

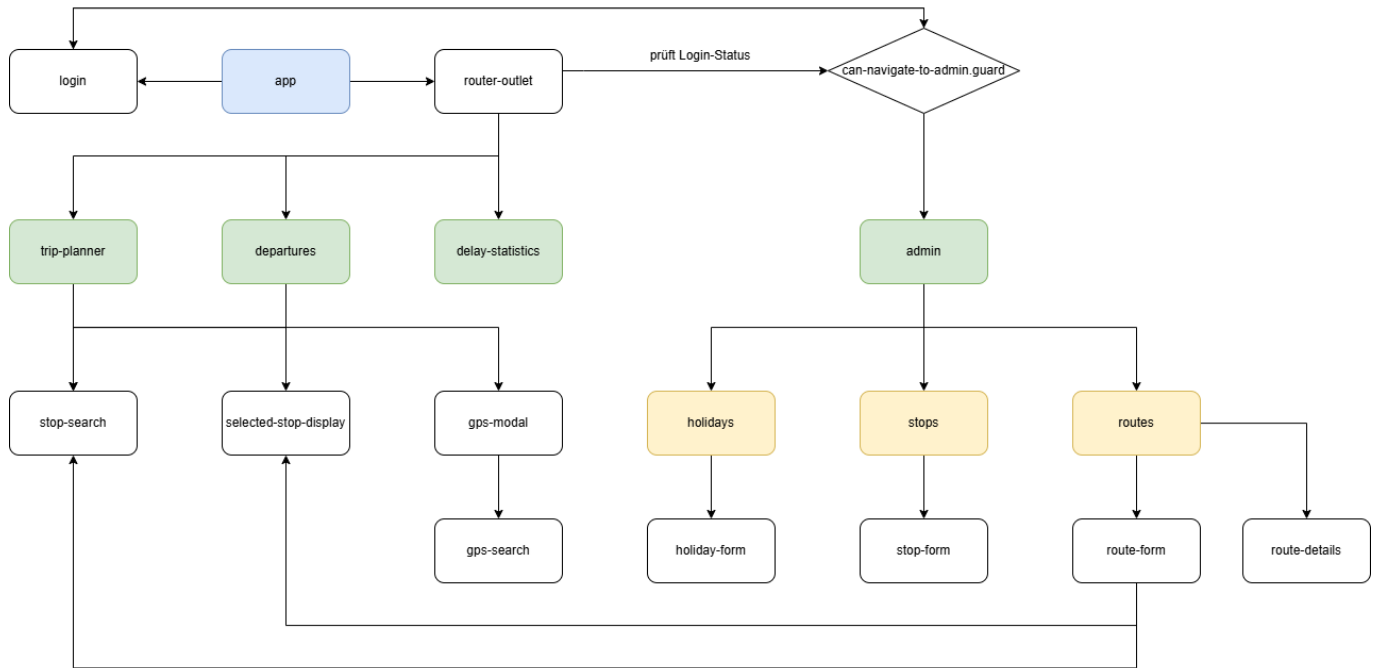
Dokumentation WEA5-Projekt Bernd Zeiler

Inhaltsverzeichnis

- [Architektur](#)
 - [Komponentenbaum-Diagramm](#)
 - [Wichtige Komponenten](#)
 - [Zusammenhang Services mit Komponenten](#)
 - [Models](#)
 - [Helpers](#)
- [Navigationswege-Diagramm](#)
- [Testlauf](#)
 - [Szenario 1: Verkehrsunternehmen melden sich am System an, um die Pflege des neuen Fahrplans durchzuführen](#)
 - [Szenario 2: Feiertage und Schulferien für 2025 werden verwaltet, um den Fahrplan davon abhängig machen zu können](#)
 - [Szenario 3: Haltestellen können angelegt und bearbeitet werden](#)
 - [Szenario 4: Routen können inkl. Abfahrtszeiten erstellt werden](#)
 - [Szenario 6: Fahrplanabfragen können gemacht werden](#)
 - [beinhaltet Szenario 5: Haltestellen können gesucht werden, z. B. nach dem Namen oder anhand des aktuellen Standorts](#)
 - [Szenario 7: Anzeigetafeln zeigen die nächsten Abfahrten für die aktuelle Haltestelle](#)
 - [Szenario 8: Busse checken ihren aktuellen Standort ein](#)
 - [Szenario 9: Verspätungen werden in der Suche und auf der Anzeigetafel berücksichtigt](#)
 - [Szenario 10: Eine Verspätungsstatistik gibt Auskunft über die Pünktlichkeit aller Verkehrsmittel](#)
 - [beinhaltet Szenario 11: Darstellung der Verspätungsstatistik in Diagrammen](#)
- [Fragen beantworten](#)
 - [Was ist zu tun, wenn sich URLs ändern? Wie invasiv ist der Eingriff in Ihre Anwendung um diese zu ändern?](#)
 - [Wie stellen Sie sicher, dass bestimmte Seiten nur nach einem Login zugreifbar sind?](#)
 - [Wie stellen Sie eine korrekte Dateneingabe sicher?](#)
 - [Was passiert, wenn Aufrufe an das Backend Fehler produzieren?](#)
- [Setup](#)
- [Externe Teile](#)

Architektur

Komponentenbaum-Diagramm



Hinweis: Alle Rechtecke demonstrieren meine Komponenten (Der Guard in der Raute ist keine direkte Komponente). Die Farben haben keine konkrete Bedeutung, sie dienen nur für die Visualisierung der hierarchischen Ebenen.

Wichtige Komponenten

TripPlannerComponent

Die Startseite der Applikation. Hier kann man Fahrplanabfragen durchführen. Die Komponente beinhaltet die Auswahl der Start- und Zielhaltestelle, Datum und Uhrzeit, Abfahrts- oder Ankunftszeit und die gewünschte Anzahl an Verbindungen.

Für die Auswahl der Start- bzw. Zielhaltestelle kann man entweder über die **StopSearchComponent** mit Namen oder über die **GpsSearchComponent** mit dem aktuellen Standort suchen.

Die gefundenen Verbindungen werden mit Berücksichtigung von möglichen Verspätungen(Check-Ins) in einer Tabelle angezeigt.

DeparturesComponent

Hier werden die nächsten Abfahrten für eine bestimmte Haltestelle angezeigt. Die Komponente beinhaltet eine Haltestelle, Datum und Uhrzeit und die gewünschte Anzahl an Verbindungen.

Für die Auswahl der Haltestelle kann der User wieder entscheiden ob er nach Namen oder GPS-Koordinaten der Haltestelle suchen will.

Die nächsten Abfahrten werden wieder mit Berücksichtigung von Verspätungen in einer Tabelle angezeigt.

DelayStatisticsComponent

Hier kann man sich eine Verspätungsstatistik über die Pünktlichkeit aller Verkehrsmittel ausgeben lassen. Die Komponente beinhaltet den gewünschten Zeitraum(Start- und Enddatum) und eine optionale Routennummer, falls die Statistikauswertung spezifisch für eine Route ausgegeben werden soll.

Die Ausgabe der durchschnittlichen Verspätungen wird in Form eines Balkendiagramms (mithilfe der Libraries **chart.js** und **ng2-charts**) angezeigt.

LoginComponent

Hier kann man sich als Administrator anmelden. Über diese Komponente wird man auf die **Keycloak**-Anmeldemaske weitergeleitet, wo sich der Admin mit seinen Zugangsdaten einloggen kann. Bei erfolgreicher Anmeldung wird man auf die eigentliche Webseite zurückgeleitet und die **AdminComponent** wird über den **CanNavigateToAdminGuard** damit freigeschalten.

AdminComponent

Hier kann man Feiertage/Schulferien, Haltestellen und Routen verwalten. Über diese Komponente gelangt man zu den jeweiligen 3 Komponenten.

HolidaysComponent

Hier kann man Feiertage bzw. Schulferien verwalten. Die vorhandenen Feiertage (inkl. Kennzeichnung von Schulferien) werden in einer Tabelle angezeigt. Man kann dort einen neuen Feiertag anlegen als auch einen vorhandenen bearbeiten, indem man auf die **HolidayFormComponent** weitergeleitet wird. Das Löschen eines vorhandenen Feiertags ist ebenfalls möglich.

StopsComponent

Hier kann man Haltestellen verwalten. Die vorhandenen Haltestellen werden in einer Tabelle angezeigt. Man kann dort eine neue Haltestelle anlegen als auch eine vorhandene bearbeiten, indem man auf die **StopFormComponent** weitergeleitet wird. Das Löschen einer vorhandenen Haltestelle ist ebenfalls möglich.

RoutesComponent

Hier kann man Verbindungen (Routen und deren Haltestellen) verwalten. Die vorhandenen Verbindungen werden in einer Tabelle angezeigt. Man kann dort eine neue Route inkl. aller Haltestellen-Informationen mit einem einzigen Request anlegen, indem man auf die **RouteFormComponent** weitergeleitet wird. Ebenso kann man sich die Details von vorhandenen Routen anzeigen lassen, indem man auf die **RouteDetailsComponent** weitergeleitet wird. Das Bearbeiten und Löschen ist hier nicht möglich, weil es lt. Aufgabenstellung nicht gefordert war.

StopSearchComponent

Diese Komponente ist ein Suchfeld, wo Haltestellen anhand ihres Namens gesucht werden können. Diese Komponente ist Teil innerhalb der **TripPlannerComponent**, **DeparturesComponent** und **RouteFormComponent**.

SelectedStopDisplayComponent

Diese Komponente zeigt z.B. in der Abfahrten-Seite die letztendlich ausgewählte Haltestelle des Users an, welche von der **StopSearchComponent** oder **GpsSearchComponent** gefunden wurde.

GpsModalComponent

Diese Komponente öffnet z.B. in der Abfahrten-Seite ein Dialogfenster, wo dann die **GpsSearchComponent** angezeigt wird.

GpsSearchComponent

In dieser Komponente kann man sich anhanden des aktuellen Standorts die nächstgelegenen Haltestellen anzeigen lassen.

Zusammenhang Services mit Komponenten

TripPlannerService

Die **TripPlannerComponent** erhält hier die gefundenen Verbindungen für die mitgegebene Start- und Zielhaltestelle und weiteren Parametern.

DepartureService

Die **DeparturesComponent** erhält hier die gefundenen nächsten Verbindungen ausgehend von der mitgegebenen Haltestelle, wo man wegfahren will.

DelayStatisticsService

Die **DelayStatisticsComponent** erhält hier die Verspätungsstatistikdaten ausgehend von dem eingegebenen Zeitraum und optionaler Routennummer.

AuthenticationService

Dieses Service wird von der **LoginComponent** verwendet, um den Login-Prozess über Keycloak zu initialisieren und den Authentifizierungsstatus des Benutzers zu prüfen.

Sie wird außerdem von der **AppComponent** genutzt, um den Login-Status abzufragen und auch um den Benutzer wieder auszuloggen.

HolidaysService

Dieses Service wird von der **HolidaysComponent** und **HolidayFormComponent** verwendet, um die notwendigen Create-Read-Update-Delete(CRUD)-Abfragen ausführen zu können.

StopsService

Dieses Service wird von der **StopsComponent** und **StopFormComponent** verwendet, um die notwendigen Create-Read-Update-Delete(CRUD)-Abfragen ausführen zu können.

Auch die **RouteDetailsComponent** verwendet das Service, um die Haltestellen für eine bestimmte Route detailliert ausgeben zu können.

Ebenfalls benötigen die **StopSearchComponent** und **GpsSearchComponent** das Service, um die Haltestellen anhand eines Suchbegriffes(Name oder GPS-Position der Haltestelle) abfragen zu können.

RoutesService

Dieses Service wird von der **RoutesComponent**, der **RouteFormComponent** und der **RouteDetailsComponent** verwendet, um Get- und Create-Anweisungen durchführen zu können.

Außerdem benötigt dieses Service auch die **DelayStatisticsComponent**, damit sich der User aus allen vorhandenen Routen die Verspätungsstatistik anzeigen lassen kann.

Models

Models wurden erstellt, um die Datenstruktur der Anwendung typischer zu definieren. Sie erleichtern die Interaktion zwischen Services und Komponenten sowie die Verarbeitung von Backend-Daten, wie z. B. Haltestellen, Verbindungen und Benutzerinformationen.

Helpers

ChartConfig

Definiert die Konfiguration und die Datenstruktur für die Darstellung der Verspätungsstatistik in einem Balkendiagramm mithilfe von `chart.js`.

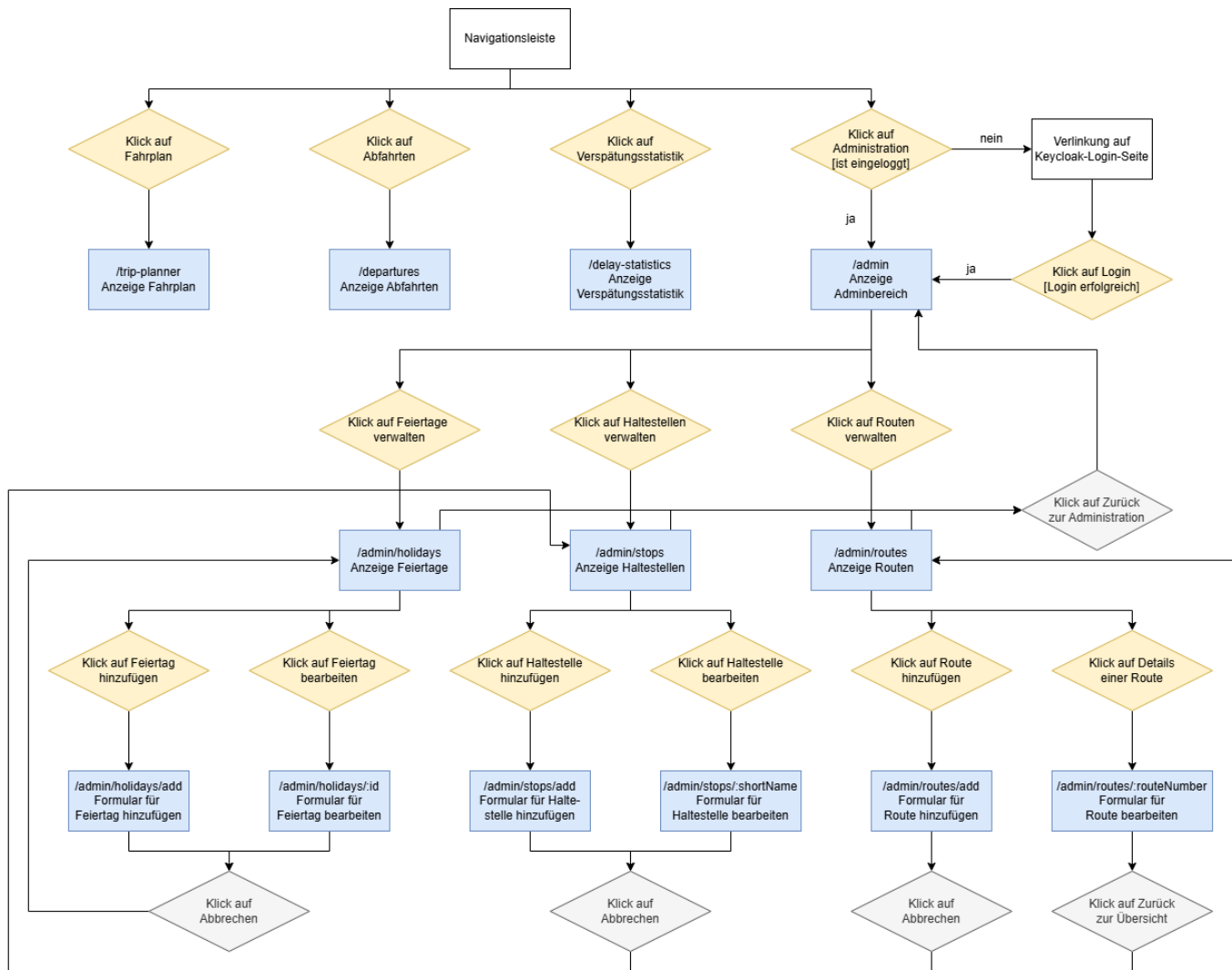
Validators (Directives)

Direktiven wie z.B. die `DateRangeValidatorDirective` validieren Benutzereingaben, z. B. ob das Enddatum nach dem Startdatum liegt, und stellen so sicher, dass die Eingabedaten korrekt sind. Ich habe für das Projekt ein großes Augenmerk auf Input-Validation gelegt.

ErrorMessages

Bündelt alle spezifischen Fehlermeldungen für Formulare wie `HolidayForm`, `StopForm`, und `RouteForm`, um konsistente und benutzerfreundliche Validierungsnachrichten anzuzeigen.

Navigationswege-Diagramm



Testlauf

Szenario 1: Verkehrsunternehmen melden sich am System an, um die Pflege des neuen Fahrplans durchzuführen

Fahrplan

Abfahrten

Verspätungsstatistik

Administration

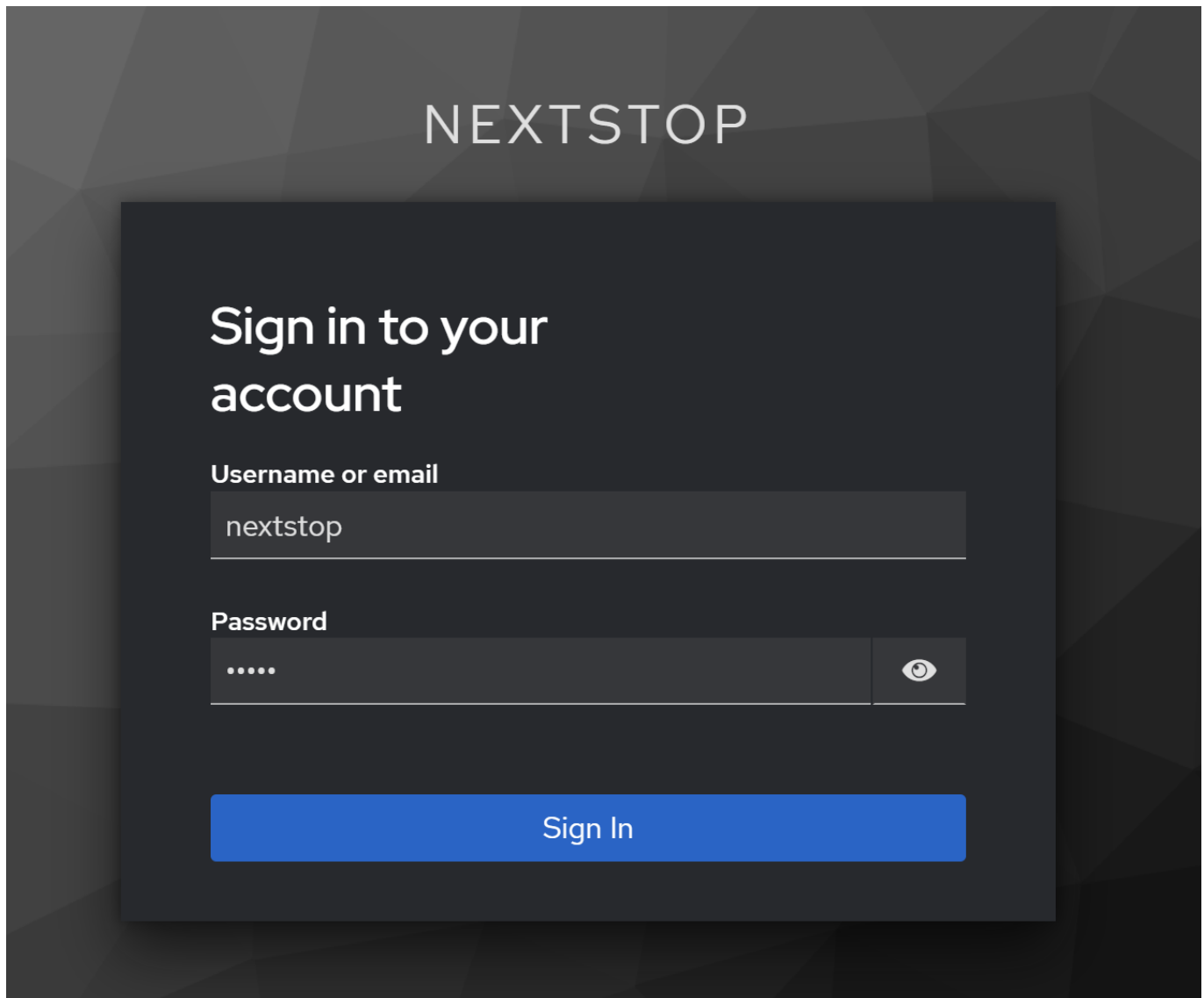
Login

Anmeldung als Admin, um Verwaltungs-Funktionen nutzen zu können

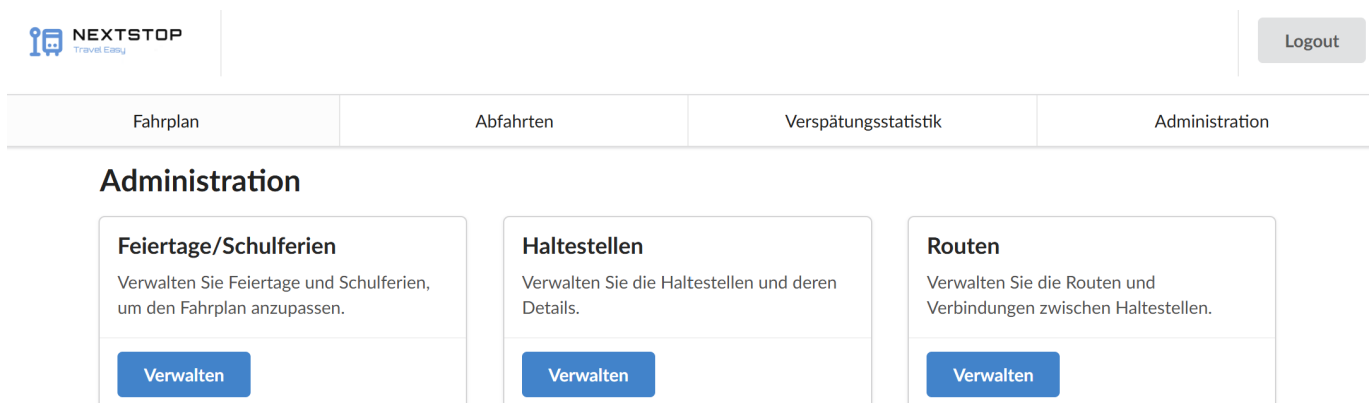
Login mit Keycloak

Klick auf den **Login**-Button auf der Hauptseite rechts oben oder in der Navbar auf den Admin-Bereich. Man wird auf das Anmeldefenster von Keycloak weitergeleitet.

Hinweis: Ein eingebauter Guard prüft, ob der Benutzer authentifiziert ist, um die Admin-Funktionen nutzen zu dürfen.



Den Anmeldevorgang übernimmt Keycloak. Bei Eingabe der gültigen Benutzer/Admindaten wird der Login erfolgreich durchgeführt, man gelangt zur eigentlichen Webseite zurück und das Access/Bearer Token wird gespeichert. Bei ungültiger Eingabe erfolgt eine Fehlermeldung von Keycloak.



Der Guard schaltet den Adminbereich jetzt frei, weil dieser eine gültige Authentifizierung entgegennimmt und alle Admin-funktionen können jetzt genutzt werden.

Szenario 2: Feiertage und Schulferien für 2025 werden verwaltet, um den Fahrplan davon abhängig machen zu können

Feiertage/Schulferien

Verwalten Sie Feiertage und Schulferien, um den Fahrplan anzupassen.

Verwalten



Klick auf den **Verwalten**-Button im Adminbereich.

Feiertage und Schulferien

← Zurück zur Administration

Feiertag hinzufügen

Bezeichnung	Startdatum	Enddatum	Schulferien	Aktionen
New Year	2024-01-01	2024-01-01		<div>BearbeitenLöschen</div>
Winter Break	2024-02-12	2024-02-16	✓	<div>BearbeitenLöschen</div>
Good Friday	2024-03-29	2024-03-29		<div>BearbeitenLöschen</div>
Easter Monday	2024-04-01	2024-04-01		<div>BearbeitenLöschen</div>
Spring Break	2024-04-02	2024-04-12	✓	<div>BearbeitenLöschen</div>
Labor Day	2024-05-01	2024-05-01		<div>BearbeitenLöschen</div>
Ascension Day	2024-05-09	2024-05-09		<div>BearbeitenLöschen</div>
Whit Mondav	2024-05-20	2024-05-20		<div>BearbeitenLöschen</div>

Hier sieht man alle Feiertage inklusive Kennzeichnung, ob es sich um Schulferien handelt, in einer Tabelle nach Startdatum sortiert aufgelistet. Neben der Abfrage der Feiertage kann man auch über Buttons Feiertage neu hinzufügen, bestehende bearbeiten oder löschen. Genauso gelangt man mit dem Button **Zurück zur Administration** zur Adminhauptseite retour.

Feiertag hinzufügen

Bezeichnung

Bezeichnung darf nicht leer sein.

Startdatum



Das Startdatum ist erforderlich.

Enddatum



Das Enddatum ist erforderlich.

☐ Schulferien

Hinzufügen

Abbrechen

Klickt man auf den Button **Feiertag hinzufügen**, kann man in einem Formular die Daten für den neuen Feiertag eingeben. Es wird besonders auf Eingabevalidierung geachtet: Es benötigt eine gültige Bezeichnung, Startdatum und Enddatum.

Startdatum

Enddatum

Enddatum muss nach dem Startdatum liegen.

Zudem wird über eine eingebundene Directive geprüft, ob das eingegebene Startdatum kleiner ist als das Enddatum, ansonsten wird auch ein Fehler angezeigt. Erst bei gültigen Eingaben wird der **Hinzufügen**-Button enabled und der Feiertag zur Tabelle hinzugefügt.

Feiertag bearbeiten

Bezeichnung

NewHoliday

Startdatum

01.01.2025



Enddatum

10.01.2025



☐ Schulferien

Speichern

Abbrechen

Klickt man auf den Button **Bearbeiten** eines Feiertags, werden die bislang gespeicherten Daten dem User angezeigt und dieser kann sie entsprechend abändern. Ansonsten wird genau dieselbe Formalkomponente wie beim Create verwendet.

Szenario 3: Haltestellen können angelegt und bearbeitet werden

Haltestellen

Verwalten Sie die Haltestellen und deren Details.

Verwalten



Klick auf den **Verwalten**-Button im Adminbereich.

Haltestellen

[← Zurück zur Administration](#)

Haltestelle hinzufügen

Bezeichnung	Kurzbezeichnung	GPS-Breitengrad	GPS-Längengrad	Aktionen
Bruck Central	BCK	47.4097	15.2685	Bearbeiten Löschen
Ebelsberg	EBSD	48.2397	14.3355	Bearbeiten Löschen
FH Hagenberg	FHHGB	48.3683	14.5156	Bearbeiten Löschen
Gmunden Main	GMND	47.9182	13.8007	Bearbeiten Löschen
Gössendorf	GSSZW	47.01	15.4911	Bearbeiten Löschen
Graz Central Station	HBFGHZ	47.0811	15.4088	Bearbeiten Löschen
Linz Central Station	HREINZ	48.2915	14.2908	Bearbeiten Löschen

Hier sieht man alle Haltestellendaten wieder in Tabellenform. Neben der Abfrage der Haltestellen kann man hier auch wieder über Buttons Haltestellen neu hinzufügen, bestehende bearbeiten oder löschen. Genauso gelangt man mit dem Button [Zurück zur Administration](#) ebenfalls zur Adminhauptseite retour.

Haltestelle hinzufügen

Bezeichnung

Bezeichnung

Bezeichnung darf nicht leer sein.

Kurzbezeichnung

Kurzbezeichnung

Kurzbezeichnung darf nicht leer sein.

Breitengrad

Breitengrad

Breitengrad ist erforderlich.

Längengrad

Längengrad

Längengrad ist erforderlich.

Hinzufügen

Abbrechen

Klickt man auf den Button [Haltestelle hinzufügen](#), kann man in einem Formular die Daten für die neue Haltestelle eingeben. Es wird erneut besonders auf Eingabevalidierung geachtet: Es benötigt eine gültige Bezeichnung, Kurzbezeichnung, Breiten- und Längengrad für die GPS-Koordinaten.

Kurzbezeichnung

FHHGB

Die Kurzbezeichnung für diese Haltestelle ist bereits vergeben.

Kurzbezeichnung

NEW *

Kurzbezeichnung darf nur Buchstaben und Zahlen enthalten.

Bei der Eingabe der Kurzbezeichnung wird geprüft, ob diese nicht bereits vorhanden ist bzw. keine Sonder- und Leerzeichen enthält.

Breitengrad

1000

Breitengrad muss eine gültige Dezimalzahl von -90.0 bis 90.0 (mit maximal 14 Nachkommastellen) sein.

Längengrad

12.111111111111111111111111

Längengrad muss eine gültige Dezimalzahl von -180.0 bis 180.0 (mit maximal 14 Nachkommastellen) sein.

Bei der Eingabe der GPS-Koordinaten habe ich auch noch zusätzliche Eingabevalidierung hinzugefügt.

Haltestelle bearbeiten

Bezeichnung

FH Hagenberg

Kurzbezeichnung

FHHGB

Breitengrad

48.3683

Längengrad

14.5156

Speichern

Abbrechen

Klickt man auf den Button **Bearbeiten** einer Haltestelle, werden die bislang gespeicherten Daten dem User wieder angezeigt und dieser kann sie entsprechend abändern. Ansonsten wird genau dieselbe Formalkomponente wie beim Create verwendet. Zusätzlich kann beim Update die Kurzbezeichnung nicht verändert werden, weil diese eindeutig ist und in der Datenbank als Primary-Key bei uns hinterlegt ist.

Szenario 4: Routen können inkl. Abfahrtszeiten erstellt werden

Routen

Verwalten Sie die Routen und Verbindungen zwischen Haltestellen.

Verwalten



Klick auf den **Verwalten**-Button im Adminbereich.

Routen

Route hinzufügen

← Zurück zur Administration

Routennummer	Start - Endhaltestelle	Gültigkeitszeitraum	Tagesgültigkeit	Aktionen
101	FHHGB - STCLNZ	01.01.2024 - 31.12.2024	Alle Tage	<div>Details</div>
102	MLA - TRTGWN	01.01.2024 - 31.12.2024	Alle Tage	<div>Details</div>
103	ZELL - LSN	01.01.2024 - 31.12.2024	Alle Tage	<div>Details</div>
104	TRTGWN - LSN	01.01.2024 - 31.12.2024	Wochenende	<div>Details</div>

Hier sieht man alle Routendaten inklusive Routennummer, Start- und Endhaltestelle der Route und Gültigkeitsbereich ebenfalls in Tabellenform. Neben der Abfrage der Routen kann man hier über Buttons Routen neu hinzufügen und bestehende genauer einsehen(Update und Delete wurde lt. Aufgabenstellung nicht eingebaut).Genauso gelangt man mit dem Button Zurück zur Administration ebenfalls zur Adminhauptseite retour.

Route mit Haltestellen hinzufügen

Routeninformationen

Gültig ab

TT.mm.jjjj

Das Startdatum ist erforderlich.

Gültig bis

TT.mm.jjjj

Das Enddatum ist erforderlich.

Tagesgültigkeit

Alle Tage

Haltestelleninformationen

Haltestelle

Suche Haltestellen...

Haltestelle hinzufügen

Abfahrtszeit

06:00

Hinweis

Noch keine Haltestellen hinzugefügt

Speichern

Abbrechen

Klickt man auf den Button Route hinzufügen, kann man in einem Formular die Daten für die neue Route und deren angefahrenen Haltestellen eingeben. Es wird erneut besonders auf Eingabevalidierung geachtet: Es

14 / 28

benötigt ein gültiges Start- und Enddatum wieder(auch hier wurde dieselbe Directive für die Überprüfung Startdatum < Enddatum eingebunden) und zumindest eine hinzugefügte Haltestelle für die neue Route.

Haltestelle

li

Q

Abfahr

06:C

Linz Central Station (HBFLNZ)

Linzerstraße Berufsschule (LZBS)

Linzer Straße Jaunitzsiedlung (LZJS)

Mödling (MDLK)

City Square Linz (STC)

Main Square Linz (STCLNZ)

Durch eingabe im Suchfeld werden mögliche Haltestellen nach ihrem Namen gefunden, welche der User auswählen kann.

Haltestelleninformationen

Haltestelle



Abfahrtszeit

06:00

Haltestelle hinzufügen

Hinzugefügte Haltestellen

- 06:00 - FR **Entfernen**
- 06:30 - FHHGB **Entfernen**
- 07:00 - HBFLNZ **Entfernen**

Mit Eingabe der Abfahrtszeit kann der User die Haltestelle für die Route zwischenzeitlich hinzufügen und aber auch dort direkt wieder entfernen falls diese doch nicht passt. Mit Klick auf **Speichern** letztendlich wird die Route mit allen angegebenen Haltestellen in einem Request angelegt.


Szenario 6: Fahrplanabfragen können gemacht werden

beinhaltet Szenario 5: Haltestellen können gesucht werden, z. B. nach dem Namen oder anhand des aktuellen Standorts

Fahrplan	Abfahrten	Verspätungsstatistik	Administration
----------	-----------	----------------------	----------------


Verbindung suchen

Start-Haltestelle




Standortbasierte Suche


Ziel-Haltestelle



Standortbasierte Suche

Datum **Uhrzeit**






☐ Nach Ankunftszeit suchen

Anzahl der Verbindungen

Suchen

Im Menüpunkt **Fahrplan** kann der User Fahrpläne abfragen. Dazu muss er zunächst Start- und Zielhaltestelle eingeben. Die Haltestellenauswahl kann entweder durch Sucheingabe des Namens (die Suchkomponente ist die gleiche wie bereits bei den Routen im Adminbereich) oder über **Standortbasierte Suche** erfolgen (Szenario 5 jetzt). Weiters benötigt wird Datum und Uhrzeit, Auswahl ob Abfahrts- oder Ankunftszeit gemeint ist und welche maximale Anzahl an Verbindungen ausgegeben werden soll.

Start-Haltestelle



Sankt Florian (SFLO)

Sankt Pölten Main (SRBG)

Suche Haltestellen...

Bei der namensbasierten Haltestellensuche kann der User Suchbegriffe direkt im Suchfeld eingeben. Auch Haltestellen mit unterschiedlichen Schreibweisen wie hier Sankt oder St. werden identifiziert.

Start-Haltestelle

Suche Haltestellen...

Standortbasierte Suche

Standortbasierte Suche

Haltestellen in der Nähe

Aktueller Standort: Breite: 48.511300, Länge: 14.505000

Maximale Entfernung (km)

20

Maximale Ergebnisse

10

Haltestellen suchen

Gefundene Haltestellen

Freistadt Stifterplatz (FR)

Breite: 48.510000, Länge: 14.510000

Linzerstraße Berufsschule (LZBS)

Breite: 48.501207, Länge: 14.502052

Neben der namensbasierten Suche kann der User durch Klick auf den Button **Standortbasierte Suche** die nächstgelegenen Haltestellen von seinem Standort aus finden. Dazu öffnet sich ein Dialogfenster, wo Dateneingaben wie maximale Entfernung bzw. Ergebnisse eingegeben werden können. Bei erfolgreicher Suche kann der User die gewünschte Haltestelle auswählen, der Dialog schließt sich und die Haltestelle wird übernommen(entweder als Start- oder halt Zielhaltestelle, je nachdem wo der User den Button klickt).

Suche Haltestellen...

Ausgewählte Haltestelle: Freistadt Stifterplatz (FR)

Standortbasierte Suche

Ziel-Haltestelle

Suche Haltestellen...

Ausgewählte Haltestelle: Linz Central Station (HBFLNZ)

Standortbasierte Suche

Datum

18.01.2025

Uhrzeit

00:49

☐ Nach Ankunftszeit suchen

Anzahl der Verbindungen

3

Suchen

Gefundene Verbindungen

Route	Abfahrt	Ankunft	Umstiege	Verspätung
106	06:00 Freistadt Stifterplatz	07:00 Linz Central Station	✓ Direkt	🕒 Pünktlich

Bei gefundenen Verbindungen werden diese dem User in tabellarischer Form wieder repräsentiert. Das Backend berücksichtigt Verbindungen mit 0 oder 1 Umstieg und die Tagesgültigkeit. Auch die Verspätung wird berücksichtigt, diese wird auch frontendmäßig angezeigt - siehe Szenario 9 dann genauer.

Hinweis
Keine Verbindungen gefunden.

Bei keinen gefundenen Verbindungen wird eine entsprechende Hinweismeldung ausgegeben.

Szenario 7: Anzeigetafeln zeigen die nächsten Abfahrten für die aktuelle Haltestelle

Fahrplan

Abfahrten

Verspätungsstatistik

Administration

Abfahrten anzeigen

Haltestelle

Suche Haltestellen...

Standortbasierte Suche

Datum

18.01.2025

Uhrzeit

01:08

Anzahl der Verbindungen

3

Abfahrten anzeigen

Im Menüpunkt **Abfahrten** kann der User die nächsten Abfahrten von der gewünschten Haltestelle aus finden. Diese Komponente ist sehr ähnlich wie die Fahrplan-Komponente aufgebaut. Die Haltestelle kann wieder über den Namen oder standortbasiert gesucht und ausgewählt werden. Ebenso können Datum/Uhrzeit und die Anzahl an Verbindungen ausgewählt werden.

Abfahrten anzeigen

Haltestelle

Suche Haltestellen...

Ausgewählte Haltestelle: Linz Central Station (HBFLNZ)

Standortbasierte Suche

Datum

18.01.2024

Uhrzeit

01:19

Anzahl der Verbindungen

6

Abfahrten anzeigen

Nächste Abfahrten

Route	Abfahrtszeit	Ziel	Verspätung
101	06:30	Main Square Linz	🕒 Pünktlich
102	07:00	Triesting-Günsser	🕒 Pünktlich
103	07:00	Leoben Süd	🕒 Pünktlich

Die nächsten gefundenen Abfahrten inkl. Zielhaltestelle werden dann dem User wieder in Tabellenform angezeigt. Bei keiner gefundenen Verbindung von der Abfahrtshaltestelle aus wird dem User wieder eine Hinweismeldung angezeigt.

Szenario 8: Busse checken ihren aktuellen Standort ein

Das hat im Frontend lt. Anforderungen nicht implementiert werden müssen. Aber für die Berücksichtigung der Verspätungen in der Fahrplan- bzw. Abfahrtskomponente als auch dann in der Verspätungsstatistik wird ein CheckIn zur Demonstration manuell durchgeführt.

Checkin wird manuell durchgeführt für:

- Route: 106 (Haltestellen: FR - FHHGB - HBFLNZ)
- CheckIn-Haltestelle: Freistadt Stifterplatz (FR)
- DateTime: 21.01.2025 06:10 (10 Minuten verspätet)

Szenario 9: Verspätungen werden in der Suche und auf der Anzeigetafel berücksichtigt

Der zuvor erstellte CheckIn wird jetzt berücksichtigt sowohl in der Fahrplansuche und in der Anzeigetafel berücksichtigt.

Suche Haltestellen...

Ausgewählte Haltestelle: FH Hagenberg (FHHGB)

Standortbasierte Suche

Ziel-Haltestelle

Suche Haltestellen...

Ausgewählte Haltestelle: Linz Central Station (HBFLNZ)

Standortbasierte Suche

Datum

21.01.2025

Uhrzeit

06:35

Nach Ankunftszeit suchen

Anzahl der Verbindungen

3

Suchen

Gefundene Verbindungen

Route	Abfahrt	Ankunft	Umstiege	Verspätung
106	06:30 (06:40) FH Hagenberg	07:00 (07:10) Linz Central Station	✓ Direkt	⌚ +10 Min

Hier wird eine Fahrplansuche für die Starthaltestelle FH Hagenberg (FHHGB) und Zielhaltestelle Hauptbahnhof Linz (HBFLNZ) am 21.01.2025 mit Abfahrtszeit 06:35 erstellt. Normalerweise würde der Bus regulär um 06:30 bereits in Hagenberg eintreffen, aber die Verspätung in Höhe von 10 Minuten von der vorherigen Haltestelle Freistadt Stifterplatz (FR) wird für alle nachkommenden Haltestellen in der Route berücksichtigt. Daher kommt der Bus erst um 06:40 in Hagenberg an und der User kann diesen Bus noch erwischen.

Verbindung suchen

Start-Haltestelle

Suche Haltestellen...

Ausgewählte Haltestelle: FH Hagenberg (FHHGB)

Standortbasierte Suche

Ziel-Haltestelle

Suche Haltestellen...

Ausgewählte Haltestelle: Linz Central Station (HBFLNZ)

Standortbasierte Suche

Datum

21.01.2025

Uhrzeit

07:05

Nach Ankunftszeit suchen

Anzahl der Verbindungen

3

Suchen

Hinweis

Keine Verbindungen gefunden.

Umgekehrt, wenn der User Ankunftszeit auswählt, funktioniert es genauso. Hier möchte der User um 07:05 in Linz spätestens ankommen. Normalerweise würde der Bus um 07:00 in Linz eintreffen aber durch die Verspätung von 10 Minuten schafft der Bus das nicht und deswegen wird die Verbindung nicht angezeigt. Gäbe es diese Verspätung nicht, würde er um 07:00 pünktlich in Linz ankommen und die Verbindung würde angezeigt werden.

Abfahrten anzeigen

Haltestelle

Suche Haltestellen...

Ausgewählte Haltestelle: FH Hagenberg (FHHGB)

Standortbasierte Suche

Datum

21.01.2025

Uhrzeit

06:40

Anzahl der Verbindungen

3

Abfahrten anzeigen

Nächste Abfahrten

Route	Abfahrtszeit	Ziel	Verspätung
106	06:30 (06:40)	Linz Central Station	+10 Min

Auf der Anzeigetafel (also bei den Abfahrten) funktioniert die Berücksichtigung der Verspätung genauso. Hier könnte der User den Bus genau noch um 06:40 erwischen.

beinhaltet Szenario 11: Darstellung der Verspätungsstatistik in Diagrammen

Im Menüpunkt **Verspätungsstatistik** kann der User sich einen Einblick über Verspätungen für bestimmte Routen und Zeiträume machen.

Fahrplan

Abfahrten

Verspätungsstatistik

Administration

Verspätungsstatistik

Startdatum

01.01.2024

Enddatum

31.12.2024

Routennummer (optional)

Alle Routen

Statistik abrufen

Diagramm: Verspätungsstatistik



Hier wird dem User eine Verspätungsstatistik für den Auswertungszeitraum des Jahres 2024 für alle Routen angezeigt. Die Auswertungsstatistik wurde mithilfe der Libraries `chart.js` und `ng2chart` erstellt.

Verspätungsstatistik

Startdatum

01.01.2024

Enddatum

31.01.2024

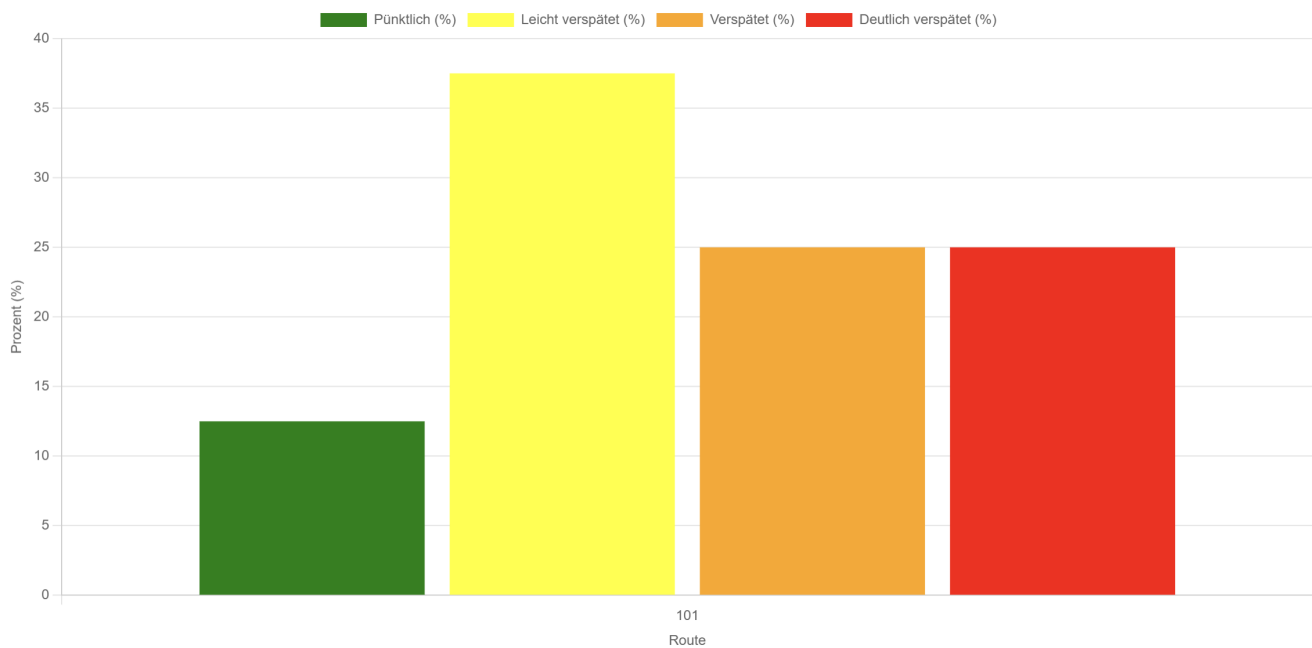
Routennummer (optional)

Route 101

Statistik abrufen

Diagramm: Verspätungsstatistik

Auswertungszeitraum: 01.01.2024 bis 31.01.2024



Hier wird die Statistik für den Zeitraum für ein Monat (Jänner 2024) für die Route 101 angezeigt.

Fragen beantworten

Was ist zu tun, wenn sich URLs ändern? Wie invasiv ist der Eingriff in Ihre Anwendung um diese zu ändern?

Wenn sich URLs ändern kann man die Routen sehr einfach in [app.routes.ts](#) abändern. Wenn die Route zudem bei einem Button z.B. angegeben wurde muss diese auch dort angepasst werden.

Z.B. die Admin-Routen sind so definiert in [app.routes.ts](#):

```
{
  path: 'admin',
  component: AdminComponent,
  canActivate: [canNavigateToAdminGuard],
  children: [
    { path: 'holidays', component: HolidaysComponent },
    { path: 'holidays/add', component: HolidayFormComponent },
    { path: 'holidays/:id', component: HolidayFormComponent },
    // ...
  ]
}
```

Ändert man z.B. die Route um einen Feiertag hinzufügen zu wollen, müsste man die Route in meinem Fall für den Button in der HolidaysComponent anpassen, welcher auf das Formular zum Anlegen eines neuen Feiertages weiter navigiert, siehe:


```
<button class="ui primary button" routerLink="/admin/holidays/add">Feiertag  
hinzufügen</button>
```

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

Ich schütze die Admin-Komponente und all deren Kindelemente(wie die Verwaltung der Feiertage, Haltestellen und Routen) mit dem Guard [can-navigate-to-admin.guard.ts](#).

```
import { inject } from '@angular/core';  
import { CanActivateFn, Router } from '@angular/router';  
import { AuthenticationService } from '../services/authentication.service';  
import { map, tap } from 'rxjs';  
  
export const canNavigateToAdminGuard: CanActivateFn = (route, state) => {  
  const auth = inject(AuthenticationService)  
  const router = inject(Router);  
  
  return auth.loadAuthState().pipe(  
    tap((isAuthenticated) => {  
      if (!isAuthenticated) {  
        router.navigate(['/login'], {  
          queryParams: { returnUrl: state.url },  
        });  
      }  
    }),  
    map((isAuthenticated) => isAuthenticated)  
  );  
};
```

Hier nochmal der Admin-Path in [app.routes.ts](#):

```
{  
  path: 'admin',  
  component: AdminComponent,  
  canActivate:[canNavigateToAdminGuard],  
  children: [  
    { path: 'holidays', component: HolidaysComponent },  
    { path: 'holidays/add', component: HolidayFormComponent },  
    { path: 'holidays/:id', component: HolidayFormComponent },  
    { path: 'stops', component: StopsComponent },  
    { path: 'stops/add', component: StopFormComponent },  
    { path: 'stops/:shortName', component: StopFormComponent },  
    { path: 'routes', component: RoutesComponent },  
    { path: 'routes/add', component: RouteFormComponent },  
    { path: 'routes/:routeNumber', component: RouteDetailsComponent },  
  ]  
}
```

Die Routen sind erst zugänglich, wenn sich der User ordentlich eingeloggt hat. Damit aktiviert der Guard die Admin-Komponente und alle darunterliegenden Kindelemente bzw. deren Routen.

Wie stellen Sie eine korrekte Dateneingabe sicher?

Wie bereits bei den ganzen Testfällen erwähnt habe ich sehr großes Augenmerk auf Input-Validierung gelegt.

Für die Erstellung der Formulare habe ich Reactive-Forms bzw. deren Validator-Klasse verwendet. Und damit dem User entsprechende Fehlermeldungen angezeigt werden, wenn die eingegebenen Daten nicht stimmen, habe ich die meisten Error-Messages in der eigenen Helper-Class [error-message.ts](#) ausgelagert. Diese Error-Messages habe ich dann entsprechend im Formular bei den Feldern binden können.

```
export class ErrorMessage {
  constructor(
    public forControl: string,
    public forValidator: string,
    public text: string
  ) { }
}

export const HolidayFormErrorMessages = [
  new ErrorMessage('name', 'required', 'Bezeichnung darf nicht leer sein.'),
  new ErrorMessage('startDate', 'required', 'Das Startdatum ist erforderlich.'),
  new ErrorMessage('endDate', 'required', 'Das Enddatum ist erforderlich.'),
];

export const StopFormErrorMessages = [
  new ErrorMessage('name', 'required', 'Bezeichnung darf nicht leer sein.'),
  new ErrorMessage('shortName', 'required', 'Kurzbezeichnung darf nicht leer sein.'),
  new ErrorMessage('shortName', 'shortNameNotUnique', 'Die Kurzbezeichnung für diese Haltestelle ist bereits vergeben.'),
  new ErrorMessage('shortName', 'pattern', 'Kurzbezeichnung darf nur Buchstaben und Zahlen enthalten.'),
  new ErrorMessage('latitude', 'required', 'Breitengrad ist erforderlich.'),
  new ErrorMessage('latitude', 'pattern', 'Breitengrad muss eine gültige Dezimalzahl von -90.0 bis 90.0 (mit maximal 14 Nachkommastellen) sein.'),
  new ErrorMessage('longitude', 'required', 'Längengrad ist erforderlich.'),
  new ErrorMessage('longitude', 'pattern', 'Längengrad muss eine gültige Dezimalzahl von -180.0 bis 180.0 (mit maximal 14 Nachkommastellen) sein.'),
];

// ...
```

Ebenso habe ich zusätzlich ein paar Directives noch erstellt wie z.B. die [Date-Range-Validator-Directive](#) (Startdatum kleiner Enddatum) oder die [ShortName-Validator-Directive](#) (welche z.B. bei Eingabe der Kurzbezeichnung beim Erstellen einer Haltestelle im Formular prüft, ob diese Haltestelle bereits existiert -> muss unique sein).

```

import { Directive } from '@angular/core';
import { AbstractControl, NG_VALIDATORS, ValidationErrors, Validator, ValidatorFn
} from '@angular/forms';

@Directive({
  selector: '[wea5DateRangeValidator]',
  standalone: true,
  providers: [
    {
      provide: NG_VALIDATORS,
      useExisting: DateRangeValidatorDirective,
      multi: true,
    },
  ],
})
export class DateRangeValidatorDirective implements Validator {
  constructor() {}

  validate(control: AbstractControl): ValidationErrors | null {
    return dateRangeValidator()(control);
  }
}

export function dateRangeValidator(): ValidatorFn {
  return (control: AbstractControl): ValidationErrors | null => {
    const startDate = control.get('startDate')?.value;
    const endDate = control.get('endDate')?.value;

    if (startDate && endDate && new Date(startDate) > new Date(endDate)) {
      return { invalidDateRange: true };
    }
    return null;
  };
}

```

Was passiert, wenn Aufrufe an das Backend Fehler produzieren?

Ein möglicher Fehler wird jedes Mal über die Hilfs-Klasse [ErrorHandlerService](#) für die jeweiligen Services abgefangen und dem entsprechenden ErrorHandler übergeben. In den Komponenten werden dann bei Bedarf bestimmte ErrorCodes behandelt und eine entsprechende Nachricht angezeigt.

Siehe ErrorHandlerService-Klasse:

```

import { Injectable } from '@angular/core';
import { Observable, of } from 'rxjs';

@Injectable({
  providedIn: 'root',
})
export class ErrorHandlerService {

```

```
handle<T>(error: Error | any): Observable<T> {  
  console.error('Service error:', error);  
  return of(null as T);  
}  
}
```

Setup

1. Docker-Container starten:

- MySQL-Datenbank für die Speicherung.
- *(Optional)* Container für Testdaten.
- Keycloak-Service für die Authentifizierung.

2. Backend starten:

- **.NET REST API** als Schnittstelle zwischen Frontend und Backend.

3. Frontend starten:

- Angular (v18) als Benutzeroberfläche.

Externe Teile

1. Fomantic-UI

- Wurde fast ausschließlich für das komplette Styling der Webseite verwendet.

2. Keycloak-Einbindung

- Integriert nach Anleitung aus dem eLearning.

3. **chart.js** und **ng2-charts**

- Genutzt für die Darstellung des Verspätungsdiagramms.