Hochschule **RheinMain**
Fachbereich Design Informatik Medien
Studiengang Medieninformatik

Abschlussarbeit

zur Erlangung des akademischen Grades

Master of Science

---

# Data Mining komplexer Datenstrukturen aus PDF-Dokumenten

---

| | |
|---|---|
| Vorgelegt von | Deniz Aydar |
| am | 25. Juli 2022 |
| Referent | Prof. Dr. Dirk Krechel |
| Korreferent | Prof. Dr. Philipp Schaible |

## Erklärung gemäß ABPO

Ich erkläre hiermit, dass ich

- die vorliegende Abschlussarbeit selbstständig angefertigt,
- keine anderen als die angegebenen Quellen benutzt,
- die wörtlich oder dem Inhalt nach aus fremden Arbeiten entnommenen Stellen, bildlichen Darstellungen und dergleichen als solche genau kenntlich gemacht und
- keine unerlaubte fremde Hilfe in Anspruch genommen habe.

Wiesbaden, 25. Juli 2022 _____

Deniz Aydar

## Erklärung zur Verwendung der Masterthesis

Hiermit erkläre ich mein Einverständnis mit den im folgenden aufgeführten Verbreitungsformen dieser Abschlussarbeit:

| **Verbreitungsform** | **Ja** | **Nein** |
|---|---|---|
| Einstellung der Arbeit in die Hochschulbibliothek mit Datenträger | | × |
| Einstellung der Arbeit in die Hochschulbibliothek ohne Datenträger | | × |
| Veröffentlichung des Titels der Arbeit im Internet | × | |
| Veröffentlichung der Arbeit im Internet | | × |

Wiesbaden, 25. Juli 2022 _____

Deniz Aydar

**Zusammenfassung**

PDF-Dokumente besitzen viele Informationen aus denen sich neue Daten generieren lassen können. Doch das Extrahieren von solchen Daten ist heutzutage immer noch mit Hürden verbunden. Dies gilt auch für die Dokumente die in den Prozessen von Autohäusern und Kfz-Werkstätten verwendet und anschließend gelagert werden. Um die Frage zu beantworten inwiefern neue Daten aus diese Art von Dokumenten verarbeiten lassen, wird im Rahmen dieser Arbeit ein System konzipiert und entwickelt welches es ermöglichen soll weiteres Wissen zu gestalten.

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

„Digitalisierung im Alltag voranbringen“ – Das war einer der Wahlslogans während der Bundestagswahl 2021. Gemeint war damit eine zunehmende Digitalisierung im privaten Alltag vieler Bürger:innen, aber auch in der Wirtschaft machte sich wachsend der Wunsch nach mehr digitalen Alternativen breit (vgl. Bundesregierung 2021). Dieser Wunsch beinhaltete vor allem einen Wechsel von gängigen Papierformen verschiedenster Dokumente zu denselben in digitaler Ausprägung.

Genau jenes Bedürfnis nach Digitalisierung betrifft auch Arbeitnehmer:innen / Händler:innen in Autohäusern und Kfz-Werkstätten, die durch ihre berufliche Tätigkeiten mit einer Vielzahl an Unterlagen, Dokumenten oder Belegen arbeiten müssen. Unter dieser Vielzahl fallen Dokumente wie Werkstatt-, Kauf- und Mietverträge sowie Rechnungen oder Diagnoseberichte.

Durch eine Digitalisierung jener Dokumente eröffnen sich Vorteile wie bessere Zugänglichkeit, größere Langlebigkeit und vor allem eine angepasste und leichtere Nutzung, die zu einer höheren Effektivität innerhalb täglicher Arbeitsschritte führt. Durch bisherige erste Schritte der Digitalisierung sind diese benötigten Dokumente bereits elektronisch aufgearbeitet und den Arbeitnehmenden in Autohäusern und Kfz-Werkstätten zur Verfügung gestellt worden.

Die Folge dessen ist, dass alle Dokumente standardisiert sind und dadurch die in den Dokumenten beinhalteten Informationen neu verarbeitet werden können. Eine Extraktion der Daten der einzelnen Dokumente ist jedoch nur eingeschränkt möglich, da die Struktur im gängigen Gebrauch im Dateiformat PDF (Portable Document Format) festgelegt ist.

Jene Limitierung der Datenextraktion wirkt sich gleichermaßen auf die Dealer-Management-Systeme (DMS), mit solchen die Arbeitnehmenden ihr Prozesse

abwickeln, aus. Der daraus resultierende Arbeitsaufwand, welcher dabei entsteht, um die Daten wiederverwendbar zu konstruieren, ist derzeit enorm.

## 1.1 Zielsetzung

Aus diesem Grund möchte ich innerhalb dieser Master-Thesis ein System entwickeln, das genau jene Problematik erleichtert und löst. Innerhalb üblicher Methoden werden heutzutage die Daten zunächst über eine grafische Anzeige mithilfe einer bereits existierenden Benutzeranwendung via gängiges Kopieren und Einfügen entnommen. Dieser Schritt funktioniert zwar im Regelfall, ist jedoch oftmals stark fehleranfällig und kann lückenhaft sein.

In so einem Fall muss das Einfügen und Kopieren manuell durchgeführt werden, was bei einer riesigen Menge an Dokumente zu einer monotonen Arbeit führt. Andernfalls können die Daten eigenhändig abgeschrieben werden – jener Vorgang benötigt jedoch viele Ressourcen. Eher geeignet ist es, einen Datenkonverter für die PDF Dateien zu nutzen, um so die Dokumente in Bilder beispielsweise umzuwandeln, wodurch die Daten zugänglicher sind, aber immer noch verarbeitet werden müssen, was den gesamten Vorgang als eine Notlösung wirken lässt.

Aufgrund dieser bisherigen teils aufwendigen und mit Fehlern verbundenen Möglichkeiten möchte ich mich in meiner Thesis von diesen Optionen abwenden und das Data-Mining nutzen. Das Data-Mining soll eine Automatisierung unterstützen, mit der Daten aus dem PDF-Dokument extrahiert werden. Allein über das Data-Mining ist es möglich, Inhalte zu extrahieren und für andere Systeme bereit zu stellen.

Der Unterschied zum gängigen Data-Mining liegt dabei in den Einschränkungen des Dateiformats: der Quelltext einer solchen Datei stellt erstens keine klare Hierarchie der Daten dar und zweitens besitzt es durch das Fehlen von Markierungen keine Informationen darüber, was die Daten an sich darstellen sollen [6].

Mit Hilfe dieser Technik des Data-Minings soll es ermöglicht werden, aktuelle Prozesse detaillierter zu steuern und zu überwachen. Des Weiteren können unter anderem Abgleiche von Rechnungen mit dem System durchgeführt werden und weitere Datenanalyse und Reporting kreiert werden. Mit dem derzeitigen Stand der Datenextraktion können jedoch lediglich Agierende aus dem technischen Bereich arbeiten. Die Technologien hierfür benutzen unterschiedliche Ansätze und müssen daher zunächst für diesen Fall ausgewählt werden.

Damit jedoch auch für Agierende innerhalb des technischen Bereichs die Nutzung nicht zu abstrakt bleibt, soll durch die Entwicklung eines Systems der Zugriff greifbarer gestaltet werden, welches wiederum eine benutzerfreundliche Anwendung zur Verfügung stellen soll. Darüber hinaus muss eine Automatisierung vorhanden sein, um auch eine große Datenmenge verarbeiten zu können, und die Vorgänge sollen außerdem eine niedrigere Fehlerquote aufzeigen. Die Prozesse müssten außerdem das ganze über eine Steuerung und Regelung kontrollieren. Solche Prozesse können zum Beispiel über Regeln festgelegt werden, die bei Erfüllung weitere Aktionen auslösen können.

Eine solche Eigenschaft kann durch eine symbolische künstlichen Intelligenz (KI) abgedeckt werden, welche eine vorgegebene Verarbeitung möglich macht. Hiermit bezeichnet man einen altmodischen Ansatz der KI, bei der über das Festlegen von Symbolen menschliches Wissen in einer Logik gehalten wird und diese benutzt wird um weiteres Wissen zu generieren [5]. Durch eine Parametrisierung einer solchen KI kann dann die Unschärfe der ausgewählten Daten festgelegt werden, damit eine Feineinstellung stattfinden kann. Das heißt, der Benutzer eines solchen Systems soll sich beginnend von groben Definitionen für die Verarbeitung zu einer detaillierten und angepassteren Definition über Feedback des Systems hinarbeiten.

Mit Hilfe dieser Annäherung an Definitionen und Regeln kann so für eine bessere Erfolgsquote gesorgt werden. Zudem kann der Ansatz des Machine Learnings (ML), wobei erfolgreiche Verarbeitungen einem System antrainiert werden, weitere Dokumente aus Daten extrahieren. Hierbei ist noch offen, welcher Typ des Machine Learnings sich für diesen Fall eignet.

Beide Ansätze – der der symbolischen KI und der des Machine Learnings – bieten als Option die Entwicklung einer grafischen Entwicklungsumgebung (IDE) an. Jene soll das Festlegen von Definitionen und Regeln erlauben, mit welchen die Dokumente verarbeitet werden können. Außerdem soll diese Oberfläche verschiedene Funktionalitäten anbieten und auch Feedback sowohl bei einem problemlosen Lauf als auch bei einem fehlerhaften Lauf zurückgeben.

Des Weiteren soll das System es ermöglichen Änderungen der Definitionen anzumerken und Unterschiede zu erstellen, womit auch bisherige Ergebnisse angezeigt werden. Die Festlegung von Definitionen und Regeln soll außerdem durch eine Ansicht der Dokumente unterstützt werden. Insgesamt wird das System die zwei Ansätze als Subsysteme aufteilen um die Umsetzung zu modularisieren und einen Vergleich zu ermöglichen.

## 1.2 Ablauf

Um genau dieses optimale System für das beschriebene Szenario entwickeln zu können, werde ich in dieser Arbeit wie folgt vorgehen:

Für eine vollständige Auseinandersetzung soll eine weitere Analyse stattfinden, damit die vorhandenen Grundlagen für den Anwendungsfall evaluiert werden können. Das heißt, es werden mit Hilfe der Funktionalitäten der Bibliotheken Ergebnisse auf Basis der Datensätze erzeugt, um die Erfolgsquoten zu überprüfen. Diese werden dann verglichen und ausgewertet. Parallel dazu soll die Umgebung für die Entwicklung eingerichtet werden, mit der die Implementierung der Systeme stattfinden soll.

Danach widme ich mich der Konzeption und der Entwicklung des Systems. Dies beginnt mit der symbolischen KI als ein Subsystem des gesamtem Projekts, sodass ein direkterer Ansatz ermöglicht wird. Die Entwicklung hierbei wird im Wasserfall-Ansatz, einem Ansatz bei der phasenweise die Eigenschaften einer solchen KI umgesetzt werden, um eine Grundlage für die nächste Komponente zu bieten.

Bei dieser Komponente handelt es sich um die IDE, die den Zugang für den Benutzer zum System darstellt. Deswegen folgt im nächsten Schritt die Konzeptionierung und Entwicklung der IDE, welche sich während der Entwicklung der symbolischen KI parallelisieren lässt. Sobald das Gerüst der Benutzeroberfläche fertiggestellt ist, möchte ich dies mit der symbolischen KI verbinden.

Als Gegenstück wird sich dann mit dem Machine Learning Subsystem, bei der auch eine Konzeptionierung und Entwicklung, gewidmet. Hier wird das ganze in einem iterativen Ansatz umgesetzt um so den Hauptteil so früh wie möglich parat zu haben, damit das Training des ML-Systems durch die Dokumente auch zeitnah starten kann. Durch das Auswerten dieses Subsystems wird es daraufhin angepasst.

Danach wird das Testing für beide Subsysteme eingeführt um so das erfolgreiche Ausführen gewisser Eigenschaften abzudecken. Dies dient wiederum als Grundlage um die Continuous Integration (CI) für das System einzuführen, womit eine sichere und konsistente Entwicklung nach dem Prototyping angeboten wird.

Zum Schluss werden sich die Ergebnisse des gesamten Systems betrachtet und ein Ausblick zu der Thematik dargestellt.

# Kapitel 2

# Hintergrund

Um die Problematik detaillierter darzustellen und um eine Übersicht zu ermöglichen, werden in diesem Kapitel der Rahmen der Problematik aus technischer Sicht vertieft dargestellt. Hierbei sollen die Technologien abgewogen und evaluiert werden.

## 2.1 Analyse

Die

# Kapitel 3

# Konzeption und Umsetzung

Die Webanwendung soll das Anzeigen und Verwalten des Trinkwasserentnahme und die Darstellung der Anlegebuchungen als Funktionalität für die neue Zielgruppe anbieten um somit die zukünftigen Arbeitsprozesse zu unterstützen. Um dies zu ermöglichen, müssen die Interaktionen der Zielgruppe in Abhängigkeit der On-/Offline Situationen gestaltet werden. Dies findet im Rahmen der Feinkonzeption [4] statt. Dort werden Layout-Ansätze entwickelt und Wireframes erstellt, die die Interaktionen der Zielgruppe mit der Webanwendung abbilden sollen.

### 3.0.1 Ausführung

Das Einnehmen dieser Perspektive geschieht durch die Erstellung einer Persona [**?**], welche wiederum für die Webanwendung die Zielgruppe als Benutzer darstellt. Eine Persona beschreibt zudem sein Verhalten, seine Ziele, seine Motivationen und weitere Informationen in welcher der Benutzer tätig ist. Da die Zielgruppe sich auf ein Berufsfeld abbildet, ist es hierfür wichtig zu verstehen woraus die Tätigkeit besteht, in der die Benutzer die Anwendung verwenden.

Als Kommandant eines Passagierschiffes sind Kapitäne in der Personenschifffahrt bei Reedereien angestellt. Die Schiffsunternehmen übertragen die Verantwortung des Schiffes an den Kapitän, welcher die Durchführung aller zugewiesenen Aufgaben obliegt [**?**].

So kümmert er sich um das Wohlbefinden der Besatzung und die Vermeidung von Unfällen. Er achtet auf die Gewässerverschmutzung und auf weitere Beeinträchtigungen der Umwelt und inspiziert regulär den Zustand des Schiffes. Des Weiteren leitet er die Aufgaben der Besatzung und überwacht die Verrichtung der Besatzung.

Die Navigation des Schiffes wird vom Kapitän mithilfe von Seekarten, Radar und weiteren Geräten durchgeführt [**?**]. Dazu finden Signalisierungen mithilfe von Lichtern zwischen den Schiffen statt. Jeden Tag auf dem Schiff beobachtet er den Zustand des Schiffes, der Geschwindigkeit des Schiffes, den Druck des Schiffmotors und die Wassertiefe.

Das Budget, das Einkaufen und das Beladen von Treibstoff, die Vorräte und die Ausrüstungen sind weitere Eigenschaften des Schiffes, die der Kapitän überschauen muss. Dazu weiß er über die Fracht Bescheid, die auf dem Schiff transportiert wird und er kennt sich in den Softwares, die sich auf dem Schiff im Betrieb befinden, aus.

Durch die hohe Verantwortung und Aufgaben muss der Kapitän also eine disziplinierte Person sein, welcher geduldig arbeitet. Des Weiteren besitzt der Kapitän die Kenntnis zur Benutzung der Software auf dem Schiff, die im Betrieb genutzt wird. Daher ist ihm der Umgang mit der allgemeinen Bedienung einer Software nicht fremd.

Der Kapitän hat entweder eine Ausbildung zum Binnenschiffer [**?**] vollzogen und hat mit weiteren Berufsjahren sich den Einsatz als Schiffsführer erlangt oder er ist für seine Tätigkeiten mit einem Hochschulabschluss in Nautik oder in einem ähnlichen Studiengang ausgebildet und besitzt somit eine Qualifikation für die Schifffahrt. In Deutschland gibt es als weitere Option das Befähigungszeugnis „Nautischer Wachoffizier“ an einer Fachschule oder Fachhochschule durch erfolgreichen Abschluss zu erhalten [**?**].

Mit diesen gesammelten Informationen über den Kapitän lässt sich eine Persona folgendermaßen erstellen:

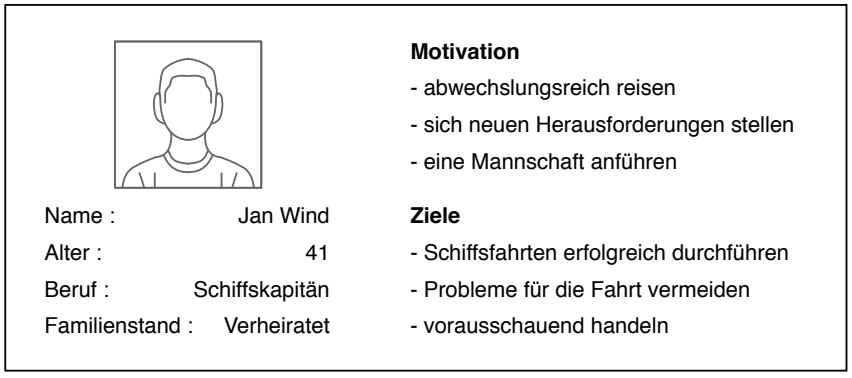



Abbildung 3.1: Steckbrief der Persona

In der Abbildung 3.1 ist die Persona des Kapitäns zu sehen. Diese heißt Jan Wind, ist 41 Jahre alt, Familienvater und leidenschaftlicher Reisender. Er studierte an einer Hochschule Nautik und fing nach dem Abschluss als Schiffsoffizier an

zu arbeiten. Nach Jahren der Berufserfahrungen hatte er dann einen weiteren Lehrgang abgeschlossen um das Binnenschifferpatent zu erlangen. Er lebt in Köln und ist gewohnt daran tage- oder auch wochenlang unterwegs zu sein und seine Familie nicht zu sehen.

Ihn motiviert es abwechslungsreiche und neue Reisen im Rahmen seiner Tätigkeit zu erleben und Herausforderungen, die sich auf einer Fahrt ergeben können, zu meistern. Jan ist hilfsbereit und verantwortungsbewusst, will aber auch im Gegensatz dazu eine Mannschaft an Bord haben, die seine Anweisungen genauestens befolgt. Sein Ziel ist es neben der Absolvierung einer Schiffsfahrt neue Orte und Menschen kennen zu lernen. Dafür nimmt er gerne die Unregelmäßigkeiten wie die Arbeitszeiten und die Wetterbedingungen, die in seinem Arbeitsalltag passieren, in Kauf.

### 3.0.2 Szenarien

Als Nächstes können mit dieser Persona Szenarien erstellen werden. Ein Szenario beinhaltet eine Situation, in welcher die Herangehensweise einer Persona mit der Webanwendung in einem festgelegten Kontext, beschrieben wird um sein jeweiliges Ziel zu erreichen. [**?**]. Es definiert somit wann, wo und wie die Geschichte der Persona stattfindet und wie sich die Persona verhält.

Die Szenarien, die einem Kapitän während seiner Arbeit auftreten können also nun mithilfe des Personas gestaltet werden. Aus den vielen Situationen, die sich aus den Tätigkeiten des Kapitäns im Bezug zur Webanwendung gestalten können, gehören die folgenden Szenarien zu der relevanten Menge:

Szenario 1:

Jan will auf der Hinfahrt die letzte Anlegung des Schiffes in Rüdesheim mit der aktuellen Buchung vergleichen. Dafür schaut er sich die vergangenen und die aktuellen Anlegungen an.

Szenario 2:

Jan legt das Schiff an, startet an der Anlegestelle die Trinkwasserentnahme und begibt sich während der Entnahme auf die Schiffsbrücke um andere Aufgaben durchzuführen. Kurz vor dem Ende der Entnahme kehrt er zum Wasserentnahmeschrank zurück um dies abzuschießen.

Szenario 3:

Jan legt das Schiff an, startet an der Anlegestelle die Trinkwasserentnahme und begibt sich während der Entnahme in ein Café gegenüber der Anlegestelle. Dort verliert er die Netzwerkverbindung mit der Anwendung. Kurz vor dem

Ende der Entnahme kehrt er zum Wasserentnahmeschrank zurück um dies abzuschließen.

Während die ersten beiden Szenarien, den erwartbaren Anwendungsfällen entsprechen, bezieht sich das dritte Szenario auf den möglichen Offline-Zustand, der in jeder Situation auftreten kann.

### 3.0.3 User Journeys

Im nächsten Schritt der User Experience verbinden wir die Persona mit jeweils einem Szenario und kreieren damit eine User Journey. Dies beschreibt dann die Schritte die ein Nutzer geht, um ein gewisses Ziel mit dem interaktiven System zu erreichen [**?**]. Dabei werden die Schritte als „Stages“ dargestellt, die mit der Aktivität, den Gedankten und den Gefühlen beschrieben werden. Zu diesem Ablauf kommt als weiterer Aspekt der Netzwerkzustand dazu, der für die Offline-Funktionalität als relevante Eigenschaft dient. Die User Journeys sind in den folgenden drei Abbildungen 3.2, 3.3, und 3.4 zu finden.

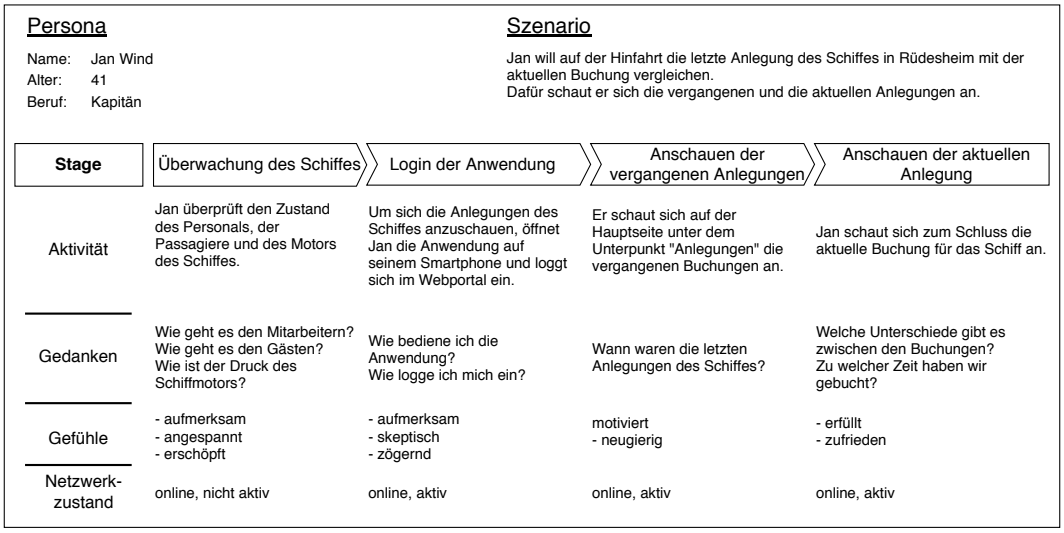



Abbildung 3.2: User Journey von einer Anlegung

In Abbildung 3.2 ist eine User Journey mit dem ersten Szenario zu sehen. Hier überwacht Jan das Schiff, führt ein Login der Anwendung durch und schaut sich die vergangenen und die aktuelle Anlegebuchungen an. Während er beim ersten Schritt noch leicht negative Gefühle hat, passen sich diese im Verlauf an bis er dann im letzten Schritt positive Gefühle zeigt. Das Gerät, welches für die Anwendung benutzt wird, ist bei allen Schritten mit dem Internet verbunden und wird erst ab dem zweiten Schritt benutzt.

Abbildung 3.3 zeigt eine weitere User Journey mit dem zweiten Szenario. Dort

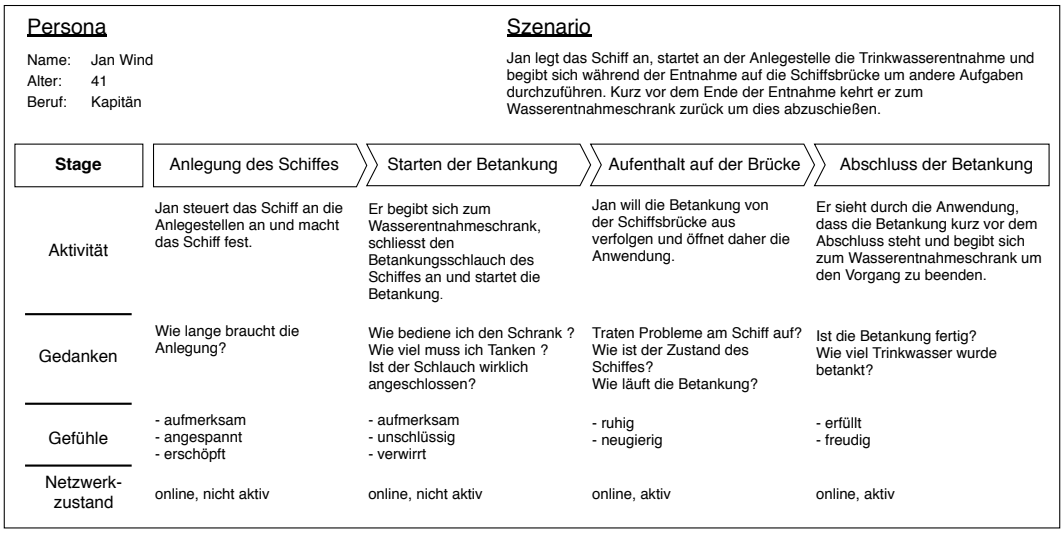



Abbildung 3.3: User Journey von einer Betankung

führt Jan die Anlegung durch, entriegelt den Wasserentnahmeschrank, stellt über einem Bildschirm am Schrank anhand der Daten wie Durchfluss und Preis die Entnahmen ein, startet die Betankung, begibt sich währenddessen auf die Brücke und schließt durch Zurückkehren zum Schrank die Betankung ab.

Während er noch beim zweiten Schritt, wo er sich unschlüssig fühlt, anwesend sein muss, erlaubt ihm im dritten Schritt die Nutzung der Anwendung die Betankung von der Ferne neugierig zu verfolgen. Dadurch ist es ihm möglich den vierten Schritt zeitnah auszuführen, was ihm die Arbeit erleichtert und freudig stimmt. In allen „Stages“ ist er mit dem Netzwerk verbunden, benutzt aber die Anwendung erst sobald er sich von dem Wasserentnahmeschrank entfernt beziehungsweise entfernen will.

Die letzte User Journey wird in Abbildung 3.4 angezeigt. Das Szenario ist hier ähnlich zum zweiten Szenario mit dem Unterschied, dass Jan sich während der Betankung in einer Gaststätte aufhält und die Verbindung zum Netzwerk dort verliert.

Durch den Ausfall der Verbindung wird er im dritten Schritt unruhig, doch kannmithilfe des letzten Stands der Betankung, welche von der Anwendung angezeigt wird, die Lage besser einschätzen und sich zum Wasserentnahmeschrank begeben. So kann er trotz dem Offline-Zustand der Anwendung die Betankung rechtzeitig abschließen.

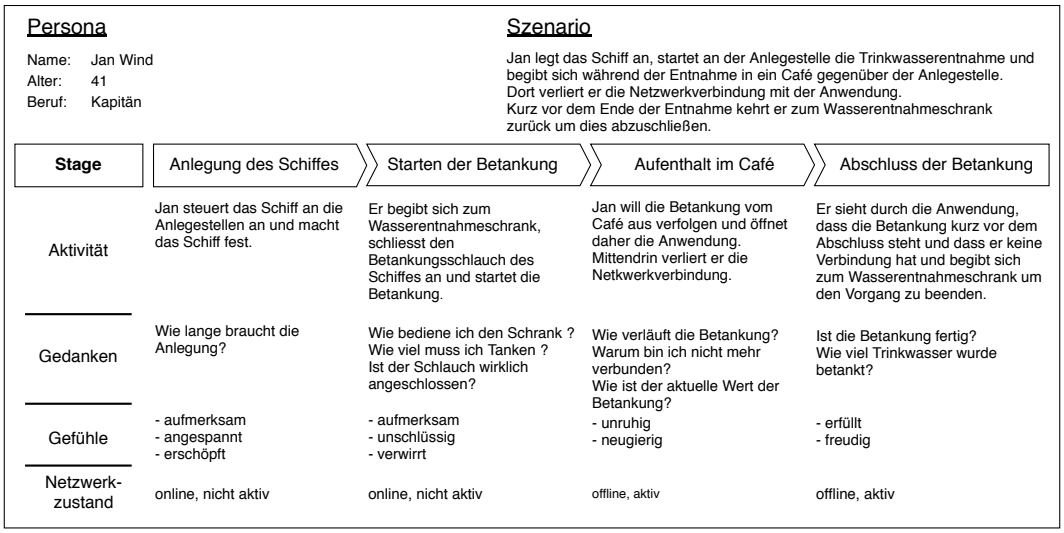



Abbildung 3.4: User Journey von einer Betankung mit einer Offline-Aktivität

### 3.0.4 Wireframes

Durch die Erstellung der User Journeys haben sich weitere Informationen für die Interaktion ergeben. Doch um diese Informationen für die Gestaltung der Wireframes zu verwenden, müssen diese zunächst durch die Nutzung einer Informationsarchitektur [4] verarbeitet werden. Es erlaubt die sinnvolle Platzierung der Informationen in der Webanwendung um den Benutzer eine Orientierung zu ermöglichen. Die Wireframes können basierend auf der Ordnung gestaltet werden und somit das initiale Design der Webseite präsentieren [**?**].

Die erste Aufbereitung der Informationen findet in den Rahmenbedingungen der Webanwendung statt. Das bedeutet, dass die Gestaltung abhängig von der Nutzung der Zielgruppe ist. Bedingt durch die Tätigkeit des Kapitäns und den Szenarien passt die Ausrichtung der Webanwendung auf mobilen Endgeräten. Doch mit diesem Medium kommen weitere Bedingungen.

So müssen die langsame, nicht ständig vorhandene Internetverbindung und die kleineren Bildschirme im Vergleich zu Desktop Computern beachtet werden. Doch bezüglich des Buchungssystems ist es für die Zielgruppe wichtig unterwegs durch die Anwendung die Zeiten der verschiedenen Anlegungen zu erhalten.

Des Weiteren will ein Kapitän während einer Betankung andere Aufgaben parallel ausführen und trotzdem noch den Zustand der Betankung, die er nur mit einem festen Aufenthalt am Wasserentnahmeschrank erfahren würde, verfolgen können. Für diese Anforderungen setzt es daher ein mobiles und internetfähiges Endgerät voraus. Auch passt hier ein responsives Design um eine Desktop Version zu unterstützen.

Bevor die Hauptinteraktion konzipiert wird, muss sich überlegt werden, wie die Authentifizierung gestaltet werden soll. Dies könnte über eine für die Webanwendung eigens erstellte Anmeldeseite oder über die bereits existierende Seite des Buchungssystem stattfinden.

Da es aus der technischen Sicht und den nötigen Informationen passender erscheint, wird die Login Seite des Buchungssystems benutzt. Um einen einfacheren Zugang für die Kapitäne zu ermöglichen, wird ein weiteres Login Formular mit einen Eingabefeld zur Startseite hinzugefügt. Statt einer E-Mail-Adresse und einem Passwort, wird für den Kapitän ein Zugangscode vom Buchungssystem generiert, welcher ihm vom Manager des Schiffes, welchem er zugeteilt ist, erhält. Dazu ist die Login Seite umrahmt von einem Header, der das Logo des Unternehmens und Menü für den Account beinhaltet und einem Footer, in dem das Impressum und weitere Daten zum System verlinkt werden.

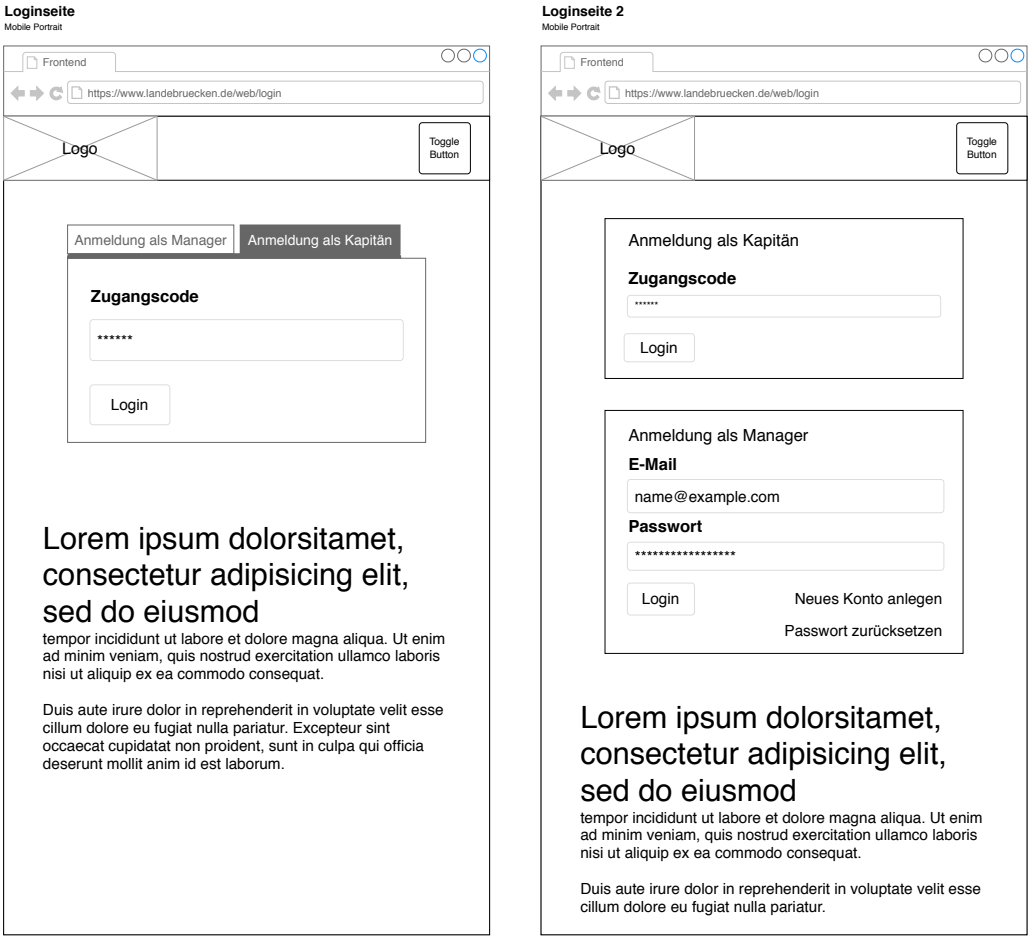
Es ergeben sich zwei mögliche Darstellungen für die Login Seite wie es in der Abbildung 3.5 zu sehen ist. Die Darstellung 3.5a zeigt eine Version bei der über Tabs zwischen den Formularen gewechselt kann. Bei 3.5b hingegen wird eine einfache Anordnung vertikal abgebildet.

Beide Darstellungen zeigen dazu eine Ansicht, die eine durch die Anordnung zu einer Gewichtung des Kapitäns führt. Hierbei ist die Idee, dass die Reihenfolge abhängig zum benutzten Endgerät seien soll, da die mobile Ansicht wahrscheinlicher von dem Kapitän genutzt werden. Die anderen Benutzer dagegen melden sich über einen Desktop Computer an.

Nachdem die Entwürfe für die Login Seite stehen, müssen die Informationen für die Übersichtsseite des Kapitäns gegliedert werden. Diese besitzt wie die Login Seite auch einen Header und Footer um das gleiche Erscheinungsbild wie vom Buchungssystem darzustellen. Die User Journeys zeigen, dass der Kapitän die Anwendung für folgende Daten benutzen will:

- Der Zustand der Trinkwasserentnahme
- Die letzten Trinkwasserentnahmen
- Die Anlegebuchungen des ihm zugeteilten Schiffes

Aus diesen Daten lassen sich zwei Themen erkennen, die man zunächst zwischen den zwei Gruppen Wasserentnahme und Anlegebuchungen aufteilen kann. Da der Vorgang der Betankung aber ein empfindlicher Prozess ist und übersichtlich bleiben soll, eignet es sich die letzten Trinkwasserentnahmen als eigene Gruppe zu betrachten.




(a) Mit Tabs (b) Ohne Tabs

Abbildung 3.5: Wireframes von der Loginansicht

Die Gruppe brauchen eine Priorisierung um die Anordnung und Position festzulegen. Diese Priorisierung findet durch eine Gewichtung der Funktionalitäten statt. So ist die Trinkwasserentnahme die wichtigste Funktionalität der Anwendung und gilt daher in der aktuellen Planung als Fokus der Webseite. Um dies auch zu gewichten muss daher der Zustand der Betankung im oberen Bereich der Webseite dargestellt werden, da dies das erste Element einer Webseite ist welches durch das Leseverhalten [**?**] eines Benutzers auftaucht. Das Leseverhalten bestimmt hierbei die Anordnung.

Der Zustand soll den Status der Entnahme darstellen. So kann die Menge der Zustände aus einer nicht stattfindenden Betankung, einer aktiven Betankung, einer aktiven Betankung mit Warnhinweisen (zum Beispiel auf Verbindungsstörungen oder auf die Türschließung durch den Alarmierungsablauf), einer

abgeschlossenen Betankung oder einer fehlerhaften Betankung bestehen. Zu diesem Zustand der Betankung gehören auch wichtige Informationen wie die aktuelle Betankungsmenge, sowie die gewünschte Menge hervorgehoben.

Als weitere Gewichtung können die Zustände mit der bekannten Assoziierung der Farben wie zum Beispiel Grün für Erfolg, Gelb für Warnung und Rot für auftretende Fehler markiert werden. Beim Zustand kann dann ein Untertitel für weitere Informationen dargestellt werden. Dies wäre bei Fällen in denen die Betankung in einem Notbetrieb läuft oder die speicherprogrammierbare Steuerung einen Verbindungsausfall hat, geeignet.

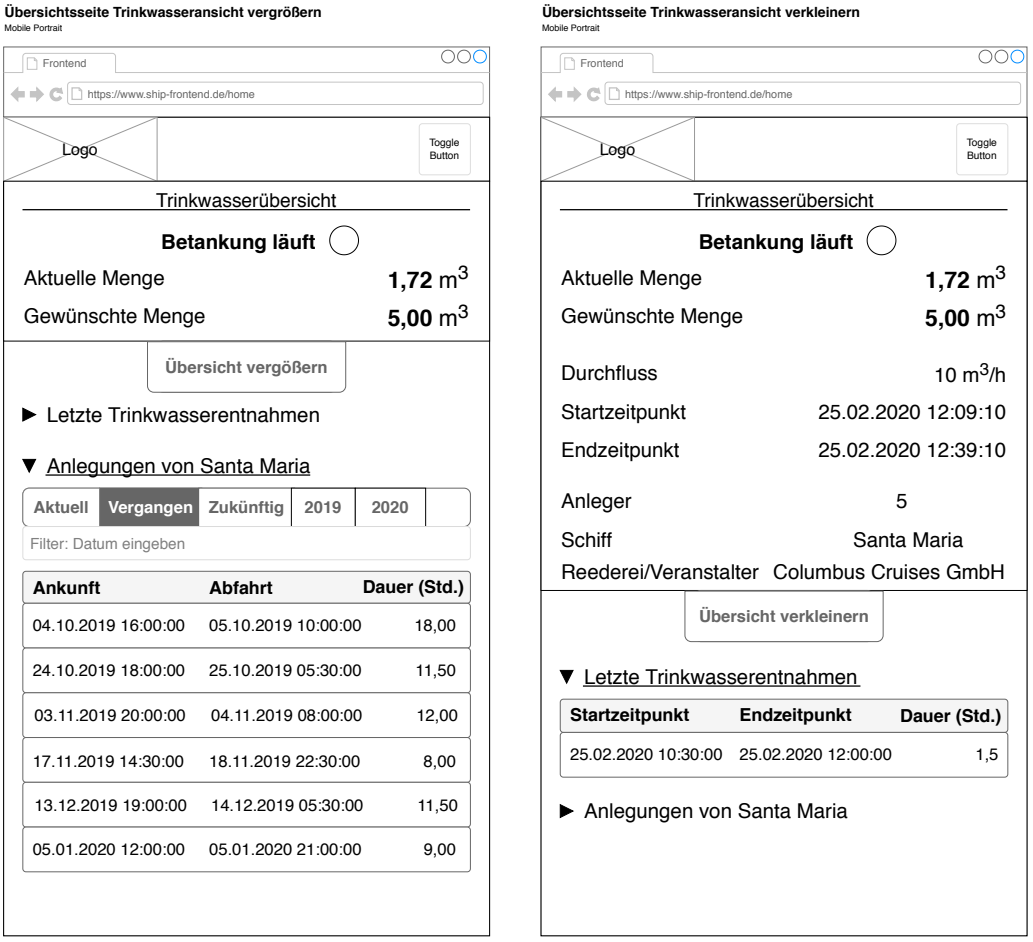
Danach können weiteren Daten wie der Durchfluss, der Start- und Endzeitpunkt und Informationen über das Schiff dargestellt werden. So werden für die jeweiligen Daten die Bezeichnungen linksbündig und die Werte rechtsbündig angezeigt um Kollisionen der Anzeige auf kleineren Bildschirmen zu vermeiden.

Der Wert der gewünschten Menge kann durch ein Doppel-Tippen mit einem Eingabefeld angepasst werden. Während sich der Startzeitpunkt bei einer Betankung nur einmal ändert, wird der Endzeitpunkt abhängig von dem Wert der gewünschten Menge und dem Wert des Durchflusses. Im unteren Bereich der Betankungsanzeige werden allgemeinere Informationen zur Anlegestelle, zum Schiff und zum Veranstalter des Schiffes abgebildet. Diese können als redundant aufgefasst werden, sollen aber den Kontext spezifizierter für den Kapitän darstellen, um mögliche Problemfälle zu vermeiden.

Als nächste Funktionalität werden die bestätigten Anlegebuchungen in einer Tabelle als Liste angezeigt. Von diesen Anlegungen braucht der Kapitän daher die relevanten Informationen wie den Startzeitpunkt, Endzeitpunkt und die Dauer. Um die Ansicht auf einem mobilen Endgerät übersichtlich zu halten, ist die Tabelle einklappbar. Außerdem besitzt die Tabelle weitere Filter, womit die Liste noch überschaubarer wird.

Die letzte Hauptfunktionalität betrifft die letzten Trinkwasserentnahmen. Hier wird wie bei den Anlegebuchungen eine einklappbare Tabelle angezeigt. Da die Länge der Tabelle in Relation kleiner ausfällt als die Liste der Anlegebuchungen und es thematisch zur Hauptfunktionalität passt, wird die Tabelle der letzte Trinkwasserentnahmen nach der Anzeige der Trinkwasserentnahme angeordnet.

In Abbildung 3.6 sind die beschriebenen Bereiche zu sehen. Hier wird außerdem gezeigt, dass die Anzeige der Trinkwasserentnahme vergrößert beziehungsweise verkleinert werden kann um den Fokus auf die wichtigsten Daten zu ermöglichen. Des Weiteren werden alle Inhalte kurz und präzise gehalten um nicht nur den Anzeigebereich nicht zu überfüllen sondern auch da längere Inhalte wie




(a) mit einer Kurzübersicht

(b) mit einer vergrößerten Übersicht

Abbildung 3.6: Wireframes von der Übersichtsseite

Texte wahrscheinlicher überflogen und Webseiten seltener gescrollt werden[2]. So werden auf ausführlichere Erklärungen in eine Anleitung ausgelagert womit die Seite simpel und übersichtlich bleibt.

## 3.1 Offline-Aktivitäten

In den Szenarien und User Journeys tritt eine weitere Problematik auf. Während die ersten zwei User Journeys erwartbare Abläufe beschreiben, ist dies bei der dritten User Journey nicht der Fall. Dort erscheinen weitere Probleme für den Benutzer die ihn verwirren können und im Unklaren lassen.

Der Verlust einer Netzwerkverbindung kann dem Benutzer jederzeit während

der Benutzung der Webanwendung passieren und möglicherweise auch während er einer Aufgabe nachgeht, bei der er sich auf die angezeigten Daten verlässt.

So ist die Betankung des Trinkwassers ein kritischer Prozess bei dem die Zeitnähe der Daten elementar ist um diesen Prozess erfolgreich abzuschließen. Die Anpassungen sollen daher Transparenz für das Zusammenspiel zwischen dem Netzwerkverlust und der Webanwendung schaffen. Dem Benutzer muss über die Interaktion klar werden, wie das Verhalten der Anwendung in solch einem Szenario aussieht.

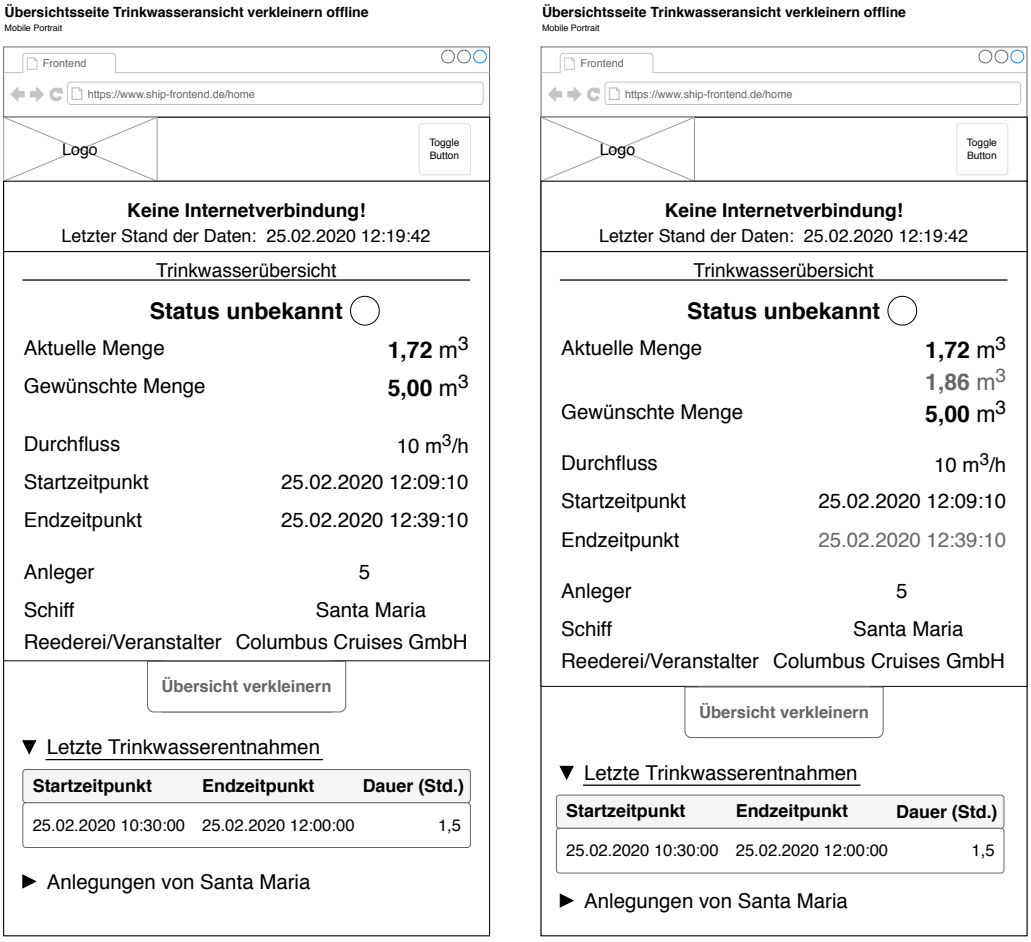
Eine fehlende Benachrichtigung beziehungsweise Signalisierung kann des Weiteren zu Frustrationen des Benutzers führen und Aspekte wie Nutzbarkeit, Benutzerfreundlichkeit und das positive Empfinden verschlechtern. Anzeichen dafür wären, dass er wahllos auf der Seite herum tippt, herum scrollt, zur Seite davor navigieren will, die Seite versucht neu zu laden oder den Browser zu schließen und zu öffnen [**?**].

Damit dies vermieden werden kann, müssen Änderungen wie der Netzwerkzustand angezeigt werden und die Daten bestimmt werden, die trotz dem Offline-Zustand bereitgestellt werden können [**?**]. Des Weiteren kann der Verlust der zuletzt angefragten Daten vermieden werden, indem die Seite bei einem Neuladen keine leere Ansicht darstellt, sondern die Anzeige wie zuvor wieder aufruft. Auch Farbveränderungen können dem Benutzer aufmerksam auf die entstandene Situation machen. So wird der Übergang zwischen On- und Offline für den Benutzer angenehmer gestaltet.

Um den Benutzer über diesen Zustand in Kenntnis zu setzen, erscheint eine Benachrichtigungsfeld über der Betankungsanzeige, welches in der Abbildung 3.7 zu sehen ist. Dies wird farblich markiert und beinhaltet die Angabe des letzten Zeitpunktes der Daten. Hierbei soll es eine Farbe benutzt werden welches eine Warnung an den Benutzer signalisiert.

Passende Farben für den Kapitän wären hierfür Gelb oder Orange, welche ihm aus dem Schiffsbetrieb bekannt seien [**?**] können und ihm einen Bezug zu einer Warnung vermitteln. Da die Farbe Gelb für den Zustand der Betankung genutzt wird und das Benachrichtigungsfeld aufgrund der Nutzung im Freien kontrastreich dargestellt werden soll, wäre Orange für diesen Fall geeignet. Die Wiederverbindung mit dem Netzwerk wird über ein weiteres Benachrichtigungsfeld kommuniziert, welches für wenige Sekunden präsentiert wird. Die Zustandsanzeige passt sich inhaltlich an und die restlichen Werte der Betankung werden vom letzten verfügbaren Stand angezeigt.

Hier kann dann die aktuelle Menge als geschätzter Wert unterhalb in einer helleren Schriftfarbe angezeigt werden, um so eine mögliche Fortführung der




(a) ohne Schätzung der Daten (b) mit Schätzung der Daten

Abbildung 3.7: Wireframes von der Übersichtsseite mit einer Netzwerkmeldung

Informationen abzubilden. Die Anpassung der gewünschten Menge kann hierbei durch ein Ausgrauen deaktiviert werden oder trotzdem noch anpassbar sein mit der Erwartung, dass die Eingabe zwischengespeichert und beim Wiederverbinden dann aktualisiert wird. Der Endzeitpunkt wird durch eine Schätzung gleich bleiben aber auch mit einer helleren Schriftfarbe dargestellt um zu zeigen, dass dieser Zeitpunkt nicht mehr auf aktuellen Daten basiert.

Weitere Daten über das Schiff sind in einem Offline-Zustand weniger relevant, da sich die Werte selten ändern. Die Anlegenummer könnte durch einen Vergleich mit der Anlegungsbuchung und der aktuellen Uhrzeit des Gerät auch ohne Netz verändert beziehungsweise entfernt werden.

Während die Daten der Betankung zeitnah seien müssen, ist dies für die Listen der letzten Trinkwasserentnahmen und der Anlegungen weniger relevant, da

diese sich seltener verändern. Diese zeigen die Liste von dem letzten Neu laden der Seite an, doch müssen für den Benutzer immer noch erreichbar sein. Dies gilt auch für die Suchleiste und die Filteroptionen.

Sobald die Netzwerkverbindung wiederaufgebaut ist, werden alle Daten aktualisiert und die Eingaben werden vom System übernommen. Die Schätzungen werden entfernt und die Anzeige wird weiter aktuell gehalten.

# Kapitel 4

# Ergebnisse

Für den Sprung zur Umsetzung bedarf es als Nächstes die Konzeption des technischen Rahmens. Da die Daten vom Prozessleitsystem, welches die Trinkwasserentnahme der Anlegestellen überwacht und verarbeitet, in Form einer Schnittstelle bereitgestellt werden, muss hier geplant werden wie eine Verbindung zwischen den Systemen aussehen könnte.

Hierzu werden Änderungen an den Datenbankmodellen und am Buchungsportal konzipiert. Die technische Integration der Ansicht des Kapitäns ist wiederum abhängig von diesem Buchungssystem, weswegen die Webanwendungen, die die Manager von Schiffsunternehmen und die Kapitäne benutzen, als Frontend [**?**] und die Softwarestrukturen, die für die komplexere Verarbeitung der Daten verantwortlich sind, als Backend [**?**] unterteilt werden.

## 4.1 Backend

Wie in Kapitel **??** beschrieben, ermöglicht das Buchungsportal den Managern Adressen anzulegen, Schiffe im System einzutragen und Anlegebuchungen zu beantragen, die von Mitarbeitern der Fremdenverkehrsgesellschaft über das selbige Portal bearbeitet werden können. Des Weiteren verwaltet der Manager auch die Kapitäne, die im selben Schiffsunternehmen angestellt sind und die eingetragenen Schiffe durch die Personenschifffahrt führt.

Das Backend besteht hierbei aus dem ERP-System Odoo [**?**], welches als primäre Programmiersprache Python [**?**] benutzt und mit dem Model-View-Controller [**?**] (MVC) Muster ein eigenständiges System darstellt. Während die Views zuständig für die Ansichten der Administratoren sind und mit Templates generiert werden und die Controller die Netzwerkanfragen (Requests) verarbeiten, be-

sitzen die Models jeweils Business Logiken [**?**] und legen die Entitätsklassen in einer objektrelationalen Datenbank über die Objektrelationale Abbildung (ORM) fest. Diese Datenbank wird im Odoo durch PostgreSQL [**?**] realisiert.

Da die Webanwendung für die Kapitäne inhaltlich abhängig von diesem System ist und die Hauptlogik in den Models stattfindet, müssen zunächst die Änderungen der Models definiert werden, welche durch die ORM zu Anpassungen in der Datenbank führen.

### 4.1.1 Anpassungen der Datenmodelle

In der relationalen Datenbank sind für das existierende System bereits Entitätsklassen definiert (siehe Abbildung **??**). Das Datenmodell „User“ bildet dabei die Manager und die Mitarbeiter der Fremdenverkehrsgesellschaft, die als Administratoren agieren, ab. Auf den ersten Blick passt auch hier der Kapitän für dieses Datenmodell, da sich gewisse Eigenschaften zwischen einem Kapitän und einem Manager und zwischen einem Kapitän und einem Administrator ab, darstellen.

Da aber ein Kapitän einen eingeschränkten Zugriff bekommen soll und er sowohl von einem Administrator als aber auch von einem Manger erstellt werden kann, ist eine einfache Abbildung des Datenmodells nicht möglich. Auch das Datenmodell für die Rechnungsadressen, die ein Manager anlegt, überdeckt sich nicht komplett mit den Funktionalitäten eines Kapitäns.

Denn der Kapitän muss sich wie ein User einloggen können, was für das Adressen nicht möglich. Genauso muss er von einem Manager angelegt werden können, was einem Manager in diesem Fall nicht möglich ist, da er keine administrativen Berechtigungen für die Erstellung eines Benutzers besitzt.

Bedingt durch die Wireframes von Kapitel 3.1 muss ein weiterer Login Mechanismus implementiert werden, da im aktuellen System eine E-Mail Adresse und ein Passwort zum Einloggen eines Benutzers erforderlich ist. Daher braucht es für den Kapitän ein eigenes Datenmodell, welches eine Relation zum „User“ besitzt, um so die Eigenschaften eines Benutzers zu haben. Außerdem ist die Zuordnung des Kapitäns zu einem Schiff durch den Manager relevant, da dies entscheidet, über welches Schiff der Kapitän die Daten verfügt.

Hierbei zeigt die Abbildung 4.1 die möglichen Änderungen an den Datenmodellen und benutzt die gleiche Notation aus dem Kapitel **??**. So beinhaltet das Datenmodell „Captain“ neben dem „User“ eine Relation zu dem „Ship“ Datenmodell in einem Verhältnis von N:1. Eine Relation zum Manager entsteht durch eine ID, die festhält von wem ein Kapitän erstellt wurde mit einem Verhältnis

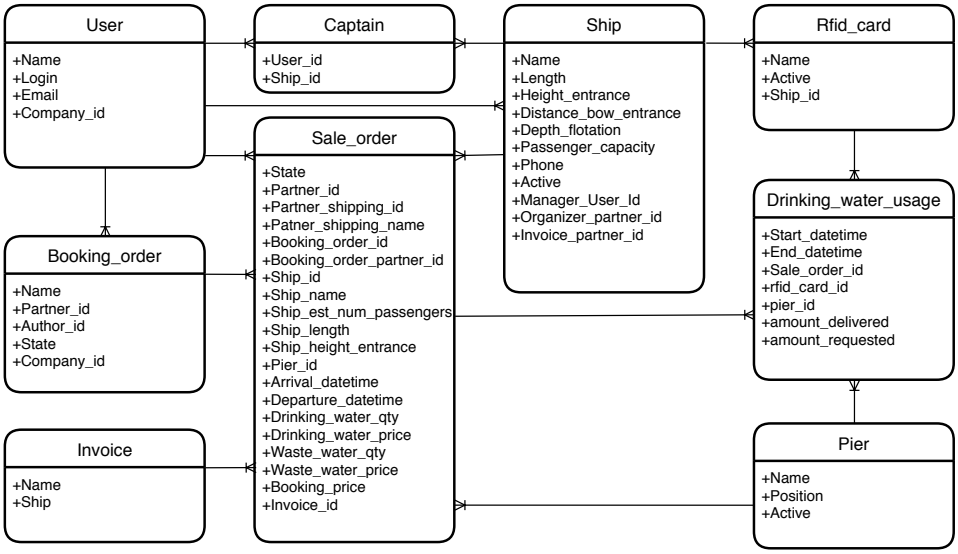



Abbildung 4.1: angepasster Datenmodellausschnitt des Buchungssystems

von 1:N. Der Manager kann also zu einem von ihm erstellten Schiff einen oder mehrere Kapitäne zuteilen. Das „Captain“ Datenmodell soll dazu eine 1:1 Beziehung zu dem „User“ Datenmodell haben, um neben dem Namen die Attribute „login“ und „password“ für den Login Prozess zu besitzen.

Durch die Beziehung zum „Ship“ Datenmodell kann dem Kapitän der Zugriff über alle Anlegebuchungen des ihm zugeteilten Schiffes ermöglicht werden. Damit er auch die Informationen über die Betankung erhalten kann, bedarf es weitere Datenmodelle. Daher sind für diesen Prozess Abbildungen für die Benutzung des Trinkwassers und für die RFID Karte nötig. Es ergeben sich die Datenmodell „Drinking_water_usage“ für die Trinkwasserentnahme und „Rfid_cart“ für die RFID Karte.

Die RFID Karte wird als Entitätsklasse beschrieben um so eine Verknüpfung zwischen der Betankung und dem Schiff beziehungsweise dem Schiffsunternehmen zu deklarieren. Zusätzlich gibt es noch ein Attribut „Active“ zur Bestimmung der Freigabe der Karte. So steht diese Entität in einer 1:N Beziehung zu dem „Ship“ Datenmodell und zu dem „Drinking_water_usage“ Datenmodell.

Letzteres wird benötigt um die Informationen der jeweiligen Trinkwasserentnahme persistent fest zu halten und um diese mit den Verkaufsaufträgen zu verknüpfen. Dadurch können Rechnungen mit den Preisen von den angeforderten und entnommenen Ressourcen erstellt werden. Neben der Verknüpfung wird in dieser Entitätsklasse der Start- und Endzeitpunkt, die angefragte und die ausgelieferte Menge des Trinkwassers und die ID der Anlegestelle eingetragen.

### 4.1.2 Weitere Änderungen

Nachdem die Datenbank um zusätzliche Datenmodelle erweitert wurde, können weitere Änderungen im Backend betrachtet werden. So können weitere Views für die neuen Datenmodelle erstellt werden. Damit die Administratoren alle erstellten Kapitäne verwalten können bedarf es für diese Datenmodelle jeweils eine Listenansicht, die alle Einträge mit einem optionalen Filter anzeigt und eine Formularansicht, die es erlaubt Einträge zu erstellen oder zu bearbeiten.

Auf der administrativen Seite sind für die Erstellung die Felder Name und ID des zugeordneten Schiffes notwendig, E-Mail-Adresse und Telefonnummer sind optional. Der Manager gibt im Formular seiner Ansicht den Namen des Kapitäns an und weist ihm einen von ihm erstellten Schiff zu. Die Erstellung des Kapitäns wird dann in dessen gleichnamigen Datenmodell bearbeitet.

Nachdem das Formular abgeschickt wird, werden weitere Änderungen an den Werten durchgeführt. Die empfangenen Daten werden zunächst validiert, um Sicherheitslücken auszuschließen. Danach bedarf es für die Authentifizierung des Kapitäns eine Erstellung eines „User“ mit angepasstem Login und Passwort Feldern. Die Idee hierbei ist den Zugangscode für beide Felder zu benutzen umso den selben Login Mechanismus des Systems für den Kapitän zu nutzen. Der Zugangscode dient somit als Passwort und muss daher vor dem Prozess erstellt werden.

Obwohl der Kapitän durch diesen Code Zugriff auf Daten erhält, die nur eine geringe Sicherheit erfordern, da sie zu Teilen öffentlich zugänglich sind, bedarf es für eine Mindestsicherheit einen zufällig generierten Code. Dies kann dann mit einem Zufallsgenerator oder einer kryptographischen Hashfunktion [**?**] realisiert werden. Nachdem der Code generiert wird kann dieser als Wert für die Felder Login und Passwort für das „User“ Datenmodell weitergeleitet werden, womit zuerst ein „User“ und danach ein „Captain“ erstellt wird.

Der Manager hat dann die Möglichkeit für einen Schiff einen oder mehrere Kapitäne zuzuweisen und ermöglicht ihm die Daten über Anlegungen und die letzten Trinkwasserentnahmen einzusehen. Damit der Kapitän auch Daten über aktuell durchgeführte Trinkwasserentnahme erhält muss von der administrativen Seite die RFID Karte, die der Kapitän zum Öffnen des Wasserentnahmeschranks benutzt, im System eingetragen und zum Schiff zugewiesen werden. Für die Informationen der entnommenen Ressource bedarf es für den Fall der Betankung des Trinkwassers die Daten, die im Wasserentnahmeschrank erfasst werden. Doch um diese Daten zu erhalten, muss das Backend zunächst mit dem zuständigen System kommunizieren. Dies ist in der folgenden Abbildung zu sehen.

Für die Übersichtlichkeit zeigt die Abbildung 4.2 ein Komponentendiagramm aus der Modellierungssprache Unified Modeling Language (UML) [**?**], welches die Bestandteile für die Durchführung einer Betankung darstellt. Die Komponenten werden von den einzelnen Systemen umrahmt. Die Kreise präsentieren die Schnittstellen der Komponenten und die gestrichelten Pfeile zeigen die Abhängigkeiten zu dem Gezeigten [**?**]. Dort sieht man wie die Beziehung von der Nutzung der RFID Karte bis zur Netzwerkanfrage des Frontends führen kann.

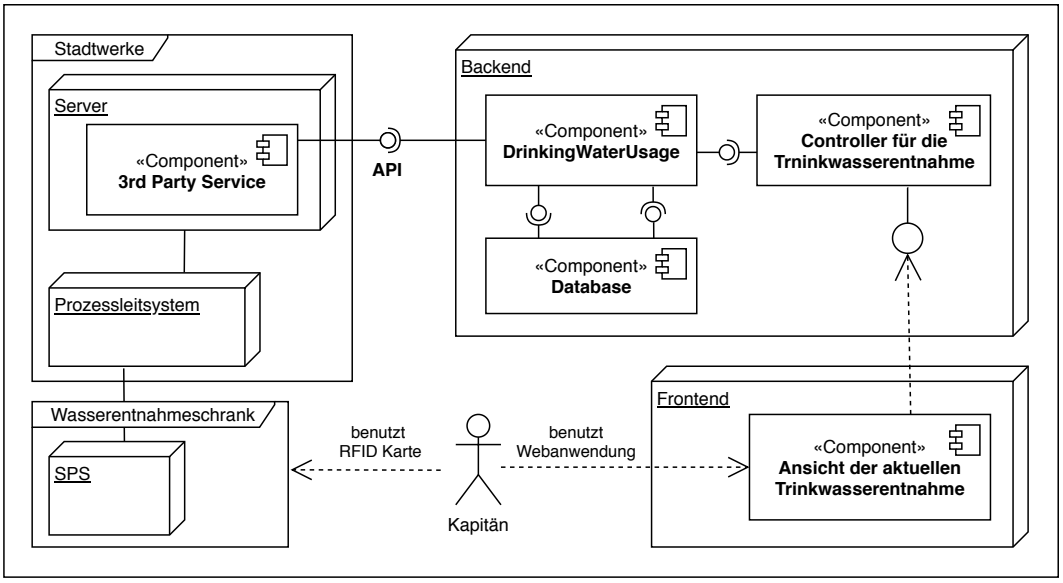



Abbildung 4.2: Zusammenspiel der Systeme

So generiert die SPS durch die Tätigkeit am Wassernentnahmeschrank Daten, die zum Prozessleitsystem weitergeleitet werden und von einer Third-Party Software als eine Schnittstelle bereitgestellt werden. Diese Schnittstelle kommuniziert über Hypertext Transfer Protocol Secure (HTTPS) [**?**] mit Simple Object Access Protocol (SOAP) [**?**] oder JavaScript Object Notation (JSON) [**?**] Daten. Um die Schnittstelle zu benutzen muss das Backend sich gegenüber dem Server authentifizieren und kann dann Requests bezüglich der eingetragenen RFID Karten verschicken.

Im Backend fragt dann das Datenmodell „DrinkingWaterUsage“ die Daten von der Schnittstelle an, speichert diese in der Datenbank zunächst ab und leitet die Daten bei einer Netzwerkanfrage über den Controller weiter. Dort werden die Daten für das Frontend verarbeitet und als Antwort (Response) bereitgestellt.

Um dies zu ermöglichen, bedarf es für die neue Webanwendung einen weiteren Controller, der für die Netzwerkkommunikation zuständig ist. Die Webanwendung wird hierbei mithilfe von einer URL im Browser des Endgerätes aufgerufen

und fragt dann weitere Ressourcen vom Backend an. Über Routing [**?**] werden im Controller diese Requests abgefangen und mit einer Funktionalität verbunden, die eine Response generiert und zurückschickt. Zu erwarten sind daher die Requests für die Webanwendung als Ganzes und für die Daten der aktuellen Betankung, der letzten Betankungen, der Anlegungen und die allgemeinen Schiffsinformationen.

## 4.2 Frontend

Damit dem Benutzer aktuelle Daten auf einer dynamischen Webseite angezeigt werden, bedarf es für das Frontend eine Webanwendung. Die Anpassungen des Backends ermöglichen daher die Architektur dieser Anwendung zu gestalten.

Zunächst muss für die Webanwendung die Login Seite angepasst werden. Da die Seite aus einzelnen Templates besteht, soll das aktuelle Login Formular durch Vererbung für das Login Formular des Kapitäns angepasst werden und zunächst unter dem anderen Formular hinzugefügt werden.

Da dies ein Hypertext Markup Language (HTML) [**?**] Formular ist, soll durch das Drücken des Buttons der Wert des Eingabefeldes an die gleiche URL vom bereits existierenden Formular geschickt werden, wo es dann vom zuständigen Controller verarbeitet. Bei einer erfolgreichen Anmeldung wird dann der Kapitän zu einem HTML Dokument weitergeleitet, welches als Einstiegspunkt für die Webanwendung dient.

Die Anwendung wird zunächst mit dem JavaScript Framework Vue.js [**?**] als Single Page Application (SPA) [**?**] geplant. Dies erlaubt es der Webseite aktuelle Daten ohne ein komplettes Neu laden der Seite zu erhalten, die Navigation im Browser auf die Anwendung einzuschränken und weitere Inhalte per asynchronen JavaScript (AJAX) [**?**] anzufragen. Durch die asynchronen Anfragen werden die Ladezeiten kürzer und die Interaktion mit der Anwendung wird nicht blockiert. Damit die Anwendung im Browser ausgeführt wird, bedarf es hierfür ein vom Backend bereitgestelltes HTML Dokument als Einstiegspunkt.

Um die Anwendung aus Entwicklungssicht übersichtlich und flexibel für zukünftige Erweiterungen zu halten, ergibt sich eine Strukturierung der Funktionalitäten zu organisieren. Die Funktionalitäten sollen als Komponenten modular entwickelt werden und somit wiederverwendbar sein. Da die Komponenten als HTML Elemente aufgerufen werden, ist die Datenstruktur der Anwendung als ein Baum abgebildet. Das heißt, dass die Anwendung über ein Wurzelelement, welches die anderen Komponenten beinhaltet, im HTML Einstiegsdokument aufgerufen wird.

So wird die SPA durch die Module, die die jeweiligen Hauptfunktionalitäten beinhalten, komponiert.

Die Module lassen sich in die Übersicht der aktuellen Trinkwasserentnahme, die Anzeige von den letzten Trinkwasserentnahmen und die Ansicht der gebuchten Anlegebuchungen unterteilen (siehe Abbildung 4.3). Für all diese Module gilt, dass sie auf der Präsentationsebene die Darstellung aus den Wireframes realisieren. Mithilfe von HTML Templates und Cascading Style Sheets (CSS) [**?**] wird das Document Object Model (DOM) [**?**] mit seiner Darstellung zunächst erstellt. Danach werden diese durch die Algorithmen der einzelnen Module mit Daten angepasst.

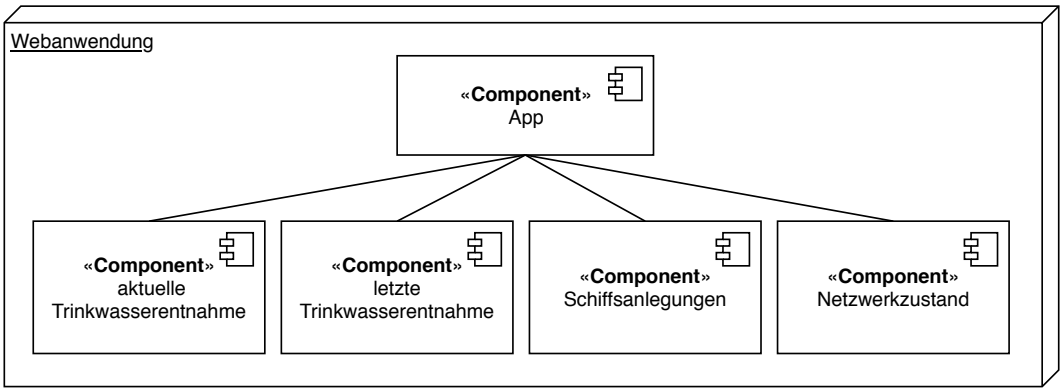



Abbildung 4.3: Aufbau der Webanwendung

Die Module deklarieren zunächst in ihrer Business Logik die Daten und initialisieren diese bei der Instanziierung der Komponente. Für das erste Modul werden so die Variablen für den Zustand der Trinkwasserentnahme, der aktuellen Wassermenge, der gewünschten Wassermenge, des Durchflusses, des Startzeitpunktes, des Endzeitpunktes und der Anlegestellennummer innerhalb eines JavaScript Objektes definiert, welche mindestens für die Präsentationsschicht gebraucht werden. Weitere Variablen wie der aktuelle Zeitpunkt der Daten werden als Metadaten benutzt.

Um die Werte diese Daten zu füllen, werden asynchrone Requests bei der Instanziierung der Komponente verschickt, welche die Daten im JSON Format zurückgeschickt bekommen. Danach werden weitere Requests mit einem festen Zeitintervall geschickt, womit die Daten aktuell bleiben. Damit die Zustände konsistent bleiben und fehlerhafte Fälle vermieden werden, muss die Menge der möglichen Zustände eingeschränkt werden. Dies geschieht durch die Definition eines Aufzählungstypen (Enums) für die Zustände jeweils im Backend und im Frontend. So können die Warnungen, die durch ein Notprogramm vom SPS ausgelöst werden, als mögliche Fälle definiert werden.

Für die Anpassung der gewünschten Menge soll es ermöglicht werden, dass der Wert durch ein Doppeltippen des Benutzers an der dargestellten Stelle mit einem Eingabefeld verändert werden kann. Mögliche Fehlerfälle können vermieden werden, wenn dem Benutzer dies nur für die Fälle bei der eine Betankung bereit zum Starten ist oder bei einer bereits aktiven Betankung. Durch ein Tippen an einer anderen Stelle würde die Darstellung dieses Feldes zurück wechseln, die Eingabe würde dann per Request verschickt werden und abhängig der Netzwerkantwort als Bestätigt dargestellt werden.

Die nächste Komponente soll die letzten Trinkwasserentnahmen darstellen. Hierfür wird bei der Instanziierung der Komponente ein Request abgeschickt, welches ein JSON als Antwort erwartet. Diese Antwort besteht aus einer Liste der vergangenen Entnahmen, durch die dann die Businesslogik iteriert und diese darstellt.

Im Gegensatz zur Komponente der aktuellen Trinkwasserentnahme bedarf es für diese Liste keine regelmäßige Aktualisierung. Die Betankungen finden in Relation zu der Anzahl der Schiffsfahrten seltener statt, weswegen diese Daten beim Neu laden der Seite oder bei einer erneuerten Netzwerkverbindung angefragt werden können. Für eine manuelle Aktualisierung kann ein Button oder ein vertikales Runterziehen der Seite implementiert werden.

Die Anlegebuchungen treten im Bezug zu den Trinkwasserentnahmen häufiger auf, woraus auch ein größerer Datensatz folgt. Diese werden auch beim initialen Laden der Seite vom Backend angefragt. Da die vergangenen, die aktuellen und zukünftigen Anlegungen dargestellt werden sollen, muss bedacht werden wie viele Einträge der jeweiligen Kategorie angezeigt präsentiert werden.

Denn die jeweiligen Zeitintervalle haben eine unterschiedliche Relevanz. Während die vergangenen und zukünftigen Buchungen nur für einzelne Überprüfungen oder Vergleiche benötigt werden, sind die aktuellen Anlegungen für die Mehrheit der Anwendungsfälle essentiell.

Daraus folgt, dass die Anzahl der vergangenen und zukünftigen Anlegungen mit einem Maximalwert eingeschränkt wird, während die aktuellen Buchungen durch ihr Zeitintervall limitiert werden. Dieses Intervall nimmt die aktuelle Systemuhrzeit als Mittelwert und legt als Radius einen definierten Wert fest.

Die Anlegungen werden also je nach gewähltem Zeitintervall gefiltert. Um weitere Requests bei einem Wechsel des Filters zu vermeiden, wird der komplette Datensatz mithilfe einer tiefen Kopie (deep copy) [**?**] dupliziert, mit dem ausgewählten Zeitraum gefiltert, dann nach der Ankunftszeit sortiert und in einer weiteren Variable überschrieben, welche als Referenz für die Darstellung dient. Somit kann der Datensatz gefiltert präsentiert werden, ohne dass der ursprüng-

liche Datensatz in der Anwendung gelöscht wird. Dieser Ansatz wird auch im Bezug zu einem möglichen Netzwerkverlust relevant sein.

Für die allgemeinen Schiffsdaten, die in verschiedenen Komponenten gebraucht werden, muss beachtet werden, in welchem Modul sie aufzurufen sind. Hierfür kann eine bereits existierende Komponente diese Daten anfragen und den anderen Komponenten zur Verfügung stellen oder es wird eine weitere Komponente für diesen Zweck erstellt.

## 4.3 Offline First

Eine weitere Komponente, die es nach den Wireframes bedarf, ist die Anzeige für den Offline-Zustand. Hierbei sollen neben dem Zustand weitere Informationen wie der zeitliche Stand der Daten anhand der vorliegenden Daten dargestellt werden. Doch bevor diese Komponente aufgerufen werden kann, müssen mögliche Ansätze für das Backend und das Frontend betrachtet werden.

Der Eintritt des Offline Szenarios kann zu verschiedenen Zeitpunkten geschehen. Während es bei der Hinfahrt nicht auffällt, ist dieses Auftreten bei einer Entnahme ungünstig für den Benutzer. Da in diesem Prozess regelmäßig Requests vom Frontend geschickt werden, kann durch einen Netzwerkausfall dies vom Backend verfolgt werden.

In diesem Ansatz könnte das Backend einen weiteren Request zu der Schnittstelle der Stadtwerke schicken, um auf diese mögliche Situation hinzuweisen. Hier wäre eine Idee, dass diese Problematik an das Prozessleitsystem gelangt, welches wiederum die Informationen an die SPS und dadurch an die Anzeige am Wasserentnahmeschrank darstellt.

Problem dabei wäre aber, dass solch eine Annahme zu unpräzise ist, als dass das Backend von solchen Fällen konkret ausgehen kann. Denn so kann es sein, dass der Benutzer sich entschlossen hat die Webanwendung vorläufig zu pausieren oder nicht weiter zu benutzen. Daher müssten hier die Bedingungen für weitere Vorgehensweisen im Backend für diese Situation präzisiert werden.

Über das Backend können nur Eingriffe durch die Kommunikation der restlichen Systeme, die hingegen eingeschränkt sind, stattfinden Im Frontend müssen somit die Technologien im Browser genutzt werden, damit die Anwendung und dessen Komponente Informationen über den Netzwerkzustand erhält.

So besitzen alle gängige Webbrowser das Attribut navigator.onLine [**?**] welches als booleschen Wert zurückgibt, ob der User Agent [**?**] mit einem Netzwerk verbunden ist. Dieses Attribut ändert sich daher sobald der Browser die Verbindung

verliert und markiert somit einen Offline-Zustand.

Doch bei einer differenzierteren Betrachtung fällt auf, dass dieses Attribut nicht verlässlich ist, wenn es um den Online Zustand geht. Denn je nach Browser wird der Wahrheitswert bei einer Verbindung mit dem lokalen Netzwerk auf True gesetzt anstatt bei einer Verbindung mit dem Internet [3].

Auf Grund dieser Inkonsistenz wäre ein weiterer Ansatz die Logik für die jeweiligen Browser anzupassen, indem die jeweiligen Schnittstellen der Browser benutzt werden. So könnten mithilfe weiterer Schnittstellen wie zum Beispiel der Network Information API [**?**], die von einigen Browser bereitgestellt wird, dieser Zustand selektiert werden.

Eine weitere Möglichkeit ist es ganz klassisch die fehlerhaften Fälle der regemäßigen Requests abzufangen und anhand dieser zu überprüfen ob es sich um eine fehlende Internetverbindung handelt. Bibliotheken wie Offline.js [**?**] nehmen diesen Ansatz und schicken die fehlgeschlagenen Requests bei einer erneuten Verbindung.

Falls der Offline-Zustand eintritt, wird also dem Benutzer dies über die zuständige Komponente präsentiert. Damit er weiterhin die Webanwendung benutzen kann, müssen die Daten zwischengespeichert werden. Um dies zu ermöglichen, bieten die Browser eine Schnittstelle an, die es der Anwendung erlaubt einen Cache der Anwendung zu erstellen. Dies wurde mithilfe der Appcache Schnittstelle zum ersten Mal ermöglicht. Bei dieser musste eine Manifest Datei angelegt werden, in der die Dateien angegeben werden, die im Browser zwischengespeichert werden damit der Benutzer diese Offline benutzen kann [**?**].

Diese Schnittstelle ist mittlerweile von den Browsern als veraltet eingestuft worden [**?**], da dies zunächst funktionierte aber zu weiteren Mutmaßungen über die Aufgaben der Anwendungen führte. Als Ersatz wird hierfür die Service Worker Schnittstelle [**?**] empfohlen, welche auch von der Mehrheit der Browser unterstützt wird [**?**].

Die Service Worker erlauben es in die Netzwerkkommunikation der Anwendung und des Netzwerkes einzugreifen und die Requests zu bearbeiten. Sie werden asynchron auf einem anderen Thread ausgeführt, wodurch die Nutzung von synchronen Schnittstellen ausgeschlossen wird. Die Requests müssen außerdem über HTTPS verschickt werden um mögliche Sicherheitslücken auszuschließen.

In diesem Fall können dadurch die angefragten Ressourcen vom Backend mithilfe eines Skriptes im Cache gespeichert werden. Dafür muss dieses Skript zunächst von der Webanwendung abgerufen und im Browser für diese Anwendung registriert werden [1]. Ein passender Zeitpunkt dafür ist das Aufrufen der

Webseite.

Hierbei muss beachtet werden, dass der Geltungsbereich (Scope) für den Service Worker nur auf Dateien per Default eingeschränkt ist, die in Unterverzeichnissen liegen. Dieser Scope ist daher vom Pfad der Datei in der Webanwendung abhängig und kann nur über ein Feld im Header der Response für die Datei angepasst werden.

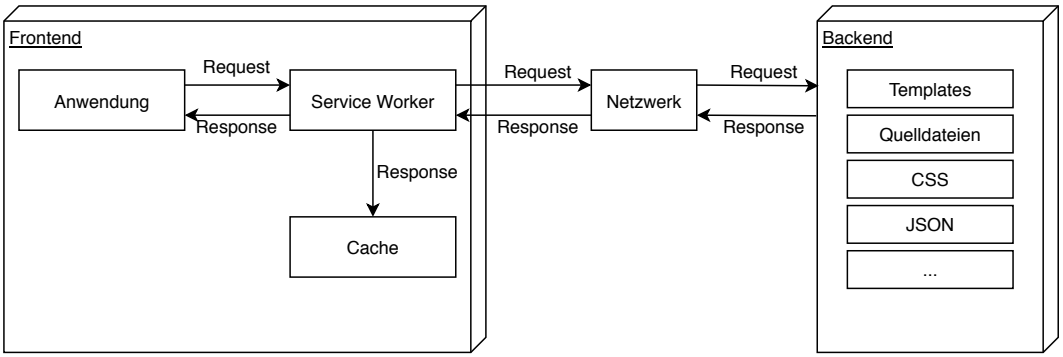



Abbildung 4.4: Requests für neue Daten

Über ein Event, welches der Browser nach der Registrierung auslöst, erhält der Service Worker Kontrolle auf die Seite. Dort kann er dann ein neues Cache Objekt erstellen, welches die im Service Worker festgelegten Pfade der Quelldateien asynchron anfragt und zwischenspeichert. Somit können dem Browser alle nötigen Quelldateien zur Verfügung gestellt werden, um die Anwendung unabhängig von zukünftigen Netzwerkszenarien lauffähig zu halten.

Im nächsten Event können die Service Worker aktiviert und gestartet werden. Dieses Event wird beim Laden und beim Neul aden der Webanwendung ausgelöst und ermöglicht außerdem den Service Worker zu aktualisieren. Wenn also Änderungen am Service Worker stattfinden, wird die daraus resultierende Version als aktuellere Datei vom Browser erkannt. Diese Version wird in einen wartenden Zustand gesetzt, da die Kontrolle bei der aktiven Version liegt. Der Browser löst dieses Problem beim Schließen und Neu Laden der Seite, da dann der alte Service Worker nicht mehr aktiv ist und somit die Kontrolle weitergegeben kann. Das Event zur Aktivierung ermöglicht dann das alte Cache Objekt zu löschen, falls dieser Speicher nicht benötigt wird.

Durch das Event, welches bei jeder Netzwerkanfrage der Anwendung ausgelöst wird, kann überprüft werden ob die angefragte Datei mit einer Datei im Cache übereinstimmt. Trifft dies zu kann der Service Worker die angefragte Datei aus dem Cache anstatt aus dem Netzwerk zurückgeben, was zu kürzeren Ladezeiten führt. Die Herangehensweise um solche Webanwendungen zu konstruieren wird auch als„ offline first“ bezeichnet [**?**].

Falls dies nicht zutrifft, kann der Request an das Netzwerk weitergeleitet werden. So wird, wie in der Abbildung 4.4 dargestellt, der Request von der Anwendung an das Netzwerk abgeschickt, diese wird weitergeleitet an das Backend, welche mit einem Response über das Netzwerk der Webanwendung antwortet. Die erfolgreiche Response wird dann, bevor die Webanwendung es erhält, im Cache überschrieben.

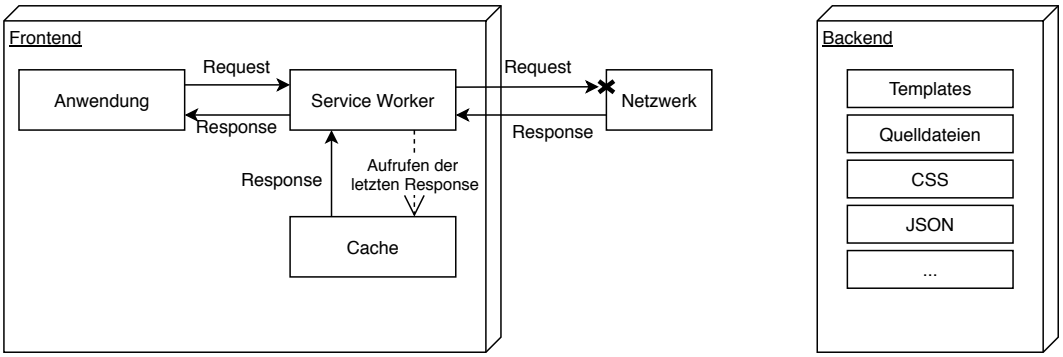



Abbildung 4.5: Requests für neue Daten im Offline-Zustand

Ein Netzwerkverlust würde sich daher erst erkennbar machen, wenn aktuellere Daten angefragt werden. In diesem Fall würde der Service Worker im ersten Schritt die Anfrage zunächst an das Netzwerk weiterschicken, dann eine Response mit einem Fehlerstatus abfangen und daraus folgend die angefragte Datei aus dem Cache aufrufen und zur Webanwendung weiterleiten [**?**]. Die Schritte zu diesem Ablauf sind in der Abbildung 4.5 zu sehen.

Als weitere Möglichkeit können die fehlerhaften Requests durch den Service Worker gesammelt und in einer Warteschlange im Browser mit einer anderen Speicher Schnittstelle aufbewahrt werden [**?**]. Durch das Reaktivieren des Servie Workers, was bei einem Neu Laden der Seite oder bei einem Wiederöffnen des Browsers geschieht, können die Requests überprüft und nochmal versendet werden. Im Fall einer erneuerten Netzwerkverbindung werden so die Request trotz Netzverlust abgeschickt und aus dem Speicher gelöscht. Bleibt jedoch der Offline-Zustand, wiederholt sich der Vorgang bis zum erfolgreichen Absenden weiter.

Der Zweck dieses Ansatzes kann in Teilen zur Anwendung zur passen. Während die regelmäßigen Requests der Daten zur aktuellen Betankung davon nicht profitieren, könnte die Eingabe der gewünschten Trinkwassermenge durch den Benutzer trotz fehlendem Internet erstellt werden.

Durch die Nutzung eines Service Worker würde die Webanwendung eines der Kriterien für eine Progressive Web App (PWA) [**?**] erfüllen. PWAs haben den Ansatz, die Vorteile wie Verlässlichkeit von nativen Anwendungen und die

Technologien von Webanwendungen zu kombinieren.

Verlässlichkeit bedeutet in diesem Fall, dass die Anwendung unabhängig vom Netzwerk lauffähig ist und Teile der Inhalte und Funktionalitäten darstellen kann. Die responsive Gestaltung durch die Webanwendung verschafft es außerdem, dass die Anwendung auf jedem Gerät angezeigt werden kann.

Mit diesen Eigenschaften kann eine PWA plattformübergreifend verfügbar und installierbar werden. Dies würde es dem Benutzer erlauben die Webanwendung auch auf seinem mobilen Endgerät zu installieren, was wiederum eine erweiterte User Experience ermöglichen würde. Des Weiteren kann die Webanwendung für Suchmaschinen optimiert werden. Da die Anwendung aber nur durch eine Weiterleitung des Buchungsportals und somit nur für autorisierte beziehungsweise private Benutzer erreichbar seien soll, eignet sich eine Optimierung für diesen Fall vorläufig nicht.

Die Konzepte der Interaktion und der technischen Struktur haben einen Eindruck über die Problematik dargestellt und ermöglichen es durch die Lösungsansätze die Webanwendung zu implementieren. Diese Ansätze sollen daher durch eine konkrete Umsetzung unter der Berücksichtigung der Rahmenbedingungen in die Realität abgebildet werden.

Hier ist zunächst zu beachten, dass die Bestandteile wie das bereits existierende Buchungsportal mit seinem Frontend für die Manager und das Backend entwickelt wurden. Da die Webanwendung für die Kapitäne das Backend des Buchungsportal als inhaltliche Abhängigkeit benutzt, kann die Anwendung im selben Entwicklungsprojekt implementiert werden. Hierfür wird neben dem Verzeichnis der Anwendung des Managers ein Ordner, welches die Quelldateien für die Webanwendung beinhaltet, angelegt.

Das Backend welches Odoo mit der Version 11.0 als Python-Framework benutzt, dient des Weiteren als Webserver für die Anwendung. Für dieses System wird die Entwicklungsumgebung Pycharm [**?**] benutzt, die auch die Bearbeitung der anderen relevanten Dateiformaten erlaubt. Alle Einträge, auf die Odoo zugreift, werden in einer PostgreSQL Datenbank persistent gespeichert. Durch die Lage der Anwendung in der Verzeichnisstruktur werden die Quelldateien von dem Versionverwaltungssystem (VCS) Git erfasst. Dies ist hilfreich für die Entwicklung, da Änderungen an diesen Dateien versioniert und für mögliche Wiederherstellungen gesichert werden.

Das Projekt wird mithilfe einer Containervirtualisierung entwickelt und ermöglicht damit die gleichen Rahmenbedingungen für alle Entwickler zu schaffen. Dafür werden die einzelnen Komponenten eines Systems jeweils in einem Docker [**?**] Container isoliert um sie jeglicher Hardware zu abstrahieren. Für

das Backend bedeutet das, dass das ERP-System, welches das Buchungsportal repräsentiert und die Datenbank von einem Container aus gestartet werden.

Im Backend wird der Kapitän, die RFID-Karte und der Trinkwasserverbrauch als Datenmodell erstellt. So wird der Kapitän sich wie ein regulärer Benutzer einloggen können, aber auch vom Manager zugeordnet werden. Die Relationen werden durch Anpassungen an den Models implementiert. Die Controller werden die angefragten Einträge aus der Datenbank aufrufen, verarbeiten und in JSON umgewandelt als Response verschickt. Für Entwicklungszwecke wird die zuständige Methode der aktuellen Trinkwasserentnahme Responses mit Demo Daten verschicken, da zu dem Zeitpunkt der Entwicklung die Schnittstelle nicht verwendet werden konnte.

Die Webanwendung wird mit dem JavaScript-Framework Vue.js entwickelt. Des Weiteren werden die konzipierten Funktionalitäten mithilfe von Single File Components (SFC) [**?**], welche ein eigenes Format für unabhängige Vue Komponenten einführen, implementiert.

So können die Vorteile eines SFC genutzt werden um die Struktur und die Gestaltung eigenständig für die Komponenten zu realisieren. Die Gestaltungen basieren hierbei auf dem CSS-Framework Bootstrap [**?**], welches durch die Einbindung der Header und Footer des Buchungsportals im Einstiegsdokument der Webanwendung für die SFC verfügbar sind.

Ein weiterer Vorteil der SFC ist die Nutzung von ECMAScript [**?**] der sechsten Ausgabe oder höher, einer standardisierten Skriptsprache basierend auf JavaScript. Da diese Sprache nicht mit allen Browsern kompatibel ist, bedarf es hier einen Prozess, der diese Sprache übersetzt.

Um dies zu ermöglichen wird Webpack [**?**], ein Entwicklungs- und Bereitstellungswerkzeug, der die Quelldateien und Ressourcen einer Webanwendung in eine Datei bündelt, benutzt. Davor können diese noch mithilfe von Babel [**?**] übersetzt werden, womit die Kompatibilität zu den Browsern entsteht. Für die Benutzung von Webpack bedarf es die Plattform Node.js [**?**], die als Laufzeitumgebung für den Bundler dient.

Zu dem Zweck, dass diese Plattform nicht vom Buchungsportal benötigt wird und dass eine weitere Abhängigkeit vermieden werden soll, erhält Node.js ein eigener Container. Dies erlaubt aus der Entwicklungssicht neben den bereits beschriebenen Funktionalitäten eine dynamischere und zeitnahe Übernahme der Anpassungen der Webanwendung. In diesem Container wird dann Webpack ausgeführt, welches die Webanwendung als eine Datei für die Browser bereitstellt. Des Weiteren wird mit dieser Plattform der Paketmanager npm [**?**] benutzt, was eine Verwaltung weiterer Programmbibliotheken erlaubt.

Ein Zusammenspiel zwischen den Container des Buchungsportals und der Datenbank existiert bereits. Hier wird daher der neue Container durch ein Werkzeug von Docker [**?**] verknüpft, wodurch die Container gleichzeitig gestartet werden und eine Orchestration des Systems weiterhin ermöglicht wird.

Nach dem der Rahmen für die Entwicklung eingestellt wurde, können so die Komponenten aus der Konzeption realisiert werden. Hierfür wird ein Einstiegsdokument für die Anwendung erstellt, welches die Header und Footer des Buchungsportals und eine JavaScript Datei, die als Brücke zur Webanwendung dient, referenziert.

In dieser Datei wird zunächst der Service Worker registriert, falls der Browser dies ermöglicht, bevor die Wurzelkomponente der Webanwendung aufgerufen wird. So wird die Anwendung durch die SFC in einer Baumstruktur entwickelt. Das Wurzelelement der Komponenten importiert die einzelnen Komponenten und ruft diese auf. Für die Netzwerkanfragen wird die Bibliothek Axios [**?**] genutzt, so auch in der ersten Komponente. Von dort werden die Requests für die allgemeinen Schiffsdaten verschickt, da dies Informationen sind, die von jeder Komponente benötigt werden.

Die Komponente der aktuellen Trinkwasserentnahme erfüllt die Darstellung aus den Wireframes, bis auf eine Anpassung der Überschrift für die Übersicht. Des Weiteren werden Requests in einem Intervall von einer Sekunde zunächst verschickt. Die Responses werden dann verarbeitet und füllen die Werte für die Darstellung. Mithilfe von einem Enum, welches sowohl im Backend als auch im Frontend definiert wird, wird der Betankungszustand nur für mögliche Fälle wie einer aktiven Betankung oder eine fehlerhafte Betankung angezeigt.

In der Komponente der letzten Trinkwasserentnahme werden die letzten durchgeführten Betankungen angefragt, als Liste verarbeitet und als Tabelle dargestellt. Der Einklapp-Mechanismus wird mithilfe von Bedingungen und Styling ermöglicht. Für die Komponente der Anlegebuchungen werden zunächst die Daten angefragt. Sobald diese ankommen, werden die Daten nach der Kategorie, die Default eingestellt ist, gefiltert.

Um den Ansatz aus dem technischen Konzept zu ermöglichen wird mithilfe eines Moduls der Bibliothek lodash [**?**] ein deep Copy vor jedem Filterprozess durchgeführt. Da die Kopie mit Zeitintervallen gefiltert wird, benötigt es die Hilfsbibliothek Moment.js [**?**], die es erlaubt Fehler mit Zeitformaten zu vermeiden. Nach der Filterung wird mit dieser Bibliothek die Liste sortiert und dann als Tabelle angezeigt.

Die Komponente der Benachrichtigung über dem Netzwerkzustand wird mithilfe von der Bibliothek v-offline [**?**] realisiert. Die Bibliothek erlaubt es der

Komponente Informationen über das Netzwerk zu erhalten und über die Vue Direktiven in der Präsentationsschicht darzustellen. In dieser Komponente werden die Bedingungen dieses Zustandes genutzt, um die Komponente anzuzeigen und um Elemente der Daten aus den anderen Komponenten anzuzeigen.

Wie bereits beschrieben, wird ein Service Worker für die Anwendung benutzt. Der Service Worker wird vor dem Aufrufen der Webanwendung vom Backend zur Registrierung angefragt. Dazu wird ein Scope für den Service Worker angegeben, damit nur die Speicherung für die Ressourcen der Anwendung zugänglich ist. Um den Scope zu bestätigen, wird der angefragte Service Worker geladen, als Inhalt der Response verwiesen, das Feld für den Scope im Header der Response mit der gleichen URL belegt und die Response verschickt.

Nach der erfolgreichen Registrierung wird das Cache Objekt instanziiert und ein Request erstellt, mit dem die URLs der benötigten Ressourcen angefragt werden. Angegebene URLs sind hierbei das Einstiegsdokument und die Anwendung als gebündelte und laufähige Datei. Für weitere Requests der Anwendung wird der Ansatz gewählt, dass Responses, die noch nicht im Cache vorhanden sind, gespeichert werden und der Webanwendung weitergeleitet werden. Außerdem wird der Service Worker aktualisierbar sein, sodass die alten Cache Objekte auch gelöscht werden.

# Kapitel 5

# Ausblick und Fazit

Im Rahmen der Abschlussarbeit wurde eine offlinefähige Webanwendung für den Bereich der Personenschifffahrt konzipiert und entwickelt. Der Kapitän als Zielgruppe kann mit der Anwendung seinen Arbeitsprozess bei der Trinkwasserentnahme flexibler und interaktiver gestalten. Des Weiteren kann ihm die Offlinefähigkeit Problematiken bei der Arbeit wie den Netzwerkverlust angenehmer formen.

Durch die User Experience wurden Erkenntnisse über die Zielgruppe gesammelt. Eine weitere Disziplin, die man hier zukünftig anwenden könnte, wären die Usability Tests [**?**]. Bei diesem Testvorgang werden den Benutzern der Zielgruppe Aufgaben, die sich um die Interaktion mit der Anwendung drehen, erteilt, um diese bei der Durchführung zu beobachten und die daraus folgenden Ergebnisse zu evaluieren. Dazu wird das Feedback der Kunden und der Zielgruppe benötigt, um nicht bedachte Probleme, die auftreten können, zu lösen.

Die Schiffsbesatzung könnte als weitere Zielgruppe in Betracht gezogen werden. Sollten Aufgaben hinsichtlich der Betankung oder der Anlegung einem Schiffsmitarbeiter zugeteilt werden, muss die Nutzung der Webanwendung über den Kapitän geschehen. Hier könnten weitere Einstellungsmöglichkeiten von Seiten des Managers oder des Kapitäns entgegenwirken.

Die Webanwendung ist neben den Anlegungen vorläufig auf eine Ressource des Schiffes ausgerichtet. Dies kann durch die Einführung der Versorgung von Schiffen mit Landstrom, welches zukünftig von der Fremdenverkehrsgesellschaft an den Landebrücken angeboten wird, erweitert werden. Mit dieser Ressource muss die Ansicht der Hauptseite dann angepasst werden, damit dort auch eine Übersicht der Anzapfung gewährleistet werden kann.

Auch das Abwasser, dass im Betrieb des Buchungsportals bereits abgerechnet

wird, ist eine weitere Ressource, die dem Kapitän über die Anwendung mitgeteilt werden kann. Konzepte wie die Eingabe der gewünschten Menge können ihm erlauben den Prozess der Betankung über die Anwendung zu bestimmen. Dies kann mit einer Reservierung des Trinkwassers erweitert werden. Hierfür müsste der Kapitän die Eingabe während der Fahrt, beziehungsweise während er nicht anlegt, eingeben.

Aus der Entwicklungssicht werden automatisierte Tests helfen problematische Fälle, insbesondere der Offline-Fälle besser zu betrachten. Des Weiteren können weitere technische Funktionalitäten eingeführt werden um dem Kapitän die Webanwendung als PWA anzubieten.

Insgesamt eignen sich Webanwendungen um Ressourcen übersichtlicher darzustellen und mobil zu beobachten. Auch können Offline-Zustände dem Benutzer trotzdem erlauben Aspekte der Anwendung weiter zu benutzen. Aus den Konzepten und der Umsetzung ergeben sich viele weitere Ideen und Ansätze, die realisierbar sind und implementiert werden können.

# Literaturverzeichnis

[1] Sean Amarasinghe. *Service worker development cookbook: Build highly available and performant native web applications that seamlessly integrate with third-party APIs*. Quick answers to common problems. Packt Publishing, Birmingham, UK, 2016.

[2] Andreas Butz and Antonio Krüger. Mensch-maschine-interaktion. In *EBOOK PACKAGE COMPLETE 2017 : EBOOK PACKAGE Engineering, Computer Sciences 2017*, De Gruyter Studium, pages 157–160, 180–181, 208–209. De Gruyter, Berlin and Boston, 2017.

[3] Matt Frisbie and Zach Tratar. *Professional JavaScript for Web Developers*. John Wiley & Sons, fourth edition edition, 2020.

[4] Jens Jacobsen. *Website-Konzeption: Erfolgreiche Websites planen, umsetzen und betreiben*. Dpunkt, Heidelberg, 7., überarb. und erw. aufl. edition, 2014.

# Online-Quellen

[5] Ben Dickson. What is symbolic artificial intelligence?, November 2019.

[6] Docsumo. PDF Scraper - Scrape data from pdf | PDF data extraction, June 2022.