Bachelor Thesis

Entwicklung einer „Package Information Tracker“-Webanwendung

Ievgen Markhai

Gutachter: Prof. Dr. Hannah Bast
Betreuer: Patrick Brosi

Albert-Ludwigs-Universität Freiburg
Technische Fakultät
Institut für Informatik
Professur für Algorithmen und Datenstrukturen

7. Februar 2022

Bearbeitungszeit

09. 11. 2021 – 09. 02. 2022

Gutachter

Prof. Dr. Hannah Bast

Betreuer

Patrick Brosi

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Zur Vereinfachung des Lesevorgangs wird auf gendergerechte Endungen verzichtet und an solchen Stellen der maskuline Fall verwendet.

Freiburg im Breisgau, 07.02.2022 *Ievgen Markhai*

Zusammenfassung

Eines der Hauptprobleme bei der Zustellung von Paketen durch Paketdienstleister ist die Übergabe an den Endverbraucher, vor allem bei der Lieferung an andere anwesende Personen, was oftmals zu einem Verlust der Pakete führt. In dieser Arbeit wird die Entwicklung und Ausführung einer Webanwendung zur Lösung des Problems von verlorenen Paketen innerhalb großer Institutionen, Firmen oder Campusse ohne zentralen Concierge-Service dargestellt. Dafür werden Anforderungen für die Realisierung der Anwendung sowie ein Konzept für die Anwendungsarchitektur definiert. Es werden die Details der Umsetzung von Frontend und Benutzeroberfläche bis hin zur Backend-Implementierung und -Bereitstellung diskutiert. Außerdem wird eine Evaluation durchgeführt mit dem Ziel, die Anwendung weitestgehend zu optimieren. Eine Instanz für die Technische Fakultät der Universität Freiburg wird unter Berücksichtigung von Evaluationsergebnissen bereitgestellt.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Beschreibung des Problems 1

1.2 Ziel der Arbeit 4

1.3 Bestehende Lösungen 5

2 Anforderungen **9**

3 Anwendungsarchitektur **17**

3.1 Allgemeine Konzeptbeschreibung 17

3.2 Frontend-Umsetzung 20

3.3 Backend-Umsetzung 23

3.4 Integration und Bereitstellung 28

4 User Interface und User Experience Design **31**

4.1 Überblick 31

4.2 Anwendungslayout, Start- und Login-Seite 32

4.3 Dashboard und Benutzereinstellungen 35

4.4 Die Seiten „Erwartet“, „Geliefert“, „Abgeholt“ und „Archivierte Pakete“ 39

4.5 Die Seite „Logistikdienste“ 46

4.6 Die Seite „Benutzer“ 46

5 Evaluation **49**

5.1 Empirische Analyse und Bewertung 49

5.2 Testdurchführung und Auswertung 51
5.3 Zusammenfassung 55

6 Fazit **57**

7 Literatur **59**

Abbildungsverzeichnis

1.1 Paketmarkt in Deutschland 2
1.2 Beschwerden und Beschwerdegründe 2014-2020 2
1.3 Verteilung der Beschwerdegründe im Jahr 2020 3
1.4 iLost Webanwendung 5
1.5 Packenda Webanwendung 6

2.1 Anforderungen und Lösungen 9

3.1 API-Endpoint zum Tracken der „cached-time“ 18
3.2 Aufbau der JSON Web Tokens 20
3.3 Verhalten von universeller JavaScript-Anwendung nach erstem Aufruf der Seite 22
3.4 Die Architektur von Node.js 24
3.5 Schematische Darstellung der API für die „Package Information Tracker“-Webanwendung 25
3.6 Die Architektur der „Package Information Tracker“-Webanwendung . 29
3.7 Anwendungsbefehle im Docker-Container 30

4.1 Verfügbare Anwendungsansichten in Abhängigkeit von der Benutzerrolle 32
4.2 Startseite der Anwendung 33
4.3 Mobile Ansicht der Startseite, des Anwendungsmenüs und des modalen Registrierungsdialogs 33
4.4 Benutzer-Registrierungsdialog 34

4.5 Login-Seite 35
4.6 Dashboard Seite 36
4.7 Mobile Ansicht der Dashboard-Seite, Navigationsuntermenü, Benutzereinstellungen 36
4.8 Navigationsuntermenü – Administratorenansicht 37
4.9 Benutzereinstellungen-Seite 38
4.10 Erwartete Pakete Seite 40
4.11 Mobile Ansicht von „Geliefert“, „Abgeholt“ und „Archivierte Pakete“ 40
4.12 Kachel für Anzeige eines erwarteten Pakets 41
4.13 Modaler Dialog für Anzeige eines erwarteten Pakets 42
4.14 Modaler Dialog für „Mich als Empfänger festlegen“ 43
4.15 Paketinformationen, die für Paketempfänger und Paketeigentümer sichtbar sind 43
4.16 E-Mail an den Paketeigentümer, dass ein Empfänger gefunden wurde 44
4.17 Erstellung einer neuen Anzeige 44
4.18 E-Mail-Benachrichtigung über die Aktualisierung der Anzeigeinformationen 45
4.19 Logistikdienste-Seite 46
4.20 Bearbeitung von Logistikdienst-Informationen 47
4.21 Benutzer-Seite 47
4.22 Benutzerdaten – Administrator Ansicht 48

1 Einleitung

1.1 Beschreibung des Problems

Die digitale Transformation und der wachsende Online-Handel haben die Paketdienstleister in letzter Zeit stark beeinflusst. Auch die Corona-Pandemie hat gezeigt, wie wichtig diese Branche ist. Laut dem Jahresbericht 2020 der Bundesnetzagentur [1] stieg das Versandvolumen in Deutschland im sogenannten KEP-Bereich (Kurier-, Express- und Paketzustellung) um 14,4% von 2,36 Mrd. Stück im Jahr 2017 auf 2,7 Mrd. im Jahr 2019. Für 2020 prognostizierte die Quelle eine Zahl von 3,21 Mrd - das hätte in nur drei Jahren eine Steigerung von 36% bedeuten können. Aufgrund der eingeführten Lockdown-Maßnahmen, Homeoffice-Arbeit und der Umstellung der Händler auf Online-Plattformen wird diese Zahl 2020 womöglich deutlich höher ausfallen.

Mit der Zunahme der Paketzustellungen steigt auch die Zahl der Verbraucherbeschwerden. Die Varianz bei den Beschwerdethemen ist besonders hoch und beträgt 32.339 im Jahr 2020 gegenüber 20.738 im Vorjahr und zeigt eine Steigerung um 55,9%. Wie in Abbildung 1.2 zu sehen ist, stehen die genannten 32.339 gegenüber 18.867 tatsächlichen Beschwerden – Grund hierfür ist, dass viele unzufriedene Kunden mehrere Mängel angegeben haben.

Abbildung 1.3 zeigt, dass am häufigsten, in 72% der Fälle, im Jahr 2020 der Grund für die Beschwerde „Mängel bei der Zustellung“ war. Auch wurde kritisiert, dass es oft keine Türzustellung gab. Stattdessen wurde eine Ersatzlieferung zu einem

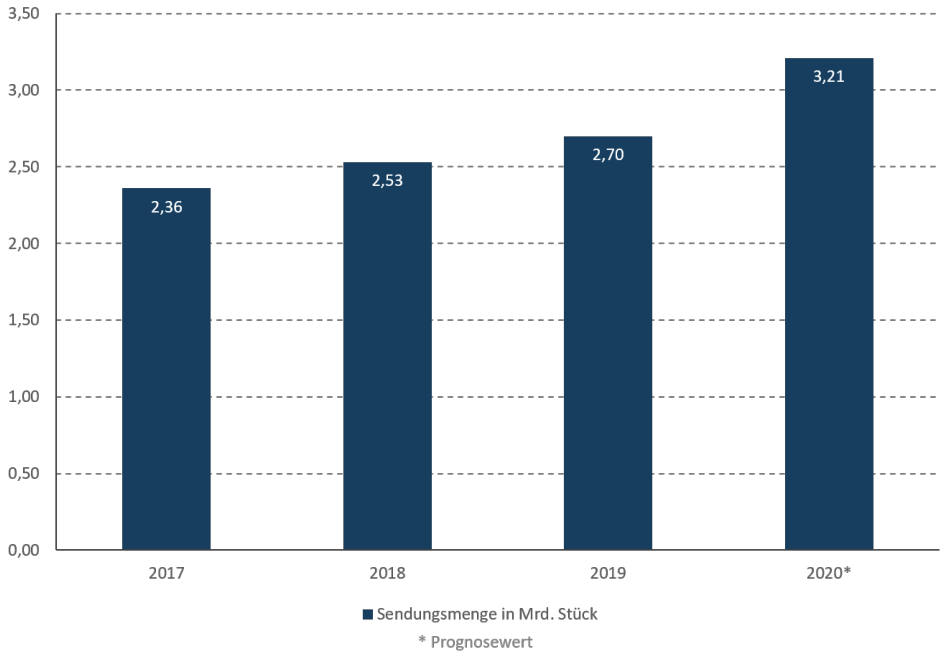


Abbildung 1.1: Paketmarkt in Deutschland [1]

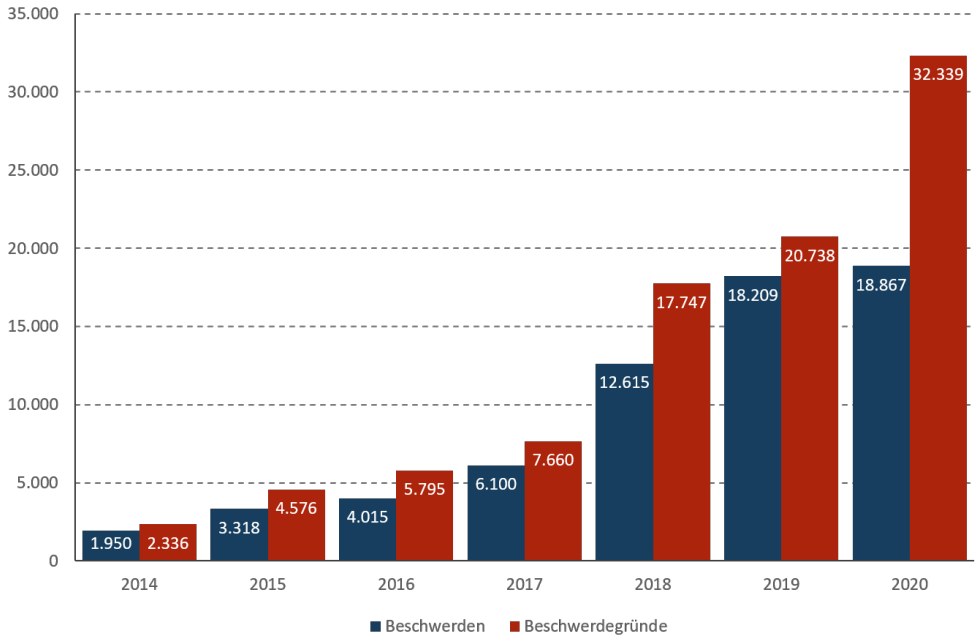


Abbildung 1.2: Beschwerden und Beschwerdegründe 2014-2020 [1]

anderen Termin vorgenommen oder das Paket an eine Abholstation weitergeleitet. Der Verlust, die Beschädigung oder der Diebstahl von Paketen wurde in 14 Prozent der Fälle gemeldet. Fünf Prozent der Beschwerden bezogen sich auf Fehlfunktionen oder Ungenauigkeiten in Tracking-Systemen. Die restlichen neun Prozent der Beschwerden beziehen sich dann laut Bundesnetzagentur auf die Infrastruktur bzw. das Beschwerdemanagement der Paketdienstleister.

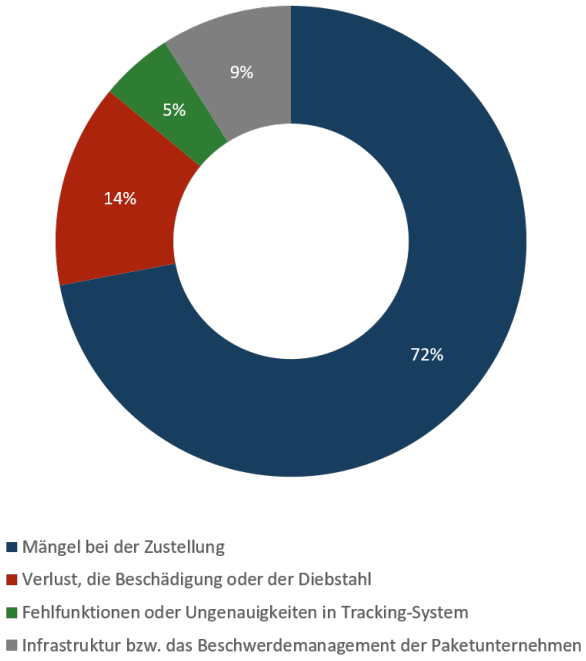


Abbildung 1.3: Verteilung der Beschwerdegründe im Jahr 2020 [1]

Diese Zahlen und Trends zeigen deutlich, dass eines der wesentlichen Probleme bei der Zustellung von Paketen die Übergabe an den Endverbraucher ist. Eine Person hat ein Paket bestellt, eine andere hat es angenommen und es ist nun verloren gegangen.

Dieses Problem tritt besonders häufig in großen Institutionen, Unternehmen oder Universitätscampussen auf. Neben der Tatsache, dass Mitarbeiter nicht immer an ihrem Arbeitsplatz sind und die Möglichkeit haben, ein Paket anzunehmen, erscheint

die Auffindung eines bestimmten Büros oder gar eines Gebäudes oft schwierig. Postboten müssen Pakete an andere Mitarbeiter abgeben sowie Pakete im Flur abstellen. Das ist nicht verwunderlich, denn laut Postmitarbeitern haben sie manchmal nur etwa 2-5 Minuten Zeit [2], um das Paket zu übergeben.

1.2 Ziel der Arbeit

Diese Arbeit ist eine Fortsetzung meines Bachelorprojekts, dessen Ziel es ist, eine Webanwendung zur Lösung des Problems von verlorenen Paketen innerhalb großer Institutionen, Firmen oder Campusse ohne zentralen Concierge-Service zu erstellen. Jedoch soll es vor allem um Pakete gehen, die der Postbote bereits an jemand anderen abgegeben und der Paketbesitzer noch nicht erhalten hat. In solchen Fällen kann das Tracking mittels Logistikdienst-Sendungsverfolgung oder herkömmlicher Anwendungen nicht weiterhelfen oder ist nur eingeschränkt möglich.

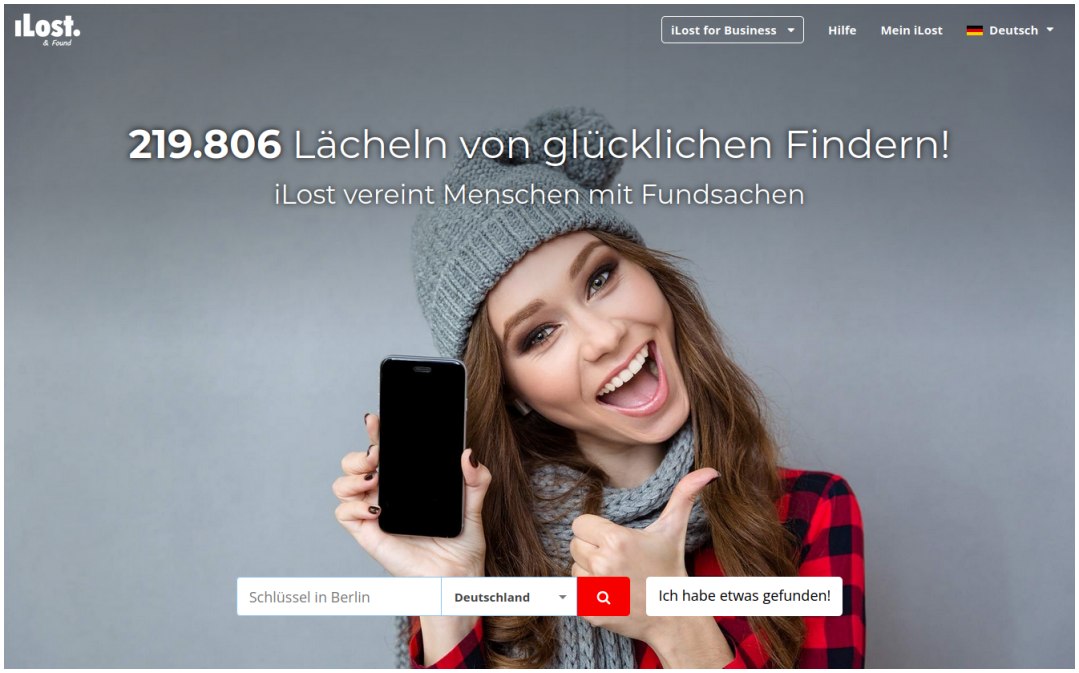
Dabei ist es wichtig, einen zentralen digitalen Ort zu schaffen, um die Pakete möglichst autonom zu verwalten, zu suchen und die Paketeigentümer und -empfänger bei Änderungen automatisch zu benachrichtigen. Auch eine Anbindung an die DHL Tracking-API zur Anzeige des aktuellen Status von Paketen soll implementiert werden.

Diese Bachelorarbeit soll zu einer Webanwendung führen, die einfach konfiguriert und bereitgestellt werden kann. Gleichzeitig wird der Konzeptions- und Entwicklungsprozess von der Definition der Produktanforderungen bis hin zur Usability-Studie untersucht. Anschließend soll eine nutzbare Instanz für die Technische Fakultät der Universität Freiburg bereitgestellt werden.

1.3 Bestehende Lösungen

Bei der Suche nach bestehenden Lösungen wurde keine gefunden, die unsere Hauptanforderungen genau erfüllte und das Problem zwischen der Zustellung des Pakets durch den Postboten und der Annahme durch den Paketbesitzer ohne zentralen Concierge-Service löste.

In Bezug auf die Anforderungen stellten sich Lösungen zum Auffinden von Fundsachen als die am nächsten stehenden heraus. Betrachten wir nun eine davon, und zwar ein Programm namens iLost (Siehe Abbildung 1.4):

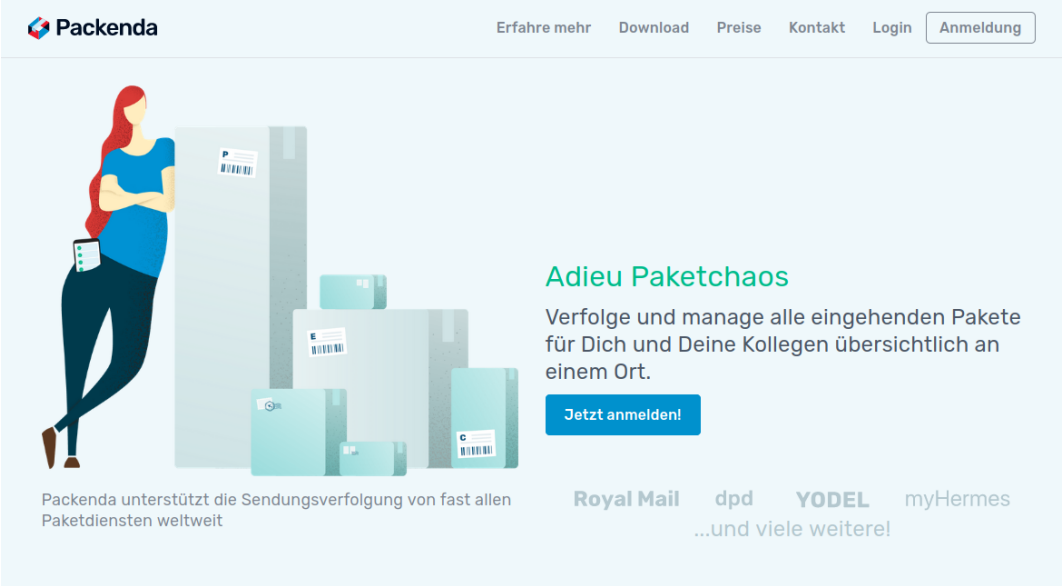


Bildquelle: `https://ilost.co/de`

Abbildung 1.4: iLost Webanwendung

Eine weitere Webanwendung, die unsere Anforderungen nur teilweise abdecken kann, ist Packenda (Siehe Abbildung 1.5). Hier handelt sich um eine App, die Sendungsverfolgungen von unterschiedlichen Paketdienstleistern aggregiert und Empfänger über den Sendungsstatus per E-Mail informiert. Sie kann privat genutzt werden, aber auch in kleinen oder großen Gruppen: Vor allem Office Managern, Sekretären

oder Rezeptionisten greift Packenda unter die Arme, um den Paketempfang ganzer Unternehmen organisieren zu können. Hierbei lassen sich mit Leichtigkeit Pakete in das System hinzufügen, auch bei Erstellung und darauffolgender Benutzung des darin verfügbaren digitalen Mailrooms [3].



Bildquelle: `https://www.packenda.com`

Abbildung 1.5: Packenda Webanwendung

Ein Mailroom ist eine Gruppe, wo beigetretene Nutzer ihre erwarteten Pakete über Sendungsverfolgungsnummern registrieren. Jedes Gruppenmitglied kann dadurch alle Infos über alle erwarteten oder bereits eingetroffenen Pakete der Gruppenmitglieder einsehen. Die App sieht aber keine Interaktion zwischen den Nutzern vor und ist nur zur Überwachung und für Statusupdates mehrerer Sendungen an einem Ort konzipiert.

Es gibt des Weiteren auch viele Softwarelösungen auf dem Markt, die einen Container oder einen speziellen Raum für die Lagerung von Paketen benötigen. Hier aufzuführen wäre bspw. „ParcelLock“, welches Firmenkunden die Installation ihrer Paketkasten-Anlagen zum Empfangen und Versenden von Paketen anbietet. Dieser Service verfügt auch über eine Anwendung zur Verfolgung des Paketstatus. Auch die Firmen „Notifii“

und „PackageX“ haben ähnliche Lösungen. Sie wiederum nehmen die Anwesenheit eines speziellen Raums sowie eines Mitarbeiters an, der mit dem Empfangen und Versenden von Paketen beschäftigt sei.

Auf all diese Alternativen wird in dieser Arbeit nicht näher eingegangen, da unsere Aufgabe darin besteht, Softwarelösungen ohne Concierge-Service zu entwickeln. Die Analyse solcher Systeme hat aber zweifellos bei der Auswahl der Funktionalität unseres Systems geholfen.

2 Anforderungen

In dieser Arbeit wurden die generellen Ziele als Marktanforderungen betrachtet. Diese bilden die Bedürfnisse und Wünsche des Kunden ab und stellen einen Problemraum dar [4]. Zusätzlich werden auch Anforderungen an die Reproduzierbarkeit über Docker für den Lehrstuhl für Algorithmen und Datenstrukturen betrachtet.

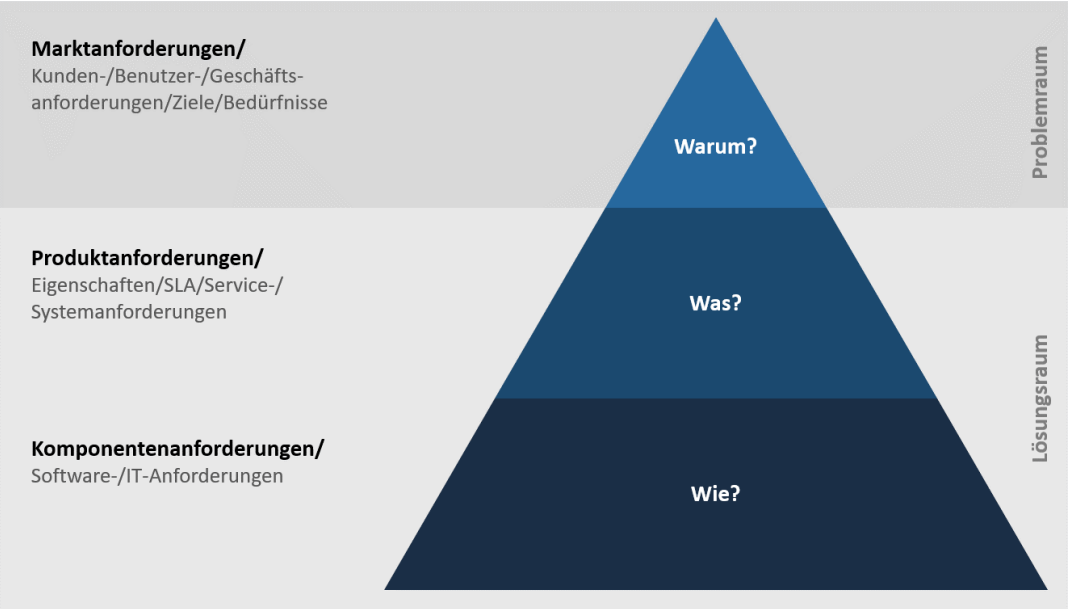


Abbildung 2.1: Anforderungen und Lösungen [4]

Aus diesen zugrunde liegenden Anforderungen werden danach die Produkt- und Komponentenanforderungen definiert. Dabei wird bei ersteren die Funktionalität aus der Sicht der Benutzer genau beschrieben und somit geklärt, wie man die Marktanforderungen in konkrete Lösungen umsetzen kann. Anschließend werden die Pro-

duktanforderungen in Komponentenanforderungen verfeinert und aus der Sicht der Realisierung konkretisiert.

Nachfolgend werden alle Marktanforderungen mit dazugehörigen Produkt- und Komponentenanforderungen in tabellarische Blöcke unterteilt. Dabei werden mithilfe von M (Marktanforderungen), P(Produktanforderungen) und K (Komponentenanforderungen) die Anforderungen gekennzeichnet und durchnummeriert.

M.1. Das System sollte eine Webanwendung sein.

P.1.1. Das Frontend sollte eine JavaScript-Webanwendung sein.

K.1.1.1. Für die Anwendungsentwicklung wird das Web Application Framework Nuxt.js verwendet.

P.1.2. Die Webanwendung sollte für mobile Geräte optimiert sein.

K.1.2.1. Das Frontend-Layout sollte responsiv gestaltet sein (Responsive Webdesign).

K.1.2.2. Mithilfe von Nuxt/PWA [5] soll eine einfache progressive Webanwendung (PWA) erstellt werden.

P.1.3. Das Backend-System soll als RESTful-Webservice implementiert werden.

K.1.3.1. Das Backend wird mithilfe von Node.js und Express.js implementiert.

M.2. Die Anwendung sollte einfach bereitgestellt werden können.

P.2.1. Die Anwendung sollte schnell und einfach auf verschiedenen Hostsystemen zu installieren sein.

K.2.1.1. Das System soll in einem „all-in-one“ Docker-Container implementiert werden.

K.2.1.2. Als Datenbanksystem soll SQLite verwendet werden.

M.3. Das System sollte die Paketzustellungen in großen Institutionen, Unternehmen oder Campussen ohne zentralen Concierge-Service optimieren.

Es soll eine nutzbare Instanz für die Technische Fakultät der Universität Freiburg bereitgestellt werden.

P.3.1. Der Benutzer sollte sich mit E-Mail und Passwort registrieren können.

P.3.2. Der Benutzer sollte sich ohne Registrierung mit einem Uni- oder Firmenkonto anmelden können (LDAP-Authentifizierung).

M.4. Es sollte möglich sein, die Pakete möglichst autonom zu verwalten.

P.4.1. Der Benutzer sollte in der Lage sein, einen Antrag für erwartete oder gelieferte Pakete zu erstellen.

K.4.1.1. Beim Erstellen des Paketantrags sollte es möglich sein, eine Paketbezeichnung, eine Sendungsnummer, eine Logistikleistung sowie einen Paket-Kommentar und ein Foto des Pakets einzugeben. Dabei muss der Name des Empfängers/die Paketbezeichnung oder die Sendungsnummer obligatorisch für die Auflistung des Pakets sein.

P.4.2. Der Benutzer sollte in der Lage sein, den eigenen Antrag zu bearbeiten.

K.4.2.1. Es sollte eine „Option“ für Paketantrag-Herausgeber geben, um den Paketantrag bearbeiten zu können.

P.4.3. Der Benutzer sollte in der Lage sein, sich selbst als Empfänger der erwarteten Pakete eintragen zu können.

K.4.3.1. Die vorhandenen erwarteten Pakete sollten eine Option haben, sich als Empfänger festzulegen.

K.4.3.2. Mit der Option, sich als Empfänger festzulegen, sollte es möglich sein, ein Foto des Pakets hinzuzufügen und eine Nachricht an den Eigentümer zu verfassen.

K.4.3.3. Mit der Option, sich als Empfänger festzulegen, sollte eine automatische Email-Benachrichtigung über die Aktualisierung des Paketstatus an den Eigentümer geschickt werden.

P.4.4. Der Benutzer sollte sich als Eigentümer der gelieferten Pakete eintragen können.

K.4.4.1. Die vorhandenen erwarteten Pakete sollten eine Option haben, sich als Eigentümer festzulegen.

K.4.4.2. Mit der Option, sich als Eigentümer festzulegen, sollte es möglich sein, eine Nachricht an den Empfänger zu verfassen.

K.4.4.3. Mit der Option, sich als Eigentümer festzulegen, sollte eine automatische Email-Benachrichtigung an den Empfänger geschickt werden.

P.4.5. Der Herausgeber der Paketanfrage sollte in der Lage sein, falsch eingetragene Paketeigentümer oder Paketempfänger zu löschen.

K.4.5.1. Bei der Verarbeitung von Paketinformationen sollte der Herausgeber der Paketanfrage andere Benutzer (des Weiteren als „Opponent“ bezeichnet) aus diesen löschen können, wenn ein solcher sich versehentlich oder fälschlicherweise als Paketeigentümer oder -empfänger eingetragen hat. In solch einem Fall sollte das Paket den Status von „geliefert“ auf „erwartet“ zurück ändern, wenn der Herausgeber der „Eigentümer“ ist. Hat sich der Opponent selbst als „Eigentümer“ festgelegt, sollte das Paket den Status „zugestellt“ beibehalten.

P.4.6. Der Paketeigentümer oder -empfänger sollte das Paket als abgeholt kennzeichnen können.

K.4.6.1. Es sollte eine Option geben, das Paket als „abgeholt“ zu markieren. Diese Option sollte nur für den Paketeigentümer und -empfänger sichtbar sein.

P.4.7. Der Herausgeber der Paketanfrage sollte in der Lage sein, sein eigenes Paket zu löschen.

K.4.7.1. Es sollte eine Option für den Herausgeber der Paketanfrage geben, den Antrag zu löschen.

P.4.8. Der Paketeigentümer oder -empfänger sollte in der Lage sein, den Gegenüberstehenden zum Abholen des Pakets leicht zu finden.

K.4.8.1. Bei der Registrierung wird dem Nutzer vorgeschlagen, die Büroadresse und die Zimmer-/Büronummer sowie weitere Angaben zur Adresse einzugeben.

K.4.8.2. Die Breiten- und Längenkoordinaten sollten mit Hilfe der OpenCage-Geocoding API automatisch aus der Benutzeradresse berechnet werden. Wenn die Benutzeradres-

se nicht eingegeben wird, werden Organisationskoordinaten verwendet.

K.4.8.3. Der Paketeigentümer oder -empfänger sollte die Büroadresse und die Zimmer-/Büronummer sowie weitere Informationen über die Adresse des Gegenüberstehenden einsehen können. Außerdem sollte der Standort dieses anderen Benutzers in OpenStreetMap als iFrame angezeigt werden.

P.4.9. Pakete, die über einen zu großen Zeitraum nicht aktualisiert oder abgeholt wurden, sollten automatisch gelöscht werden.

K.4.9.1. Alle Pakete, die 30 Tage lang nicht aktualisiert wurden, sollten ins Archiv verschoben werden.

K.4.9.2. Die Pakete, die einen Paketeigentümer und einen -empfänger haben und von keinem von beiden innerhalb von sieben Tagen als „abgeholt“ markiert wurden, sollten ins Archiv verschoben werden.

K.4.9.3. Pakete, die als „abgeholt“ gekennzeichnet sind, sollten binnen 24 Stunden ins Archiv verschoben werden.

K.4.9.4. Alle Pakete im Archiv sollten nach 30 Tagen endgültig gelöscht werden.

M.5. Es sollte möglich sein, die Pakete zu suchen.

P.5.1. Der Benutzer sollte in der Lage sein, mithilfe einer Suchfunktion ein Paket zu finden.

K.5.1.1. Es sollte ein Suchfeld für gelieferte und erwartete Pakete geben, um nach dem Paket suchen zu können. Es sollte möglich sein, nach dem Namen des Empfängers (falls dieser eingegeben wurde)/der Paketbezeichnung , Sendungsverfolgungsnummer, Logistikdienstleistung und Paketkommentar zu suchen.

K.5.1.2. Es sollte möglich sein, Pakete zu filtern und nur Pakete, die in Verbindung mit dem Benutzer stehen („Kann ich abholen“, „Habe ich angenommen“, „Von mir erwartet“, „Von mir abgeholt“, „Vom Besitzer abgeholt“), Pakete ohne Sendungsverfolgungsnummer oder alle Pakete anzuzeigen.

K.5.1.3. Der Benutzer sollte in der Lage sein, alle Pakete nach Sendungsnummer

zu sortieren, jedoch ebenso nach „Zuletzt Aktualisierte zuerst“, „Älteste zuerst“ und „Neueste zuerst“.

P.5.2. Der Nutzer sollte in der Lage sein, den Link zum Paket kopieren, um ihn mit anderen teilen zu können und so die Suche zu erleichtern.

K.5.2.1. Es sollte eine Option geben, um den Link in die Zwischenablage zu kopieren.

M.6. Das System sollte den Paketeigentümer und -empfänger automatisch über die Änderungen benachrichtigen.

P.6.1. Der Paketeigentümer sollte über alle Änderungen an seinen Paketen per E-Mail informiert werden.

K.6.1.1. Der Paketeigentümer sollte über alle Änderungen an seinen Paketen mithilfe einer automatisierten E-Mail-Benachrichtigung informiert werden.

K.6.1.2. In den Benutzereinstellungen sollte es möglich sein, E-Mail-Benachrichtigungen zu deaktivieren.

M.7. Das System sollte an mindestens eine Tracking-API zur Anzeige des aktuellen Status von Paketen angebunden sein.

P.7.1. Eine Anbindung an die DHL-API sollte implementiert sein.

K.7.1.1. Beim Anlegen eines Paketantrags und der Eingabe einer Sendungsverfolgungsnummer und DHL als Logistikdienst sollten Paketdaten von DHL abgefragt werden.

K.7.1.2. Wenn bei der Aktualisierung eines Paketantrags die Sendungsverfolgungsnummer und DHL als Logistikdienstleistung eingegeben wurde, sollten die Paketdaten von DHL angefordert und aktualisiert werden.

P.7.2. Wenn eine Tracking-API-Anbindung besteht, sollten die Paketinformationen automatisch einmal täglich aktualisiert werden.

K.7.2.1. Es sollte ein „Worker“ implementiert werden, der alle bestehenden Tracking-

APIs registriert und jede registrierte Logistikdienstleistung einmal täglich nach Paketänderungen abfragt.

P.7.3. Auch wenn keine Tracking-API-Verbindung zum Logistikdienstleister besteht, sollte eine Weiterleitung auf die Tracking-Seite des Logistikdienstleisters ermöglicht werden können.

K.7.2.1. Verfügt ein Paketantrag über eine Sendungsverfolgungsnummer und einen Logistikdienstleister, wird ein Button zur Weiterleitung auf die Sendungsverfolgungsseite des Logistikdienstleisters angezeigt.

M.8. Die Anwendung sollte konfigurierbar sein.

P.8.1. Das Frontend-System sollte konfigurierbar sein.

K.8.1.1. Es sollte eine Konfigurationsdatei für das Frontend geben. Die wichtigsten Einstellungen sollten auch über Umgebungsvariablen definierbar sein.

K.8.1.2. Die Benutzeroberfläche sollte Mehrsprachigkeit unterstützen und in Englisch und Deutsch verfügbar sein. Es sollte einfach sein, eine neue Sprache hinzuzufügen.

P.8.2. Das Backend-System sollte konfigurierbar sein.

K.8.2.1. Es sollte eine Konfigurationsdatei für das Backend geben. Die wichtigsten Einstellungen sollten auch über Umgebungsvariablen definierbar sein.

K.8.2.2. Es sollte sollte möglich sein, eine neue Tracking-API-Verbindung mit dem Logistikdienstleister herzustellen.

K.8.2.3. Es sollte möglich sein, eine andere SQL-Datenbank anstelle von SQLite zu verwenden.

K.8.2.4. Es sollte möglich sein, die E-Mail-Templates zu bearbeiten und zu erweitern.

K.8.2.5. Es sollte möglich sein, bestehende „Workers“ anzupassen und mit Leichtigkeit weitere hinzuzufügen. Für die Planung der „Workers“-Aufgabe sollte die crontab-Syntax verwendet werden.

P.8.3. Das System sollte beim Starten des Docker-Containers eine Kommandozeilenschnittstelle haben.

K.8.3.1. Nach dem Starten des Docker-Containers im interaktiven Modus sollte eine Meldung ausgegeben werden: „Welcome to this Docker container, type ‘make help’ to get some help“. Alle möglichen Befehle zum Starten, Verwalten, Testen und Überwachen des Systems sollten nach Eingabe des Befehls „make help“ aufgelistet sein.

In den folgenden Kapiteln wird ein genauerer Blick auf die Anwendungsarchitektur und „User Interface und User Experience Design“ geworfen und beschrieben, warum es die soeben aufgelisteten Komponentenanforderungen sind, die aus den Marktanforderungen abgeleitet wurden.

3 Anwendungsarchitektur

3.1 Allgemeine Konzeptbeschreibung

Als JavaScript 1995 entwickelt wurde, war es hauptsächlich für dynamische Effekte und Änderungen des Inhalts einer Website im Browser [6] verantwortlich. Dagegen sind die Anwendungsgebiete heute vielfältiger und die Rolle der Programmiersprache viel wichtiger geworden. So lassen sich beispielsweise mit Hilfe von Node.js und Express.js serverseitige JavaScript-Anwendungen entwickeln, die hochskalierbar, extrem effizient und echtzeitfähig sind [7]. Es gibt auch viele JavaScript-basierte Client-seitige Frameworks wie Vue.js, Nuxt.js, React.js oder Angular.js, die es ermöglichen, eine moderne dynamische Webanwendung zu entwickeln. Durch die Kombination von Node.js, Chromium und Frontend-Technologien ist es nun auch möglich, plattformübergreifende Desktop-Anwendungen oder sogar Progressive Web Apps (PWA) und mobile Anwendungen zu entwickeln [8, 9].

Aufgrund der Vielfältigkeit von JavaScript wurde entschieden, diese Sprache für die Entwicklung des Backends und Frontends einer „Package Information Tracker"-Webanwendung zu verwenden. Dies macht dann die Implementierung von Full-Stack-Anwendungen (Backend und Frontend) einheitlicher und schneller unter Verwendung der gleichen Technologien, Entwicklungstools und Frameworks (z.B. Testframeworks).

Die Logik- und Inhaltsverwaltung ist in einem REST-Architekturstil (Representational State Transfer) implementiert. Dies ermöglicht auch die Anbindung von Desktop- oder mobilen Anwendungen bei der Weiterentwicklung des Projekts. Da wir davon ausgehen, dass es nur einige hundert Anzeigen gleichzeitig geben wird, wurde sich für die Verwendung eines SQLight Datenbanksystems entschieden. Zudem werden die veralteten und abgeschlossenen Anzeigen automatisch gelöscht und somit wächst die Datenmenge nicht ständig an. Um die Anzahl der Anfragen an die Datenbank zu reduzieren, eine Skalierung zu ermöglichen und die Antwortzeit von REST-API-Anwendungen zu verbessern, wird eine In-Memory-Caching-Lösung namens Redis verwendet.

Das Frontend der „Package Information Tracker“-Webanwendung ist als Responsive Single-Page-Anwendung mit serverseitigem Rendering implementiert. Dies ermöglicht ein schnelles Laden beim ersten Aufruf der Seite, entlastet den Server und reduziert den Datenaustausch bei weiterer Benutzung der Anwendung. Zudem wird die Anwendung um eine „Progressive Web App“ (PWA) erweitert, wodurch sie mit geringem Aufwand aus dem Browser auf dem Homescreen gespeichert werden. Die Anwendung erhält dann ein eigenes Icon und sieht aus wie eine native App, die den Nutzern einen schnelleren Zugriff und die bestmögliche Benutzererfahrung bieten soll. Die Kommunikation zwischen REST-API und Webanwendung wird mit Hilfe

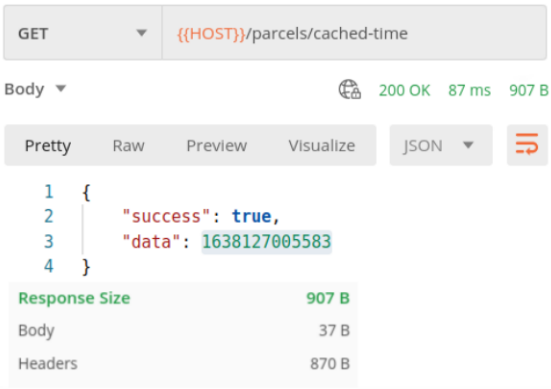


Abbildung 3.1: API-Endpoint zum Tracken der „cached-time“

des Datenformats JavaScript Object Notation (JSON) realisiert. Durch die Verwendung von Cache und zusätzlichen API-URLs für „cached-time“ (siehe Abbildung 3.1) hat die Anwendung die Möglichkeit, die Änderungen zu verfolgen und die Daten nur bei Bedarf zu aktualisieren. Solche Anfragen sind recht klein und werden vom Server sehr schnell verarbeitet. Dies ermöglicht auf Kundenseite eine Pseudo-Echtzeit-Aktualisierung in einem eingegebenen Zeitraum. Das bedeutet, dass die Daten im Browser im Hintergrund aktualisiert werden, ohne die Seite neu laden zu müssen oder Socket-Kommunikation zu verwenden.

Die Benutzer der Anwendung sind in vier Rollen mit unterschiedlichen Benutzerrechten unterteilt:

- Nicht angemeldete Benutzer;
- Angemeldete (interne) Benutzer;
- Moderatoren;
- Administratoren.

Hierbei werden die Benutzerrechte sowohl im Frontend als auch im Backend bei jeder Anfrage überprüft.

Um eine sichere Benutzerauthentifizierung und Autorisierung zu erstellen, wird ein JSON Web Token (JWT) im Autorisierungsheader der HTTP-Anfrage an die Backend-API verwendet. Solche Token werden gebraucht, um „zustandslose Sitzungen“ zu implementieren, da alle erforderlichen authentifizierungsbezogenen Informationen in einem Token übergeben werden und die Sitzung nicht auf dem Server gehalten werden muss [10]. Unsere Anwendungs-JWT enthält folgende benutzerspezifische Daten (siehe Abbildung 3.2):

- “id” – Benutzer ID;
- “role” – Benutzerrolle;
- “activeUser” – zeigt an, ob ein Benutzer gesperrt ist oder nicht;
- “dh” – Gerät-Hashwert;

- “iat” – die Zeit, zu der das JWT ausgestellt wurde;
- “exp” – das Ablaufdatum des Tokens.

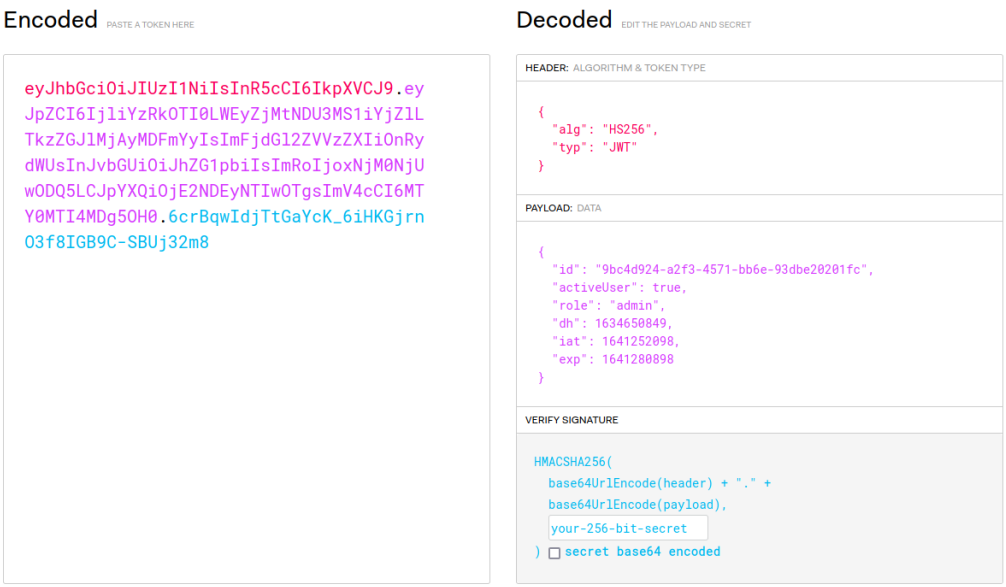


Abbildung 3.2: Aufbau der JSON Web Tokens [11]

Der Geräte-Hashwert bildet die ausgewählten nutzer- und geräteabhängigen Daten mit Hilfe einer schnellen String-Hashing-Funktion auf eine Zahl zwischen 0 und 4294967295 ab. Dieser Token wird bei jeder Anfrage an das Backend überprüft und stellt so sicher, dass er von Dritten nicht ausgenutzt wird.

In den nächsten Unterkapiteln werfen wir einen genaueren Blick auf die Frameworks, die für die Entwicklung von Frontend und Backend verwendet werden und diskutieren die wichtigen Aspekte. Am Ende des Kapitels werden wir uns dann mit der Integration und Bereitstellung der Anwendung befassen.

3.2 Frontend-Umsetzung

Für die Frontend-Entwicklung wurde das Web Application Framework Nuxt.js (Version v2.15.8) verwendet. Nuxt.js ist ein Open-Source-Webanwendungs-Framework

basierend auf Vue.js, Node.js, Webpack und Babel.js [12]. Es ist vorkonfiguriert, um eine produktionsreife, universelle (oder isomorphe) JavaScript-Anwendung zu entwickeln, die sowohl auf dem Client als auch auf dem Server ausgeführt und gerendert werden kann [13]. Dies bedeutet, dass eine solche App die Vorteile von dynamischen Websites und Single-Page-Anwendungen (SPA) nutzen kann [14]:

1. Beim Öffnen der Seite wird eine Anfrage vom Client an den Server gesendet (siehe Abbildung 3.3). Der Server holt die aktuellen Daten von der REST-API, generiert ein HTML-Dokument mit ausgefüllten Daten und sendet diese Daten zurück an den Browser des Benutzers. Der generierte HTML-Code wird sofort vom Browser gerendert und angezeigt.

2. Daraufhin beginnt ein „Rehydration"-Prozess, der die Anwendung in eine SPA umwandelt.

3. Nach der Konvertierung in eine Single-Page-Anwendung wird das Navigieren zwischen Seiten auf der Client-Seite erfolgen. Das serverseitige Rendering wird dabei nicht erneut ausgeführt.

Eine solche Implementierung ermöglicht nicht nur ein schnelleres Laden und eine kürzere Zeit bis zur ersten Interaktion, sondern ist auch suchmaschinenfreundlich.

Nuxt verfügt über eine komponentenbasierte Architektur und bietet eine vordefinierte Dateistruktur basierend auf „Best Practices" [15]. Auch diese Dateistruktur kann an die Anforderungen der Anwendung angepasst und erweitert werden. Darüber hinaus verfügt Nuxt über eine vorinstallierte State-Management-Bibliothek namens Vuex. Diese dient als zentraler, nicht persistenter Speicher für alle Komponenten der Anwendung. Das bedeutet, dass alle vom Benutzer benötigten Daten lokal im Vuex-Store verwaltet und nur bei Bedarf aktualisiert werden. Aufgrund der Reaktivität (Abhängigkeitsverfolgung) von Vuex bei Änderungen im Store werden die dem Benutzer präsentierten Daten in Echtzeit aktualisiert.

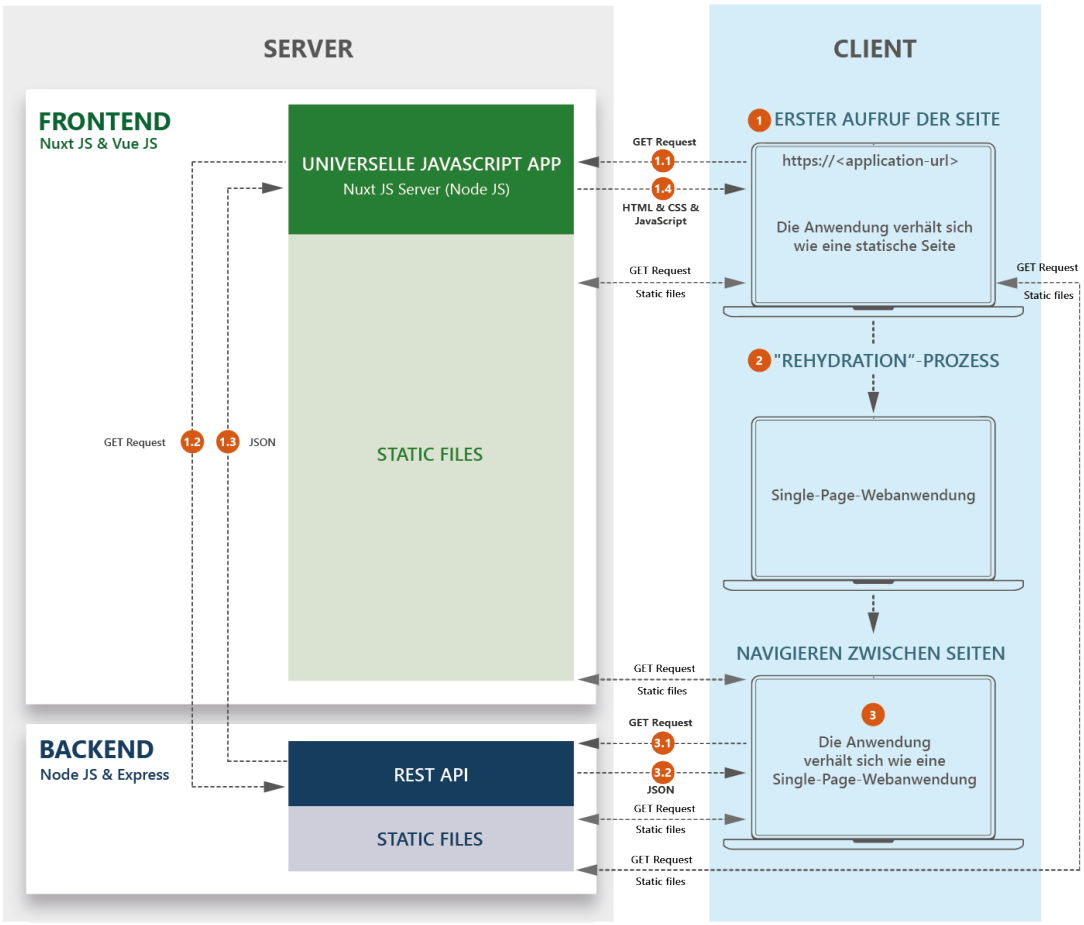


Abbildung 3.3: Verhalten von universeller JavaScript-Anwendung nach erstem Aufruf der Seite

Zur Umsetzung des Frontend-Designs wurde Vuetify – ein „Material Design Framework“ für Vue.js – verwendet. Vuetify bietet eine Reihe von User-Interface-Komponenten, die an ein eigenes Projektdesign angepasst werden können und so den Entwicklungsprozess beschleunigen. Außerdem verfügt Vuetify in Kooperation mit Nuxt.js über ein automatisches „tree-shaking“, das ungenutzte Elemente aus dieser UI-Bibliothek nicht in den Produktions-Build einbezieht.

Der Frontend-Projektordner enthält außerdem einen Konfigurationsordner, in dem viele verschiedene Einstellungen wie das Aktivieren oder Deaktivieren von LDAP und E-Mail-Login, Weiterleitung auf die Impressums- und Datenschutzseite und vieles

mehr zu finden sind. Auch Informationen zum Anwendungs-Branding können hier angepasst werden.

Im gleichen Ordner befindet sich ein Ordner „lang“, der die Übersetzungen und die Liste der verfügbaren Anwendungssprachen enthält. Um eine neue Sprache hinzuzufügen, muss die englische Version en-US.js kopiert, in eine andere Sprache umbenannt (z.B. uk-UA.js) und dann die entsprechenden Einträge übersetzt werden. Außerdem müssen ggf. die Dateien „Impressum“ und „Datenschutz“ (mit Markdown geschrieben) im Ordner „content“ für andere Sprachen übersetzt werden, da ansonsten die Informationen in der Standardsprache Englisch angezeigt werden.

Weitere Nuxt.js-spezifische Einstellungen sowie die Einstellungen für die PWA sind in der Datei nuxt.config.js im Root-Verzeichnis des Frontends aufzufinden.

3.3 Backend-Umsetzung

Die Backend-REST-API wurde mit Hilfe von Node.js und Express.js implementiert. Node.js ist eine JavaScript-Laufzeitumgebung basierend auf der V8-JavaScript-Engine von Chrome [16]. Express.js ist das de-facto-Standard-Server-Framework für Node.js und wurde zum Erstellen von Webanwendungen und APIs entwickelt [17].

Die Architektur von Node.js basiert auf unterschiedlichen Softwarekomponenten, die sich auf die Erledigung nur einer bestimmten Aufgabe jeweils konzentrieren und über Schnittstellen zusammenarbeiten. Eine solche Organisation wurde von der Unix-Philosophie inspiriert und gilt nicht nur für die Kernbibliotheken, sondern auch für die mit Node.js entwickelten Anwendungen [18]. Dank des mitgelieferten Paketmanagers namens NPM kann die Kernfunktionalität von Node.js erweitert werden. Entwickler können auch eigene Pakete erstellen und diese der NPM-Community zur Verfügung stellen.

Eine wichtige Besonderheit von Node.js ist, dass es ein ereignisgesteuertes Design hat und blockierende Ein- und Ausgabeoperationen asynchron verarbeiten kann. So kann ein Node-Server im Vergleich zu herkömmlichen Serveranwendungen eine Vielzahl von Anfragen und Verbindungen verarbeiten und dafür nur einen Thread verwenden. Handelt es sich bei der Anfrage um eine blockierende Ein- oder Ausgabeoperation, wird ein zusätzlicher interner Node.js-Prozess (Thread) gestartet und, sobald dies erfolgt ist, das Ergebnis an den Hauptthread übergeben (siehe Abbildung 3.4).

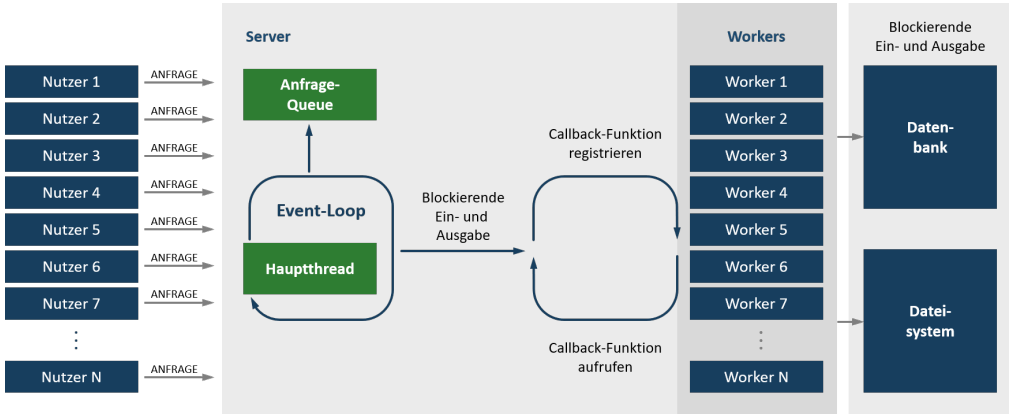


Abbildung 3.4: Die Architektur von Node.js [19]

Die „Package Information Tracker"-Webanwendungs-API kann in Backend-Logic, Worker, Service Broker, Mailer, Image processing und Logging unterteilt werden (siehe Abbildung 3.5).

Diese Backend-REST-API besteht aus 7 API-Controllern und 7 Hauptrouten, die in 39 Unterrouten unterteilt sind. Die Hauptrouten sind für folgende Aufgaben zuständig:

- „auth" dient der Authentifizierung und Autorisierung sowie der Verwaltung der eingeloggten Benutzerdaten. Diese Route wird mit einem sogenannten „rate-limiter" gegen Brute-Force-Angriffe geschützt;
- „users" dient der Benutzerverwaltung und steht nur Administratoren zur Verfügung. Der Administrator kann Benutzerdaten ändern und bestimmte Benutzer sperren und löschen;

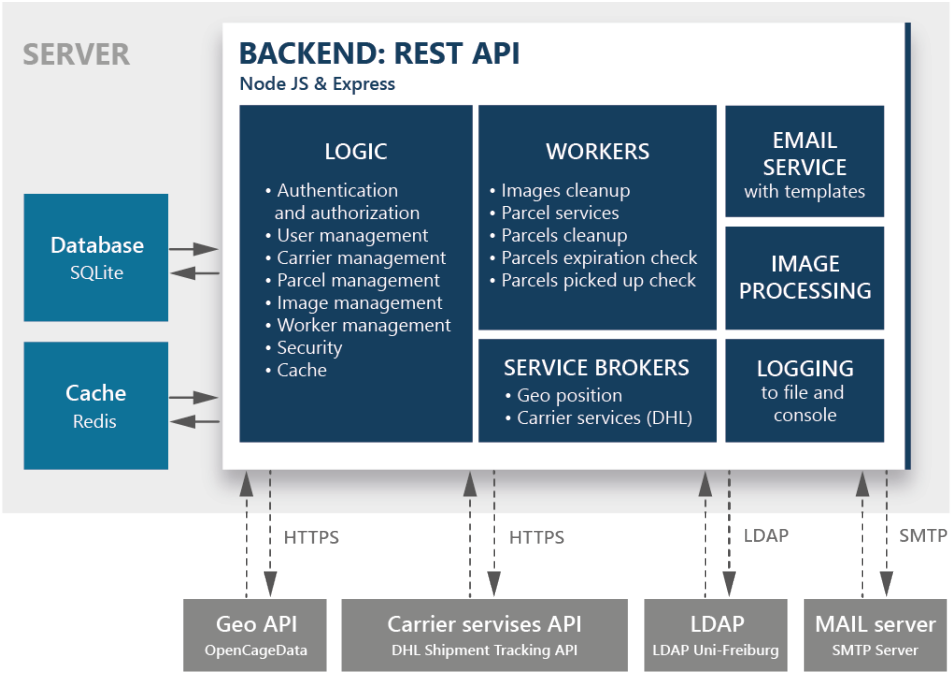


Abbildung 3.5: Schematische Darstellung der API für die „Package Information Tracker"-Webanwendung

- „carriers" wird verwendet, um Transportdienste zu verwalten. Diese Route steht Administratoren und Moderatoren zur Verfügung. Neben der Verwaltung von Transportdiensten wird hier angezeigt, ob ein solcher eine zusätzliche Anbindung an die Transportdienst-API hat und wann die letzte Paketstatus-Aktualisierung stattgefunden hat;
- „parcels" wird zur Verwaltung von Paketen verwendet und ist der wichtigste Teil der Backend-Logic;
- „images" wird verwendet, um Paket-Fotos zu verwalten;
- „workers" hat nur eine Route, ist mit einem „rate-limiter" und einem Schlüssel geschützt und wird hauptsächlich vom internen „Web-Cron-Server" zum Starten von Backend-Workern verwendet;
- „healthz" hat nur eine Route und dient dazu, den Server-Zustand zu beobachten. Es gibt eine JSON-Response mit „success: true", wenn der Server richtig funktioniert und alle für den Serverbetrieb wichtigen Umgebungsvariablen gesetzt

sind.

Die Daten der Routen „parcels“, „carriers“ und „users“ werden mit Hilfe von Redis gehasht und verfügen auch über eine zusätzliche Unterroute namens „cached-time“, die wir bereits in Kapitel 3.1, Abbildung 3.1 betrachtet haben.

Service Brokers bündeln die Dienste von Drittanbietern. Dazu zählen der Dienst „geo-position“, bei dem die OpenCage-API Adressen in Koordinaten umwandelt, sowie „carriers“, die die Anbindung an die Transportdienst-APIs realisieren. Bei Letzterem können die Module hinzugefügt werden, die den Paketstatus verschiedener Anbieter per API abrufen und diese Informationen in die Paketanzeige einfügen. Ein solcher Datenabruf – sofern ein solches Modul für den angegebenen Carrier-Dienst implementiert ist – erfolgt beim Anlegen, Aktualisieren oder zu einem in der Modulkonfiguration eingetragenen Zeitpunkt.

Im Rahmen dieser Arbeit wurde ein DHL-Modul implementiert und die Daten für alle DHL-Paketanzeigen einmal täglich aktualisiert. Außerdem erhält jeder Transportdienst bei der Erstellung einen Tracking-Link-Präfix. Mit dem Link kann der aktuelle Paketstatus abgerufen werden, indem auf die Seite des Spediteur-Trackings umgeleitet wird, auch auf die von Anbietern, deren API nicht mit der „Package Information Tracker“-Webanwendung verbunden ist.

Die folgenden „Workers“ werden verwendet, um die Daten automatisch zu verwalten und zu bereinigen:

- images-cleanup – löscht täglich die Bilder, die zu keiner Paketanzeige gehören;
- parcel-services – aktualisiert alle Paketstatus-Informationen für alle registrierten Paketzustelldienst-APIs;
- parcels-cleanup – entfernt alle Pakete mit dem Status „abgelaufen“ nach der in der Konfiguration eingetragenen Zeit seit dem letzten Update. Standardmäßig beträgt diese Zeit drei Tage;

- parcels-expiration-check – ändert den Status des Pakets nach der in der Konfiguration angegebenen Zeit seit dem letzten Update von „erwartet“ oder „zugestellt“ auf „abgelaufen“. Für Pakete, die keine Eigentümer oder Empfänger gefunden haben, ist diese Zeit standardmäßig auf 30 Tage gesetzt. Für Pakete, bei welchen Eigentümer und Empfänger eingetragen sind, beträgt diese Zeit sieben Tage;
- parcels-picked-up-check – ändert den Status des Pakets einen Tag nach dem letzten Update von „abgeholt“ auf „abgelaufen“.

Ein weiterer wichtiger Teil des Backends ist der Mailer. Dieser wurde mit Hilfe des Node.js-Moduls Nodemailer implementiert. Es wurden 15 E-Mail-Vorlagen erstellt, die mit Nutzerdaten oder Paketdaten ausgefüllt und in verschiedenen Szenarien an Nutzer verschickt werden. Wenn in Kapitel 4 das Interface-Design näher angeschaut wird, werden auch einige E-Mail-Vorlagen betrachtet.

Dank „Image processing“ werden die vom Benutzer hinzugefügten Bilder optimiert und in drei verschiedenen Auflösungen gespeichert, um den Traffic bei der Nutzung der App zu reduzieren und gleichzeitig eine gute Qualität zu gewährleisten.

Zur Fehlerverfolgung und zum Debuggen wurde ein Logging-System entwickelt. Es wird der Modulname „winston“ verwendet und die Logging-Daten in eine Datei und an die Konsole ausgegeben. Die Protokolle werden täglich archiviert und nach 30 Tagen automatisch gelöscht, damit unsere Anwendung DSGVO-konform bleibt und die Protokolldaten nicht unnötig Speicherplatz verbrauchen.

Zum Schutz der Backend- und Benutzerdaten wurden die folgenden funktionalen und nicht funktionalen Sicherheitsbestandteile integriert [19]:

- Verschlüsselung von Passwort und Passwort-Reset-Token in der Datenbank;
- Autorisierung mit JSON Web Token (JWT);
- Protokollierung von Fehlermeldungen und Fehlversuchen;
- Autorisierung und Vergabe von minimalen Berechtigungen;

- Schutz von Authentifizierungs- und Autorisierungs-Routen mithilfe von „rate-limitern“ gegen Brute-Force-Angriffe;
- Eingabe- und Ausgabevalidierung;
- Schutz vor Cross-Site-Scripting;
- Schutz vor SQL Injection;
- Schutz vor HTTP Parameter Pollution.

Der Backend-Projektordner enthält, ähnlich wie das Frontend, einen Konfigurationsordner, in dem viele verschiedene Einstellungen vorgenommen werden können. Die sicherheitsrelevanten und wichtigsten davon werden mit Hilfe von Umgebungsvariablen eingesetzt.

3.4 Integration und Bereitstellung

Wie im vorherigen Unterkapitel beschrieben wurde, handelt es sich bei Node.js um eine Single-Thread-Anwendung. Um von Multicore-Systemen zu profitieren und die Möglichkeit von Ausfällen zu reduzieren, kann die Anwendung horizontal skaliert werden. Dies kann verwirklicht werden, indem mehrere Instanzen gleichzeitig gestartet werden. Für diesen Schritt können interne Node.js-Module oder produktionsreife Module von Drittanbietern wie Phusion Passenger oder PM2 verwendet werden. Im Kontext unserer Anwendung wurde entschieden, PM2 in Einsatz zu nehmen und damit nicht nur das Backend, sondern auch die Frontend-Prozesse für das serverseitige Rendering zu starten. „Web-Cron-Server“ wird ebenfalls vom Prozessmanager eingeleitet, um die Möglichkeit von Fehlern zu reduzieren und die sichere Arbeit der Worker zu gewährleisten.

PM2 ist ein Prozessmanager für Node.js-Anwendungen mit integriertem Load Balancer. Es ermöglicht, eine Anwendung immer am Leben zu erhalten und sie ohne Ausfallzeiten neu zu laden [20]. PM2 verfügt auch über andere Funktionen wie Überwachung und Protokollierung, um Systemverwaltungsaufgaben zu erleichtern.

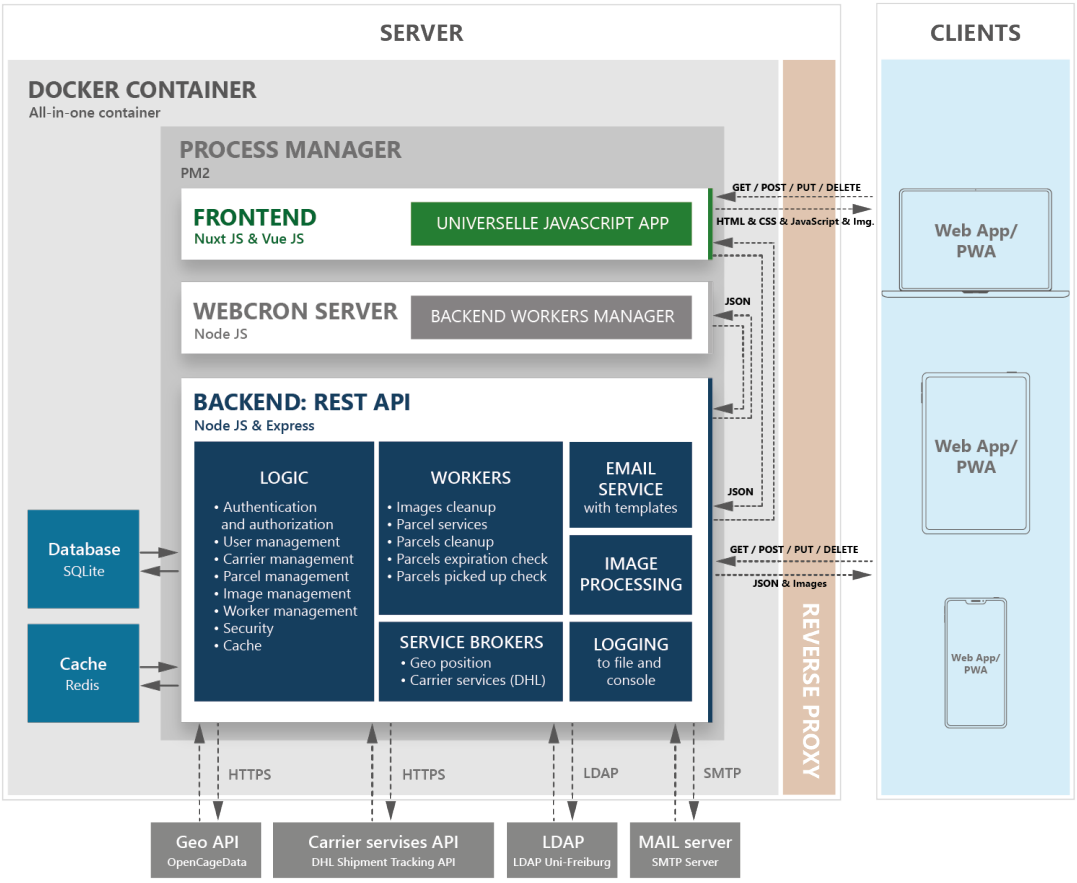


Abbildung 3.6: Die Architektur der „Package Information Tracker“-Webanwendung

Abbildung 3.6 zeigt die gesamte Architektur der Anwendung einschließlich des Prozessmanagers und des Dockers. Ein Reverse-Proxy ist nicht Bestandteil der Anwendung und müsste auf dem Hosting-Server konfiguriert werden. Zudem sollten die Anwendungsdomänen mit SSL-Zertifikaten verschlüsselt werden.

Im Docker-Container wurde ein Makefile angelegt, das die wichtigsten Befehle zum Starten, Seeding der Datenbank, Monitoring und Testen der Anwendung bereitstellt. So werden bei der Eingabe eines „make help“-Befehls alle anderen anwendungsspezifischen Befehle aufgelistet (siehe Abbildung 3.7).

Bei der Entwicklung von Backend und Frontend wurde ein Linter-Tool names ESLint

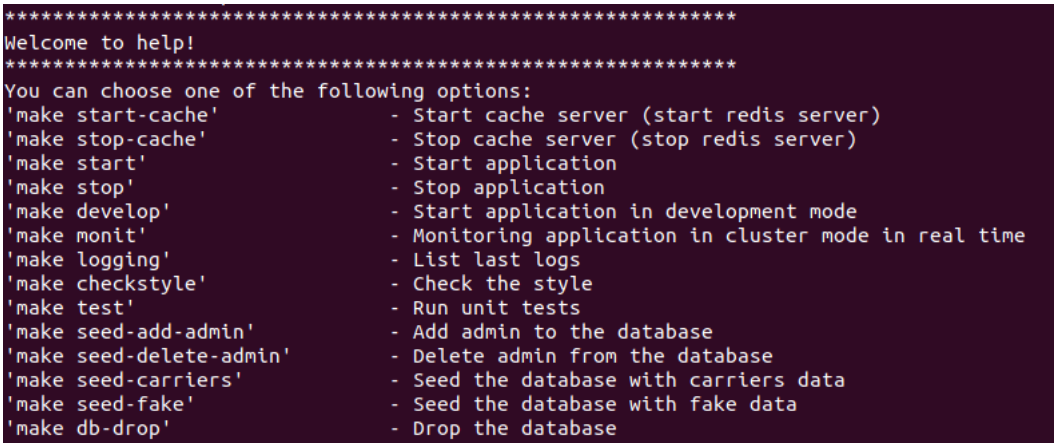


Abbildung 3.7: Anwendungsbefehle im Docker-Container

verwendet, um den Quellcode zu analysieren und problematische Stellen und Muster im Javascript-Code zu identifizieren. Das Backend und Frontend wurden auch mit dem JavaScript-Testframework getestet. So kann der Code durch Eingabe von „make checkstyle“ oder „make test“ jederzeit überprüft bzw. getestet werden. 3

4 User Interface und User Experience Design

4.1 Überblick

Die Benutzeroberfläche der „Package Information Tracker“-Webanwendung wurde auf eine Art gestaltet, die sie für Desktop, Tablet und mobile Geräte geeignet macht (Responsive Webdesign). Die Desktop- und Tablet-Versionen unterscheiden sich nicht wesentlich voneinander, daher beschränken wir uns auf die Übersicht der Benutzeroberfläche auf der Desktop- und der mobilen Version.

Die Anwendung wurde zweisprachig – auf englisch und deutsch – gestaltet, was sich jedoch problemlos erweitern lässt. Beim Aufruf der Anwendung wird die Sprache des Webbrowsers erkannt und – falls es sich nicht um eine der genannten Sprachen handelt – wird die englische Sprache als Standardsprache verwendet.

Wie schon in Kapitel 3.1 erwähnt wurde, sind die Benutzer der Anwendung in vier Rollen unterteilt. Abhängig von der aktuellen Benutzerrolle wird auch die Benutzeroberfläche der Anwendung angepasst. Die Abbildung 4.1 zeigt die wichtigsten Anwendungsansichten in diesem Kontext.

In diesem Kapitel betrachten wir alle Anwendungsansichten mit Ausnahme des Impressums, des Datenschutzes und der API-Dokumentation in Bezug auf Benutzerrollen. Dabei beäugen wir auch die Interaktionen, die verschiedene Benutzer durchführen

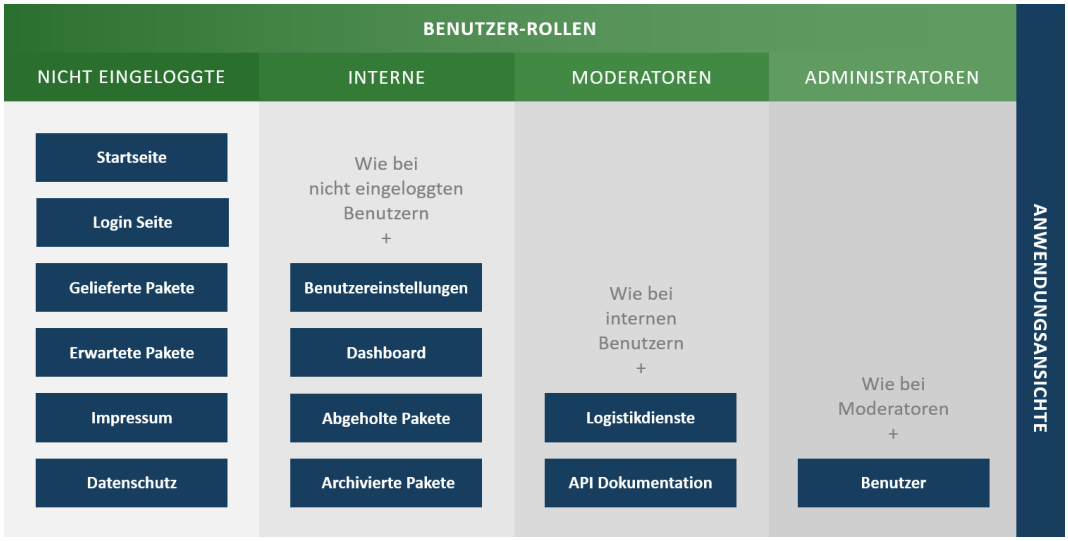


Abbildung 4.1: Verfügbare Anwendungsansichten in Abhängigkeit von der Benutzerrolle

können. Um die Nutzungsszenarien vollständig zu berücksichtigen, sehen wir auch die wichtigsten E-Mails an, die von der REST-API der Anwendung an Benutzer gesendet werden.

4.2 Anwendungslayout, Start- und Login-Seite

Beim Starten der Webanwendung wird der Benutzer auf die Startseite geleitet. Diese Übersicht stellt die Anwendung kurz vor und bietet die Möglichkeit, direkt anzufangen und sich anzumelden oder ein neues Konto zu erstellen. Hat der Benutzer die Seite bereits in letzter Zeit genutzt und ist bereits eingeloggt, wird er beim Drücken des Buttons „Loslegen“ zum Dashboard weitergeleitet.

Darüber hinaus verfügt die Startseite über ein Logo, Anwendungsnavigation, Copyright-Hinweis mit dem Link zur Impressum- und Datenschutzseite sowie Sprachumschaltung (siehe Abbildung 4.2). All diese Schaltflächen sind im Anwendungslayout verankert und begleiten den Benutzer durch die gesamte Anwendung. Erst bei der Nutzung auf

mobilen Endgeräten verschwindet das Logo und die Anwendungsnavigation wird für die mobile Nutzung angepasst (siehe Abbildung 4.3).

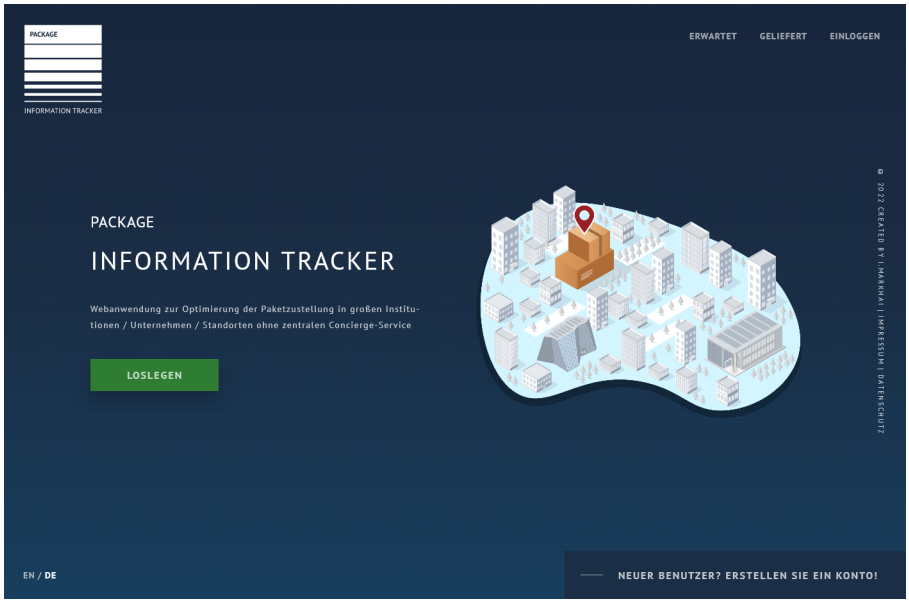


Abbildung 4.2: Startseite der Anwendung

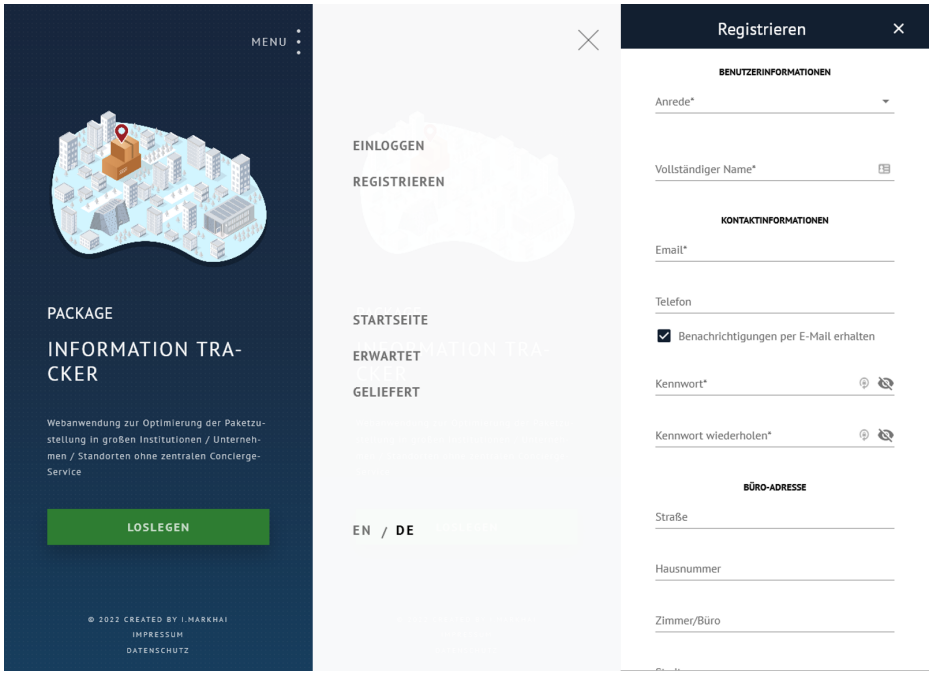


Abbildung 4.3: Mobile Ansicht der Startseite, des Anwendungsmenüs und des modalen Registrierungsdialogs

Durch Anklicken des Buttons „Neuer Benutzer? Erstellen Sie ein Konto!“ öffnet sich ein modaler Registrierungsdialog (siehe Abbildungen 4.3 und 4.4). Hier werden alle notwendigen Benutzerdaten abgefragt, wobei die Pflichtfelder mit einem Stern gekennzeichnet sind. In das Registrierungsformular ist eine Datenvalidierung integriert, sodass der Benutzer eine Benachrichtigung erhält, wenn beispielsweise das Passwort nur aus numerischen Zeichen besteht und keine Groß- oder Kleinbuchstaben enthält.

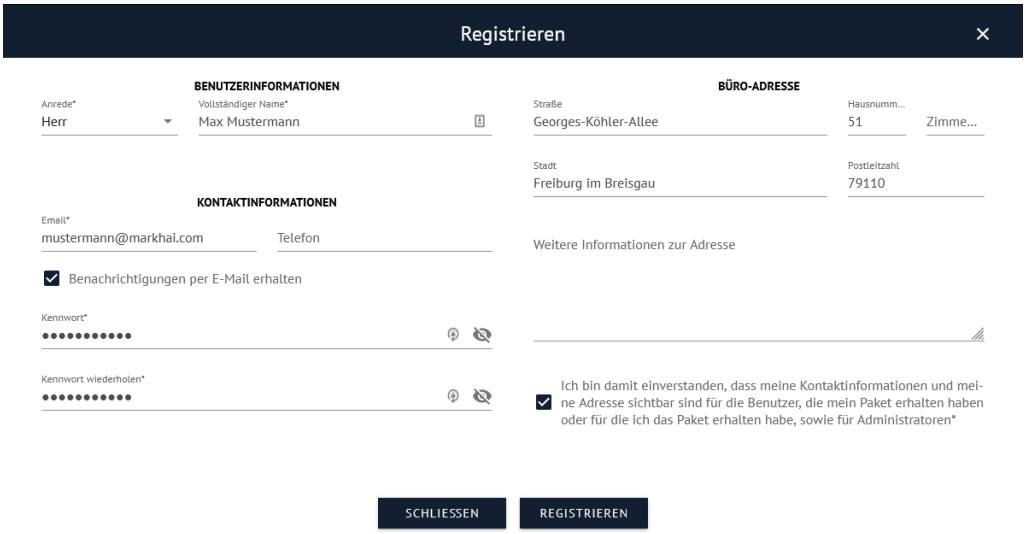


Abbildung 4.4: Benutzer-Registrierungsdialog

Nach erfolgreicher Registrierung erhält der Nutzer eine Willkommens-E-Mail und kann sich sofort in die Anwendung einloggen.

Der Registrierungsprozess kann auch ohne Registrierung durchgeführt werden, indem man ein Uni-Konto besitzt und auf der Login-Seite „Einloggen mit Uni-Konto“ auswählt (siehe Abbildung 4.5). Die Zugangsdaten werden von der LDAP-Schnittstelle der Universität geprüft und, sofern diese korrekt sind, wird der Benutzer eingeloggt. Nach der erstmaligen Nutzung des LDAP-Login-Verfahrens wird der Benutzer auf die Seite mit den Benutzereinstellungen navigiert, wo er die minimal notwendigen Informationen wie Anrede, vollständiger Name sowie E-Mail-Adresse einzugeben hat.

Die Login-Seite bietet auch die Möglichkeit, das Passwort zurückzusetzen. Diese Möglichkeit steht jedoch nur denjenigen Benutzern zur Verfügung, die sich mit ihrer E-Mail-Adresse registriert haben und kein Uni-Konto verwenden. Das Zurücksetzen des Uni-Konto-Passworts muss auf der zuständigen Uni-Website erfolgen.

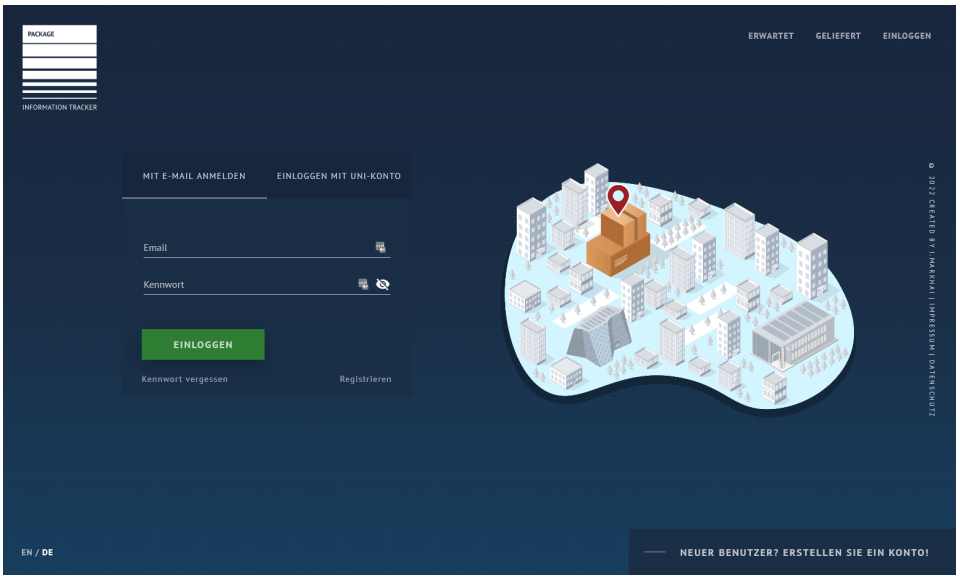


Abbildung 4.5: Login-Seite

4.3 Dashboard und Benutzereinstellungen

Als eingeloggter Benutzer gelangt man nach erfolgreicher Anmeldung auf die Dashboard-Seite (siehe Abbildungen 4.6, 4.7). Diese Seite bietet einen schnelleren Zugriff auf die Pakete, die sich auf den Benutzer beziehen, wie zum Beispiel:

- Pakete, die der Benutzer bestellt, im System als „erwartet“ eingegeben – aber noch nicht bekommen hat;
- Angemeldete (interne) Benutzer;
- Pakete, die der Benutzer bestellt hat, aber von anderen Benutzern angenommen wurden;
- Pakete, die der Benutzer für andere Benutzer erhaltet hat.

Anschließend kann ein neues erwartetes oder geliefertes Paket hinzugefügt werden, indem auf die Plus-Schaltfläche geklickt und dann „Ich erwarte ein Paket“ oder „Ich habe ein Paket angenommen“ ausgewählt wird.

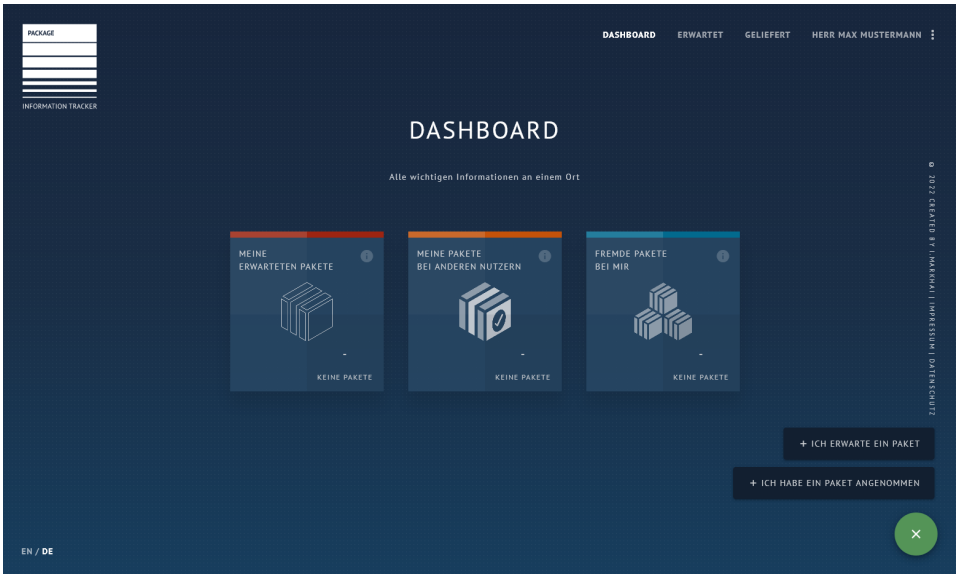


Abbildung 4.6: Dashboard Seite

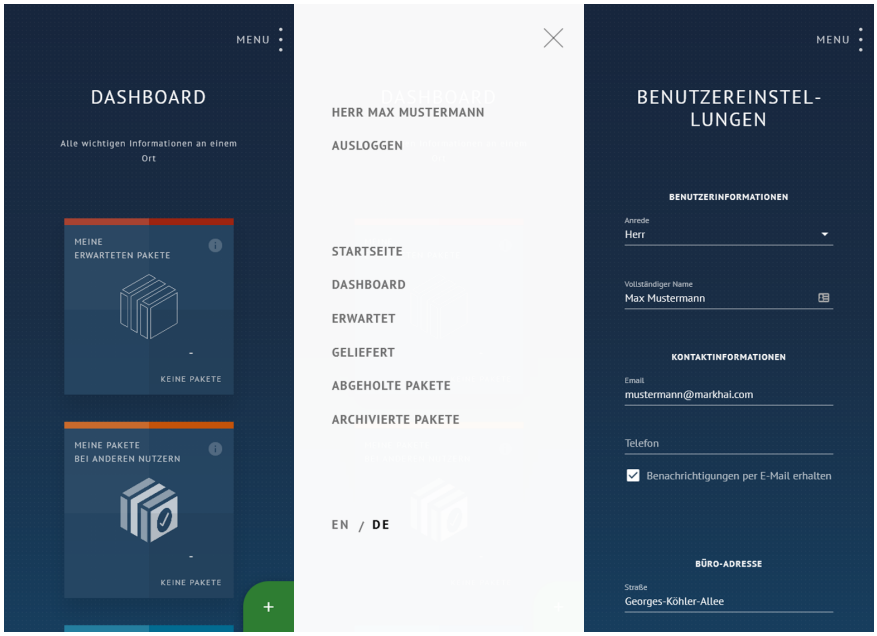


Abbildung 4.7: Mobile Ansicht der Dashboard-Seite, Navigationsuntermenü, Benutzereinstellungen

Beim Klick auf eine Kachel wird der Nutzer auf die Seite Erwartet oder Geliefert weitergeleitet und ein entsprechender Filter – „Von mir erwartet“, „Kann ich abholen“ oder „Habe ich angenommen“ – aktiviert. Es besteht auch die Möglichkeit, alle erwarteten oder gelieferten Pakete anzuzeigen, indem man den Filter „Alle Erwartete“ oder „Alle Gelieferte“ anschaltet oder einfach auf der Dashboard-Seite in der Anwendungsnavigation oben „Erwartet“ oder „Geliefert“ auswählt.

Nach erfolgreicher Anmeldung ändert sich die Navigationsanwendung dahingehend, dass ein zusätzlicher Link zum Dashboard erscheint und der Login-Link sich zu einem Untermenü mit dem Benutzernamen verändert. Wenn man dieses Untermenü anklickt, hat man die Möglichkeit, zu den Benutzereinstellungen zu navigieren, das eigene Passwort zu ändern und zu den abgeholten und archivierten Paketen weitergeleitet zu werden. Auch kann sich jeder Benutzer über das Navigationsmenü abmelden. Moderatoren und Administratoren können auch zur Seite mit den Logistikdiensten und zur API-Dokumentation navigieren. Letztere erhalten noch einen zusätzlichen Zugriff, um die Benutzer zu verwalten (siehe Abbildungen 4.7, 4.8).

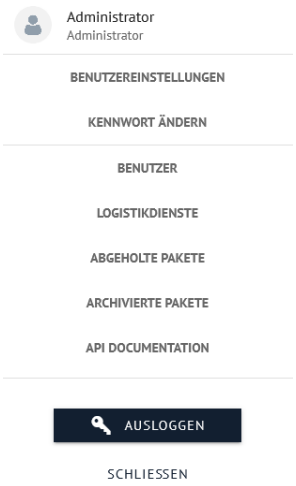


Abbildung 4.8: Navigationsuntermenü – Administratorenansicht

Die Seite Benutzereinstellungen (siehe Abbildungen 4.9) bietet die Möglichkeit, die während des Registrierungsprozesses eingegebenen Daten zu ändern. Hier kann man

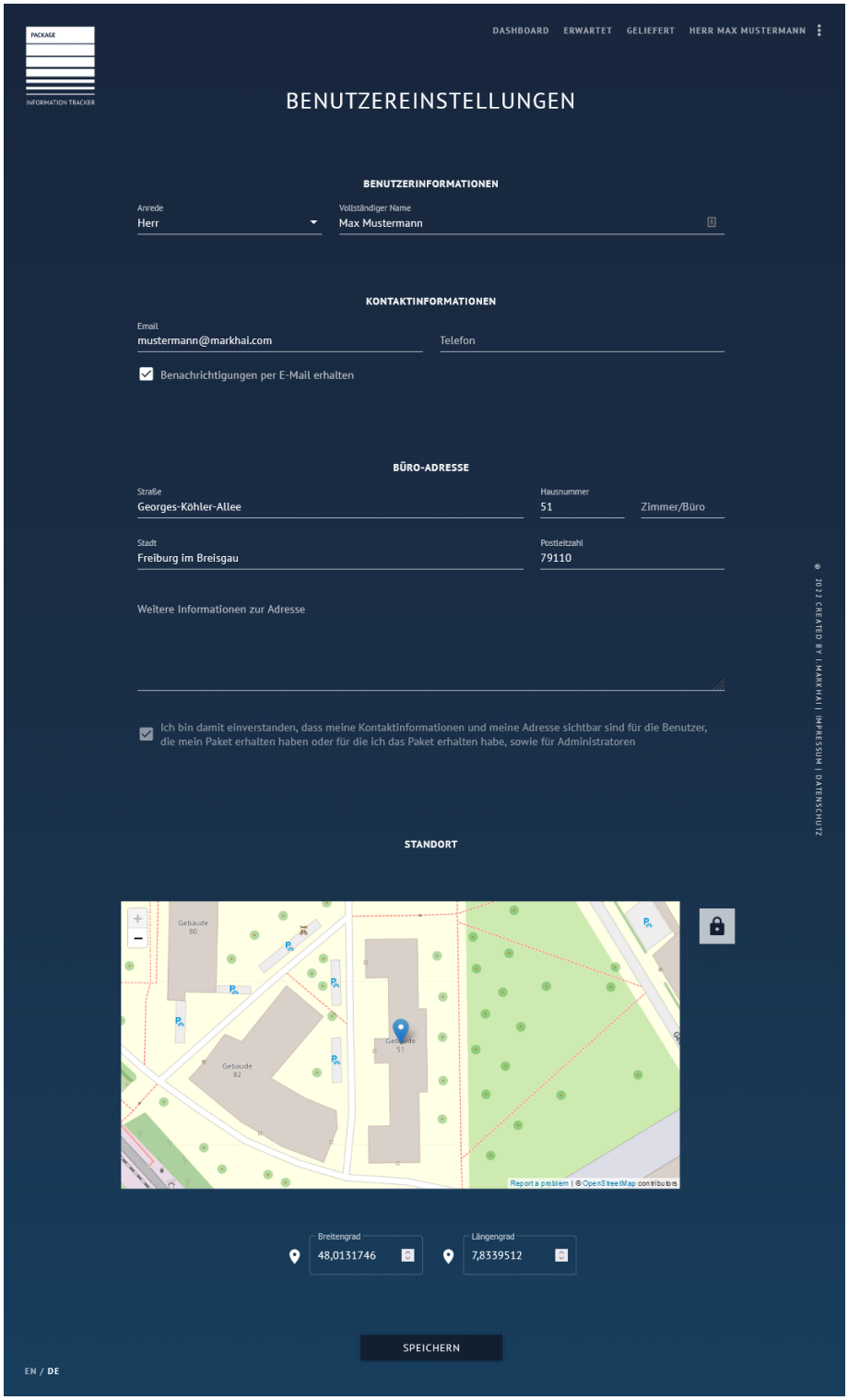


Abbildung 4.9: Benutzereinstellungen-Seite

sich von E-Mail-Benachrichtigungen abmelden oder die Geo-Koordinaten, die automatisch aus der Benutzeradresse berechnet wurden, anpassen. Dank einer Standortkarte werden die Geo-Koordinaten als Marker auf einer OpenStreetMap angezeigt. Alle Daten aus Benutzereinstellungen sowie die Karte können von anderen Benutzern eingesehen werden, wenn man sich eingeloggt und als Paketeigentümer oder -empfänger in den vom Benutzer angegebenen Anzeigen gemeldet hat. Diese Daten sind auch Moderatoren und Administratoren ersichtlich.

4.4 Die Seiten „Erwartet“, „Geliefert“, „Abgeholt“ und „Archivierte Pakete“

Die Paket-Seiten „Erwartet“, „Geliefert“, „Abgeholt“ und „Archivierte Pakete“ haben eine ähnliche Struktur und Funktionalität. Oben wird ein Suchfeld angezeigt, in dem man alle Anzeigen durchsuchen kann (siehe Abbildung 4.10, 4.11). Es wird nach allen Merkmalen gesucht, die explizit vom Anzeigen-Herausgeber ausgefüllt wurden sowie nach der Anzeigen-ID. Hat ein Herausgeber im Feld „Name/Paketbezeichnung“ seinen Namen eingetragen, wird dieser ebenfalls gefunden. Standardmäßig sind die Benutzerdaten aus Datenschutzgründen nicht ersichtlich.

Nach dem Suchfeld folgt eine Auswahl an Filtern, die sich je nach Paket-Seite unterscheiden. Beispielsweise werden „Alle Erwartete“ und „Von mir erwartet“ bei „Erwartet“ angezeigt (siehe Abbildung 4.10). Die Seite „Geliefert“ hingegen enthält „Alle Gelieferte“, „Kann ich abholen“ und „Habe ich angenommen“ (siehe Abbildung 4.11). Auf der Seite „Abgeholt“ wird der Filter „Alle Abgeholte“, „Von mir abgeholt“ und „Vom Eigentümer abgeholt“ angezeigt. Die Seite „Archivierte Pakete“ ist ein Ort, in den die Paketanzeigen verschoben werden, bevor sie gelöscht werden, und hat keine Filterfunktion. Wenn man auf „Sortierung und Filterung“ klickt, werden zusätzliche Sortierungs- und Filtermöglichkeiten angezeigt, sofern diese für die entsprechende Seite vorhanden sind.

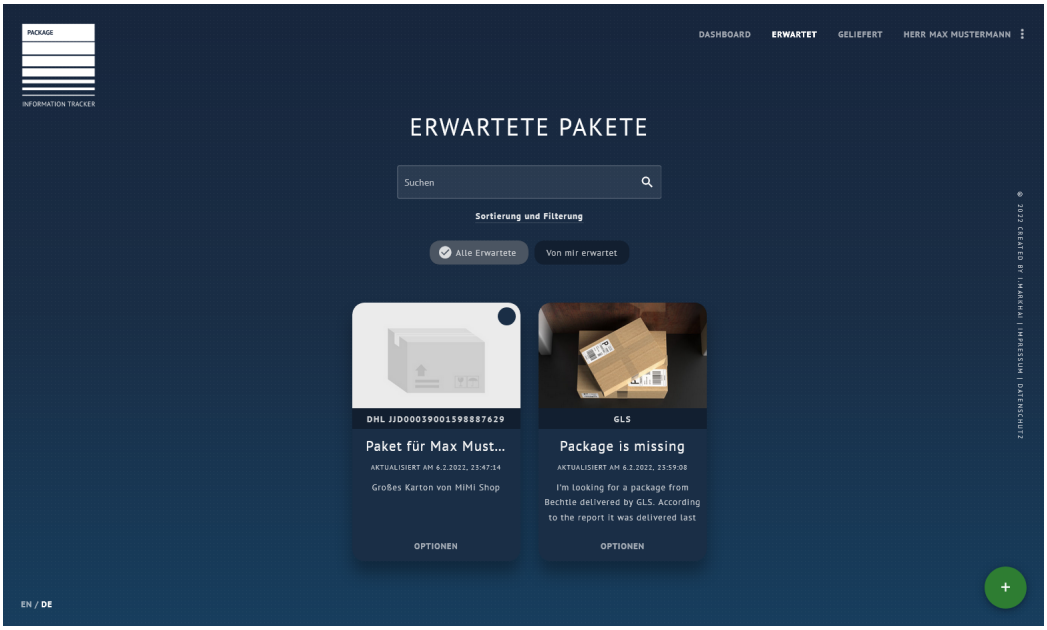


Abbildung 4.10: Erwartete Pakete Seite

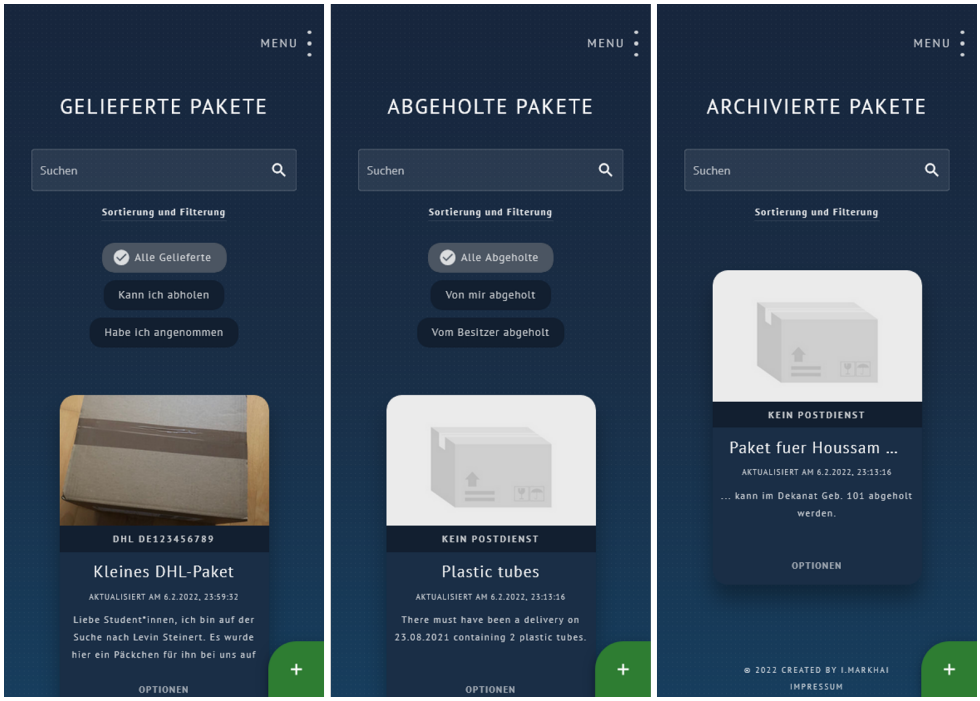


Abbildung 4.11: Mobile Ansicht von „Geliefert“, „Abgeholt“ und „Archivierte Pakete“

Die Paketanzeigen werden in Form von Kacheln dargestellt. Dieser Formfaktor ist sehr informativ und sowohl für den Desktop als auch für alle mobilen Geräte geeignet. Die Kachel enthält eine Markierung, wenn sich eine Anzeige auf den Nutzer bezieht sowie alle vom Anzeige-Herausgeber eingegebenen Informationen und – sofern vorhanden – ein Foto. Die genannte Markierung ist ein blauer Punkt, welcher sich beim Darüber-Schweben mit dem Cursor bspw. zu „von mir erwartet“ erweitert. Wenn eine Anzeige kein Foto enthält, wird ein Platzhalterbild angezeigt. Wenn man unten auf „Optionen“ klickt, dreht sich die Kachel um und die dem Benutzer zur Verfügung stehenden Optionen werden angezeigt (siehe Abbildung 4.12). Diese können sich hinsichtlich der Benutzerrollen unterscheiden. Die nicht eingeloggten Nutzer können lediglich die Anzeige öffnen oder den Link kopieren. Der Herausgeber bzw. ein Moderator oder Administrator kann dagegen die Anzeige noch bearbeiten, löschen und das Paket als abgeholt markieren. Andere registrierte Nutzer können sich als Eigentümer oder Empfänger festlegen, indem sie auf „Habe ich angenommen!“ oder „Von mir erwartet!“ klicken; das Kopieren des Links ist ihnen ebenso freigestellt.

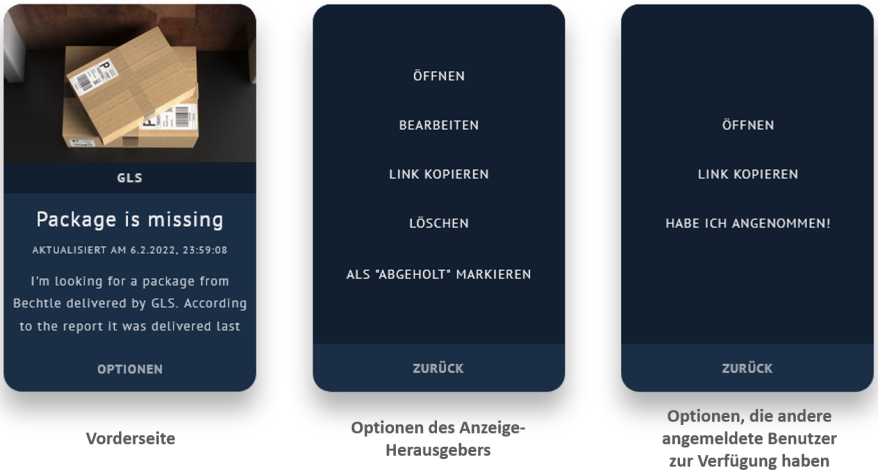


Abbildung 4.12: Kachel für Anzeige eines erwarteten Pakets

Beim Öffnen der Anzeige öffnet sich ein modaler Dialog mit Paketinformationen (siehe Abbildung 4.13). Der Herausgeber kann diese Daten auch bearbeiten und vervollständigen und bei Bedarf ein Foto hinzufügen.

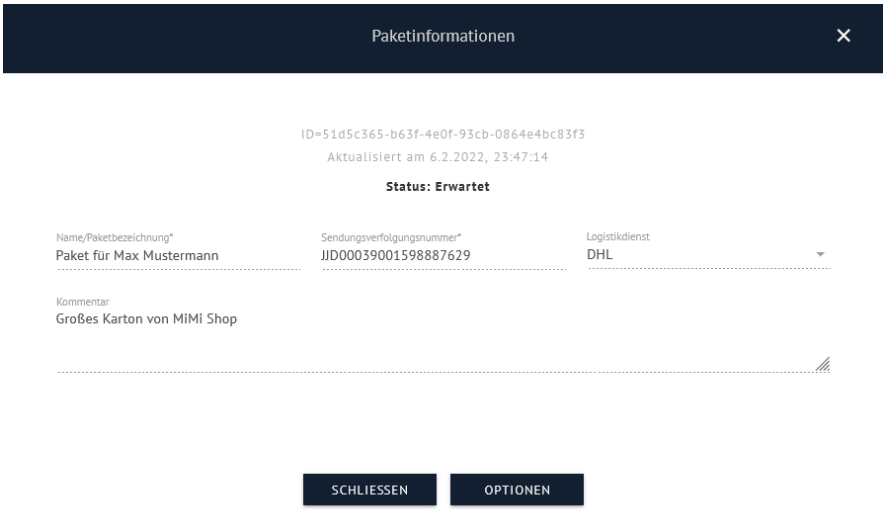


Abbildung 4.13: Modaler Dialog für Anzeige eines erwarteten Pakets

Wählt ein angemeldeter Benutzer aus den Optionen „Habe ich angenommen“ oder „Von mir erwartet“, wird ein modaler Dialog angezeigt (siehe Abbildung 4.14). In diesem Fenster kann der Paketempfänger oder -eigentümer eine Nachricht an den Anzeige-Herausgeber verfassen, in der er beispielsweise angibt, wann das Paket abgeholt werden kann, sofern er ein Paketempfänger ist – oder wann er das Paket abholen kann, wenn er ein -eigentümer ist. Ein Empfänger kann im Zweifelsfall auch ein Foto hinzufügen.

Nach dem Speichern wird der Dialog der Paketanzeige wieder geöffnet (siehe Abbildung 4.15). Nun werden die Daten des Anzeige-Herausgebers dem Paketempfänger/-eigentümer sichtbar und er kann ihn per E-Mail oder Telefon (sofern vorhanden) kontaktieren. Um unnötige Fragen zu umgehen, werden ihm die Adresse und dazugehörige Position auf der Karte angezeigt. Gleichzeitig wird der Anzeige-Herausgeber per E-Mail informiert (siehe Abbildung 4.16). Handelt es sich um ein erwartetes Paket, wird dieses automatisch in „Geliefert“ verschoben. Sofern die Informationen des vom Anzeige-Herausgeber erwarteten oder gelieferten Pakets mit denen des anderen Benutzers nicht übereinstimmen, kann er bei der Bearbeitung des Pakets den Paketempfänger bzw. -eigentümer löschen.

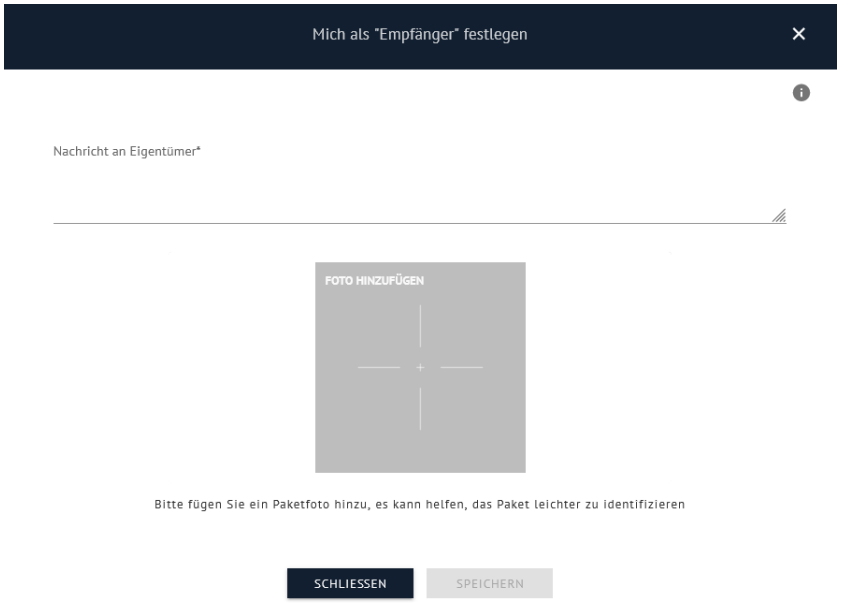


Abbildung 4.14: Modaler Dialog für „Mich als Empfänger festlegen“

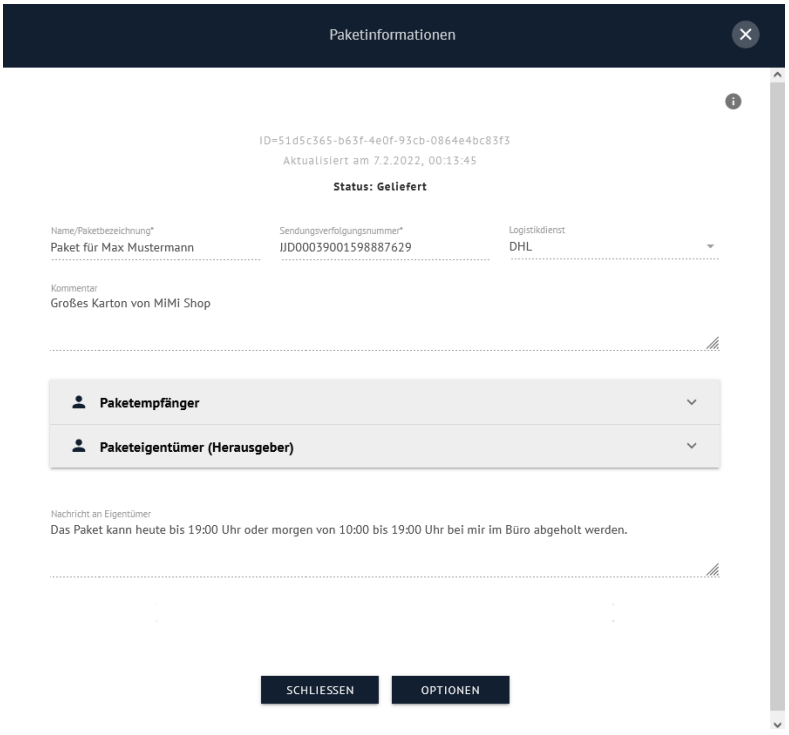


Abbildung 4.15: Paketinformationen, die für Paketempfänger und Paketeigentümer sichtbar sind

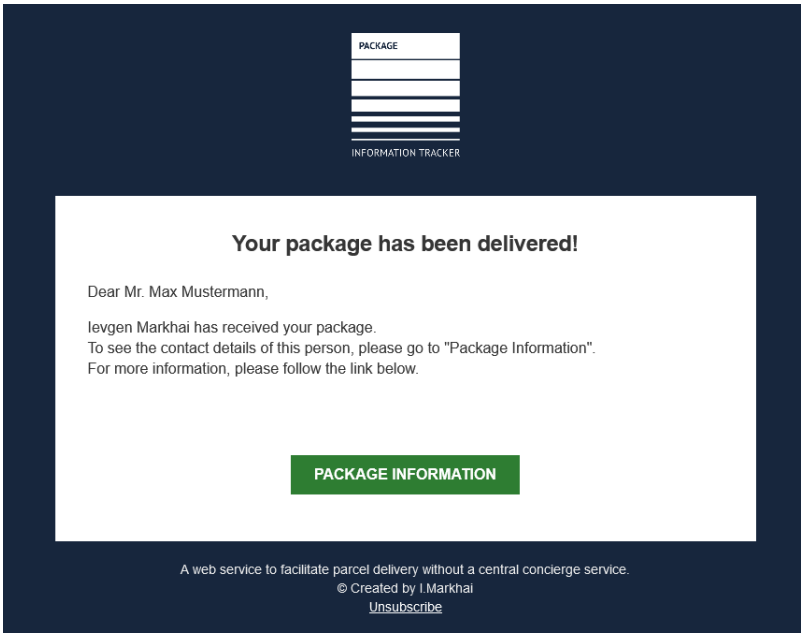


Abbildung 4.16: E-Mail an den Paketeigentümer, dass ein Empfänger gefunden wurde

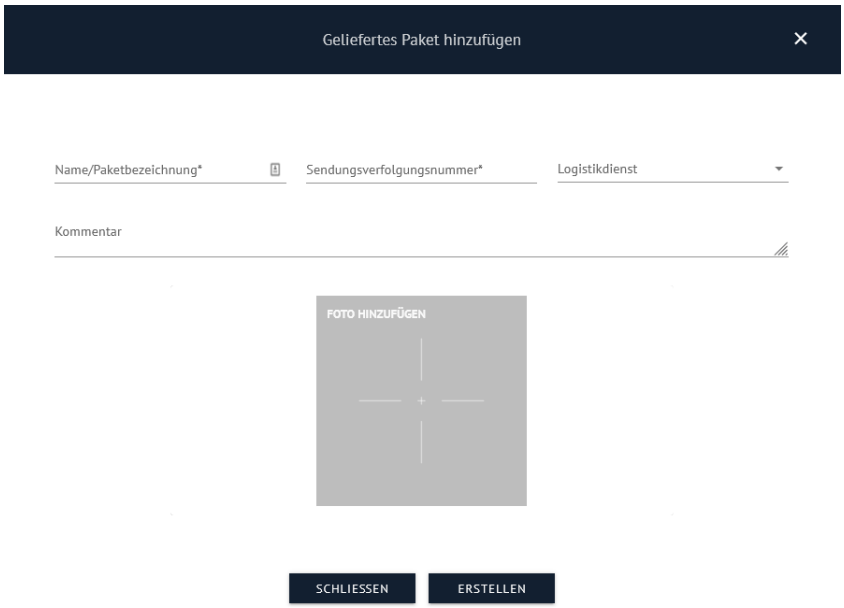


Abbildung 4.17: Erstellung einer neuen Anzeige

Wenn der Eigentümer oder Empfänger eine Anzeige als abgeholt markiert, wird das Paket zu „Abgeholt“ verschoben und nach einer bestimmten Zeit (standardmäßig

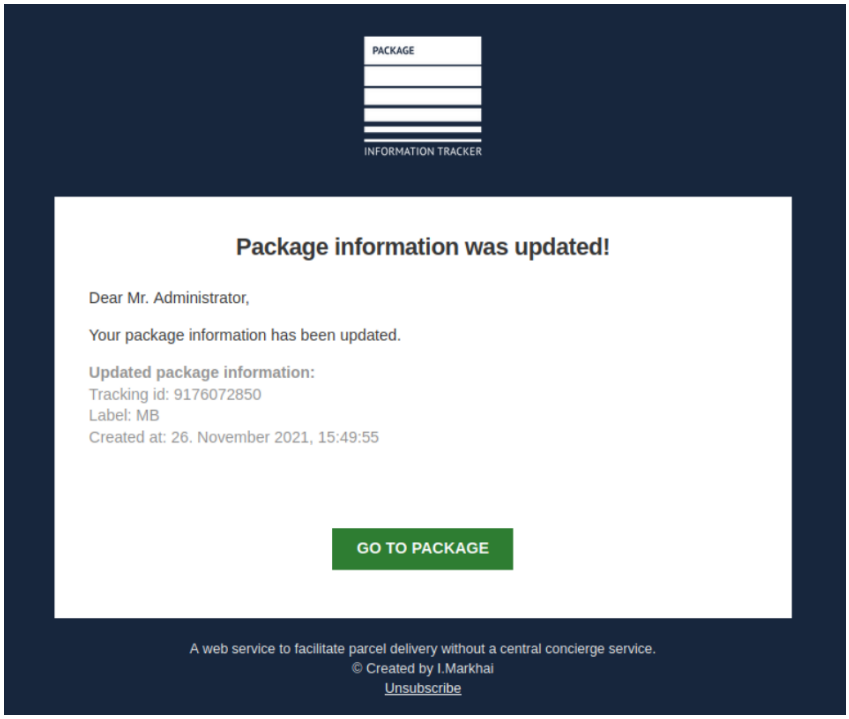


Abbildung 4.18: E-Mail-Benachrichtigung über die Aktualisierung der Anzeigeinformationen

drei Tage) automatisch in die archivierten Pakete geleitet. Nachfolgend wird die Paketanzeige dann standardmäßig nach vierzehn Tagen aus den archivierten Paketen automatisch gelöscht.

Anschließend kann jeder angemeldete Benutzer durch Klicken auf die Schaltfläche „+“ ein erwartetes oder geliefertes Paket hinzufügen. Der Anzeige-Herausgeber muss mindestens einen „Namen/Paketbeschreibung“ oder eine Sendungsnummer eingeben. Zusätzlich können ein Logistikdienst, ein Kommentar und ein Foto, falls es sich um ein bereits geliefertes Paket handelt, eingetragen werden (siehe Abbildung 4.17).

Ändern sich die Anzeigeinformationen, zum Beispiel bei der täglichen Statusprüfung des Logistikdienstes, erhält ein Anzeige-Herausgeber eine E-Mail-Benachrichtigung (siehe Abbildung 4.18) über diese Änderungen und kann mit einem Klick auf „Go to package“ zur Anzeige weitergeleitet werden.

4.5 Die Seite „Logistikdienste“

Auf der Seite der Logistikdienste können Moderatoren und Administratoren Logistikdienste hinzufügen, bearbeiten und löschen. Für diejenigen, die eine Verbindung zur Carrier-Dienst-API haben, wird der Status „Sync ist activ“ und Datum und Uhrzeit der letzten Synchronisation angezeigt (siehe Abbildung 4.19).

Beim Erstellen von neuen Logistikdiensten kann man den Namen, Informationen sowie einen Tracking-Link-Präfix eingeben. Letzteres ist ein Link, mit dem man die Informationen zu einem Paket auf der Logistikdienstleistungs-Seite einsehen kann (siehe Abbildung 4.20).

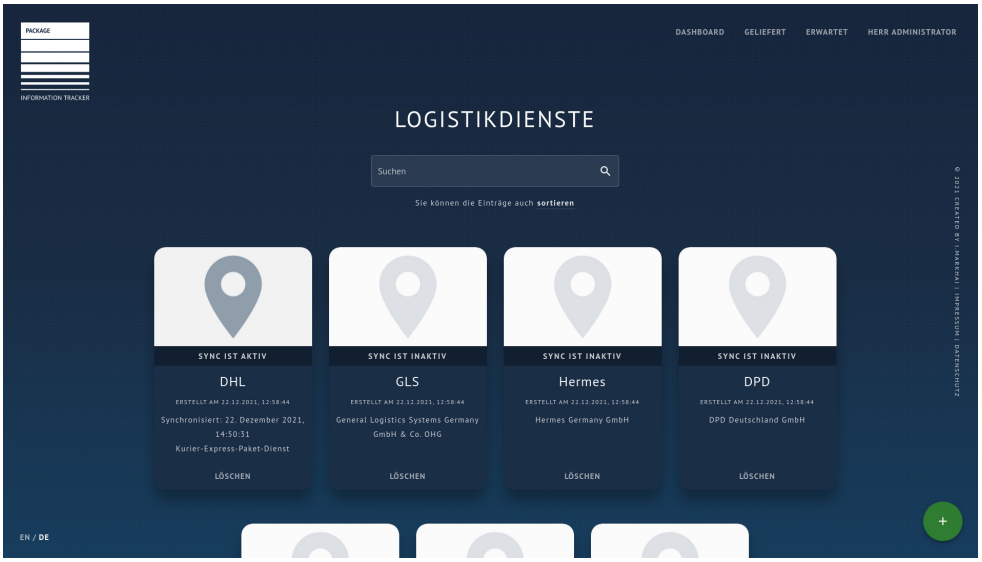


Abbildung 4.19: Logistikdienste-Seite

4.6 Die Seite „Benutzer“

Die einzigen Zugriffsberechtigten auf der Seite „Benutzer“ sind Administratoren. Hier sind alle Benutzer aufgelistet (siehe Abbildung 4.21). Die Administratoren können nach bestimmten Personen suchen, deren Daten bearbeiten (siehe Abbildung

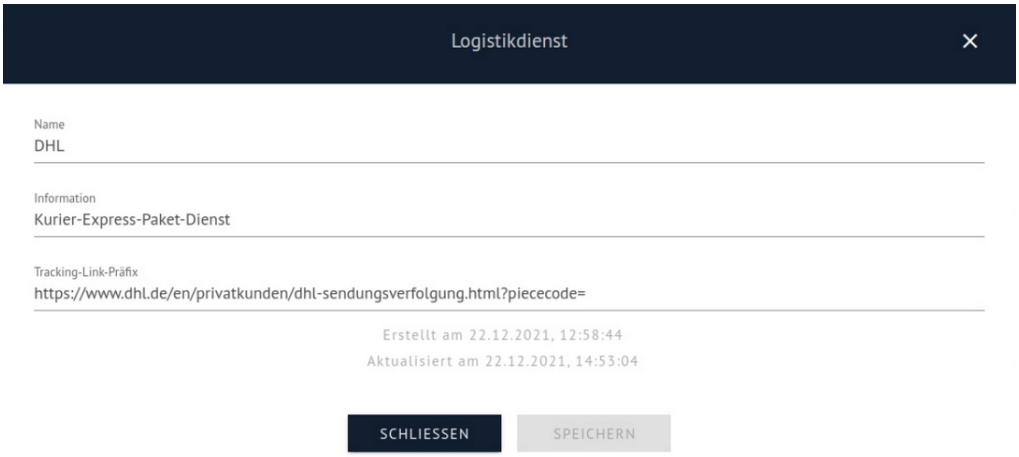


Abbildung 4.20: Bearbeitung von Logistikdienst-Informationen

4.22), ihre Konten deaktivieren, löschen und anlegen. Dabei ist es mithilfe einer jeweiligen Bezeichnung auch leicht, zwischen aktiven und inaktiven Benutzern zu unterscheiden.

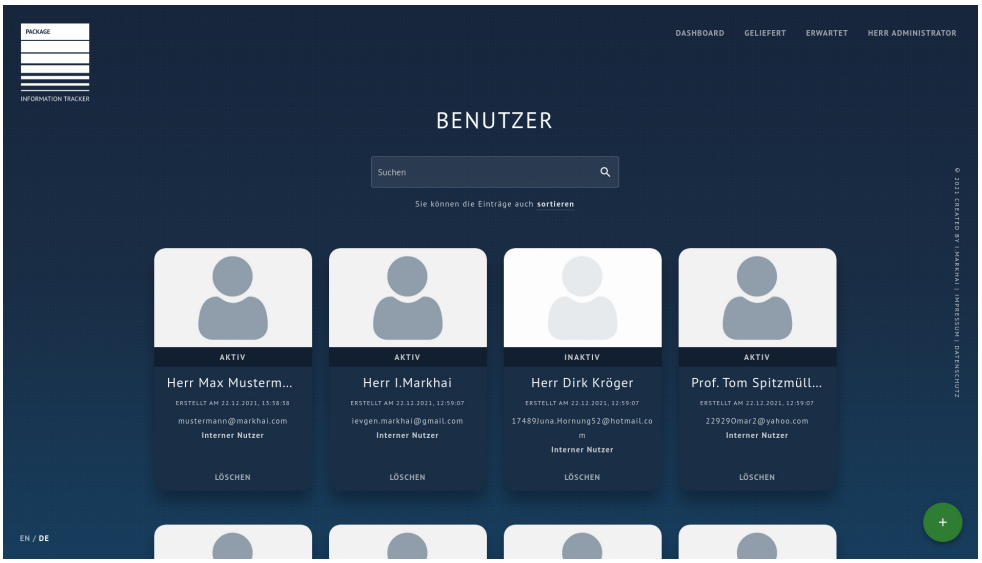


Abbildung 4.21: Benutzer-Seite

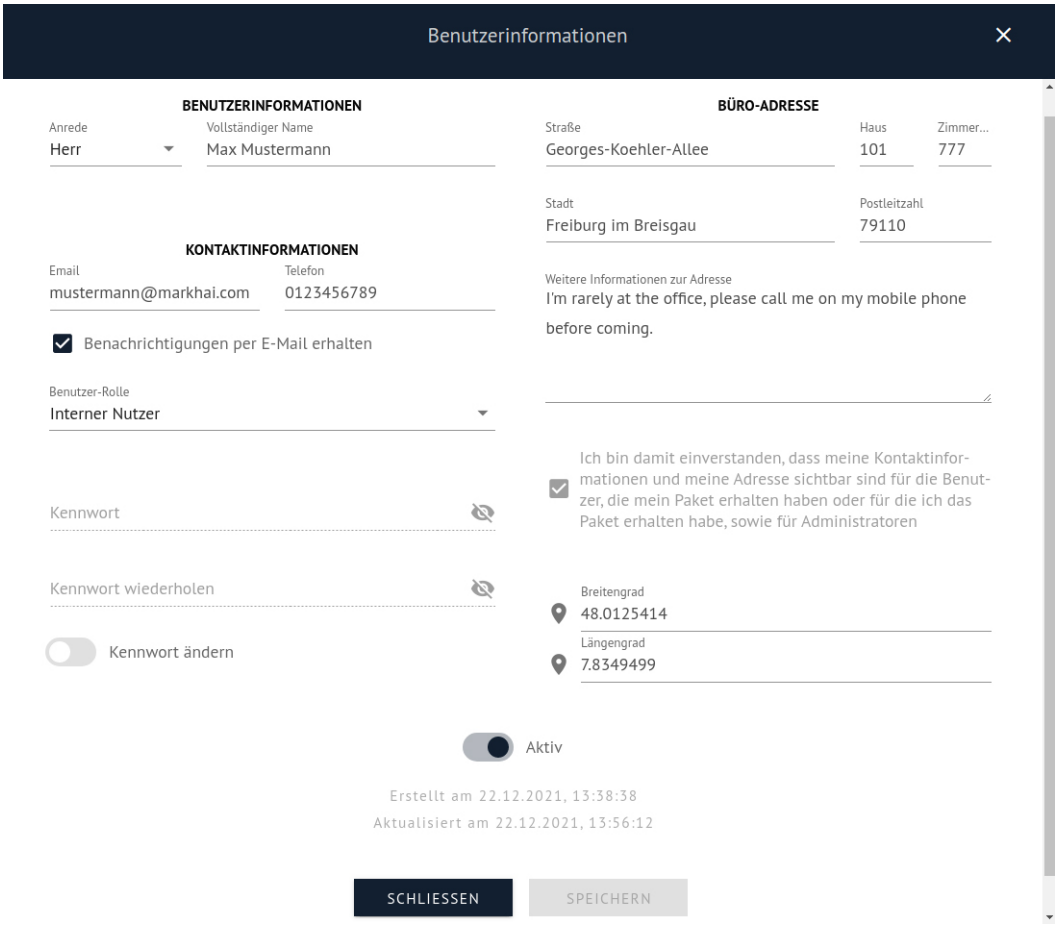


Abbildung 4.22: Benutzerdaten – Administrator Ansicht

5 Evaluation

5.1 Empirische Analyse und Bewertung

Laut Craig Tomlin gibt es vier grundlegende Schritte zur Optimierung einer Webseite oder App, und zwar [21]:

1. Definition von Personen;
2. Analyse von Daten über Benutzerverhalten;
3. Durchführung von UX und Usability Tests;
4. Analyse von Ergebnissen und Erarbeitung von Optimierungsvorschlägen.

Sein Ansatz besteht also darin, quantitative Daten (wie z.B. Google Analytics) mit qualitativen Daten (User Tests) zu kombinieren. Da die „Package Information Tracker"-Webanwendung sich noch im Entwicklungs- und Teststadium befindet, muss leider auf den zweiten Punkt verzichtet werden, weil noch kein richtiges behavioral UX Data vorliegt. Umso mehr gewinnen die zwei anderen Schritte an Bedeutung: Entwicklung von Personas und Benutzer Tests.

Unter Persona versteht man einen generalisierten Benutzer aus einer Zielgruppe mit bestimmten Merkmalen und Bedürfnissen, die mittels der App zu befriedigt versucht werden [21]. Als Zielgruppe können im Prinzip Mitarbeiter einer jeglichen Organisation ohne zentrale Poststelle betrachtet werden, zur Klarheit und Veranschaulichung wird Beispiel hypothetisches Personal der Uni Freiburg genommen. Es werden hierfür zwei möglichst gegensätzliche und dadurch komplementäre Personas entworfen.

Persona 1: Thomas, 40 Jahre alt, männlich, wissenschaftlicher Mitarbeiter. An seine Büroadresse wird häufig wissenschaftliche Literatur bestellt, aber auch einige technische Hilfsmittel sowohl für dienstliche als auch für private Nutzung. Da sich Thomas aber nicht immer an seinem Arbeitsplatz befindet, wird oftmals die Paketannahme erschwert, sodass seine Pakete von anderen Kollegen oder Sekretären angenommen werden müssen. Auf dem Lieferschein steht aber häufig nur „an Nachbarn abgegeben“, ohne diese Person namentlich kenntlich zu machen. Thomas braucht daher ein Werkzeug, um die Paketempfänger ausfindig zu machen, ohne ständig Rundmails schreiben zu müssen. Er besitzt ein überdurchschnittliches technisches Wissen und ist mit unterschiedlichen Webanwendungen gut vertraut. Von einer App erwartet er an erster Stelle eine logische Struktur; dank seiner Erfahrung werden in seinem Fall nur wenige zusätzliche Hinweise oder Einführungen gebraucht. In der Regel liest er alle Benachrichtigungen aufmerksam durch und will die volle Funktionalität jeder App nutzen. Was seine technische Ausstattung angeht, kommen sowohl PC als auch Laptop und Smartphone zum Einsatz.

Persona 2: Agnes, 52 Jahre alt, weiblich, Lehrstuhlsekretärin. Agnes ist für Postverteilung zuständig und nimmt häufig Pakete für andere Mitarbeiter an, ab und zu auch für Kollegen von anderen Lehrstühlen, sodass ihr die Personen oftmals nicht bekannt sind. Manche Pakete liegen über längere Zeiten in ihrem Büro, weil die Paketeigentümer sich bei ihr nicht melden. Agnes wünscht sich eine einfache und intuitiv bedienbare Anwendung, mithilfe derer sie sich schnell mit Paketeigentümern in Verbindung setzen und einen Abholtermin vereinbaren kann. Agnes wird dafür wahrscheinlich ihren Büro-PC nutzen, mit dem sie schon gut vertraut ist. Ihr technisches Wissen ist eher unterdurchschnittlich, weshalb sie etwas mehr Zeit braucht, um sich die Verwendung einer neuen Anwendung einzuverleiben – daher sind für sie Anwendungshinweise besonders hilfreich. Ihr ist es oft nicht bekannt, wo genau sie nach bestimmten Optionen/Einstellungen zu suchen hat, weswegen alle relevanten Schaltflächen sofort sichtbar und als solche erkennbar sein müssen.

Da die App nur einer Post-Launch Testung unterzogen wird, bei welcher keine radikalen Änderungsvorschläge mehr umsetzbar sind, sollen die zwei Personas vor allem dazu dienen, eine optimale Nutzeroberfläche zu entwickeln, die den zwei gegensätzlichen Anforderungen gerecht wird.

5.2 Testdurchführung und Auswertung

Das Ziel dieser User Tests ist herauszufinden, ob die App für ihre Zielgruppe einfach zu benutzen ist und ggf. Anhaltspunkte für ihre Weiterentwicklung zu bestimmen.

Hauptfragen, die sich an die User Tests stellen, sind folgende:

- Ist die App benutzerfreundlich?
- Ist der Ablauf selbstverständlich und die dahinterstehende Logik nachvollziehbar?
- Erfüllt die App ihren Zweck (konnten die Benutzer ihre Aufgaben erfolgreich und vollständig erledigen)?
- Was kann vom Ablauf der Testaufgaben gelernt werden? Was kann an der App noch verbessert werden?

Es wurde sich für einen moderierten Testablauf entschieden, um den Testpersonen, wenn notwendig, mit den Aufgaben weiterhelfen zu können, damit das Testszenario unbedingt komplett durchgespielt wird. Insgesamt wurden fünf Testpersonen gefunden, die für die hypothetische Zielgruppe (Lehrstuhlmitarbeiter) repräsentativ sind und so der Benutzererfahrung potentieller echter App-Nutzer auch nahe kommen. Die Tests haben zwischen 65 und 102 Minuten gedauert und wurden mit Einwilligung jeder Testperson als Video mit Ton aufgezeichnet und gespeichert. Da eine detaillierte Beschreibung jedes einzelnen Testablaufs den Rahmen dieser Arbeit sprengen würde, werden im Folgenden die Eckdaten und eine analytische Zusammenfassung der Testergebnisse präsentiert.

- Testperson 1: Männlich, 28 Jahre alt, Designer (Mac-PC);
- Testperson 2: Männlich, 33 Jahre alt, Software Engineer (Ubuntu-PC);
- Testperson 3: Weiblich, 47 Jahre alt, Büroangestellte (Windows-PC);
- Testperson 4: Weiblich, 25 Jahre alt, Studentin (Android-Smartphone);
- Testperson 5: Männlich, 39 Jahre alt, Projektmanager (iOS-Smartphone).

Vor jedem Test wurde die Idee der App kurz erklärt und die Testpersonen gebeten, während der Aufgaben laut zu denken und Kritik jeder Art frei zu äußern.

Testaufgaben (in Kurzform):

- Registrierung;
- Änderung der Büroadresse;
- Zurücksetzen des Passworts;
- Das Eintragen zweier fremder Pakete, die für „Nachbarn“ angenommen wurden, wobei ein Paket bereits in der Rubrik der erwarteten Pakete existiert (der Testperson ist letztere Tatsache nicht bekannt);
- Das Vereinbaren eines Abholtermins mit den soeben genannten „Nachbarn“;
- Das Eintragen eines eigenen Pakets, das laut Szenario an einen unbekannten „Nachbarn“ abgegeben wurde;
- Das Festsetzen eines Abholtermins für ein eigenes Paket.

Der Testleiter hat zugleich die gegenseitigen Rollen (Paketbesiter oder -empfänger) gespielt. Nach jeder Aufgabe wurden zusätzlich offene Fragen gestellt, wie z.B. *Was war einfach und was fiel eher schwer oder zu umständlich? / Würden Sie bestimmte Buttons mehr hervorheben oder anders platzieren, damit sie sichtbarer werden? Welche Infos oder Optionen fehlen Ihnen noch an der Stelle?*

Nachdem die fünf Tests beendet waren, wurden die Aufnahmen erneut inspiziert und auf übereinstimmende Merkmale durchsucht. Zusammenfassend lässt sich folgendes benennen:

1. Probleme/Schwierigkeiten/Anmerkungen:

- Button „Neuer Benutzer? Erstellen Sie ein Konto!“ wurde auf Desktopversion übersehen;
- In den Benutzereinstellungen: Wenn die Hausnummer bei der Straße fehlerhaft eingegeben wurde, wird es auf der Karte nicht korrekt angezeigt;
- Bei der Änderung der Benutzerdaten wird oft das Speichern vergessen;
- Beim Paket-Hinzufügen stellte sich die Funktion des Felds „Kommentar“ als unklar heraus;
- Die Benachrichtigung, alle gelieferten Pakete einzusehen, fällt bei der Erstellung eines Eintrags in „Erwartet“ nicht auf und wird übersehen (in Mobilversion aber gut sichtbar);
- Die Benachrichtigung, dass das Paket möglicherweise bereits existiere, verschwindet zu schnell;
- Die Funktion, ein Paket als abgeholt zu markieren ist schwer auffindbar;
- Es fehlt ein „Call-to-Action“ bezüglich der Kontaktaufnahme, wenn ein Anzeige-Herausgeber die E-Mail-Benachrichtigung erhält, dass der Paketeigentümer/-empfänger gefunden wurde;
- E-Mails sind nur auf Englisch.
- Die „+“-Schaltfläche zur Hinzufügung eines neuen erwarteten oder gelieferten Pakets wird bei der ersten Nutzung der Anwendung oft übersehen. Auf mobiler Version tritt dieses Problem nicht auf.

2. Daraus ergaben sich von Testpersonen selbst formulierte Verbesserungsvorschläge:

- Erklärung über das Funktionsprinzip der App zu Beginn/bei der Registrierung, beispielsweise in Form eines Videos;
- Bei der Registrierung und auf der Seite der Benutzereinstellungen: Hausnummer statt Haus;

- Beispielhafte Beschreibung für Paketbezeichnung und Kommentar (bei Erstellen einer neuen Anzeige) in Form einer Informationsmitteilung in den Dialog integrieren;
- Benachrichtigung „Möchten Sie zuerst alle gelieferten/erwarteten Pakete durchgehen?“ (bei Erstellen einer neuen Anzeige) in Form einer Informationsmitteilung in den Dialog integrieren;
- Button „Paket hinzufügen“ als vierte Kachel statt „+“-Schaltfläche, die dahinter versteckten Kategorien zur Erstellung erwarteter/gelieferter Pakete als Auswahlmenü; als Alternative beide Optionen und den Button geöffnet auf der Dashboard-Seite lassen;
- „Benutzerdaten ändern“ ins Navigationsuntermenü übertragen;
- In den E-Mail-Benachrichtigungen an Anzeige-Herausgeber über gefundene Paketeigentümer/-empfänger eine Aufforderung zur Kontaktaufnahme per E-Mail oder Telefon mithilfe der ihm nun zur Verfügung gestellten Benutzerdaten einfügen.

3. Als Positiv empfunden wurde:

- Stilvolles Design, gut abgestimmte Farben und Fade-Animationen zwischen Seitenübergängen;
- Das selbsterstellte Bild auf der Startseite mit der Universitätsbibliothek Freiburg, Gebäude 101 der Technischen Fakultät und einem farblich hervorgehobenen Paket erregte die Aufmerksamkeit der Probanden;
- Registrierung mit Uni-Konto;
- Klare Informationen bezüglich der Sichtbarkeit von Nutzerdaten je nach Benutzerrolle;
- E-Mail-Benachrichtigung über erfolgreiche Registrierung und Änderung des Paketstatus und der Paketinformationen;
- Leichtigkeit der Änderung von Passwort oder Benutzerdaten;
- Klare Logik für Dashboard und zusätzliche Infos als Hilfe;

- Anzeige von Benutzerdaten und Karte zum Auffinden des Paketeigentümers/-empfängers sehr komfortabel;
- Labels auf Paketen (z.B. „von mit erwartet“) zur Verdeutlichung der Logik;
- Natives Aussehen der Anwendung bei Installation als PWA auf Mobilgerät;
- Bequemlichkeit der Fotoaufnahme auf Mobilgerät;
- Ermöglichung der Nutzung von „Face ID“ oder Fingerabdruck zum Entsperren innerhalb der PWA.

5.3 Zusammenfassung

Zusammenfassend lassen sich die User Tests Ergebnisse als positiv beurteilen:

- Alle aufgestellten Anforderungen wurden erfolgreich erfüllt. Nutzer konnten die Webanwendung sowohl auf dem Desktop als auch auf dem Handy benutzen, ohne sie aus dem App-Store/Play-Store installieren zu müssen;
- Die Registrierung mit einem Uni-Account war möglich, sodass die App für Instanzen wie die Technische Fakultät der Uni Freiburg optimiert ist;
- Die Nutzer konnten eigenständig Einträge über erwartete oder gelieferte Pakete erstellen, korrigieren und löschen;
- Die Nutzer konnten erfolgreich Such- und Filteroptionen anwenden; sie wurden über Änderungen im Paketstatus per E-Mail informiert.

Viel wichtiger ist aber, dass alle Nutzer die dahinter stehende Logik der App verstanden hatten. Alle fünf Testpersonen haben das Testszenario erfolgreich durchgespielt und konnten Kontakt mit Paketeigentümern/-empfängern aufnehmen und einen Abholtermin ausmachen, was letztlich auch den Sinn der Anwendung darstellt. In den künstlichen Testbedingungen scheint die App also ihrer Hauptaufgabe – Paketeigentümer und -empfänger zusammen zu bringen – gerecht zu sein, auch wenn an dieser kritischen Stelle manche Testpersonen etwas Unsicherheit zeigten, was genau nach der Auffindung eines Paketeigentümers/-empfängers zu tun ist. Durch kleine

Anpassungen an den E-Mail-Benachrichtigungen ließ sich das Problem jedoch leicht beheben.

Hier muss aber an die Grenzen der durchgeführten User Tests hingewiesen werden, denn sie bilden einen realen Nutzeralltag nur bedingt ab. Zum Einen stellt sich die zentrale Frage, ob reale Nutzer in der Tat den Aufwand auf sich nehmen werden, sich in der App zu registrieren und Paketinformationen verständlich genug anzugeben, sodass der Opponent sie auch erkennen und das Paket richtig zuordnen kann. Auch die Kontaktaufnahme, die außerhalb der App geschehen soll, kann aus mehreren Gründen scheitern. Das sind aber Faktoren, auf die man als App-Entwickler nur wenig Einfluß nehmen kann. Jedoch wurde die Registrierung dank der Möglichkeit, sich mit einem Uni-Konto einzuloggen weitestgehend vereinfacht.

Zum anderen ist die App für eine kollektive Nutzung konzipiert und kann nur dann erfolgreich funktionieren, wenn eine kritische Mehrheit aller Uni-/Unternehmensmitarbeiter die App aktiv nutzt. Dieses Ziel lässt sich gut erreichen, wenn man die potentiellen Nutzer auf die App per E-Mail oder im Intranet darauf aufmerksam macht. Zugleich kann man in der Anzeige das Funktionsprinzip gut erklären und eventuell ein Einführungsvideo demonstrieren, was die erste App-Verwendung erleichtern würde.

6 Fazit

In dieser Arbeit wurde die Entwicklung und Ausführung einer Webanwendung names „Package Information Tracker“ beschrieben, die den Anspruch erhebt, Probleme mit verlorenen Paketen innerhalb großer Institutionen, Firmen oder Campusse ohne zentralen Concierge-Service zu lösen.

Bei der Analyse der Ziele für diese Arbeit und der auf dem Markt vorhandenen Lösungen für ähnliche Problemstellungen wurden Komponentenanforderungen definiert. Diese Anforderungen dienten als Grundgerüst für die Realisierung der Anwendung.

Das Verwenden des Frameworks Nuxt.js für die Entwicklung einer universellen JavaScript-Anwendung auf Frontend-Seite, sowie Node.js und Express.js auf Backend-Seite hat sich als positiv für den Erstellungsprozess herausgestellt. Das Ergebnis ist eine App, die unabhängig von externen Cloud-Lösungen funktionieren kann und die Schnittstellen externer Dienste – wie der DHL Shipment Tracking API, der OpenCage Geocoding API, des Uni-Freiburg LDAP-Servers und eines SMTP-Servers – nur zur Erweiterung der eigenen Funktionalität benutzt. Durch die Containerisierung mithilfe von Docker und die Verwendung von PM2 als Prozessmanager hat sich die Anwendung als einfach bereitzustellen, konfigurierbar und ausfallsicher erwiesen.

In der Evaluation wurden generalisierte Benutzer (Personas) entsprechend der Zielgruppe der Anwendung entwickelt und ein Benutzertest durchgeführt. Aus den Ergebnissen wurden Probleme und Schwierigkeiten bei der Nutzung der Anwendung

sowohl auf dem Desktop als auch auf dem Handy herausgefunden und Verbesserungsvorschläge aus Nutzersicht gesammelt. Die Evaluierung zeigte, dass die Anforderungen der Anwendung erfolgreich erfüllt wurden und die Benutzer die Klarheit, Einfachheit und den Nutzen der Anwendung erkannten. Unter Berücksichtigung der Verbesserungsvorschläge der Nutzer wurde im Anschluss eine Instanz für die Technische Fakultät der Universität Freiburg bereitgestellt, die durch Verbreitung innerhalb des Instituts den dortigen Mitarbeitern das Erhalten von Paketen möglichst vereinfachen würde.

7 Literatur

1. Bundesnetzagentur (2020): Jahresbericht 2020. Märkte im digitalen Wandel, [online] https://www.bundesnetzagentur.de/SharedDocs/Mediathek/Jahresberichte/JB2020.pdf;?__blob=publicationFile&v=7 [22.01.2022]

2. iLost B.V., https://ilost.co [22.01.2022].

3. Pakenda, https://www.packenda.com/faq [22.01.2022].

4. Ebert, Christof (2019): Systematisches Requirements Engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten, 6. Aufl., Heidelberg: dpunkt.verlag.

5. Nuxt/PWA, https://pwa.nuxtjs.org/ [22.01.2022].

6. JavaScript (2022): Wikipedia, https://de.wikipedia.org/w/index.php?title=JavaScript&oldid=218918285 [22.01.2022].

7. Ackermann, Philip (2018): Professionell entwickeln mit JavaScript: Design, Patterns und Praxistipps für Enterprise-fähigen Code, 2. Aufl., Bonn: Rheinwerk Computing.

8. OpenJS Foundation, https://www.electronjs.org/ [22.01.2022].

9. Capacitor, https://capacitorjs.com/ [22.01.2022].

10. JSON Web Token (2022): Wikipedia, https://de.wikipedia.org/w/index.php?title=JSON_Web_Token&oldid=212574521 [22.01.2022].

11. Auth0, https://jwt.io/ [22.01.2022].

12. Nuxt.js (2022): Wikipedia, https://en.wikipedia.org/w/index.php?title=Nuxt.js&oldid=1067162988 [22.01.2022].

13. Isomorphic JavaScript (2022): Wikipedia, https://en.wikipedia.org/w/index.php?title=Isomorphic_JavaScript&oldid=1051914027 [22.01.2022].

14. Nuxt Server Side Rendering, https://nuxtjs.org/docs/concepts/server-side-rendering/ [22.01.2022].

15. Nuxt Directory Structure, https://nuxtjs.org/docs/get-started/directory-structure/ [22.01.2022].

16. Node.js (2022): Wikipedia, https://de.wikipedia.org/w/index.php?title=Node.js&oldid=217920284 [22.01.2022].

17. Express.js (2022): Wikipedia, https://en.wikipedia.org/w/index.php?title=Express.js&oldid=1067162595 [22.01.2022].

18. Springer, Sebastian (2018): Node.js: Das umfassende Handbuch. Serverseitige Web-Applikationen mit JavaScript entwickeln, 3. Aufl., Bonn: Rheinwerk Computing.

19. Bundesamt für Sicherheit in der Informationstechnik (2013): Leitfaden zur Entwicklung sicherer Webanwendungen. Empfehlungen und Anforderungen

an Auftraggeber aus der öffentlichen Verwaltung, [online] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Webanwendungen/Webanw_Auftraggeber.pdf?__blob=publicationFile&v=1 [22.01.2022].

20. Npm, Inc., https://www.npmjs.com/package/pm2 [22.01.2022].

21. Tomlin, Craig (2018): UX Optimization. Combining Behavioral UX and Usability Testing Data to Optimize Websites, New York City: Apress.