



IT-Projekt Schuljahr 2022/23
Fachinformatiker für Anwendungsentwicklung
Dokumentation zum IT-Projekt

SuS-Fahrgemeinschaften (MiFaZ)

Abgabetermin: 19.03.2023

Schüler*innen

Florian Stuchly
Tobias Hantke
David Luge
Patrik Lakatos

Prüfungsort

Berufsschule 1 Bayreuth
IT-Container
Kerschensteinerstraße 6
95448 Bayreuth

2**Inhaltsverzeichnis**

| | |
|---|----|
| Abbildungsverzeichnis..... | 4 |
| Tabellenverzeichnis..... | 5 |
| Glossar..... | 6 |
| Abkürzungsverzeichnis..... | 9 |
| 1 Einleitung | 10 |
| 1.1 Projektbeschreibung..... | 10 |
| 1.2 Projektziel | 10 |
| 1.3 Projektumfeld | 12 |
| 1.4 Projektbegründung..... | 12 |
| 1.5 Projekteinschränkungen..... | 13 |
| 1.6 Projektabgrenzung | 13 |
| 2 Projektplanung..... | 13 |
| 2.1 Projektphasen..... | 13 |
| 2.1.1 Ressourcenplanung | 16 |
| 2.1.2 Entwicklungsprozess | 16 |
| 3 Analysephase | 17 |
| 3.1 Wirtschaftlichkeitsanalyse..... | 17 |
| 3.1.1 Projektkosten | 18 |
| 3.2 Anwendungsfälle | 18 |
| 3.3 Lastenheft..... | 18 |
| 4 Entwurfsphase..... | 18 |
| 4.1 Zielplattform | 18 |
| 4.2 Entwurf der Benutzeroberfläche | 18 |
| 4.2.1 Mockup..... | 18 |
| 4.3 Datenmodell | 19 |
| 4.3.1 ER-Modell | 19 |
| 4.3.2 Tabellenmodell | 19 |
| 4.4 Maßnahmen zur Qualitätssicherung | 19 |
| 4.4.1 Planung Schreibtisch-Tests..... | 19 |
| 4.4.2 Planung Softwareergonomie..... | 20 |
| 4.5 Pflichtenheft | 20 |
| 5 Implementierungsphase | 20 |
| 5.1 Planung der Entwicklungsstufen | 20 |
| 5.2 Implementierung des Backends | 20 |
| 5.2.1 Implementierung der Persistenz-Ebene | 20 |

| | | |
|-------|---|----|
| 5.2.2 | Implementierung der API-Ebene..... | 20 |
| 5.2.3 | Implementierung der Authentifizierung | 21 |
| 5.3 | Implementierung des Frontends..... | 21 |
| 6 | Abnahme und Deployment..... | 21 |
| 6.1 | Deployment..... | 21 |
| 7 | Bedienungsanleitung | 22 |
| 8 | Fazit..... | 29 |
| 8.1 | Soll-/Ist-Vergleich | 29 |
| 8.2 | Resümee..... | 29 |
| 8.3 | Ausblick | 30 |
| A | Anhang..... | 31 |
| A.1 | Hardware..... | 31 |
| A.2 | Software | 31 |
| A.3 | Personal..... | 31 |
| A.4 | Lastenheft | 31 |
| A.5 | Pflichtenheft..... | 32 |
| A.6 | Use-Case-Diagramm..... | 35 |
| A.7 | Mock-Ups | 36 |
| A.8 | Entity-Relationship-Modell | 46 |
| A.9 | Sequenzdiagramm..... | 47 |
| A.10 | Test-Protokoll Black-Box-Test..... | 47 |
| A.11 | Screenshots ...Quellcode | 49 |

Abbildungsverzeichnis

| | |
|--|----|
| Abbildung 1 Fahrt erstellen | 10 |
| Abbildung 2 Fahrten suchen | 11 |
| Abbildung 3 Profil anzeigen | 12 |
| Abbildung 4 Tickets Sprint 0 | 14 |
| Abbildung 5 Tickets Sprint 1 | 15 |
| Abbildung 6 Tickets Sprint 2 | 16 |
| Abbildung 7 Logisches Modell der Datenbank | 19 |
| Abbildung 8 Registrieren | 22 |
| Abbildung 9 Login | 23 |
| Abbildung 10 Profil vervollständigen | 24 |
| Abbildung 11 Fahrten suchen | 25 |
| Abbildung 12 Fahrt anzeigen | 26 |
| Abbildung 13 Fahrt erstellen | 27 |
| Abbildung 14 Profil anzeigen | 28 |
| Abbildung 15 Profil bearbeiten | 28 |
| Abbildung 16 Dialog Änderungen speichern | 29 |
| Abbildung 17 Use-Case Diagramm Pflichtenheft | 34 |
| Abbildung 18 Use-Case Diagramm | 35 |
| Abbildung 19 Mockup Login-Seite | 36 |
| Abbildung 20 Mockup Registrieren Seite | 37 |
| Abbildung 21 Mockup Profil vervollständigen Seite | 38 |
| Abbildung 22 Mockup Fahrten suchen Seite | 39 |
| Abbildung 23 Mockup Fahrt anzeigen Seite | 40 |
| Abbildung 24 Mockup Fahrt erstellen Seite | 41 |
| Abbildung 25 Mockup Fahrt anlegen Dialog Seite | 42 |
| Abbildung 26 Mockup eigenes Profil anzeigen Seite | 43 |
| Abbildung 27 Mockup Profil bearbeiten Seite | 44 |
| Abbildung 28 Mockup Profil bearbeiten Dialog Seite | 45 |
| Abbildung 29 Mockup fremdes Profil anzeigen Seite | 46 |
| Abbildung 30 ER-Modell der Datenbank | 46 |
| Abbildung 31 Sequenzdiagramm des Logins | 47 |
| Abbildung 32 Quellcode des BottomNavigation Components | 49 |
| Abbildung 33 Quellcode Hashing und Validierung des Passworts | 50 |
| Abbildung 34 Definition des Datenbankschemas in der Prisma-eigenen Schemasprache | 51 |
| Abbildung 35 Teil 1 des Codes der Login Seite | 52 |
| Abbildung 36 Teil 2 des Codes der Login Seite | 53 |

Tabellenverzeichnis

| | |
|---|----|
| Tabelle 1 Grobe Zeitplanung..... | 13 |
| Tabelle 2 vier Schüler fahren getrennt von Hof nach Bayreuth..... | 17 |
| Tabelle 3 vier Schüler fahren gemeinsam von Hof nach Bayreuth | 17 |
| Tabelle 4 gesamte Projektkosten..... | 18 |

Glossar

| | Erklärung |
|---------------------|--|
| Webanwendung | Eine Webanwendung ist ein Anwendungsprogramm nach dem Client-Server-Modell. Anders als klassische Desktopanwendungen werden Webanwendungen nicht lokal auf dem Rechner des Benutzers installiert, sondern über den Browser aufgerufen. |
| Sprint | Sprint ist ein Begriff im agilen Projektmanagement, spezifisch in der Methode SCRUM, in diesem Projekt ist ein Sprint eine Blockwoche. |
| SCRUM | SCRUM ist ein Framework für eine bestimmte Art des agilen Projektmanagements. Es zeichnet sich durch schlanke Prozesse, schrittweise Entwicklung und regelmäßige Feedbackschleifen aus. |
| Figma | Figma ist ein Webtool, mit dem mehrere Personen gemeinsam Benutzeroberflächen entwickeln können. |
| Next.js | Next.js ist ein von Vercel erstelltes Open-Source-Webentwicklungs-Framework, das React-basierte Webanwendungen mit serverseitigem Rendering und der Generierung statischer Websites ermöglicht. |
| TypeScript | TypeScript ist eine Weiterführung von JavaScript, gemacht von Microsoft, mit dem man typisiert und klassenbasiert JavaScript programmieren kann. |
| trPC | Eine TypeScript Library, um typisierte Remote Procedure Calls auszuführen. |
| Prisma | Prisma ist ein ORM, welches speziell für Node.js und Typescript konzipiert wurde. Es dient dazu, Objekte in einer relationalen Datenbank abzulegen. |
| TailwindCSS | TailwindCSS ist ein Utility-First-CSS-Framework, das seinen Nutzern Utility Klassen bereitstellt. Durch die Verwendung, dieser Klassen lassen sich schnell und einfach eigene, einzigartige Designs erstellen. |
| NextAuth.js | Technologie, welche verschiedene Methoden zur Authentifizierung zu einem Projekt hinzufügt |
| Webapp | Eine Webapp ist ein Anwendungsprogramm nach dem Client-Server-Modell. Anders als klassische Desktopanwendungen werden Webanwendungen nicht lokal auf dem Rechner des Benutzers installiert, sondern über den Browser aufgerufen. |
| Vercel | Vercel ist eine amerikanische Cloud-Plattform Service Unternehmen. Es pflegt das Webentwicklungs-Framework Next.js. |
| PlanetScale | PlanetScale ist eine amerikanische Cloud-Plattform für das Hosting von Datenbanken. Des Weiteren haben sie eine Plattform um MySQL herum gebaut. |
| MySQL | MySQL ist eines der weltweit verbreitetsten relationalen Datenbankverwaltungssysteme. Es ist als Open-Source-Software sowie als kommerzielle Enterprise Version für verschiedene Betriebssysteme verfügbar und bildet die Grundlage für viele dynamische Webauftritte. |
| JavaScript | JavaScript ist eine Skriptsprache, die für dynamisches HTML in Webbrowsern entwickelt wurde, um Benutzeraktionen |

| | | |
|---|--|--|
| | | auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und so die Möglichkeiten von HTML zu erweitern. |
| Mockup | | Ein Mockup ist ein komplettes Produkt oder eine Attrappe, die genutzt wird, um Design und/oder Funktionen eines geplanten oder bereits eingeführten Produktes zu demonstrieren. Es ist oft ein maßstäbliches Modell bzw. eine Nachbildung zu Präsentationszwecken |
| ChakraUI | | ChakraUI ist eine einfache, modulare und zugängliche Komponentenbibliothek, die Bausteine für das Erstellen von React Anwendungen bietet. |
| ORM | | Ein ORM ist eine Technik, mit der in einer objektorientierten Programmiersprache Objekte in einer relationalen Datenbank abgelegt werden können. |
| Remote Procedure Call | | Remote Procedure Call (RPC) ist ein Protokoll, das ein Programm verwenden kann, um einen Dienst von einem Programm anzufordern, das sich auf einem anderen Computer in einem Netzwerk befindet, ohne die Details des Netzwerks verstehen zu müssen. RPC wird verwendet, um andere Prozesse auf den entfernten Systemen wie ein lokales System aufzurufen. Ein Procedure Call (Prozeduraufruf) wird manchmal auch als Function Call (Funktionsaufruf) oder Subroutine Call (Unterprogrammaufruf) bezeichnet. |
| Zod | | Zod ist eine JavaScript Library zur Typensicherung von Objekten und Variablen. Im Zusammenhang mit TypeScript wird die Library verwendet, um die Typensicherheit der Build-Time auf die Run-Time zu erweitern. |
| Build-Time | | Build-Time beschreibt die Zeit vor dem Ausführen eines Programmes. In dieser Phase wird beispielsweise der Code kompiliert. |
| Run-Time | | Run-Time beschreibt die Laufzeit eines Programmes vom Start (von der Ausführung) bis zum Ende (bis zum Verlassen), für die Laufzeitumgebung wird er manchmal als synonym verwendet. Wichtig ist der Run-Time-Begriff vor allem für die Fehlersuche und Umgebungskonfiguration. |
| Deployment | | Deployment oder Software Deployment bezeichnet die meistens halb- oder vollautomatischen ablaufenden Prozesse, der Softwareverteilung. Das Deployment umfasst dabei Aspekte wie Installation, Konfiguration, Aktualisierung und Wartung von Betriebssystemen und Anwendungssystemen auf PCs oder Servern. Auch Updates und Patches sowie deren Bereitstellungen gehören zum Deployment. |
| Continuous Integration/Continuous Deployment | Integration/Continuous Deployment | Continuous Integration (CI) ist ein Entwicklungsprozess, bei dem Code-Änderungen kontinuierlich und automatisch in eine gemeinsame Codebasis integriert werden, um Konflikte und Fehler frühzeitig zu erkennen. Continuous Deployment (CD) baut auf CI auf und umfasst die automatische Bereitstellung und Freigabe der integrierten Änderungen in die Produktion, sobald sie erfolgreich getestet wurden. Zusammen ermöglichen CI/CD schnellere, zuverlässigere und qualitativ hochwertigere Software-Entwicklung und -Lieferung. |
| Pipeline | | Pipelines beziehen sich auf die automatisierte Durchführung von mehreren Schritten im Rahmen von CI/CD-Prozessen. Eine Pipeline |

| | |
|----------------------------|---|
| | ist eine Reihe von Automatisierungsschritten, die ausgeführt werden, um Software von der Codeänderung bis zur Bereitstellung in der Produktionsumgebung zu führen. Pipelines ermöglichen eine effiziente und konsistente Durchführung von Tests, Builds, Integrationen und Bereitstellungen, um die Softwarequalität und -verfügbarkeit zu verbessern und die Entwicklungszeit zu verkürzen. |
| GitHub | GitHub lässt sich als eine Art soziales Netzwerk für Softwareentwickler beschreiben. Die Mitglieder können einander folgen, die Arbeit der anderen bewerten, Updates für bestimmte Projekte erhalten und öffentlich oder privat kommunizieren. |
| Commit | Commit ist ein Ausdruck aus der Softwaretechnik, der die bestätigende Freischaltung einer oder mehrerer Änderungen beschreibt. Er wird sowohl im Zusammenhang mit der Persistierung von Daten in einer Datenbank als auch beim Einpflegen von Programm-Quelltext in Versionsverwaltungssystemen verwendet. |
| Push | Der Befehl push wird verwendet, um Inhalte aus einem lokalen Repository in ein Remote-Repository hochzuladen. Per Push überträgt man die Commits aus deinem lokalen Repository in ein Remote-Repository. |
| Branch | In Git sind Branches Bestandteil deines alltäglichen Entwicklungsprozesses. Git-Branches sind quasi Verweise auf einen Snapshot von Änderungen. Wenn man ein neues Feature hinzufügen oder einen Fehler beheben möchte, legt man einen neuen Branch an, der große oder kleine Änderungen enthält. |
| T3-Stack | Zusammensetzung aus den Technologien: Next.js, Typescript, tRPC, Prisma, TailwindCSS und NextAuth.js. Diese sind auch im Glossar aufgeführt. |
| Routing | Mit Routing können Komponenten, Routen zugeordnet werden und so mit der URL auf bestimmte Komponenten verwiesen. |
| (React-)Komponenten | Komponenten sind voneinander unabhängige Funktionselemente, die in einer Anwendung wiederverwendet werden können und stellt den Grundbaustein aller React-Anwendungen dar. Häufig bestehen Komponenten aus einfachen JavaScript-Funktionen und – Klassen, aber man verwendet sie so, als wären sie angepasste HTML-Elemente. |
| HTML-Tag | Ein HTML-Tag (HTML-Element) ist eine Art HTML-Dokumentkomponente. |
| CSRF-Token | Das CSRF-Token ist ein geheimer Wert, der sicher verwahrt werden sollte, um während Cookie-basierter Sitzungen gültig zu bleiben. Das Token sollte in einem verborgenen Feld eines HTML-Formulars an den Client übertragen werden, das mit HTTP-POST-Anfragen übermittelt wird. Als bewährtes Verfahren wird empfohlen, die Herkunft von Anfragen anhand von Standard-Headern zu überprüfen. Zusätzliche Maßnahmen sollten auch die Herkunft von Quelle und Ziel identifizieren und vergleichen. Wenn die Ursprünge übereinstimmen, wird die Anfrage als legitim angesehen, wenn nicht, deutet dies auf eine domänenübergreifende Anfrage hin und wird verworfen. |
| Date-fns | Date-fns ist eine Sammlung von Funktionen, mit der man mit Datumswerten arbeiten kann. |

Abkürzungsverzeichnis

| Abkürzung | Bedeutung |
|-----------|--|
| BS1 | Berufsschule 1 |
| IT | Informationstechnologie |
| CO2 | Kohlenstoffdioxid |
| tRPC | typed Remote Procedure Calls |
| CSS | Cascading Style Sheets |
| ER-Modell | Entity-Relationship-Model |
| UI | Userinterface |
| ORM | Object-Relational-Mapping |
| CI/CD | Continuous Integration/Continuous Deployment |
| KISS | Keep it smart and simple |
| JWT | JSON-Web-Token |
| JSON | JavaScript Object Notation |
| CSRF | Cross-Site Request Forgery |

1 Einleitung

Die folgende Projektdokumentation schildert den Ablauf des IT-Projektes, welches die Mitarbeiter der SuS-AG im Rahmen ihrer Ausbildung zum Fachinformatiker der Fachrichtung Anwendungsentwicklung durchgeführt haben. Der Prüfungsort ist die Berufsschule 1 in Bayreuth.

1.1 Projektbeschreibung

Da viele Berufsschüler der BS1-Bayreuth nicht aus Bayreuth kommen, oder kein Auto haben, ist der Schulweg meist sehr aufwendig oder teuer.

Deshalb möchte die BS1-Bayreuth den Schülern eine Website bieten, auf der die Schüler und Schülerinnen Online, Fahrgemeinschaften bilden können.

1.2 Projektziel

Ziel des Projektes ist das erleichterte Bilden von Fahrgemeinschaften für die Berufsschule 1 Bayreuth.

Dazu können Schüler bzw. Fahrer eine Fahrt ausschreiben, bei der sie Datum, Abfahrtszeit, voraussichtliche Ankunft, Preis, Sitzplätze Treffpunkt sowie Ziel angeben können.

The screenshot shows a mobile application interface for creating a carpooling trip. At the top, there is a back button labeled '< Zurück' and a title 'Fahrt'. Below this, there are four input fields arranged in a 2x2 grid: 'Hinfahrt' (7:00 Uhr), 'Ankunft ca.' (16:00 Uhr), 'Preis / Fahrt' (4,00 €), and 'Sitzplätze' (4). Below these fields, there is a location pin icon labeled 'Treffpunkt' with a text input field containing 'REWE parkplatz'. Further down, another location pin icon labeled 'Ankunftsort' with a link 'Eigene Ankunftsort erstellen' and a text input field containing 'Musterstraße 22'. At the bottom of the form is a large green button labeled 'Erstellen'. The bottom of the screen features a dark blue navigation bar with three icons: a magnifying glass, a car, and a person.

Abbildung 1 Fahrt erstellen

Diese Fahrten können dann von Mitfahrern gesucht werden.

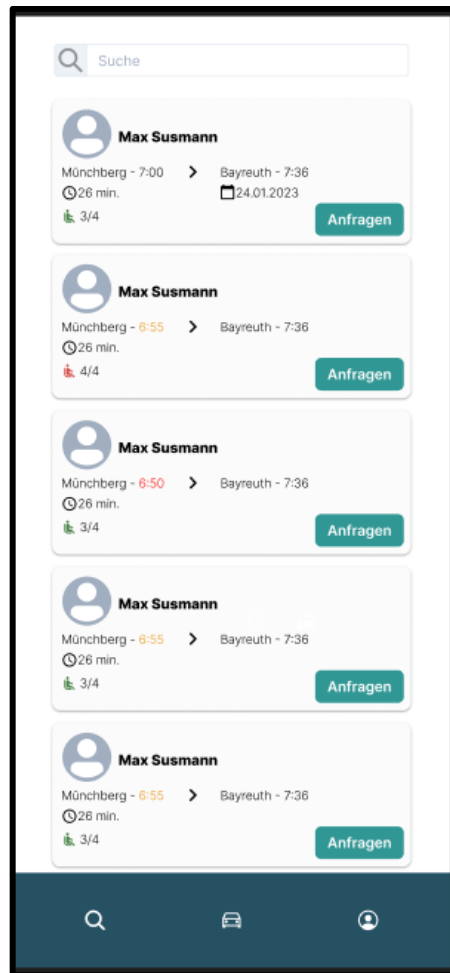


Abbildung 2 Fahrten suchen

Bei Nachfragen können die Mitfahrer auf das Profil des Fahrers gehen und ihm unter der hinterlegten Nummer schreiben.

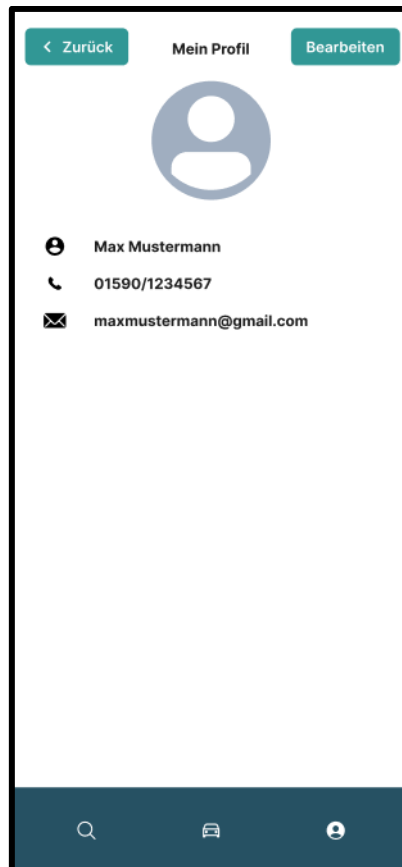


Abbildung 3 Profil anzeigen

Danach kann der Mitfahrer der Fahrt beitreten.
 Die Zahlung erfolgt privat-bar oder via Onlineüberweisung.

1.3 Projektumfeld

Auftraggeber des Projektes ist die Berufsschule 1 Bayreuth bzw. Die Lehrerin, Gabriele Ziegler.

Die wichtigsten Aufgaben der BS1-Bayreuth sind die Schüler*innen und die Schule materiell und ideell bei der Entwicklung beruflicher Handlungskompetenz und Förderung der Schlüsselqualifikationen zu unterstützen.

Die Aufgabe der Lehrer ist es die Schüler bei diesen Aufgaben zu unterstützen und ihnen die für den Beruf wichtigen Lehrstoffe beizubringen.

1.4 Projektbegründung

Der Hauptgrund ist, dass die Schüler meist getrennt in die Schule fahren und dadurch erhöhte Kosten entstehen, da die meisten Auszubildenden während ihrer Ausbildung nicht so viel Geld zur Verfügung haben, hilft dies den Schülern beim Einsparen von Kosten.

Zudem wird indirekt zur Entlastung der Umweltschädigung beigetragen, weil die meisten Schüler nicht mehr einzeln fahren, wird der CO₂ Ausstoß von beispielsweise 4 Autos, der meist höher ist, auf nur 1 Auto, der niedriger ist aufgeteilt.

Aufgrund dieser Probleme und der steigenden Zahl an Auszubildenden hat sich die BS-1 Bayreuth dazu entschieden, die Entwicklung einer Webanwendung in Auftrag zu geben, durch die schnell, einfach und unabhängig von Zeit und Ort, Fahrgemeinschaften gebildet werden können.

1.5 Projekteinschränkungen

Zum einen sind die Kosten eine Projekteinschränkung, da die Berufsschule die finanziellen Mittel nur begrenzt einsetzen kann, soll die App in der Implementierung so günstig wie möglich sein bzw. keine Kosten verursachen, dies gilt auch für die Inbetriebnahme der App.

Zum anderen war die Zeit eine weitere Einschränkung des Projektes, aufgrund dessen, dass das Projekt während der Berufsschulzeit erstellt werden musste, hatten die Autoren nur 3 Wochen Zeit das Projekt zu implementieren.

1.6 Projektabgrenzung

Es handelt sich um ein komplett eigenständiges Projekt, ohne Bezug zu anderen Projekten bzw. Teilprojekten.

2 Projektplanung

2.1 Projektphasen

Für die Umsetzung des Projektes standen der SuS-AG für jede Blockwoche insgesamt 57 Stunden, pro Person, zur Verfügung. Diese wurden auf festgelegte Phasen verteilt. Die grobe Zeitplanung lässt sich der folgenden Tabelle entnehmen.

| Projektphase | Geplante Zeit |
|---------------|-------------------|
| Information | 6 Stunden |
| Planung | 8 Stunden |
| Durchführung | 30 Stunden |
| Kontrolle | 13 Stunden |
| | |
| Gesamt | 57 Stunden |

Tabelle 1 Grobe Zeitplanung

Die einzelnen Hauptphasen wurden in drei Sprints aufgeteilt, ein Sprint entspricht immer einer Blockwoche, sie wurden wie folgt abgearbeitet:

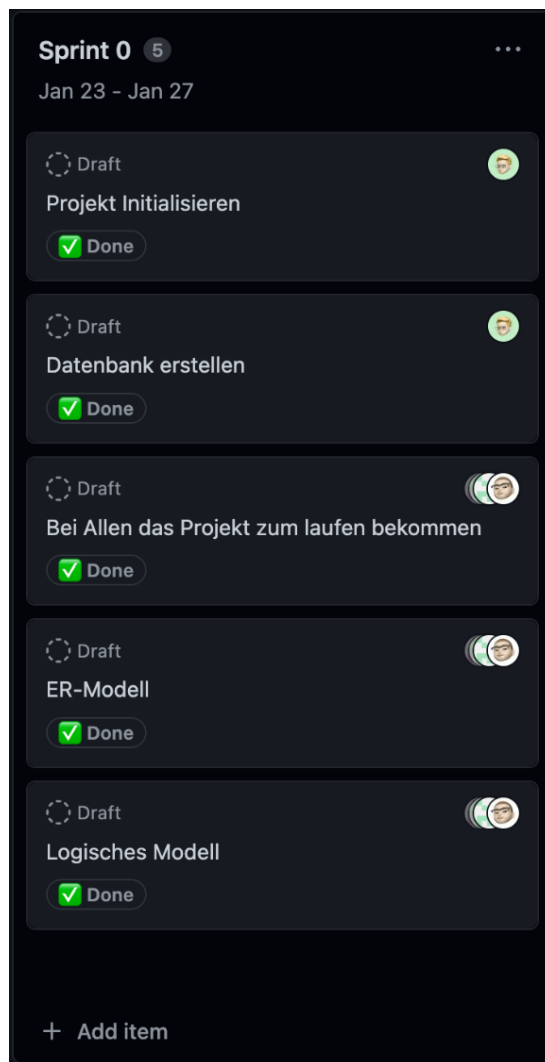


Abbildung 4 Tickets Sprint 0

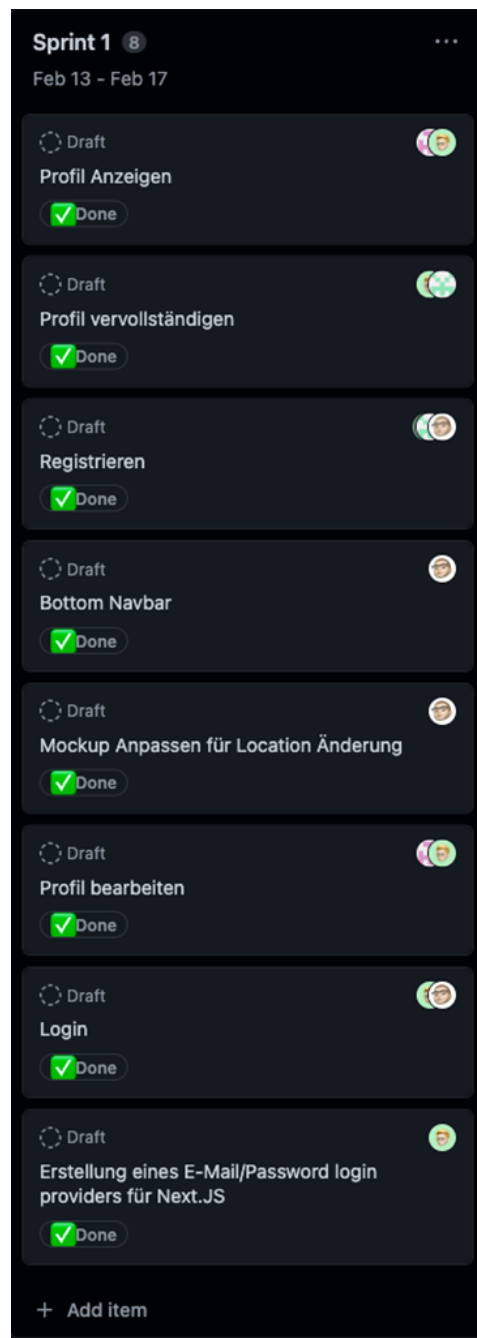


Abbildung 5 Tickets Sprint 1



Abbildung 6 Tickets Sprint 2

2.1.1 Ressourcenplanung

Da es sich bei der SuS-AG um eine fiktive Firma der Berufsschule 1 Bayreuth handelt, wurde darauf geachtet, dass die Anschaffungs- und Folgekosten des Projektes möglichst gering bzw. nicht vorhanden sind. Deshalb wurde das komplette Projekt mittels kostenloser Services und Software entwickelt, die verwendete Hardware sowie Software befinden sich im Anhang A.1 bzw. im Anhang A.2.

2.1.2 Entwicklungsprozess

Um eine flexible Umsetzung der Anforderungen zu ermöglichen, hat sich die SuS-AG für einen agilen Entwicklungsprozess entschieden. Da es sich um ein kleines Projekt mit wenigen Projektteilnehmern handelt, wurde die agile Methode SCRUM verwendet.

Die Methode SCRUM ermöglicht die flexible Änderung der Anforderungen und einen fließenden Entwicklungsprozess, ohne dabei viele Einschränkungen, zu definieren.

Die Umsetzung erfolgte mithilfe von einem SCRUM Board, dass via Figma dargestellt und verwaltet wurde. Ein Ausschnitt dieses Boards befindet sich im Anhang.

3 Analysephase

3.1 Wirtschaftlichkeitsanalyse

Im Folgenden sind zwei Beispiele gegeben, mit der sich die Wirtschaftlichkeit analysieren lässt:

Es wurde mit folgenden Werten gerechnet:

Entfernung: 111km (der Weg von Hof nach Bayreuth)

Benzinverbrauch: 10Liter/100km

Treibstoff: Benzin

Spritpreis: 1,80€/Liter

4 Schüler fahren getrennt von Hof nach Bayreuth:

| | Kosten | CO ² -Emission |
|---------------|---------------|---------------------------|
| Person 1 | 20,00€ | 0,045 t |
| Person 2 | 20,00€ | 0,045 t |
| Person 3 | 20,00€ | 0,045 t |
| Person 4 | 20,00€ | 0,045 t |
| | | |
| Gesamt | 80,00€ | 0,18 t |

Tabelle 2 vier Schüler fahren getrennt von Hof nach Bayreuth

4 Schüler fahren gemeinsam von Hof nach Bayreuth:

| | Kosten | CO ² -Emission |
|---------------|---------------|---------------------------|
| Person 1 | 5,00€ | 0,01125 t |
| Person 2 | 5,00€ | 0,01125 t |
| Person 3 | 5,00€ | 0,01125 t |
| Person 4 | 5,00€ | 0,01125 t |
| | | |
| Gesamt | 20,00€ | 0,045 t |

Tabelle 3 vier Schüler fahren gemeinsam von Hof nach Bayreuth

Ersparnis pro Person:

Kosten: 75% = 15€

CO²-Emission: 75% = 0.03375 t

Selbst wenn das Gewicht der vier Personen in einem Auto höher ist und dadurch mehr Sprit kosten bzw. mehr CO²-Emissionen entstehen, ist diese alternative Wirtschaftlicher, da ein Puffer von 75% gegeben ist.

3.1.1 Projektkosten

Folgende Gehälter wurden für die Kosten der Kalkulation verwendet, die Arbeitszeit pro Mitarbeiter beträgt 57 Stunden:

Projektleiter: 140,00€ pro Stunde

Entwickler: 120,00€ pro Stunde

Sämtliche anfallenden Projektkosten, können der folgenden Tabelle entnommen werden:

| | Kosten | Florian | Patrik | Tobias | David |
|---------------------------|------------|-----------|-----------|-----------|-----------|
| Leasing der Geräte | 340,00€ | 140,00€ | 80,00€ | 10,00€ | 110,00€ |
| Gehalt | 28.500,00€ | 7.980,00€ | 6.840,00€ | 6.840,00€ | 6.840,00€ |
| Miete | 2.150,00€ | 650,00€ | 500,00€ | 500,00€ | 500,00€ |
| | | | | | |
| Summe | 30.990,00€ | 8.770,00€ | 7.420,00€ | 7.350,00€ | 7.450,00€ |

Tabelle 4 gesamte Projektkosten

Die Gesamtkosten des Projektes betragen 30.990,00€.

3.2 Anwendungsfälle

Um eine grobe Übersicht über die Anwendungsfälle zu erhalten, die von dem umzusetzenden Programm abgedeckt werden sollen, wurde im Laufe der Analysephase ein Use-Case-Diagramm erstellt. Dieses Diagramm bildet alle Funktionen ab, die aus Endanwendersicht benötigt werden (siehe Anhang A.6).

3.3 Lastenheft

Am Ende der Analysephase wurde zusammen mit dem Kunden ein Lastenheft erstellt. Dieses umfasst alle Anforderungen des Auftraggebers an die zu erstellende Anwendung. Ein Auszug aus dem Lastenheft befindet sich im Anhang A.4.

4 Entwurfsphase

4.1 Zielplattform

Das Projekt wurde mit Hilfe eines sogenannten T3-Stacks erstellt, in diesem sind folgende Technologien enthalten: Next.js, TypeScript, tRPC, Prisma, TailwindCSS und NextAuth.js. Die fertige Webapp wird auf Vercel gehostet. Die Daten werden auf PlanetScale gespeichert. PlanetScale basiert auf MySQL.

Bei der Auswahl der Programmiersprache haben wir uns für JavaScript entschieden, da der Großteil der Gruppe bereits Kenntnisse mit dieser Programmiersprache hatte. Zudem kann man mit JavaScript Back- sowie Frontend implementieren, wodurch die Einarbeitung in das Projekt, um einiges leichter war.

4.2 Entwurf der Benutzeroberfläche

4.2.1 Mockup

Um die neue Anwendung möglichst benutzerfreundlich bedienen zu können, soll eine klar strukturierte und einfache Benutzeroberfläche entwickelt werden. Mit Hilfe von Mockups wurde hierfür zunächst ein Prototyp der Benutzeroberfläche erstellt, dafür wurde die Software Figma verwendet. Damit die Benutzeroberfläche am Ende den Anforderungen und Vorstellungen des Auftraggebers

entspricht, wurde darauf geachtet, dass die Software nach dem Mobile-First Konzept gestaltet wurde.

In der Hauptansicht soll auf Wunsch von Frau Ziegler eine Navigationsleiste sein - mit der man zu den Seiten (eigenes) Profil anzeigen, Fahrten suchen und Fahrt erstellen, navigieren kann.

Das Mockup der Anwendungssoftware befindet sich im Anhang A.7.

4.3 Datenmodell

Um einen groben Überblick über alle Daten zu erhalten, wurde schrittweise eine Analyse des Verarbeitungsablaufs durchgeführt. Dafür wurde hinterfragt, welche Daten gespeichert werden müssen und in welcher Beziehung sie zueinander stehen.

4.3.1 ER-Modell

Auf Grundlage der herausgearbeiteten Entitätstypen wurde ein Entity-Relationship-Model erstellt, welches die Typen und ihre Beziehung graphisch veranschaulichen soll. Dieses befindet sich im Anhang A.8. Folgende Entitätstypen wurden herausgearbeitet und abgebildet: Account, User, Trip, Sessions, _Passengers, bei _Passengers handelt es sich um eine Beziehungstabelle zwischen User und Trip.

4.3.2 Tabellenmodell

Da die Protagonisten sich dafür entschieden haben, eine relationale Datenbank zu verwenden, wurde das Entity-Relationship-Model in ein Tabellenmodell überführt.

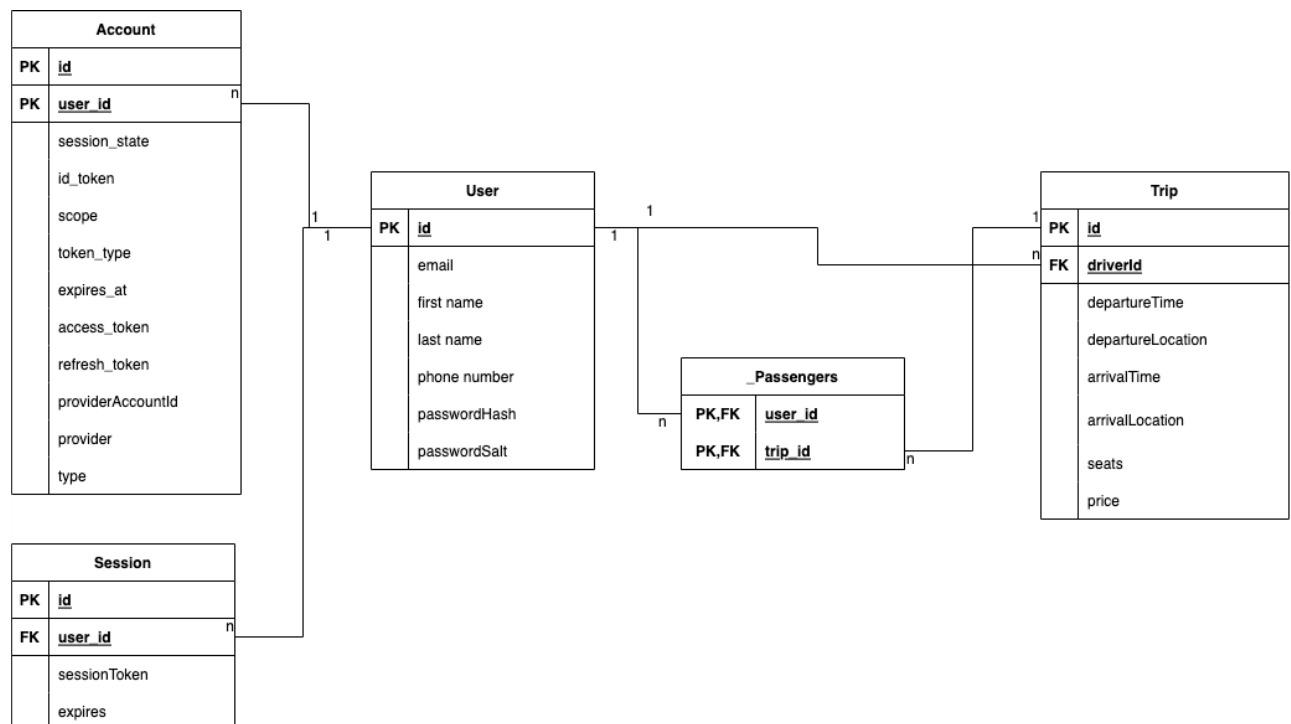


Abbildung 7 Logisches Modell der Datenbank

Es wurden aus dem Entity-Relationship-Model die folgenden 5 Tabellen übernommen: Account, User, Session, Trip und _Passengers .

4.4 Maßnahmen zur Qualitätssicherung

4.4.1 Planung Schreibtisch-Tests

Um eine angemessene Softwarequalität zu gewährleisten, müssen verschiedene Tests durchgeführt werden. Die zu testenden Funktionen umfassen:

- Registration

- Login
- Profil vervollständigen
- Profil bearbeiten
- Fahrt anzeigen
- Fahrt beitreten
- Fahrt anlegen

Zu diesen Funktionalitäten liegt im Anhang A.10 ein Testprotokoll vor.

4.4.2 Planung Softwarergonomie

Aufgrund der Zeiteinschränkungen haben wir uns dafür entschieden, dass die Ergonomie und Barrierefreiheit durch eine Library erledigt wird. Deshalb haben wir uns für ChakraUI entschieden, da diese eine in der Industrie verbreitete Frontend-Library ist.

4.5 Pflichtenheft

Das Pflichtenheft baut auf dem Lastenheft (vgl. dazu Abschnitt 3.3) auf und beschreibt, wie und womit die Entwickler die Anforderungen des Kunden umsetzen möchte. Es dient somit als Leitfaden für die Realisierung des Projektes. Das Pflichtenheft befindet sich im Anhang A.5.

5 Implementierungsphase

5.1 Planung der Entwicklungsstufen

Die Planung der Entwicklungsstufen wurde zusammen mit der Sprint Planung durchgeführt. Wir haben zu Beginn folgende Meilensteine definiert:

Login/Registrierung-, Profil- und Fahrt-Meilensteine.

Unser Ziel war es, jeden Sprint einen Meilenstein abzuschließen.

5.2 Implementierung des Backends

5.2.1 Implementierung der Persistenz-Ebene

Um die Nutzerdaten persistent zu speichern haben wir uns hier für die Datenbank Plattform PlanetScale entschieden, da sie im Zusammenspiel mit dem Prisma ORM eine simple und komfortable Abstraktion über MySQL ist. Durch PlanetScale ist es möglich die Datenbank wie in Git durch Branches weiterzuentwickeln. Durch Prisma ist es dagegen möglich automatisch die Daten in der Datenbank auf ein neues Schema zu migrieren, ohne selbst dafür SQL-Skripte schreiben zu müssen. Der Vorteil hiervon ist es, dass hier menschliche Fehler umgangen werden können.

In der Schema-Sprache von Prisma kann eine Relation so definiert werden:

```
model <NAME> {  
  <ATTRIBUT_NAME> <ATTRIBUT_TYP>  
}
```

Aus diesem Schema wird dann beispielsweise automatisch das SQL-DDL Skript zur Erstellung des Datenbankschemas erstellt.

5.2.2 Implementierung der API-Ebene

Für den API-Layer haben wir uns wegen der Simplizität, der TypeScript und NextAuth Integration für die Library "tRPC" entschieden. Durch diese ist es möglich mit Leichtigkeit typensichere "Remote Procedure Calls" durch TypeScript an das Backend zu stellen. Des Weiteren ist es möglich durch die Library "Zod" die Typensicherheit von der Build-Time sogar in die Run-Time zu erweitern, da diese automatisch die Objekte validiert.

5.2.3 Implementierung der Authentifizierung

Um die Sicherheit der App zu gewährleisten haben wir uns für die Authentifizierung die Library NextAuth entschieden. NextAuth ist eine Open Source JavaScript Library für die Authentifizierung von Usern im Web. Da wir als Anforderung E-Mail/Passwort Authentifizierung umsetzen sollten mussten wir von Datenbankbasierten Sessions umsteigen auf JWT-Basierten Sessions. Der JWT wird zusammen mit einem CSRF-Token als Cookie im Browser des Nutzers gespeichert.

5.3 Implementierung des Frontends

Das Frontend wurde mit React implementiert, da es ein modernes, einfaches und leicht verständliches Frontend Framework ist. Als Frontend-Library haben wir uns für ChakraUI entschieden, weil es ein schlankes, für Web-Entwickler intuitives, einfaches und leistungsstarkes Frontend Framework ist, um mobile Webseiten schneller entwickeln zu können.

Für das benutzerdefinierte Styling wurde die Bibliothek TailwindCSS benutzt. Durch die Verwendung von TailwindCSS lassen sich schnell und leicht eigene, einzigartige Design erstellen. Für die Gestaltung von Webseiten enthält TailwindCSS die Grundlagen, wie z.B. Farben, Größen, Rändern, Positionierung, etc...

Die Navigationsleiste wurde beispielsweise mit ChakraUI und TailwindCSS implementiert. Durch React musste diese nur einmal als Komponente erstellt werden, somit konnte die Navigationsleiste immer wieder, wie ein HTML-Tag verwendet werden.

Zudem wurde Next.js verwendet, diese Bibliothek hat das Routing übernommen, wodurch eine Single Page Applikation erstellt werden konnte.

Date-fns ist eine Sammlung von Funktionen, mit der man mit Datumswerten arbeiten kann. Dies wurde verwendet um die Abfahrtszeit, Ankunftszeit oder das Datum, korrekt darzustellen.

Für die Icons wurde React-Icons angewendet, React-Icons ist eine kleine Bibliothek mit deren Hilfe man Icons von vielen verschiedenen Anbietern verwenden kann.

6 Abnahme und Deployment

6.1 Deployment

Deployments werden in diesem Projekt nach dem CI/CD (Continuous Integration/Continuous Deployment) Prinzip durchgeführt. Ein Deployment ist Teil einer "Pipeline". Eine Pipeline besteht hier aus dem Bauen der Applikation und dem anschließenden Starten der Anwendung auf einem Webserver von Vercel. Die Pipeline wird bei jedem Pushen von Commits auf GitHub ausgeführt, dabei wird unterschieden ob auf den "Haupt-Branch" (main-branch) oder auf andere Branches gepusht wurde. Dies passiert, um die Produktionsumgebung vor Fehlern zu schützen, denn auf den "Haupt-Branch" kann in GitHub nicht direkt gepusht werden, dies kann nur nach einem Code-Review der gewünschten Änderungen passieren. Während die "Preview"-Deployments unter URLs nach diesem Muster erreichbar sind: `sus-fahrgemeinschaften-git-{BRANCH_NAME}-sus-ag.vercel.app`, ist die Produktionsumgebung immer unter der URL sus-fahrgemeinschaften.vercel.app erreichbar.

7 Bedienungsanleitung

The image shows a mobile app registration screen. At the top is a teal circular icon with a white car silhouette. Below it, the text 'Willkommen bei SuS-Fahrgemeinschaften' is displayed in bold. There are three input fields: 'E-Mail', 'Passwort', and 'Passwort wiederholen'. Below these is a teal 'Anmelden' button. A link 'Hast Du bereits ein Konto? Melde dich hier an.' is positioned below the button. The bottom of the screen features a dark teal footer with the 'SUS' logo, the text 'Schüler und Schülerinnen', and '© SuS AG'.

Abbildung 8 Registrieren

The image shows a mobile app login screen. At the top is a teal circular icon with a white car silhouette. Below it is the word "Login" in bold black text. There are two input fields: "E-Mail" and "Passwort". Below these is a teal "Login" button. Under the button is a link: "Hast Du noch kein Konto? Erstelle hier eins." At the bottom is a dark teal footer with the text "SUS", "Schüler und Schülerinnen", and "© SuS AG".

Abbildung 9 Login

Zu Beginn hat man die Möglichkeit, einen Account zu registrieren. Nach Angabe einer korrekten Mailadresse sowie eines Passworts mit mindestens 8 Zeichen, mindestens einem Großbuchstaben und mindestens einer Zahl kann man über "Anmelden" den Account anlegen. Danach wird man auf die Login-Seite weitergeleitet. Alternativ, wenn man schon einen Account hat, kann man auch direkt von der Registrieren-Seite zur Login-Seite gelangen. Von dort kann man sich dann mit seiner Mailadresse und dem Passwort einloggen.

The screenshot shows a mobile application interface for completing a profile. At the top right, there is a green button labeled "Speichern". The main heading in the center is "Bitte vervollständige Deine Profildaten". Below this is a large, light blue circular placeholder for a profile picture, with the text "Bild bearbeiten" underneath it. Further down are three white input fields with light blue borders, labeled "Vorname", "Nachname", and "Telefonnummer". At the bottom of the screen is a dark blue footer bar containing the "SUS" logo, the text "Schüler und Schülerinnen", and "© SUS AG".

Abbildung 10 Profil vervollständigen

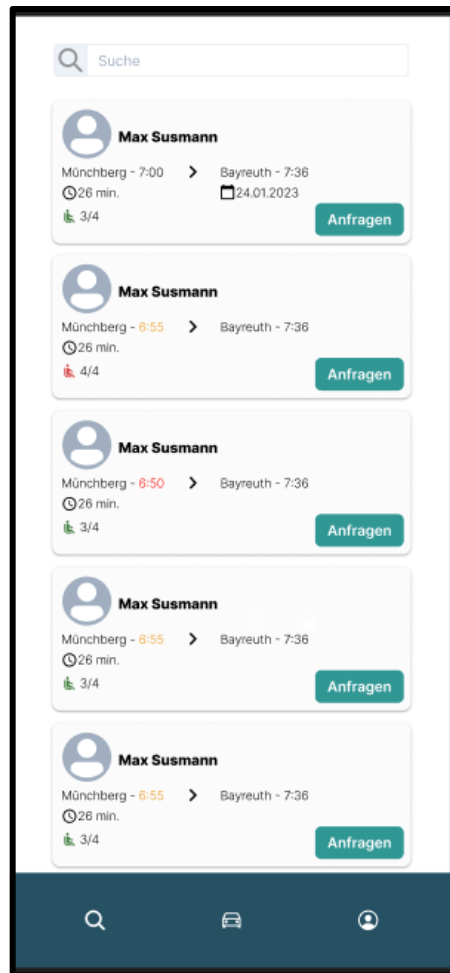


Abbildung 11 Fahrten suchen

Nach dem ersten Einloggen wird man aufgefordert, sein Profil zu vervollständigen. Hier kann man seinen Vor- und Nachnamen sowie eine Telefonnummer hinterlegen. Zukünftig soll es auch möglich sein, ein Profilbild zu hinterlegen. Über den Button “Speichern” wird dann das Profil angelegt. Danach wird man auf die Hauptseite weitergeleitet – eine Anzeige aller verfügbaren Fahrten. Hier hat man die Möglichkeit, einer Fahrt beizutreten. Dies geschieht über den Button “Anfragen”.

*Abbildung 12 Fahrt anzeigen*

The screenshot displays the 'Fahrt' (Trip) creation interface. At the top, there is a back button labeled '< Zurück' and the title 'Fahrt'. Below this, the form is organized into two columns. The left column contains 'Hinfahrt' (Departure) with a car icon and a time field set to '7:00 Uhr', and 'Preis / Fahrt' (Price / Trip) with a Euro symbol and a field set to '4,00 €'. The right column contains 'Ankunft ca.' (Arrival approx.) with a car icon and a time field set to '16:00 Uhr', and 'Sitzplätze' (Seats) with a car icon and a field set to '4'. Below these fields, the 'Treffpunkt' (Meeting point) is indicated with a location pin icon and a text field containing 'REWE parkplatz'. The 'Ankunftsort' (Destination) is also indicated with a location pin icon, a text field containing 'Musterstraße 22', and a link 'Eigenen Ankunftsort erstellen'. At the bottom of the form is a large green button labeled 'Erstellen'. The bottom of the screen features a dark blue navigation bar with three icons: a magnifying glass, a car, and a person.

Abbildung 13 Fahrt erstellen

Man wird nun zur detaillierten Fahrtenansicht weitergeleitet. Klickt man erneut auf “Anfragen”, so wird man der Fahrt hinzugefügt. Hat man sich anders entschieden, so kann man über den Button “Zurück” wieder zur Fahrtenübersicht. Über die Navigationsleiste werden dem Nutzer noch weitere Möglichkeiten geboten: Neben der Fahrtenübersicht (Suche) ist es möglich, über das Auto-Symbol eine eigene Fahrt anzulegen. Hier kann man die Abfahrt und Ankunftszeit sowie den Treffpunkt und den Ankunftsort angeben. Außerdem kann der Fahrer die verfügbaren Sitzplätze und den Preis pro Mitfahrer festlegen. Über “Erstellen” wird die Fahrt dann angelegt und anderen Nutzern angezeigt.



Abbildung 14 Profil anzeigen

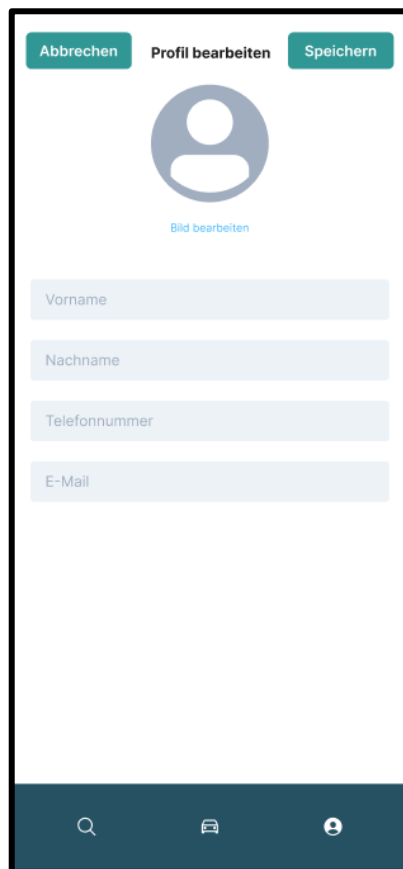


Abbildung 15 Profil bearbeiten

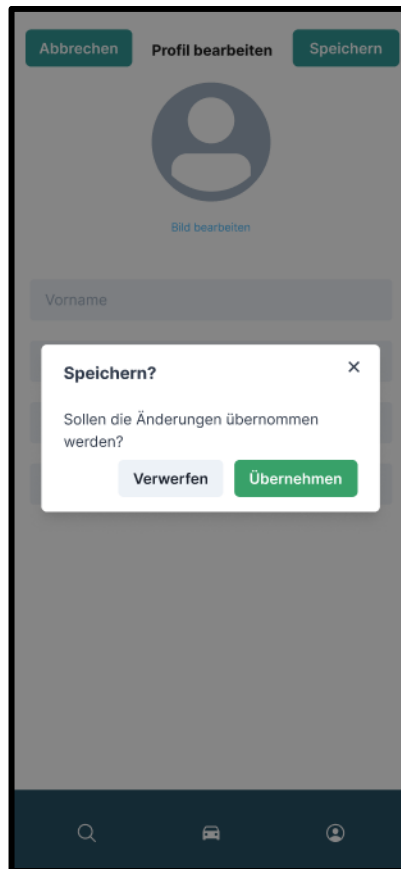


Abbildung 16 Dialog Änderungen speichern

Über das Profil-Icon in der Navigationsleiste gelangt man auf sein eigenes Profil. Hier hat man die Möglichkeit, sein eigenes Profil noch einmal zu bearbeiten oder wieder zurück zur Fahrtenübersicht zu gehen. Hat man sein Profil bearbeitet kann man die Änderungen speichern. Zur Sicherheit kommt noch eine Abfrage.

8 Fazit

8.1 Soll-/Ist-Vergleich

Derzeit werden Fahrgruppen meist in der ersten Berufsschulwoche in Person oder über Messaging-Dienste gebildet. Hierdurch entstehen Probleme, meist für Schüler ohne Führerschein bzw. Leute, die einen weiteren Weg zur Schule haben, da diese am ersten Tag noch keine Fahrgruppe haben, bei der diese mitfahren können.

Durch unsere „SuS-Fahrgruppen“-App soll es möglich sein jederzeit Fahrgruppen bzw. Mitfahrgelegenheiten zu finden.

8.2 Resümee

Im Zuge des Projektes konnten die Autoren wertvolle Erfahrungen bzgl. der Planung und Durchführung von Projekten sammeln. Dabei wurde besonders deutlich, von welcher großer Bedeutung stetige Kommunikation untereinander und Rücksprachen mit den Kunden für eine erfolgreiche Projektumsetzung ist. Außerdem konnten neue Erkenntnisse in Bezug auf das Einbinden und Nutzen von Frameworks gewonnen werden. Der T3-Stack erwies sich beispielsweise als sehr hilfreich, da er sich um das Routing kümmert und die Autoren sich somit auf die wesentlichen Komponenten der Anwendung konzentrieren konnten.

Abschließend kann man sagen, dass die Realisierung des Projektes nicht nur ein Mehrwert für die Schule und Schüler hat, sondern auch für die Autoren eine große Bereicherung war.

8.3 Ausblick

Zukünftig soll eine Historie der Fahrten im Profil angezeigt werden, um eventuell Fahrer wiederzufinden. Außerdem soll bei einer Anfrage zum Beitritt der Fahrt eine Mitteilung an den Fahrer geschickt werden, der dann entscheiden kann, ob der User mitfahren darf. Eventuell kann noch eine Chatfunktion hinzugefügt werden, die als dritte Kommunikationsmöglichkeit neben E-Mail und Telefonnummer existiert. Auch ist geplant, dass man dem Profil noch ein Bild hinzufügen kann. Außerdem fehlt noch die Funktion, ein fremdes Profil anzuzeigen.

A Anhang

A.1 Hardware

- Laptops (MacBook, Fujitsu Laptop)
- Tablet (zweiter Bildschirm)

A.2 Software

- Windows 10 – Betriebssystem
- MacOS – Betriebssystem
- Git – Versionsverwaltung
- GitHub Projects – Ticketsystem
- Discord – Sprachtausch / Bildschirm teilen
- Microsoft Teams – Sprachtausch / Bildschirm teilen
- Microsoft Word – Textverarbeitungssoftware zur Erstellung der Projektdokumentation
- Visual Studio Code – Entwicklungsumgebung
- IntelliJ IDEA – Entwicklungsumgebung
- Google Chrome – Webbrowser
- Vercel – Webserverhost
- PlanetScale – Datenbankhost

A.3 Personal

- 1x Projektleiter– Entwickler, Tester, Autor, Deployment, Code-Review
- 3x Anwendungsentwickler – Entwickler, Tester, Autor

A.4 Lastenheft

1. Visionen und Ziele

/LV10/ Die BS1-Bayreuth soll durch eine Webapp, den Schülern eine Plattform anbieten, auf der die Schüler*innen Fahrgemeinschaften bilden können. Dadurch wird der Stress am ersten Tag der Schüler vermindert, da sie leichter Fahrgemeinschaften bilden können.

2. Rahmenbedingungen

/LR10/ Die Webapp SuS-Fahrgemeinschaften ist eine Webapp, um Fahrgemeinschaften zu bilden.

/LR20/ Die Zielgruppe sind die Schüler, der BS1-Bayreuth.

3. Kontext und Überblick

/LK10/ Die Clients haben alle einen Internetbrowser.

/LK20/ Die Clients haben alle einen Internetzugriff in Form von WLAN oder Datenvolumen.

4. Funktionale Anforderungen

/LF10/ Die Webapp muss eine Datenbank auf einem Server haben, um darin die benötigten Daten zu speichern.

- /LF20/** Die Webapp soll eine responsive und mobile-first Webapp sein, auf dieser muss sich der Nutzer registrieren bzw. Anmelden können, die Konten sind self-managed.
- /LF30/** Die Webapp soll barrierefrei, ergonomisch und responsive sein.
- /LF40/** Die Webapp muss eine Anmeldeoberfläche haben, auf der man sich anmelden kann.
- /LF50/** Hat der Nutzer sich angemeldet, muss er in der Lage sein, Fahrer zu suchen oder eine Fahrt anzubieten. Hierfür kann der Mitfahrer Fahrten suchen und der Fahrer Fahrten erstellen.
- /LF60/** Die Webapp muss in der Grundversion, als Reiseziel immer die BS1-Bayreuth, Kerschensteinerstraße 6, 95488 Bayreuth haben.
- /LF70/** In der Webapp soll der Fahrer einen Festpreis für die Mitfahrer angeben können, die Zahlung erfolgt in Bar.
- /LF80/** In der Webapp muss man nach unten wischen können, um die Seite zu aktualisieren und somit die neusten Fahrten angezeigt zu bekommen.
- /LF90/** In der Webapp soll der Fahrer die Anzahl der Sitzplätze ändern können, standardmäßig sind 4 Sitzplätze vorgelegt.
- /LF100/** In der Webapp soll der Fahrer bestätigen können, ob der Antragsteller mitfahren darf oder nicht.
- /LF110/** Die Webapp muss eine Funktion haben, in der, der Nutzer eine Handynummer angeben kann, als Kontaktmöglichkeit.
- /LF120/** In der Webapp muss, man die Abfahrt 60 Minuten im Voraus buchen.

5. Qualitätsanforderungen

| Systemqualität | 4 | 3 | 2 | 1 |
|-------------------------------|---|---|---|---|
| /LQ10/ Funktionalität | | x | | |
| /LQ20/ Zuverlässigkeit | x | | | |
| /LQ30/ Benutzbarkeit | | | x | |
| /LQ40/ Effizienz | | | | x |
| /LQ50/ Änderbarkeit | | x | | |
| /LQ60/ Übertragbarkeit | | x | | |

A.5 Pflichtenheft

| Version | Autor | Quelle | Status | Datum | Kommentar |
|---------|------------|---|----------------|------------|------------------|
| 1.0 | David Luge | Webapp, um Fahrgemeinschaften zu bilden | In Bearbeitung | 25.01.2023 | Initiale Version |

1. Vision und Ziele

- /V10/(/LV10/)** Die Firma SuS AG als Auftraggeber, soll den Schüler*innen die Möglichkeit bieten Fahrgemeinschaften innerhalb einer Webapp zu bilden.

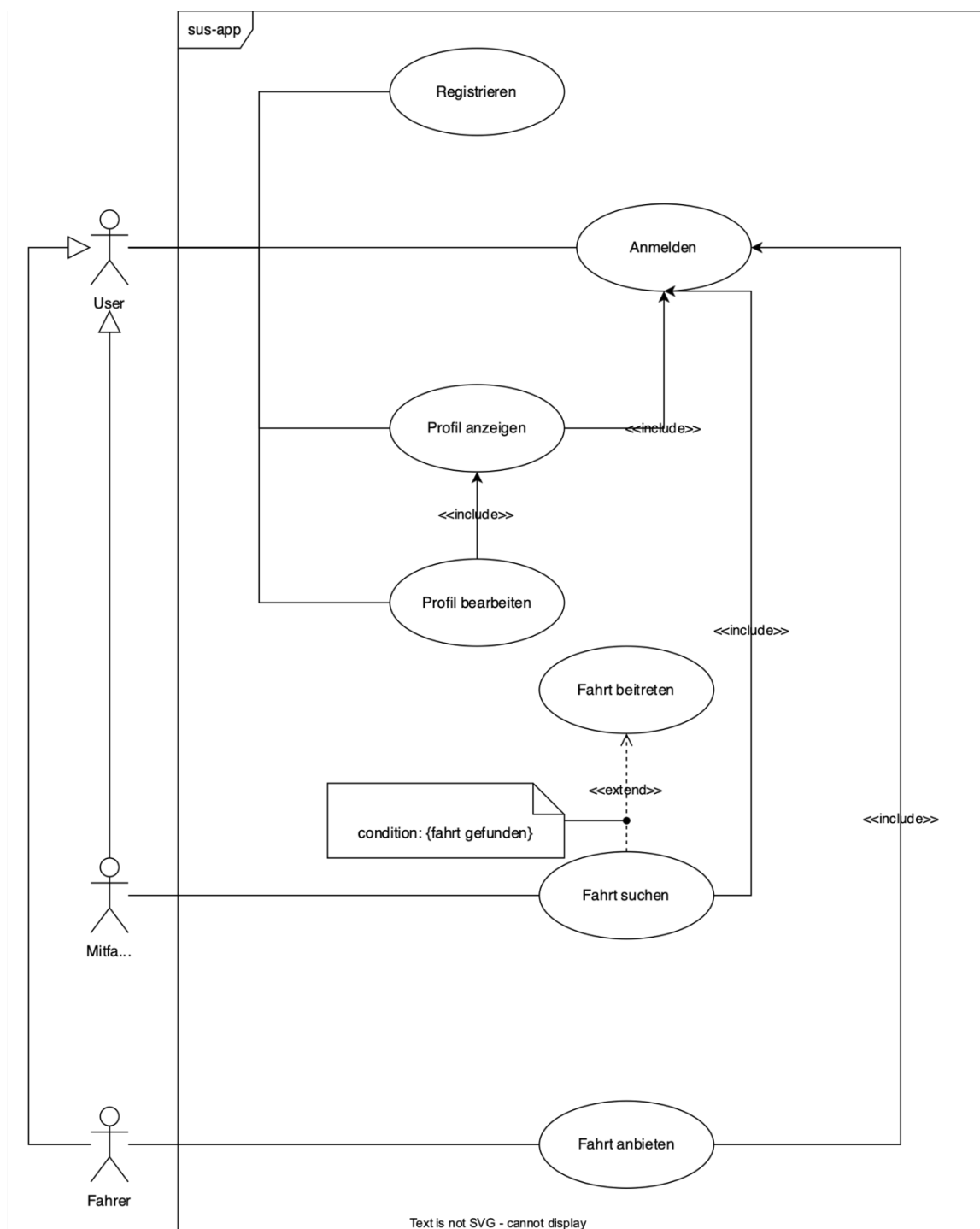
2. Rahmenbedingungen (geplante Realisierung)

- /R10/** Die Software benutzt Next.js, als Frontend-Framework, da bereits Vorwissen von React vorhanden ist, zudem ist es der aktuelle Industriestandard.
- /R20/** Die Software benutzt Chakra UI als Component-Library und Designgrundlage, da es ähnlich zu Material UI ist, es komplett Open Source ist und es eine kostenlose Figma Library gibt.
- /R30/** Für die Erstellung des Mockups benutzten wir, die SuS AG Figma, da bereits Vorwissen aus anderen Projekten vorhanden ist.
- /R40/** Wir benutzen Prisma als ORM, da wir so viel SQL wie möglich abstrahieren möchten, um die Persistenz Schicht der Applikation so simpel wie möglich zu halten, außerdem ist es möglich, leicht zwischen verschiedenen Datenbanksystemen zu wechseln.
- /R50/** Durch NextAuth können wir sicherstellen, dass unsere App die aktuellen Sicherheitsstandards einhält, da diese durch die Open-Source-Community GitHub stetig weiterentwickelt wird. Außerdem können wir hierdurch absichern, dass keine Fehler in der Authentifizierungslogik entstehen, da der Code bereits durch die Open Source Community überprüft wird.
- /R60/** tRPC benutzen wir, um die Kommunikation zwischen Front- und Backend so simpel wie möglich gestalten wollen ("KISS").
- /R70/** Die Webapp benutzt TypeScript, um Leichtsinnfehler durch die Typisierung zu vermeiden.
- /R80/** Als deployment Plattform der Webapp nutzen wir Vercel. Da in Zukunft noch einige Features hinzugefügt werden sollen, haben wir uns hierfür entschieden, da eine CI/CD Pipeline "eingebaut" ist. Eine CI/CD Pipeline ermöglicht es automatisch bei Änderungen die Webapp in der neuen Version zu veröffentlichen. Zudem ist der Service kostenlos.
- /R90/** Als Datenbank verwenden wir PlanetScale, da noch einige Features hinzugefügt werden sollen haben wir uns hierfür entschieden, da die Datenbank ein eingebautes Feature zur Migration von Daten hat, außerdem ist der Service kostenlos.

3. Kontext und Überblick

- /K10/ (/LK10/)** Die Clients haben alle einen Internetbrowser.
- /K20/(/LK20/)** Die Clients haben alle einen Internetzugang in Form von WLAN oder Datenvolumen.

4. Anwendungsfalldiagramm



Text is not SVG - cannot display
 Abbildung 17 Use-Case Diagramm Pflichtenheft

A.6 Use-Case-Diagramm

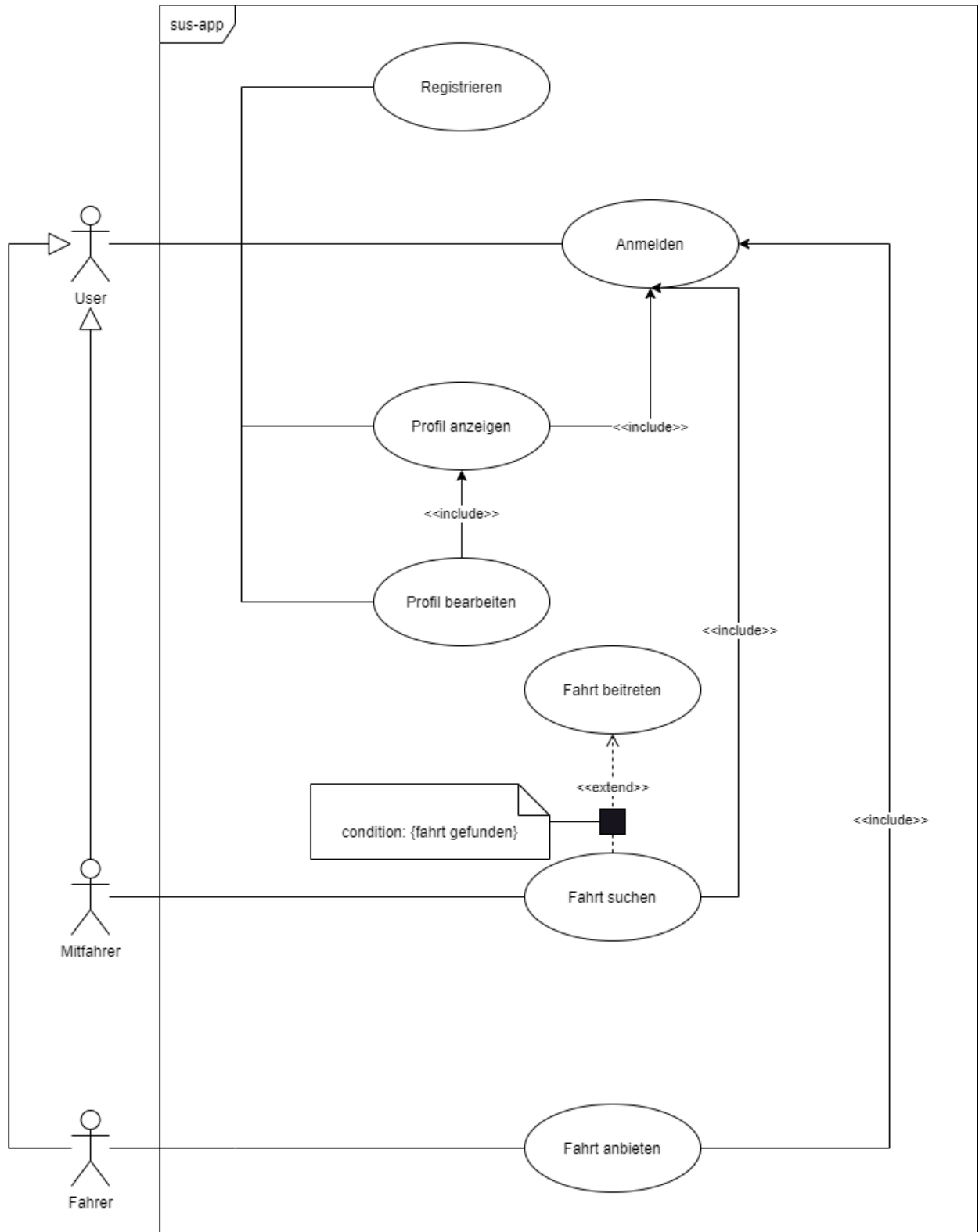


Abbildung 18 Use-Case Diagramm

A.7 Mock-Ups



Login

Du hast noch kein Konto? Erstelle [hier](#) eins.

SUS

Schüler und Schülerinnen
© SUS AG

Abbildung 19 Mockup Login-Seite



Willkommen bei SUS-Fahrgemeinschaften

Hast Du bereits ein Konto? Melde dich [hier](#) an.

SUS

Schüler und Schülerinnen
© SuS AG

Abbildung 20 Mockup Registrieren Seite

Speichern

**Bitte vervollständige Deine
Profildaten**[Bild bearbeiten](#)

Vorname

Nachname

Telefonnummer

SUSSchüler und Schülerinnen
© SuS AG*Abbildung 21 Mockup Profil vervollständigen Seite*

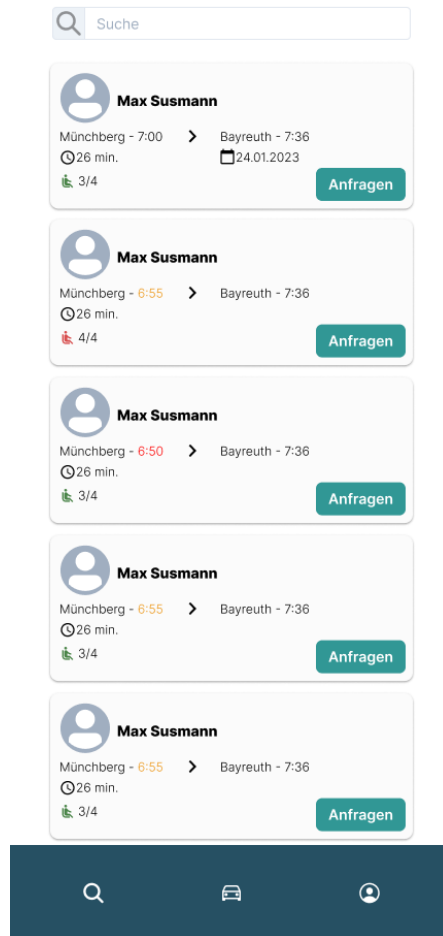


Abbildung 22 Mockup Fahrten suchen Seite

[< Zurück](#)

Fahrt


Datum

24.01.2023


Hinfahrt

6:55 Uhr


Ankunft ca.

16:00 Uhr

€

Preis / Fahrt

3,00€

Sitzplätze

3/4


Treffpunkt

Pendler Parkplatz
Münchberg Süd


Ankunftsort

BS1 Bayreuth

Anfragen








Abbildung 23 Mockup Fahrt anzeigen Seite

[< Zurück](#)[Fahrt](#)

Hinfahrt

7:00 Uhr

Ankunft ca.

16:00 Uhr

€

Preis / Fahrt

4,00 €

Sitzplätze

4

Treffpunkt

REWE parkplatz

Ankunftsart
Eigene Ankunftsart erstellen

Musterstraße 22

Erstellen



Abbildung 24 Mockup Fahrt erstellen Seite

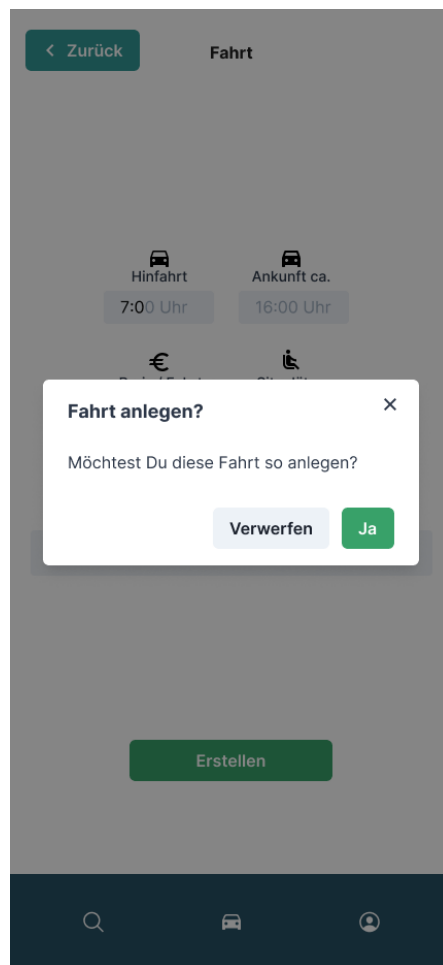


Abbildung 25 Mockup Fahrt anlegen Dialog Seite



Abbildung 26 Mockup eigenes Profil anzeigen Seite

Abbrechen

Profil bearbeiten

Speichern



[Bild bearbeiten](#)



Abbildung 27 Mockup Profil bearbeiten Seite

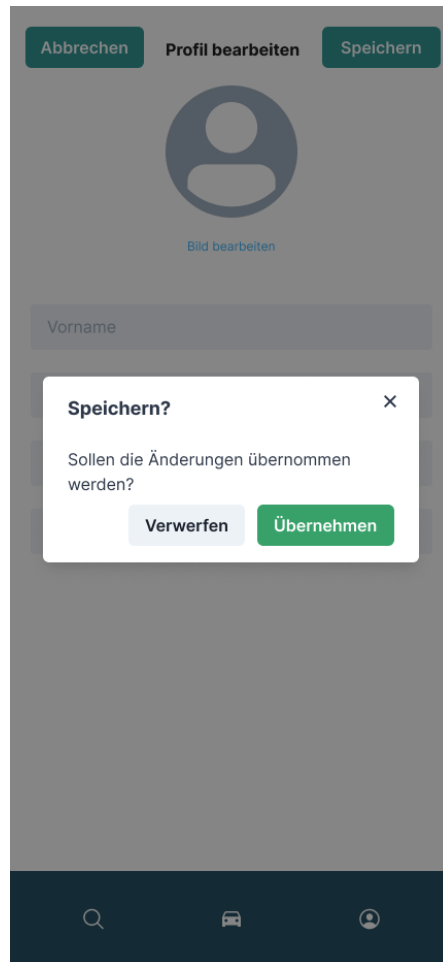


Abbildung 28 Mockup Profil bearbeiten Dialog Seite

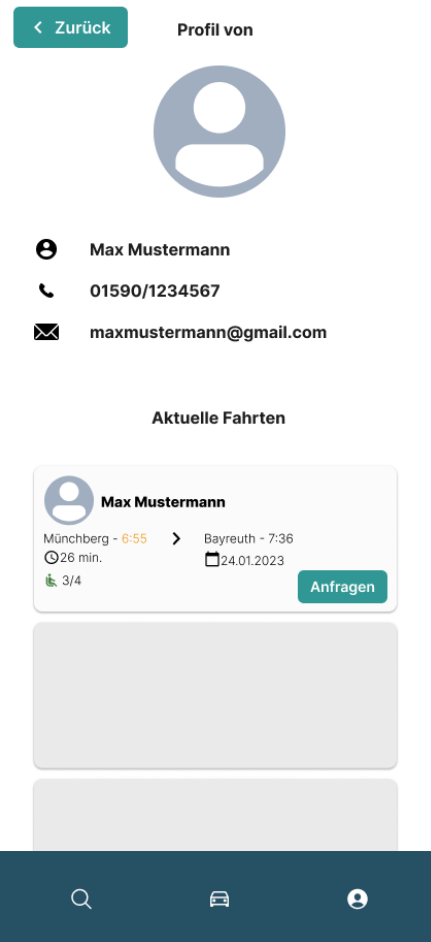


Abbildung 29 Mockup fremdes Profil anzeigen Seite

A.8 Entity-Relationship-Modell

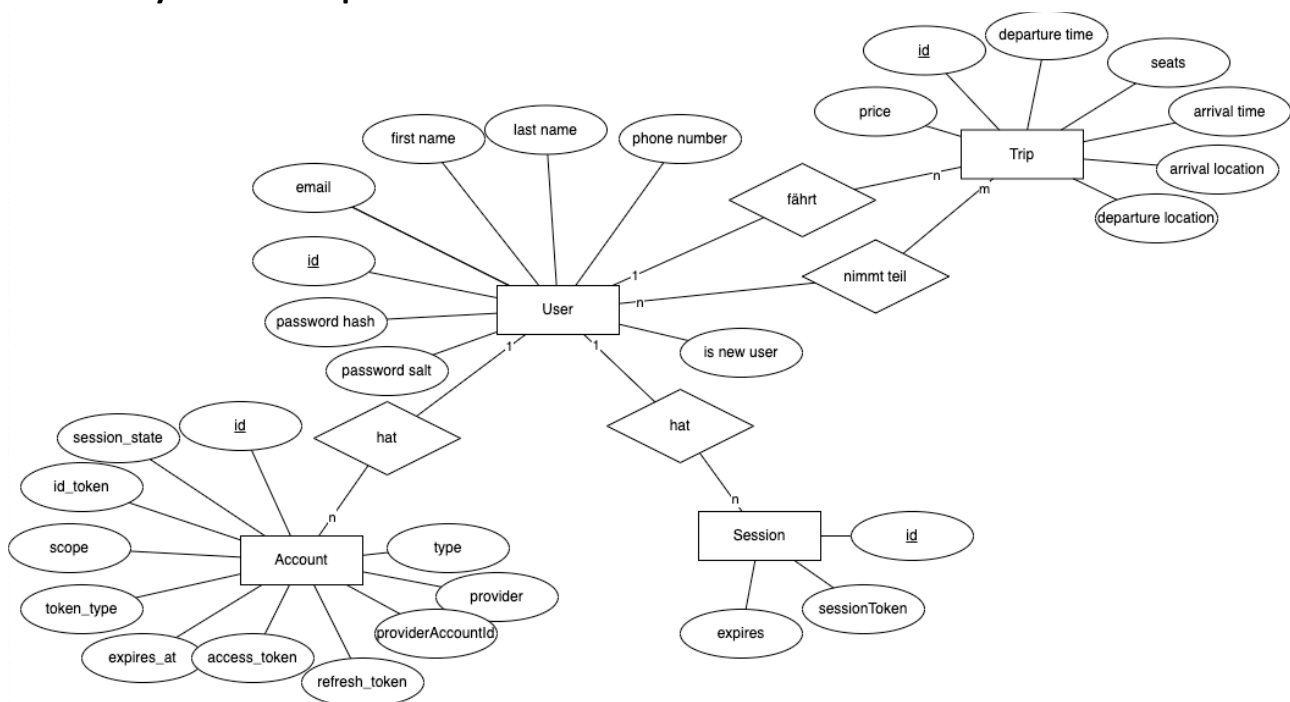


Abbildung 30 ER-Modell der Datenbank

A.9 Sequenzdiagramm

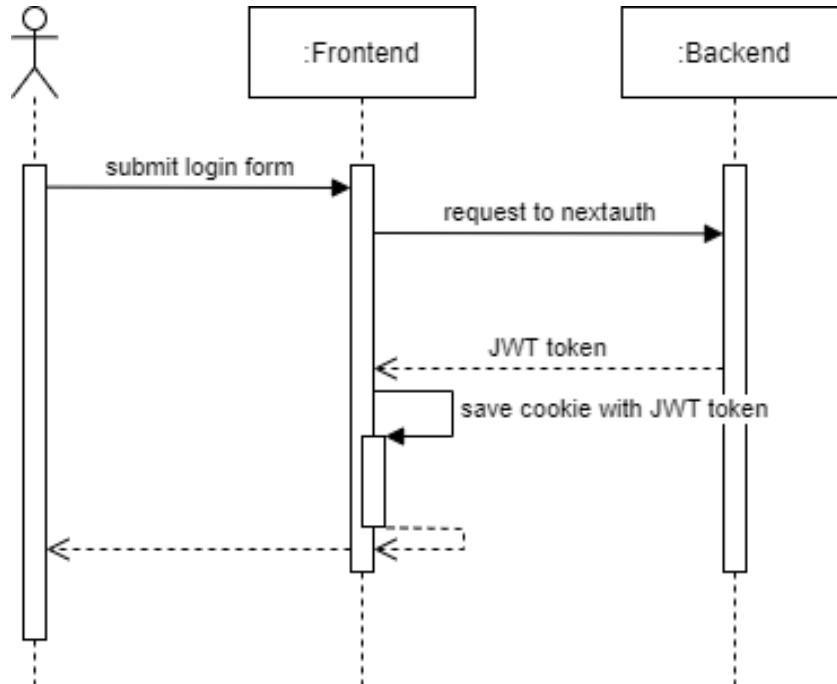


Abbildung 31 Sequenzdiagramm des Logins

A.10 Test-Protokoll Black-Box-Test

- Test T01: Registrierung mit nicht gültiger E-Mail-Adresse
 1. Die URL <https://sus-fahrgemeinschaften.vercel.app/register> wurde aufgerufen
 2. Die Seite wurde geladen
 3. Es wurde in das Feld 'E-Mail-Adresse' die E-Mail-Adresse 'keineEmailAdresse' eingegeben
 4. Es wurde in das Feld 'Passwort' 'Ads;lkfajsdio123' eingegeben
 5. Es wurde auf den Button 'Anmelden' geklickt
 6. Der Nutzer wurde nicht auf die Login-Seite weitergeleitet
- Test T02: Registrierung mit gültiger E-Mail-Adresse
 1. Die URL <https://sus-fahrgemeinschaften.vercel.app/register> wurde aufgerufen
 2. Die Seite wurde geladen
 3. Es wurde in das Feld 'E-Mail-Adresse' die E-Mail-Adresse 'stuchlyf@stuchlyf.dev' eingegeben
 4. Es wurde in das Feld 'Passwort' 'Ads;lkfajsdio123' eingegeben
 5. Es wurde auf den Button 'Anmelden' geklickt
 6. Der Nutzer wurde auf die Login-Seite weitergeleitet
- Test T03: Login mit ungültigen Daten
 1. Die URL <https://sus-fahrgemeinschaften.vercel.app/auth/signin> wurde aufgerufen
 2. Die Seite wurde geladen
 3. Es wurde in das Feld 'E-Mail-Adresse' die E-Mail-Adresse 'keineEmailAdresse' eingegeben
 4. Es wurde in das Feld 'Passwort' 'Ads;lkfajsdio123' eingegeben
 5. Es wurde auf den Button 'Anmelden' geklickt
 6. Der Nutzer wurde nicht auf die Trips-Seite weitergeleitet
- Test T04: Login mit gültigen Daten
 1. Die URL <https://sus-fahrgemeinschaften.vercel.app/auth/signin> wurde aufgerufen
 2. Die Seite wurde geladen

3. Es wurde in das Feld 'E-Mail-Adresse' die E-Mail-Adresse 'stuchlyf@stuchlyf.dev' eingegeben
 4. Es wurde in das Feld 'Passwort' 'Ads;lkfajsdio123' eingegeben
 5. Es wurde auf den Button 'Anmelden' geklickt
 6. Der Nutzer wurde auf die Complete-Profile-Seite weitergeleitet
- Test T05: Komplettieren des Profils
 1. Der Nutzer meldete sich zum ersten Mal mit seinem Konto an
 2. Der Nutzer wurde auf die <https://sus-fahrgemeinschaften.vercel.app/auth/signin> weitergeleitet
 3. In das Feld 'Vorname' wurde der Vorname 'Florian' eingetragen
 4. In das Feld 'Nachname' wurde der Vorname 'Stuchly' eingetragen
 5. In das Feld 'Telefonnummer' wurde die Telefonnummer '09280 207119' eingetragen
 6. Der Button 'Speichern' wurde angeklickt
 7. Der User wurde auf die Trips-Seite weitergeleitet
 - Test T06: Erstellung einer Fahrt
 1. Der Nutzer meldete sich an.
 2. Der Nutzer hat auf den Fahrt-Erstellen-Button in der Mitte der unteren Navigationsleiste geklickt
 3. Der Nutzer wurde auf die Seite <https://sus-fahrgemeinschaften.vercel.app/trips/create> weitergeleitet
 4. Es wurde in das Feld 'Datum' der Wert '19.03.2023' eingetragen
 5. Es wurde in das Feld 'Hinfahrt' der Wert '17:55' eingetragen
 6. Es wurde in das Feld 'Ankunft ca.' der Wert '20:54' eingetragen
 7. Es wurde in das Feld 'Preis / Person' der Wert '4.00' eingetragen
 8. Es wurde in das Feld 'Sitzplätze' der Wert '4' eingetragen
 9. Es wurde in das Feld 'Treffpunkt' der Wert 'REWE Parkplatz Hof' eingetragen
 10. Es wurde in das Feld 'Ankunftsort' der Wert 'Staatliche Berufsschule 1 Bayreuth' eingetragen
 11. Es wurde der Button 'Erstellen' geklickt
 - Test T07: Editieren des Profils
 1. Der Nutzer meldete sich an
 2. Der Nutzer hat auf den Profil-Button rechts in der unteren Navigationsleiste geklickt
 3. Der Nutzer wurde auf die <https://sus-fahrgemeinschaften.vercel.app/profiles/me/edit> weitergeleitet
 4. Es wurde in das Feld 'Vorname' der Wert 'Max' eingetragen
 5. Es wurde in das Feld 'Nachname' der Wert 'Mustermann' eingetragen
 6. Es wurde in das Feld 'Telefonnummer' der Wert '+49 151 3242313' eingetragen
 7. Es wurde in das Feld 'E-Mail' der Wert 'florian.stuchly@gmail.com' eingetragen
 8. Es wurde der 'Speichern' Button geklickt
 - Test T08: Einer Fahrt beitreten
 1. Der Nutzer meldete sich an
 2. Der Nutzer hat auf den Suche-Button links in der unteren Navigationsleiste geklickt
 3. Der Nutzer wurde auf die Seite <https://sus-fahrgemeinschaften.vercel.app/trips> weitergeleitet
 4. Der Nutzer klickt auf den Anfrage-Button von einer der Fahrten
 5. Der Nutzer wurde auf die Seite [https://sus-fahrgemeinschaften.vercel.app/trips/\[id\]](https://sus-fahrgemeinschaften.vercel.app/trips/[id]) weitergeleitet
 6. Der Nutzer klickt auf den 'Anfragen'-Button

A.11 Screenshots ...Quellcode

```

1  import {Box, Icon, Link} from "@chakra-ui/react";
2  import NextLink from "next/link";
3  import {useRouter} from "next/router";
4  import React, {useMemo} from "react";
5  import {IoCar, IoCarOutline, IoPersonCircle, IoPersonCircleOutline, IoSearch, IoSearchOutline,} from "react-icons/io5";
6
7  const BottomNavigation: React.FC = () => {
8    const router = useRouter();
9
10   const currentSite = useMemo<'search-trip' | 'create-trip' | 'profiles' | undefined>(() => {
11     if (router.pathname.startsWith('/trips/create')) {
12       return 'create-trip';
13     } else if (router.pathname.startsWith('/trips')) {
14       return 'search-trip';
15     } else if (router.pathname.startsWith('/profiles')) {
16       return 'profiles'
17     } else {
18       return undefined;
19     }
20   }, [router.pathname]);
21
22   return (
23     <Box className="grid grid-cols-3 h-[5.875rem] w-full bg-cyan-900 place-content-stretch justify-items-stretch">
24       <Link as={NextLink} href="/trips" className="flex justify-center items-center">
25         <Icon as={currentSite === 'search-trip' ? IoSearch : IoSearchOutline} color="white" boxSize={`1.5rem`} />
26       </Link>
27       <Link as={NextLink} href="/trips/create" className="flex justify-center items-center">
28         <Icon as={currentSite === 'create-trip' ? IoCar : IoCarOutline} color="white" boxSize={`1.5rem`} />
29       </Link>
30       <Link
31         as={NextLink}
32         href="/profiles/me"
33         className="flex justify-center items-center"
34       >
35         <Icon as={currentSite === 'profiles' ? IoPersonCircle : IoPersonCircleOutline} color="white"
36           boxSize={`1.5rem`} />
37       </Link>
38     </Box>
39   );
40 };

```

Abbildung 32 Quellcode des BottomNavigation Components

```

1  import { z } from "zod";
2  import { createTRPCRouter, publicProcedure } from "../trpc";
3  import { pbkdf2Sync, randomBytes } from 'crypto';
4
5  const validateEmail = (email: string | null | undefined) => {
6    if (!email) return false;
7
8    const re = /^[^\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$/g;
9    return re.test(email);
10 }
11
12
13 const validatePassword = (password: string | null | undefined) => {
14   if (!password) return false;
15
16   const re = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)[a-zA-Z\\d]{8,}$/;
17   return re.test(password);
18 }
19
20 export const registerRouter = createTRPCRouter({
21   register: publicProcedure
22     .input(z.object({ email: z.string(), password: z.string() }))
23     .mutation(({ input, ctx }) => {
24       // 1. validate input
25       if (!validateEmail(input.email)) throw new Error('Invalid email')
26       if (!validatePassword(input.password)) throw new Error('Invalid password')
27
28       // 2. hash password
29       const salt = randomBytes(16).toString('hex')
30       const hash = pbkdf2Sync(
31         input.password,
32         salt,
33         10000,
34         64,
35         'sha512'
36       )
37       .toString('hex')
38
39       // 3. save user to db
40       return ctx.prisma.user.create({
41         data: {
42           email: input.email,
43           passwordHash: hash,
44           passwordSalt: salt,
45         }
46       })
47     })
48 })

```

Abbildung 33 Quellcode Hashing und Validierung des Passworts

```

1 // This is your Prisma schema file,
2 // learn more about it in the docs: https://pris.ly/d/prisma-schema
3
4 generator client {
5   provider      = "prisma-client-js"
6   previewFeatures = ["ReferentialIntegrity"]
7 }
8
9 datasource db {
10  provider      = "mysql"
11  // NOTE: When using postgresql, mysql or sqlserver, uncomment the @db.Text annotations in model Account below
12  // Further reading:
13  // https://next-auth.js.org/adapters/prisma#create-the-prisma-schema
14  // https://www.prisma.io/docs/reference/api-reference/prisma-schema-reference#string
15  url           = env("DATABASE_URL")
16  relationMode  = "prisma"
17 }
18
19 model Trip {
20   id           String   @id @default(cuid())
21   driverId     String
22   departureTime DateTime
23   departureLocation String
24   arrivalTime  DateTime
25   arrivalLocation String
26   seats        Int
27   price        Float
28   passengers   User[]   @relation("Passengers")
29   driver       User      @relation(fields: [driverId], references: [id], onDelete: Cascade)
30 }
31
32 // Necessary for Next auth
33 model Account {
34   id           String   @id @default(cuid())
35   userId       String
36   type         String
37   provider     String
38   providerAccountId String
39   refresh_token String? @db.Text
40   access_token  String? @db.Text
41   expires_at    Int?
42   token_type    String?
43   scope         String?
44   id_token      String? @db.Text
45   session_state String?
46   user         User      @relation(fields: [userId], references: [id], onDelete: Cascade)
47
48   @@unique([provider, providerAccountId])
49 }
50
51 model Session {
52   id           String   @id @default(cuid())
53   sessionToken String   @unique
54   userId       String
55   expires      DateTime
56   user         User      @relation(fields: [userId], references: [id], onDelete: Cascade)
57 }
58
59 model User {
60   id           String   @id @default(cuid())
61   email        String?   @unique
62   passwordHash String?
63   passwordSalt String?
64   firstname    String?
65   lastname     String?
66   phoneNumber  String?   @unique
67   accounts     Account[]
68   sessions     Session[]
69   tripsAsDriver Trip[]
70   tripsAsPassenger Trip[] @relation("Passengers")
71   isNewUser    Boolean   @default(true)
72 }

```

Abbildung 34 Definition des Datenbankschemas in der Prisma-eigenen Schemasprache

```

12 export default function SignIn() {
13   const router = useRouter();
14   const meQuery = api.profile.me.useQuery(undefined, {
15     enabled: false
16   });
17   const [email, setEmail] = useState<string>();
18   const [password, setPassword] = useState<string>();
19
20   const handleEmailChange = useCallback<ChangeEventHandler<HTMLInputElement>>((e) => {
21     const value = e.currentTarget.value;
22     setEmail(value);
23   }, [setEmail]);
24
25   const handlePasswordChange = useCallback<ChangeEventHandler<HTMLInputElement>>((e) => {
26     const value = e.currentTarget.value;
27     setPassword(value);
28   }, [setPassword]);
29
30   const handleSubmit = useCallback<MouseEventHandler<HTMLButtonElement> & FormEventHandler<HTMLFormElement>>(async (e) => {
31     e.preventDefault()
32     if (!email || !password) return;
33     const signInResponse = await signIn('credentials', {
34       password,
35       email,
36       redirect: false
37     });
38
39     if(signInResponse?.ok) {
40       await meQuery.refetch().then(async v => {
41         v.data?.isNewUser ? await router.push('/complete-profile') : await router.push('/')
42       }).catch(console.error);
43     }
44   }, [email, meQuery, password, router])

```

Abbildung 35 Teil 1 des Codes der Login Seite

```

46  return (
47    <Box className="h-full w-full grid grid-rows-layout">
48      <Box>
49        <Box className="mx-auto flex">
50          <Image
51            src={logo as string}
52            alt=""
53            className="mx-auto my-[4.938rem] h-[8rem] w-[8rem]"
54          />
55        </Box>
56        <Box className="justify-center text-center">
57          <Text className="text-4xl font-extrabold">Login</Text>
58        </Box>
59
60        <Box className=" m-5 mx-auto w-56 justify-center content-center">
61          <form onSubmit={handleSubmit}>
62            <FormControl>
63              <Input
64                type="email"
65                placeholder="E-Mail"
66                className="my-2"
67                value={email}
68                onChange={handleEmailChange}
69              />
70              <Input
71                type="password"
72                placeholder="Passwort"
73                className="my-2"
74                value={password}
75                onChange={handlePasswordChange}
76              />
77
78              <Button
79                type="submit"
80                colorScheme="teal"
81                size="md"
82                className="mx-7 w-[10.625rem]"
83                onClick={handleSubmit}
84                onSubmit={handleSubmit}
85              >
86                Login
87              </Button>
88            </FormControl>
89          </form>
90        </Box>
91        <Box className=" mx-auto justify-center text-center text-xs">
92          <Text>
93            Hast Du noch kein Konto? Erstelle {" "}
94            <Link as={NextLink} color="teal.500" href="/register">
95              hier{" "}
96            </Link>
97            eins.
98          </Text>
99        </Box>
100      </Box>
101
102      <Box>
103        <Footer />
104      </Box>
105    </Box>
106  );
107 }

```

Abbildung 36 Teil 2 des Codes der Login Seite