Architekturdokumentation

Demonstrator

Crunchtime Wonders
Joshua Krcmar
Erik Martens
Tuncay Yildiz
Heike Welter
Shayan Mohajerani

Version
2.0

Datum der Abgabe
30.06.2022

Projektrahmen

Kunde(n)	objective partner AG (i. A. von ligenium GmbH)
Kundenkontakt E-Mail: **objective partner:** **ligenium:**	 michael.thron@objective-partner.com marian.wensky@objective-partner.com angela.grimmer@ligenium.de christoph.alt@ligenium.de
Produktname	LigMaTrack
Management	Prof. Dr. Peter Knauber p.knauber@hs-mannheim.de Prof. Kirstin Kohler k.kohler@hs-mannheim.de
Zeitraum	24.03.2022 bis 30.06.2022

1 Einleitung

Das Architekturdokument beschreibt den Aufbau des Demonstrators vom Softwaresystem *LigMaTrack*. Dieser Demonstrator wird an der Hochschule Mannheim durch das Team “Crunchtime Wonders” innerhalb des Kurses Projekt Software Engineering 2 (*PSE2)* entwickelt und implementiert.

Hinter dem Projekt steht der Auftraggeber objective partner (*op*). Dieser Auftraggeber arbeitet mit *ligenium* zusammen. Die Firma ligenium ist wiederum der Auftraggeber von op und stellt *Ladungsträger* zum Transport von Werkstücken her.

Der Aufbau des Dokuments orientiert sich am bekannten *arc42*-Template (https://www.arc42.de/overview/). Alle *kursiv* markierten Begriffe und Abkürzungen (Erstes Vorkommen) befinden sich im Glossar und Abkürzungsverzeichnis (siehe Kapitel 9).

2 Einführung und Ziele

In diesem Kapitel werden die Aufgabenstellung, die Qualitätsziele sowie die im Demonstrator berücksichtigten Anforderungen vorgestellt.

2.1 Aufgabenstellung

Die Nutzer des Softwaresystems LigMaTrack sind die Logistikmitarbeiter. Diese sind für den Transport von Ladungsträgern innerhalb des Lagers zuständig. Werkstücke werden unter Verwendung von Ladungsträgern von einem Standort zum nächsten befördert. Der Transport beinhaltet unter anderem lange Wegstrecken. Eine Nachverfolgbarkeit der Standorte von Ladungsträgern existiert bisher bei ligenium nicht.

Das System LigMaTrack ermöglicht das Auffinden der Ladungsträger eines Auftrags im Lager. Befindet sich ein Ladungsträger nicht am angegebenen Lagerplatz, kann das System dessen Standort auf einem digitalen Lagerplan anzeigen, sodass dieser leicht lokalisiert werden kann. Die verbesserte Auffindbarkeit von Ladungsträgern kann die Produktivität durch Zeitersparnisse erhöhen.

Sendet ein *Sensor* keine Positionsdaten mehr, kann das folgende Hintergründe besitzen:

- Die Verbindung des Sensors wurde unterbrochen
- Der Sensor ist beschädigt oder defekt
- Die Batterie des Sensors ist leer

Die Überwachung der Positionsdaten kann daher auch als Indikator für nötige Wartungsarbeiten an Ladungsträgern verwendet werden. Außerdem hat der Logistikmitarbeiter die Möglichkeit, Ladungsträger zu markieren, welche dieser augenscheinlich als beschädigt erkennt. LigMaTrack verbessert somit die Betriebsbereitschaft der Ladungsträger.

2.2 Übersicht der funktionalen Anforderungen

Die folgende Tabelle enthält die im Demonstrator berücksichtigen funktionalen Anforderungen (*FA*), welche vom System LigMaTrack abgedeckt werden. Dazu hat die jeweilige Anforderung eine textuelle Beschreibung, welche diese unter Berücksichtigung der Umgebung des Systems LigMaTrack schildert.

Die aufgezählten FA stehen im Kapitel 3.3 der Anforderungsspezifikation LigMaTrack und können dort nachgelesen werden.

Die Anforderungsspezifikation entstand im Rahmen des Kurses Projekt Software Engineering 1 (*PSE1*) unter Kooperation durch das Team Binary Riot der Hochschule Mannheim und der Firma objective partner AG (*op*). Darin wird das Produkt “Ligenium Material Tracking” (LigMaTrack). Zu diesem sind Anforderungen sowie Qualitätsanforderungen beschrieben und festgehalten.

FA-Nr.	**Beschreibung**	**Systemkontextbeschreibung**
9	LigMaTrack muss alle Waren auf einem Ladungsträger auflisten.	Das System listet die Inhalte des Ladungsträgers des ausgewählten Auftrags auf. Der Logistikmitarbeiter kann anhand der zur Verfügung stehenden Informationen den Inhalt des Ladungsträgers prüfen und abgleichen.
10	LigMaTrack muss die Visualisierung des physischen Lagers digital für die Endanwender darstellen.	Das System stellt für die Nutzer (Logistikmitarbeiter) eine digitale Visualisierung des Lagers dar. Im System erfolgt eine Darstellung der Ladungsträger des ausgewählten Auftrags im Lagerplan, sodass der Logistikmitarbeiter die Standorte aller erforderlichen Ladungsträger für diesen Auftrag auf dem Lagerplan sieht. Der Logistikmitarbeiter kann mithilfe der Darstellung nachvollziehen, wo sich die benötigten Ladungsträger derzeit im Lager befinden.
11	LigMaTrack soll die Kommissionieraufträge auflisten.	Das System listet die Aufträge unabhängig des Bearbeitungsstatus auf.

Tabelle 2.1 – Systemkontextbeschreibung der FA

2.3 Projektbeteiligte

In diesem Kapitel sind alle Projektbeteiligte gelistet, welche ein Interesse am Verlauf oder Ergebnis des Demonstrators LigMaTrack haben.

Projektbeteiligte	**Interesse/Bezug**
Logistikmitarbeiter	Logistikmitarbeiter sind die Endnutzer unserer Applikation. Sie werden mit dem System arbeiten, da sie für die Erledigung der Arbeitsaufträge Ladungsträger mit Produktionsteilen vom Lager in die Produktionsstätte transportieren. Dabei handelt es sich um Logistikmitarbeiter, welche bei Firmen angestellt sind, die ligenium Ladungsträger und den Software-Service nutzen. Die Idee für diesen Demonstrator mit dem Logistikmitarbeiter als Endnutzer entstand durch die vorausgegangene Anforderungsspezifikation von Binary Riot.
objective partner AG (op)	Der Auftraggeber arbeitet mit ligenium zusammen. Ligenium beauftragte objective partner ein Asset as a Service Angebot (*AaaS*-Angebot) für die Ladungsträger zu konzipieren. Der Demonstrator dient als Konzeptstudie. Er übernimmt auf einen Teilbereich eines solchen AaaS-Angebots. Dafür orientiert er sich an ausgewählten Funktionalitäten aus der Anforderungsspezifikation des Teams Binary Riot.
ligenium GmbH	Das Unternehmen konstruiert und verkauft Ladungsträger aus Holz an Geschäftskunden. Zur Erweiterung des Geschäftsmodells möchte ligenium gegen Bezahlung eine AaaS-Platform für ihre Kunden anbieten. Ligenium beauftragte objective partner.
Team Crunchtime Wonders	Das Team erstellt im Auftrag von objective

	partner einen Demonstrator.
Management Prof. Dr. Peter Knauber p.knauber@hs-mannheim.de Prof. Kirstin Kohler k.kohler@hs-mannheim.de	Die Professoren der Hochschule Mannheim betreuen das Projekt in PSE2 im Sommersemester (*SS*) 2022.

Tabelle 2.3 – Projektbeteiligte mit Interesse an LigMaTrack

3 Randbedingungen

Für die Entwicklung des Demonstrators wurden durch den Auftraggeber op keine technischen Randbedingungen definiert und vorgegeben. Die Entwicklung kann mit beliebigen gängigen Technologien (Programmiersprachen, Frameworks, Toolchains)umgesetzt werden. Der Software zum Ausrollen und installieren des entwickelten Softwaresystems wurden ebenfalls keine Randbedingung gesetzt und ist somit frei wählbar. Auch das Zielsystem wurde nicht näher spezifiziert.

4 Lösungsstrategie

Die Umsetzung von LigMaTrack erfolgt in der Form einer *WebApp*. Eine genaue Begründung für die gewählte Architektur sowie die gewählten Technologien erfolgt in Kapitel 8.

Das System untergliedert sich in drei Schichten und folgt einer Client-Server-Architektur. So gibt es eine Komponente für das *Frontend*, den Client-Anteil. Zusätzlich existieren eine *Backend*- und eine Datenpersistierungs-Komponente. Diese machen den Server-Anteil der Architektur aus. Die Schichten kommunizieren untereinander jeweils mit Hilfe von Webschnittstellen (a.A. *REST*).

Für die Umsetzung wird der MERN-Technologie-Stack[1] verwendet, welcher sich in der Industrie als Grundlage für skalierbare und verlässliche Webanwendungen durchgesetzt hat. D.h. das Frontend wird mit *React* realisiert, das Backend verwendet *Node.js* und *Express*. Für die Persistierung von Daten wird eine *NoSQL*-Datenbank eingesetzt, wobei die Entscheidung auf *MongoDB* fiel.

LigMaTrack kommuniziert mit externen Systemen, welche Teil der ligenium Infrastruktur sind bzw. Teil einer erweiterten *Industrie 4.0* Infrastruktur sein können. Dabei handelt es sich um Systeme die bestimmte Daten zentral verwalten. Sie dienen als Datenquelle und -speicher für diese Informationen, die LigMaTrack benötigt und ggf. verändert. LigMaTrack lässt sich

[1] Technologien zum Umsetzen von Webanwendung, bestehend aus MongoDB, Express, React und Node.js (siehe Kapitel 8)

zusätzlich zu diesen Systemen betreiben. Es baut lediglich auf Ihnen auf und nutzt bereitgestellte Daten. In diesem Zusammenhang können auch andere Applikationen existieren, welche die besprochenen externen Systeme als Datenquelle und -speicher nutzen. Eine Darstellung des Systemkontexts erfolgt in Kapitel 5.1 durch Abbildung 5.1.

5 Bausteinsicht

Die nachfolgend enthaltenen Diagramme der Bausteinsichten geben den Datenfluss als Kontrollfluss an.

5.1 Bausteinsicht Ebene 0

Ebene 0 der Bausteinsicht (siehe Abbildung 5.1) zeigt LigMaTrack in dessen Systemkontext. Das System bietet eine Bedienschnittstelle, über die Nutzer mit der WebApp interagieren können.

Bestimmte Daten erhält die WebApp über eine *Websocket*-Schnittstelle mithilfe derer es mit einem externen System kommuniziert. Bei diesen Daten handelt es sich um GPS-Positionsdaten von Ladungsträgern. Das Fremdsystem ist der *Ligenium Data Pool*. In einer finalen Umsetzung wird er Teil der von ligenium betriebenen *Verwaltungsschale* sein. Für den Demonstrator wird der Lignenium Data Pool simuliert.

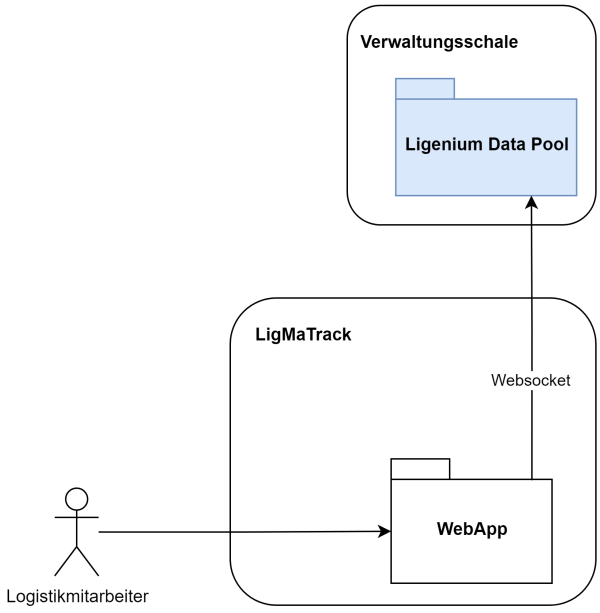

Abbildung 5.1 – Bausteinsicht Ebene 0

5.2 Bausteinsicht Ebene 1

Bausteinsicht Ebene 1 (siehe Abbildung 6.2) geht genauer auf den Aufbau der in Ebene 0 (vgl. Abbildung 6.1) gezeigten WebApp ein. Wie bereits in Kapitel 4 erläutert, unterteilt sich das System dabei in drei Schichten, einer Ebene für das Benutzerinterface, einer Ebene für die Business-Logik und einer Ebene für die Datenpersistierung.

Das Benutzerinterface ist eine eigenständige Komponente und macht den Client-Anteil bzw. das Frontend der Client-Server-Architektur aus. Sie wird in der Form einer React-Anwendung umgesetzt und kommuniziert mit dem Backend über REST und Websockets. Für den Demonstrator erfolgt die Anbindung an den externen Ligenium Data Pool direkt im Frontend. Es baut eine Websocket-Verbindung zur simulierten Komponente auf. Für die finale Architektur ist eine Websocket-Verbindung vom Frontend an das Backend geplant. Letzteres handhabt wiederum intern über eine Webschnittstelle die Anbindung an das externe System.

Der Server-Anteil besteht dagegen aus zwei Komponenten, dem Backend und der Datenpersistierungskomponente. Das Backend enthält alle Sub-Komponenten, die für die Ausführung der Business-Logik und letztendlich für die Beantwortung der Anfragen des Frontends benötigt werden.

Die Datenpersistierungskomponente setzt für die Speicherung von Daten auf eine NoSQL-Datenbank. Dabei handelt es sich um eine MongoDB. Die Kommunikation bzw. der Datenaustausch der beiden Server-Komponenten erfolgt über einen MongoDB Driver. Dieser wird durch Node.js und den MongoDB Node Driver realisiert. Letzterer nutzt intern Express, ein Framework für die Erstellung RESTful Webservices, sodass eine Kommunikation zwischen den beiden Containern ermöglicht wird.

Frontend und Backend können auf dem Zielsystem direkt installiert und ausgeführt werden. Die Datenbank läuft in einer eigenen Umgebung, welche mit Hilfe eines *Docker Containers* umgesetzt wird.

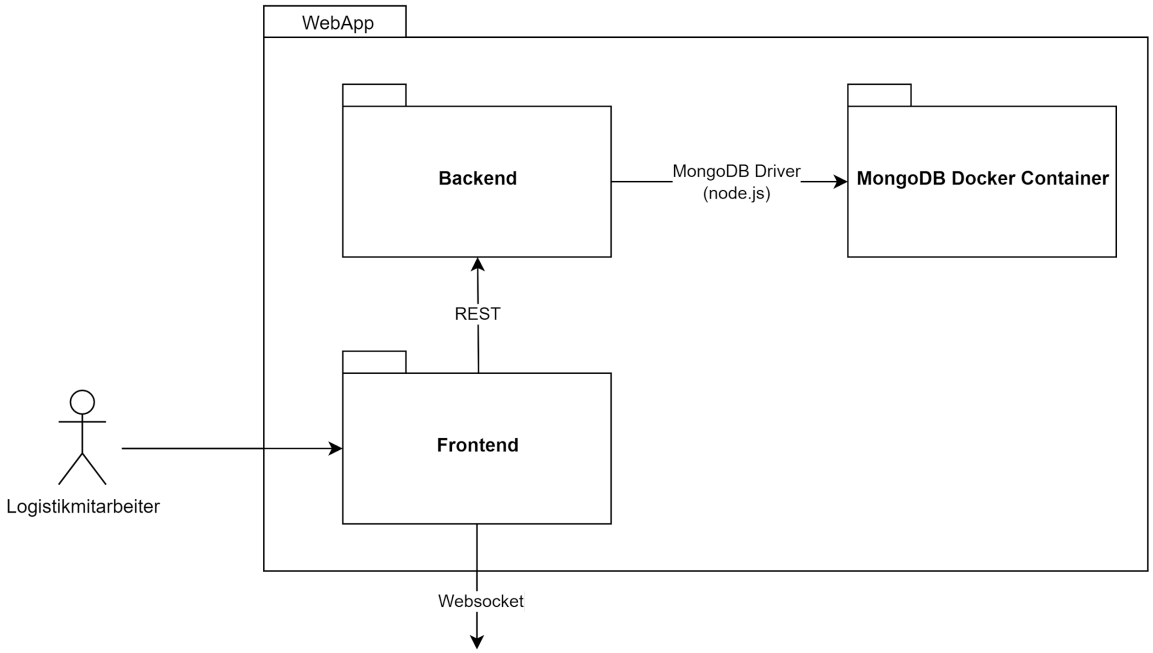

Abbildung 5.2 – Bausteinsicht Ebene 1

5.3 Bausteinsicht Ebene 2

Bausteinsicht Ebene 2 (siehe Abbildung 5.3) beleuchtet das in Kapitel 5.2 beschriebene Backend des Systems. Das Frontend- sowie das Persistenzmodul werden dagegen nicht näher betrachtet, da es sich um Standardimplementierungen handelt. Es entstünde kein Mehrwert, die dort enthaltenen Komponenten näher zu behandeln.

Das Backend besteht aus drei Bausteinen, dem frontend-com, dem backend-controller und dem data-controller. Die Aufteilung in die verschiedenen Komponenten erfolgt, um eine Aufgabenteilung (Separation of Concerns) zu erreichen. Jede Komponente hat einen Zuständigkeitsbereich, der anderen Komponenten per Schnittstelle bereitgestellt wird. Dadurch müssen Funktionen nicht mehrfach implementiert werden, Komponenten können einfach überarbeitet und erweitert werden und die Fehlersuche wird vereinfacht.

Frontend-Com ist für die Kommunikation mit dem Frontend zuständig. Es stellt die nötigen Endpunkte über REST bereit. Das Frontend kann für Anfragen GET- und POST-Methoden verwenden, um Daten zu erhalten und um von Nutzern durchgeführte Änderungen zur Verarbeitung und Speicherung weiterzureichen.

Die Ausführung der Business-Logik findet in Services statt. Aufgrund der limitierten Funktionalität des Demonstrators gibt es dazu nur den backend-controller. Dieser führt alle Verarbeitungsaufgaben durch. Für die finale Architektur wird es untergliederte Services geben, die spezifische Aufgaben übernehmen. Der backend-controller besitzt dort die Aufgabe Anfragen an den zuständigen Service weiterzureichen.

Die Kommunikation mit dem externen Ligenium Data Pool wird in der finalen Architektur von einer Komponente im Backend übernommen. Für den Demonstrator baut das Frontend direkt eine Websocket-Verbindung zur simulierten Datenquelle auf. Daher gibt es in der Demonstratorarchitektur des Backends keine Komponente für diese Aufgabe.

Der data-controller handhabt alle Datenanfragen der Dienste des Backends. In der finalen Architektur ist seine Aufgabe zu entscheiden, ob Daten über die lokale Datenbank verfügbar sind oder von einem externen System angefragt werden müssen und ob externe Daten anschließend in der lokalen Datenbank zwischengespeichert werden. Die Beschaffung der Daten wird also von den Services entkoppelt sein. Sie fragen Daten an und erhalten diese anschließend, unabhängig davon, ob die Quelle ein externes System ist oder es sich Datenpersistenzkomponente handelt. Für den den Demonstrator nutzt der data-controller ausschließlich die Datenpersistenzkomponente. Das genutzte externe System ist wie besprochen direkt an das Frontend gebunden. Der data-controller nutzt zur Persistierung eine MongoDB Datenbank und spricht sie über den MongoDB Node Driver an. Wie bereits beschrieben (vgl. Kapitel 5.2) erfolgt diese Kommunikation mit gängigen Web-Kommunikationsprotokollen. Die Datenbank selbst ist nicht Teil des Backends und läuft als gesonderte Komponente in einer separaten Docker-Umgebung.

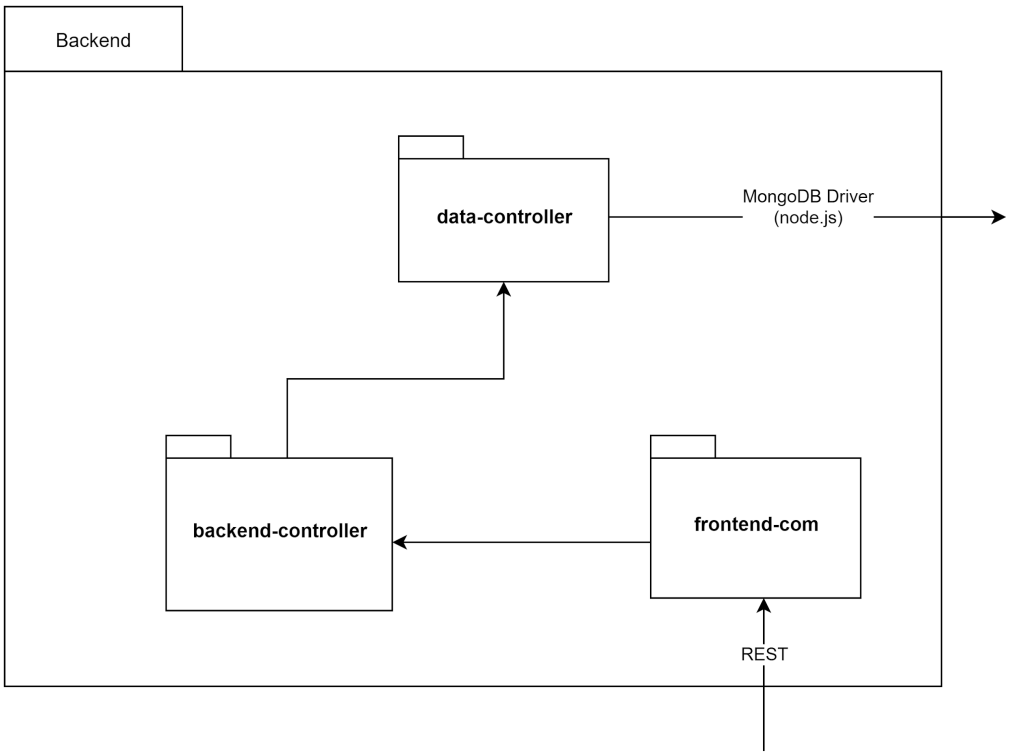

Abbildung 5.3 – Bausteinsicht Ebene 2 (Backend)

6 Laufzeitsicht

In der Anwendung gibt es zwei wesentliche Prozesse, die sich in abgewandelter Form immer wieder wiederholen; Die Kommunikation mit der Datenbank, um die vorhandenen Auftragsdaten abzufragen (Abbildung 6.1) und die Kommunikation mit dem Ligenium Data Pool, um u. a. Live-Standortdaten der Ladungsträger über Websockets zu erhalten (Abbildung 6.2 und 6.3). Es gibt noch weitere Prozesse, wie den Status eines Ladungsträgers als “Beschädigt” zu markieren, die im Folgenden ebenfalls beschrieben werden.

6.1 Alle Aufträge abfragen

Das folgende Diagramm zeigt den Abfrageprozess aller Aufträge.

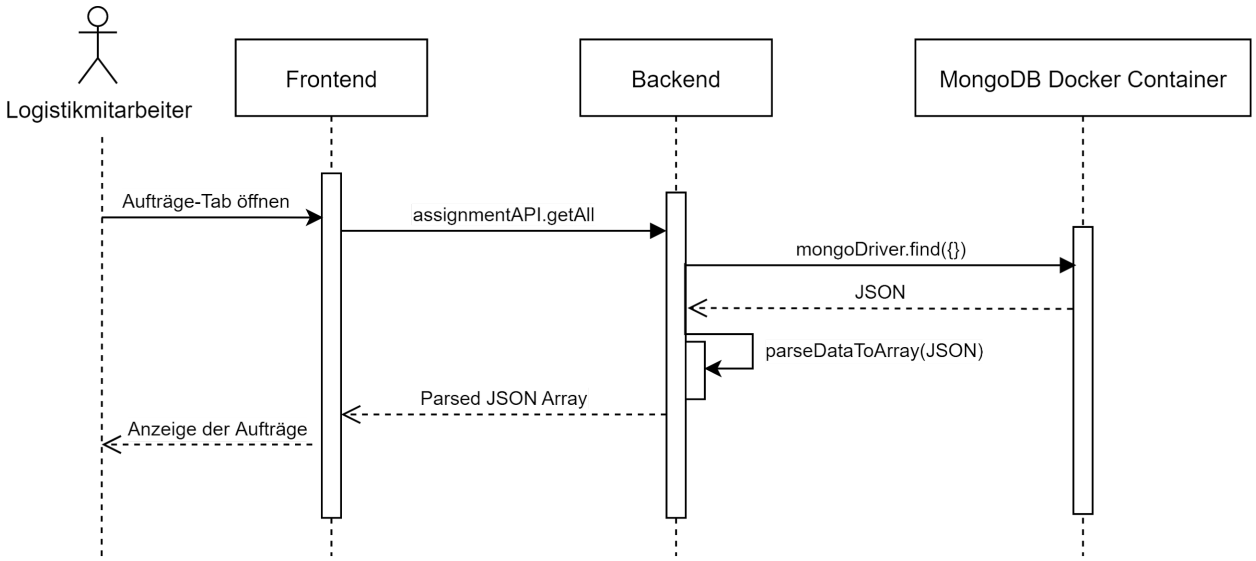

Abbildung 6.1 – Laufzeitsicht Abfrageprozess aller Aufträge

Durch eine Benutzerinteraktion auf den Aufträge-Tab über die Navigationsleiste der WebApp wird zuerst eine Datenbankabfrage an das Backend gestellt, um alle Aufträge zu erhalten. Nachdem die Informationen aus der Datenbank entnommen wurden, werden die Aufträge mit den jeweiligen Daten als *JSON* Array dem Frontend zurückgesendet und dem Benutzer angezeigt.

6.2 Auftragsdetails abfragen & Ladungsträger Websocket Initialisierung

Die folgende Abbildung zeigt den Abfrageprozess der Auftragsdetails bis hin zur Initialisierung der Websocket-Verbindung zwischen Frontend und dem für den Demonstrator simulierten Ligenium Data Pool.

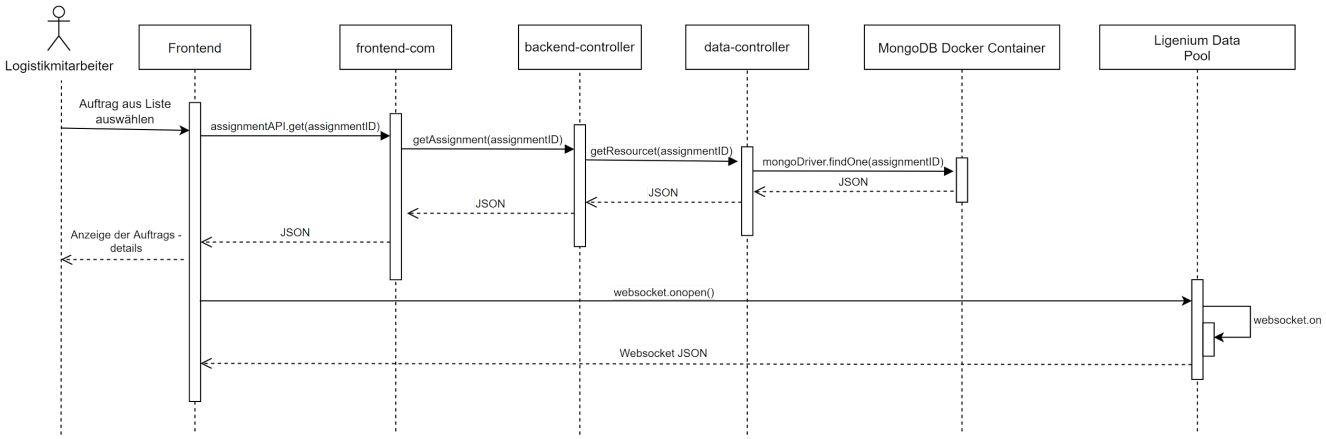

Abbildung 6.2 – Laufzeitsicht Abfrageprozess Auftragsdetails

Nachdem alle Aufträge geladen wurden (siehe Abbildung 6.1) und der Nutzer einen der Aufträge in der WebApp auswählt, werden anhand der eindeutigen Auftrags-ID die jeweiligen Auftragsdetails abgefragt und im Anschluss daran dem Nutzer angezeigt.

Anschließend wird eine Websocket-Verbindung vom Frontend zum Ligenium Data Pool hergestellt und initialisiert. Zuletzt werden nun jegliche Änderungen der Standortdaten der Ladungsträger über die Websocket-Verbindung gesendet (siehe Abbildung 6.3).

Sobald der Benutzer die WebApp komplett schließt oder sich auf einen anderen Tab über die Navigationsleiste begibt, wird die offene Websocket-Verbindung geschlossen.

6.3 Ladungsträger Standortdaten aktualisieren

Das folgende Diagramm zeigt den Aktualisierungsprozess der Standortdaten einzelner Ladungsträger eines ausgewählten Auftrags über den Websocket, um diese dem Nutzer im Frontend anzuzeigen.

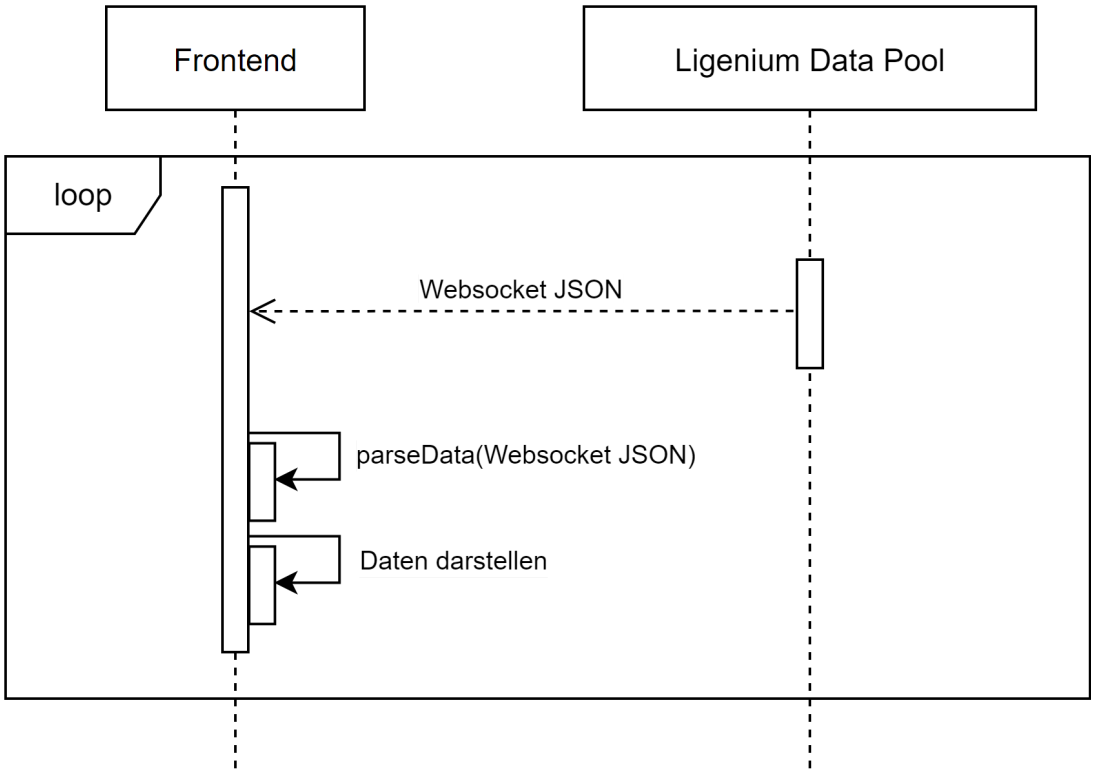

Abbildung 6.3 – Laufzeitsicht Aktualisieren der Standortdaten

Nach der Initialisierung der Websocket-Verbindung zwischen Frontend und dem Ligenium Data Pool, zu sehen in Abbildung 6.2, sendet der für den Demonstrator simulierte Ligenium Data Pool jegliche Änderungen der Standortdaten an das Frontend. Zur Vereinfachung sendet er hierbei die Standortdaten aller im Demonstrator hinterlegten Ladungsträger. Diese werden somit noch einmal im Frontend umgewandelt und gefiltert, sodass letztendlich nur die Ladungsträger des ausgewählten Auftrags auf der Karte dargestellt werden.

6.4 Ladungsträger eines ausgewählten Auftrags als beschädigt markieren

Das folgende Diagramm zeigt den Aktualisierungsprozess des Ladungsträger-Status eines ausgewählten Auftrags über REST.

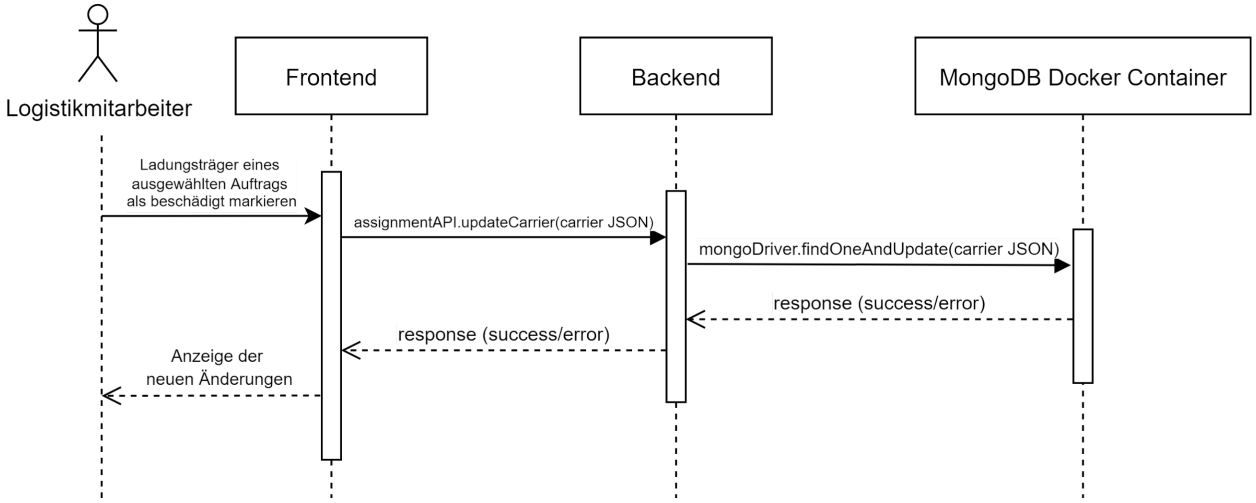

Abbildung 6.4 – Laufzeitsicht Aktualisierungsprozess des Ladungsträgerstatus

Durch die Benutzerinteraktion auf den “Als beschädigt markieren”-Button zu einem bestimmten Ladungsträger wird eine REST-Anfrage vom Frontend an das Backend geschickt, um den Status zu aktualisieren.

Nachdem die Informationen in der Datenbank erfolgreich geändert wurden, werden dem Nutzer die jeweiligen Änderungen im Frontend angezeigt.

7 Verteilungssicht

Die folgende Abbildung zeigt die Verteilungssicht des Demonstrators.

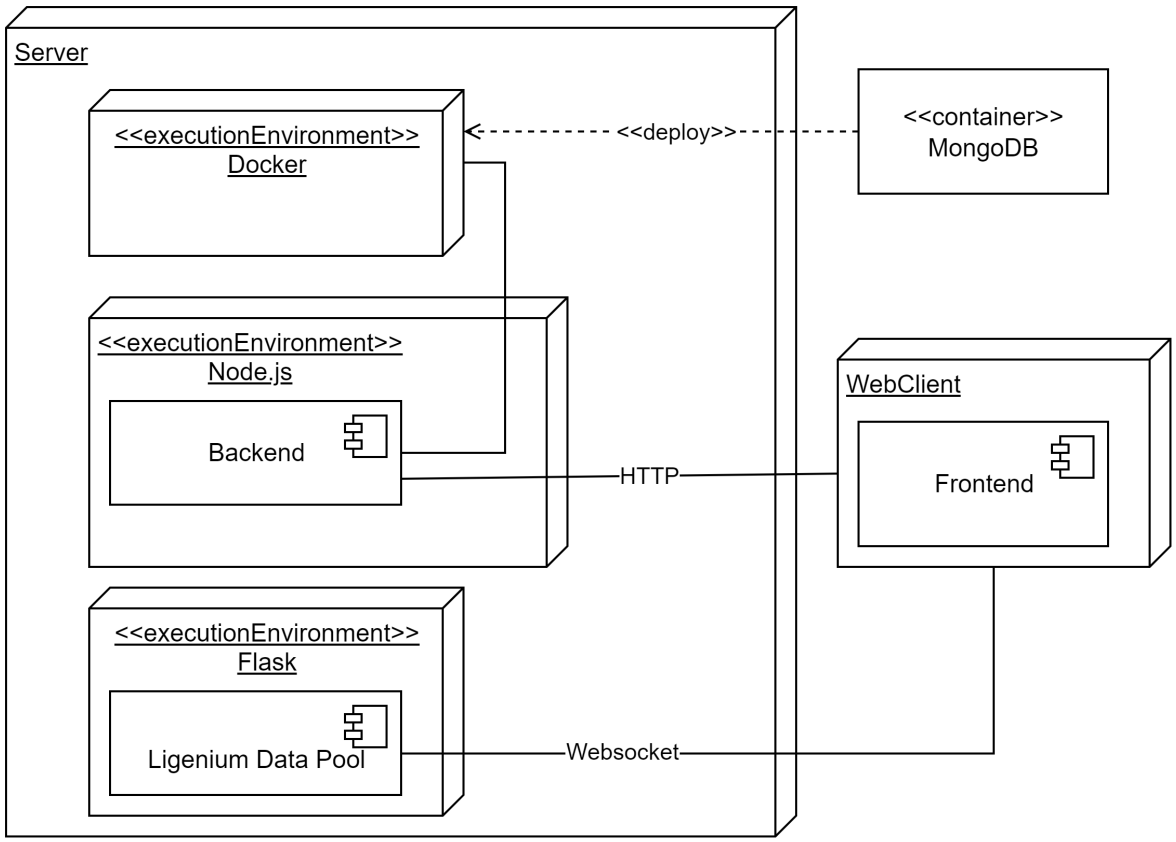

Abbildung 7.1 – Verteilungssicht des Demonstrators

Auf dem Server läuft eine Docker Instanz für die MongoDB Datenbank, ein Flask Webserver für den Ligenium Data Pool und eine Node.js Umgebung, die das Backend bereitstellt.

8 Entwurfsentscheidungen

Im Folgenden wird auf die Entwurfsentscheidungen des Demonstrators eingegangen. Für alle Entscheidungen gilt, dass die Erfahrung der Teammitglieder im Umgang mit besagter Technologie eine Rolle gespielt hat.

8.1 React WebApp

Mit React ist es möglich, die WebApp mit geringem Aufwand cross-platform-fähig und responsive zu gestalten. Es genügt eine Version der WebApp zu entwickeln. Diese läuft angepasst bspw. auf Tablets oder im Browser, wodurch nicht mehrere Versionen gewartet werden müssen und Entwicklungszeit gespart wird.
Das dynamische Rendering der WebApp mithilfe der leichtgewichtigen React-Bibliothek ist sehr performant. Da live Standortdaten der Ladungsträger angezeigt werden, bietet dies einen großen Vorteil. Da die React Applikation keinerlei Geschäftslogik enthält und ausschließlich die Daten vom Backend bezieht, entsteht eine lose Kopplung zwischen dem Frontend und Backend, welche einen weiteren Vorteil mit sich bringt. Da im Frontend von LigMaTrack keinerlei Geschäftslogik abläuft, könnte man es ohne Probleme durch ein Alternatives austauschen, welches z.B. in Vue.js oder Angular implementiert wurde.

8.2 Client/Server-Architektur

Die Entscheidung für eine *Client/Server-Architektur* fiel einfach, da das Produkt als WebApp realisiert wird. Hierbei übernimmt das Backend die Rolle des Servers und das Frontend die des Clients, da die WebApp an die Browser der Clients ausgeliefert und dort ausgeführt wird. Dadurch ist es möglich, Wartungs- bzw. Installationsaufwand für die Anwendung auf ein Minimum zu reduzieren, da die Datenhaltung auf einem zentralen Server erfolgt.

8.3 MongoDB-Datenbank

Der Vorteil einer NoSQL-Datenbank ist, dass man kein kompliziertes Datenbankschema erstellen muss. Die verwendeten Daten der WebApp befinden sich ausschließlich im JSON-Format, die direkt und unkompliziert in die Datenbank abgelegt werden können. Umgesetzt wird die NoSQL-Datenbank als MongoDB, die eine kostenfreie und performante Lösung ist. Als Alternative hätte auch eine Relationale Datenbank eingesetzt werden können, die im Falle des Demonstrators durch sein Entity-Relationship-Modell jedoch unnötig Komplexität aufgewiesen hätte.

8.4 Docker Container

Die Intention hinter der Nutzung eines Docker-Containers für die Datenbank des Demonstrators liegt darin, dass die Datenbank problemlos ohne Konfiguration und Installation von Software für die Entwicklung genutzt werden kann. Entwickler, die wenig Erfahrung im Aufsetzen einer MongoDB haben, können den Docker Container starten, zurücksetzen und löschen, um am

Frontend oder Backend mit “echten” Daten zu entwickeln. Die Alternative wäre die native Windows-Installation der Datenbank, die komplexer und umständlicher ist.

8.5 REST und Websockets

Der Austausch der Daten erfolgt u. a. zwischen Front- und Backend sowie Backend und Ligenium Data Pool. Dafür wird eine Kombination von REST und Websockets verwendet. REST wird verwendet, um einfach und simpel mit den Auftragsdaten zu arbeiten (CRUD). Websockets werden wiederum verwendet, um Live Standortdaten der Ladungsträger ohne dauerhaftes Polling abzufragen.

8.6 Node.js-Express-Backend

Als Backend wurde Node.js gewählt, um mit dem Frontend, Ligenium Data Pool und der Datenbank zu kommunizieren. Node.js ist weit verbreitet, bietet eine Vielzahl an Bibliotheken an, ist gut dokumentiert und hat eine große Community. Das Framework bietet einen einfachen Entwicklungseinstieg und ist sehr leichtgewichtig. Es eignet sich daher besonders für das schnelle Entwickeln eines Demonstrators. Als Alternative zu Express kann Nest.js gewählt werden, jedoch fehlt die nötige Erfahrung damit im Team.

8.7 Ligenium Data Pool

Für den simulierten Ligenium Data Pool Server des Demonstrators wird Flask als Backend-Technologie verwendet - als Alternative hätte genauso ein Node.js Backend eingesetzt werden können. Der Grund für die Verwendung von Python und Flask liegt darin, dass JavaScript nicht allen Backend Entwicklern bekannt ist. Flask bietet dieselben genannten Vorteile wie Node.js (siehe Kapitel 8.6).

9 Glossar und Abkürzungsverzeichnis

	Abkürzung	Begriff	Beschreibung
A	AaaS	Asset as a Service	Ein Service, welcher unter nutzungsbasierten Finanzierungskonzepten angeboten wird .
	API	Application Programming Interface	“Die Abkürzung API steht für Application Programming Interface und bezeichnet eine Programmierschnittstelle. Die Anbindung erfolgt auf Quelltext-Ebene. APIs kommen in vielen Anwendungen zum Einsatz und werden im Webumfeld in Form von Web-APIs genutzt.” (https://www.dev-insider.de/was-ist-eine-api-a-583923/) [14.06.2022]
B	-	Backend	“Als Backend wird der Teil eines IT-Systems bezeichnet, der sich mit der Datenverarbeitung im Hintergrund beschäftigt – der Data Layer. Der Begriff dient der Unterteilung bei komplexeren Softwarestrukturen.” (http://www.softselect.de/business-software-glossar/backend) [14.06.2022]
C	-	Client/Server-Architektur	“Die C/S-Architektur bezeichnet ein Systemdesign, bei dem die Verarbeitung einer Anwendung in zwei separate Teile aufgespaltet wird. Ein Teil läuft auf dem Server (Backend-Komponente), der andere Teil auf einer Workstation (Client oder Front-End).” (https://www.it-administrator.de/lexikon/client-server-architektur.html) [14.06.2022]
D	-	(Docker) Container	“Ein Container ist eine Standard-Softwareeinheit, die den Code und alle seine Abhängigkeiten zusammenfasst, damit die

			Anwendung schnell und zuverlässig in einer anderen Computerumgebung ausgeführt werden kann. Ein Docker-Container-Image ist ein leichtgewichtiges, eigenständiges, ausführbares Softwarepaket, das alles enthält, was zur Ausführung einer Anwendung erforderlich ist: Code, Laufzeit, Systemtools, Systembibliotheken und Einstellungen." (https://www.docker.com/resources/what-container/) [14.06.2022]
E	-	Express	"Express ist ein einfaches und flexibles Node.js-Framework von Webanwendungen, das zahlreiche leistungsfähige Features und Funktionen für Webanwendungen und mobile Anwendungen bereitstellt." (https://expressjs.com/de/) [14.06.2022]
F	FA	Funktionale Anforderung	"Funktionale Anforderungen spezifizieren, welche Funktionalität oder welches Verhalten das Softwareprodukt unter festgelegten Bedingungen besitzen bzw. erfüllen soll." (https://link.springer.com/content/pdf/10.1007%2F978-3-8274-2246-0_9.pdf) [14.06.2022]
	-	Frontend	"Das Frontend eines Softwareprogramms oder einer Website ist alles, mit dem der Benutzer interagiert." (https://techterms.com/definition/frontend) [14.06.2022]
I	-	Industrie 4.0	"Industrie 4.0 bezeichnet die intelligente Vernetzung von Maschinen und Abläufen in der Industrie mit Hilfe von Informations- und Kommunikationstechnologie."

			(https://www.plattform-i40.de/IP/Navigation/DE/Industrie40/WasIndustrie40/was-ist-industrie-40.html) [14.06.2022]
J	JSON	JavaScript Object Notation	“JSON ist ein Textformat zum Speichern und Übertragen von Daten.” (https://www.w3schools.com/js/js_json_intro.asp) [14.06.2022]
L	-	Ladungsträger	“Ein Ladungsträger ist ein Hilfsmittel aus der Logistik, welches es Ihnen ermöglicht mehrere Waren zu einer Ladeeinheit zusammenzufassen. Dazu gehören Paletten und Gitterboxen aber auch Container und individuelle Transportgestelle, sogenannte Sonderladungsträger.” (https://feil-systeme.de/ladungstraeger/) [14.06.2022]
	-	Ligenium Data Pool	Eine der möglichen Verwaltungsschalen, die am Ende ein externes System repräsentieren. Für das System LigMaTrack wird angenommen, dass die Positionsdaten der Ladungsträger, sowie die Daten der Aufträge hierüber bezogen werden.
	LigMaTrack	Ligenium Material Tracking	Die Endanwendung dieses Projekts.
M	-	MongoDB	“MongoDB ist ein nichtrelationales, auch NoSQL genanntes Datenbankmanagementsystem (DBMS). Es handelt sich um ein Software-Paket, das die Daten innerhalb einer Datenbank speichert, verarbeitet und verwaltet.” (https://www.ovhcloud.com/de/public-cloud/mongodb/) [14.06.2022]
N	-	Node.js	“Node.js ist eine asynchrone, ereignisgesteuerte JavaScript-Laufzeitumgebung, die für die Entwicklung skalierbarer

			Netzwerkanwendungen konzipiert ist.” (https://nodejs.org/en/about/) [14.06.2022]
	NoSQL	Not only SQL	“NoSQL-Datenbanken verwenden verschiedene Datenmodelle für den Zugriff auf und die Verwaltung von Daten. Diese Typen von Datenbanken sind speziell für Anwendungen optimiert, die große Datenmengen, geringe Latenz sowie flexible Datenmodelle erfordern.” (https://aws.amazon.com/de/nosql/) [14.06.2022]
O	op	objective partner AG	Eine der projektbeteiligten Firmen (siehe Kapitel 2.4).
P	PSE1	Projekt Software Engineering 1	Die Vorlesung im Master Studiengang Informatik der Hochschule Mannheim, in der die Anforderungsspezifikation für dieses Projekt entsteht.
	PSE2	Projekt Software Engineering 2	Die Vorlesung im Master Studiengang Informatik der Hochschule Mannheim, in der dieses Projekt entsteht.
R	-	React	“React ist eine JavaScript-Bibliothek zur Erstellung von Benutzeroberflächen.” (https://reactjs.org/) [14.06.2022] (Darüber hinaus: siehe Kapitel 9.1)
	REST	Representational State Transfer	“REST- (Representational State Transfer) bzw. RESTful-API ist ein Application-Program-Interface-Typ (API-Typ), der webbasierte Apps in der Kommunikation miteinander unterstützt.” (https://www.talend.com/de/resources/was-ist-rest-api/) [14.06.2022]
S	-	Sensor	Die jeweilige Technik, die in diesem Projekt für die Positionsbestimmung der Ladungsträger eingesetzt wird.

	SS	Sommersemester	-
V	-	Verwaltungsschale	“Die Verwaltungsschale ist ein herstellerübergreifender und branchenneutraler Standard für die Bereitstellung von Informationen und für die Kommunikation in einheitlicher Sprache. Jedes „Ding“ im „Internet der Dinge“ (IoT) erhält eine eigene Verwaltungsschale. [...] Jedes Asset, wie beispielsweise ein Gerät oder Bauteil, kann über seine eigene Verwaltungsschale weltweit identifiziert und angesprochen werden. Sie stellt Informationen über Eigenschaften und Fähigkeiten des Gegenstands bereit. Über ihre genormten Schnittstellen und eine einheitliche Sprache können die „Dinge“ miteinander kommunizieren.” (https://www.dke.de/de/arbeitsfelder/industry/verwaltungsschale) [14.06.2022]
W	WebApp	Webapplikation	“Web Apps sind Anwendungen, die über die Cloud bzw. einen Server bereitgestellt und im Browser beliebiger Endgeräte abgerufen werden.“ (https://www.dev-insider.de/was-ist-eine-web-app-a-596814/) [14.06.2022]
	-	Websocket	“WebSockets ist eine fortschrittliche Technologie welche es möglich macht eine interaktive Kommunikations-Session zwischen dem Browser des Benutzers und dem Server herzustellen. Mit dieser API können Sie Nachrichten zum Server senden und ereignisorientierte Antworten erhalten ohne beim Server die Antwort abzufragen.” (https://developer.mozilla.org/de/docs/Web/API/WebSockets_API) [14.06.2022]