

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
PROFESSUR FÜR ANWENDUNGSSYSTEME UND E-BUSINESS

PROJEKTSEMINAR SYSTEMENTWICKLUNG
ENTWICKLUNG MOBILER ANWENDUNGEN
SOMMERSEMESTER 2025

**Entwicklung eines mobilen Dashboards zur Visualisierung und Analyse
von vernetzten Transportfahrzeugen in der Intralogistik**

Ahmed Ali (21849494)
Fared Alsayegh (24502840)
Alina Angermann (21962842)
Yanzhe Peng (24025061)
Till Sernau (21976380)
Christopher Zech (21870926)

Inhaltsverzeichnis	
Abbildungsverzeichnis	IV
Tabellenverzeichnis	V
Abkürzungsverzeichnis	VI
1 Einleitung	1
2 Theoretische Grundlagen.....	3
2.1 Definition des Begriffs „Intralogistik“	3
2.2 Innerbetriebliche Lagersysteme	3
2.3 Überwachung autonomer Transportfahrzeuge in der Intralogistik	4
3 Methodisches Vorgehen bei der Prototypentwicklung.....	6
4 Erläuterung des fiktiven Szenarios	8
4.1 Elemente des fiktiven Lagers	8
4.2 Einsatz autonomer Transportfahrzeuge im fiktiven Lager.....	9
4.3 Beispielhafter Prozess im fiktiven Lager	10
5 Anforderungen an den Prototypen.....	12
5.1 Funktionale Anforderungen.....	12
5.1.1 Visualisierung.....	12
5.1.2 Ereignisbehandlung	15
5.1.3 Interaktion und Steuerung.....	15
5.2 Nicht-funktionale Anforderungen.....	16
6 Konzeption des Prototyps	18
6.1 Funktionsmodellierung	18
6.2 Mockups zur Gestaltung der Benutzeroberfläche	20
6.3 Prozessmodellierung.....	23
6.4 Datenmodellierung	24
7 Implementierung des Prototyps.....	26
7.1 Überblick über die verwendeten Technologien und Frameworks	26
7.2 Datenhaltung des Prototyps	28

7.3 Umsetzung der Anforderungen	30
7.3.1 Visualisierung.....	30
7.3.2 Ereignisbehandlung	35
7.3.3 Interaktion und Steuerung.....	36
7.3.4 Systemanforderungen.....	36
7.4 Umsetzung des fiktiven Szenarios	38
8 Schlussbetrachtung.....	39
Anhang	VII
Literaturverzeichnis	XVI
Ehrenwörtliche Erklärung.....	XX

Abbildungsverzeichnis

Abbildung 1: Layout des fiktiven Lagers	8
Abbildung 2: Use Case Diagramm	19
Abbildung 3: Mockup der Startseite (Desktop-Ansicht)	20
Abbildung 4: Mockup der Fahrzeugübersicht (Desktop-Ansicht)	21
Abbildung 5: Mockup der Statistikseite (Desktop-Ansicht)	21
Abbildung 6: Mockup der mobilen Seite	22
Abbildung 7: Startseite in der Desktopansicht.....	31
Abbildung 8: Lagerkarte	32
Abbildung 9: Ausschnitt aus der Detailansicht der Fahrzeuge	33
Abbildung 10: Ausschnitt aus der Fahrzeugübersicht	33
Abbildung 11: Ausschnitt aus der Transportübersicht	34
Abbildung 12: Statistik-Seite.....	35
Abbildung 13: Benachrichtigungssystem.....	35
Abbildung 14: Startseite in der mobilen Ansicht	37
Abbildung 15: EPK zur Abbildung des Wareneingangsprozesses	VIII
Abbildung 16: EPK zur Abbildung des Warenausgangsprozesses	IX
Abbildung 17: Aktivitätsdiagramm zur Priorisierung von Aufträgen.....	XII
Abbildung 18: Aktivitätsdiagramm Route anzeigen	XIII
Abbildung 19: Aktivitätsdiagramm Route anpassen	XIV
Abbildung 20: ERM zur Datenmodellierung	XV

Tabellenverzeichnis

Tabelle 1: Anforderungstabelle.....	XI
-------------------------------------	----

Abkürzungsverzeichnis

AGV	Automated Guided Vehicles
EPK	Ereignisgesteuerte Prozesskette
ERM	Entity-Relationship-Modell
MVC	Model-View-Controller
ORM	Objekt Relationaler Mapper
VDMA	Verband Deutscher Maschinen- und Anlagenbau

1 Einleitung

Die Nachfrage nach mobilen Robotern in der Intralogistik steigt. Allein 2023 wurden rund 113.000 Roboter für Transport- und Logistikaufgaben verkauft (vgl. VDMA 2024). Mit dem fortschreitenden Automatisierungsgrad innerbetrieblicher Material- und Warenflüsse gewinnt auch der Einsatz autonomer Transportfahrzeuge (Automated Guided Vehicles, AGVs) zunehmend an Bedeutung (vgl. VDMA 2025). Diese spielen eine wichtige Rolle, um die Effizienz und Produktivität der Intralogistik zu erhöhen (vgl. Li/Schulze 2024, S. 268). AGVs sind fahrerlose Transportsysteme, mit denen Waren automatisiert in intralogistischen Umgebungen wie z. B. einem Lager oder einer Fertigungsumgebung transportiert werden können (vgl. Correia et al. 2020, S. 241 f.). Störungen, wie Pannen, Konflikte oder ein niedriger Ladezustand können jedoch die Effizienz der AGVs beeinträchtigen (vgl. Gao et al. 2024, S. 2). Während einige Störungen durch die AGVs selbst behoben werden können, erfordern andere ein manuelles Eingreifen. Sofern solche Störungen nicht zeitnah behoben werden können, kann dies die Verfügbarkeit der Fahrzeuge reduzieren und zu Verzögerungen im intralogistischen Warenfluss führen (vgl. Li/Schulze 2024, S. 269 f.). Eben dies können sich Unternehmen, in Anbetracht der hohen zeitlichen Anforderungen an die Intralogistik, nicht erlauben, denn sie führen im schlimmsten Fall zu Unterbrechungen in der Produktion oder verzögter Lieferung von Waren (vgl. Baginski 2006, S. 266).

Zur Verbesserung der Reaktionsfähigkeit auf Störungen besteht der Bedarf nach Systemen, welche den Zustand der AGVs in Echtzeit überwachen und kritische Ereignisse melden (vgl. Gao et al. 2024, S. 2; Li/Schulze 2024, S. 268). In der wissenschaftlichen Literatur sind zahlreiche Arbeiten über AGV-Systeme zur Kontrolle und Optimierung des AGV-Einsatzes zu finden (z. B. Correia et al. 2020; Le-Anh/De Koster 2006; Li/Schulze 2024; Schmidt et al. 2020). Der Fokus dieser Arbeiten liegt jedoch primär auf der präventiven Vermeidung von Störungen, beispielsweise durch Optimierung der Routenfindung oder verbesserte La-deplanung, und nicht auf dem Umgang mit bereits eingetretenen Störungen. Eine Möglichkeit, die sich in anderen Anwendungskontexten zur Überwachung laufender Aktivitäten in Echtzeit bereits bewährt hat, sind Dashboards (vgl. Pappas/Whitman 2011, S. 205 ff.). Ihr Potenzial in der Intralogistik wurde bisher wenig untersucht. Es fehlt an zentralen Anforderungen für mobile Dashboards für

diesen spezifischen Anwendungsfall. Vor diesem Hintergrund stellt sich die zentrale Forschungsfrage: „*Wie kann ein mobiles Dashboard gestaltet werden, um die Überwachung, Steuerung und Analyse von AGVs in intralogistischen Umgebungen effizient zu unterstützen?*“.

Zur Beantwortung dieser Frage wurde ein Prototyp auf Grundlage eines fiktiven Lagerszenarios entwickelt, welcher im Rahmen dieser Projektdokumentation beschrieben werden soll. Aufbauend auf den in Kapitel 2 gelegten theoretischen Grundlagen wird in Kapitel 3 das methodische Vorgehen beschrieben. Kapitel 4 stellt das fiktive Szenario vor, bevor in Kapitel 5 die funktionalen Anforderungen an das Dashboard abgeleitet werden. Die Konzeption des Prototyps erfolgt in Kapitel 6, während Kapitel 7 die technische Umsetzung dokumentiert. Abschließend reflektiert Kapitel 8 die Ergebnisse und bietet einen Ausblick auf zukünftige Entwicklungen.

2 Theoretische Grundlagen

Ziel dieses Kapitels ist, die theoretischen Grundlagen der Intralogistik, innerbetrieblichen Lagersystemen und autonomen Fahrzeugen in der Intralogistik sowie Systemen zur Überwachung innerbetrieblicher Logistikprozesse darzustellen. Sie schaffen die Grundlage für das im weiteren Verlauf entwickelte Szenario.

2.1 Definition des Begriffs „Intralogistik“

Der Begriff „Intralogistik“ wurde im Jahr 2003 vom Fachverband für Fördertechnik und Logistiksysteme des Verbands Deutscher Maschinen- und Anlagenbau (VDMA) entwickelt. Ziel war es, eine eigenständige Bezeichnung für eine sich dynamisch entwickelnde Branche zu schaffen, die sich einerseits klar von der klassischen Logistik, also dem reinen Transport von Waren, abgrenzt, andererseits über technikorientierte Begriffe wie „Materialflusstechnik“ oder „Fördertechnik“ hinaus geht (vgl. Günther 2006, S. 5). Der Begriff sollte den innerbetrieblichen Materialfluss innerhalb eines Logistikknotens beschreiben und gleichzeitig die notwendige Interdisziplinarität der beteiligten Technologien und Prozesse widerspiegeln (vgl. Arnold 2006, S. 1). Nach der daraus folgenden Definition des Fachverbandes umfasst die Intralogistik „die Organisation, Steuerung, Durchführung und Optimierung des innerbetrieblichen Materialflusses, der Informationsströme sowie des Warenumschlags in Industrie, Handel und öffentlichen Einrichtungen“ (VDMA 2025).

Die entstandene Definition verdeutlicht die Komplexität der Branche, denn nicht nur bezieht sich „innerbetrieblich“ dabei auf geschlossene Unternehmensstandorte der verschiedensten Anwendungsbereiche, wie z. B. Produktionsstätten von Industrieunternehmen, Distributionslogistik großer Handelsketten oder Umschlagplätze in Häfen (vgl. Arnold 2006, S. 2; Fottner et al. 2021, S. 1). Auch umfasst die Intralogistikbranche ein breites Spektrum an Anbietern und Lösungen. Dazu gehören beispielsweise die Förder- und Lagertechnik und die Verpackungstechnik sowie Softwarelösungen zur Unterstützung innerbetrieblicher Logistikprozesse (vgl. Günther 2006, S. 6).

2.2 Innerbetriebliche Lagersysteme

Die zentralen Prozesse innerhalb eines Lagers lassen sich im Wesentlichen auf die Kernschritte Wareneingang, Lagerung, Kommissionierung und Warenausgang reduzieren (vgl. Gudehus 2007, S. 583; Hompel/Schmidt 2010, S. 5;

Klaus/Krieger 2009, S. 294). Der Materialfluss im Lager beginnt mit der Warenannahme, in dem die Ware von einem Spediteur übernommen wird, und dem Wareneingang, wo unter anderem die Ware im innerbetrieblichen Warehouse-Management-System registriert wird (vgl. Hompel/Schmidt 2010, S. 24 ff.). Anschließend erfolgt die Einlagerung der Ware im Lager. Dazu wird der Ware ein Lagerplatz zugewiesen und sie entsprechend dorthin transportiert (vgl. Hompel/Schmidt 2010, S. 28 ff.).

Liegt ein Kundenauftrag oder eine interne Anforderung vor, wird die benötigte Ware aus dem Lager entnommen (Auslagerung). Häufig ist dieser Schritt mit der Kommissionierung, sprich dem geordneten Auslagern und systematischen Zusammenstellen von Lagereinheiten, verbunden (vgl. Gudehus 2007, S. 583). Abschließend erfolgt der Transport der Waren in den Warenausgang, wo sie kontrolliert und für den Versand vorbereitet sowie für den anschließenden Transport bereitgestellt werden (vgl. Hompel/Schmidt 2010, S. 53; Klaus/Krieger 2009, S. 294).

Die Ausgestaltung von Lagersystemen variiert stark und richtet sich nach den individuellen Anforderungen der einzelnen Unternehmen. Es ergeben sich zahlreiche Möglichkeiten in Bezug auf Gestaltung der Lagerplätze sowie Auswahl passender Lagergeräte (vgl. Gudehus 2007, S. 583).

2.3 Überwachung autonomer Transportfahrzeuge in der Intralogistik

AGVs sind mobile Roboter, die in der Intralogistik zur automatisierten Beförderung von Gütern eingesetzt werden (vgl. Boehning 2014, S. 245; Correia et al. 2020, S. 241; Fottner et al. 2021, S. 2 f.). Dabei können nahezu alle gängigen Flurförderfahrzeuge auch als AGVs realisiert werden (vgl. Hompel/Schmidt 2010, S. 106). Verschiedene Sensoren der Fahrzeuge ermöglichen es den autonomen Systemen, ihre Umwelt zu erfassen. Gleichzeitig können die Sensoren der Identifikation, Lokalisation und Zustandsüberwachung der AGVs dienen (vgl. Fottner et al. 2021, S. 5 f.).

Die Gründe für den Einsatz von AGVs sind vielfältig. Beispielsweise erhöhen sie die Arbeitssicherheit durch Transporte in Risikobereichen, erfüllen die hohen fördertechnischen Anforderungen wie Präzision und Wiederholgenauigkeit, tragen zur Reduktion von Personalkosten bei und können zu einer gesteigerten Produktivität und Effizienz in der Intralogistik führen (vgl. Correia et al. 2020, S. 242;

Hompel/Schmidt 2010, S. 106; Li/Schulze 2024, S. 268). Trotzdem können beim Einsatz von AGVs Störungen auftreten, beispielsweise durch Konflikte mit anderen AGVs oder Gegenständen, eine unzureichende Ladeplanung, technischen Ausfällen oder menschlichem Versagen (vgl. Correia et al. 2020, S. 243; Gao et al. 2024, S. 2 ff.; Schmidt et al. 2020, S. 15). Besonders kritisch sind sogenannte Deadlock-Situationen, in denen die Fahrzeuge nicht mehr selbst in der Lage sind, den Konflikt aufzulösen und zur Weiterfahrt auf menschliches Eingreifen angewiesen sind (vgl. Le-Anh/De Koster 2006, S. 17).

Um die Komplexität dynamischer intralogistischer Systeme wie AGVs beherrschbar zu machen, braucht es Systeme zur Überwachung dieser. Ein Ansatz zur Visualisierung und Analyse solcher Systeme können sogenannte Dashboards sein, welche sich in anderen Anwendungskontexten bereits als bewährte Methode gezeigt haben. Durch eine einfache, übersichtliche Darstellung der Prozesse ermöglichen sie schnelle Reaktionszeiten bei Störungen sowie detaillierte Analysen, um Ursachen für Probleme zu erkennen und Maßnahmen abzuleiten (vgl. Pappas/Whitman 2011, S. 205 ff.). Mobile Dashboards erweitern die Form des traditionellen Dashboards, meist angepasst an kleinere Endgeräte wie Tablets oder Smartphones. Sie ermöglichen die orts- und zeitunabhängige Bereitstellung entscheidungsrelevanter Informationen und können damit die Steuerung sowie Optimierung intralogistischer Transportprozesse unterstützen (vgl. Gröger et al. 2016, S. 1335 f.). Im Umfeld autonomer Produktionssysteme können mobile Dashboards bereits einen Beitrag zur kontinuierlichen Überwachung des Systemzustands leisten, indem sie Abweichungen und Störungen frühzeitig sichtbar machen und Einleitung geeigneter Gegenmaßnahmen unterstützen (vgl. Flores-García et al. 2022, S. 521 ff.). Zur Überwachung von autonomen Fahrzeugen haben THAMRONGAPHICHARTKUL ET AL. (2020, S. 4) beispielsweise ein Dashboard zur Überwachung von mobilen Service-Robotern in einer Krankenhausumgebung entwickelt, auf welchem die Nutzer Zustand sowie die Position des Roboters in Echtzeit überwachen, aber auch interaktive Funktionen zur Steuerung nutzen können. Für die Intralogistik ergeben sich daraus vielversprechende Anwendungsmöglichkeiten, deren konkrete Anforderungen jedoch bislang nur unzureichend untersucht sind.

3 Methodisches Vorgehen bei der Prototypentwicklung

Im Rahmen der Projektarbeit wurde das Wasserfallmodell als Grundlage für das methodische Vorgehen genutzt. Die Umsetzung dessen soll im Folgenden beschrieben werden.

Das Wasserfallmodell gliedert die Softwareentwicklung in klar voneinander abgegrenzte, aufeinanderfolgende Phasen: von der Analyse über die Konzeption und das Design bis hin zur Implementierung und zum Testen des Prototyps. Diese sequenzielle Struktur ermöglicht eine systematische und nachvollziehbare Vorgehensweise im Projekt. Jede Phase verfolgt ein eigenes Ziel und liefert Ergebnisse, die als Grundlage für die nachfolgende Phase dienen (vgl. Adenowo/Adenowo 2013, S. 429). Da die grundlegenden Anforderungen bereits zu Beginn des Projekts grob definiert sind, kann eine strukturierte und vorausschauende Planung erfolgen. Durch die kontinuierliche Dokumentation in jeder Phase wird der Projektverlauf nachvollziehbar und transparent gestaltet (vgl. Adenowo/Adenowo 2013, S. 432)

Die klare Phasenstruktur des Wasserfallmodells erleichtert sowohl die Priorisierung als auch die zielgerichtete Bearbeitung einzelner Schritte ohne inhaltliche Überschneidungen. Aufgaben lassen sich effizient im Team verteilen und der Projektfortschritt kann systematisch bewertet werden. In der Praxis zeigt sich jedoch, dass bestimmte Probleme erst in späteren Phasen erkannt werden, obwohl sie ihren Ursprung in vorherigen Schritten haben. So treten beispielsweise in der Designphase Lücken in den zuvor formulierten Anforderungen zutage (vgl. Saravacos/Curinga 2023, S. 5). Durch regelmäßige Meetings, kontinuierliche Kommunikation und aktives Feedback lassen sich solche Herausforderungen effizient adressieren. Auch durch iterative Elemente innerhalb des Wasserfallmodells ist eine flexible Reaktion auf solche Erkenntnisse möglich, obwohl das Modell grundsätzlich sequenziell aufgebaut ist (vgl. Fritzsche/Keil 2007, S. 8). Auf diese Weise kann der Projektfortschritt kontinuierlich mit den definierten Zielen der jeweiligen Phase abgeglichen und gegebenenfalls korrigiert werden. Die Entscheidung gegen ein agiles Vorgehen wie Scrum wurde bewusst getroffen. Unter den gegebenen Rahmenbedingungen, insbesondere der begrenzten Zeit und klar definierten Anforderungen, erschien das Wasserfallmodell als besser geeignet, um eine planbare und strukturierte Umsetzung sicherzustellen (vgl. Saravacos/Curinga 2023, S. 4 f.).

Im Rahmen dieses Projekts kann das Wasserfallmodell wie folgt umgesetzt werden. In der ersten Phase, der Analysephase, werden Problemstellung und Zielsetzung des Dashboards mithilfe einschlägiger Literatur definiert. Auf dieser Grundlage wird ein fiktives Szenario entwickelt und anschließend die funktionalen und nicht-funktionalen Anforderungen formuliert. In der Konzeptionsphase werden zentrale Anwendungsfälle modelliert und die grundlegende Systemstruktur entworfen. Dazu werden die Benutzerinteraktionen durch ein Use-Case-Diagramm visualisiert sowie die Abläufe einzelner Anwendungsfälle mit Hilfe von Aktivitätsdiagrammen dargestellt. So entsteht eine ganzheitliche Sicht auf das Systemverhalten. Während der Designphase werden auf Basis der zuvor erarbeiteten Anforderungen erste Entwürfe für das Dashboard und ein konzeptionelles Datenbankschema erstellt. Ziel ist die Entwicklung eines benutzerfreundlichen und funktionalen Prototyps sowie einer strukturierten Backend-Architektur. In der Implementierungsphase entsteht in mehreren Iterationen ein funktionaler Prototyp. Der Schwerpunkt liegt auf der Visualisierung von Fahrzeugpositionen, der Darstellung relevanter Auftragsdaten und den Fahrerrouten. Durch die inkrementelle Entwicklung lassen sich Teifunktionen frühzeitig testen und gezielt optimieren. Anschließend erfolgten das Testen und Auswerten. Der Prototyp wird regelmäßig in Reviews präsentiert, was die Umsetzung gezielter Verbesserungsvorschläge ermöglicht. Die Tests konzentrieren sich insbesondere auf Benutzerfreundlichkeit, Darstellungsqualität und technische Stabilität.

4 Erläuterung des fiktiven Szenarios

Im Rahmen dieses Projekts wurde ein fiktives Lager entwickelt, welches die typischen Abläufe und wichtigsten Gesichtspunkte darstellt, in seiner Komplexität jedoch weitestgehend reduziert wurde. Als Grundlage für die anschließenden Kapitel soll das fiktive Szenario im Folgenden beschrieben werden.

4.1 Elemente des fiktiven Lagers

Bei dem fiktiven Lager handelt es sich um ein halbautomatisches Warenlager. Dies ist dadurch gekennzeichnet, dass Teile der Lagerprozesse automatisiert ablaufen, während manche Tätigkeiten manuell erfolgen (vgl. Yuan et al. 2019, S. 355). Das fiktive Lager (dargestellt in Abbildung 1: Layout des fiktiven Lagers) verfügt über einen Wareneingangs-, Warenausgangs-, Kommissionierbereich, ein Hochregallager sowie einen Wartungsbereich für AGVs. Während Kommisionierung, Warenein- und Warenausgang manuell bedient werden, werden alle weiteren Transportbewegungen innerhalb des Lagers durch zwei autonome Flurförderfahrzeuge übernommen.

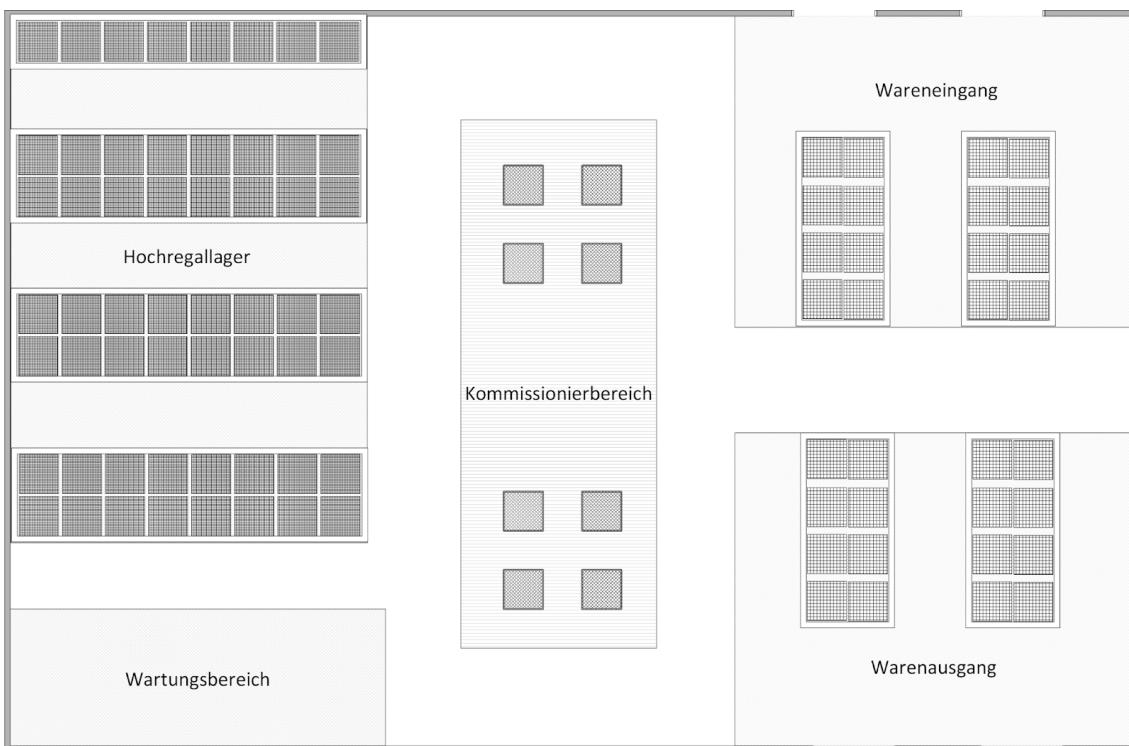


Abbildung 1: Layout des fiktiven Lagers

Zum Wareneingangsbereich gehören, ebenso wie zum Warenausgangsbereich, sieben Fachregallager. Diese Fachregallager bestehen aus einzelnen Modulen, die je einen Lagerplatz haben. Das hat den Vorteil, dass auf alle Ladeeinheiten

direkt zugegriffen werden kann (vgl. Gudehus 2007, S. 595 ff.). Jeder Lagerplatz ist mit einem Etikett gekennzeichnet, welches den Lagerplatz mit Nummer in Klarschrift und als Barcode kennzeichnet (vgl. Hompel/Schmidt 2010, S. 283 f.). Das Hochregallager besteht aus elf je zwölf Meter hohen, mehrstöckigen Fachregallagern, welche in parallelen Reihen stehen. Die Kommissionierung im fiktiven Lager wird als dynamische Kommissionierung (Ware-zum-Mann-Prinzip) durchgeführt. Dabei wird die Ware aus dem Lager zu einem Kommissionierplatz bewegt, um dort die Bestellung zusammenzustellen (vgl. Klaus/Krieger 2009, S. 278). Da her befinden sich im fiktiven Lager mehrere Bereitstellplätze, zu denen die Waren zur Kommissionierung transportiert werden können. In dem Wartungsbereich des fiktiven Lagers befinden sich Parkplätze mit Ladungsvorrichtungen für die im Lager eingesetzten AGVs.

4.2 Einsatz autonomer Transportfahrzeuge im fiktiven Lager

Die Transportaufgaben zwischen den verschiedenen Bereichen des fiktiven Lagers werden von AGVs übernommen. Gemäß der vereinfachenden Annahme, dass alle Waren mit Hilfe von Euro-Paletten als Ladehilfsmittel transportiert werden, können diese von den AGVs automatisiert auf- und abgeladen werden (vgl. Hompel/Schmidt 2010, S. 283 ff.). Dabei werden im fiktiven Lager zwei verschiedene Arten von Transportfahrzeugen eingesetzt. Zum Transport zwischen Kommissionierbereich und Hochregallager werden Hochregalstapler eingesetzt. Hochregalstapler verfügen über seitlich ausfahrbare Teleskopgabeln oder einen drehbaren Gabelträger, der die Gabel beidseitig schwenken kann und somit die Bedienung von Regalplätzen erleichtert. Sie ermöglichen die Bedienung von Regalen bis zu einer Höhe von etwa zwölf Metern und arbeiten dabei oft mit mechanischer oder induktiver Zwangsführung sowie automatischen Fachwahlsystemen oder Kamerasyttemen zur präzisen Positionierung (vgl. Hompel/Schmidt 2010, S. 101). Der Transport zwischen Wareneingang, Warenausgang und Kommissionierbereich wird von automatischen Hochhubwagen übernommen, welche die kleineren Hilfsregale in diesen Bereichen bedienen können.

Im Rahmen des fiktiven Szenarios wird angenommen, dass die autonomen Fahrzeuge über eine gemeinsame AGV-Steuerungssoftware vernetzt sind. Über dieses System berechnen die Fahrzeuge ihre Routen, reagieren autonom auf Hindernisse und übermitteln Daten zu ihren internen Zuständen. Transportaufträge

mit den jeweiligen Start- und Ziellagerplätzen werden über das Warehouse-Management-System an die Fahrzeuge übermittelt, womit die Routenfindung initiiert wird. Für die Erkennung ihrer internen Zustände sowie ihrer Umwelt haben die AGVs eine Reihe von technischen Ausstattungen, welche Funktionen wie Identifikation, Lokalisation und Sicherheit erfüllen. Sensoren liefern dafür große Mengen an Informationen über die unmittelbare, erweiterte und erwartete Umgebung sowie den internen Zustand der AGVs (vgl. Ellithy et al. 2024, S. 25 f.; Fottner et al. 2021, S. 5 ff.).

4.3 Beispielhafter Prozess im fiktiven Lager

Angelehnt an den Kernprozess des Lagerns, welcher aus dem Einlagern der Waren, dem Aufbewahren der Waren auf Lagerplätzen und der Warenauslagerung besteht, ergänzt um Zusatzfunktionen wie dem Kommissionieren, wird nachfolgend ein typischer Prozess in dem betrachteten fiktiven Warenlager beschrieben (vgl. Gudehus 2007, S. 583). Für eine graphische Darstellung der Prozesse wurden ereignisgesteuerte Prozessketten (EPK) erstellt (siehe Anhang A, Abbildung 15: EPK zur Abbildung des Wareneingangsprozesses und Abbildung 16: EPK zur Abbildung des Warenausgangsprozesses).

Im Wareneingangsbereich des Lagers wird die Ware vom Spediteur übernommen und im innerbetrieblichen Warehouse-Management-System registriert (vgl. Hompel/Schmidt 2010, S. 24 ff.). Anschließend wird die Ware manuell auf einen Lagerplatz im Wareneingang verbracht. Das Warehouse-Management-System initiiert daraufhin einen Transportauftrag für die Waren. Je nach Auftrag werden die Waren aus dem Abholbereich per autonomen Hochhubwagen im Lager verteilt (vgl. Gudehus 2007, S. 584). Besteht ein direkter Bedarf nach den Waren, wird eine sogenannte Durchlagerung durchgeführt, wobei die Waren vom Wareneingang direkt in den Warenausgang gebracht werden (vgl. Hompel/Schmidt 2010, S. 29). Besteht ein direkter Bedarf mit der Notwendigkeit einer Kommission, werden die Waren von dem Wareneingang in den Kommissionierbereich gebracht. Hier werden die Waren von einem Mitarbeiter individuell neu zusammengestellt (vgl. Gudehus 2007, S. 685; Hompel/Schmidt 2010, S. 34). Die kommissionierten Waren werden daraufhin von einem Mitarbeiter im Warehouse-Management-System markiert, woraufhin ein neuer Transportauftrag initiiert wird, um die kommissionierten Waren vom Kommissionierbereich zu einem Lagerplatz im Warenausgang zu befördern.

Wenn die Waren langfristig eingelagert werden sollen, werden sie von dem Lagerplatz im Wareneingang zu einem Bereitstellplatz im Kommissionierbereich gebracht und hier an einen autonomen Hochregalgabelstaplern übergeben, welcher die Ware in einem entsprechenden Fach im Hochregallager platziert. Besteht ein Auslagerungsauftrag für gelagerte Waren, werden sie von AGVs in das Hilfslager des Warenausgangs gebracht (vgl. Gudehus 2007, S. 585). Dabei transportiert ein automatischer Hochregalgabelstapler die Waren vom Hochregallager auf einen Bereitstellplatz im Kommissionierbereich, wo die Ware gegebenenfalls kommissioniert und anschließend von einem automatischen Hochhubwagen in den Warenausgang transportiert wird. Mit der Warenbereitstellung im Warenausgang endet der Prozess, welcher im Rahmen dieser Arbeit betrachtet wird.

In dem beschriebenen Prozess können Störungen auftreten, welche die autonomen Transportfahrzeuge betreffen (vgl. Kapitel 2.3). Ohne ein mobiles Dashboard zur Überwachung der AGVs werden Störungen wie technische Defekte oder blockierte Routen nicht sofort erkannt. Mitarbeiter müssen in diesem Fall Störungen manuell entdecken und beheben, etwa durch physisches Eingreifen oder Umplanen im AGV-Navigationssystem oder Warehouse-Management-System. Dadurch verlängern sich Reaktionszeiten und es kommt zu Verzögerungen im Transportprozess. Mit Hilfe des entwickelten mobilen Dashboards soll die Störung automatisch im System erkannt und über das Dashboard gemeldet werden. Mitarbeitende erhalten direkt eine Benachrichtigung und können gezielt, z.B. durch eine manuelle Routenanpassung im Dashboard, eingreifen. So verkürzt sich die Reaktionszeit, Störungen werden effizienter behoben und die Prozessstabilität steigt.

5 Anforderungen an den Prototypen

Aufbauend auf der bereits beschriebenen Problemstellung bildet dieses Kapitel die Grundlage für die spätere Implementierung eines mobilen Dashboards zur Überwachung und Steuerung der AGVs. Im Folgenden sollen die Anforderungen an das mobile Dashboard dargestellt werden. Eine vollständige tabellarische Übersicht aller funktionalen und nicht-funktionalen Anforderungen befindet sich im Anhang B in Tabelle 1: Anforderungstabelle.

Wie bereits in Kapitel 2 dargelegt, eignen sich mobile Dashboards, um Störungen durch Vorhersagen zu vermeiden und Reaktionszeiten auf Störungen zu verkürzen. Der entwickelte Prototyp soll das Lagerpersonal (darunter Lagerleitung, Lagermitarbeiter) dabei unterstützen, einen effizienten Betriebsablauf sicherzustellen. Durch ein mobiles, interaktives Dashboard soll ein umfassender Überblick über den Zustand des Lagers und der eingesetzten AGVs in Echtzeit ermöglicht werden. Die Anzeige relevanter Informationen sowie automatische Benachrichtigungen bei kritischen Situationen, wie etwa Störungen oder Staus, sollen helfen, unnötige Ausfallzeiten zu vermeiden.

Um einen reibungslosen und effizienten Ablauf zu gewährleisten, ist eine klare, strukturierte und detaillierte Definition der Anforderungen unerlässlich. Diese bildet nicht nur die Grundlage für die spätere Implementierung, sondern ermöglicht auch eine gezielte Bewertung und Weiterentwicklung des Systems. Es wird zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden: Funktionale Anforderungen beschreiben, welche Funktionen das System einem Nutzer, für dessen Zielerreichung oder Problemlösung, bereitstellen muss. Nicht-funktionale Anforderungen legen fest, unter welchen Qualitätsmerkmalen diese Funktionen umgesetzt werden sollen. Zur besseren Übersicht sind die funktionalen Anforderungen in drei Kategorien unterteilt: Visualisierung, Ereignisbehandlung und Interaktion & Steuerung.

5.1 Funktionale Anforderungen

5.1.1 Visualisierung

Wie FEW (2006, S. 6, S. 81) betont, besteht die zentrale Funktion eines Dashboards darin, „die wichtigsten Informationen auf einem einzigen Bildschirm so zu verdichten, dass ein effizientes Monitoring möglich ist“, wobei eine maximale Informationsdichte bei minimaler kognitiver Belastung angestrebt werden sollte.

Dieses Prinzip soll im vorliegenden System durch eine interaktive Lagerkarte als zentrale Überwachungsebene umgesetzt werden. Ergänzt wird diese durch übersichtliche Listen zur Detailanalyse von Transportaufträgen und Fahrzeugen, eine Live-Positionsanzeige sowie eine Übersichtsseite für Statistiken und Auswertungen. Die Daten sollen klar organisiert und gezielt verdichtet dargestellt werden, um den Nutzern einen schnellen, umfassenden Überblick zu ermöglichen und fundierte Entscheidungen im Betriebsalltag zu unterstützen.

Lagerkarte

Die interaktive 2D-Lagerkarte (Anforderung 1.1) soll einen vollständigen Überblick über das Lager ermöglichen. Zentrale Bereiche wie Wareneingang und Warenausgang sollen deutlich gekennzeichnet sein, um die Orientierung zu erleichtern. Die Bewegungen und Positionen der AGVs sollen in Echtzeit visualisiert und überwacht werden können. Diese Darstellung soll die Kontrolle der Auftragsausführung in Echtzeit unterstützen und die Analyse operativer Abläufe erleichtern, um fundierte Entscheidungen zu ermöglichen. Lagerplätze sollen auf der Karte mit Belegungsinformationen (z. B. „5 von 10 belegt“) angezeigt werden. Dies soll die visuelle Planung sowie eine effiziente Zuweisung von Transportaufträgen unterstützen.

Live-Positionsanzeige der Fahrzeuge

Die Transportfahrzeuge sollen auf der Lagerkarte in Echtzeit (Anforderung 1.2) als Icons mit der jeweiligen Fahrzeug-ID visualisiert werden (Anforderung 1.2.1). Bei Auswahl eines Fahrzeugs soll zusätzlich die aktuelle Route auf der Karte dargestellt werden (Anforderung 1.2.4), sodass der Fahrverlauf unmittelbar nachvollzogen werden kann. Unterschiedliche Fahrzeugtypen sollen durch verschiedene Symbole gekennzeichnet werden (Anforderung 1.2.2), um eine schnelle visuelle Unterscheidung zu ermöglichen. Bei einer Störung soll ein rotes Warnsymbol am betroffenen Fahrzeug erscheinen (Anforderung 1.2.3), sodass potenzielle Probleme sofort erkennbar sind. Im Falle eines Staus soll direkt ersichtlich sein, welches Fahrzeug betroffen ist, um gezielte Eingriffe effizient durchführen zu können.

Kontextuelle Fahrzeuginformationen

Beim Klicken auf das Fahrzeug-Icon soll eine Detailansicht geöffnet werden, die folgende Informationen enthält: Fahrzeug-ID, Ladestand, Akkuzustand, aktueller

Transportauftrag, Störungsmeldungen sowie den aktuellen Status (z. B. aktiv oder inaktiv) (Anforderung 1.3).

Anpassung der Lagerkarte

Damit das Dashboard in unterschiedlichen Lagerumgebungen einsetzbar ist, soll die Lagerkarte flexibel anpassbar sein (Anforderung 1.4). Dies betrifft insbesondere die Konfiguration von Lagerzonen und Fahrerrouten.

Fahrzeugliste

Eine ergänzende Tabelle soll alle Fahrzeuge mit ihren aktuellen Eigenschaften Fahrzeug-ID, Status, Ladestand, Akkuzustand, aktueller Transportauftrag und Meldungen zeigen. Zudem steht eine Filterfunktion zur Verfügung, über die nach Fahrzeugen gesucht, bzw. die Menge angezeigter Fahrzeuge eingeschränkt werden kann (Anforderung 1.5). So lassen sich z. B. alle Fahrzeuge mit Akkustand < 20 % schnell identifizieren und zur Ladung einplanen.

Transportauftragsliste

Transportaufträge sollen in einer strukturierten Liste angezeigt werden. Zu jedem Auftrag sollen Informationen wie Start- und Zielort, Priorität, aktueller Status sowie das zugewiesene Fahrzeug dargestellt werden. Die Liste soll es ermöglichen, z. B. nicht zugewiesene oder zeitkritische Aufträge rasch zu erkennen und manuell zu priorisieren. Zusätzlich soll eine Filterfunktion zur Verfügung stehen, welche den Benutzern eine gezielte Suche nach bestimmten Aufträgen ermöglicht (Anforderung 1.6).

Statistik-Übersicht

Zur Unterstützung der Analyse und Optimierung des Lagerbetriebs soll das Dashboard eine Übersichtsseite für Statistiken und Auswertungen bereitstellen (Anforderung 1.7). Auf dieser Seite sollen aktuelle Kennzahlen zu Fahrzeugen, Aufträgen und Meldungen sowie ergänzende Tabellen und Listen z. B. zu Fahrzeugtypen, aktuellen Störungen oder auffälligen Fahrzeugen visualisiert werden. Diese Übersicht soll der Lagerleitung helfen, wiederkehrende Engpässe oder ineffiziente Prozesse zu erkennen und gezielt Maßnahmen zur Verbesserung einzuleiten.

Benutzerrollen

Das System verschiedene Benutzerrollen unterscheiden können (Anforderung 1.8). Je nach Rolle (in diesem Fall Lagermitarbeitende und Lagerleitung) sollen unterschiedliche Funktionen zur Verfügung stehen. Die Lagerleitung soll Zugriff auf die Statistik- und Auswertungsübersicht erhalten, während Lagermitarbeitende primär operative Informationen einsehen und ausgewählte Auftragsfunktionen nutzen können. Dieses rollenbasierte Berechtigungskonzept soll sicherstellen, dass jede Nutzergruppe ausschließlich auf für sie relevante Informationen und Steuerungsoptionen zugreifen kann.

5.1.2 Ereignisbehandlung

Wie SHNEIER/BOSTELMAN (2015, S. 6) betonen, ist die sensorische Erkennung von Hindernissen und Systemverzögerungen entscheidend, um die Effizienz von AGVs sicherzustellen. Daraus ergibt sich die Anforderung, dass das Dashboard aktiv bei der frühzeitigen Erkennung von Störungen und kritischen Zuständen unterstützen soll (Anforderung 2.1). Durch automatische Push-Nachrichten (Anforderung 2.2), z. B. Wenn ein Fahrzeug trotz aktivem Auftrag über längere Zeit keine Bewegung zeigt, soll das erkannt und gemeldet werden, um längere Stillstände zu vermeiden.

5.1.3 Interaktion und Steuerung

Auftragssteuerung und Priorisierung

Nutzer sollen Transportaufträge manuell priorisieren oder neu zuweisen können (Anforderung 3.1). Dies soll eine flexible Reaktion auf betriebliche Ereignisse wie Fahrzeugausfälle oder geänderte Prioritäten ermöglichen. Beispielsweise soll ein Auftrag mit hoher Dringlichkeit, der einem blockierten Fahrzeug zugewiesen ist, auf ein verfügbares Fahrzeug übertragen werden können.

Steuerung der Fahrzeuge

Rein automatische Kollisionsvermeidungssysteme, wie sie KUBASAKOVA ET AL. (2024, S. 9) beschreiben, bringen AGVs bei Konflikten lediglich zum Stillstand, ohne aktive Umleitungen vorzunehmen. Dies kann zu ineffizienten Wartezeiten führen. Um dem entgegenzuwirken, soll das System ermöglichen, einzelne Fahrzeuge bei Bedarf aktiv oder inaktiv zu schalten (Anforderung 3.2), um flexibel auf kurzfristige Störungen oder Wartungsbedarf reagieren zu können. Darüber hinaus soll das System die manuelle Anpassung von Fahrtrouten bei Störungen

oder Blockaden erlauben (Anforderung 3.3), um Stillstandszeiten zu minimieren. Die Routenanpassung soll direkt auf der Karte erfolgen und es ermöglichen, betroffene Fahrzeuge effizient umzuleiten. So kann ein Fahrzeug beispielsweise bei einem gestauten Bereich im Lager über eine alternative Route zum Ziel geführt werden, um den Betrieb aufrechtzuerhalten.

5.2 Nicht-funktionale Anforderungen

Neben den funktionalen Anforderungen muss das System auch über bestimmte Eigenschaften verfügen, die seine Qualität, Nutzbarkeit und technische Umsetzung betreffen. Im Folgenden sind die für dieses Projekt in besonderem Maße relevanten nicht-funktionalen Anforderungen dargestellt.

Mobilgerätekompatibilität

Das Dashboard muss auf mobilen Endgeräten wie Tablets oder Smartphones reibungslos funktionieren, um es den Mitarbeitenden zu ermöglichen, auch direkt in der Lagerhalle auf relevante Informationen und Funktionen zuzugreifen, ohne an einen festen Arbeitsplatz gebunden zu sein. Ein Techniker soll beispielsweise direkt am Ort einer Störung die zugehörigen Fahrzeugdaten abrufen können. Dafür muss sich die Größe und die Anordnung der Elemente des Systems dynamisch kleineren Bildschirmen anpassen (Anforderung 4.1).

Echtzeitaktualisierung

Wie KUBASAKOVA ET AL. (2024, S. 5) betonen, ermöglicht die Echtzeitüberwachung von Fahrzeugen eine effizientere Planung im Lageralltag und trägt dazu bei, Konflikte frühzeitig zu vermeiden. Daraus ergibt sich die Anforderung 4.2, dass alle relevanten Daten wie Fahrzeugpositionen, Statusmeldungen und Störungen in nahezu Echtzeit aktualisiert werden müssen, um fundierte Entscheidungen zu ermöglichen. Eine verzögerungsfreie Darstellung ist insbesondere dann entscheidend, wenn unerwartete Betriebszustände auftreten. Stoppt ein Fahrzeug beispielsweise plötzlich während der Fahrt, muss dieser Zustand unmittelbar visualisiert werden, sodass schnell reagiert und ein längerer Stillstand vermieden werden kann.

Systemanbindung

Das Dashboard soll über Schnittstellen zur Anbindung an bestehende operative Systeme verfügen, insbesondere an das Navigationssystem der AGVs sowie an das Warehouse-Management-System (Anforderung 4.3). Es soll deren Daten

nutzen, um aktuelle Informationen wie Fahrzeugpositionen, Auftragsstatus oder Lagerplatzbelegungen in Echtzeit zu visualisieren. Dadurch soll sichergestellt werden, dass das Dashboard auf verlässliche und aktuelle Informationen zugreift, ohne redundante Datenerfassung erforderlich zu machen.

6 Konzeption des Prototyps

Ziel dieses Kapitels ist es, die grundlegenden Überlegungen und strukturellen Entscheidungen darzulegen, die das Fundament des geplanten Dashboards bilden. Dafür wurden die im Vorhinein aufgestellten Anforderungen an das System in technische Modelle überführt. Im Folgenden werden die Funktionsmodellierung, die Prozessmodellierung, die Gestaltung der Benutzeroberfläche, sowie die Datenmodellierung näher betrachtet.

6.1 Funktionsmodellierung

Abgeleitet aus den Anforderungen (vgl. Kapitel 5) wurde zur Veranschaulichung der Funktionen ein Use-Case-Diagramm (dargestellt in Abbildung 2: Use Case Diagramm) entwickelt, welches die zentralen Anwendungsfälle und Rollenbeziehungen innerhalb des Systems abbildet. Das Diagramm basiert auf dem in Kapitel 4 beschriebenen Anwendungsszenario und unterscheidet zwei Hauptnutzergruppen: Lagermitarbeitende und Lagerleitung. Während die Lagerleitung grundsätzlich zu allen Funktionen des Systems berechtigt sein soll, unterliegen Lagermitarbeitende gewissen Einschränkungen. Im Folgenden werden zunächst die Use Cases aus Sicht der Lagermitarbeitenden dargestellt, bevor auf die exklusiven Funktionen der Lagerleitung eingegangen wird.

Lagermitarbeitende sollen sich die Übersichtskarte des Lagers, die Fahrzeugliste sowie die Transportauftragsliste anzeigen lassen können. Diese drei Hauptelemente bilden den vorgesehenen Einstiegspunkt für alle weiteren Funktionalitäten. Wird die Kartenansicht aufgerufen, sollen die Nutzer die Echtzeitpositionen aller Fahrzeuge im Lager sehen können. Zusätzlich soll es möglich sein, durch einen Klick auf ein Fahrzeugsymbol die zugehörige Route anzuzeigen sowie bei Bedarf anzupassen. Über die Fahrzeug- oder Auftragsliste sollen die Lagermitarbeitenden umfassende Informationen zu den Fahrzeugen und Aufträgen erhalten. Dabei sollen beide Listen gezielt über eine Filterfunktion nach bestimmten Eigenschaften durchsucht werden können. In der Transportauftragsliste sollen einzelne Aufträge priorisiert werden können, um bei operativen Änderungen flexibel reagieren zu können. Über die Fahrzeugliste wiederum sollen konkrete Aktionen für ausgewählte Fahrzeuge ausführbar sein, beispielsweise das Aktivieren und Deaktivieren von Fahrzeugen oder deren Anzeigen auf der Karte.



Abbildung 2: Use Case Diagramm

Die Lagerleitung soll auf alle bereits genannten Funktionen zugreifen können und darüber hinaus Zugriff auf zwei exklusive Funktionen erhalten. Zum einen soll nur die Lagerleitung Fahrzeuge aus dem System löschen können. Diese Funktion soll ebenfalls über die Fahrzeugliste zugänglich sein. Zum anderen soll die Lagerleitung Zugang zu einer separaten Statistikseite haben. In dieser werden Lagerkennzahlen und langfristige Trends in Form von Tabellen und Diagrammen aufbereitet dargestellt. Diese Funktionen sollen die Lagerleitung bei der strategischen Analyse und Entscheidungsunterstützung unterstützen.

6.2 Mockups zur Gestaltung der Benutzeroberfläche

Zur Visualisierung der geplanten Benutzeroberfläche des mobilen Dashboards wurden Mockups erstellt. Neben der nachvollziehbaren Darstellung von Designentscheidungen dienen sie als Grundlage für die technische Umsetzung des Frontends. Um die Anforderung der Mobilgerätekompatibilität zu erfüllen, soll es neben der Desktop-Darstellung auch eine Mobilgeräte-Ansicht geben.

In der Desktop-Ansicht soll die Startseite einer dreigeteilten Struktur folgen, welche Übersichtlichkeit mit direktem Zugriff auf die Elemente kombiniert (siehe Abbildung 3: Mockup der Startseite (Desktop-Ansicht)). Im Zentrum steht die Karte des Lagers, mit einer Darstellung der autonomen Fahrzeuge sowie deren aktuellen Routen in Echtzeit. Sie soll den Zustand des Lagers auf einen Blick sichtbar machen und eine visuelle Orientierung im komplexen industriellen Umfeld bieten. Auf der rechten Seite soll sich eine reduzierte Detailansicht der Fahrzeuge in Form einer aufklappbaren Liste befinden. Auf dieser soll der Nutzer die Detailinformationen zu dem ausgewählten Fahrzeug einsehen können, sowie die Möglichkeit haben ein Fahrzeug zu stoppen oder dessen Route anzupassen. Auf der linken Seite des Bildschirms soll eine Navigationsleiste integriert werden, welche eine schnelle Navigation zu den Hauptseiten des Dashboards ermöglicht. Darunter soll sich der Button zum Ausloggen und eine Verlinkung zu einer Profilübersicht befinden. Zudem sollen am oberen Bildschirmrand eine Mitteilungsanzeige und die Weiterleitung zu der Statistikseite integriert werden.

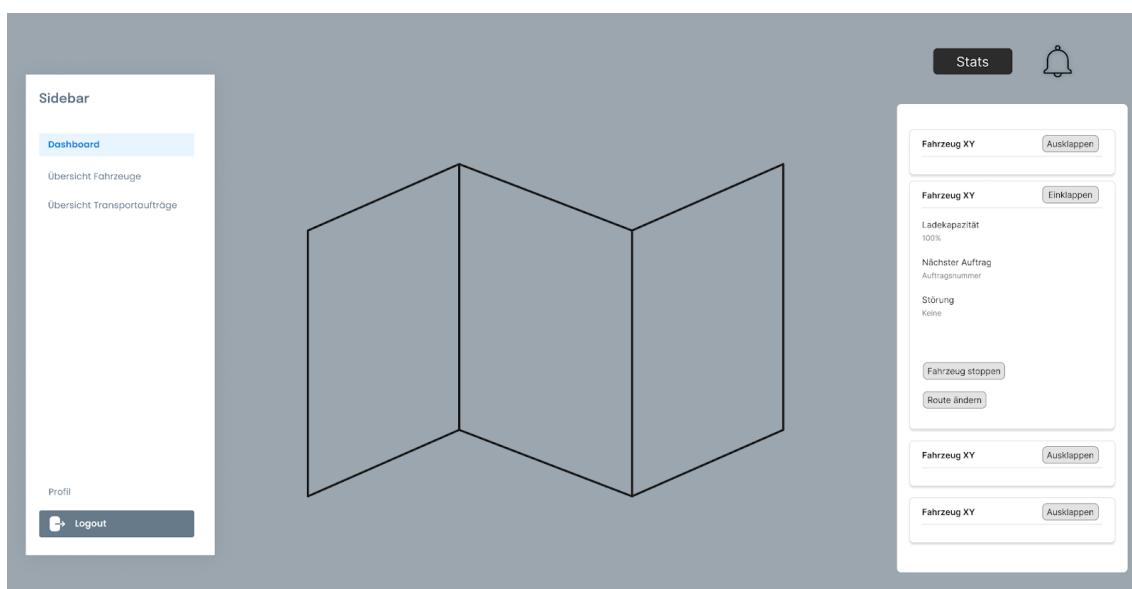


Abbildung 3: Mockup der Startseite (Desktop-Ansicht)

Neben der Startseite soll es eine Übersichtsseite der Fahrzeuge geben, auf der sich eine Liste aller Fahrzeuge im System (Abbildung 4: Mockup der Fahrzeugübersicht (Desktop-Ansicht)) befindet. Hier werden alle im System gespeicherten Fahrzeuge, sowie deren Eigenschaften dargestellt. Um zu den Steuerungsfunktionen für die Fahrzeuge zu kommen, soll das konkrete Fahrzeug ausgewählt werden können. Zudem soll es einen Filter geben, um die angezeigten Fahrzeuge einzuschränken und nach spezifischen Fahrzeugen zu suchen. Analog dazu soll eine Transportauftragsübersicht aufgebaut sein.

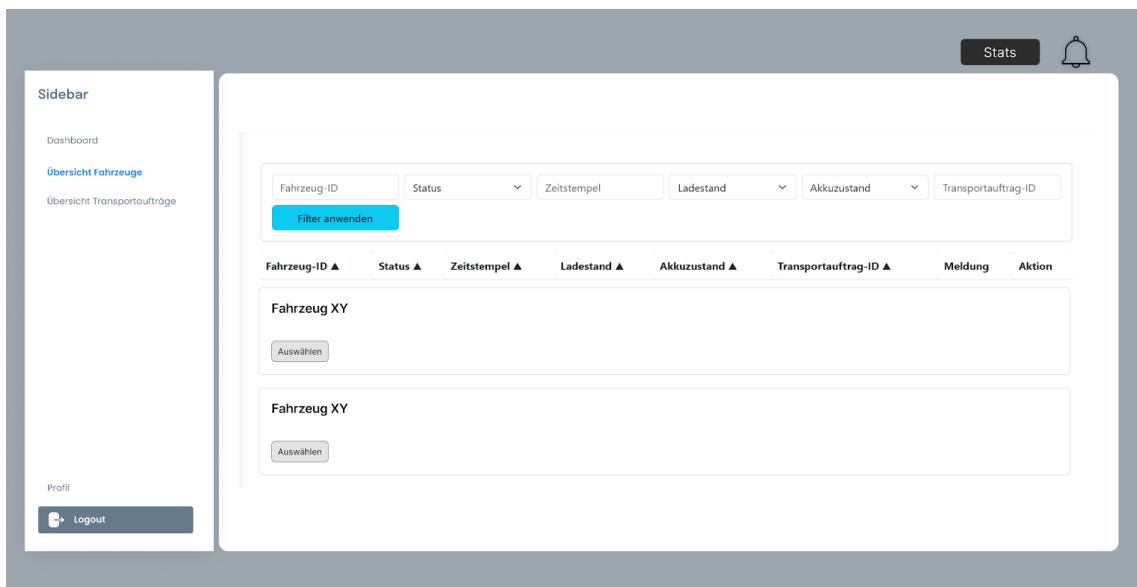


Abbildung 4: Mockup der Fahrzeugübersicht (Desktop-Ansicht)

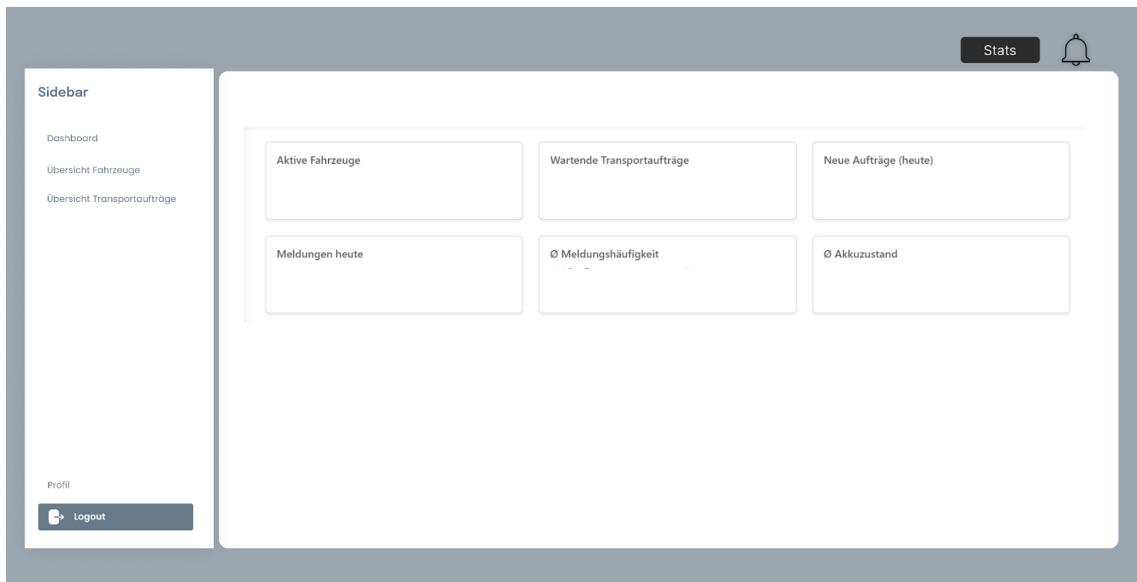


Abbildung 5: Mockup der Statistikseite (Desktop-Ansicht)

Für die Lagerleitung wird zusätzlich zu den bereits gezeigten Systembereichen die Statistikseite zur Verfügung stehen (siehe Abbildung 5: Mockup der Statistikseite (Desktop-Ansicht)). Hier sollen Kennzahlen über das Lager und die eingesetzten autonomen Transportfahrzeuge gezeigt werden. Über Kacheln sollen die wichtigsten Werte auf einen Blick übersichtlich dargestellt werden. Darunter sollen weitere Auswertungsformen und Visualisierungsarten, wie z.B. Tabellen oder Graphen, Platz finden.

Um eine mobile Ansicht zu ermöglichen, muss sich die Anordnung der Elemente verändern (siehe Abbildung 6: Mockup der mobilen Seite). So soll die Navigationsleiste an den unteren Bildschirmrand verschoben werden, wo sie leicht mit dem Finger erreicht werden kann. Auf der Startseite soll die reduzierte Detailansicht der Fahrzeuge ebenfalls im unteren Bildschirmbereich integriert. Die Darstellung der Fahrzeug- und Auftragsliste soll unterschiedlich zu der Desktop-Ansicht sein. Hier werden die jeweiligen Eigenschaften nicht direkt, sondern erst bei Auswahl eines Fahrzeuges angezeigt, um den Bildschirm nicht zu überladen.

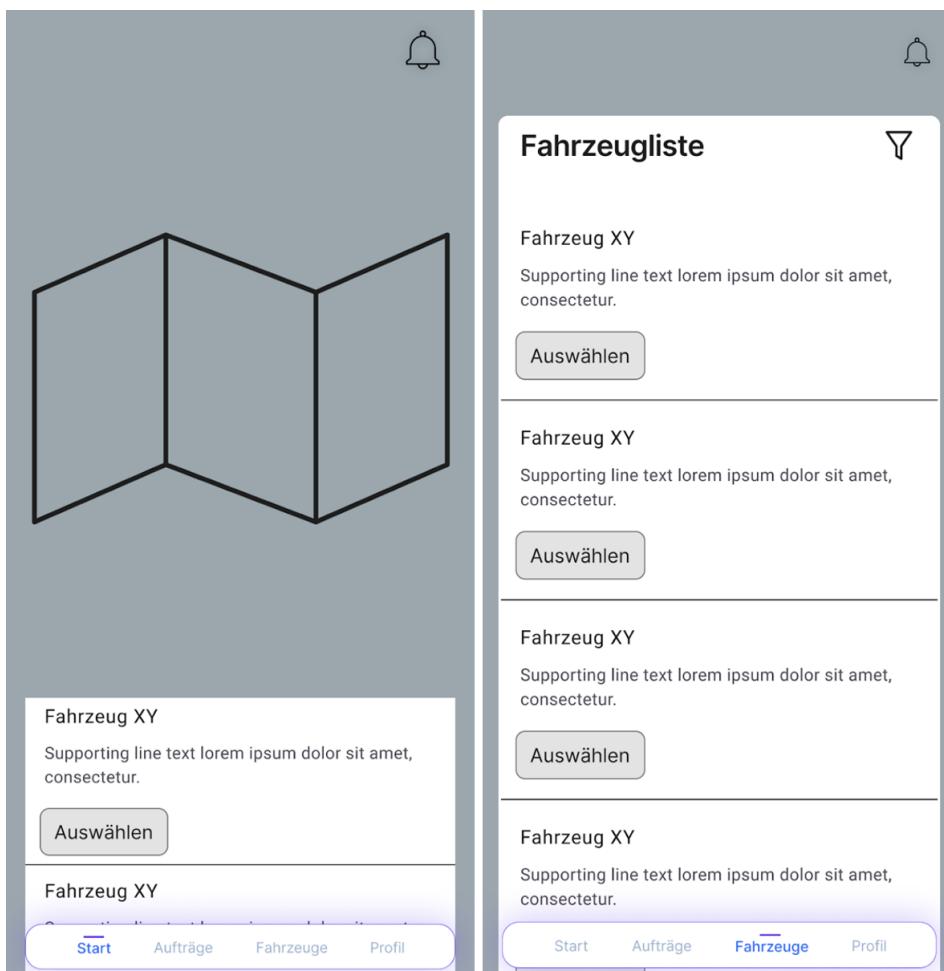


Abbildung 6: Mockup der mobilen Seite

6.3 Prozessmodellierung

Aufbauend auf den zuvor beschriebenen Use Cases wurden zentrale Systemprozesse in Form von Aktivitätsdiagrammen modelliert. Diese sollen die konkrete Prozesslogik der Use Cases darstellen und die Abläufe im System nachvollziehbar machen. Im Folgenden werden hier exemplarisch drei Use Cases in Form von Aktivitätsdiagrammen dargestellt und beschrieben.

Ein erster zentraler Anwendungsfall ist die Priorisierung von Transportaufträgen, das entsprechende Aktivitätsdiagramm ist Anhang C.1, Abbildung 17: Aktivitätsdiagramm zur Priorisierung von Aufträgen zu entnehmen. Zur Priorisierung soll der Nutzer das Dashboard öffnen und zu der Transportauftragsliste unter „Übersicht Transportaufträge“ navigieren. Anschließend kann die Liste gefiltert werden, um den gesuchten Transportauftrag zu finden. Falls kein Treffer erscheint, soll der Nutzer den Filter anpassen oder den Vorgang abbrechen können. Wird der gesuchte Transportauftrag gefunden, soll die Person den Transportauftrag auswählen können und somit Zugriff auf dessen Steuerungsfunktionen erhalten. Hier soll der Nutzer den Transportauftrag priorisieren können, sodass er an oberster Stelle der Warteschlange nicht bearbeiteter Aufträge erscheint.

Ein weiterer Use Case ist, sich die Route eines Fahrzeugs anzeigen zu lassen (siehe Anhang C.2, Abbildung 18: Aktivitätsdiagramm Route anzeigen). Dafür soll der Nutzer das Dashboard öffnen und vor der Entscheidung stehen, das Fahrzeug, dessen aktuelle Route angezeigt werden soll, direkt auf der Karte auszuwählen, oder über die Fahrzeugliste zu suchen. Wenn der Mitarbeitende das Fahrzeug auf der Karte auswählt, soll die aktuelle Route des Fahrzeugs auf der Karte angezeigt werden und der Prozess endet. Wenn der Nutzer das Fahrzeug nicht auf der Karte lokalisieren kann, weil er z.B. nur eine Transportauftrags_ID als Referenz hat, soll die Fahrzeugliste über den bereits beschriebenen Prozess durchsucht werden können. Wurde das Fahrzeug auf diese Weise gefunden, soll es ausgewählt und über die Steuerungsfunktionen auf der Karte angezeigt werden können. Nun soll das Dashboard auf die Startseite wechseln und die Route des Fahrzeugs anzeigen.

Als letzter Use Case wird das Anpassen einer Fahrtroute beschrieben (siehe Anhang C.3, Abbildung 19: Aktivitätsdiagramm Route anpassen). Dieser ist eng mit dem Prozess hinter Route anzeigen verknüpft. Um eine Route anzupassen, soll

im ersten Schritt die Route angezeigt werden. Wenn eine Route angezeigt wird, soll auch die reduzierte Detailansicht des Fahrzeugs geöffnet sein. Hier soll dem Nutzer nun eine spezifische Steuerungsfunktion „Route anpassen“ zur Verfügung stehen. Beim Auswählen der Funktion soll sich eine Ansicht öffnen, auf der alle Routenpunkte im Lager verzeichnet sind. Diese Routenpunkte sollen nun einzeln ausgewählt werden können. Nach dem Speichern der neuen Route soll diese im System hinterlegt sein und das Fahrzeug soll dieser folgen.

6.4 Datenmodellierung

Im Rahmen der Systemkonzeption wurde ein Entity-Relationship-Modell (ERM) konzipiert (siehe Anhang D, Abbildung 20: ERM zur Datenmodellierung) welches als Grundlage für die Entwicklung der relationalen Datenbanken dienen soll. Dafür wurden sechs Entitäten ermittelt: Transportauftrag, Transportfahrzeug, Lagerplatz, Routenpunkt, Route, sowie Fehlermeldung.

Die Entität Transportauftrag beschreibt einen konkreten Transportauftrag innerhalb des Lagers. Dieser ist eindeutig identifizierbar durch eine ‘Auftrags_ID’. Die Entität ist über den Fremdschlüssel ‘Fahrzeug_ID’ mit der Entität Transportfahrzeug und über den kombinierten Fremdschlüssel ‘Lagerplatz_Start_ID & Lagerplatz_Ziel_ID’ mit der Entität Lagerplatz verbunden. Daneben enthält sie das Attribut ‘Status’, welches angibt, ob ein Transportauftrag bearbeitet wird, in der Warteschlange, sowie das Attribut ‘Priorität’, welches angibt, ob ein Auftrag priorisiert ist.

Die Entität Transportfahrzeug speichert die verschiedenen AGVs, welche sich in dem Lager bewegen. Diese besitzen eine eindeutige ‘Fahrzeug_ID’, sowie den Fremdschlüssel ‘Auftrags_ID’. Dabei bearbeitet ein Transportfahrzeug N Transportaufträge, ein Transportauftrag wird jedoch immer von einem Transportfahrzeug bearbeitet. Als Attribute hat die Entität ‘Status’, ‘Zeitstempel’, ‘Fahrzeugtyp’, ‘Akkuzustand’, ‘Ladestand’, ‘Meldung’ sowie ‘X-Koordinate’ und ‘Y-Koordinate’. Der ‘Status’ beschreibt dabei, ob ein Fahrzeug aktiv oder inaktiv ist, eine Störung hat oder außer Betrieb ist. Der ‘Zeitstempel’ gibt den Zeitpunkt der letzten Rückmeldung des Fahrzeugs an, ‘Akkuzustand’ und ‘Ladestand’ enthalten Informationen zu dem internen Zustand der Fahrzeuge und ‘Typ’ gibt den Fahrzeugtypen wieder. ‘Meldung’ hält den aktuellen Eintrag zum Fahrzeug fest, der eine der fol-

genden Ausprägungen haben kann: Keine Rückmeldung, Route blockiert, Akku-zustand schlecht, technische Wartung, lädt oder keine. Der aktuelle Standort der Fahrzeuge wird über die ‘X-Koordinaten’ und ‘Y-Koordinaten’ gespeichert.

Die Entität Fehlermeldung speichert die Fehlermeldungen, welche bei Betrieb des Lagers auftreten. Ein Fahrzeug kann beliebig viele Fehlermeldungen erzeugen, eine Fehlermeldung wird jedoch immer von nur einem Fahrzeug erzeugt. Jede Fehlermeldung hat eine eindeutige ‘Fehler_ID’ und ist einem Fahrzeug über dessen ‘Fahrzeug_ID’ zugeordnet. Zudem hält sie die Information ‘Zeitstempel’, welcher den Zeitpunkt des Auftretens speichert, sowie den ‘Typ’ einer Fehlermeldung und eine ‘Beschreibung’ der Fehlermeldung.

Die Entität Fahrroute ist mit der Entität Transportfahrzeug verbunden. Da jeder Transportauftrag eine neue Route generiert, kann ein Fahrzeug N Fahrtrouten befahren, während eine Fahrroute immer von einem Fahrzeug genutzt wird. Dabei ist eine Fahrroute aus mehreren Routenpunkten aufgebaut. Diese Routenpunkte sind wiederum Teil von mehreren Fahrtrouten. In der Implementation muss also eine eigene Relation für diese Beziehung erstellt werden. Die Entität Routenpunkt speichert die verschiedenen ‘X-Koordinaten’ und ‘Y-Koordinaten’. Diese sind durch eine eindeutige ‘Punkt_ID’ identifizierbar. Ebenso ist die Entität Fahrtroute eindeutig über den Primärschlüssel ‘Fahrtrouten_ID’ identifizierbar, enthält den Fremdschlüssel ‘Fahrzeug_ID’ und speichert den kombinierten Fremdschlüssel des ‘Lagerplatz_Start_ID’ und ‘Lagerplatz_Ziel_ID’, über welchen sie mit der Entität Lagerplatz verbunden ist.

Jeder Lagerplatz kann dabei Teil von mehreren Routen und von mehreren Transportaufträgen sein. Die Entität ist eindeutig über ihre ‘Lagerplatz_ID’ identifizierbar und besitzt die Attribute ‘Anzahl_frei’ und ‘Anzahl_belegt’, welche die Belegungsinformationen für den Lagerplatz enthalten, ‘X-Koordinate’ und ‘Y-Koordinate’, sowie ‘Lagerplatztyp’, welche angibt, ob es sich um einen Lagerplatz im Hochregallager, im Hilfsregal des Wareneingangs oder Warenausgangs oder um einen bereitstellplatz handelt.

7 Implementierung des Prototyps

Nach Aufstellung der Anforderungen für das fiktive Szenario und der Konzeption erfolgt die Implementation. Dazu sollen zunächst die verwendeten Technologien und die Struktur der genutzten Datenbank kurz dargestellt werden. Anschließend erfolgt die Erläuterung der Implementierung des Prototyps. Dabei wird zunächst die Umsetzung der Anforderungen und anschließend die Umsetzung des fiktiven Szenarios beschrieben.

7.1 Überblick über die verwendeten Technologien und Frameworks

Das **Model-View-Controller-Schema (MVC-Schema)** ist ein Architekturmuster für die Strukturierung von Webanwendungen. Es trennt die Datenlogik (Model), die Präsentation (View) und die Steuerung der Benutzerinteraktion (Controller) voneinander. Dadurch wird die Anwendung übersichtlicher, modularer und besser wartbar (vgl. Krasner et al. 1988).

Bei **Laravel** handelt es sich um ein modernes MVC-Framework für PHP-basierte Webanwendungen. **Laravel Blade** ist die schlanke, aber leistungsfähige Template-Engine von Laravel. Sie erlaubt die Einbettung von PHP-Logik direkt in .blade.php-Dateien und kompiliert diese beim ersten Aufruf zu puren PHP-Klassen, die anschließend gecacht werden. Dadurch entsteht praktisch kein Laufzeit-Overhead, während Entwickler dennoch mit komfortablen Direktiven wie @if, @foreach oder Layout-Vererbung arbeiten können (vgl. Laravel 2025a). Mit Hilfe von **Laravel Eloquent** bietet Laravel einen objektrelationalen Mapper (ORM), wodurch die Arbeit mit einer relationalen Datenbank vereinfacht werden kann. Eloquent verknüpft jede Datenbanktabelle mit einem dazugehörigen Modell, wodurch, beispielsweise in der Kombination mit MySQL, auf die SQL-Queries verzichtet werden kann (vgl. Laravel 2025b). **Laravel Breeze** ist das minimalistische Starter-Kit von Laravel, das alle grundlegenden Authentifizierungs-Funktionen (Login, Registrierung, Passwort-Reset, E-Mail-Verifizierung, Passwort-bestätigung sowie eine einfache Profilseite) in wenigen Artisan-Befehlen bereitstellt (vgl. Laravel 2025c).

Leaflet ist eine Open-Source-JavaScript-Bibliothek, mit deren Hilfe interaktive Karten erstellt werden können. Benutzerdefinierte Karten können hinterlegt und wissenswerte Informationen beispielsweise als Pop-Ups auf der Karte umgesetzt werden (vgl. Agafonkin 2025).

Bei **MySQL** handelt es sich um ein Open-Source-Relationales-Datenbankmanagementsystem. In MySQL können die Daten in SQL-basierten relationalen Datenbanken strukturiert gespeichert und verwaltet werden. MySQL gilt als sehr zuverlässig, gut skalierbar und einfach mit Laravel zu verknüpfen (vgl. Oracle 2025).

PHP ist eine freie serverseitige Skriptsprache, die speziell für die Webentwicklung entworfen wurde. Sie erzeugt dynamische HTML-Ausgaben, kann Formulardaten verarbeiten, Cookies senden / empfangen und eignet sich ebenso für Kommandozeilen-Skripte. Durch ihre Flexibilität und den geringen Einarbeitungsaufwand treibt PHP von kleinen Blogs bis zu stark frequentierten Webseiten einen Großteil des modernen Webs an (vgl. The PHP Group 2025).

Bootstrap 5 ist ein quelloffenes Frontend-Toolkit, das ein responsives Grid-System, vorgefertigte CSS-Klassen und JavaScript-Plugins für Komponenten wie Modals, Dropdowns oder Toasts bereitstellt. Mit Sass-Variablen lässt sich das Design anpassen, und die mitgelieferten Utility-Klassen ermöglichen schnelles Prototyping bis hin zu produktionsreifen Oberflächen (vgl. Bootstrap Team 2025).

jQuery ist eine kleine, schnelle JavaScript-Bibliothek, die DOM-Traversierung, Ereignisbehandlung, Animationen und Ajax-Aufrufe über eine einheitliche API vereinfacht. Sie gleicht Browser-Inkonsistenzen aus und hat mit ihrer einfachen Syntax maßgeblich beeinflusst, wie JavaScript im Web eingesetzt wird (vgl. OpenJS Foundation 2025).

SortableJS ist eine leichtgewichtige Bibliothek, die per Drag-and-Drop sortierbare Listen auf modernen Browsern und Touch-Geräten ermöglicht. Sie kommt ohne jQuery aus, unterstützt Klonen, das Verbinden mehrerer Listen und Erweiterungen wie MultiDrag oder Swap (vgl. Mills 2025).

Die **Fetch API** ist die moderne JavaScript-Schnittstelle, mit der Web-Apps HTTP-Anfragen versenden und Antworten verarbeiten können, ganz ohne das ältere XMLHttpRequest-Objekt. Sie arbeitet Promise-basiert, erlaubt GET-, POST-, PUT- oder DELETE-Aufrufe und integriert sich nahtlos mit Funktionen wie CORS oder Service-Workern. Wird der Rückgabewert der Fetch-Methode mit `.json()` verarbeitet, entsteht ein Ajax-Workflow, bei dem das Frontend JSON-Endpunkte eines REST-Backends anspricht und die erhaltenen Objekte direkt weiterverwen-

det. So lassen sich beispielsweise periodisch aktuelle Fahrzeug-Koordinaten abrufen oder veränderte Auftragsdaten an den Server senden, ohne die Seite neu zu laden (vgl. Gupta 2023; MDN 2025a).

CSS ist die Stylesheet-Sprache, mit der das Aussehen von HTML- oder XML-Dokumenten beschrieben wird. Layout-Eigenschaften wie Seitenraster, Abstände, Farben, Schriften oder Animationen werden deklarativ in Stylesheets festgelegt und vom Browser auf alle passenden Elemente „kaskadierend“ angewandt. Damit trennt CSS Präsentation konsequent von Struktur, ermöglicht responsive Designs für verschiedene Geräte und spart durch wiederverwendbare Regeln viel Pflegeaufwand (vgl. MDN 2025b).

Carbon ist eine objektorientierte PHP-Bibliothek, die die native DateTime-Klasse erweitert und dadurch komfortable Methoden für Datums- und Zeitberechnungen bietet. Entwickler können damit Zeiträume addieren oder subtrahieren, menschlich lesbare Unterschiede erzeugen („vor 3 Stunden“) ohne selbst komplexe Format-Strings zusammenstellen zu müssen. Carbon bleibt vollständig kompatibel zur DateTime-Basis, sodass vorhandener Code weiterfunktioniert, bietet aber einen weit größeren Funktionsumfang bei nahezu identischer Performance (vgl. Carbon 2025).

7.2 Datenhaltung des Prototyps

Die relationale Datenbank wurde mit MySQL umgesetzt. Für die Interaktion zwischen Anwendung und Datenbank wird das Laravel-Framework mit dem ORM „Eloquent“ verwendet, wodurch der Zugriff auf die Datenbank deutlich vereinfacht wird. Als konzeptionelle Grundlage für die Struktur der Datenbank diente das in Kapitel 6.4 ausgearbeitete ERM. Eine detaillierte Beschreibung der Entitäten und ihrer Attribute findet sich dort. Um Code-Konventionen des Frameworks einzuhalten und die Entwicklung zu beschleunigen, wurden die Bezeichner einiger Entitäten und Attribute leicht angepasst. Diese Anpassungen betreffen ausschließlich die Benennung, nicht jedoch die fachliche Struktur des Modells. Im Folgenden werden die in der Datenbank umgesetzten Relationen Fahrzeuge, Lagerplatz, Aufträge, Meldung, Routen, Routenpunkte, Routen_Routenpunkte sowie User beschrieben. Eine erneute Auflistung der Attribute und Beziehungen wird nur vorgenommen, wenn zusätzliche oder vom ERM abweichende Informationen vorliegen.

Die Transportfahrzeuge werden in der Tabelle Fahrzeuge gespeichert. Sie enthält die Attribute 'fahrzeug_id' als Primärschlüssel, 'transportauftrag_id', 'type', 'status', 'zeitstempel', 'ladestand', 'akkuzustand', 'meldung', 'x' und 'y'. Mit den Attributen 'status' und 'meldung' wird im Störungsfall die Benachrichtigung von Systemkomponenten ermöglicht.

Die Informationen zu den Transportaufträgen sind in der Tabelle Aufträge hinterlegt. Diese umfasst die Attribute 'transportauftrag_id' als Primärschlüssel, 'fahrzeug_id', 'status', 'prioritaet', 'startort_id' und 'zielort_id'.

Die Tabelle Lagerplatz speichert Informationen zu allen Lagerorten. Sie enthält die Attribute 'lagerplatz_id' als Primärschlüssel, 'x', 'y', 'hoehe', 'breite', 'type', 'anzahl' und 'belegt'. Die Attribute 'hoehe' und 'breite' beschreiben die räumliche Ausdehnung des Lagerplatzes auf der Karte und dienen der grafischen Darstellung im Frontend.

Die Fehlermeldungen werden in der Tabelle Meldung verwaltet. Diese umfasst die Attribute 'meldung_id', 'fahrzeug_id', 'typ', 'beschreibung' und 'gemeldet_am'. Jede Meldung ist einem Fahrzeug zugeordnet.

Zur Abbildung von Routen werden die Tabellen Routen, Routenpunkte und Routen_Routenpunkte verwendet. In Routen werden die Attribute 'routen_id' als Primärschlüssel sowie 'fahrzeug_id', 'zielort_id' und 'startort_id' gespeichert. Die Tabelle Routenpunkte beschreibt die einzelnen Punkte, aus denen sich eine Route zusammensetzt, und enthält die Attribute 'routenpunkte_id', 'x' und 'y'. Die Zuordnung der Routenpunkte zu einer Route wird über die Verknüpfungstabelle Routen_Routenpunkte realisiert, welche als Pivot-Tabelle die N:M-Beziehung zwischen Routen und Routenpunkten abbildet. Sie speichert zusätzlich die Reihenfolge der Punkte einer Route.

Die Tabelle User wird zur Verwaltung von Benutzern und deren Authentifizierung verwendet. Sie war im ursprünglichen ERM nicht enthalten, da sie keine direkten Beziehungen zu den übrigen Entitäten aufweist. In der Implementierung besitzt ein Eintrag in User die Attribute 'id', 'name', 'email', 'password' und 'role'. Das Passwort wird aus Sicherheitsgründen ausschließlich als Hashwert gespeichert. Bei den Rollen wird zwischen 'Lagerleitung' und 'Lagermitarbeiter' unterschieden.

7.3 Umsetzung der Anforderungen

7.3.1 Visualisierung

Die Benutzeroberfläche bildet den zentralen Zugang zur Anwendung. Sie wurde so gestaltet, dass Nutzende je nach Endgerät und zugewiesener Rolle effizient durch die verschiedenen Bereiche geführt werden. Im Folgenden soll die Umsetzung der Anforderungen an die Visualisierung dargestellt werden.

Das Layout orientiert sich an modernen Webanwendungen im Dashboard-Stil, mit klar getrennten Bereichen für Navigation, Inhalte und Steuerungselemente.

Grundlage bildet eine zentrale Layout-Datei, in der feste Bestandteile wie die Hauptnavigation, die Top-Bar (siehe Abschnitt Navigation) sowie globale Scripts definiert sind. Mithilfe des Blade-Templating-Systems von Laravel basieren die einzelnen Seiten auf diesem zentralen Layout, wobei der jeweilige Inhalt an den vorgesehenen Stellen eingefügt wird. Wiederkehrende Elemente wie Listen oder die Kartenansicht wurden ebenfalls mithilfe des Blade-Templating-Systems in kleine, wiederverwendbare Bausteine (Partials) ausgelagert. Dies erhöht nicht nur die Übersichtlichkeit im Code, sondern verbessert auch die Wartbarkeit und die spätere Erweiterbarkeit der Anwendung. Routen werden in web.php registriert, Controller holen über Eloquent-ORM die Datensätze aus der Datenbank und geben sie an Blade-Templates weiter. Blade rendert das erste HTML. Alle weiteren Interaktionen passieren im Browser mit jQuery 3. Bootstrap 5 liefert das Basis-CSS und Icons, während kleine JavaScript-Bibliotheken wie Leaflet für die Landkarte und SortableJS für Drag-and-Drop zuständig sind.

Profil- und Rollenverwaltung

Die Profile der Mitarbeitenden werden von der IT-Abteilung registriert und verwaltet. Die Mitarbeitenden melden sich mit ihrer E-Mail und dem Passwort an, das sie erhalten haben. Sie können ihr Passwort ändern und sich abmelden. Zudem soll das System zwischen verschiedenen Benutzerrollen (Lagermitarbeitende und Lagerleitung) mit unterschiedlichen Zugriffsrechten unterscheiden (Anforderung 1.8). Die Seiten sind durch Laravel Breeze mit der auth-Middleware geschützt, um nicht eingeloggte oder unbefugte Benutzer vom Zugriff auf die Anwendung oder bestimmte Funktionen und Seiten auszuschließen.

Navigation

Für die Hauptnavigation wurde eine Navigationsleiste umgesetzt, die auf Desktop-Geräten am linken Rand als Sidebar (siehe Abbildung 7: Startseite in der Desktopansicht) angezeigt wird und sich auf Mobilgeräten in eine Bottom-Bar (siehe Abbildung 14: Startseite in der mobilen Ansicht) verwandelt. Es handelt sich dabei um dieselbe Leiste, die sich responsiv dem Gerät anpasst und den schnellen Wechsel zwischen den Hauptbereichen der Anwendung ermöglicht. Ergänzt wird sie durch eine Top-Bar, die die Benachrichtigungen enthält. Zusätzlich befindet sich dort ein Button zur Statistikseite, der ausschließlich in der Rolle „Leitung“ angezeigt wird und für alle anderen Nutzer durch die auth-Middleware geschützt ist.

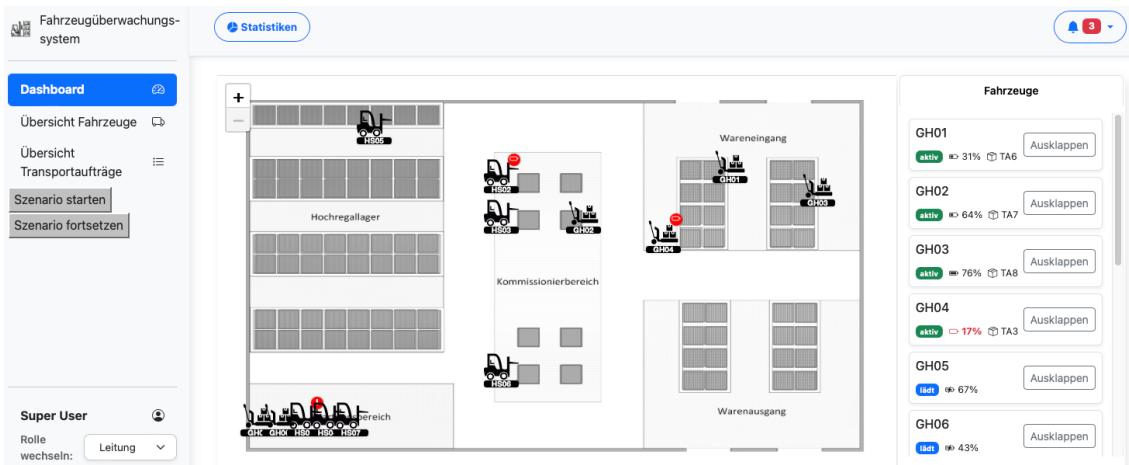


Abbildung 7: Startseite in der Desktopansicht

Lagerkarte

Die Karte ist der wichtigste Bestandteil des Dashboards. Mit Hilfe von Leaflet werden alle wichtigen Informationen über sie visualisiert. Die Karte selbst zeigt das Lagerlayout (siehe Abbildung 8: Lagerkarte).

Beim Klicken auf einen Lagerplatz ist es möglich, dessen ID sowie Belegungsinformationen zu erhalten (Anforderung 1.1). Für die eingesetzten Fahrzeuge sind je nach Fahrzeug unterschiedliche Symbole inklusive ihrer Fahrzeug-ID auf der Karte zu sehen. Hovert man über eines der Fahrzeuge, wird dessen aktueller Status über einen Tooltip angezeigt. Liegt bei einem Fahrzeug eine Störung vor, wird zudem ein rotes Warnsymbol an dem entsprechenden Fahrzeug angezeigt (Anforderungen 1.2, 1.2.1, 1.2.2). Fahrzeuge mit aktiven Meldungen werden in

der Kartenansicht visuell hervorgehoben. Liegt eine Meldung vor (z. B. bei niedrigem Akkustand oder einem technischen Defekt), erscheint ein rotes Warnsymbol direkt am Marker des entsprechenden Fahrzeugs. Zusätzlich zeigt ein Tooltip beim Hovern den genauen Fehlertext an (Anforderung 1.2.3). Die Darstellung basiert auf Meldungsdaten, die im Controller bereitgestellt und dynamisch über Blade und JavaScript in die Karte eingebunden werden.

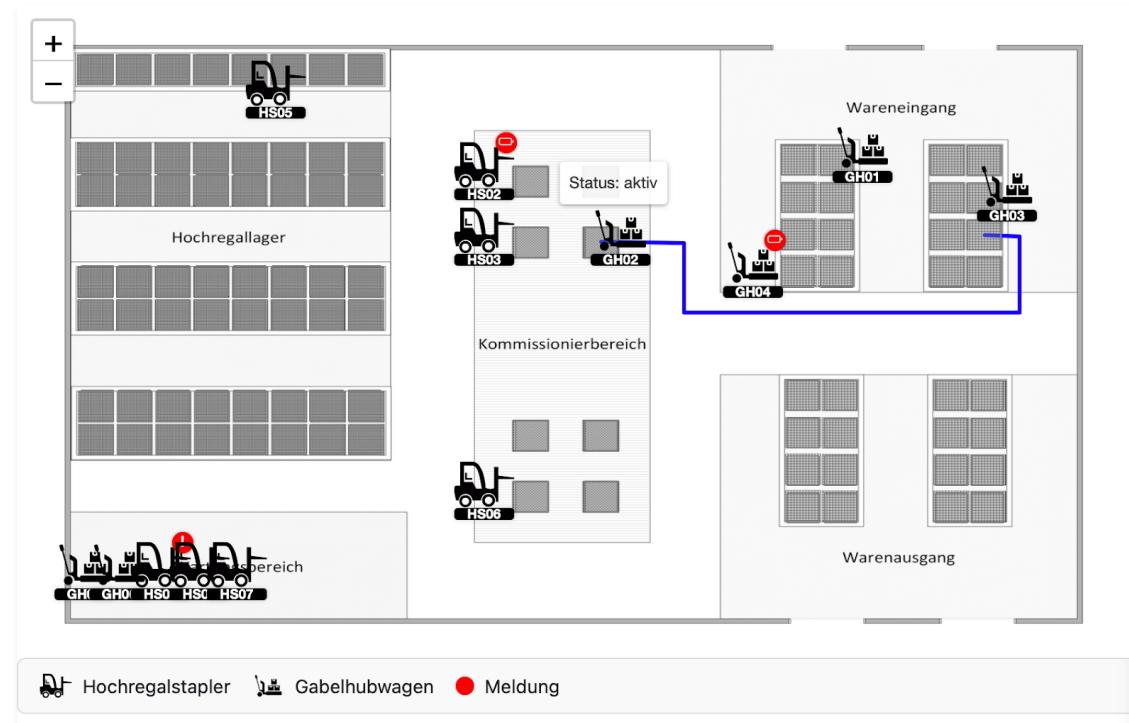


Abbildung 8: Lagerkarte

Durch einen Klick auf ein Fahrzeug wird die Route angezeigt (Anforderung 1.2.4). Gleichzeitig öffnet sich in einer Liste aller Fahrzeuge neben der Karte eine Detailübersicht zu dem ausgewählten Fahrzeug (Anforderung 1.3, siehe Abbildung 9: Ausschnitt aus der Detailansicht der Fahrzeuge). Sowohl die Route als auch die Detailansicht können ebenso per Auswahl in der Fahrzeugliste angezeigt werden.

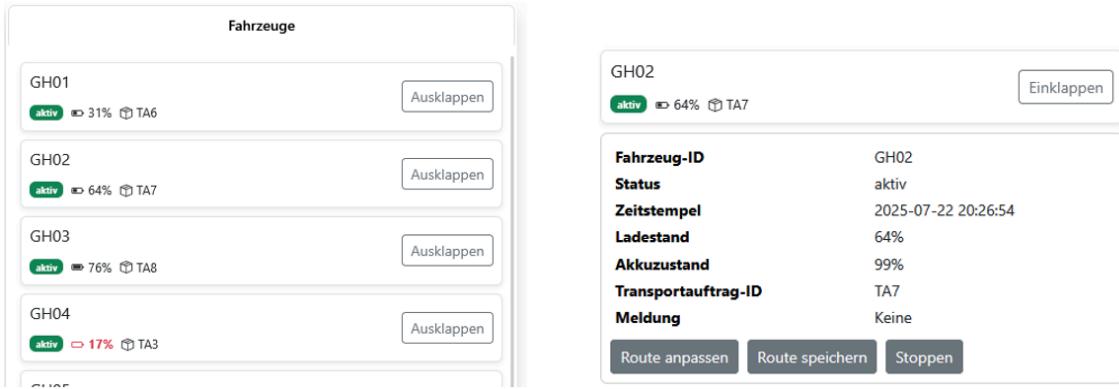


Abbildung 9: Ausschnitt aus der Detailansicht der Fahrzeuge

Fahrzeugübersicht

Neben der Detailansicht der Fahrzeuge neben der Lagerkarte gibt es eine weitere Fahrzeugübersicht mit einer Übersicht über alle Fahrzeuge (mit Informationen zu Fahrzeug-ID, Status, Zeitstempel, Ladestand, Akkustand sowie Transportauftrag-ID) und einer implementierten Filterfunktion (Anforderung 1.5, Abbildung 10: Ausschnitt aus der Fahrzeugübersicht).

Fahrzeug-ID	Status	Zeitstempel	Ladestand	Akkuzustand	Transportauftrag-ID	Meldung	Aktion
Filter anwenden							
GH01	aktiv	2025-07-30 19:28:31	31%	85%	TA6	Keine	Ausklappen
GH02	aktiv	2025-07-30 19:28:31	64%	99%	TA7	Keine	Einklappen
GH03	aktiv	2025-07-30 19:28:31	76%	87%	TA8	Keine	Ausklappen

Abbildung 10: Ausschnitt aus der Fahrzeugübersicht

Für das Filtern liegen oberhalb der Tabelle Eingabefelder mit Platzhaltern. Die Funktion filterFahrzeuge liest diese Werte, prüft jedes Element des Arrays fahrzeugArray und rendert anschließend Tabelle und Kartenansicht neu. Sobald das Ergebnis kleiner als die Gesamtmenge ist, blendet jQuery einen „Filter zurücksetzen“-Knopf ein. Die Tabelle lässt sich zudem sortieren, indem der Nutzer auf einen Spaltenkopf klickt. Das Skript merkt sich Spalte und Richtung, sortiert das Array entsprechend und ruft dieselben Render-Funktionen auf, sodass Filter- und Sortierzustand kombiniert werden. Der Button „Anzeigen“ führt den Nutzer direkt auf die interaktive Karte und zeigt das ausgewählte Fahrzeug inklusive seiner Route an.

Transportauftragsliste

Analog zur Fahrzeugliste wurde eine Transportauftragsliste mit Filterfunktion (Anforderung 1.6, siehe Abbildung 11: Ausschnitt aus der Transportübersicht) implementiert, hier können Transportaufträge priorisiert, bearbeitet und gelöscht werden. Zudem kann durch eine Verknüpfung mit dem Fahrzeug der entsprechende Transportauftrag auf der Karte gezeigt werden.

Transportauftrag-ID	Status	Startort	Zielort	Fahrzeug-ID	Filter anwenden	
Transportauftrag-ID	Status	Priorität	Startort	Zielort	Fahrzeug-ID	Aktion
TA1	wird ausgeführt		BP-1	HR-6-5	HS02	Ausklicken
TA2	wird ausgeführt		BP-2	HR-1-4	HS03	Einklappen
TA3	wird ausgeführt		WE-1-1	BP-6	GH04	Ausklicken

TA2 Priorisieren Löschen Bearbeiten Anzeigen

Abbildung 11: Ausschnitt aus der Transportübersicht

„Bearbeiten“ schaltet die Zeile in den Edit-Modus. Bestätigt der Nutzer die Eingabe der geänderten Daten, werden durch einen AJAX-PUT-Aufruf die Änderungen an die Datenbank ermittelt, der Edit-Modus abgeschaltet und die ursprünglichen Felddaten durch die neuen ersetzt. Klickt man auf „Abbrechen“, werden alle vorgenommenen Änderungen verworfen und der Edit-Modus abgeschaltet. Alle Daten liegen nach dem ersten Seitenaufruf als Arrays im Browser. Die Funktionen Sortieren, Filtern, Aus- und Einklappen stehen ab sofort zur Verfügung. Im Fall einer Änderung wird lediglich ein schlanker JSON-Packet an den Server geschickt und eine knappe Antwort zurücksendet, ohne dass ein schweres Frontend-Framework nötig wäre.

Statistik-Seite

Auf Statistik-Seite (siehe Abbildung 12: Statistik-Seite) werden aktuelle Werte zu Fahrzeugen, Aufträgen und Meldungen dargestellt, sowie weitere Listen und Tabellen, etwa zu Fahrzeugtypen, aktuellen Meldungen und auffälligen Fahrzeugen (mit häufigen Meldungen). Die Tabellen lassen sich per Button ein- und ausklappen (Anforderung 1.7). Die angezeigten und abgeleiteten Werte werden im Controller berechnet und über Blade in der Benutzeroberfläche dargestellt. Dadurch ist die Seite flexibel anpassbar und alle angezeigten Kennzahlen lassen sich leicht verändern, erweitern oder durch neue Faktoren ergänzen.

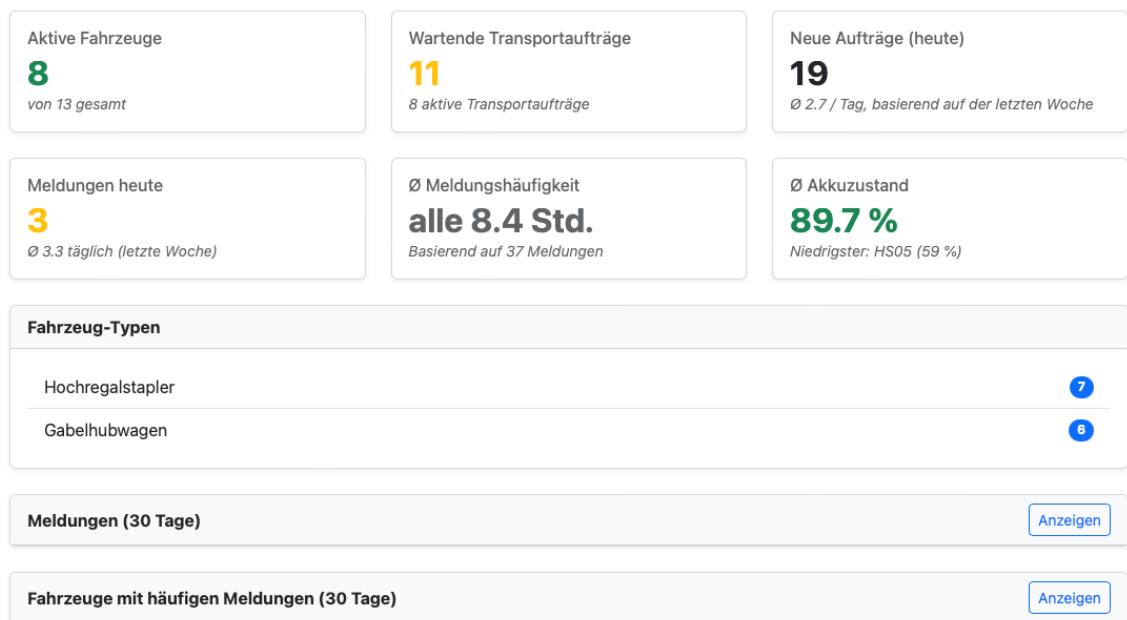


Abbildung 12: Statistik-Seite

7.3.2 Ereignisbehandlung

Um die Reaktionszeiten auf Störungen verkürzen zu können, bildet die Ereignisbehandlung ein zentrales Element. Treten Störungen auf, soll der Nutzer automatisch über diese benachrichtigt werden (Anforderung 2.2). Dazu ist es zunächst notwendig, dass das System die Störungen automatisch erkennt (Anforderung 2.1). Die Meldungen wurden als Benachrichtigungssystem in der Top-Bar umgesetzt (siehe Abbildung 13: Benachrichtigungssystem). Dabei werden aktuelle Meldungen per JavaScript und der Fetch API von einer Laravel-Route abgerufen und im Dropdown angezeigt. Ein Zähler zeigt neue Benachrichtigungen an, zusätzlich erscheinen Live-Popups. Bereits gesehene Meldungen werden mithilfe von localStorage im Browser gespeichert, um Wiederholungen zu vermeiden.

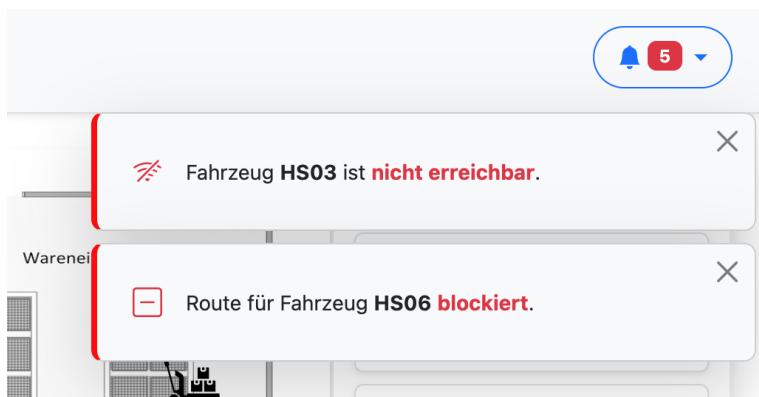


Abbildung 13: Benachrichtigungssystem

7.3.3 Interaktion und Steuerung

Um adäquat auf Störungen reagieren zu können, wurden verschiedene Anforderungen an die Interaktion und Steuerung gestellt, deren Umsetzung im Folgenden dargestellt werden.

Auftragssteuerung und Priorisierung

Der Button „Priorisieren“ in der Transportauftragsübersicht (Abbildung 11: Ausschnitt aus der Transportübersicht) tätigt einen POST-Aufruf. Daraufhin wird im Backend der Prioritätswert erhöht, anschließend wird die Liste neu vom Server abgefragt und clientseitig sortiert. Da allerdings keine realen Daten verfügbar sind, ist die Priorisierung lediglich durch die Hinzufügung und Veränderung der Zahlen in der Tabelle realisiert. (Anforderung 3.1).

Fahrzeugaktivierung

Der Button in der Fahrzeugübersicht (siehe Abbildung 10: Ausschnitt aus der Fahrzeugübersicht) „Aktivieren/Deaktivieren“ sendet einen POST-Aufruf und ändert den Status des Fahrzeugs (Anforderung 3.2).

Manuelle Routenanpassung

Kommt es zu Blockaden, muss die Route eines Fahrzeuges gegebenenfalls angepasst werden. Dazu kann man in der Detailanzeige der Fahrzeuge neben der Lagerkarte (siehe Abbildung 9: Ausschnitt aus der Detailansicht der Fahrzeuge) den Button „Route anpassen“ nutzen. Hierdurch werden die Fahrzeuge ausgeblendet und die Routenpunkte über einen fetch geladen. Die Routenpunkte werden als blaue Punkte dargestellt und das Fahrzeug ist durch einen schwarzen Punkt zu erkennen. Um die Route anzupassen, müssen die Routenpunkte der neuen Route entsprechend angepasst werden. Speichert der Nutzer die Route, werden die ausgewählten Routenpunkte in der Pivot-Tabelle gespeichert sowie eine Erfolgsmeldung an den Nutzer ausgegeben. Danach werden die Fahrzeuge wieder geladen und die Routenpunkte ausgeblendet.

7.3.4 Systemanforderungen

Im Folgenden soll die Umsetzung der im Rahmen dieses Projekts als besonders relevanten bewerteten nicht-funktionalen Anforderungen beschrieben werden. Anzumerken ist, dass die Systemanbindung auf Grund fehlender realer Daten und anderer Systeme nicht umgesetzt wurde (Anforderung 4.3).

Mobilgerätekompatibilität

Die Anwendung wurde responsiv umgesetzt, sodass sie sich an verschiedene Bildschirmgrößen anpasst und sowohl auf Desktop- als auch auf Mobilgeräten nutzbar bleibt (Anforderung 4.1). Als Beispiel soll Abbildung 14: Startseite in der mobilen Ansicht dienen.

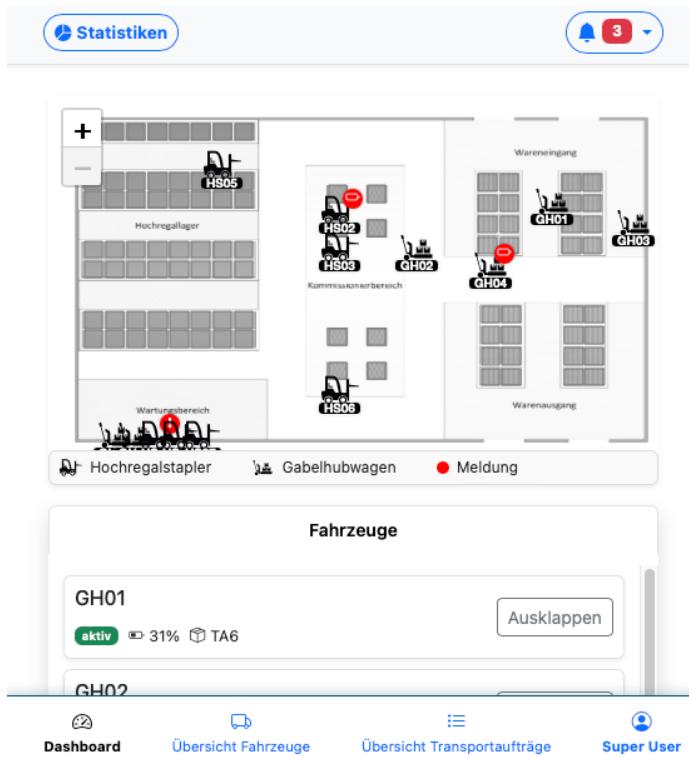


Abbildung 14: Startseite in der mobilen Ansicht

Grundlage hierfür ist das CSS-Framework Bootstrap 5, dessen Grid-System und Utility-Klassen (z. B. `d-none d-lg-flex` zur Steuerung der Sichtbarkeit) eine flexible Layoutgestaltung ermöglichen. Ergänzend kamen eigene CSS-Regeln zum Einsatz, insbesondere für die Darstellung der Kartenansicht. Die Einbindung eines sogenannten Viewport-Tags sorgt zusätzlich dafür, dass die Darstellung auf mobilen Endgeräten korrekt skaliert wird. Zentrale Layoutbereiche wie Navigation und Inhalte passen sich dynamisch an. So wechselt die Navigation beispielsweise von einer Sidebar auf dem Desktop zu einer Bottom Bar auf Mobilgeräten. Auch die Fahrzeug- und Auftragslisten unter „Übersicht Fahrzeuge/Aufträge“ werden auf kleineren Bildschirmen angepasst: Jede Listenkomponente wird dort als Informationskarte dargestellt, die alle Inhalte übersichtlich anzeigt und sämtliche Bedienelemente funktional hält.

Echtzeitaktualisierung

Um alle Informationen über die Fahrzeuge in Echtzeit erhalten zu können, werden diese im Hintergrund, ohne dass es der Benutzer mitbekommt, nachgeladen. Dies wird mit Hilfe einer `fetch`-Funktion realisiert. Die Daten der Fahrzeuge können somit in Echtzeit (Anforderung 4.2) geladen werden. Die Funktion wird mit einem `setInterval` (Zeit in Millisekunden) regelmäßig im Hintergrund nachgeladen. Somit kann der Benutzer alle Informationen abrufen.

7.4 Umsetzung des fiktiven Szenarios

Da keine Live-Daten und keine Schnittstelle zu einem Warehouse-Management-System oder einer Navigationssoftware der AGVs besteht, wurden zum Testen des Prototyps Testdaten für eine Simulation erstellt. Die Funktionen wurden zum Testen des Systems Hard-Coded, also fest im Programm hinterlegt, sodass es nur für das Szenario funktioniert. Die benötigten Testdaten wurden mithilfe eines PHP-Skripts durchgeführt, welches die angelegten Daten in die Relationen einfügt. Sie umfassen alle Fahrzeuge, Lagerplätze, Aufträge, Routen und Routenpunkte. Alle Positionen der Lagerplätze und Routenpunkte, also x- und y-Koordinaten, wurden durch das Testen über eine Drag-and-Drop-Funktion geprüft. Die Daten müssen einmal im Skript händisch angepasst werden und können so bei jedem neuen Laden des Skripts an der gewünschten Position sein. Die Routen der einzelnen Fahrzeuge wurden so getestet und angepasst, dass sie nicht ineinander fahren. Die initialen Daten des Szenarios werden über einen Button geladen. Wenn sich bestimmte Daten im Laufe des Szenarios verändern, werden sie auch im Hintergrund über einen Button nachgeladen.

8 Schlussbetrachtung

Ziel dieser Arbeit war es, die eingangs formulierte Forschungsfrage zu beantworten: Wie kann ein mobiles Dashboard gestaltet werden, um die Überwachung, Steuerung und Analyse von AGVs in intralogistischen Umgebungen effizient zu unterstützen? Dazu wurde im Rahmen dieses Projekts ein Prototyp für ein mobiles Dashboard zur Überwachung, Analyse und Steuerung autonomer Transportfahrzeuge in intralogistischen Umgebungen entwickelt.

Zentrales Element des Dashboards ist die interaktive Lagerkarte, auf welcher Transportfahrzeuge anhand passender Icons in Echtzeit visualisiert werden. Über filterbare Listen können Nutzer, die in zwei verschiedene Nutzergruppen unterteilt werden, gezielt Informationen zu Fahrzeugen und Transportaufträgen einsehen. Störungen werden mit Hilfe von Push-Benachrichtigungen gemeldet und betroffene Fahrzeuge visuell hervorgehoben, sodass eine schnelle Reaktion möglich ist. Kritische Situationen, wie beispielsweise Deadlock-Situationen, lassen sich durch eine manuelle Routenanpassung direkt über das System auflösen und Transportaufträge können vom Nutzer priorisiert werden. Weiterhin lässt sich die Anwendung dynamisch an verschiedene Bildschirmgrößen anpassen, um auch die mobile Anwendung im Lageralltag zu gewährleisten. Ergänzend bietet eine Statistikseite der Lagerleitung die Möglichkeit, Störungen systematisch zu analysieren und daraus präventive Maßnahmen abzuleiten.

Die bisher implementierten Funktionen zeigen das Potenzial des mobilen Dashboards, den Einsatz von AGVs zu unterstützen. Er kann Transparenz über Abläufe von AGVs schaffen, die Reaktionszeiten durch schnelle Benachrichtigungen verkürzen, Möglichkeiten zur Steuerung der AGVs bieten sowie die Analyse von AGVs durch geeignete Übersichten unterstützen. Gleichzeitig zeigen sich verschiedene Möglichkeiten, den bestehenden Prototypen zu erweitern. Denkbar sind die Erweiterung um weitere Nutzerrollen mit nutzerspezifischen Ansichten und Funktionen, umfassendere Analysefunktionen oder ein Ausbau des Bereichs der vorausschauenden Wartung.

Aufgrund zeitlicher und technischer Limitationen konnten nicht alle Anforderungen umgesetzt werden. So wurde keine Anbindung an externe Systeme (Anforderung 4.3), wie eine AGV-Navigationssoftware oder ein Warehouse Manage-

ment System, realisiert. Der Fokus lag daher auf der Visualisierung und Gestaltung eines funktionalen Prototyps. Die fehlende reelle Datenumgebung ergibt eine weitere Limitation. Die Konzeption basiert auf einem stark vereinfachten Lagerlayout und einer simulierten Datenumgebung, weshalb weder die tatsächliche Komplexität intralogistischer Umgebungen noch die tatsächlichen Störungen der AGVs abgebildet werden konnten und auf Testdaten zurückgegriffen werden musste. Eine Erprobung des Prototyps in einer realen Umgebung war im Rahmen dieses Projekts nicht möglich, sodass Aussagen über die Praxistauglichkeit offenbleiben.

Trotz der genannten Limitationen bietet der entwickelte Prototyp einen praxisnahen Ansatz zur Verbesserung intralogistischer Prozesse im Umgang mit AGVs. So kann der Prototyp wertvolle Impulse für weiterführende Forschung und Entwicklung liefern. Es ist denkbar, dass auf Grundlage des Prototyps weitere Forschungen zum Nutzen mobiler Dashboards zur Überwachung, Steuerung und Analyse von AGVs sowie den getroffenen Funktions- und Designentscheidungen betrieben werden. Auch für die Praxis bietet der entwickelte Prototyp einen Ansatzzpunkt für die Entwicklung komplexer AGV-Systeme und bildet damit einen Baustein für eine effiziente und transparente Steuerung intralogistischer Prozesse.

Anhang**Anhangsverzeichnis**

A Ereignisgesteuerte Prozessketten.....	VIII
A.1 EPK Wareneingang.....	VIII
A.2 EPK Warenausgang.....	IX
B Anforderungstabelle	X
C Aktivitätsdiagramme im Rahmen der Prozessmodellierung.....	XII
C.1 Aktivitätsdiagramm: Priorisierung von Aufträgen	XII
C.2 Aktivitätsdiagramm: Route anzeigen.....	XIII
C.3 Aktivitätsdiagramm: Anpassen einer Fahrtroute	XIV
D Entity-Relationship-Modell im Rahmen der Datenmodellierung.....	XV

A Ereignisgesteuerte Prozessketten

A.1 EPK Wareneingang

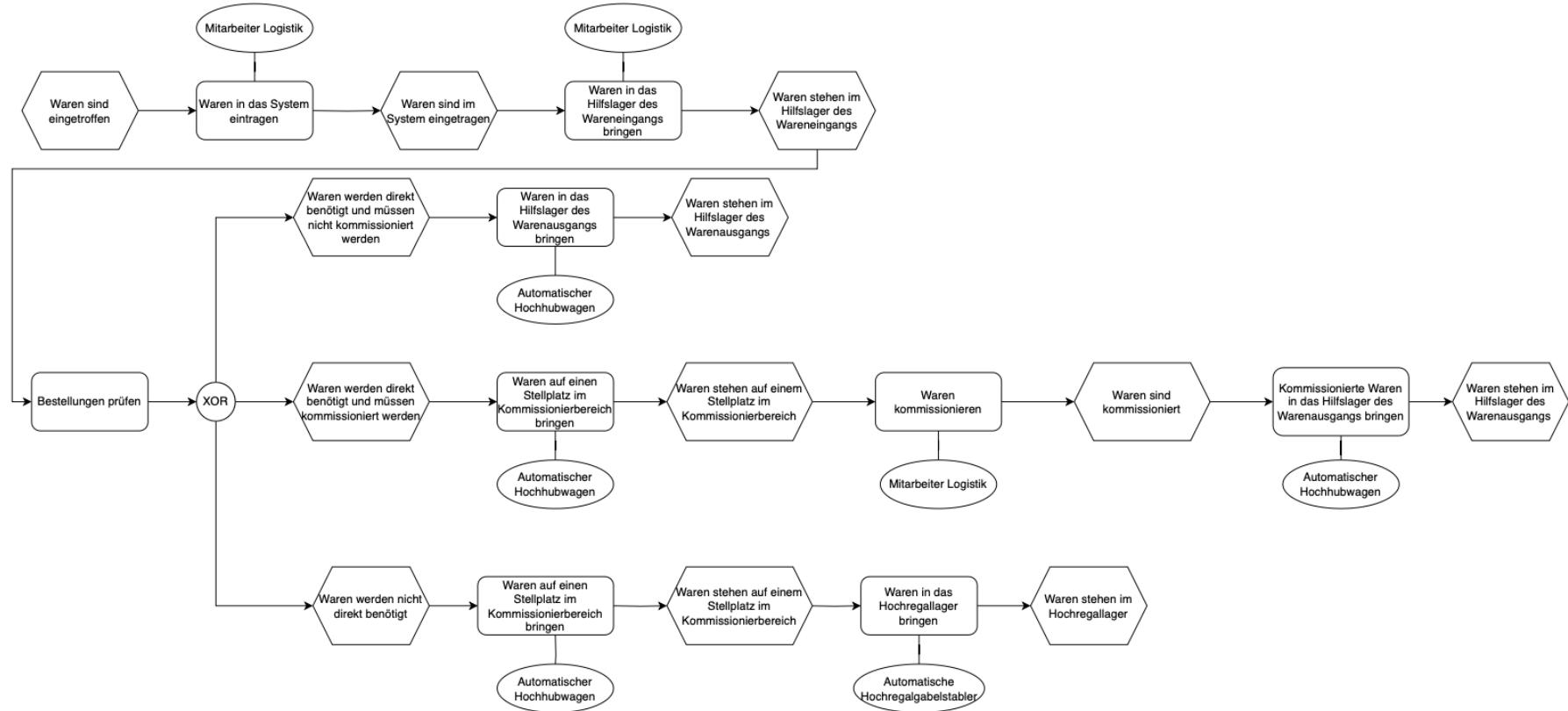


Abbildung 15: EPK zur Abbildung des Wareneingangsprozesses

A.2 EPK Warenausgang

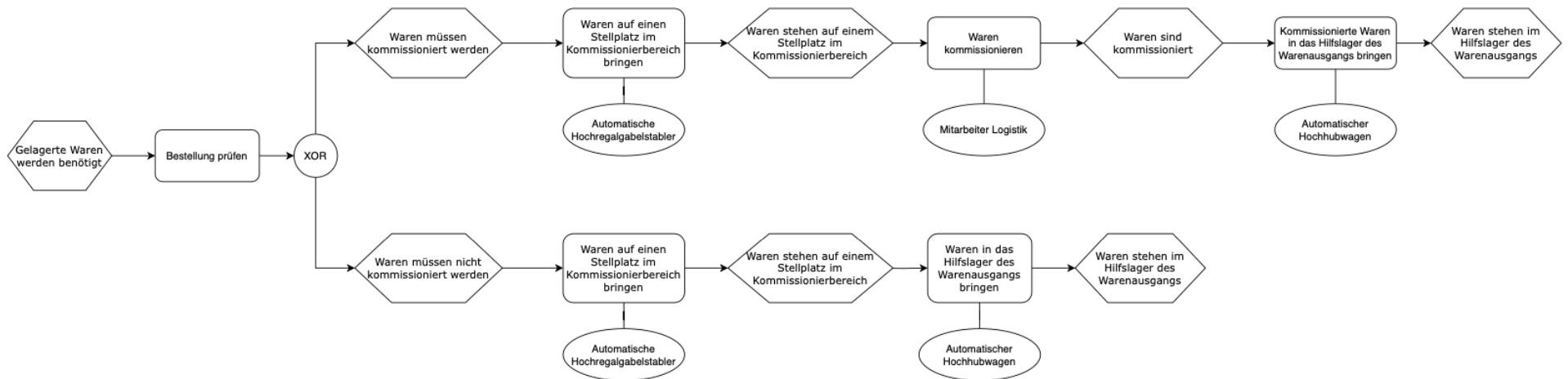


Abbildung 16: EPK zur Abbildung des Warenausgangsprozesses

B Anforderungstabelle

Nr.	Bezeichnung	Beschreibung	Status
1.	Visualisierung		
1.1.	Lagerkarten-Visualisierung	Das Dashboard stellt eine interaktive 2D-Karte des Lagerlayouts dar, auf der alle Zonen klar markiert sind (z. B. Lagerbereiche, Wege).	Umgesetzt
1.2.	Live-Positionsanzeige	Die Position aller Fahrzeuge wird in Echtzeit auf der Karte angezeigt.	Umgesetzt
1.2.1.	Fahrzeugsymbole	Jedes Fahrzeug wird durch ein symbolisches Icon dargestellt und ist mit einer eindeutigen ID beschriftet.	Umgesetzt
1.2.2.	Unterschiedliche Fahrzeuge	Unterschiedliche Fahrzeugtypen werden durch verschiedene Symbole voneinander abgegrenzt.	Umgesetzt
1.2.3.	Störungssymbol	Bei einer Störung wird ein rotes Warnsymbol am betroffenen Fahrzeug eingeblendet.	Umgesetzt
1.2.4.	Routendarstellung	Bei Klick auf ein Fahrzeug wird die aktuelle Route auf der Karte angezeigt.	Umgesetzt
1.3.	Kontextuelle Fahrzeuginformationen	Durch Klick auf ein Fahrzeug öffnet sich eine reduzierte Detailansicht der Fahrzeuge, sowie deren aktuelle Route.	Umgesetzt
1.4.	Anpassung der Lagerkarte	Die Lagerkarte ist anpassbar, um den Einsatz auf unterschiedliche Lagervarianten zu ermöglichen.	Umgesetzt
1.5.	Fahrzeugliste	Eine Liste aller Fahrzeuge mit Filterfunktion steht zur Verfügung, mit Fahrzeug-ID, Status, Ladestand, Akkuzustand, aktueller Transportauftrag und Störungsmeldungen	Umgesetzt
1.6.	Transportauftragsliste	Eine Liste aller Transportaufträge mit Filterfunktion steht zur Verfügung, mit Status, Fahrzeug, Start- und Zielplatz.	Umgesetzt

1.7.	Statistiken- & Auswertungsübersicht	Das Dashboard bietet eine Übersichtsseite für Statistiken und Auswertungen an (z. B. durchschnittliche Transportzeiten).	Umgesetzt
1.8.	Benutzerrollen	Das System unterscheidet zwischen Lagerleitung und Lagermitarbeitenden mit jeweils unterschiedlichen Zugriffsrechten (z. B. Statistikzugriff nur für Lagerleitung).	Umgesetzt
2.	Ereignisbehandlung		
2.1.	Automatische Störungserkennung	Das System erkennt automatisch kritische Zustände wie Störungen, Blockaden oder Akuprobleme.	Umgesetzt
2.2.	Alarmierung des Nutzers	Bei kritischen Zuständen wie Akkuschwäche, Blockade oder Kommunikationsausfall werden automatische Push-Nachrichten an Nutzer gesendet.	Umgesetzt
3.	Interaktion & Steuerung		
3.1.	Transportauftragsteuerung & Priorisierung	Transportaufträge priorisieren und Fahrzeuge manuell zuweisen.	Umgesetzt
3.2.	Fahrzeugaktivierung	Fahrzeuge können aktiv und inaktiv geschaltet werden.	Umgesetzt
3.3.	Manuelle Routenanpassung	Geplante Fahrtrouten können manuell angepasst werden (z.B. zur Umgehung von Blockaden).	Umgesetzt
4.	Systemanforderungen (nicht-funktional)		
4.1.	Mobilgerätekompatibilität	Das System ist optimiert für Smartphones und Tablets.	Umgesetzt
4.2.	Echtzeitaktualisierung	Alle Systeminformationen (Positionen, Aufträge, Akkustände) werden verzögerungsfrei aktualisiert.	Umgesetzt
4.3.	Systemanbindung	Das System enthält Schnittstellen für eine Anbindung an operative Steuersysteme.	Nicht umgesetzt

Tabelle 1: Anforderungstabelle

C Aktivitätsdiagramme im Rahmen der Prozessmodellierung

C.1 Aktivitätsdiagramm: Priorisierung von Aufträgen

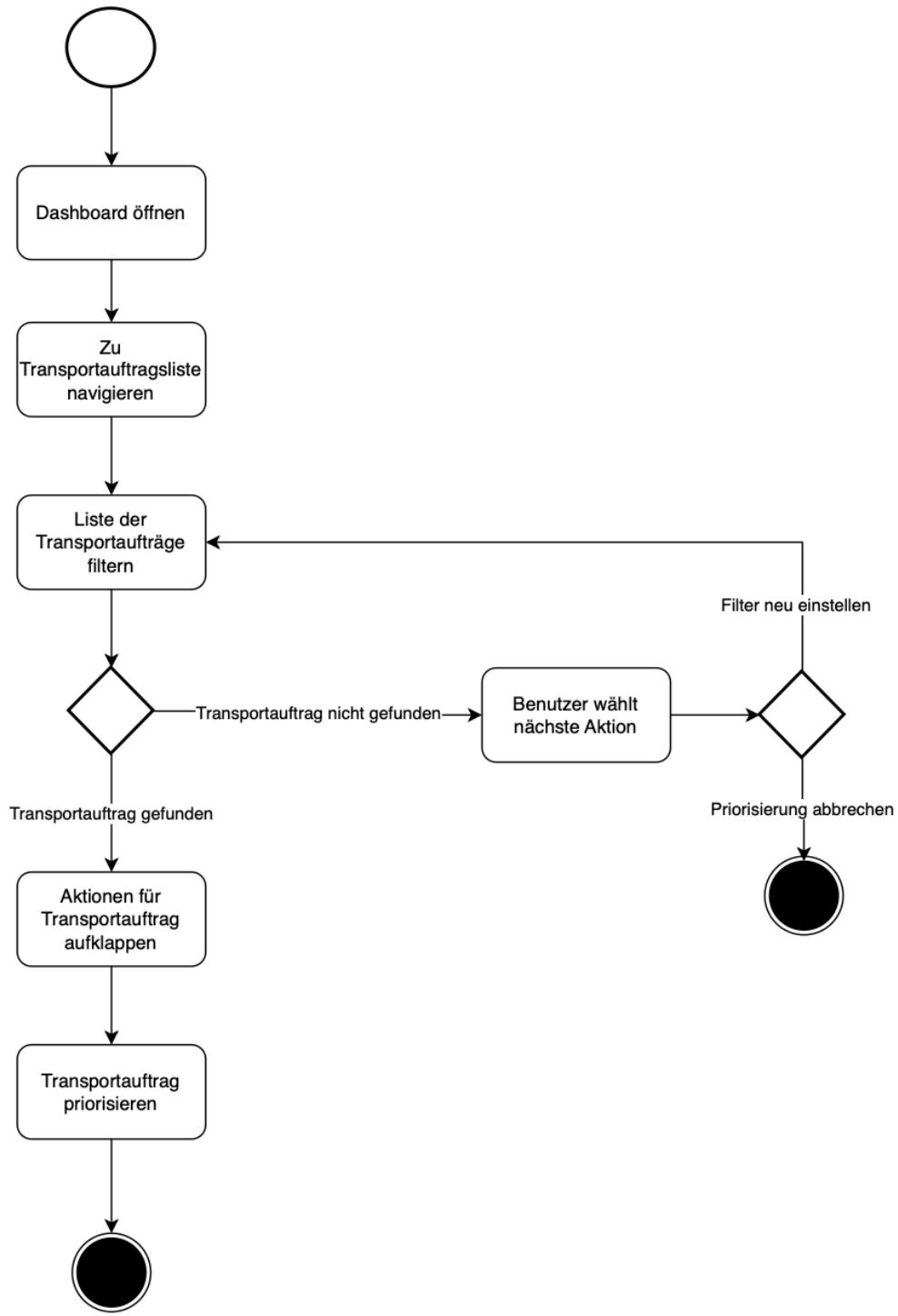


Abbildung 17: Aktivitätsdiagramm zur Priorisierung von Aufträgen

C.2 Aktivitätsdiagramm: Route anzeigen

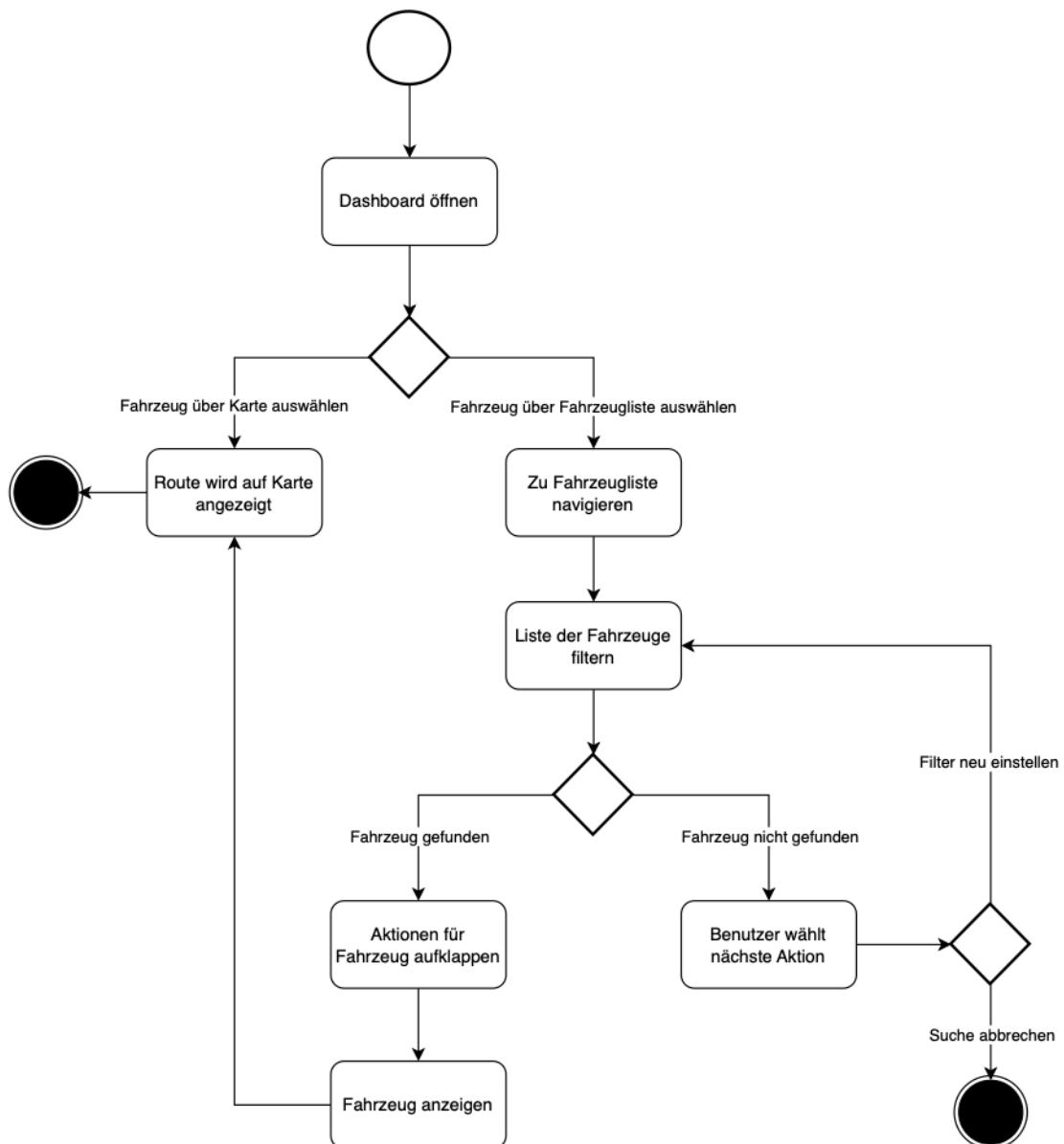


Abbildung 18: Aktivitätsdiagramm Route anzeigen

C.3 Aktivitätsdiagramm: Anpassen einer Fahrtroute

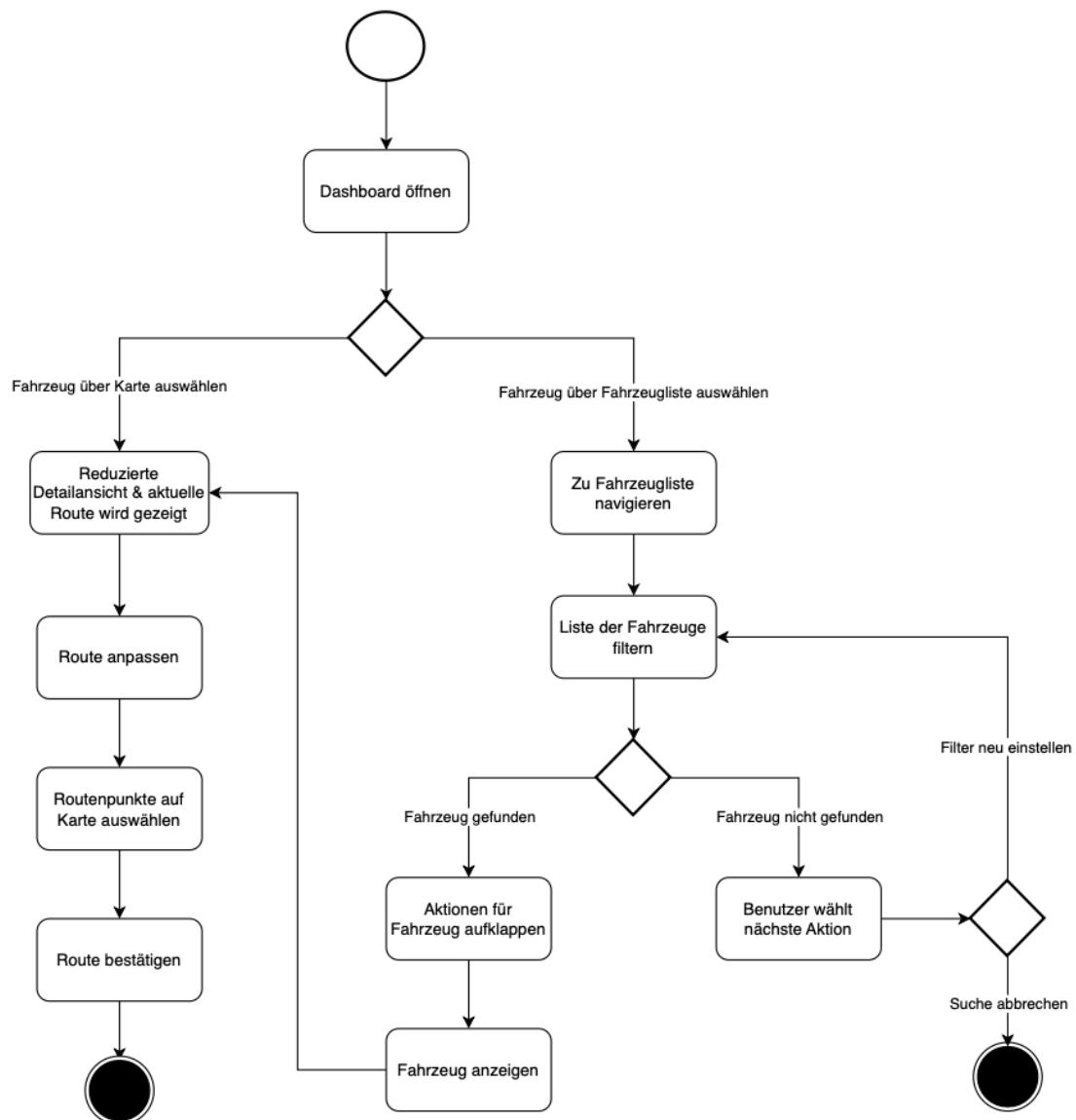


Abbildung 19: Aktivitätsdiagramm Route anpassen

D Entity-Relationship-Modell im Rahmen der Datenmodellierung

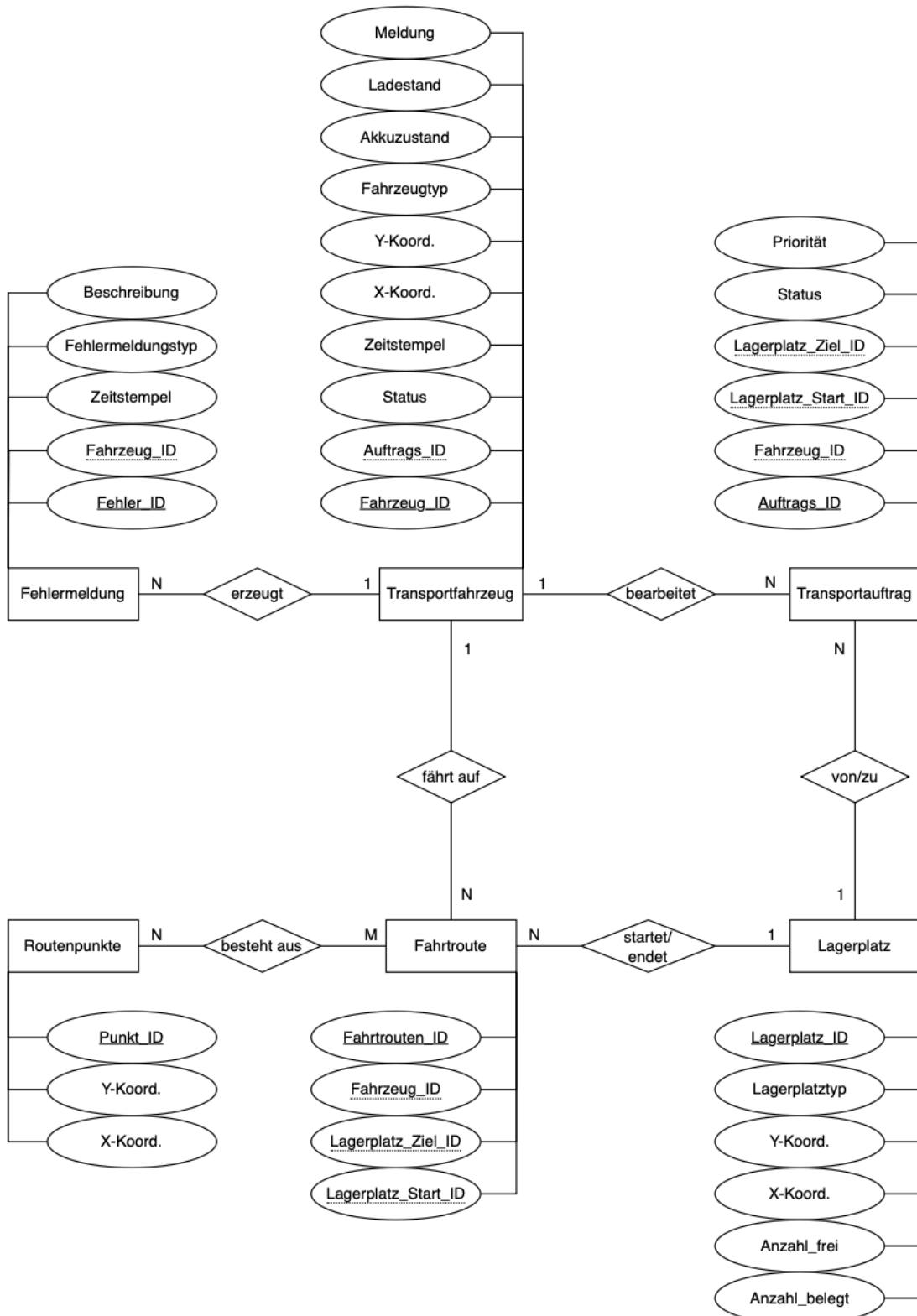


Abbildung 20: ERM zur Datenmodellierung

Literaturverzeichnis

- (Adenowo/Adenowo 2013): Adenowo, A. A. A.; Adenowo, B. A.: Software Engineering Methodologies: A Review of the Waterfall Model and Object- Oriented Approach. In: International Journal of Scientific & Engineering Research, 4 (2013) 7, S. 427 - 434.
- (Agafonkin 2025): Agafonkin, V.: Leaflet - an open-source JavaScript library for interactive maps, 2025. <https://leafletjs.com/>, Abruf am: 29.07.2025.
- (Arnold 2006): Arnold, D: Einleitung des Autors. In: Arnold, D (Hrsg.): Intralogistik: Potentiale, Perspektiven, Prognosen. Berlin 2006, S. 1 - 4.
- (Baginski 2006): Baginski, R.: Dezentrale Informationstechnologien für Flurförderzeuge und Ladungsträger zur Optimierung der Intralogistik. In: Arnold, D (Hrsg.), Intralogistik: Potentiale, Perspektiven, Prognosen, Berlin, Heidelberg 2006, S. 224 - 237.
- (Boehning 2014): Boehning, M.: Improving safety and efficiency of AGVs at warehouse black spots. In: 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca 2014, S. 245 - 249.
- (Bootstrap Team 2025): Bootstrap Team: Bootstrap, 2025. <https://getbootstrap.com/>, Abruf am: 29.07.2025.
- (Carbon 2025): Carbon: Carbon - A simple PHP API extension for DateTime., 2025. <https://carbon.nesbot.com/docs/>, Abruf am: 29.07.2025.
- (Correia et al. 2020): Correia, N.; Teixeira, L.; Ramos, A. L.: Implementing an AGV System to Transport Finished Goods to the Warehouse. In: Advances in Science, Technology and Engineering Systems Journal 5 (2) (2020), S. 241 - 247.
- (Ellithy et al. 2024): Ellithy, K.; Salah, M.; Fahim, I. S.; Shalaby, R.: AGV and Industry 4.0 in warehouses: a comprehensive analysis of existing literature and an innovative framework for flexible automation. In: The International Journal of Advanced Manufacturing Technology 134 (2024), S. 15 - 38.
- (Few 2006): Few, S.: Information dashboard design: the effective visual communication of data, 1. Auflage, 2006.

- (Flores-García et al. 2022): Flores-García, E.; Jeong, Y.; Wiktorsson, M.; Woo, J.; Schmitt, T.; Hanson, L.: Characterizing Digital Dashboards for Smart Production Logistics. In: IFIP Advances in Information and Communication Technology, AMPS 2022, S. 521 - 528.
- (Fottner et al. 2021): Fottner, J.; Clauer, D.; Hormes, F.; Freitag, M.; Beinke, T.; Overmeyer, L.; Gottwald, S. N.; Elbert, R.; Sarnow, T.; Schmidt, T.; Reith, K. B.; Zedek, H.; Thomas, F.: Autonomous Systems in Intralogistics – State of the Art and Future Research Challenges. In: Logistics Research 14 (2021) 1, S. 1 - 41.
- (Fritzsche/Keil 2007): Fritzsche, M.; Keil, P.: Kategorisierung etablierter Vorgehensmodelle und ihre Verbreitung in der deutschen Software-Industrie.
- (Gao et al. 2024): Gao, Y.; Chang, D.; Chen, C.-H.; Sha, M.: A digital twin-based decision support approach for AGV scheduling. In: Engineering Applications of Artificial Intelligence 130 (2024) 107687.
- (Gröger et al. 2016): Gröger, C.; Stach, C.; Mitschang, B.; Westkämper, E.: A mobile dashboard for analytics-based information provisioning on the shop floor. In: International Journal of Computer Integrated Manufacturing 29 (2016) 12, S. 1335 - 1354.
- (Gudehus 2007): Gudehus, T.: Logistik. 2: Netzwerke, Systeme und Lieferketten, 3., aktualis. und erw. Auflage, Berlin [u. a.] 2007.
- (Günther 2006): Günther, P.: Eine Branche entdeckt ihre Potenziale. In: Arnold, D. (Hrsg.), Intralogistik: Potentiale, Perspektiven, Prognosen. Berlin 2006, S. 5 - 16.
- (Gupta 2023): Gupta, L.: JSON with Ajax, 2023. <https://restfulapi.net/json-with-ajax/>, Abruf am: 29.07.2025.
- (Hompel/Schmidt 2010): Hompel, M.; Schmidt, T.: Warehouse Management: Organisation und Steuerung von Lager- und Kommissioniersystemen. 4. Auflage, Berlin 2010.
- (Klaus/Krieger 2009): P. Klaus; W. Krieger (Hrsg.): Gabler-Lexikon Logistik: Management logistischer Netzwerke und Flüsse. 4., komplett durchgesehene und aktualisierte Auflage, Wiesbaden 2009.

- (Krasner et al. 1988): Krasner, G. E.; Pope, S. T.; Systems, P.: A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. In: *Journal of object oriented programming*, 1 (1988) 3, S. 26 - 49.
- (Kubasakova et al. 2024): Kubasakova, I.; Kubanova, J.; Benco, D.; Kadlecová, D.: Implementation of Automated Guided Vehicles for the Automation of Selected Processes and Elimination of Collisions between Handling Equipment and Humans in the Warehouse. In: *Sensors* 24 (2024) 3, S. 1029.
- (Laravel 2025a): Laravel: Blade Templates - Laravel 12.x - The PHP Framework For Web Artisans, 2025a. <https://laravel.com/>, Abruf am: 29.07.2025.
- (Laravel 2025b): Laravel: Eloquent: Getting Started - Laravel 12.x - The PHP Framework For Web Artisans, 2025b. <https://laravel.com/>, Abruf am: 29.07.2025.
- (Laravel 2025c): Laravel: Starter Kits - Laravel 10.x - The PHP Framework For Web Artisans, 2025c. <https://laravel.com/>, Abruf am: 31.07.2025.
- (Le-Anh/De Koster 2006): Le-Anh, T.; De Koster, M. B. M.: A review of design and control of automated guided vehicle systems. In: *European Journal of Operational Research* 171 (2006) 1, S. 1 - 23.
- (Li/Schulze 2024): Li, L.; Schulze, L.: Failure Prediction of Automated Guided Vehicle Systems in Production Environments through Artificial Intelligence. In: *Tehnicki Glasnik* 18 (2024) 2, S. 268–272.
- (MDN 2025a): MDN: Using the Fetch API - Web APIs, 2025a. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch, Abruf am: 29.07.2025.
- (MDN 2025b): MDN: CSS: Cascading Style Sheets, 2025b. <https://developer.mozilla.org/en-US/docs/Web/CSS>, Abruf am: 29.07.2025.
- (Mills 2025): Mills, O.: SortableJS, 2025. <https://sortablejs.github.io/Sortable/>, Abruf am: 29.07.2025.
- (OpenJS Foundation 2025): OpenJS Foundation: jQuery, 2025. <https://jquery.com/>, Abruf am: 29.07.2025.

- (Oracle 2025): Oracle: MySQL Explained: Your Guide to Mastering This Powerful Database, 2025. <https://www.oracle.com/mysql/what-is-mysql/>, Abruf am: 29.07.2025.
- (Pappas/Whitman 2011): Pappas, L.; Whitman, L.: Riding the Technology Wave: Effective Dashboard Data Visualization. In: Human Interface and the Management of Information. Interacting with Information, Berlin [u. a.], 2011, S. 249 - 258.
- (Saravanos/Curinga 2023): Saravanos, A.; Curinga, M. X.: Simulating the Software Development Lifecycle: The Waterfall Model. In: Applied System Innovation 6 (2023) 6, S. 108.
- (Schmidt et al. 2020): Schmidt, T.; Reith, K.-B.; Klein, N.; Däumler, M.: Research on Decentralized Control Strategies for Automated Vehicle-based In-house Transport Systems – a Survey. In: Logistics Research (2020).
- (Shneier/Bostelman 2015): Shneier, M.; Bostelman, R.: Literature Review of Mobile Robots for Manufacturing.
- (Thamrongaphichartkul et al. 2020): Thamrongaphichartkul, K.; Worrasittichai, N.; Prayongrak, T.; Vongbunyong, S.: A Framework of IoT Platform for Autonomous Mobile Robot in Hospital Logistics Applications. In: 2020 15th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), 2020, S. 1 - 6.
- (The PHP Group 2025): The PHP Group: PHP: Hypertext Preprocessor, 2025. <https://www.php.net/index.php>, Abruf am: 29.07.2025.
- (VDMA 2024): VDMA: Absatz von mobilen Robotern in der Intralogistik steigt um 24 Prozent, 2024. <https://www.vdma.eu/viewer/-/v2article/render/130122118>, Abruf am: 25.07.2025.
- (VDMA 2025): VDMA: Deutsche Intralogistikbranche in Zahlen, 2025. <https://www.vdma.eu/viewer/-/v2article/render/141422542>, Abruf am: 20.07.2025.
- (Yuan et al. 2019): Yuan, R.; Graves, S. C.; Cezik, T.: Velocity-Based Storage Assignment in Semi-Automated Storage Systems. Production and Operations Management 28 (2019) 2, S. 354 - 373.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die Arbeit (den gekennzeichneten Anteil der Arbeit) selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen oder anderen Quellen entnommen sind, sind als solche kenntlich gemacht. Die Richtlinien zur Sicherung der guten wissenschaftlichen Praxis an der Universität Göttingen wurden von mir beachtet. Zudem versichere ich, dass die Arbeit in gleicher oder ähnlicher Form noch nicht Bestandteil einer Studien- oder Prüfungsleistung war. Die schriftliche und die elektronische Form der Arbeit stimmen überein.

In der hier vorliegenden Arbeit habe ich ChatGPT oder eine andere KI wie folgt genutzt:

[] gar nicht

[x] bei der Ideenfindung

[] bei der Erstellung der Gliederung

[] zum Erstellen einzelner Passagen, insgesamt im Umfang von ...% am gesamten Text

[x] zur Entwicklung von Software-Quelltexten

[x] zur Optimierung oder Umstrukturierung von Software-Quelltexten

[x] zum Korrekturlesen oder Optimieren

[] Weiteres, nämlich:

Ich versichere, alle Nutzungen vollständig angegeben zu haben. Fehlende oder fehlerhafte Angaben werden als Täuschungsversuch gewertet. Mir ist bewusst, dass bei Verstoß gegen diese Grundsätze die Prüfung mit nicht bestanden bewertet wird

16.07.2025 Göttingen, Akademie
Datum, Ort, Unterschrift

16.07.2025, Göttingen, Yanzhe Peng
Datum, Ort, Unterschrift

16.07.2025 Göttingen, Akademie
Datum, Ort, Unterschrift

16.07.2025, Göttingen, Yanzhe Peng
Datum, Ort, Unterschrift

16.07.2025 Göttingen, Akademie
Datum, Ort, Unterschrift

16.07.2025, Göttingen, Trifler
Datum, Ort, Unterschrift