

# Prototyping Projektdokumentation

Name: Stefan Ulrich

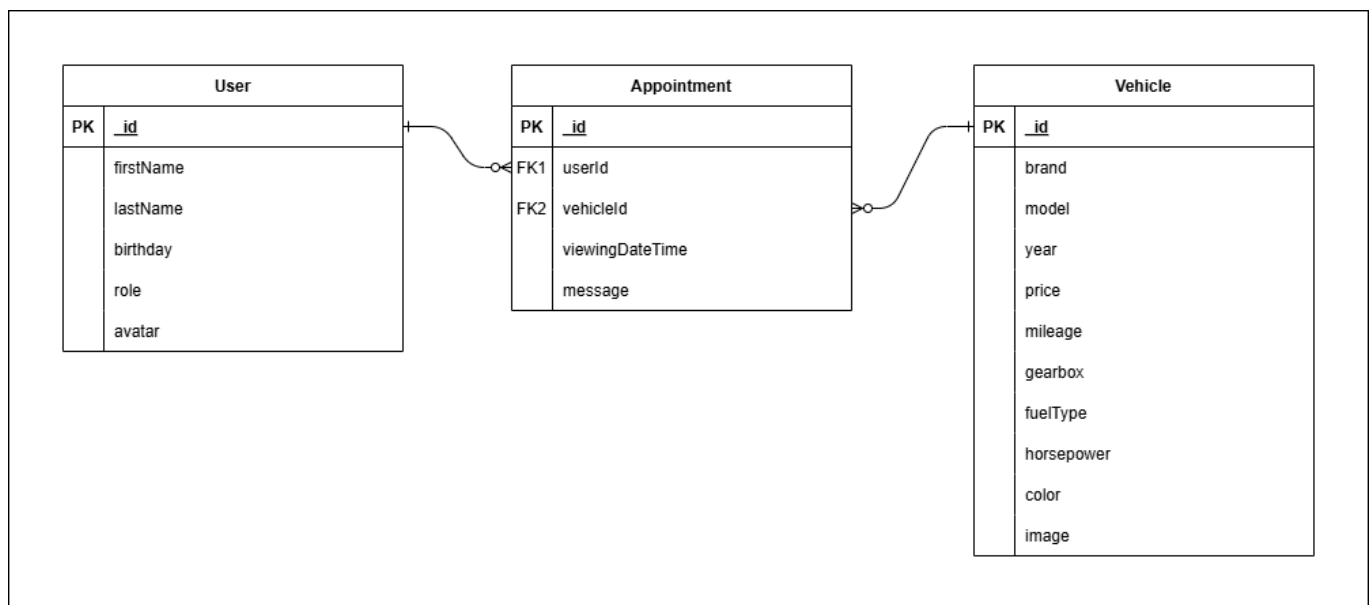
E-Mail: [ulricst3@students.zhaw.ch](mailto:ulricst3@students.zhaw.ch)

URL der deployten Anwendung: <https://effervescent-baklava-b3d894.netlify.app/>

## 1. Einleitung

VisiRide (Visit & Ride) ist die Admin-Oberfläche für eine Auto-Besichtigungstermin-App. Sie richtet sich an Autoliebhaber, die entspannt und unverbindlich Probefahrten machen möchten. Auf der Homepage sehen Sie auf einen Blick die Anzahl aller Datensätze insgesamt und in den einzelnen Collections. Per Schnellzugriff gelangen Sie von der Homepage direkt zu «Benutzer», «Fahrzeuge» oder «Besichtigungstermine». Unter «Benutzer» sind alle CRUD-Funktionen verfügbar, wobei für die Anlegung und Bearbeitung eine server- und clientseitige Validierung implementiert wurde. Die Fahrzeugübersicht bietet eine Listenansicht und eine Detailansicht mit technischen Daten und Bildern. Unter «Besichtigungstermine» können Sie neue Termine anlegen, vorhandene anzeigen und nicht mehr benötigte löschen. Es gibt noch kein Login-System mit Authentication/Authorization, aktuell ist das eine reine Admin-Ansicht. Normale Nutzer sehen später nur «Fahrzeuge» und «Besichtigungstermine». Da es sich um eine Demo-Version handelt, steht oben rechts der Button «Datenbank zurücksetzen» zur Verfügung, mit dem alle Einträge auf die Standardwerte zurückgesetzt werden.

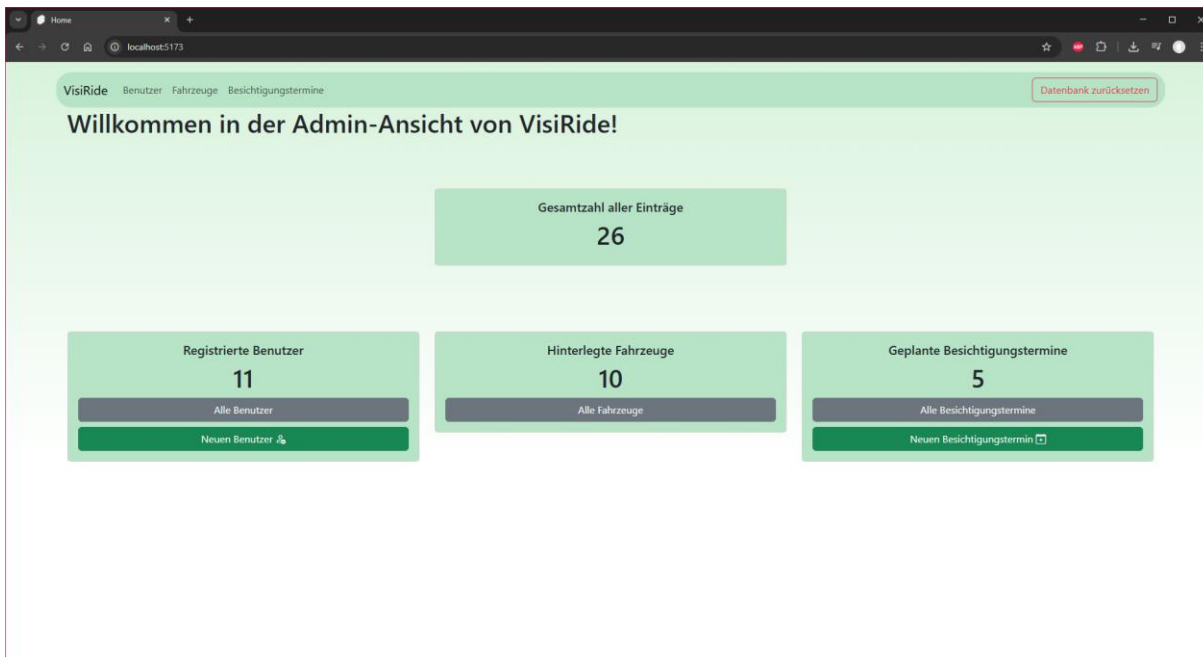
## 2. Datenmodell



### 3. Beschreibung der Anwendung

#### 3.1. Homepage

Route: /



Auf dieser Seite werden alle Datenbank-Einträge übersichtlich in Cards dargestellt. Über die Navigation gelangen Sie zur Homepage, indem Sie auf «VisiRide» oder «Datenbank zurücksetzen» klicken. Mit den zusätzlichen Schnellzugriff-Links (dargestellt als Buttons) können Sie direkt auf die wichtigsten Funktionen zugreifen. Die Navigation-Bar wurde einmalig in der Datei «+layout.svelte» implementiert, sodass diese nicht redundant in anderen Dateien hinzugefügt werden muss.

Auf folgende Routing-Optionen kann direkt via Link zugegriffen werden:

- «Alle Benutzer» → /users
- «Neuen Benutzer» → /users/create
- «Alle Fahrzeuge» → /vehicles
- «Alle Besichtigungstermine» → /appointments
- «Neuen Besichtigungstermin» → /appointments/create

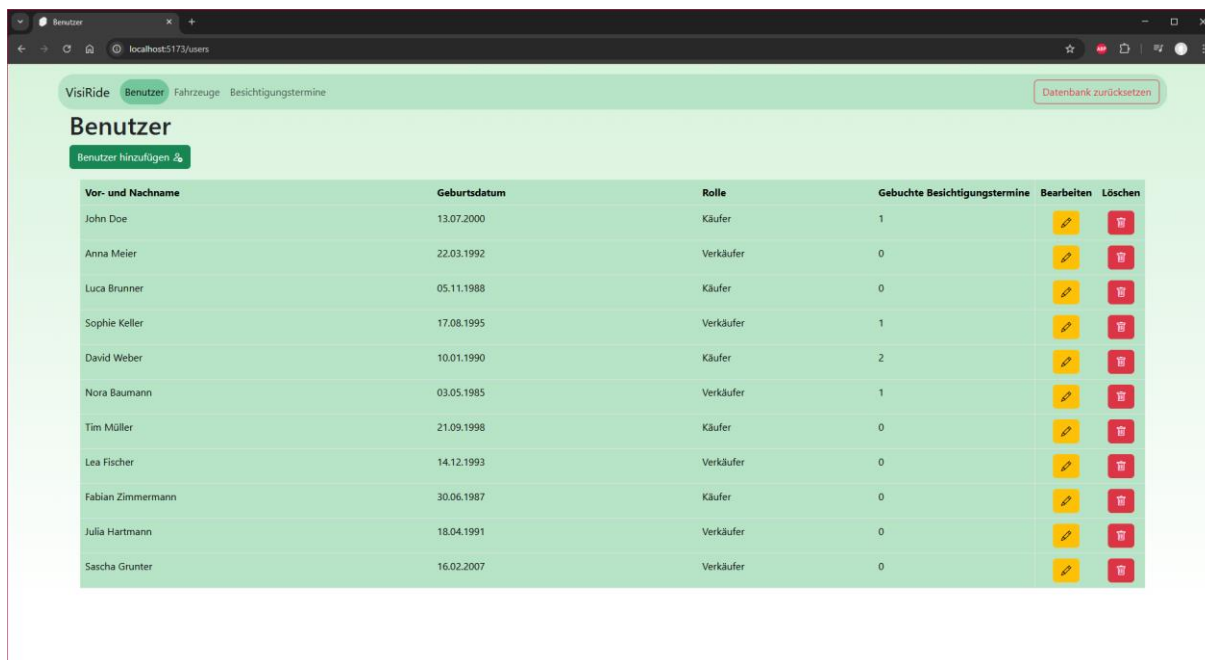
Dateien:

- lib/db.js
- routes/+layout.svelte
- routes/+page.svelte
- routes/+page.server.js

## 3.2. Benutzer

### 3.2.1. Benutzer auflisten

Route: /users



The screenshot shows a web browser window with the URL `localhost:5173/users`. The application has a navigation bar with 'VisiRide', 'Benutzer', 'Fahrzeuge', and 'Besichtigungstermine'. Below the navigation bar, there is a 'Benutzer' section with a 'Benutzer hinzufügen' button. The main content is a table with the following data:

Vor- und Nachname	Geburtsdatum	Rolle	Gebuchte Besichtigungstermine	Bearbeiten	Löschen
John Doe	13.07.2000	Käufer	1		
Anna Meier	22.03.1992	Verkäufer	0		
Luca Brunner	05.11.1988	Käufer	0		
Sophie Keller	17.08.1995	Verkäufer	1		
David Weber	10.01.1990	Käufer	2		
Nora Baumann	03.05.1985	Verkäufer	1		
Tim Müller	21.09.1998	Käufer	0		
Lea Fischer	14.12.1993	Verkäufer	0		
Fabian Zimmermann	30.06.1987	Käufer	0		
Julia Hartmann	18.04.1991	Verkäufer	0		
Sascha Grunter	16.02.2007	Verkäufer	0		

Auf dieser Seite werden alle Benutzer in der Collection «User» in einer Tabelle aufgelistet. Zusätzlich zu den Informationen der Benutzer wird ebenfalls eine Spalte mit der Anzahl der gebuchten Besichtigungsterminen, eine «Bearbeiten»- und eine «Löschen»-Funktion angezeigt. Über den Button «Benutzer hinzufügen» gelangt man zur Seite «/users/create». Für jeden Dateneintrag gibt es einen gelben «Bearbeiten» Button welcher zu «/users/[user\_id]» führt und einen roten Button, welcher den Dateneintrag löscht.

Dateien:

- `lib/db.js`
- `routes/users/+page.svelte`
- `routes/users/+page.server.js`

### 3.2.2. Benutzer erstellen

Route: /users/create

VisiRide Benutzer Fahrzeuge Besichtigungstermine Datenbank zurücksetzen

## Neuen Benutzer erstellen

Vorname\*

Nachname\*

Geburtsdag\*  Fülle dieses Feld aus.

Rolle\*

Auf dieser Seite kann ein neuer Benutzer angelegt werden. Hierfür wurde die wiederverwendbare «UserForm.svelte»-Datei verwendet. Der Screenshot zeigt den fehlgeschlagenen Versuch, einen Benutzer ohne Daten anzulegen. Die clientseitige Validierung von Chrome blockiert die Anfrage an den Server und weist darauf hin, dass es sich um ein Pflichtfeld handelt. Die Felder «Nachname», «Geburtsdag» und «Rolle» sind ebenfalls Pflichtfelder. Chrome springt automatisch zum nächsten Pflichtfeld und markiert es, sobald das vorherige korrekt ausgefüllt wurde. Nachdem alle Felder ausgefüllt sind, werden die Daten an den Server geschickt und dort erneut validiert. Im untenstehenden Bild sehen Sie, wie eine fehlerhafte serverseitige Validierung im Formular angezeigt wird. Wurden alle Daten korrekt eingegeben, erscheint die Bestätigung «Benutzer wurde erfolgreich gespeichert!».

Vorname\*  Der Vorname muss mindestens 3 Zeichen lang sein.

Nachname\*  Der Nachname muss mindestens 3 Zeichen lang sein.

Geburtsdag\*  Bitte gib ein gültiges Geburtsdatum in der Vergangenheit an.

Rolle\*

Dateien:

- lib/db.js
- lib/form/Controller.js
- lib/form/Validator.js
- lib/components/UserForm.svelte
- routes/users/create/+page.svelte
- routes/users/create/+page.server.js

### 3.2.3. Benutzer aktualisieren

Route: /users/[user\_id]

The screenshot shows a web browser window with the address bar displaying 'localhost:5173/users/682499bcc8fb7f3d9e70a15b'. The page title is 'Benutzer bearbeiten'. The breadcrumb 'Benutzer' is highlighted. The main content area shows a user profile with a photo of a man. Below the photo is a form with the following fields:

Id	Avatar
682499bcc8fb7f3d9e70a15b	/images/user_John-Doe.png

Vorname*	Nachname*
John	Doe

Geburtsdatum*	Rolle*
13.07.2000	Käufer

A blue 'Speichern' button is located at the bottom left of the form.

Auf dieser Seite kann ein Benutzer bearbeitet werden. Wie bei der Erstellung eines Benutzers kommt die wiederverwendbare «Userform.svelte»-Datei zum Einsatz. Oben wird zudem ein Breadcrumb angezeigt. Klickt man auf «Benutzer», gelangt man zurück zur Seite «/users». Im Formular werden zusätzlich zwei automatisch generierte Werte «Id» und «Avatar» angezeigt; diese sind jedoch nicht veränderbar. Die Validierung funktioniert genauso wie in Abschnitt 3.2.2 *Benutzer erstellen*. Ist die Validierung erfolgreich, so erscheint unten die Bestätigung «Benutzer wurde erfolgreich aktualisiert!».

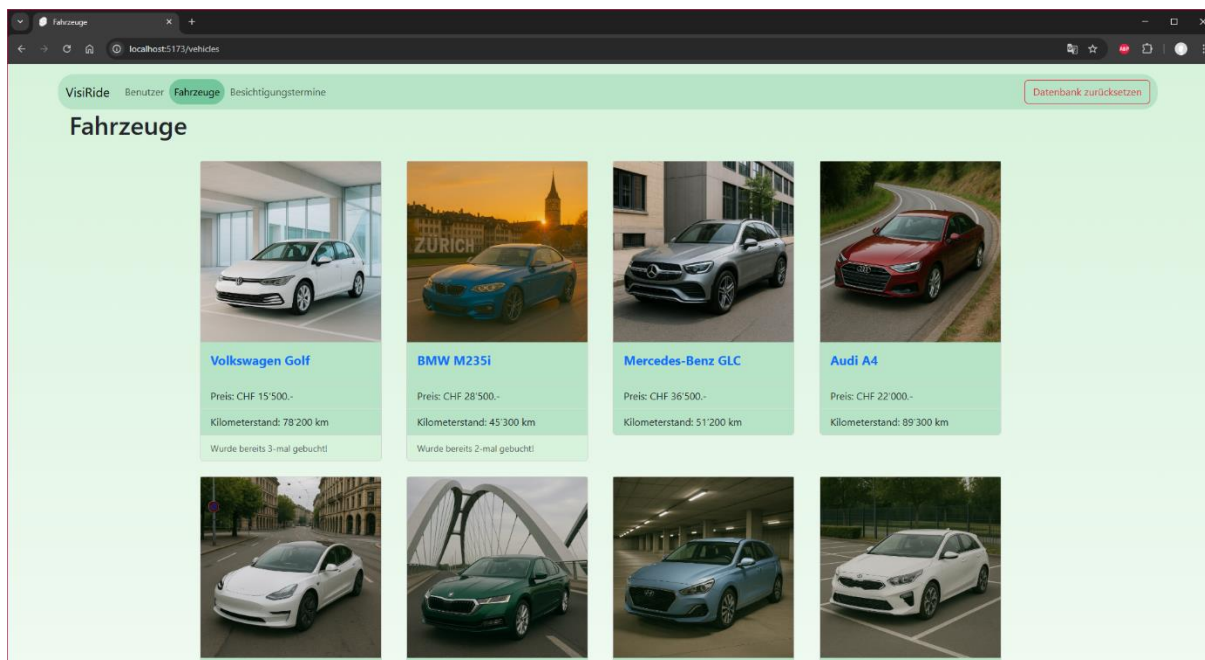
Dateien:

- lib/db.js
- lib/form/Controller.js
- lib/form/Validator.js
- lib/components/UserForm.svelte
- routes/users/[user\_id]/+layout.svelte
- routes/users/[user\_id]/+page.svelte
- routes/users/[user\_id]/+page.server.js

### 3.3. Fahrzeuge

#### 3.3.1. Fahrzeuge auflisten

Route: /vehicles



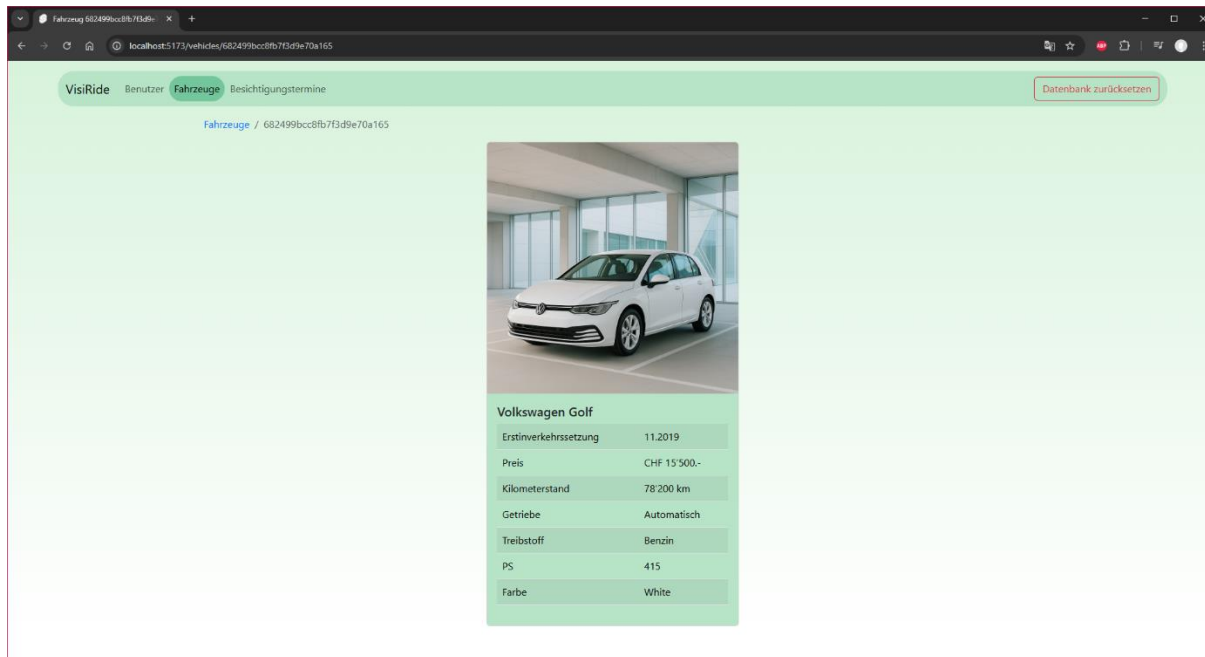
Auf dieser Seite werden alle Fahrzeuge aus der Collection «Vehicles» in einer Cards-Liste angezeigt. Für jede Fahrzeug-Card kommt die Datei «VehicleCard.svelte» zum Einsatz, die nur die wichtigsten Informationen anzeigt. Klickt ein Benutzer auf den Link eines Fahrzeugs, zum Beispiel «Volkswagen Golf», gelangt er zur Detailübersicht unter «/vehicles/[vehicle\_id]». Zusätzlich zu den Basisinformationen erscheint unter jedem Fahrzeug ein weiteres Feld, sofern das Fahrzeug gebucht wurde. Dieses Feld zeigt, wie oft ein Fahrzeug bereits gebucht wurde und wie beliebt es ist.

Dateien:

- lib/db.js
- lib/components/VehicleCard.svelte
- routes/vehicles/+page.svelte
- routes/vehicles/+page.server.js

### 3.3.2. Fahrzeugdetails

Route: /vehicles/[vehicle\_id]



Auf dieser Seite lassen sich alle Informationen zu einem Fahrzeug einsehen. Oben wird zudem ein Breadcrumb angezeigt. Ein Klick auf «Fahrzeuge» führt zurück zur Seite «/vehicles». Im zweiten Teil des Breadcrumbs wird, wie beim Benutzer, die jeweilige ID dynamisch angezeigt.

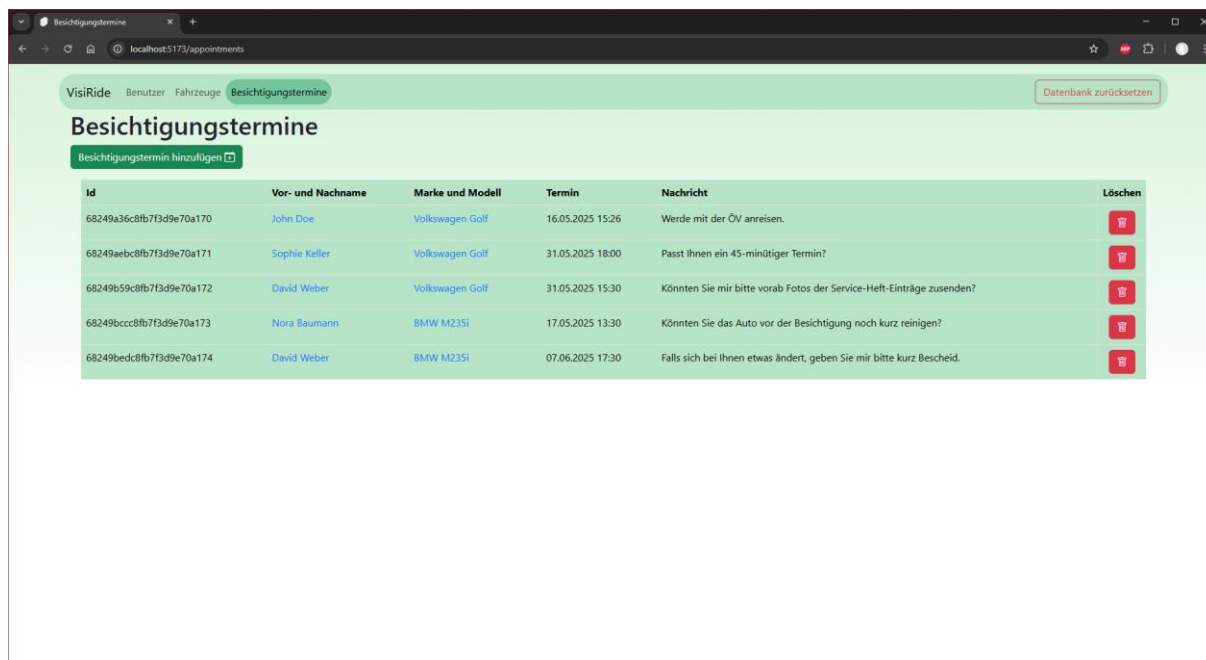
Dateien:






- lib/db.js
- routes/vehicles/[vehicle\_id]/+layout.svelte
- routes/vehicles/[vehicle\_id]/+page.svelte
- routes/vehicles/[vehicle\_id]/+page.server.js

### 3.4. Besichtigungstermine

#### 3.4.1. Besichtigungstermine auflisten

Route: /appointments



Id	Vor- und Nachname	Marke und Modell	Termin	Nachricht	Löschen
68249a36c8fb7f3d9e70a170	<a href="#">John Doe</a>	<a href="#">Volkswagen Golf</a>	16.05.2025 15:26	Werde mit der ÖV anreisen.	
68249aebc8fb7f3d9e70a171	<a href="#">Sophie Keller</a>	<a href="#">Volkswagen Golf</a>	31.05.2025 18:00	Passt Ihnen ein 45-minütiger Termin?	
68249b59c8fb7f3d9e70a172	<a href="#">David Weber</a>	<a href="#">Volkswagen Golf</a>	31.05.2025 15:30	Könnten Sie mir bitte vorab Fotos der Service-Heft-Einträge zusenden?	
68249bccc8fb7f3d9e70a173	<a href="#">Nora Baumann</a>	<a href="#">BMW M235i</a>	17.05.2025 13:30	Könnten Sie das Auto vor der Besichtigung noch kurz reinigen?	
68249bedc8fb7f3d9e70a174	<a href="#">David Weber</a>	<a href="#">BMW M235i</a>	07.06.2025 17:30	Falls sich bei Ihnen etwas ändert, geben Sie mir bitte kurz Bescheid.	

Auf dieser Seite werden alle Besichtigungstermine der Collection «Appointment» in einer Tabelle aufgelistet. Zusätzlich zu den Termininformationen werden auch Vor- und Nachname des Benutzers sowie Marke und Modell des Fahrzeugs als Links angezeigt. Ein Klick auf einen Benutzer führt zu «/users/[user\_id]», ein Klick auf ein Fahrzeug zu «/vehicles/[vehicle\_id]». Ganz rechts befindet sich eine «Löschen»-Funktion, die genauso wie auf der Seite «/users» funktioniert. Über den Button «Besichtigungstermin hinzufügen» gelangt man zur Seite «/appointments/create».

Dateien:

- lib/db.js
- routes/appointments/+page.svelte
- routes/appointments/+page.server.js



### 3.4.2. Besichtigungstermine erstellen

Route: /appointments/create

The screenshot shows a web browser window with the URL `localhost:5173/appointments/create`. The page title is 'Neuen Besichtigungstermin erstellen'. The breadcrumb navigation shows 'VisiRide > Benutzer > Fahrzeuge > Besichtigungstermine'. A 'Datenbank zurücksetzen' button is in the top right. The form contains the following fields and elements:

- Benutzer\***: A dropdown menu with 'John Doe' selected.
- Fahrzeug\***: A dropdown menu with 'Volkswagen Golf' selected.
- Termin\***: A date and time picker showing '19.05.2025 17:00'.
- Nachricht\***: A large text input field that is currently empty. A tooltip below it says 'Fülle dieses Feld aus.' (Fill out this field).
- Wettervorhersage ZHAW Winterthur**: A blue box displaying weather information for the selected date: 'Temperatur: 22.7 °C' and 'Niederschlag: 0 mm'.
- Speichern**: A blue button at the bottom left.

Auf dieser Seite kann ein neuer Besichtigungstermin angelegt werden. Der Screenshot zeigt den fehlgeschlagenen Versuch, einen Besichtigungstermin ohne Nachricht anzulegen. Auch hier blockiert die clientseitige Validierung von Chrome die Anfrage an den Server und weist darauf hin, dass das Feld «Nachricht» ein Pflichtfeld ist. Das Feld «Termin» ist ebenfalls erforderlich. Wird das Feld «Termin» geändert, wird – sofern API-Daten verfügbar sind – eine Wettervorhersage für das ausgewählte Datum angezeigt. Sind alle Angaben korrekt, wird der Benutzer auf die Seite «/appointments» weitergeleitet.

Dateien:

- `lib/db.js`
- `routes/appointments/create/+page.svelte`
- `routes/appointments/create/+page.server.js`

## 4. Erweiterungen

### 4.1. Bootstrap-Icons

In den unten aufgeführten «+page.svelte»-Dateien wurden verschiedene Icons für Buttons und Links verwendet. Hierfür wurde das CDN für Bootstrap-Icons CDN in der «app.html» eingebunden. Die Icons unterstützen die Nutzerführung und steigern die Benutzerfreundlichkeit der Oberfläche.

Dateien:

- app.html
- routes/+page.svelte
- routes/users/+page.svelte
- routes/appointments/+page.svelte
- routes/appointments/create/+page.svelte

### 4.2. Navigation Aktiv-Anzeige

Auf jeder Seite – ausgenommen der Homepage – wird stets hervorgehoben, in welchem Bereich des Navigationsmenüs sich der Benutzer befindet. Dadurch behält der Nutzer jederzeit den Überblick über seine aktuelle Position innerhalb der Anwendung. Dies wurde im Root-Layout implementiert.

Datei:

- routes/+layout.svelte

### 4.3. Homepage Number-Counter-Animation

Um einen schönen Effekt auf der Homepage zu erzeugen wurde eine Number-Counter-Animation hinzugefügt. Sie verleiht der Seite mehr Dynamik und lenkt die Aufmerksamkeit der Besucher gezielt auf zentrale Kennzahlen. Um die Kennzahlen abzurufen, wurde in der Funktion «getHomeScreenCounts» eine neue MongoDB-Query eingeführt.

Datei:

- lib/db.js
- routes/+page.svelte
- routes/+page.server.js

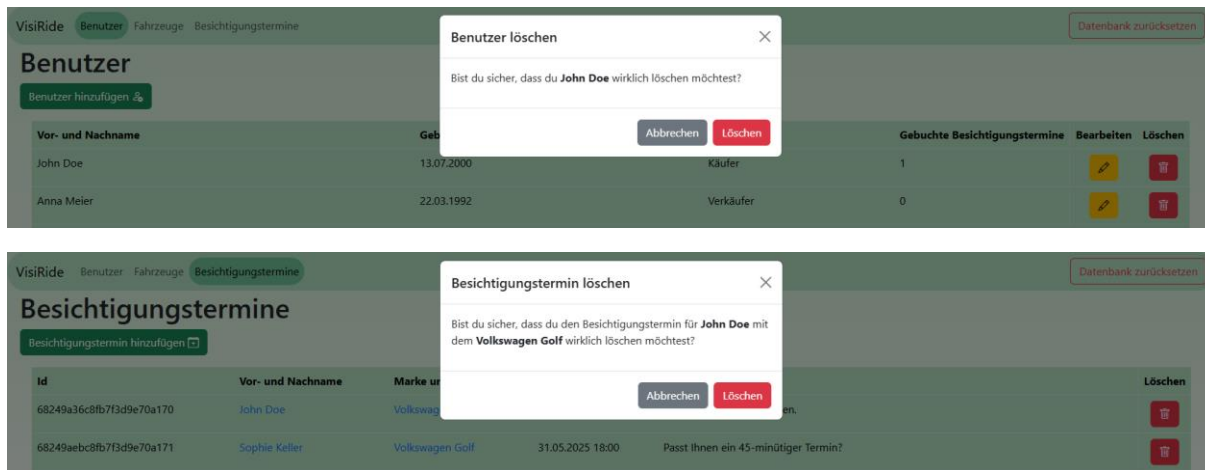
### 4.4. Datenbank Zurücksetzung

Für Demo-Zwecke wurde in der Navigation ein «Datenbank zurücksetzen»-Button eingeführt. In der Datei «db.js» wurden dafür die Methoden «clearAllCollections» zum Löschen aller Dokumente in den Collections und «insertDataFromJSON» zum Einfügen der Standarddaten implementiert.

Dateien:

- lib/db.js
- routes/+layout.svelte
- routes/+page.server.js

## 4.5. Löschen-Modal



Wenn der Löschen-Button von einem Benutzer oder einem Besichtigungstermin gedrückt wird, erscheint ein dynamisches Modal-Window, wie oben in den Screenshots gezeigt. Bestätigt der Benutzer mit «Löschen», wird der Dateneintrag endgültig in der Datenbank gelöscht. Klickt er hingegen auf «Abbrechen», schliesst sich das Modal-Window, und es wird kein Löschvorgang in der Datenbank durchgeführt.

Dateien:

- lib/db.js
- routes/users/+page.svelte
- routes/users/+page.server.js
- routes/appointments/+page.svelte
- routes/appointments/+page.server.js

## 4.6. Open-Meteo API-Aufruf

Wie in Abschnitt 3.4.2 *Besichtigungstermine erstellen* beschrieben, wird basierend auf dem Feld «Termin» angegeben Datum ein API-Aufruf zur Open-Meteo-API durchgeführt. Die Open-Meteo-API ermöglicht eine Wettervorhersage für bis zu 16 Tage in Voraus; als Beispielort wurden die Koordinaten (Latitude/Longitude) der ZHAW Winterthur gewählt. Zusätzlich ändern sich die Symbole für Temperatur und Niederschlag je nach Wetterlage, das Thermometer ist bei höheren Temperaturen stärker gefüllt und das Niederschlagssymbol wird bei stärkerem Regen deutlicher angezeigt. Befindet sich der Termin ausserhalb des Vorhersagezeitraums, erscheint ein Alert mit dem Text «Keine Wettervorhersage gefunden.». Da sich die Daten der API ständig ändern, wird bei jeder Änderung des Termins ein neuer API-Call ausgeführt – nicht beim Laden der Seite, sondern ausschliesslich, wenn sich der Termin ändert. Ursprünglich war geplant die offizielle SRF-Weather-API der SRG SSR zu verwenden; diese ist jedoch aufgrund einer begrenzten Anzahl von Aufrufen sowie eines nach sieben Tagen ablaufenden Bearer-Tokens eingeschränkt. Deshalb fiel die Wahl auf die Open-Meteo-API, da diese bis zu 10'000 API-Calls pro Tag erlaubt und keinen API-Key benötigt.

Datei:

- routes/appointments/create/+page.svelte