck.gl/docs/api-reference/core/layer>, Zugriff am: 20.02.2024.
- [59] Masinter, Larry: RFC 2397 - The data URL scheme, Adresse: <https://datatracker.ietf.org/doc/html/rfc2397>, 1998, Zugriff am: 28.02.2024.
- [60] OpenJS Foundation: PathLayer | deck.gl, Adresse: <https://deck.gl/docs/api-reference/layers/path-layer>, Zugriff am: 20.02.2024.
- [61] OpenJS Foundation: PathStyleExtension | deck.gl, Adresse: <https://deck.gl/docs/api-reference/extensions/path-style-extension>, Zugriff am: 20.02.2024.
- [62] OpenJS Foundation: Adding Interactivity | deck.gl, Adresse: <https://deck.gl/docs/developer-guide/interactivity>, Zugriff am: 20.02.2024.
- [63] OpenJS Foundation: Deck | deck.gl, Adresse: <https://deck.gl/docs/api-reference/core/deck>, Zugriff am: 20.02.2024.

- [64] OpenJS Foundation: FlyToInterpolator | deck.gl, Adresse: <https://deck.gl/docs/api-reference/core/fly-to-interpolator>, Zugriff am: 20.02.2024.
- [65] OpenJS Foundation: Animations and Transitions | deck.gl, Adresse: <https://deck.gl/docs/developer-guide/animations-and-transitions>, Zugriff am: 20.02.2024.
- [66] Tobit.Software: PersonFinder | chayns-components, Adresse: <https://github.com/TobitSoftware/chayns-components/blob/master/docs/components/person-finder.md>, Zugriff am: 20.02.2024.
- [68] 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.
- [69] 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.
- [71] Tobit.Software: chayns-api, Verfügbar unter: <https://github.com/TobitSoftware/chayns-api>, Version: 1.0.20, Zugriff am: 20.02.2024.
- [72] Deveria, Alexis: Can I use webp?, Adresse: <https://caniuse.com/?search=webp>, Zugriff am: 20.02.2024.
- [73] McCurdy, Don: glTF-Transform, Verfügbar unter: <https://github.com/donmccurdy/gltf-transform>, Version: 3.10.0, Zugriff am: 20.02.2024.
- [74] Pernice, Kara: How to Maximize Insights in User Testing: Stepped User Tasks, Adresse: <https://www.nngroup.com/articles/user-testing-stepped-tasks>, 2020, Zugriff am: 15.02.2024.
- [75] Nielsen, Jakob: Thinking Aloud: The #1 Usability Tool, Adresse: <https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool>, 2012, Zugriff am: 15.02.2024.
- [76] Schade, Amy: Pilot Testing: Getting It Right (Before) the First Time, Adresse: <https://www.nngroup.com/articles/pilot-testing>, 2015, Zugriff am: 15.02.2024.

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, 6.3.2024

(Ort, Datum)

(Eigenhändige Unterschrift)