

Meilenstein 3

Firma: Doubleslash

Projekt: Fuhrparkmanagement System

Betreuer: Andreas Heinrich

Kunde: Johannes Mayer


Team: Leonard Deininger, Okan Kizilagil, Cedric Schaaf, Nick Habermann, Edwin Starz

Inhaltsverzeichnis

Zielgruppe, Problem, Eigenschaften.....	1
UI Entwürfe.....	3
Architektur / Technologie.....	4
Softwarearchitektur.....	6
Struktursicht.....	6
Verteilungssicht.....	6
Verhaltenssicht.....	7
Verhaltenssicht.....	7
Projektmanagement.....	8
User Stories & Aufwandsschätzung.....	9
Schnittstellentechnologie.....	11
Fahrzeugverwaltungs-Schnittstelle.....	11
Reservations-Schnittstelle.....	14
Fahrten-Schnittstelle.....	16
Statistik-Schnittstelle.....	20
Übersicht Endpunkte (Swagger UI).....	21
Externer User Management Service (von doubleSlash).....	22
Codegenerierung zu Rest API Schnittstellen mit Swagger.....	24
Datenbankmodell (logisch).....	25
Datenbankmodell (physisch).....	26
CI/CD Pipeline.....	27
Backend-Workflow (mit Gradle).....	27
Frontend-Workflow (mit npm).....	27
Technischer Prototyp.....	28
Login-Ansicht.....	28
Kalenderansicht.....	29
Sidebar-Ansicht.....	29
Auto anlegen.....	30
Fahrt anlegen.....	30

Zielgruppe, Problem, Eigenschaften

Die Zielgruppe sind alle Mitarbeiter der Firma doubleSlash an den Standorten Stuttgart, Karlsruhe, München und Friedrichshafen.



Jörg Baier ♂


BILDUNG: Bachelor-Abschluss in Informatik
JOB-TITEL: Softwareentwickler bei doubleSlash
ORT: Stuttgart
ALTER: 32

BESCHREIBUNG:

- Jörg ist sportlich aktiv und legt großen Wert auf Fitness und Gesundheit. Er betreibt regelmäßig Sport, sei es Laufen, Radfahren, oder Fußball, um körperlich fit zu bleiben.
- Sein Beruf erfordert viele Meetings und Zusammenarbeiten mit Teams an verschiedenen Standorten. Dies bedeutet, dass er häufig mit dem Firmen-Auto reisen muss, um persönlich an Besprechungen teilzunehmen.

NUTZER ZIELE:

- Er sucht eine zuverlässige Web-Anwendung, um die Firmen internen Fahrzeuge zu buchen, damit er konfliktfrei seine Meetings und Kundentermine an verschiedenen Standorten erreichen kann.



Lea Weiß ♀

BILDUNG: Abitur
JOB-TITEL: Ausbildung – Fachinformatikerin bei doubleSlash
ORT: Karlsruhe
ALTER: 26

BESCHREIBUNG:

- Lea ist leidenschaftlich an der Softwareentwicklung interessiert und möchte ihre Fähigkeiten und Kenntnisse in diesem Bereich vertiefen.
- Sie liebt es, mit ihren Freunden zu reisen und diese Reisen zu organisieren

NUTZER ZIELE:

- Lea Organisiert in der Firma Events und Ausflüge, damit sie besser Planen kann, benötigt Sie ein Managementsystem, um die Fahrzeuge rechtzeitig zu reservieren.

Problem: Buchung der Fuhrpark Fahrzeuge ist nicht übersichtlich und nicht zentral mit einer Plattform, in der Vergangenheit wurden Fahrzeuge doppelt verbucht, da zum Teil über Outlook und Confluence parallel gebucht wurde.

Eigenschaften

- **Responsive Webanwendung:** Die Benutzeroberfläche sollte so konzipiert sein, dass verschiedene Geräte und Bildschirmgrößen optimal funktionieren. So ist es auch möglich, dass man diese auch von unterwegs nutzen kann.
- **Zentrale Plattform:** Das System schafft eine zentrale Plattform zur Buchung und Verwaltung von Firmenfahrzeugen. Damit sind auch doppelte Buchungen und sonstige Unstimmigkeiten nicht mehr möglich. Alle Daten sind an einem logischen Ort und nicht mehr verteilt.
- **Statistikerfassung:** Das System erfasst statistische Daten über die Nutzung der Fahrzeuge, um unter anderem die Ressourcenplanung und -verwaltung zu verbessern.

- **Erweiterbarkeit:** Das System sollte mit dem Aspekt der Erweiterbarkeit entwickelt werden, sodass es auch nach Abschluss des Projekts zu erweiterten Anforderungen noch von einer externen Person weiterentwickelt werden könnte.
- **Container-Software-Kompatibilität:** Das System kann mittels Container-Software wie Docker virtualisiert werden, um eine effizientere Ausführung (und ggfs. Skalierbarkeit) zu gewährleisten.

UI Entwürfe

Um eine Idee zu bekommen, wie die Anwendung aussehen könnte, und worauf wir achten müssen, haben wir eine einfach gehaltene UI Prototyp Skizze erstellt.

Hand-drawn UI prototype for the login page of 'FPM' (Fahrplanmanagement). The browser address bar shows 'fpm.fahrplanmanagement.de'. The page has a header with a hamburger menu and the 'FPM' logo. The main content area contains a stick figure icon, a 'Benutzername:' input field, a 'Kennwort:' input field, and a red link 'Passwort vergessen?' below the password field.

Hand-drawn UI prototype for the main dashboard of 'FPM' (Fahrplanmanagement). The browser address bar shows 'fpm.fahrplanmanagement.de'. The page has a header with a hamburger menu, the 'FPM' logo, and an 'OK' button. The main content area is titled 'Startseite' and is divided into two main sections: 'Neue Fahrt Buchen' on the left and 'Übersicht aktueller Fahrten' on the right. The left section has a car icon and buttons for 'Aktuelle Buchung', 'Buchung Stornieren', 'Mitfahrte Buchen', 'Letzte Fahrten', and 'Ranking'. The right section shows a table with columns for 'Von:', 'Nach:', 'Zeitraum:', 'Auto:', and 'Mitfahrer:', followed by a list of circular buttons labeled 'ES', 'LD', 'UH', 'CC', and 'OK'.

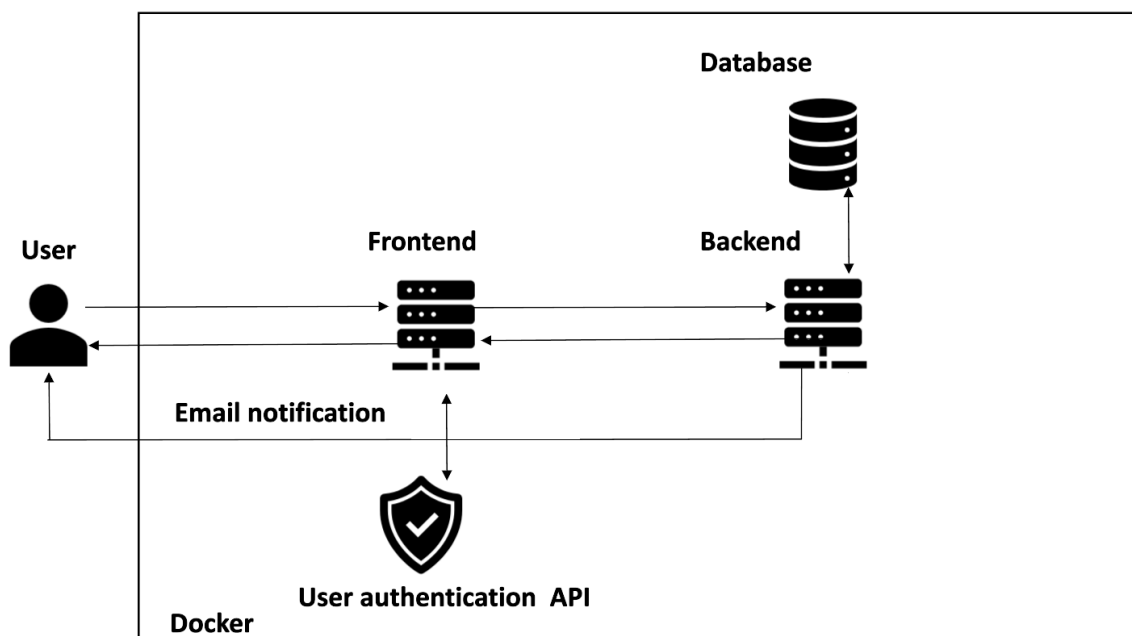
Architektur / Technologie

Konkrete Vorgaben, welche Technologien wir zu verwenden haben, wurden von der Firma doubleSlash nicht gestellt, jedoch sind Spring Boot und Angular erwünscht.

Die anderen Architektur-/Technologieentscheidungen, wie z.B. Git, Docker etc. haben wir vor allem an Kriterien wie Einsatzzweck, Praxisbewährung oder auch Erlernbarkeit entschieden. Wir haben uns auch über die Lizenzen der einzelnen Technologien informiert und überprüft, ob diese die kommerzielle Nutzung oder Veränderungen zulassen.

- Spring Boot als Backend Framework
 - Gradle als Build-Management-Automatisierungs-Tool
 - Lizenz: [Apache 2.0](#)
 - Incremental Builds, Sehr flexibel, Tasks lassen sich einfach erstellen und einbinden
 - Lombok als Code Generierungs Tool zur Vermeidung von Boilerplate-Code
 - Lizenz: [Eigene](#)
 - fusionauth-jwt, um Json Web Tokens (JWTs) auszulesen und zu validieren
 - Lizenz: [Apache 2.0](#)
 - swagger-codegen, um Modelle und Schnittstellen-Methoden aus der Definition der REST-API als Java Code zu generieren
 - Lizenz: [Apache 2.0](#)
 - ModelMapper, um Data Transfer Objects (DTOs) zu den Entity Klassen zu konvertieren
 - Lizenz: [Apache 2.0](#)
 - JUnit für Unit-Tests
 - Lizenz: [Eclipse Public License 2.0](#)
 - H2 Database als In-Memory Testing Datenbank
 - Lizenz: [MPL 2.0](#)
 - Integrationstests mit Spring Boot Test
 - Lizenz: [Apache 2.0](#)
 - Wenig Konfigurationsaufwand, einfache Entwicklung von REST-Anwendungen, Ecosystem mit großer Anzahl von Benutzern, Praxisbewährt, (Erwünscht von doubleSlash)
- Angular als Frontend Framework
 - npm als Abhängigkeits-/Paketmanager
 - Lizenz: [Artistic License 2.0](#)
 - Testing-Frameworks: Jasmine & Karma
 - Lizenz: [MIT](#)
 - Angular Material als generelle Komponenten Bibliothek
 - Lizenz: [MIT](#)
 - [angular-calendar](#) als Komponenten Bibliothek für eine Kalenderansicht
 - Lizenz: [MIT](#)
 - ng-openapi-gen um Modelle und Schnittstellen-Methoden aus der Definition der REST-API als TypeScript Code zu generieren
 - Lizenz: [MIT](#)
 - Lizenz: [MIT](#)

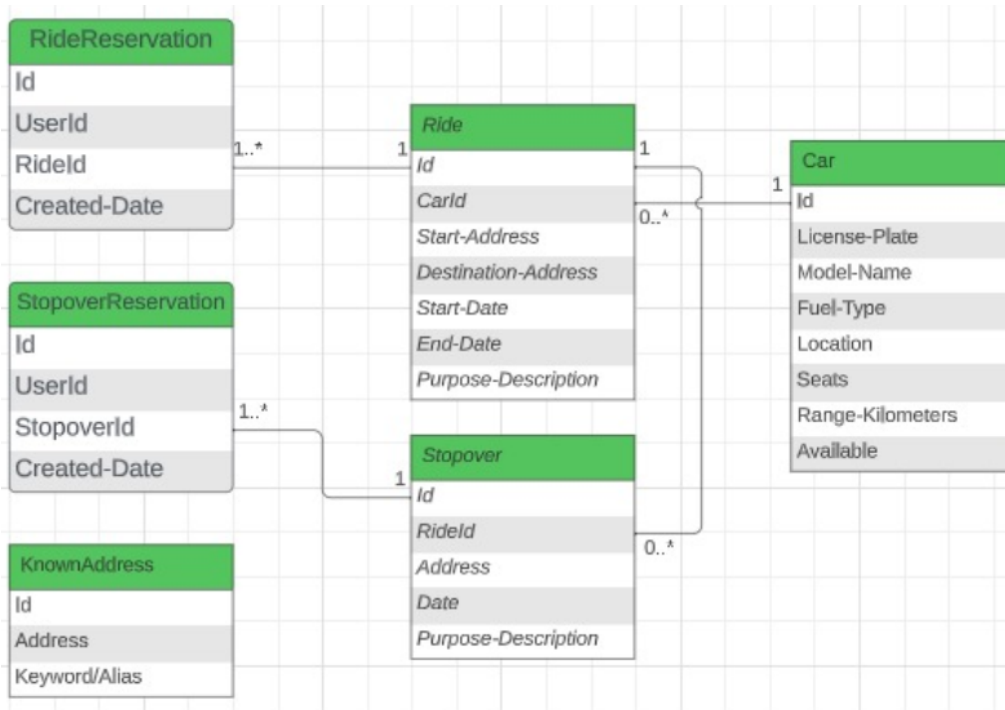
- Modularisierung durch Komponente-System, in der Praxis weit verbreitet, (Erwünscht von doubleSlash)
- PostgreSQL als Datenbank-Managementsystem (DBMS)
 - Lizenz: [PostgreSQL Licence](#)
 - Open Source, hat sich in der Praxis bewährt (Robust), gute SQL Unterstützung
- Git als Version Control System (VCS)
 - GitHub Repository als Code-Hosting-Plattform
 - GitHub Kanban Board für das Projektmanagement / Aufgabenverteilung
 - Github Actions als CI/CD Pipeline
 - Zentrale Verwaltung des Quellcodes, des Projektmanagements und der CI/CD Pipeline auf einer Plattform sichert eine bestmögliche Integration der einzelnen Dienste untereinander und verringert den Arbeitsaufwand
 - Basisfunktionen sind kostenfrei mit optionalen Abonnement Modellen für erweiterte Features und mehr Performance (z.B. mehr Speicherplatz)
- Docker zur Containerization und Virtualisierung
 - Lizenz: [Apache 2.0](#)
 - Industriestandard, gute Integration in z.B. IntelliJ IDEA und hat außerdem einen eigene Desktopanwendung zur Verwaltung von Docker Containern (Docker Desktop)
- IntelliJ IDEA Ultimate als Entwicklungsumgebung (IDE)
 - All-in-One Lösung, beinhaltet z.B. auch Git- und Datenbanken Verwaltungs Funktionalität
 - Kostenlose Bildungslizenzen für unter anderem Studierende



Erstes Datenmodell

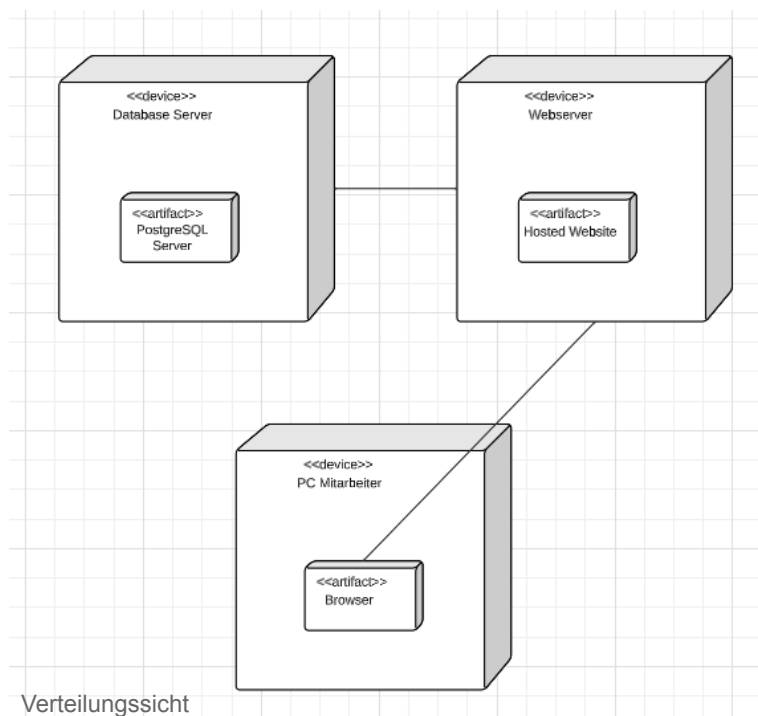
Softwarearchitektur

Struktursicht



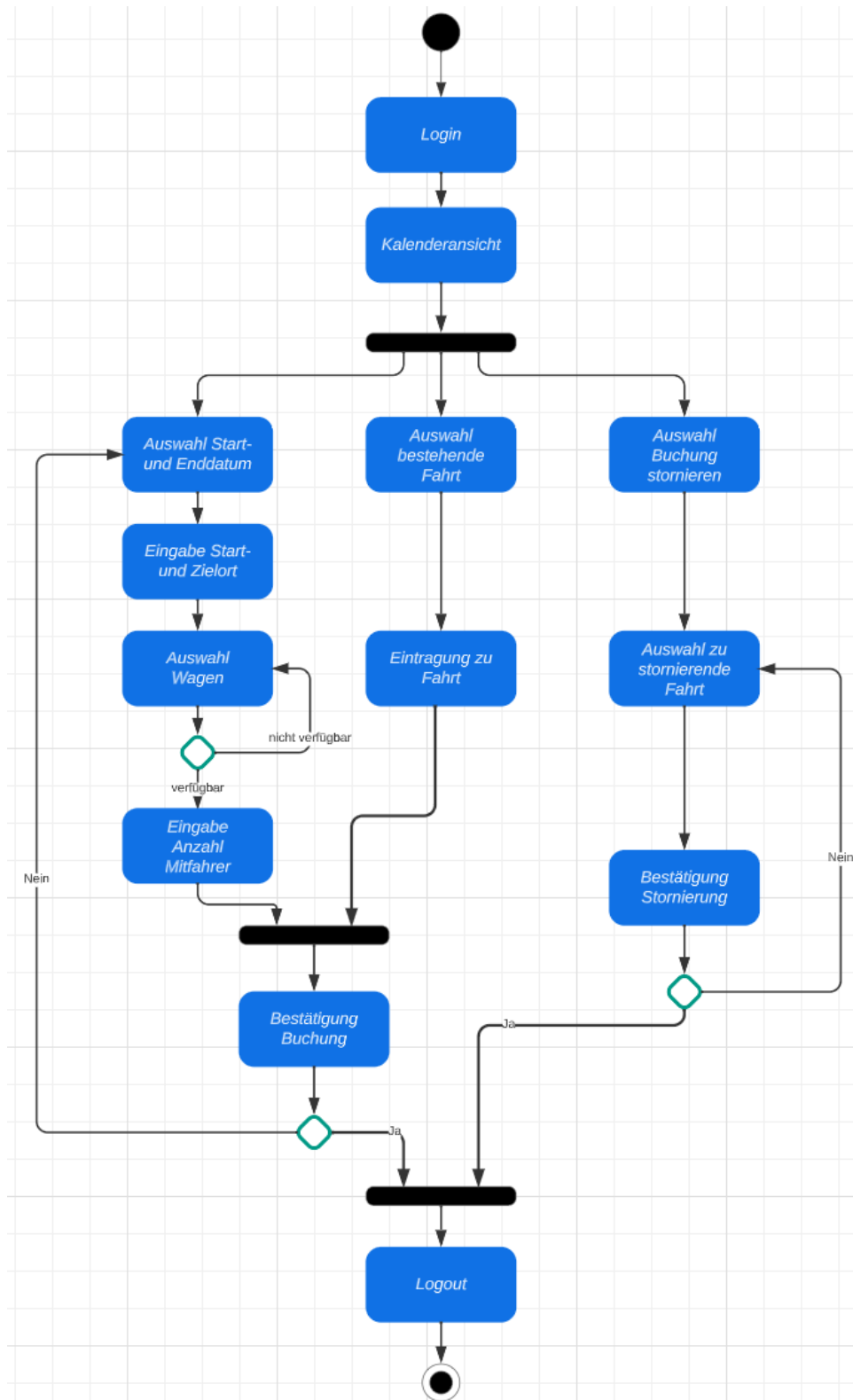
Struktursicht

Verteilungssicht



Verteilungssicht

Verhaltenssicht



Verhaltenssicht

Projektmanagement

Gruppenintern haben wir uns dazu entschieden, wöchentlich im Team zu präsentieren, was man erreicht hat (Weekly Standup).

Anschließend werden wir mit dem Kunden den aktuellen Stand durchgehen, um sofort zu sehen, ob die Features, die wir schon implementiert haben, und noch implementieren wollen, der Vorstellung entsprechen.

Unser Backlog haben wir in das Kanban Board von GitHub eingetragen. Hier nimmt sich jeder vor, was er bis zur nächsten Woche geschafft haben will.

Was andere Infos oder Absprachen angeht, haben wir einen Discord Server und eine WhatsApp Gruppe erstellt.

Definition of Done (DoD):

- Präsentation für Team vorbereitet (Jeder präsentiert Dienstagabend vor dem Meeting mit Kunden, was er erreicht hat, und sagt, was wichtig zu wissen fürs Team ist)
- Feature Dokumentiert
- Feature manuell getestet, ggfs. Unit-/Integrationstest

User Stories & Aufwandsschätzung

Wir haben geschätzt, was wir denken, wie viel Aufwand die einzelnen Backlog Items, in Stunden, sein könnten. Dies ist uns sehr schwer gefallen, da unterschiedliche Teammitglieder unterschiedliche Ideen hatten, aber auch natürlich, da das Thema Inhaltlich und das Schätzen allgemein für uns mit wenig Erfahrung verbunden ist. 15 Stunden pro Woche, pro Person, mal fünf Personen, mal acht Wochen für die Implementierung ergibt ein Zeitkontingent von 600 Stunden. Hier müssen natürlich Puffer für unvorhergesehene Probleme zugerechnet werden und jedes Feature muss dokumentiert und zur Präsentation aufbereitet werden.

- Als Mitarbeiter möchte ich mich mit meinem Doubleslash Account einloggen können.
 - Login Ansicht erstellen: 8 Stunden
 - Session Handling: 10 Stunden
 - Daten über Kunden-API validieren und Token erhalten: 18 Stunden
 - Kundendaten und Token in der Datenbank und Browser halten: 16 Stunden
 - **=> 52h**
- Als Mitarbeiter möchte ich in einer Kalenderansicht einen Zeitraum für meine Buchung auswählen können.
 - Kalenderansicht im Frontend erstellen, die Buchungen über einen oder mehrere Tage zulässt, und start und ende anwählbar macht: 36 Stunden
 - Daten in der Datenbank zu einer Buchung hinzufügen: 24 Stunden
 - **=> 60h**
- Als Mitarbeiter möchte ich bereits gebuchte Fahrten sehen, um zu sehen, wann kein Auto verfügbar ist.
 - Ansicht erstellen, die existierende Buchungen übersichtlich anzeigt: 30 Stunden
 - Möglichkeit, sich bei einer Fahrt hinzuzubuchen: 18 Stunden
 - Möglichkeit, sich bei einer Fahrt zu entfernen: 18 Stunden
 - **=> 66h**
- Als Mitarbeiter möchte ich Firmenstandorte und selbst gesetzte Adressen als Ziel und Startpunkt verwenden können.
 - Adressbuch einbinden, eingegebene Adressen werden automatisch vervollständigt/vorgeschlagen: 36 Stunden
 - Firmenstandorte hinterlegt und direkt auswählbar: 18 Stunden
 - **=> 54h**
- Als Mitarbeiter möchte ich Buchungen löschen können: **=> 18 Stunden**
- Als Mitarbeiter möchte ich Statistiken angezeigt bekommen.
 - OpenStreetView-Berechnung der Kilometer: 30 Stunden
 - Leaderboard-Design: 30 Stunden
 - **=> 60h**
- Als Mitarbeiter möchte ich im Buchungsprozess alle aktuell freien Fahrzeuge auswählen können **=> 30h**
- Als Mitarbeiter möchte ich automatisch einen Outlook-Termin an die Fahrtteilnehmer versenden können.
 - Versenden von ICS-Datei: 24 Stunden
 - Konfiguration SMTP: 22 Stunden

○ => 46h

- Als Mitarbeiter möchte ich Zwischenstopps zu einer Route hinzufügen können: 30 Stunden
- Als Mitarbeiter möchte ich eine Buchung vorzeitig beenden können: 18 Stunden

Insgesamt sind es ca. 386 Stunden für die Features. Puffer ca. 20%: 100 Stunden. Meetings mit dem Kunden: ca. 40 Stunden. Da wir uns in einem agilen Umfeld bewegen kommen für regelmäßige Überarbeitungen der Anforderungen noch Zeit hinzu: ca. 50h.

Aufgaben	Wochen	KW40	KW41	KW42	KW43	KW44	KW45	KW46	KW47	KW48	KW49	KW50	KW51	KW52	KW01	KW02	KW02
Phase 1: Planung																	
Kundenanforderung	2																
Verwendung von Technologien	2																
Zeit und Aufgabenmanagement	2																
Phase 3: Vorbereitung																	
Einrichtung & Einarbeitung der Technologien	2																
Initialisierung	2																
Phase 4: Umsetzung																	
Entwicklung	8																
Tests	8																
Phase 5: Dokumentation																	
Dokumentation fertigstellen	2																

Zeitplan

Schnittstellentechnologie

Fuhrparkmanagement RESTful API (Backend)

Wir haben uns an den [Zalando RESTful API Guidelines](#) orientiert. Der Inhalt der Anfragen und Rückgaben ist im JSON Format, ist fest definiert und manche Werte sind mit Konditionen verknüpft. So muss z.B. "seats" (aus Anfragen, siehe (*)) größer als 0 sein, sonst wird ein HTTP Status Code 400 (Bad Request) zurück gesendet. Generell wird ein HTTP Status Code 403 (Forbidden) zurück gesendet, falls kein Authentifizierungs-Token bei einer Anfrage mitgesendet wird.

Fahrzeugverwaltungs-Schnittstelle

Anfrage

HTTP Schnittstelle	http://localhost:8080/cars
HTTP Anfragemethode	GET
Parameter	-
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	<pre>[{ "id": 1, "licensePlate": "S-DS-123", "modelName": "BMW 118i", "fuelType": "PETROL", "location": "Stuttgart Feuerbach", "seats": 5, "range": 500, "available": true }]</pre>
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8080/cars
HTTP Anfragemethode	POST

Parameter	-
Beispiel-Anfrage-Inhalt (*)	<pre>{ "licensePlate": "S-DS-123", "modelName": "BMW 118i", "fuelType": "PETROL", "location": "Stuttgart Feuerbach", "seats": 5, "range": 500, "available": true }</pre>

Rückgaben

HTTP Status Code	201 (Created)
Beispiel-Rückgabe-Inhalt	<pre>{ "id":1, "licensePlate": "S-DS-123", "modelName": "BMW 118i", "fuelType": "PETROL", "location": "Stuttgart Feuerbach", "seats": 5, "range": 500, "available": true }</pre>
HTTP Status Code	400 (Bad Request)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8080/cars/{id}
HTTP Anfragemethode	PUT
Parameter	id
Beispiel-Anfrage-Inhalt	<pre>{ "licensePlate": "S-DS-123", "modelName": "BMW 118i", "fuelType": "PETROL", "location": "Stuttgart Feuerbach", "seats": 5, "range": 500, "available": true }</pre>

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	<pre>{ "id": 1, "licensePlate": "S-DS-123", "modelName": "BMW 118i", "fuelType": "PETROL", "location": "Stuttgart Feuerbach", "seats": 5, "range": 500, "available": true }</pre>
HTTP Status Code	400 (Bad Request)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	404 (Not found)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8080/cars/{id}
HTTP Anfragemethode	DELETE
Parameter	id
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	404 (Not found)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Reservations-Schnittstelle

Anfrage

HTTP Schnittstelle	http://localhost:8080/reservations
HTTP Anfragemethode	GET
Parameter	-
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	<pre>[{ "id": 1, "userId": "1", "rideId": 1, "createdAt": "2023-12-06T12:54:02.479Z", "deletedDate": "null" }]</pre>
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8080/reservations
HTTP Anfragemethode	POST
Parameter	-
Beispiel-Anfrage-Inhalt	<pre>{ "userId": "1", "rideId": 1, }</pre>

Rückgaben

HTTP Status Code	201 (Created)
Beispiel-Rückgabe-Inhalt	<pre>{ "id": 1, "userId": "1", "rideId": 1, "createdAt": "2023-12-06T12:54:02.479Z", }</pre>

	<pre>"deletedDate": "null" }</pre>
HTTP Status Code	400 (Bad Request)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8080/reservations/{id}
HTTP Anfragemethode	DELETE
Parameter	id
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	404 (Not found)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8080/reservations/{id}
HTTP Anfragemethode	PUT
Parameter	id
Beispiel-Anfrage-Inhalt	<pre>{ "userId": "1", "rideId": 1, }</pre>

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	<pre>{ "id": 1, "userId": "1", "rideId": 1, "createdDate": "2023-12-06T12:54:02.479Z", "deletedDate": "null" }</pre>
HTTP Status Code	400 (Bad Request)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	404 (Not found)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Fahrten-Schnittstelle

Anfrage

HTTP Schnittstelle	http://localhost:8080/rides
HTTP Anfragemethode	POST
Parameter	-
Beispiel-Anfrage-Inhalt	<pre>{ "carId": 1, "startAddress": "Stuttgart Feuerbach", "destinationAddress": "Karlsruhe Stadtmitte", "startDate": "2023-11-20T12:08:00.000Z", "endDate": "2023-11-20T12:11:00.000Z", "purpose": "Geschäftstreffen" }</pre>

Rückgaben

HTTP Status Code	201 (Created)
Beispiel-Rückgabe-Inhalt	<pre>{ "id": 1, "carId": 1, "startAddress": "Stuttgart Feuerbach", "destinationAddress": "Karlsruhe Stadtmitte",</pre>

	<pre> "startDate": "2023-11-20T12:08:00.000Z", "endDate": "2023-11-20T12:11:00.000Z", "purpose": "Geschäftstreffen" } </pre>
HTTP Status Code	400 (Bad Request)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8080/rides/{id}
HTTP Anfragemethode	PUT
Parameter	id
Beispiel-Anfrage-Inhalt	<pre> { "carId": 1, "startAddress": "Stuttgart Feuerbach", "destinationAddress": "Karlsruhe Stadtmitte", "startDate": "2023-11-20T12:08:00.000Z", "endDate": "2023-11-20T12:11:00.000Z", "purpose": "Geschäftstreffen" } </pre>

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	<pre> { "id": 1, "carId": 1, "startAddress": "Stuttgart Feuerbach", "destinationAddress": "Karlsruhe Stadtmitte", "startDate": "2023-11-20T12:08:00.000Z", "endDate": "2023-11-20T12:11:00.000Z", "purpose": "Geschäftstreffen" } </pre>
HTTP Status Code	400 (Bad Request)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	404 (Not found)

Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8080/rides/{id}
HTTP Anfragemethode	DELETE
Parameter	id
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	404 (Not found)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8080/rides
HTTP Anfragemethode	GET
Parameter	-
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	[{ "id": 1, "carId": 1, "startAddress": "Stuttgart Feuerbach", "destinationAddress": "Karlsruhe" }]

	<pre> Stadtmitte", "startDate": "2023-11-20T12:08:00.000Z", "endDate": "2023-11-20T12:11:00.000Z", "purpose": "Geschäftstreffen" }] </pre>
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Statistik-Schnittstelle

Anfrage

HTTP Schnittstelle	http://localhost:8080/statistics/users
HTTP Anfragemethode	GET
Parameter	-
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	To be decided
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8080/statistics/users/{id}
HTTP Anfragemethode	GET
Parameter	-
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	To be decided
HTTP Status Code	404 (Not found)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Übersicht Endpunkte (Swagger UI)

Swagger
Powered by SMARTBEAR

Explore

Fuhrparkmanagement API 1.0.0 OAS 3.0

Fuhrparkmanagement API

Servers
http://localhost:8080

Authorize

Cars

Cars with their attributes

- GET /cars GET cars
- POST /cars POST cars
- GET /cars/{id} GET cars/{id}
- PUT /cars/{id} PUT cars/{id}
- DELETE /cars/{id} DELETE cars/{id}

Rides

Combines rides with reservations

- GET /rides GET rides
- POST /rides POST rides
- GET /rides/{id} GET rides/{id}
- PUT /rides/{id} PUT rides/{id}
- DELETE /rides/{id} DELETE rides/{id}

Reservations

Reservation

- GET /reservations GET reservations
- POST /reservations POST reservations
- GET /reservations/{id} GET reservations/{id}
- PUT /reservations/{id} PUT reservations/{id}
- DELETE /reservations/{id} DELETE reservations/{id}

Statistics

Statistics; driven kilometers

- GET /statistics/users GET statistics/users
- GET /statistics/users/{id} GET statistics/users/{id}

Schemas

- Car >
- Ride >
- Reservation >
- Statistic >

Swagger UI Schnittstellen

Externer User Management Service (von doubleSlash)

Anfrage

HTTP Schnittstelle	http://localhost:8082/usermanagement/v2/authenticate
HTTP Anfragemethode	POST
Parameter	-
Beispiel-Anfrage-Inhalt	<pre>{ "name": "mbeck", "password": "mbeck-pw", }</pre>

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	<pre>{ "token": "eyJhbGciOiJIUzI1NiIsInR...", "user": { "id": 1, "name": "mbeck", "givenName": "Martin", "surname": "Beck", "mail": "Martin.Beck@dev.doubleslash.de" } }</pre>
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8082/usermanagement/v2/tokens/renew
HTTP Anfragemethode	POST
Parameter	-
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	<pre>{ "token": "eyJhbGciOiJIUzI1NiIsInR..."</pre>

	}
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8082/usermanagement/v2/tokens/rev oke
HTTP Anfragemethode	POST
Parameter	-
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	-
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8082/usermanagement/v2/users
HTTP Anfragemethode	GET
Parameter	-
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	[{ "id": 1, "name": "mbeck", "givenName": "Martin", "surname": "Beck", "mail": "Martin.Beck@dev.doubleslash.de" }]

]
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

Anfrage

HTTP Schnittstelle	http://localhost:8082/usermanagement/v2/users/{id}
HTTP Anfragemethode	GET
Parameter	-
Beispiel-Anfrage-Inhalt	-

Rückgaben

HTTP Status Code	200 (OK)
Beispiel-Rückgabe-Inhalt	<pre>{ "id": 1, "name": "test", "givenName": "test", "surname": "test", "mail": "test@doubleslash.org" }</pre>
HTTP Status Code	403 (Forbidden)
Beispiel-Rückgabe-Inhalt	-

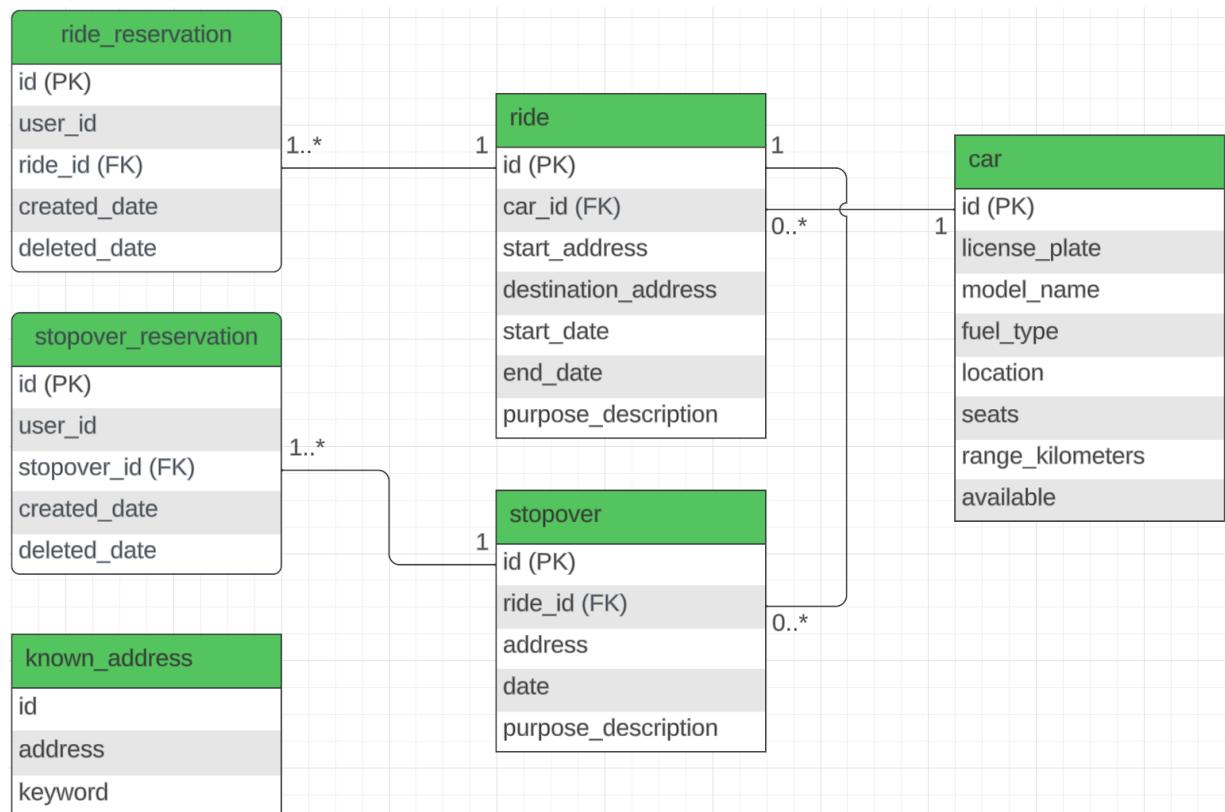
Codegenerierung zu Rest API Schnittstellen mit Swagger

Unsere Schnittstellen definieren wir als [OpenAPI Spezifikation](#) (siehe swagger/swagger-fuhrparkmanagement.yaml im Projekt Repository) und generieren aus dieser für das Backend und Frontend die jeweiligen Modell-Strukturen und Schnittstellen-Methoden (ohne fertige Implementation im Backend). Im Backend binden wir ein [extra Plugin](#) ein (siehe build.gradle Datei im Projekt Repository), welches eigene Tasks bereitstellt, die die Java-Klassen und Methoden generieren und in den Kompilierungsvorgang inkludieren. Für die eigentliche Implementation auf Backend-Seite

vererbt man die API Komponenten und überschreibt die zu implementierenden Schnittstellen-Methoden.

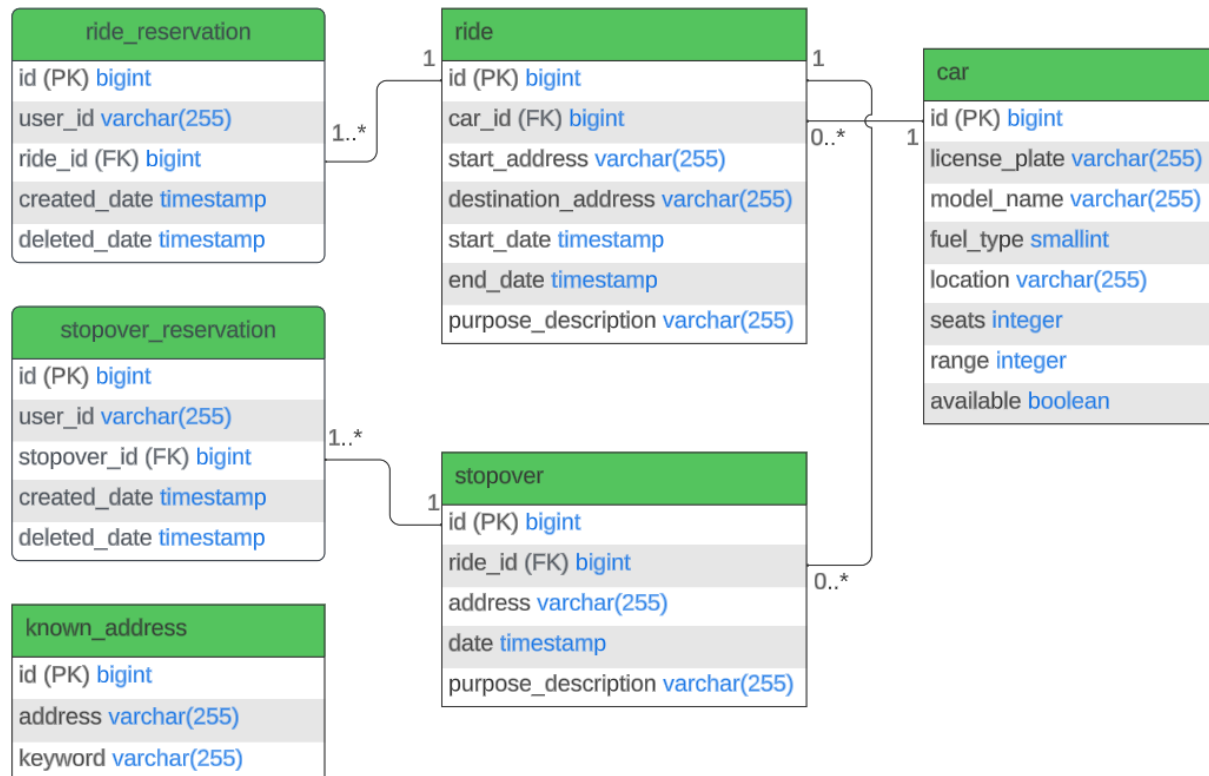
Für das Frontend benutzen wir eine [zusätzliche Abhängigkeit](#), um TypeScript Modell-Interfaces und voll implementierte Service Klassen mit Methoden zu den jeweiligen Schnittstellen zu generieren.

Datenbankmodell (logisch)



Logisches Datenbankmodell

Datenbankmodell (physisch)



Physisches Datenbankmodell

CI/CD Pipeline

Um eine höhere Codequalität zu erreichen, haben wir uns dazu entschieden, unseren Code kontinuierlich zu testen. Dafür verwenden wir bei GitHub sogenannte [Workflows](#), die jeweils verschiedene Jobs mit vordefinierten Schritten ausführen, wenn Commits auf den main-Branch gepusht werden oder ein Pull Request auf diesem eröffnet wird. Derzeit haben wir zwei Workflows (im `.github/workflows/` Ordner) definiert: einen um das Backend und einen, um das Frontend zu kompilieren und zu testen.

Backend-Workflow (mit Gradle)

Im ersten Schritt wird der Quellcode aus dem main-Branch abgerufen (mittels `git checkout`). Danach wird das Java Development Kit (JDK) mit der Java Version 17 aufgesetzt. Nun wird der Hashwert der Gradle-Wrapper Jar mit den offiziellen Hashwerten verglichen, um sicherzustellen, dass die Gradle-Wrapper Jar valide ist. Und letztendlich wird der `build`-Task mit dem Gradle-Wrapper ausgeführt, der unter anderem auch die Tests zum Backend ausführt.

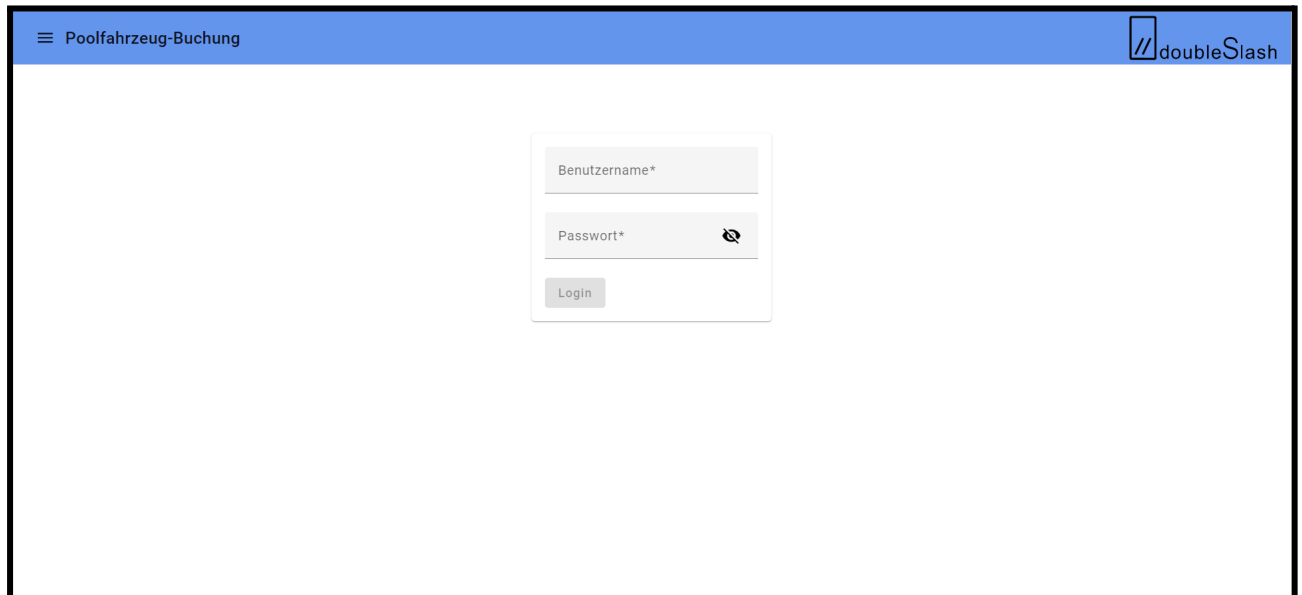
Frontend-Workflow (mit npm)

Ebenfalls wird im ersten Schritt der Quellcode aus dem main-Branch abgerufen (mittels `git checkout`). Danach wird Node.js mit der Version 18 aufgesetzt. Nun werden mit dem Befehl `$ npm ci` alle Abhängigkeiten im Frontend komplett neu installiert. Schließlich werden zwei `npm Run-Scripts` ausgeführt, welche das Frontend kompilieren und dann die Tests dafür ausgeführt werden.

Technischer Prototyp

Login-Ansicht

Wird aufgerufen, wenn der Benutzer nicht eingeloggt ist oder seine Anmeldeinformationen abgelaufen sind und daher nicht mehr funktionieren.

A screenshot of a web application's login page. The page has a blue header bar with the text "Poolfahrzeug-Buchung" on the left and a logo with the text "doubleSlash" on the right. In the center of the page is a white login form with a light gray border. The form contains two input fields: "Benutzername*" and "Passwort*", both with light gray borders. The "Passwort*" field has a small eye icon to its right. Below the input fields is a "Login" button with a light gray background and a dark gray border.

Login-Ansicht

Für das Anmelden benötigt man einen Benutzername und ein Kennwort. Bei erfolgreicher Authentifizierung mit dem externen User Management Service von doubleSlash wird zuerst ein "Long Time" Token (welcher 8 Stunden gültig sind) zurückgegeben und mit diesem ein "Short Time" Token (welcher 10 Minuten gültig ist) erstellt. Der "Long Time" Token wird im LocalStorage im Browser gespeichert, damit eine erneute Anmeldung nicht benötigt wird. Vorteilhaft ist daran, dass der LocalStorage persistent ist und sich nicht nach jeder Browser-Sitzung zurückgesetzt wird. Außerdem ist keine andere Website berechtigt, Informationen aus diesem zu unserer Website abzurufen und zu verändern.

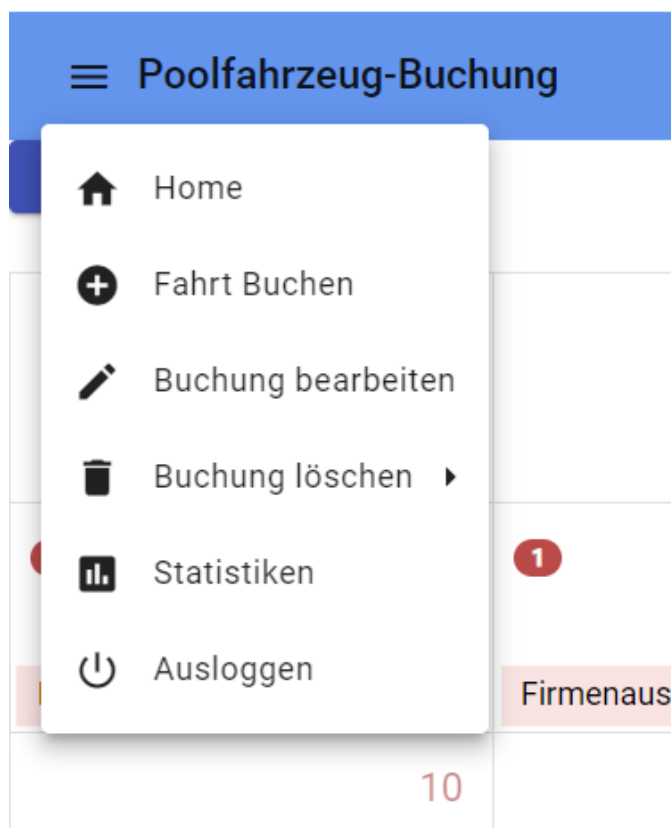
Kalenderansicht

Großer Kalender auf der Hauptseite, der alle gebuchten Fahrten pro Monat oder Woche anzeigt.

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
26	27	28	29	30	1	2
		Firmenausflug	Firmenausflug	Firmenausflug	Firmenausflug	Firmenausflug
3	4	5	6	7	8	9
Firmenausflug	Firmenausflug	Firmenausflug	Firmenausflug			
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Kalenderansicht

Sidebar-Ansicht



Sidebar-Ansicht

Auto anlegen

Eingabefelder:

- Nummernschild (Texteingabe): Bsp.: ES-OA-2023
- Modell-Name (Texteingabe): Bsp.: BMW 116i
- Treibstoff (Dropdown-Eingabe): Benzin, Diesel oder Elektrisch
- Standort (Dropdown-Eingabe): Die Standorte von doubleSlash (Stuttgart, Friedrichshafen, etc.)
- Sitze (Nummerneingabe, Ganze Zahl, größer 0): 1, 2, 3, etc.
- Reichweite (Nummerneingabe, Ganze Zahl, größer 0): Bsp.: 800

Nummernschild*	Modell-Name*	Treibstoff*	Standort*	Anzahl Sitze*	Reichweite (in KM)*	
ES-OA-2023	BMW 116i	Benzin	Stuttgart	5	800	<button>Auto anlegen</button>

Id	Nummernschild	Modell-Name	Treibstoff	Standort	Sitze	Reichweite	Verfügbar
1	ES-OA-2023	BMW 116i	PETROL	stuttgart	5	800	true

Neues Auto anlegen

Fahrt anlegen

Eingabefelder:

- Zweck (Texteingabe): Bsp.: Ausflug
- Startadresse (Texteingabe): Bsp.: Feuerbach
- Zieladresse (Texteingabe): Bsp.: München
- Buchungszeitraum (Datumseingabe, Start/Ende): Bsp: 12/6/2023-12/14/2023
- Startzeit (Uhrzeiteingabe, 24 Stunden): Bsp: 19:00
- Endzeit (Uhrzeiteingabe, 24 Stunden): Bsp: 18:00
- Auto (Dropdown-Eingabe): Bsp: BMW 116i

Zweck*	Startadresse*	Zieladresse*	Buchungszeitraum*	Startzeit*	Endzeit*
Ausflug	Feuerbach	München	12/6/2023 - 12/14/2023	19:00	18:00
MM/DD/YYYY - MM/DD/YYYY					
Auto*	<button>Fahrt buchen</button>				
BMW 116i					

Ride Id	Zweck	Startadresse	Zieladresse	Startdatum	Enddatum	Auto (Id)
5	Ausflug	Feuerbach	München	2023-12-06T19:00:00Z	2023-12-14T18:00:00Z	1

Neue Buchung erstellen