Architekturdokumentation

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 Softwaresystems *LigMaTrack* und zukünftig geplante Features, welches im Rahmen des Kurses Projekt Software Engineering 2 (*PSE2*) an der Hochschule Mannheim durch das Team “Crunchtime Wonders” entwickelt und implementiert wird.

Hinter dem Projekt steht der Auftraggeber objective partner AG (*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

Dieses Kapitel enthält die Aufgabenstellung, die abgedeckten Anforderungen wie auch zukünftig geplante Funktionalitäten und deren Anforderungen vom Softwaresystem LigMaTrack. Dazu stellt es die Qualitätsziele des Systems vor.

2.1 Aufgabenstellung

Die Nutzer des Softwaresystems LigMaTrack sind die Logistikmitarbeiter. Die Logistikmitarbeiter 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 ligenium Ladungsträger über Standortdaten per Software existiert bisher noch nicht.

Das System LigMaTrack ermöglicht das Auffinden der Ladungsträger eines Auftrags im Lager. Die Standorte der Ladungsträger werden auf einem digitalen Lagerplan angezeigt, sodass diese leicht lokalisiert werden können. Diese verbesserte Auffindbarkeit der Ladungsträger 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.

Als nächstes folgt die Vorstellung der in Zukunft geplanten Funktionalitäten bzw. Erweiterungen vom Softwaresystem LigMaTrack. Diese sind folgende:

Geplante Funktionalität	**Hintergrund**
Suchen, Filtern, Sortieren von Aufträgen	Der Logistikmitarbeiter soll in der Lage sein, die Aufträge zu suchen, zu filtern und ggf. zu sortieren. Mit Hilfe der Suche kann der Logistikmitarbeiter nach Aufträgen suchen. Dazu soll es möglich sein, die Aufträge zu sortieren. Ein Beispiel hierfür ist die Sortierung nach dem Liefertermin oder Bearbeitungstermin. Je näher der Termin desto dringlicher die Ausführung des Auftrags. Mittels eines Filters können die Aufträge nach unterschiedlichen Status angezeigt werden. So kann sich der Logistikmitarbeiter oder ggf. Logistikleiter jederzeit nur die abgeschlossenen Aufträge auflisten lassen.
Anzeigen der (optimalen) Route zur Destination auf der Karte	Sobald ein Auftrag ausgewählt wurde, soll das System dem Logistikmitarbeiter im Lagerplan Routen zu den erforderlichen Ladungsträgern und zum Zielort darstellen. Optimal bezieht sich auf eine kurze Wegstrecke. Möglichst kurze Wegstrecken sollen anfallen. Die Route ist optimal, wenn weniger Zeit für die Bearbeitung des Auftrages bzw. der Transport des Ladungsträgers benötigt wird. Der Logistikmitarbeiter kann die vorgegeben Routen nutzen. Abweichungen von der Route würden durch eine spätere Betrachtung und Analyse der Bewegungshistorie gesehen. Vielleicht hat die abgewichene Route eine kürzere Wegstrecke. Ein möglicher Lösungsansatz zur Umsetzung der (optimalen) Route sowie Aufwandsabschätzung oder evtl. Anfallende Technologieauswahl steht aus.
Abruf der Bewegungshistorie des Ladungsträgers	Mittels mit Sensoren ausgestatteten Ladungsträger werden Positionsdaten gesendet. Diese können in Zukunft verwendet werden, um eine

	Bewegungshistorie zu erstellen. Diese Historie kann die im Auftrag bewegten Ladungsträger oder die Ladungsträger allgemein enthalten. Im Falle des Systems LigMaTrack die im Auftrag bewegten Ladungsträger. Mittels der Daten kann die Firma eine Analyse durchführen, um möglicherweise eine Lageroptimierung durchzuführen. Ladungsträger mit viel verwendeten Produkten können näher am entsprechenden Zielort eingelagert werden.
Anzeige von Ladungsträger-Optionen (falls mehrere Ladungsträger den gleichen erforderlichen Inhalt besitzen)	Mehrere Ladungsträger können dieselben Artikel enthalten. Der Logistikmitarbeiter bekommt diese Ladungsträger angezeigt und kann sich einen von beiden mit der erforderlichen Stückzahl aussuchen.
Übersicht aller im Lager befindlichen Ladungsträger auf dem Lagerplan	Das System LigMaTrack zeigt einen Auftragsunabhängigen Lagerplan mit aller im Lager befindlichen Ladungsträger an.
Automatisches Abschließen von Aufträgen, wenn alle Ladungsträger am Zielstandort angekommen sind	Haben alle für den Auftrag erforderlichen Ladungsträger den vorgesehenen Zielstandort erreicht, erkennt das System dies und schließt den Auftrag mit der Setzung des Status auf “Abgeschlossen” ab.

Tabelle 2.1 – Systemkontextbeschreibung der geplanten funktionalen Anforderungen

2.2 Übersicht der funktionalen Anforderungen

Dieses Kapitel stellt die Anforderungen des Systems LigMaTrack vor. Zwei Unterscheidungen werden vorgenommen. In der ersten Tabelle stehen jene Anforderungen, welche von LigMaTrack abgedeckt werden. Die zweite Tabelle enthält jene Anforderungen, welche in Zukunft berücksichtigt werden, wenn das System weiterentwickelt wird. Diese Funktionalitäten stehen in Tabelle 2.1.

Die aufgezählten Anforderungen 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 op. Darin wird das Produkt “Ligenium Material Tracking” (LigMaTrack). Zu diesem sind Anforderungen sowie Qualitätsanforderungen beschrieben und festgehalten.

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

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 2.2 – Systemkontextbeschreibung der umgesetzten FA

Mit einer zukünftigen Weiterentwicklung von LigMaTrack werden weitere Anforderungen aus der Anforderungsspezifikation im System berücksichtigt und adressiert. Diese Anforderungen stehen in der nachfolgenden Tabelle, welche jene Anforderungen unter Berücksichtigung der Umgebung des Systems LigMaTrack beschreibt.

FA-Nr.	Beschreibung	Systemkontextbeschreibung
6	LigMaTrack muss eine optimierte Route für die Kommissionierung bereitstellen.	Die Umsetzung der Funktionalität, der Darstellung der (optimalen) Route zur Destination auf dem Lagerplan, enthält die Bereitstellung einer Route. Diese Route kann zuvor vom System errechnet werden, um den schnellstmöglichen Weg zum Ladungsträger herauszufinden. Die Transportzeit soll durch die Routen weniger Zeit in Anspruch nehmen. Als Ausgangsbasis der Daten kann der hinterlegte Lagerplatz im System oder die Positionsdaten verwendet werden. Während der Bearbeitung eines Auftrags kann der Logistikmitarbeiter die vorgegebene Route nutzen, um die Ladungsträger vom Ausgangspunkt zur Destination zu transportieren. Die Nutzung der Route durch den Logistikmitarbeiter soll die in Anspruch genommene Zeit für den Transport reduzieren.
7	LigMaTrack muss einen Lagerplan mit den Standorten von Ladungsträgern für die Optimierung des Lagers erstellen.	Das System LigMaTrack soll in Zukunft während dem Transport der Ladungsträger die Positionsdaten sammeln, um den Abruf einer Bewegungshistorie der Ladungsträger zu ermöglichen. Diese Bewegungshistorie enthält den verwendeten Weg während der Auftragsbearbeitung. Das Unternehmen kann einen abgeschlossenen Auftrag auswählen, um sich die Bewegungshistorie der Ladungsträger auf einer Repräsentation des Lagers anzeigen zu lassen. Diese Repräsentation stellt die gelaufenen Routen dar. Auf Basis dieser Bewegungshistorie, der vorgeschlagenen Route und der Inhalt der Ladungsträger kann eine Standortoptimierung sowie evtl. eine Routenoptimierung vorgenommen werden.

		Mittels LigMaTrack soll es unter anderem möglich sein, alle Ladungsträger auf dem Lagerplan einzusehen. Dadurch kann u. a. die Auslastung des Lagers erkannt werden, da Lagerstandorte mit wenig oder vielen Ladungsträger angezeigt werden. Auf Basis dieser Daten können Standortoptimierungen durchgeführt werden.

Tabelle 2.3 – Systemkontextbeschreibung der funktionalen Anforderungen, die für die zukünftige Weiterentwicklung geplant sind

2.3 Qualitätsziele

Dieses Kapitel beinhaltet die berücksichtigten Qualitätsziele innerhalb des Systems LigMaTrack.

Genauere Informationen zu den genannten nichtfunktionalen Anforderungen (*NFA*) befinden sich in der Anforderungsspezifikation Binary Riot, Kapitel 3.4.

NFA-ID	Qualitätsziel	Motivation/Erläuterung
2	Die Plattform muss für die Logistik- und Produktionsarbeiter intuitiv und ohne Anleitungen zu bedienen sein. (Usability and Humanity Requirements)	Die Logistikmitarbeiter entscheiden bereits bei der ersten Nutzung, ob sie die Applikation gut oder schlecht finden. Um zu prüfen, ob das Softwaresystem LigMaTrack intuitiv zu bedienen ist, empfiehlt es sich, dass ligenium oder verschiedene Logistikmitarbeiter das System bedienen.
3	Jede Interaktion zwischen einem Benutzer und dem automatisierten System muss eine maximale Reaktionszeit von 3 Sekunden (nicht länger als 1 Sekunde) haben. (Performance Requirements)	Das System LigMaTrack wird innerhalb des Lagers eingesetzt und soll dem Logistikmitarbeiter bei seiner Arbeit unterstützen und den Arbeitsablauf nicht stören bzw. verzögern. Daher muss das System in nahezu Echtzeit auf Funktionsaufrufe reagieren.
4	Die Technik der Positionsbestimmung muss auf einen Meter genau sein.	Damit der Logistikmitarbeiter die erforderlichen Ladungsträger des Auftrages findet, ist eine einwandfreie

	(Performance Requirements)	Bestimmung der Position erforderlich.
5	Die Route muss in nahezu Echtzeit aktualisiert werden. (Performance Requirements)	Die Aktualisierung in nahezu Echtzeit bedeutet, dass die Position der betroffenen Ladungsträger innerhalb einer Sekunde aktualisiert werden. Das Softwaresystem LigMaTrack hat einen physischen Prototypen. Innerhalb dessen können Ladungsträger bewegt werden. Deren Bewegung wird im Lagerplan angezeigt. Eine Erprobung und Analyse möglicher Umsetzungen in einem realen Lager steht aus.
6	Das Produkt muss mit den aktuellsten Versionen der drei meist genutzten Browser funktionieren. (Operational and Environmental Requirements)	Da das System auf Scanner und Desktop-Rechner laufen soll, können auf diesen Geräten unterschiedliche Browser existieren. Daher muss das System auf den drei gängigsten Browsern (Chrome, Safari, Edge) laufen.
8	Es müssen Benutzerrollen vorhanden sein. (Security Requirements)	Das System muss über verschiedene Nutzerrollen verfügen, um die Funktionen einzelner Endanwender voneinander abzuschirmen. Um diese NFA zu erfüllen, kann in Zukunft eine umfangreiche Benutzerverwaltung hinzugefügt werden.
11	Das Produkt soll auf unterschiedlichen Bildschirmgrößen verwendbar sein. (Operational and Environmental Requirements)	Die Logistikmitarbeiter nutzen mobile Scanner, welche andere Bildschirmgrößen haben. Daher muss das System auf unterschiedlichen Bildschirmgrößen verwendbar sein.
13	Das Produkt darf nur Ladungsträger der eigenen Organisation eines Endanwenders zeigen.	Eine Organisation A darf keine Einsicht auf die Ladungsträger der Organisation B erhalten. Eine Trennung erfolgt durch die

	(Security Requirements)	entsprechenden Schnittstellen vom System. An diesen können in Zukunft andere Komponenten, welche die Aufträge der Firma enthalten, angebunden werden.

Tabelle 2.4 – Auflistung der NFA des Systems LigMaTrack

2.4 Projektbeteiligte

In diesem Kapitel sind alle Projektbeteiligte gelistet, welche ein Interesse am Endergebnis haben.

Projektbeteiligte	**Interesse/Bezug**
Logistikmitarbeiter	Logistikmitarbeiter sind die Endnutzer der 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 mit dem Logistikmitarbeiter als Endnutzer für das System LigMaTrack 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. Das System übernimmt einen Teilbereich eines solchen AaaS-Angebots und orientiert sich dafür an 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 sowie die beiden Architekturdokumente für diesen und das Endprodukt.
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 2.5 – Projektbeteiligte mit Interesse an LigMaTrack

3 Randbedingungen

Für die Entwicklung des Softwaresystems wurden durch den Auftraggeber op keine Randbedingungen definiert und vorgegeben.

3.1 Technische Randbedingungen

Die technische Entwicklung kann in einer beliebigen, gängigen Programmiersprache umgesetzt werden. Der Software wurden ebenfalls keine technischen Randbedingungen in Bezug auf das Ausrollen, sowie das Installieren des Systems gesetzt und sind somit frei wählbar.

3.2 Organisatorische Randbedingungen

Randbedingung	Beschreibung
Organisation und Struktur	Das Team besteht aus Joshua Krcmar, Erik Martens, Tuncay Yildiz, Heike Welter und Shayan Mohajerani, unterstützt von Prof. Dr. Peter Knauber und Prof. Kirstin Kohler. Weitere Stakeholder sind dem Kapitel 2.4 zu entnehmen.
Zeitplan	Beginn der initialen Implementierung war der 01.05.2022. Die Endpräsentation und Vorführung des Demonstrators, entwickelt und aufbauend auf diesem Architekturdokument, erfolgt am 30.06.2022.
Organisatorische Standards	Zur Dokumentation der Architektur wird sich am arc42-Template (https://www.arc42.de/overview/) orientiert. Weiterhin wird die Software inkrementell entwickelt.
Entwicklungswerkzeuge	Die Software kann in eine beliebige, gängige Entwicklungsumgebung implementiert werden. Es sind keine Randbedingungen definiert.

Tabelle 3.1 – Organisatorische Randbedingungen

3.3 Konventionen

Randbedingung	Beschreibung
Sprache (Deutsch vs. Englisch)	Angewendet werden englische Bezeichnungen für die Benennung von Klassen, Methoden und Variablen im Code der Endanwendung LigMaTrack. Die Architekturdokumentation wird in deutscher Sprache verfasst.
Kodierrichtlinien	Es sind keine Kodierrichtlinien definiert worden.

Tabelle 3.2 – Konventionen

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 9.

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.

Das Frontend, Backend und die Datenpersistenz-Ebene laufen jeweils eigenständig in ihren eigenen *Docker*-Umgebungen ab und kommunizieren über Web-Kommunikationsprotokolle (u.A. *REST* und *Websockets*). Eine genaue Beschreibung erfolgt in Kapitel 6. Die Intention dahinter ist, das System unabhängig vom Zielsystem einsatzfähig zu machen und dem Endkunden die Flexibilität einer *Kubernetes*-Cluster-fähigen Anwendung zu bieten. Hierdurch sind Strategien, wie Lastverteilung, Skalierung, Rolling Updates und Continuous Deployment möglich.

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. Für einen zustandslosen Betrieb der Datenpersistenz-Komponente werden Netzwerk-Volumes eingesetzt.

5 Kontextabgrenzung

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 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.

Im Detail betrachtet handelt es sich bei den Fremdsystemem um *Verwaltungsschalen*. Eine Schale ist Teil einer verteilten Architektur. Sie beschreibt ein Persistenzsystem mit dessen Datenschema zur Speicherung von, sowie REST-basierte CRUD-Operationen für eine Datenart. Aktuell wird davon ausgegangen, dass die benötigten Verwaltungsschalen von ligenium gehostet werden und mithilfe des *Ligenium Data Pool* Systems bereitgestellt werden. Zum einen handelt es sich dabei um Positionsdaten, die ligenium von Ladungsträgern einsammelt. Der Ligenium Data Pool ist die Schale für Positionsdaten von Ladungsträgern in der Form von

[1] Technologien zum Umsetzen von Webanwendung, bestehend aus MongoDB, Express, React und Node.js (siehe Kapitel 9)

GPS-Koordinaten mit einem jeweiligen Zeitstempel. Zum anderen stellt der Ligenium Data Pool auch Lageraufträge bereit, d.h. Informationen über Ladungsträger mit deren Inhalten, die in einem Lager bspw. zum Warenausgang bewegt werden sollen. Der Datenaustausch mit Verwaltungsschalen findet über Webschnittstellen statt. Daher können sie beliebig ausgetauscht und nach ihren Aufgabenbereichen aufgesplittet werden.

Eine Abbildung des Systemkontexts findet sich in Kapitel 6.1.

6 Bausteinsicht

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

6.1 Bausteinsicht Ebene 0

Ebene 0 der Bausteinsicht (siehe Abbildung 6.1) zeigt LigMaTrack in dessen Systemkontext (vgl. Kapitel 5). Das System bietet eine Bedienschnittstelle, über die Nutzer mit der WebApp interagieren können.

Bestimmte Daten erhält die WebApp von einem externen System über eine Websocket-Schnittstelle und per REST. Dieses Fremdsystem ist der bereits beschriebene Ligenium Data Pool (vgl. Kapitel 5). Der Ligenium Data Pool liefert GPS-Positionsdaten von Ladungsträgern per Websocket-Verbindung. Außerdem erfolgt der Zugriff auf Auftragsdaten von der WebApp per REST-Schnittstelle.

Der Ligenium Data Pool ist Teil einer Verwaltungsschale. Wie bereits beschrieben können verschiedene Verwaltungsschalen für verschiedene Datenquellen existieren. Die Änderung der finalen Konfiguration ist durch die lose Kopplung mithilfe von Webschnittstellen in Zukunft einfach möglich, was eine Anforderung an das System ist. Datenquellen können also beliebig ausgetauscht werden. Die finale Konfiguration ist zu diesem Zeitpunkt jedoch unbekannt, daher wird aktuell von einem einzigen Ligenium Data Pool ausgegangen.

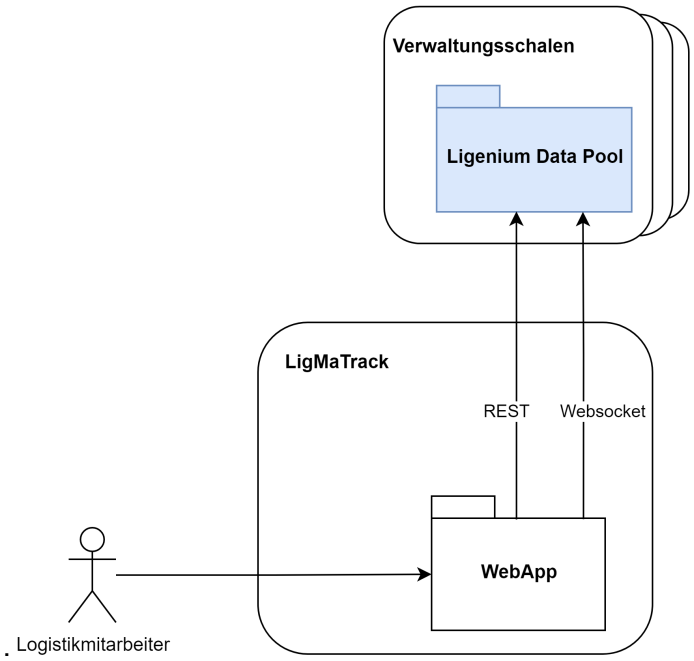

Abbildung 6.1 – Bausteinsicht Ebene 0

6.2 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. Alle Ebenen werden containerisiert, d.h. sie erhalten ihre eigene Docker-Ablaufumgebung und werden als Docker-Images bereitgestellt. Die Kommunikation zwischen den Ebenen findet über Web-Schnittstellen statt.

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.

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. Außerdem übernimmt es die Kommunikation mit dem Ligenium Data Pool über REST und Websockets. Das Backend wird mit Node.js und Express umgesetzt.

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.

Die Trennung von Business-Logik und Datenpersistenz erfolgt, damit das Backend zustandslos ist. Dadurch und durch die Containerisierung ist eine Last-Skalierung und ein Rolling-Update des Backends jederzeit möglich. Dabei sei erwähnt, dass selbiges auch für den Frontend-Container und Datenpersistenz-Container gilt. Letzterer ist zustandslos, da die tatsächliche Datenspeicherung auf einem Netzwerk-Volume erfolgt. Für die Umsetzung einer Last-Skalierung und einer Rolling-Update Strategie müssen die Container mithilfe eines Kubernetes-Clusters in Betrieb genommen werden. Ein Kubernetes-Cluster ist allerdings nicht zwingend notwendig, ein Deployment kann mit Docker auch direkt auf einem Docker-fähigen Zielsystem bzw. Zielrechner erfolgen.

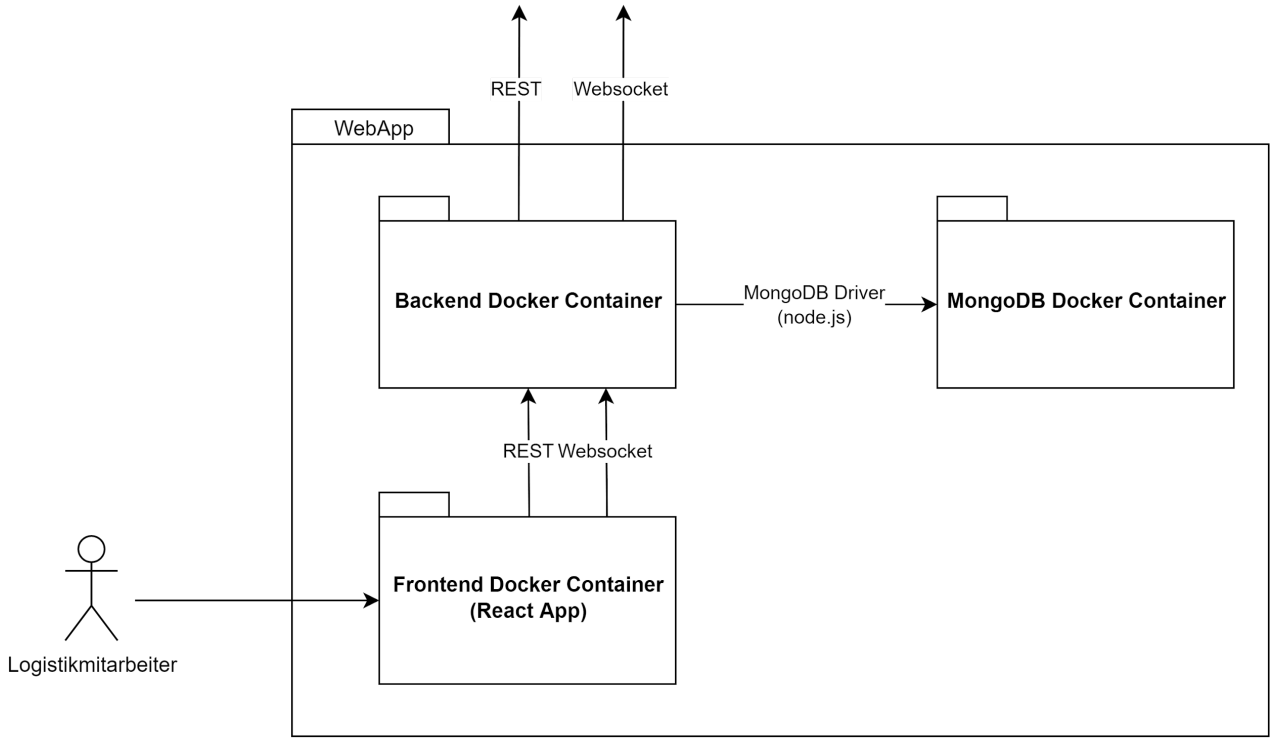

Abbildung 6.2 – Bausteinsicht Ebene 1

6.3 Bausteinsicht Ebene 2

Bausteinsicht Ebene 2 (siehe Abbildung 6.3) beleuchtet das in Kapitel 6.2 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 intern aus vier grundlegenden Bausteinen, dem frontend-com, dem backend-controller, dem data-pool-com und dem data-controller. Zusätzlich existieren drei Dienste bzw. Services, welche spezifische und logisch zusammengehörige Aufgaben der Business-Logik übernehmen. Hierbei handelt es sich um den user-service, den assignment-service und den load-carrier-navigation-service. 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.

Der frontend-com Baustein ist für die Kommunikation mit dem Frontend zuständig. Es stellt die nötigen Endpunkte über REST und Websockets 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. Für Daten, die laufend aktualisiert werden müssen, wird eine Websocket-Verbindung genutzt.

Die Ausführung der Business-Logik erfolgt durch die Services (user-service, assignment-service und load-carrier-navigation-service). Alle Aufgaben bezüglich der Benutzerverwaltung und -authentifizierung übernimmt der user-service. Die Beschaffung von Auftragsdaten, sowie die Bearbeitung dieser (bspw. das Als-Erledigt-Markieren eines Auftrags) führt der assignment-service aus. Im aktuellen Systemkonzept können alle Aufträge von allen Mitarbeitern angenommen werden, die sich erfolgreich am System anmelden können. Wird ein Auftrag bearbeitet, so wird dieser durch den assignment-service als gesperrt markiert. Der assignment-service speichert dazu eine Identifikationsnummer des betreffenden angemeldeten Mitarbeiters, welcher den Auftrag angenommen hat, mithilfe einer Anfrage an den user-service. Ladungsträger und deren Inhalte sind Teil von Aufträgen, d.h. die zugehörigen Daten sind Teil der Auftragsdaten. Unter Berufung auf die dort aufgeführte Identifikationsnummer eines Ladungsträgers können seine GPS-Positionsdaten abgerufen werden. Die Positionsabfrage erfolgt aus dem Frontend sobald ein Nutzer dort einen Ladungsträger auswählt und wird Backend-intern an den load-carrier-navigation-service delegiert. Die Daten werden fortlaufend über eine Websocket-Verbindung geliefert.

Die Verteilung der vom frontend-com angenommen Anfragen auf die Services erfolgt durch den backend-controller. Dieser kennt die Aufgabenbereiche der drei Dienste und führt die Zuweisung mithilfe von Methodenaufrufen durch. Die asynchrone Antwort der Dienste leitet er an die frontend-com Komponente, welche diese ebenfalls asynchron als Antwort auf die REST-Requests vom Frontend bzw. über eine bestehende Websocket-Verbindung an selbiges übermittelt. Benötigte Daten werden von den Diensten eigenständig über den data-controller abgerufen.

Die Kommunikation mit dem externen Ligenium Data Pool übernimmt data-pool-com mithilfe von REST und Websockets. Daten kann diese Komponente an den data-controller weitergeben, um eine potentielle Persistierung oder Zwischenspeicherung durchzuführen. Der data-controller kann die Daten auch direkt an die anfragende Komponente durchreichen. Dies wäre bspw. der load-carrier-navigation-service, der die fortlaufend aktualisierte GPS-Position eines Ladungsträgers benötigt.

Der data-controller handhabt alle Datenanfragen der Dienste des Backends. Sollten Daten nicht in der lokalen Datenbank verfügbar sein, so kümmert sich der data-controller um die Beschaffung dieser von einem externen System, im Kontext dieses Dokuments vom Ligenium Data Pool, indem er den data-pool-com beauftragt, die Daten anzufordern. Die Beschaffung der

Daten ist also von den Services entkoppelt. Sie fragen Daten an und erhalten diese anschließend, unabhängig davon, ob die Quelle ein externes System ist oder es sich Datenpersistenzkomponente handelt. Der data-controller verfügt außerdem über ein Entscheidungsmuster, welche Daten von externen Systemen für einen erneuten schnellen Lesezugriff zwischengespeichert werden. Der data-controller nutzt zur Persistierung eine MongoDB Datenbank und spricht sie über den MongoDB Node Driver an. Wie bereits beschrieben (vgl. Kapitel 6.2) 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.

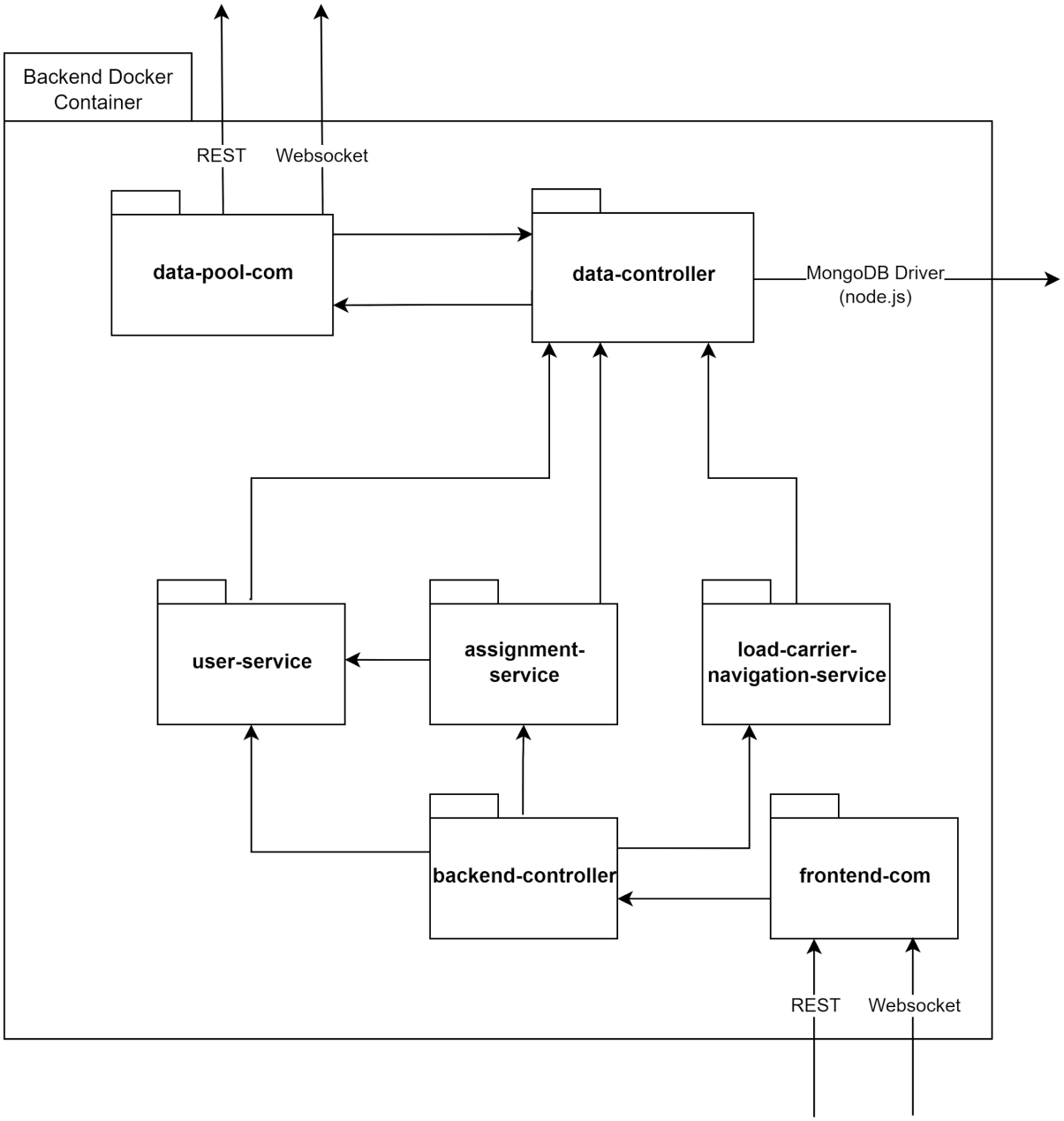

Abbildung 6.3 – Bausteinsicht Ebene 2 (Backend Docker Container)

7 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 7.1) und die Kommunikation mit dem Ligenium Data Pool, um u. a. Live-Standortdaten der Ladungsträger über Websockets zu erhalten (Abbildung 7.2 und 7.3). Es gibt noch weitere Prozesse, wie den Status eines Ladungsträgers als “Beschädigt” zu markieren, die im Folgenden ebenfalls beschrieben werden.

7.1 Alle Aufträge abfragen

Das folgende Diagramm zeigt den Abfrageprozess aller Aufträge.

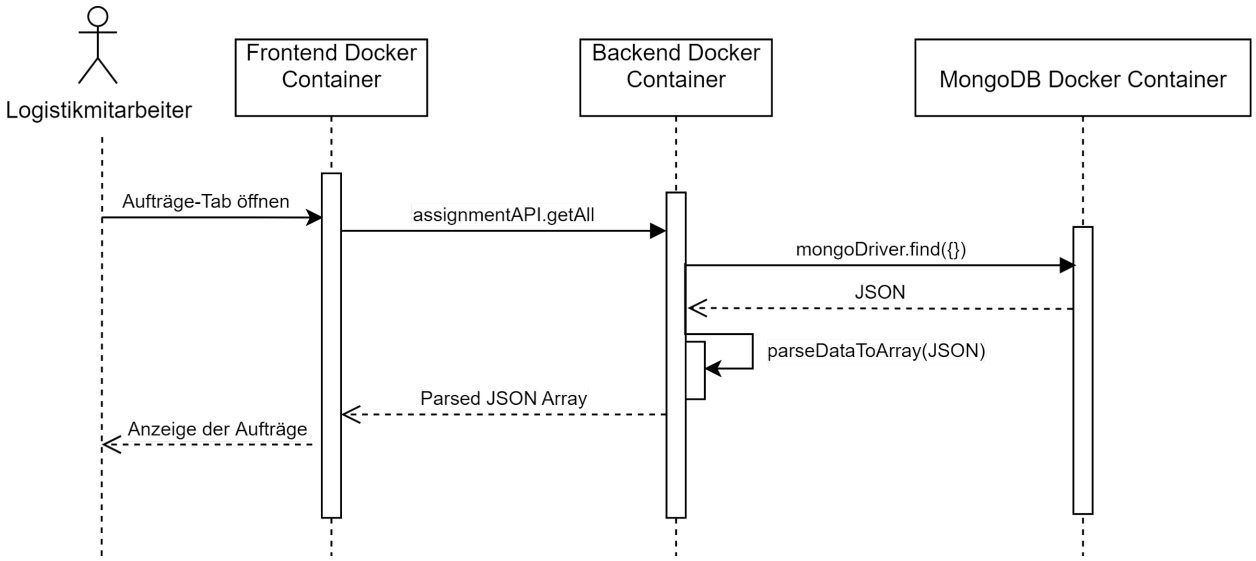

Abbildung 7.1 – Laufzeitsicht Abfrageprozess aller Aufträge

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.

7.2 Auftragsdetails abfragen & Ladungsträger Websocket Initialisierung

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

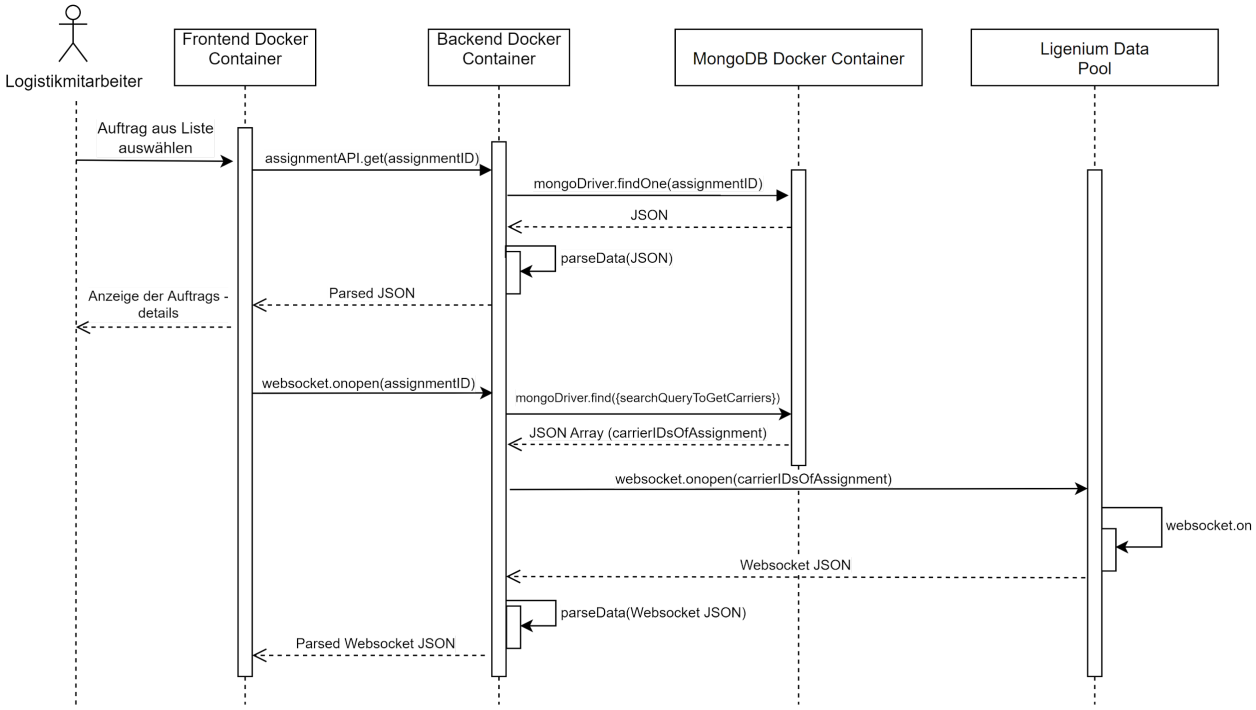

Abbildung 7.2 – Laufzeitsicht Abfrageprozess Auftragsdetails

Nachdem alle Aufträge geladen wurden (siehe Abbildung 7.1) 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 für die Ladungsträger des Auftrags eine Websocket-Verbindung vom Frontend mit dem Backend aufgebaut, welches seinerseits die Positionsdaten daraufhin über eine weitere Websocket-Verbindung mit dem Ligenium Data Pool bezieht.

Sobald der Benutzer die WebApp komplett schließt oder sich auf einen anderen Tab über die Navigationsleiste begibt, werden die beiden offenen Websocket-Verbindungen geschlossen.

7.3 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.

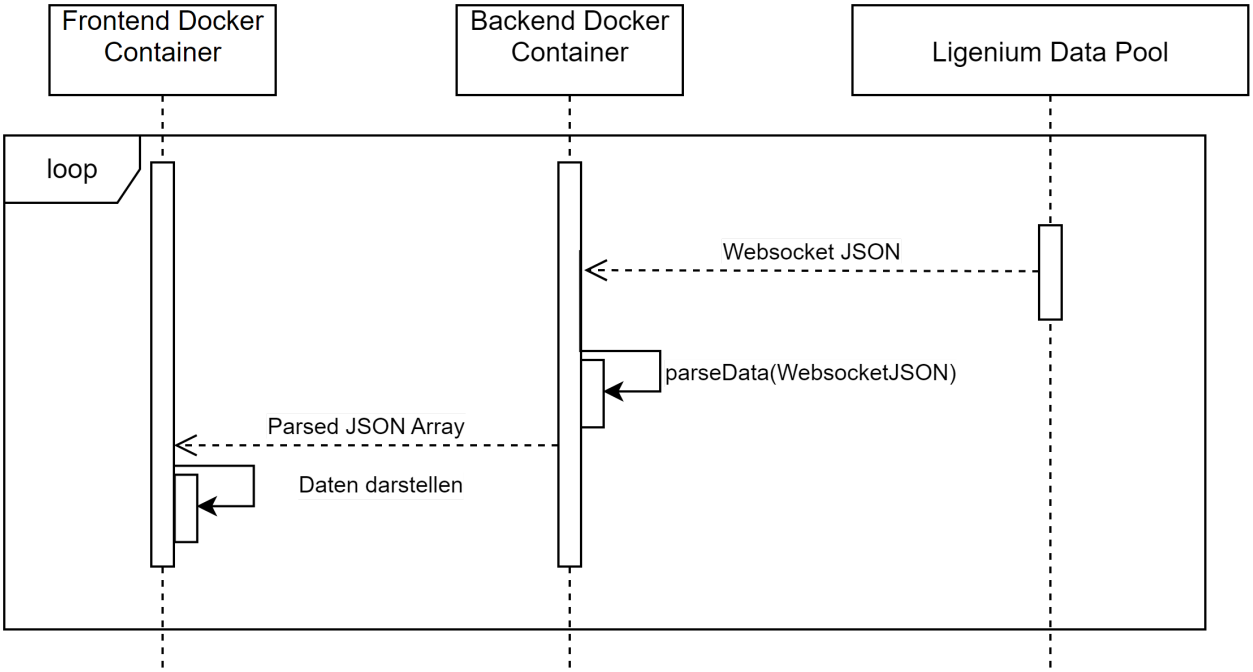

Abbildung 7.3 – Laufzeitsicht Aktualisieren der Standortdaten

Nach der Verbindung zwischen Frontend, Backend und Ligenium Data Pool, zu sehen in Abbildung 7.2, sendet das Backend jegliche Änderungen der Standortdaten an das Frontend.

7.4 Ladungsträger eines ausgewählten Auftrags als beschädigt markieren

Das folgende Diagramm zeigt den Aktualisierungsprozess des Ladungsträgerstatus eines ausgewählten Auftrags über REST.

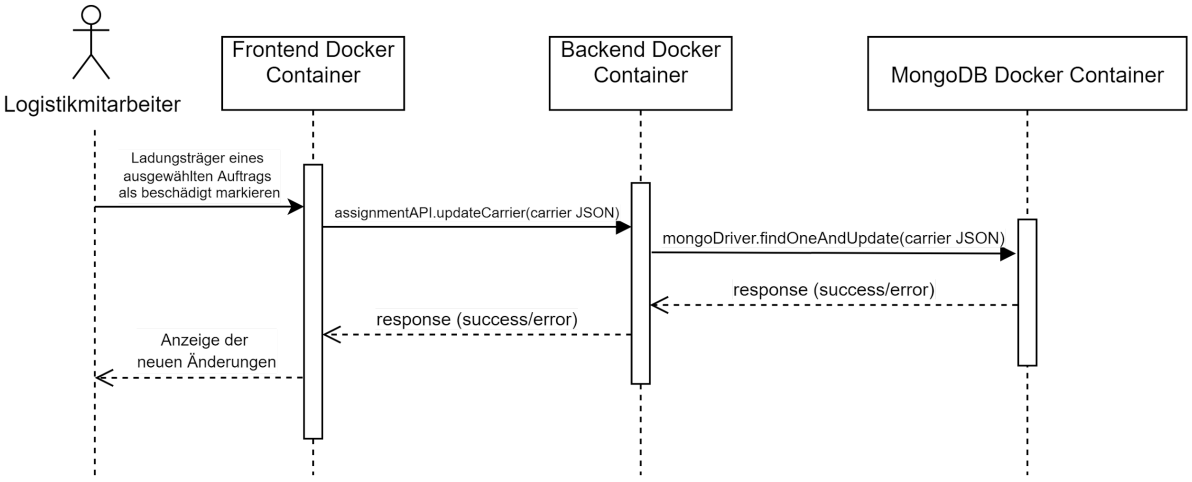

Abbildung 7.4 – Laufzeitsicht Aktualisierungsprozess des Ladungsträgerstatus

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.

8 Verteilungssicht

Die folgende Abbildung zeigt die Verteilungssicht der Endanwendung.

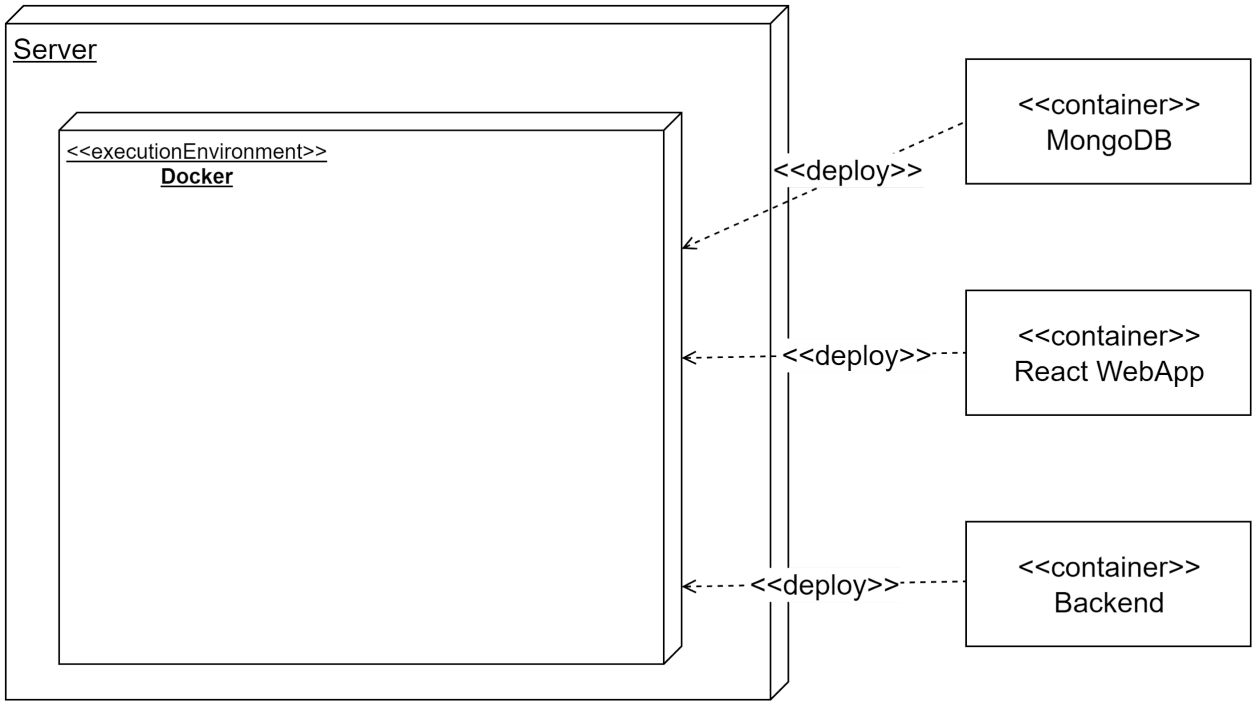

Abbildung 8.1 – Verteilungssicht der Endanwendung

Die drei Docker Container der LigMaTrack Endanwendung können auf einem (zu sehen in Abbildung 8.1) oder auf mehreren Servern verteilt werden, auf denen eine Docker Umgebung aufgesetzt wurde. Hierbei kann zusätzlich auf eine Orchestrierungstechnologie wie Kubernetes zurückgegriffen werden, um Lastverteilung und Ausfallsicherheit zu gewährleisten.

9 Entwurfsentscheidungen

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

9.1 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. Weil 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.

9.2 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.

9.3 MongoDB Datenbank

Der Vorteil einer NoSQL-Datenbank ist, dass man kein komplexes 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. Daraus folgend wurde gegen eine relationale Datenbank mit komplexem Entity-Relationship-Modell entschieden.

9.4 Docker Container

Die Intention der Nutzung von Docker Containern für die React WebApp, Backend und die Datenbank des Endprodukts liegt darin, dass die Container besonders leichtgewichtig sind. Dadurch eignen sie sich besonders für den Servereinsatz, da keine schwergewichtigen Betriebssysteme benötigt werden und Konfigurationen einmalig für die Container erstellt werden müssen. Der Fokus für den Einsatz besteht ausschließlich darin, die Docker Umgebung selbst auf dem System lauffähig zu machen. Docker und zusätzliche Technologien wie Kubernetes bieten weitere Vorteile wie z.B. Lastverteilung, Ausfallsicherheit und die Programmiersprachunabhängigkeit der einzelnen Container, wodurch das Endsystem sehr flexibel wird, Skalierbarkeit bietet und leicht erweitert werden kann. Die Alternative wäre eine native Installation der oben genannten Komponenten. Dies würde jedoch zu einer

Vervielfachung der Komplexität und des Installationsaufwandes führen, sowie eine Einführung von Skalierung, Lastverteilung und Ausfallsicherheit deutlich erschweren.

9.5 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.

9.6 Node.js-Express-Backend

Für das Backend wurde Node.js in Kombination mit Express gewählt, um mit dem Frontend, dem 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, ist sehr leichtgewichtig und stellt dadurch eine einfache Integration in Docker bereit. Als Alternative zu Express kann Nest.js gewählt werden, jedoch fehlt die nötige Erfahrung damit im Team.

10 Risiken

Risiko-Nr.	Beschreibung	Maßnahme
1	Der Ausfall der Websocket Verbindung vom Frontend zum Backend führt zu einem Verlust von externen Daten, welche nicht mehr abrufbar wären. Es können somit keine neue Aufträge und Live Standortdaten aus dem Ligenium Data Pool abgerufen werden.	Backend und Frontend Container neu starten.
2	Der Ausfall der REST Verbindung vom Frontend zum Backend verhindert es den ausgewählten Auftrags als beschädigt zu markieren.	Backend und Frontend Container neu starten.
3	Der Ausfall der REST/Websocket Verbindung von Backend zum Ligenium Data Pool Führt zu einem Verlust von GPS-Positionsdaten der Ladungsträger sowie den Auftragsdaten an die WebApp.	Backend Container neu starten.
4	Der Ausfall des MongoDB Node Driver führt zu einem Verlust der Verbindung des Backends zur Datenbank MongoDB. Es können keine persistieren Auftragsdaten mehr abgerufen werden.	MongoDB Container neu starten.

11 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]
K	-	Kubernetes	“Kubernetes (K8s) ist ein Open-Source-System zur Automatisierung der Bereitstellung, Skalierung und Verwaltung von containerisierten Anwendungen.” (https://kubernetes.io/de/) [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	NFA	Nichtfunktionale Anforderung	“Nichtfunktionale Anforderungen, auch Technische Anforderungen genannt, beschreiben Aspekte, die typischerweise mehrere oder alle funktionalen Anforderungen betreffen bzw. überschneiden (cross-cut). Sie haben in der Regel einen Einfluss auf die gesamte Softwarearchitektur. Außerdem beeinflussen sich nichtfunktionale Anforderungen gegenseitig.” (https://link.springer.com/content/pdf/10.1007%2F978-3-8274-2246-0_9.pdf) [14.06.2022]
	-	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]