

FH Hagenberg

WEA5 Projekt

DeliFHery

Fischlhammer Patrick - s2210307012

13.12.2025

Inhaltsverzeichnis

1.	Architektur	3
1.1.	Überblick (Compodoc Overview)	3
1.2.	Komponentenstruktur	4
1.3.	Services und Abhängigkeiten	5
1.4.	Routing-Architektur.....	6
1.5.	Allgemeine Informationen	7
2.	Navigationswege	7
2.1.	Öffentliche Seiten	7
2.2.	Geschützte Seiten (Login erforderlich)	7
3.	Bebilderter Testlauf	8
3.1.	Start / Login	8
3.2.	Register	9
3.3.	Preisberechnung.....	10
3.4.	Shipment anlegen.....	11
3.5.	Status / Tracking	13
3.6.	Statistik.....	14
3.7.	Kontakt möglichkeiten verwalten.....	15
3.8.	Benachrichtigung Aktivieren	16
4.	Einsatz von KI-Werkzeugen	17
4.1.	4.1 Verwendete Werkzeuge (z.B. ChatGPT)	17
4.2	Mit KI erstellte Codebereiche.....	18
5.	Beantworten Sie folgende Fragen:	20
5.1.	Was ist zu tun, wenn sich URLs ändern?	20
5.2.	Wie invasiv ist der Eingriff in Ihre Anwendung, um diese zu ändern?	20
5.3.	Wie stellen Sie sicher, dass bestimmte Seiten nur nach einem Login zugreifbar sind?....	20
5.4.	Wie stellen Sie eine korrekte Dateneingabe sicher?.....	20
5.5.	Was passiert, wenn Aufrufe an das Backend Fehler produzieren?.....	20

1. Architektur

1.1. Überblick (Compodoc Overview)

DeliFHery ist eine Client-Server-Anwendung bestehend aus einem Backend (SWK5) und einem Frontend (WEA5).

In diesem Dokumentationsteil wird ausschließlich das Frontend (WEA5) beschrieben.

Das Frontend ist als Angular Single-Page-Application mit Standalone-Components umgesetzt. Die Anwendung ist klar in funktionale Bereiche gegliedert, um Wartbarkeit und Erweiterbarkeit zu gewährleisten.

Die Projektstruktur gliedert sich wie folgt:

Components

Enthält alle UI-Komponenten der Anwendung.

Für jede Anzeigeseite existiert eine eigene Komponente (z.B. Preisberechnung, Sendung anlegen, Statusanzeige, Statistik). Die Komponenten sind primär für Darstellung und Benutzerinteraktion zuständig.

Environments

Beinhaltet die Umgebungsdefinitionen der Anwendung.

Hier wird unter anderem die Basis-URL des Backends (API-Endpunkt) konfiguriert, wodurch Änderungen an der Zielumgebung zentral vorgenommen werden können.

Guards

Enthält Zugriffsschutzmechanismen für geschützte Routen.

Mithilfe eines AuthGuards wird geprüft, ob ein gültiger Keycloak-Authentifizierungstoken vorhanden ist, bevor bestimmte Seiten angezeigt werden dürfen.

Services

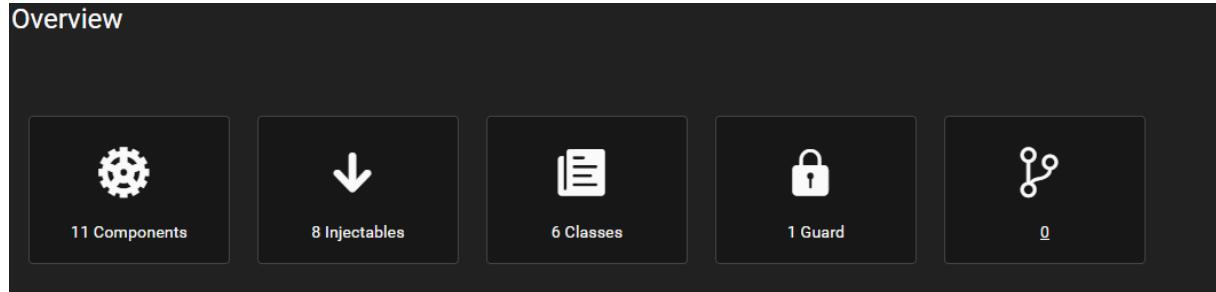
Kapseln die gesamte Kommunikation mit dem Backend.

Alle API-Aufrufe werden zentral in Services implementiert, wodurch Komponenten keine direkte Abhängigkeit von HTTP-Logik haben.

Shared

Enthält gemeinsame Datenmodelle (Klassen und Enums), welche die Datenstrukturen des Backends abbilden.

Diese werden von Komponenten und Services gemeinsam genutzt. Komponentenstruktur (Components / Komponentenbaum)



1.2. Komponentenstruktur

Die Anwendung ist in mehrere Angular-Komponenten aufgeteilt.

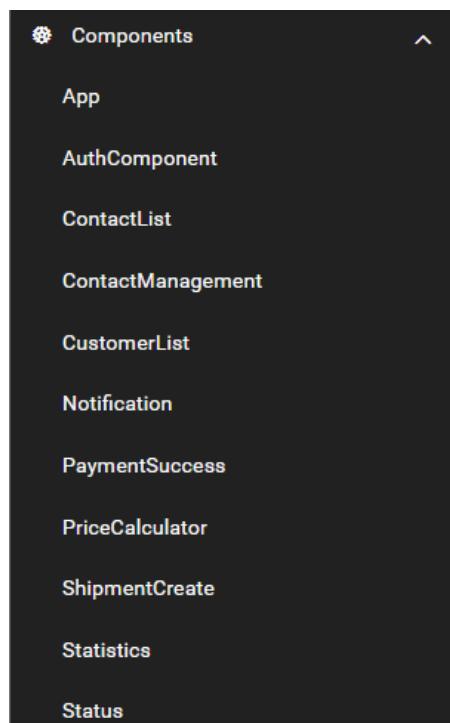
Für jede Seite der Anwendung existiert eine eigene Komponente, die für die Darstellung und die Interaktion mit dem Benutzer zuständig ist.

Die Komponente App bildet den Einstiegspunkt der Anwendung und stellt den Router bereit, über den zwischen den einzelnen Seiten gewechselt wird.

Zu den wichtigsten Seitenkomponenten gehören PriceCalculator, ShipmentCreate, Status, Statistics, AuthComponent sowie Notification.

Diese Komponenten decken die zentralen Funktionen der Anwendung ab, wie etwa die Preisberechnung, das Anlegen von Sendungen, die Anzeige des Versandstatus, statistische Auswertungen sowie Login- und Benachrichtigungsfunktionen.

Durch diese klare Aufteilung bleibt der Code übersichtlich und einzelne Seiten können unabhängig voneinander angepasst oder erweitert werden.



1.3. Services und Abhängigkeiten

Die Kommunikation mit dem Backend erfolgt zentral über Services.

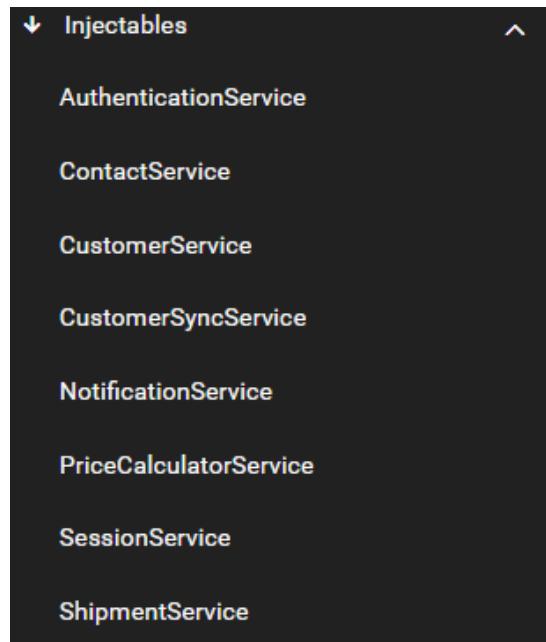
Diese Services enthalten alle HTTP-Aufrufe und werden von den einzelnen Komponenten genutzt, um Daten zu laden oder zu speichern.

Wichtige Services sind unter anderem der ShipmentService, PriceCalculatorService, ContactService und NotificationService.

Durch diese Struktur enthalten die Komponenten selbst keine Backend-Logik, sondern greifen ausschließlich auf die Services zu.

Zusätzlich existieren Services für Authentifizierung und Sitzungsverwaltung.

Ein HTTP-Interceptor sorgt dafür, dass bei Backend-Aufrufen automatisch der Authentifizierungstoken mitgesendet wird.



1.4. Routing-Architektur

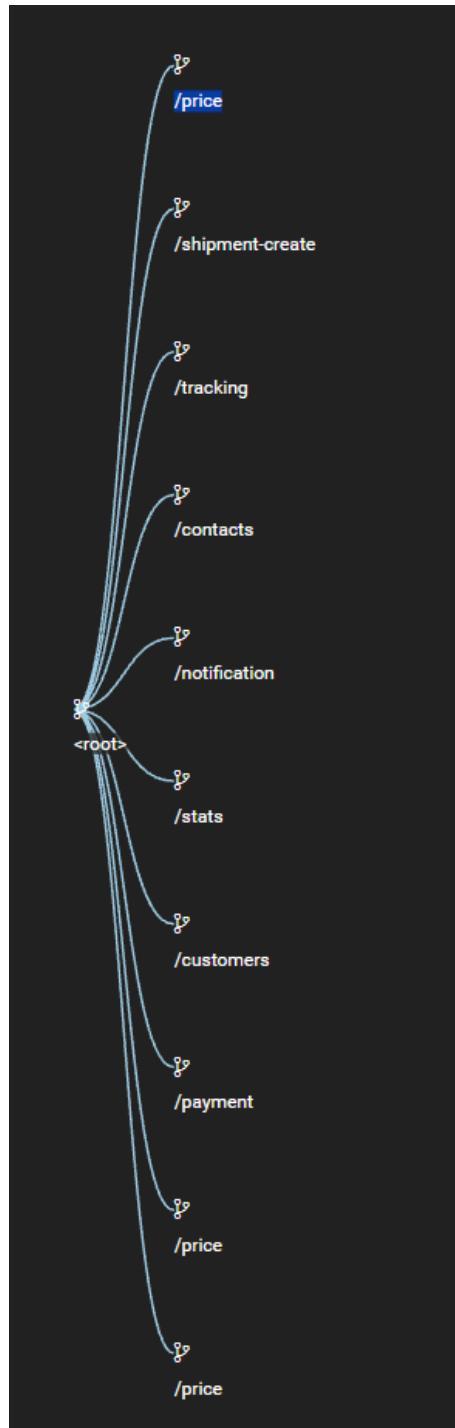
Die Navigation innerhalb der Anwendung erfolgt über das Angular-Routing.

Alle Routen sind zentral definiert und verknüpfen URLs mit den jeweiligen Seitenkomponenten.

Öffentliche Seiten (z.B. Login) sind ohne vorherige Authentifizierung erreichbar.

Geschützte Seiten sind über einen AuthGuard abgesichert und können nur nach erfolgreichem Login aufgerufen werden.

Durch diese zentrale Routing-Definition ist die Navigation übersichtlich gehalten und Änderungen an Navigationswegen können an einer Stelle vorgenommen werden.



Die Route /price erscheint mehrfach, da sie sowohl als Standard-Weiterleitung (Root-Pfad und index.html) als auch als eigene Seitenroute definiert ist.

1.5. Allgemeine Informationen

Das Frontend wurde parallel zur Backend-Entwicklung umgesetzt. Dadurch kam es im Laufe des Projekts mehrfach zu Änderungen an Schnittstellen, Datenstrukturen und Rückgabewerten, was sich direkt auf das Frontend ausgewirkt hat.

In diesem Zusammenhang wurden teilweise Komponenten erstellt, die im finalen Projekt nicht mehr aktiv verwendet werden. Ein Beispiel dafür ist die Komponente `customer-list`, welche ursprünglich zur Überprüfung der ersten Backend-Anbindung diente. Diese Komponente wird im fertigen Projekt nicht mehr benötigt und ist daher im Routing auskommentiert, wurde jedoch nicht vollständig entfernt.

Zusätzlich kam es während der Entwicklung zu Änderungen an gemeinsamen Datenstrukturen, beispielsweise an der Klasse `Status`. In einer frühen Entwicklungsphase lieferte das Backend numerische Statuswerte, die in einer späteren Ausbaustufe auf textuelle Statusbezeichnungen umgestellt wurden. Solche Änderungen erforderten Anpassungen im Frontend und führten dazu, dass ältere Codebestandteile oder Kommentare teilweise nur noch informativen Charakter haben.

Bewusst wurden bereits erstellte Komponenten und Services nicht gelöscht, da diese nach Backend-Änderungen wiederholt zur Kontrolle und zum Testen der Funktionalität herangezogen wurden. Dadurch konnte nach jeder Anpassung überprüft werden, ob bestehende Funktionen weiterhin korrekt arbeiten.

Insgesamt spiegelt der Codebestand den iterativen Entwicklungsprozess wider, bei dem sich Anforderungen und Implementierungen schrittweise weiterentwickelt haben.

2. Navigationswege

2.1. Öffentliche Seiten

Öffentliche Seiten können ohne vorherigen Login aufgerufen werden.

Dazu zählen Seiten, die keine sensiblen Daten anzeigen oder keinen Benutzerkontext benötigen

In der DeliFHer App sind das konkret:

- **Login-Seite** (/auth bzw. Einstieg über Login)
- **PaymentSuccess** (/payment)
(wird nach externer Zahlung aufgerufen)

2.2. Geschützte Seiten (Login erforderlich)

Der Großteil der Anwendung ist nur nach einem erfolgreichen Login zugänglich.

Diese Seiten sind über einen AuthGuard abgesichert, welcher vor dem Laden der Seite prüft, ob ein gültiger Authentifizierungstoken vorhanden ist.

Geschützte Seiten sind unter anderem:

- /price
- /shipment-create
- /tracking
- /contacts
- /notification
- /stats

3. Bebilderter Testlauf

3.1. Start / Login

The screenshot shows a simple login form with a single input field and a blue 'Login' button.

Willkommen bei DeliFHery
<input type="text"/>
<input type="button" value="Login"/>

Abbildung 1: Start Seite

The screenshot shows a Keycloak login interface with a dark background. It features a title 'WEA5 AUTH DEMO', a 'Sign in to your account' heading, and two input fields for 'Email' (containing 'wea5user') and 'Password' (containing masked text). Below the password field is a 'Forgot Password?' link. A large blue 'Sign In' button is at the bottom, and a 'New user? Register' link is at the bottom right.

WEA5 AUTH DEMO
Sign in to your account
Email wea5user
Password
Forgot Password?
<input type="button" value="Sign In"/>
New user? Register

Abbildung 2: Keycloak Login Seite

3.2. Register

The screenshot shows a registration form titled "Register". It includes fields for Email, Password, Confirm password, First name, and Last name. The "Email" field contains "Test@email.at", the "Password" field contains "****", the "Confirm password" field also contains "****", the "First name" field contains "TestFirstname", and the "Last name" field contains "TestLastName". A blue "Register" button is at the bottom. Below the form is a link "« Back to Login".

Abbildung 3: KeyColak Registirene

Gitter	AZ -> Id	AZ FirstName	AZ LastName	AZ UserName
1	31944eed-5208-4021-a5bd-4ab7f532e5f4	TestFirstname	TestLastName	test@email.at

Abbildung 4: Anlage in der Datenbank

3.3. Preisberechnung

Willkommen bei DeliHery

Preis berechnen Paket anmelden Kontakte Benachrichtigungen Statistik Tracking Logout

Preisberechnung
Bitte Paketdaten eingeben

Länge (cm)

Breite (cm)

Höhe (cm)

Gewicht (kg)

Preis berechnen

Preisregeln

Paketdaten	Volumenpreis	Gewichtspreis	Endpreis (~10% Dezember)
10x10x10 cm, 1 kg	0.20 €	1.20 €	1.08 €
20x20x20 cm, 2 kg	1.60 €	2.40 €	2.16 €
50x30x20 cm, 3 kg	6.00 €	3.60 €	5.40 €
20x20x20 cm, 10 kg	1.60 €	12.00 €	10.80 €

Abbildung 5: Preisberchung übersicht mit Dezember Preisen

Preisberechnung
Bitte Paketdaten eingeben

Länge (cm)

Breite (cm)

Höhe (cm)

Gewicht (kg)

Preis berechnen

Berechneter Preis
1.08 €

Preisregeln

Paketdaten	Volumenpreis	Gewichtspreis	Endpreis (~10% Dezember)
10x10x10 cm, 1 kg	0.20 €	1.20 €	1.08 €
20x20x20 cm, 2 kg	1.60 €	2.40 €	2.16 €
50x30x20 cm, 3 kg	6.00 €	3.60 €	5.40 €
20x20x20 cm, 10 kg	1.60 €	12.00 €	10.80 €

Abbildung 6: Preise vom Backend berechnet

3.4. Shipment anlegen

 Neuen Paketversand anlegen
Bitte Versanddaten eingeben

Absender

Straße
Softwarepark

Hausnummer
11

PLZ
4232

Stadt
Hagenberg in Mühlkreis

Empfänger

Straße
Garnisonstraße

Hausnummer
21

PLZ
4020

Stadt
Linz

Paketdaten

Länge
10

Breite
10

Höhe
10

Gewicht (kg)
1

 Paket versenden

Abbildung 7: Schipment anlegen

Complete Payment

PAYMENT ID: 7bb7b91c-4a13-459a-9250-700a146bd700
AMOUNT: €1,08

Card Number: *

4111111111111111

Use special numbers to simulate errors:

- 4111111111111111 - Success
- 4000000000000002 - Card Expired
- 4000000000000003 - Invalid Card Number
- 40000000000000028 - Wrong Security Code
- 40000000000000036 - Insufficient Funds

Expiry Date (MM/YY): *

09/27

CVV: *

123

Cardholder Name: *

john Doe

Pay €1,08

Payment Already Completed

This payment has been completed successfully.

Abbildung 9: Weiterleitung auf Payment system

Willkommen bei DeliFhery

Preis berechnen Paket anmelden Kontakte Benachrichtigungen Statistik Tracking Logout

Zahlung erfolgreich!

Tracking-ID: ME7J-9HGC-TAHX

Preis: 1.08 €

Abbildung 10: Rückleitung aufs Frontend

Zahlung fehlgeschlagen!

Abbildung 11: Rückleitung bei Zahlung Fehlgeschlagen

3.5. Status / Tracking

Willkommen bei DeliFHery

Preis berechnen Paket anmelden Kontakte Benachrichtigungen Statistik Tracking [Logout](#)

Paket Status Abfrage

Trackingnummer
ME7J-9HGC-TAHX

PLZ
4020

[Status Check](#)

Absender
Softwarpark 11
4232 Hagenberg

Empfänger
Granisonstraße 21
4020 Linz

Status Historie

Datum	Status	Info
2025-12-13T14:51:37.947	Registered	Package registered

Abbildung 12: Tracking mit richtiger Empfänger PLZ

Paket Status Abfrage

Trackingnummer
ME7J-9HGC-TAHX

PLZ
4022

[Status Check](#)

Sendung nicht gefunden

Abbildung 13: Tracking mit Falscher PLZ

3.6. Statistik

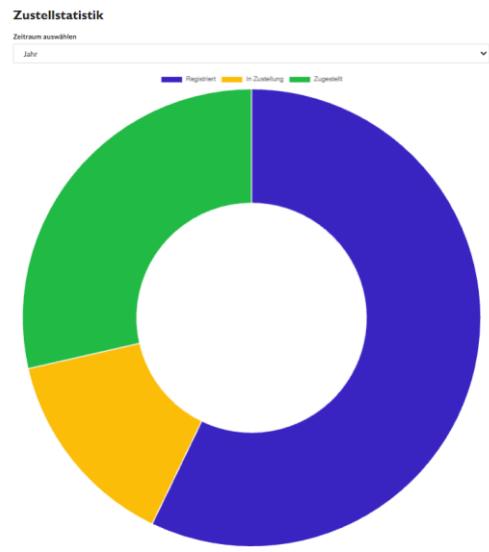


Abbildung 15: Statistik Lezten 365 Tage

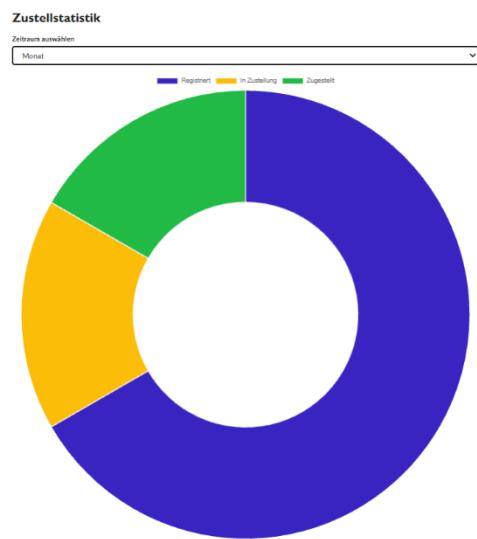


Abbildung 14: Statistik letzten 30 Tage

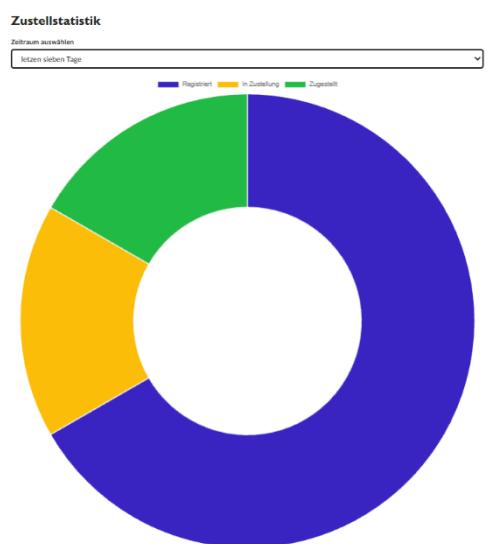


Abbildung 16: Statistik Lezten 7 Tage

3.7. Kontaktmöglichkeiten verwalten

Kontaktmöglichkeiten verwalten

Type	Bitte auswählen	
Daten		
Hinzufügen		
Type	Wert	
📞 Phone	06602346269	Löschen
✉ Email	s22103070102@fh-hagenberg.at	Löschen

Abbildung 17 Kontakt Möglichkeiten hinzufügen und löschen

Kontaktmöglichkeiten verwalten

Type	Bitte auswählen	
Daten		
Hinzufügen		
Type	Wert	
✉ Email	p.fischlhammer@yahoo.de	Löschen
📞 Phone	06602346269	Löschen
✉ Email	s22103070102@fh-hagenberg.at	Löschen

Abbildung 18 Nach hinzufügen

3.8. Benachrichtigung Aktivieren

Benachrichtigungen:

Trackingnummer
ME7J-9HGC-TAHX

PLZ
4021

🔔 Benachrichtigung aktivieren 🔕 Benachrichtigung deaktivieren

Konnte Benachrichtigungen nicht aktivieren.

Abbildung 19 Benachrichtigung Aktivieren/Deaktivieren mit Falscher PLZ

Benachrichtigungen:

Trackingnummer
ME7J-9HGC-TAHX

PLZ
4020

🔔 Benachrichtigung aktivieren 🔕 Benachrichtigung deaktivieren

Benachrichtigungen aktiviert.

4. Einsatz von KI-Werkzeugen

4.1. 4.1 Verwendete Werkzeuge (z.B. ChatGPT)

Als KI-Werkzeug wurde ChatGPT (Version 5.1) eingesetzt. Der Hauptfokus lag dabei auf der Fehlersuche, Fehleranalyse sowie auf der Unterstützung bei Syntaxfragen. Besonders bei komplexeren Problemen im Zusammenspiel von Backend und Frontend erwies sich ChatGPT als hilfreich, da dadurch langwierige Recherchen und Fehlversuche reduziert werden konnten.

Beim Aufsetzen der Keycloak-Authentifizierung sowie bei der Implementierung der Statistik-Seite wurde ChatGPT unterstützend verwendet, um typische Konfigurationsfehler schneller zu identifizieren und Lösungsansätze zu erhalten. Dadurch konnte viel Zeit eingespart werden, die sonst für das Durchsuchen von Dokumentationen und Foren notwendig gewesen wäre.

Zusätzlich wurde ChatGPT während der Entwicklung genutzt, um schnell HTML-Beispielcode zu erzeugen, mit dem einzelne Funktionen oder Seiten isoliert getestet werden konnten. Auch bei der Formatierung und Gestaltung der Benutzeroberfläche half die KI, da dadurch keine einzelnen CSS-Klassen manuell recherchiert werden mussten und rasch ein brauchbares Ausgangsergebnis zur Verfügung stand.

Auch bei der Erstellung dieser Dokumentation kam die KI unterstützend zum Einsatz, insbesondere zur Korrektur der Rechtschreibung und zum Ausformulieren von Textpassagen.

Insgesamt war ChatGPT vor allem in Phasen mit vielen Backend-Änderungen und anschließenden Fehlerkorrekturen im Frontend sehr hilfreich. Dadurch entwickelte sich ein besseres Verständnis dafür, an welchen Stellen Fehler entstehen können und wie systematisch bei der Fehlersuche vorzugehen ist.

4.2 Mit KI erstellte Codebereiche

Ich habe hauptsächlich den code vom Statistik generieren lassen und habe es auf ein kreis diagram umgeändert und auch noch die einzelnenfarben etwas angepasst

```
ngOnInit() {
  this.shipmentService.getAllByCustomer()
    .subscribe(res => {
      if (res) {
        this.shipments = res;
        this.renderChart();
      }
    });
}

// Datum aus dem letzten History-Eintrag (echtes Shipment-Datum)
private getShipmentDate(s: Shipment): Date {
  const history = s.history as ShipmentStatusEntry[];
  if (!history || history.length === 0)
    return new Date(0); // "sehr alt" fällt aus jedem Filter raus
  const last = history[history.length - 1];
  // falls timestamp evtl. als string ankommt
  return new Date(last.timestamp);
}

// Zeitraum-Filter: Woche / Monat / Jahr
private getFilteredShipments(): Shipment[] {
  const now = new Date();
  let days = 7;

  switch (this.selectedRange) {
    case "week":
      days = 7;
      break;
    case "month":
      days = 30;
      break;
    case "year":
      days = 365;
      break;
  }

  const threshold = new Date(now.getTime() - days * 24 * 60 * 60 * 1000);

  return this.shipments.filter(s => {
    const d = this.getShipmentDate(s);
    return d >= threshold;
  });
}
```

```

    });
}

// Chart bauen
private renderChart() {
    if (this.chart) this.chart.destroy();

    const filtered = this.getFilteredShipments();

    // Status laut deinem Enum:
    // 0 = Registered
    // 1 = PackageInTransit
    // 2 = Received
    const registered = filtered.filter(s => s.status === "Registered").length;
    const inTransit = filtered.filter(s => s.status ===
"PackageInTransit").length;
    const delivered = filtered.filter(s => s.status === "Received").length;

    this.chart = new Chart('statsChart', {
        type: 'doughnut',
        data: {
            labels: [
                'Registriert',
                'In Zustellung',
                'Zugestellt'
            ],
            datasets: [{
                label: 'Status der Pakete',
                data: [registered, inTransit, delivered],
                backgroundColor: [
                    '#3924c2ff', // blau - registriert
                    '#fbbd08', // gelb - in Zustellung
                    '#21ba45' // grün - zugestellt
                ]
            }]
        },
        options: {
            plugins: {
                legend: { display: true }
            }
        }
    });
}

onRangeChange() {
    this.renderChart();
}

```

5. Beantworten Sie folgende Fragen:

5.1. Was ist zu tun, wenn sich URLs ändern?

Wenn sich eine URL ändert, muss diese zentral in der Datei environment.ts angepasst werden, da dort die Basis-URL für das Backend definiert ist. Die Services greifen ausschließlich über diese Konfiguration auf die API zu und enthalten keine hardcodierten Backend-URLs.

5.2. Wie invasiv ist der Eingriff in Ihre Anwendung, um diese zu ändern?

Der Eingriff ist gering invasiv, da eine Änderung der Basis-URL an einer einzigen Stelle ausreicht.

Alle Services verwenden die URL über das environment-Objekt, zum Beispiel:

```
private baseUrl = `${environment.api}/customer`;
```

Dadurch wirken sich Änderungen automatisch auf alle API-Aufrufe aus.

Ein höherer Anpassungsaufwand entsteht nur dann, wenn sich einzelne API-Endpunkte selbst ändern (z. B. Pfad oder HTTP-Methode). In diesem Fall muss lediglich der betroffene Service angepasst werden, nicht jedoch die Komponenten.

5.3. Wie stellen Sie sicher, dass bestimmte Seiten nur nach einem Login zugreifbar sind?

Der Zugriff auf geschützte Seiten wird über Route Guards umgesetzt.

Dabei wird geprüft, ob ein gültiger Authentifizierungs-Token von Keycloak vorhanden ist. Ist kein Token vorhanden oder ist dieser ungültig, wird der Zugriff auf die entsprechende Route verhindert und der Benutzer nicht auf die Seite weitergeleitet.

5.4. Wie stellen Sie eine korrekte Dateneingabe sicher?

Die korrekte Dateneingabe wird durch Eingabeverifikation im Frontend sichergestellt.

Dabei wird überprüft, ob alle Pflichtfelder ausgefüllt sind und ob die eingegebenen Werte ein gültiges Format besitzen (z. B. vierstellige Postleitzahl). Ungültige Eingaben führen zu einer entsprechenden Fehlermeldung und verhindern das Absenden der Daten

5.5. Was passiert, wenn Aufrufe an das Backend Fehler produzieren?

Tritt bei einem Backend-Aufruf ein Fehler auf (z. B. HTTP 4xx oder 5xx), wird dieser im Frontend abgefangen.

Der Fehler wird in der Browserkonsole protokolliert und dem Benutzer wird eine passende Fehlermeldung angezeigt. Dadurch ist sowohl eine technische Nachvollziehbarkeit für die Entwicklung als auch eine verständliche Rückmeldung für den Benutzer gewährleistet.