IFS INSTITUTE FOR SOFTWARE

OST Ostschweizer Fachhochschule

City Trip Planner: Kurztrip-Planer für Fussgänger

Studienarbeit

Studiengang Informatik

OST – Ostschweizer Fachhochschule

Campus Rapperswil-Jona

Herbstsemester 2021

Autoren:	Lukas Leuenberger
	Jan Ruch
Betreuer:	Prof. Stefan Keller
	Nicola Jordan
Projektpartner:	Schweiz Tourismus
	Markus Dittli

Abstract

Stadtrundtrips in der Schweiz erfreuen sich immer grösserer Beliebtheit und es gibt bislang kein passendes Produkt, welches die individuelle Planung als Dienst anbietet. Die Applikation City Trip Planner erstellt individuelle Rundtrips im städtisch-urbanen Raum für Fussgänger:innen. Sie können persönliche Präferenzen wie Kategorie der Interessen und maximale Laufdistanz eingeben, woraufhin ein passender Stadtrundtrip erstellt wird. Dieses Projekt ist eine Kooperation mit Schweiz Tourismus.

Das Ziel ist es ein Minimal Viable Product einer responsiven Webapplikation, mit welcher Routen intuitiv geplant und später auf einem internetfähigen Mobilgerät abgerufen werden können, zu entwickeln. Um die Attraktivität zu steigern, werden zusätzliche Informationen zu den jeweiligen Stationen einer Tour angezeigt. So können Benutzer:innen ganz einfach Städte in der Schweiz erkunden.

Um möglichst schnell ein passendes Resultat zu finden, verwendet die Applikation drei Suchstrategien. Sie können konfigurativ auf die Bedürfnisse des Betreibers angepasst werden, um den Anwendungszweck zu erfüllen. Sie sind dazu optimiert in den Bereichen von kurzen, mittleren und langen Distanzen sowie wenigen, einigen und vielen Punkten eine passende Route, die den Kriterien der Benutzer:innen entspricht, zu finden. Für die Umsetzung dieses Projektes wurden folgende Technologien eingesetzt: Python, FastAPI, Angular, PostgreSQL, OpenStreetMap und OSRM.

Management Summary

Ausgangssituation

Wegfindung ist ein gut erforschtes Gebiet in den Bereichen Verkehrsleitung und individuelle Routenplanung für Autos. Es gibt diverse Dienste von namhaften Firmen, welche dieses Problem recht gut und unter Einbezug von aktuellem Verkehrsaufkommen und Ereignissen lösen. Auch für die Planung von Rundtouren gibt es bereits Lösungen, um beispielsweise individuelle Motorradtouren zu planen. Dazu kann man persönliche Präferenzen eingeben und sich bequem eine Route berechnen lassen.

Es gibt bislang aber keine kommerzielle Softwarelösung, welche sich auf das Planen von Stadtrundtouren spezialisiert hat. Die Schweiz gilt als attraktive Feriendestination im In- und Ausland. Nebst den weltbekannten Alpen und der Idylle besitzt die Schweiz auch viele interessante Städte mit historischem Hintergrund und anderen Attraktionen. Die Aufgabe dieser Studienarbeit ist es, anhand von OpenStreetMap-Daten Rundtrips für Tourist:innen aus dem In- und Ausland in Schweizer Städten zu erstellen.

Dabei liegen die persönlichen Interessen und eine einfache Bedienung im Vordergrund. Zusätzlich sollen Bilder und Rundtrip-Details zu den einzelnen Stationen verfügbar sein, welche bei einer herkömmlichen Stadtrundtour nicht oder erst vor Ort verfügbar sind. Somit kann den Benutzer:innen ein individuelles Erlebnis geschaffen werden, welches auf ihre persönlichen Bedürfnisse zugeschnitten ist.

Ergebnissituation

In der vorliegenden Arbeit konnte ein Minimal Viable Product geschaffen werden, das die in der Aufgabenstellung genannte Funktionalität bietet. Benutzer:innen können zuhause passende Stadtrundtrips in der gesamten Schweiz erstellen und diese bequem auf einem internetfähigen Mobilgerät abrufen.

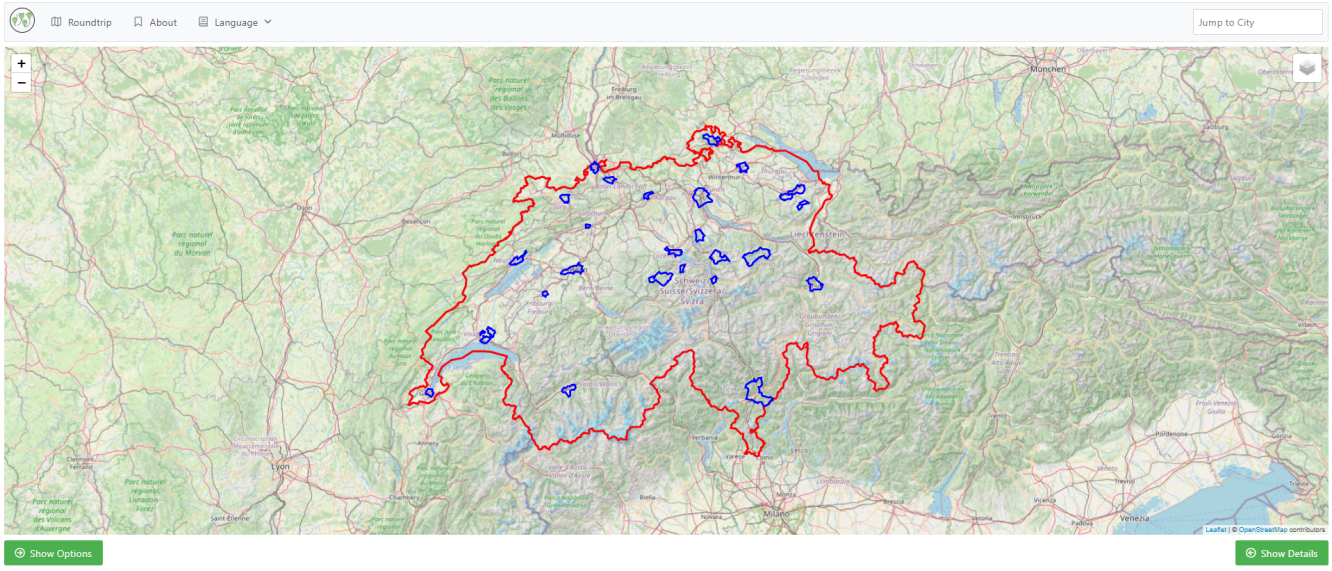


Abbildung 1. Überblickseite zur Planung eines Stadtrundtrips mit individualisierbaren, hervorgehobenen Kantonshauptorten

Es wurden zehn unterschiedliche Routing Engines in Betracht gezogen und evaluiert. Eine Engine muss unter anderem Routen für Tourist:innen zu Fuss im städtisch-urbanen Raum erstellen können. Das Resultat ist auf folgenden Bildern zu sehen. Die Applikation ist webbasiert und braucht keine zusätzliche Software auf dem Gerät der Benutzer:innen.

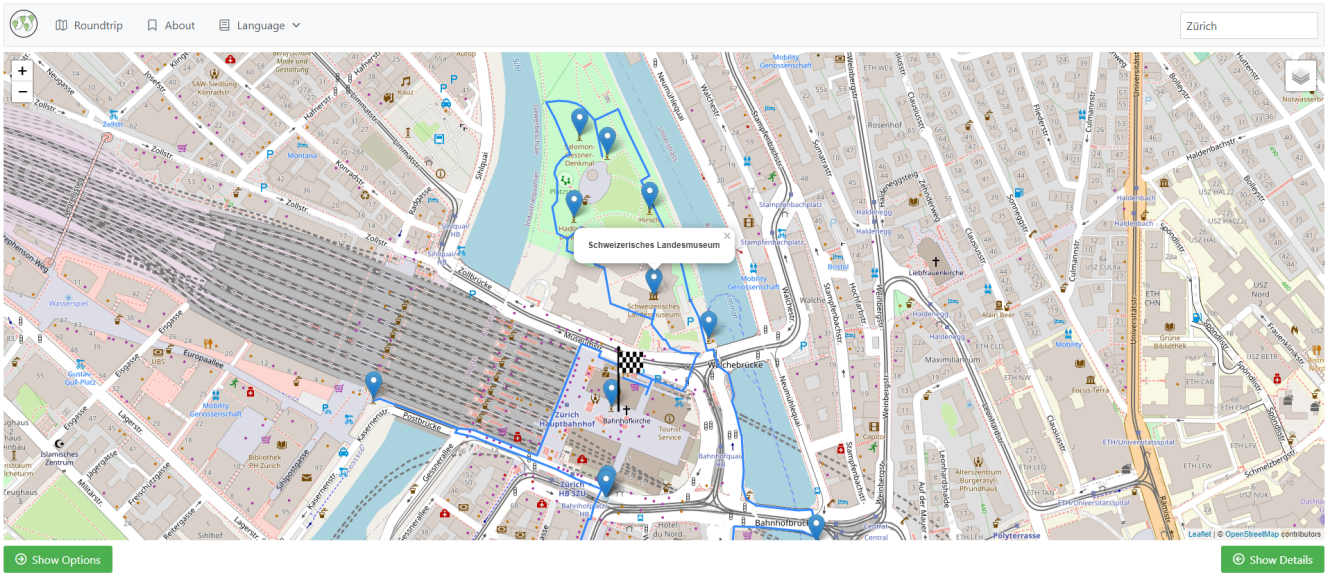


Abbildung 2. Rundtrip um den Zürcher Hauptbahnhof mit markiertem Start- / Zielpunkt und Namensanzeige beim Selektieren eines POI

Auf den Rundtrips können, mit dem eigenen Mobilgerät, interessante Details zu der selektierten Route und den POIs abgerufen werden. Falls vorhanden, werden Bilder und Links zu weiterführenden Informationen angezeigt. Der entsprechende Wikipedia-Artikel wird direkt in der gewählten Sprache der Benutzer:innen verlinkt. Mittels QR-Code kann der Rundtrip einfach geteilt oder übertragen werden und ist somit jederzeit auf internetfähigen Geräten abrufbar. Für die Umsetzung dieses Projektes wurden folgende Technologien eingesetzt: Python, FastAPI, Angular, PostgreSQL, OpenStreetMap und OSRM.

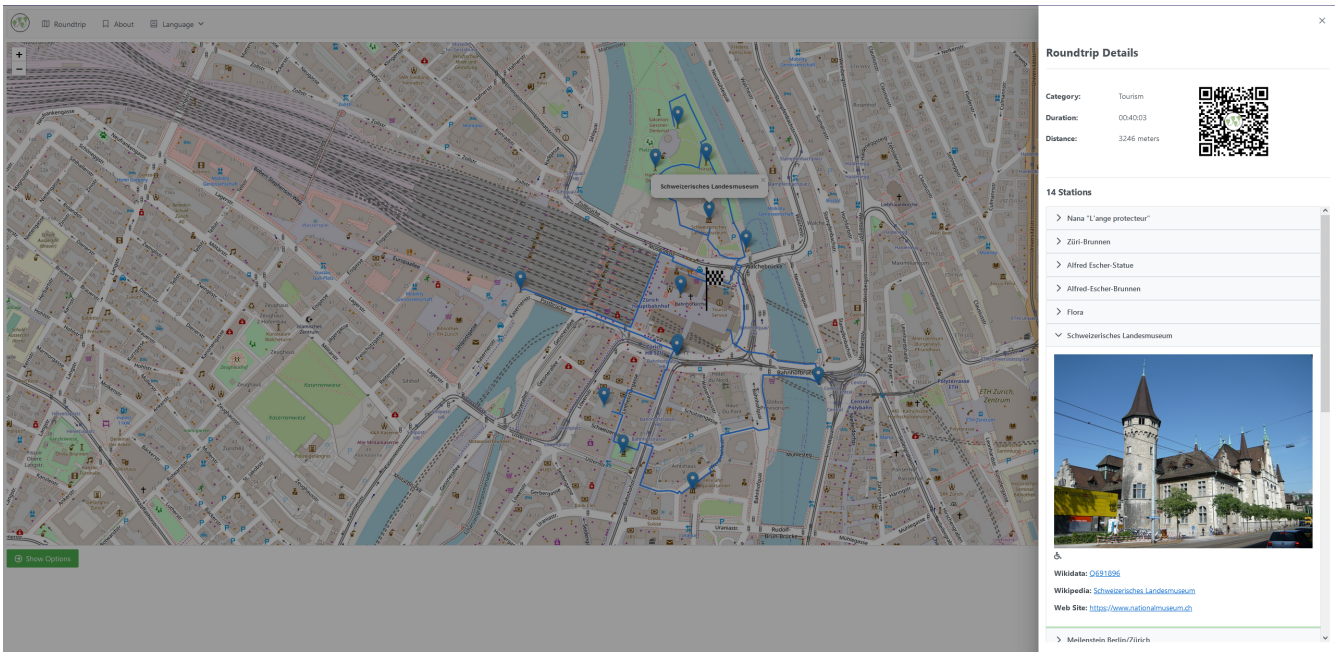


Abbildung 3. Rundtrip-Detailansicht mit sämtlichen Informationen zum Rundtrip, QR-Code, Bilder der POIs und weiterführenden Links

Erweiterungen

Das entwickelte Minimal Viable Product besitzt einen grossen Spielraum für weitere Funktionen. Als erster wichtiger Schritt wäre eine Kampagne über das Verhalten und die Wünsche der Benutzer:innen angebracht. Sie kann wichtige Erkenntnisse zur Bedienung und zu den häufigsten verwendeten Funktionalitäten bringen. Daraus lassen sich weitere Massnahmen zur Optimierung der Algorithmen ableiten.

Eine weitere Funktionalität wie das Planen von Besichtigungstouren von einem Start- zu einem Endpunkt, ist eine attraktive Erweiterung. Dadurch sind auch Kombinationen von Trips und Rundtrips möglich, was bei mehrtägigen Stadtreisen häufig der Fall ist.

Ein Mehrwert für die Benutzer:innen würde ein eigenes Benutzerprofil darstellen, in dem persönliche Interessen erfasst werden könnten. Basierend auf diesen Präferenzen könnte man die gesamte Schweiz bereisen und genau die Attraktionen besuchen, welche auf die Benutzer:innen passen. Für Touristen aus dem Ausland wäre eine Priorisierung der Top 10 Attraktionen einer Stadt interessant. Sie könnten basierend auf unterschiedlichen Metriken berechnet und angeboten werden.

Ein weiterer grosser Nutzen kann die Optimierung eines Routing-Profils darstellen. Denkbar ist hier ein Profil für Menschen im Rollstuhl oder für Familien mit Kinderwagen. So sollen Treppen, grosse Kreuzungen oder Hauptstrassen vermieden werden.

Sämtliche Features basieren auf den OpenStreetMap-Daten. Es wäre eine interessante Herausforderung nicht nur die Schweiz, sondern auch weitere Länder an das System anzuschliessen.

Danksagungen

An dieser Stelle bedanken wir uns bei all denjenigen, welche uns während dieser Studienarbeit unterstützt und motiviert haben.

An erster Stelle gebührt unser Dank Prof. Stefan Keller, der unsere Studienarbeit als Betreuer begleitet hat und sie bewerten wird. Wir danken herzlich für die aufgebrachte Zeit, die konstruktive Kritik und die zahlreichen Ideen und Anregungen. Wir konnten von seiner Erfahrung profitieren und uns neues Wissen aneignen.

Wir danken auch Nicola Jordan, welcher uns technisch und beratend unterstützt hat. Seine Inputs und Anregungen waren bei der Implementierung des Endprodukts dieser Studienarbeit sehr wertvoll.

Ein besonderer Dank wollen wir auch Markus Dittli von Schweiz Tourismus aussprechen. Wir sind dankbar für die gelungene Zusammenarbeit bei der Anforderungsanalyse und der Priorisierung der Features. Der Dank gilt auch seinem Interesse an der Arbeit, seinen Einschätzungen und Denkanstössen.

Inhaltsverzeichnis

Aufgabenstellung

City Trip Planner: Kurztrip-Planer für Fussgänger

Semesterarbeit im Herbstsemester 2021/22, Bachelor-Studiengang Informatik

Hintergrund und Aufgabenstellung

Wegfindung ist ein gut erforschtes Gebiet in der Navigation und Verkehrsleitung von Autos. Es gibt diverse Dienste von namhaften Firmen, welche dieses Problem recht gut lösen. Auch für die Planung von Rundtouren - die zum Teil andere Algorithmen verlangen - gibt es bereits Lösungen, beispielsweise um individuelle Motorradtouren zu planen. Stadtrundtrips (kurz "City Trips") zu Points-of-Interest (kurz "POIs") erfreuen sich immer grösserer Beliebtheit und es gibt bislang kein passendes Produkt, das eine solche Planung als Webapplikation anbietet.

Das Ziel dieser Arbeit ist ein Minimal Viable Product für einen Rundtour-Planer zu entwickeln. Dieser Rundtour-Planer soll für Fussgänger:innen und Tourist:innen Touren zu kategorisierten POIs im städtisch-urbanen Raum erstellen und in einer responsiven Webapplikation zugänglich machen.

Aufgaben

- Requirements-Engineering in Zusammenarbeit mit Schweiz Tourismus
- Evaluation Open-Source Routing Engines
- Daten-Vorverarbeitung und Aufbereitung von OpenStreetMap-Daten
- Implementation eines MVP
- Evaluation der Ergebnisse

Lieferobjekte

1. Dokumentation, inkl. Text-Abstract, Management Summary und Anhänge (Inhalt, Literaturverzeichnis, etc.)
2. Lauffähige und webbasierte Applikation
3. Die vom Studiengang geforderten Lieferobjekte: Poster (nur digital), Broschüren-Abstract

Vorgaben – Technologien – Rahmenbedingungen

- Frontend: responsive Webapp mit HTML5, JavaScript/TypeScript in Angular
- Backend: Python
- Persistenz: Spatial SQL, PostgreSQL
- Daten: Open Data (OpenStreetMap)

Die Studierenden wählen nach Rücksprache ein Vorgehensmodell zur Softwareentwicklung.

Termine und Bewertung

Es gelten die üblichen Termine und Regelungen zum Ablauf und zur Bewertung der Arbeit (fünf Aspekte) des Bachelor-Studiengangs Informatik der OST. Zusätzlich wird Gewicht gelegt auf moderne Softwareentwicklung (Versionierung, Erweiterbarkeit, CI/CD) sowie auf ein lauffähiges Minimal Viable Product.

Beteiligte

Autoren: Jan Ruch und Lukas Leuenberger
Betreuer: Prof. Stefan Keller und Nicola Jordan, Institut für Software OST
Industriepartner: Markus Dittli, Schweiz Tourismus

Teil I: Technischer Bericht

1. Einführung

1.1. Problemstellung

Der Industriepartner für diese Studienarbeit ist Schweiz Tourismus. Schweiz Tourismus ist eine öffentlich-rechtliche Körperschaft des Bundes. Sie hat den Auftrag, die touristische Nachfrage für die Schweiz als Destination für Ferien, Reisen und Kongresse im In- und Ausland zu fördern. Die Kernaufgaben von Schweiz Tourismus lauten:

- Sorge über gesamtschweizerisches Tourismusmarketing
- Partner für gemeinsame Marktauftritte zu gewinnen und koordinieren
- Proaktive Beratung von Mitgliedern durch erlangtes Fachwissen aus Marktforschung und -beobachtungen

Schweiz Tourismus möchte in der Zukunft eine Möglichkeit zur Planung von individuellen Stadtrundgängen übers Internet anbieten. Dabei im Fokus steht der Langsamverkehr wie Fussgänger:innen und Kleinfahrzeuge.

Für die Ermittlung der Anforderungen und die Erstellung eines Prototyps wurde die Kooperation mit der Ostschweizer Fachhochschule eingegangen. Das Ziel dieser Kooperation ist es, die prinzipielle Durchführbarkeit des Vorhabens mittels eines Minimal Viable Products aufzuzeigen. Die vorliegende Arbeit setzt sich mit diesem Thema auseinander.

1.2. Produktvision

Die Applikation City Trip Planner soll ein intuitiv bedienbarer Routenplaner für Fussgänger:innen sein. Die Kernfunktion der Applikation ist es, Fussgänger:innen einen Rundtrip in einer Stadt, basierend auf eigenen Interessen, zu berechnen. Dazu kann eine Person sich anhand von vordefinierten Kriterien wie verfügbare Zeit, persönlichen Interessen und Standort einen Trip oder Rundtrip berechnen lassen.

Durch diese Funktion soll es beispielsweise Tourist:innen möglich sein, einen in der gewünschten Destination individuell auf sie abgestimmten Rundtrip zu erhalten. Eine Person kann aus einem Themenkatalog Kriterien zusammenstellen. Basierend auf diesen wird dann ein individualisierter Rundtrip erstellt. Die Kriterien für die Planung sind persönliche Interessen, verfügbare Zeit und die zurückzulegende Distanz.

Dieser Dienst soll den Tourist:innen ermöglichen ein auf sie abgestimmtes Erlebnis zu gestalten und bei der Erkundung der Feriendestination Schweiz helfen.

1.3. Ziele

- Eine Webapplikation für das Erstellen von Rundtrips in Schweizer Städten
- Individualisierbarkeit durch persönliche Interessen und unterschiedliche Destinationen
- Erhalt von Informationen durch Anzeige von Internetmedien an den POIs
- Intuitive Bedienung mit Integrationsmöglichkeit in bereits existierende Systeme

1.4. Rahmenbedingungen

- Das zu entwickelnde Produkt entspricht einem Minimal Viable Product, welches die konzeptionelle Umsetzbarkeit beweist.
- Der zu entwickelnde Dienst beschränkt sich auf die Schweiz.
- Die Datenbasis wird durch die OpenStreetMap Community zur Verfügung gestellt und nicht durch die Studierenden erfasst.
- Das Deployment der Applikation ist nicht Teil dieser Arbeit.
- Das Produkt wird nach gängigen Standards erstellt und erlaubt die Weiterentwicklung.

1.5. Vorgehen

1.5.1. Phase 1: Wissensbeschaffung

In einer ersten Phase ist die Wissensbeschaffung ein wichtiger Aspekt. Dabei sollen die Wissenslücken in folgenden Bereichen geschlossen werden:

- OpenStreetMap
  - Import von Daten mittels ETL-Tools in PostgreSQL Datenbank
  - Tabellenstruktur von OpenStreetMap-Daten
  - Spatial Queries basierend auf den OpenStreetMap-Daten
- Python
  - Erlernen der Sprache
  - Verwendung von FastAPI als Framework
  - Erstellen des Prototyps für die Referenzarchitektur

1.5.2. Phase 2: Anforderungsanalyse und Projektinitialisierung

In der zweiten Phase wird die Anforderungsanalyse erledigt und das Projektmanagement aufgesetzt. Dazu wird die Dokumentation in ihren groben Zügen und die entsprechenden Abschnitte für die Abgabe erstellt.

Die zu erstellenden Artefakte sind:

- Dokumentation
  - Automatisierung der Erstellung
  - Anforderungsanalyse inklusive Diagramme
  - Bestimmung des Scope für die Studienarbeit mit Schweiz Tourismus
- Projektmanagement
  - Meilensteinplanung des gesamten Projektes
  - Grobplanung einzelner Arbeitspakete für Meilensteine
  - Aufsetzen des Tools für Issue Tracking und Zeiterfassung

1.5.3. Phase 3: Implementation

In der dritten Phase werden die gestellten Anforderungen an das System implementiert. Dazu werden die Meilensteine und groben Arbeitspakete in konkrete Arbeitspakete heruntergebrochen und im Issue Tracking Tool zur weiteren Planung erfasst. Die Artefakte für diese Phase sind:

- Gesamtsystem
    - Source Code Versionierung sichergestellt
    - Continuous Integration und Testing sichergestellt
    - Anforderungen gemäss Spezifikation
- Dokumentation
    - Fortlaufende Dokumentation der erstellten Komponenten
        - Architektur und Domäne
        - Abläufe und Schnittstellen
    - Ausführungen und Erklärungen zum System erstellen

1.5.4. Phase 4: Abschluss und Dokumentation

In der letzten Phase kümmern wir uns um den Abschluss des Projektes und die Dokumentation. Das Ziel dieser Phase ist es, sämtliche Artefakte für die Abgabe der Studienarbeit erstellt und das weiterführende Vorgehen definiert zu haben.

- Funktions- und lauffähiges Minimal Viable Product
- Dokumentation der Studienarbeit
- Präsentation
- Demonstration und Produktabnahme

2. Stand der Technik

In diesem Kapitel wird der Stand der Technik hinsichtlich Problemen, Algorithmen und Lösungsansätzen untersucht. Er dient einer Orientierung und Einordnung der Studienarbeit. So soll bestimmt werden, ob es bereits ähnliche Umsetzungen oder Lösungsverfahren eines Produkts gibt. Die untersuchten Arbeiten werden bewertet und die Relevanz bestimmt.

2.1. Hamiltonkreisproblem

Das vorliegende Problem, einen Weg zu finden, auf dem jeder Point of Interest genau einmal vorkommt, nennt man in der theoretischen Informatik *Hamiltonkreisproblem*. Dieses Problem ist NP-vollständig. Das bedeutet, dass das Problem sich nichtdeterministisch in polynomieller Zeit lösen lässt. Dabei unterscheidet man noch zwischen gerichteten und ungerichteten Hamiltonkreisen. Eine Verallgemeinerung des Problems ist das Problem des Handlungsreisenden. Beim Problem des Handlungsreisenden geht es darum, den kürzesten Pfad in einem Graphen zu finden, welcher jeden Knoten eines Subset genau einmal besucht. Das Problem des Handlungsreisenden unterscheidet sich vom Hamiltonkreisproblem dadurch, dass nicht alle Knoten im Graph besucht werden müssen. Der resultierende Graph der Verbindungen aller Knoten des Subset muss dabei allerdings minimal sein. Die Anzahl an Knoten im Subset kann variieren. Ausserdem handelt es sich beim Hamiltonkreisproblem, wie der Name bereits beschreibt, um einen Kreis (zyklischer Graph) und nicht um eine Verbindung von Start- und Endpunkt (azyklischer Graph).

2.2. Algorithmen

Für die Lösungsfindung des Problems des Handlungsreisenden kommen häufig Heuristiken zum Einsatz. Dieser Lösungsansatz für das kombinatorische Optimierungsproblem wird beispielsweise auch für die künstliche Intelligenz beim Trainieren von neuronalen Netzwerken und in anderen Ingenieurswissenschaften angewendet. Die Anwendung von Heuristiken ist zwar schneller, aber nicht exakt. Für die Vorprozessierung ist oftmals grosser Rechenaufwand notwendig.

Bei der Verkehrsführung werden beispielsweise Contraction Hierarchies eingesetzt. Das Prozedere besteht aus zwei Phasen. In der ersten Phase werden Daten vorprozessiert, um sie in der zweiten Phase schneller Abfragen zu können. Der Name entstammt dem Umstand, dass es bei der Verkehrsführung verschiedene Hierarchiestufen gibt. So können Autobahnen den Überlandstrassen vorgezogen werden, um den schnellsten Weg zwischen zwei Punkten zu finden.

Ein weiterer Ansatz sind genetische Algorithmen. Sie orientieren sich an der natürlichen Evolution zur Lösung von Optimierungsproblemen. Sie werden mittels dynamischer Programmierung berechnet. Dabei werden die Zwischenresultate mithilfe von Kriterien geprüft. Wenn das Zwischenresultat besser - oder *optimaler* - ist, dann wird es übernommen. Ansonsten wird es verworfen und die Eingangsparameter modifiziert und erneut berechnet. [1]

Computing Optimal Road Trips Using Operations Research [2]

> In diesem Artikel beschreibt Nathan Brixius, wie er einen Pfad durch US-amerikanische State Capitols mithilfe eines TSP-Solvers und eines genetischen Algorithmus sucht. Das Problem des Handlungsreisenden tritt auch hier auf und Nathan Brixius erklärt seine Lösung dazu.

2.3. Bestehende Lösungsansätze

***POI Tour* [3]**

Die Arbeit beschäftigt sich mit dem Thema Routenplanung für Fussgänger:innen, diese scheint auf den ersten Blick sehr ähnlich zu sein, wie unsere Aufgabe. Sie beschäftigt sich ebenfalls damit, wichtige Punkte in eine Route zu integrieren. In der Arbeit werden OSRM und pgRouting miteinander verglichen.

***PlazaRoute* [4]**

Ehemalige Studienarbeit zum Thema Fussgänger:innen-Routing über offene Flächen, anstatt über offizielle Strassen und Wege. Das Routing basiert auf existierenden Algorithmen und OpenStreetMap.

***A Unified Pedestrian Routing Model Combining Multiple Graph-Based Navigation Methods* [5]**

Ein Fussgänger:innen Routing Modell, dass mehrere Graphen basierte Navigations-Modelle kombiniert. Das Modell behandelt die räumliche Wahrnehmung von Routing Modellen und wurde mit einer Fallstudie untersucht. In dieser Arbeit wurden auf Graphen basierende Routing Methoden kombiniert, jede Methode formalisiert einen Aspekt der räumlichen Wahrnehmung.

***Route Selection Algorithm for Blind Pedestrian* [6]**

Ein adaptierter Routing Algorithmus für blinde Fussgänger:innen, basierend auf dem Dijkstra Algorithmus. Eine Route soll nicht möglichst schnell sein, sondern den grössten Komfort bieten.

Route planning for blind pedestrians using OpenStreetMap [7]

Entwicklung eines Routing Algorithmus für blinde Fussgänger:innen. Dieser basiert auf OpenStreetMap und räumlichen Kriterien. Er dient dazu, den Fussgänger:innen mehr Selbstständigkeit, höhere Sicherheit und gesellschaftliche Integration zu bieten.

***Campus Guidance System for International Conferences Based on OpenStreetMap* [8]**

Behandelt das Erstellen von möglichst kurzen Routen für Fussgänger:innen, dabei werden OpenStreetMap-Daten genutzt, um möglichst kurze Routen über Campus-Gelände zu berechnen. Das System erstellt Pfade über Wege ausserhalb, sowie Korridore innerhalb von Gebäuden. Für zusätzliche visuelle Unterstützung werden Bilder mit geografischen Koordinaten (Geotagging) angeboten.

***A System for Generating Customized Pleasant Pedestrian Routes Based on OpenStreetMap Data* [9]**

Routing entlang von benutzerdefinierten Kriterien basierend auf OpenStreetMap-Daten. Die Arbeit beschreibt wie Faktoren aus OpenStreetMap-Daten extrahiert und in Routing-Funktionen integriert werden können.

***Trail Router* [10]**

Trail Router sucht nach Grünflächen, Wasser oder Wanderrouten in der Nähe des gewählten Startpunktes und erstellt Punkte, die der Routing Engine übergeben werden. Mit diesen Punkten wird eine möglichst schöne Route erstellt.

2.4. Kriterien

Der aktuelle Stand der Technik wird anhand von verschiedenen Kriterien bewertet:

- Welchen Nutzen haben die Vorarbeiten?
- Welche Nachteile oder Defizite haben die Vorarbeiten?
- Gibt es eine deckungsgleiche Vorarbeit zu unserem Auftrag?

2.5. Defizite

Die meisten Arbeiten bieten keine Möglichkeiten, um POIs in die Routen-Berechnung miteinzubeziehen. Dieses Feature ist der Kern unserer Aufgabe. Da dies nicht möglich ist, können auch keine zusätzlichen Informationen oder Bilder mit einem Point of Interest assoziiert werden.

Der Vorarbeit POI Tour fehlt ein Mobile-First Ansatz. Das Routing entlang von interessanten Punkten könnte verbessert werden, sodass Interessen unter Umständen nicht doppelt abgedeckt werden, oder die Routen entlang schönen Gegenden führen. [11]

Die Vorarbeit der Universität Heidelberg ist umfangreich, scheint aber im Bereich Routenplanung und nicht Rundtrip angesiedelt zu sein. Auch hier können nicht explizit Interessen definiert werden. [9]

Das Projekt Trail Router ist ein interessanter Ansatz, würde uns aber unter Umständen viele wichtige Punkte, in grosser Distanz zu schönen Orten, aus der erstellten Route entfernen. Deshalb verfolgen wir diesen Ansatz nicht weiter.

2.6. Schlussfolgerung

2.6.1. Algorithmus

Für unsere Arbeit ist die Exaktheit eines Resultats weniger wichtig als die Geschwindigkeit. Unsere Aufgabe verlangt, eine Webapplikation zur Verfügung zu stellen, in der immer wieder neue und unterschiedliche Rundtrips berechnet werden sollen. In diesem Umfeld erwarten Benutzer:innen kurze Wartezeiten, da sie ansonsten das Interesse an der Applikation verlieren.

> Aufgrund dieses Umstands verzichten wir auf die Suche eines optimalen Resultats. Wir wollen möglichst schnell ein Resultat den Benutzer:innen präsentieren. Ein optimales Resultat für jede Abfrage der Benutzer:innen zu finden, würde aufgrund der Komplexität zu viel Zeit in Anspruch nehmen. Wir befürchten, dass die Benutzer:innen bei zu langen Wartezeiten das Interesse an der Applikation verlieren. Wir nehmen deshalb in Kauf, dass die angezeigten Routen nicht optimal sind, solange sie schnell und innerhalb der von den Benutzer:innen eingegebenen Kriterien sind.

Die Herausforderung für diese Arbeit wird also sein, mithilfe einer geeigneten Routing Engine möglichst schnell zu einem passenden Resultat zu kommen und nicht die Berechnung von optimalen Rundtrips.

2.6.2. Bestehende Lösungsansätze

Aus den betrachteten Arbeiten heben sich zwei besonders ab. Die frühere Studienarbeit POI Tour, sowie A System for Generating Customized Pleasant Pedestrian Routes Based on OpenStreetMap Data der Universität Heidelberg. Wir werden uns diese beiden Artikel im Verlaufe der Arbeit genauer anschauen.

Die Studienarbeit POI Tour könnte uns Aufschluss darüber geben, wie man wichtige Punkte definieren und in eine Route einberechnen kann. Die Applikation basiert auch auf OpenStreetMap-Daten und benutzt eine Routing Engine. Aufgrund des Technologiefortschritts in der verstrichenen Zeit, wäre es interessant, die beiden Produkte miteinander hinsichtlich Funktionsumfang, Kompatibilität und Performanz zu vergleichen.

Das Dokument der Universität Heidelberg könnte uns für die Wahl der Kriterien, der Extraktion der Daten und die Integration in unsere Applikation nützlich sein. Insbesondere das Routing entlang von schönen Routen ist eine spannende Vorarbeit. Wir könnten diese Arbeit als Hilfestellung bei Problemen verwenden.

Die übrigen Arbeiten beziehen sich nur entfernt auf unsere Aufgabe. Wir werden diese deshalb nicht eingehender betrachten. Es scheint als wäre ein solches Routing Tool, wie wir es planen, zurzeit nicht auf dem Markt. Aus dieser Analysephase entnehmen wir, dass eine eigene Implementation der Projektidee zielführend ist.

3. Umsetzungskonzept

Die Arbeit ist in verschiedene Teilgebiete aufgeteilt. Um die gestellte Aufgabe zu erfüllen, müssen wir diese Teilgebiete analysieren und einen Plan erarbeiten, wie die unterschiedlichen Technologien zu einem System zusammengeschlossen werden können. Dieses Umsetzungskonzept legt die Grundlage für das Projektmanagement von City Trip Planner.

3.1. Wissensbeschaffung

Der erste und grundlegende Schritt dieser Arbeit wird sein, sich in Geoinformationssysteme mit PostgreSQL und den dazugehörigen Tools einzuarbeiten. Für eine umfassende Analyse der Anforderungen bedarf es der Kenntnis der Funktionsweise von Geoinformationssystemen. Wir haben deshalb diesen Teil auch in unserem Projektmanagement als Meilenstein aufgenommen.

3.2. Anforderungsanalyse und Systemarchitektur

Bevor ein System gebaut wird, müssen zuerst die Anforderungen gestellt werden. Nach den Erkenntnissen aus dem ersten Teil erfassen wir einen Anforderungskatalog. Darin sollen Ideen und Vorstellungen für mögliche Features festgehalten werden, damit wir sie zusammen mit dem Industriepartner Schweiz Tourismus besprechen sowie priorisieren können. Das Resultat der Anforderungsanalyse wurde im Kapitel Anforderungen festgehalten.

Die Anforderungen an das System werden die Systemarchitektur stark prägen. In dieser Phase gilt es, die Anforderungen der Aufgabenstellung und des Industriepartners in eine geeignete Architektur einzuarbeiten und einen passenden Entwurf zu erstellen.

3.3. Evaluation von Routing Engines

Nachdem feststeht, was in der vorliegenden Arbeit erwartet wird, muss eine passende Routing Engine gefunden werden. Dazu werden wir bekannte Open-Source Routing Engines analysieren und anhand von diversen Kriterien einstufen. Diejenige Engine, welche sich als geeignetste herausstellt, werden wird für das Minimal Viable Product an das System anschliessen.

Für die Evaluation wird ein Kriterienkatalog zusammengestellt. Die Kriterien werden aus der Aufgabenstellung und der Anforderungsanalyse erstellt. Die Routing Engine soll als Komponente im Gesamtsystem integriert werden. Das bedeutet, dass wir keine Erweiterung einer Routing Engine implementieren, sondern mithilfe einer Routing Engine unser Vorhaben umsetzen wollen. Die evaluierten Routing Engines und deren Resultate sind im Kapitel Evaluation Routing Engine niedergeschrieben.

3.4. Implementierung

Gleichzeitig wie die Evaluation der Routing Engine, können wir anhand eines Prototyps das Grundgerüst für die Entwicklung legen. Hierzu werden die Erkenntnisse aus der Wissensbeschaffung und Anforderungsanalyse zu Hilfe genommen, um ein erweiterbares, performantes und robustes System zu bauen. In den Kapiteln Architektur und Implementation sind die Artefakte dieser Phase dokumentiert.

Nach dem Erstellen des Prototyps müssen sämtliche Risiken, welche ein Scheitern der Studienarbeit zur Folge hätten, ausgeschlossen werden können. Wenn das sichergestellt ist, werden wir mit der Umsetzung der priorisierten Anforderungen beginnen.

3.5. Abschluss

In der letzten Phase dieser Arbeit widmen wir uns dem Abschluss. In diesem sollen die Erfahrungen und das Ergebnis reflektiert und festgehalten werden, um daraus weitere Massnahmen für das weitere Vorgehen abzuleiten.

4. Resultate

In diesem Kapitel wird die Zielerreichung anhand der Anforderungen beurteilt und die zusätzlichen, nicht spezifizierten Features aufgelistet. Die Beurteilung ergibt Aufschluss darüber, welche Weiterentwicklungsmöglichkeiten existieren.

4.1. Zielerreichung

Die Zielerreichung legen wir anhand der Anforderungsanalyse fest. Die Rückmeldungen von Markus Dittli und Prof. Stefan Keller fliessen in die Bewertung ein. Anforderungen, welche mit *Will Not Have* gekennzeichnet sind, werden hier nicht erwähnt.

4.1.1. REQ1 - Stadtrundgang

Anforderung	Der geplante Ausflug endet am gleichen Ort wie er begonnen hat. (Kritischer Faktor Distanz)
Kommentar	Die Distanz kann durch den Benutzer eingeschränkt werden und wird nie überschritten. Je nach Eingabe dauert die Suche länger. Um die Geschwindigkeit in diesen Bereichen weiter zu optimieren, brauchen wir weitere Informationen zum Nutzungsverhalten der Anwender.
Beurteilung	Dieses Ziel ist erfüllt.

4.1.2. REQ3 - Auswahl einer Route

Anforderung	Ein Anwender erhält verschiedene Vorschläge von Routen für seinen gewünschten Trip.
Kommentar	Es werden nicht mehrere Vorschläge gleichzeitig berechnet und angezeigt. Allerdings können die Anwender den Startpunkt verschieben und der Rundtrip wird neu berechnet. Diese Neuberechnungen werden als verschiedene Routen gewertet. Diese Lösung wurde so Markus Dittli demonstriert und besprochen. Die Anforderung gilt als erfüllt.
Beurteilung	Dieses Ziel ist erfüllt.

4.1.3. REQ3.3 - Anzahl an zu besuchenden POI - Kategorien

Anforderung	Für die folgenden Kategorien kann die Anzahl an POIs angegeben werden: • Kultur • Geschichte • Architektur (Die Aufzählung ist nicht abschliessend und kann in kommenden Iterationen angepasst werden.)
Kommentar	Die Anzahl an POIs für Kategorien kann nicht angegeben werden. Die Anforderung ist als *Could Have* eingestuft und konnte im Rahmen der Studienarbeit nicht umgesetzt werden.
Beurteilung	Diese Anforderung wurde aufgrund der niedrigen Priorität nicht umgesetzt und gilt als nicht erfüllt.

4.1.4. REQ3.4 - Anzahl an zu besuchenden POI - Priorisierung

Anforderung	Für die Priorisierung der Top POIs werden folgende Kriterien berücksichtigt: • Besitzt einen Wikipedia-Artikel • Anzahl Aufrufe des Wikipedia-Artikels
Kommentar	Eine Priorisierung anhand der Daten von OpenStreetMap ist nicht möglich. Es braucht dafür zusätzliche Informationen, welche aktuell nicht zur Verfügung stehen. Es gibt Projekte wie Wikidata QRank [12], welche sich mit dieser Thematik auseinandersetzen. Diese Anforderung ist als *Must Have* eingestuft und konnte im Rahmen der Studienarbeit nicht umgesetzt werden.
Beurteilung	Diese Anforderung ist aufgrund fehlender Daten nicht realisierbar. Das Ziel ist nicht erfüllt.

4.1.5. REQ4 - Kategorisierung von Interessen

Anforderung	Ein Benutzer kann seine Interessen in eigene Kategorien unterteilen.
Kommentar	Es können neue Kategorien über die REST-Schnittstelle erstellt werden. Der initiale Daten-Import verwendet dieselbe Schnittstelle zur Erstellung der Standard-Kategorien. Mittels Konfiguration kann der Betreiber der Software seine Kategorien selber definieren. Der Betreiber wird auch als Benutzer eingestuft. Eine Umsetzung im Frontend fand noch nicht statt. Zuerst braucht es ein Security-Konzept für die Benutzer und eigene Benutzer-Accounts.
Beurteilung	Dieses Ziel ist erfüllt.

4.1.6. REQ4.1 - Vordefinierte Kategorien von Interessen

Anforderung	Es gibt folgende vordefinierte Kategorien: • Kultur • Geschichte • Architektur (Die Aufzählung ist nicht abschliessend und kann in kommenden Iterationen angepasst werden.)
Kommentar	Wie in REQ4 - Kategorisierung von Interessen bereits erklärt, ist diese Funktion bereits umgesetzt. Aktuell sind folgende Kategorien erfasst: • Familie • Sport • Kultur • Erlebnis • Natur • Gebäude • Religion • Geschichte • Tourismus
Beurteilung	Dieses Ziel ist erfüllt.

4.1.7. REQ6 - Sharing Möglichkeit

Anforderung	Der Anwender kann eine erstellte Route teilen.
Kommentar	Die URL beinhaltet ein UUID für jede erstellte Route, sie kann einfach versendet werden. Zusätzlich gibt es einen QR-Code, um die Route zu teilen.
Beurteilung	Dieses Ziel ist erfüllt.

4.1.8. REQ8 - Benutzerprofil

Anforderung	Ein Anwender kann ein Profil anlegen, um dort verschiedene Rundgänge und Trips zu speichern, um sie später wiederzufinden.
Kommentar	Die Registrierung und die Benutzerprofile wurden aufgrund der Einstufung von *Could Have* und der fehlenden Zeit nicht in dieser Studienarbeit umgesetzt. Allerdings kann sich ein Benutzer verschiedene URLs als Favoriten in seinem Browser speichern, somit könnte die Funktionalität dem Benutzer überlassen werden.
Beurteilung	Dieses Ziel ist teilweise erfüllt.

4.1.9. REQ9 - Anforderungen an Rundgang/Trip

Anforderung	Ein Anwender hat die Möglichkeit rollstuhlgängige Pfade berechnen zu lassen.
Kommentar	Ein Anwender kann zurzeit keine rollstuhlgängigen Routen berechnen lassen. Die Komplexität für dieses Problem ist für den Umfang der Studienarbeit zu gross. Auf den einzelnen POIs haben wir die Information zur Rollstuhlgängigkeit in den Details dargestellt, wo vorhanden. Diese Anforderung war als *Should Have* markiert. Die Anforderung kann mittels Anbindung einer Routing Engine, welche rollstuhlgängige Pfade berechnen kann, umgesetzt werden.
Beurteilung	Dieses Ziel ist nicht erfüllt.

4.2. Zusätzliche Features

Zusätzlich zu den geforderten Funktionalitäten werden bereits weitere Features angeboten.

- Die Software funktioniert für die gesamte Schweiz und nicht nur für auserwählte Städte.
- Landes-, Kantons- und Ortschaftsgrenzen können per Konfiguration hervorgehoben werden.
- Zusätzliche Informationen zu den POIs werden dargestellt. Bilder sind ersichtlich und weiterführende Links werden angezeigt.
- Die Applikation kann in drei Sprachen (Deutsch, Französisch, Englisch) bedient werden.
  Wikipedia-Artikel werden ebenfalls in der gewählten Sprache verlinkt.
  Zusätzliche Sprachen können beliebig ergänzt werden.

4.3. Vergleich mit POI Tour

In diesem Abschnitt wird die frühere Arbeit POI Tour mit dem Resultat der jetzigen Arbeit City Trip Planner verglichen.

Arbeit	**POI Tour**	**City Trip Planner**
Vision	Die Vision von POI Tour basiert darauf, dass man spontan einen Rundtrip starten kann, um wichtige Daten in OSM zu verbessern.	Die Kernfunktion der Applikation ist es, Fussgänger:innen einen Rundtrip in einer Stadt, basierend auf eigenen Interessen, zu berechnen. Dazu kann eine Person sich anhand von vordefinierten Kriterien wie verfügbare Zeit, persönlichen Interessen und Standort einen Trip oder Rundtrip berechnen lassen.
Technologien	Python, Flask, AngularJS, PostgreSQL, OpenStreetMap und OSRM	Python, FastAPI, Angular, PostgreSQL, OpenStreetMap und OSRM
Mobile-First	POI Tour basiert auf AngularJS und ist zurzeit nicht Mobile-fähig.	City Trip Planner basiert auf Angular 13, und unterstützt Mobile-First.
Architektur	POI Tour verwendet eine Client-Server-Architektur ohne weiteren Detaillierungsgrad der einzelnen Komponenten.	City Trip Planner entwirft die kritischen Komponenten in der Gesamtarchitektur und besitzt für einzelne Komponenten eigene Referenzarchitekturen.

Arbeit	POI Tour	City Trip Planner
Erweiterbarkeit	Die Kategorien sind erweiterbar.	Es ist explizit vorgesehen, dass neue Kategorien definiert und unterschiedliche Routing Engines angeschlossen werden können. Je nach Anwendungsfall wird die passende Engine gewählt.
Kategorien von Interessen	Die Kategorien entspringen den Tags der OpenStreetMap -Tabellen.	Unsere Kategorien stellen ein Mapping zwischen unserer Domäne und den gekennzeichneten Interessen der OpenStreetMap-Tabellen dar. Dadurch entsteht die Entkopplung zu den OpenStreetMap-Daten, die auch nach einem Neuimport der Daten funktioniert.
Algorithmen	POIs ausserhalb des Laufradius werden ausgeschlossen. [11]	Zu weit entfernte POIs werden auf ähnliche Art ausgeschlossen. Genügend nahe POIs werden zusätzlich über einen von drei verschiedenen Algorithmen in den Rundtrip einberechnet.
Darstellen von Informationen	Zu POIs werden Fotos von Flickr dargestellt. Die Fotos werden aufgrund des Standortes gesucht. Es kann vorkommen, dass auch nicht POI-relevante Fotos angezeigt werden.	Es werden Informationen, Icons, Bilder und weiterführende Links basierend auf Wikidata-Einträgen zu den jeweiligen POIs dargestellt. Es ist nicht möglich, dass nicht POI -relevante Daten angezeigt werden.
Containerisierung	Die Applikation wurde auf Host-Maschinen installiert.	Die verschiedenen Komponenten können komplett in Containern betrieben werden. Eine Möglichkeit wird im Anhang gezeigt.

Basierend auf den ursprünglich unterschiedlichen Visionen und der unterschiedlichen Implementation war es aus unserer Sicht sinnvoll, diese Arbeit neu zu implementieren. Insbesondere die verwendete Referenzarchitektur eröffnet weiteres Verbesserungspotential mit minimalen Veränderungen an der existierenden Codebasis. Hätte der City Trip Planner auf POI Tour aufgebaut, wäre unserer Ansicht nach noch die Einarbeitungszeit in die existierende Software hinzugekommen. Die korrekte Umsetzung und die Erweiterbarkeit wären dabei nicht garantiert gewesen. Zusätzlich besteht eine grosse technische Schuld aufgrund der alten Python- und Angular-Versionen.

4.4. Weiterentwicklungen

4.4.1. Offene Anforderungen

Diese offenen Anforderungen waren nicht im aktuellen Auslieferungsfenster oder wurden nur teilweise erfüllt.

- [REQ2]
- [REQ3.1]
- [REQ3.2]
- [REQ3.3]
- [REQ3.4]
- [REQ5]
- [REQ5.1]
- [REQ7]
- [REQ7.1]
- [REQ8]
- [REQ9]

4.4.2. Sonstige Weiterentwicklungen

- Herunterladen von Trips für andere Zwecke (GPS-Uhren o.ä.)
- Nutzer können den vorgeschlagenen Trip individuell anpassen
- Routing über offene Flächen mittels *PlazaRoute*
- Zusätzliches Routing Profil als Lua-Script
- Weitere Routing Engines anbinden und evaluieren
- Fehler im PrimeNG Accordion-Element beheben
- Verweildauer an POIs miteinberechnen

Auf die Konzepte der Weiterentwicklungen sind wir in unserer Projektdokumentation genauer eingegangen.

Teil II: Projektdokumentation

5. Anforderungsspezifikation

In diesem Kapitel wird das Lastenheft des Gesamtsystems von City Trip Planner definiert. Dafür werden alle möglichen Anforderungen in einem Katalog festgehalten. Dieser Katalog dient als Backlog für das Projekt. Für die Planung des Minimal Viable Product wird durch Markus Dittli eine Priorisierung der Tasks vorgenommen. Diese Tasks werden dann innerhalb dieser Studienarbeit umgesetzt.

Dieses Kapitel legt die Basis für die Software Architektur. Dazu werden sämtliche Anforderungen ans System, auch diejenigen ausserhalb des Auslieferungsfensters, berücksichtigt. Obschon einzelne Anforderungen nicht konkretisiert sind, sollten sie trotzdem mit der gewählten Gesamtarchitektur umsetzbar sein. Dieser Umstand muss während dem Requirement Engineering im Hinterkopf behalten und beim Erstellen der Architektur miteinbezogen werden.

5.1. Anforderungen

5.1.1. Priorisierung

Die untenstehenden Anforderungen wurden mittels MoSCoW-Modell priorisiert.

Abkürzung	Name	Bedeutung
M	Must Have	Kritisch für das Auslieferungsfenster
S	Should Have	Wichtig, aber nicht notwendig
C	Could Have	Wünschenswert, aber nicht wichtig
W	Will Not Have	Nicht geplant im Auslieferungsfenster

5.1.2. Funktionalität

Während der Anforderungsanalyse haben sich folgende Anforderungen ergeben.

#	Titel	Beschreibung	Priorität
REQ1	Stadtrundgang	Der geplante Ausflug endet am gleichen Ort wie er begonnen hat. (Kritischer Faktor Distanz)	M
REQ2	Besichtigungstrip	Der geplante Ausflug hat einen anwenderdefinierten Start- und Endpunkt, welche sich unterscheiden. (Kritischer Faktor Zeit)	W
REQ3	Auswahl einer Route	Ein Anwender erhält verschiedene Vorschläge von Routen für seinen gewünschten Trip.	S

#	Titel	Beschreibung	Priorität
REQ3.1	Auswahlkriterien	Ein Anwender kann aus folgenden Kriterien für die Auswahl einer Route entscheiden: • Natur (Parks, Gewässer) • Verkehrsarm (Quartierstrassen) • Einkaufsmeile (Die Aufzählung ist nicht abschliessend und kann in kommenden Iterationen angepasst werden.)	W
REQ3.2	Anzahl an zu besuchenden Points of Interest	Ein Anwender kann eine Mindestanzahl an POIs einer Kategorie angeben.	W
REQ3.3	Anzahl an zu besuchenden Points of Interest - Kategorien	Für die folgenden Kategorien kann die Anzahl an POIs angegeben werden: • Kultur • Geschichte • Architektur (Die Aufzählung ist nicht abschliessend und kann in kommenden Iterationen angepasst werden.)	C
REQ3.4	Anzahl an zu besuchenden Points of Interest - Priorisierung	Für die Priorisierung der Top POIs werden folgende Kriterien berücksichtigt: • Besitzt einen Wikipedia-Artikel • Anzahl Aufrufe des Wikipedia-Artikels	M
REQ4	Kategorisierung von Interessen	Ein Benutzer kann seine Interessen in eigene Kategorien unterteilen.	C
REQ4.1	Vordefinierte Kategorien von Interessen	Es gibt folgende vordefinierte Kategorien: • Kultur • Geschichte • Architektur (Die Aufzählung ist nicht abschliessend und kann in kommenden Iterationen angepasst werden.)	S
REQ5	VIPOI - Very Important Point of Interest	Ein Anwender hat die Möglichkeit VIPOI zu spezifizieren, welche zwingend auf der Route vorkommen müssen.	W

#	Titel	Beschreibung	Priorität
REQ5.1	VIPOI - Auswahl	Ein Anwender kann aus folgenden VIPOI entscheiden: • Geldautomaten • Öffentliche Toiletten • Anwenderdefinierter POI (Die Aufzählung ist nicht abschliessend und kann in kommenden Iterationen angepasst werden.)	W
REQ6	Sharing Möglichkeit	Der Anwender kann eine erstellte Route teilen.	M
REQ7	Öffentliche Rundgänge/Trips	Der Anwender kann aus vorgefertigten Routen aussuchen. (Die Routen wurden gemeinschaftsorientiert erstellt)	W
REQ7.1	Rating für öffentliche Rundgänge/Trips	Die öffentlichen Routen können bewertet und kommentiert werden.	W
REQ8	Benutzerprofil	Ein Anwender kann ein Profil anlegen, um dort verschiedene Rundgänge und Trips zu speichern und sie später wiederzufinden.	C
REQ9	Anforderungen an Rundgang/Trip	Ein Anwender hat die Möglichkeit rollstuhlgängige Pfade berechnen zu lassen.	S

Die Priorisierung wurde am 13.10.2021 zusammen mit Markus Dittli von Schweiz Tourismus gemacht. Das zu Beginn erwähnte Auslieferungsfenster bezieht sich auf die Dauer der Studienarbeit in der Form des Minimal Viable Product.

5.2. Anwendungsfalldiagramm

5.2.1. Gesamtüberblick

Die definierten Anwendungsfälle decken die gesamten Anforderungen ab. Für das, während der Studienarbeit zu erarbeitende, Minimal Viable Product werden nicht sämtliche Anwendungsfälle umgesetzt. Der Fokus der Studienarbeit liegt auf den weiter unten aufgeführten Anwendungsfällen.

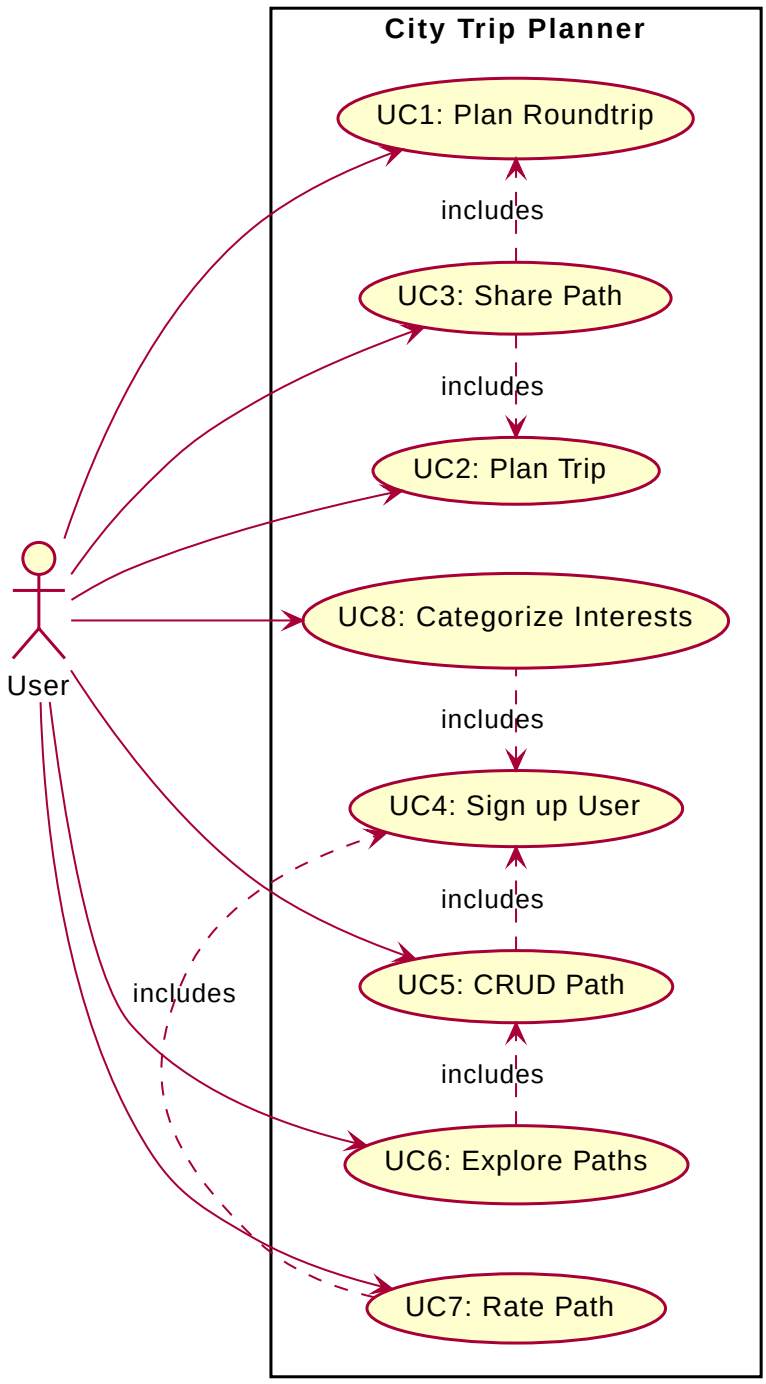


Abbildung 4. Anwendungsfälle Gesamtübersicht

5.2.2. Fokus Studienarbeit

Aus der Gesamtübersicht der Anwendungsfälle entsteht durch die Priorisierung mit Markus Dittli folgendes Anwendungsfalldiagramm.

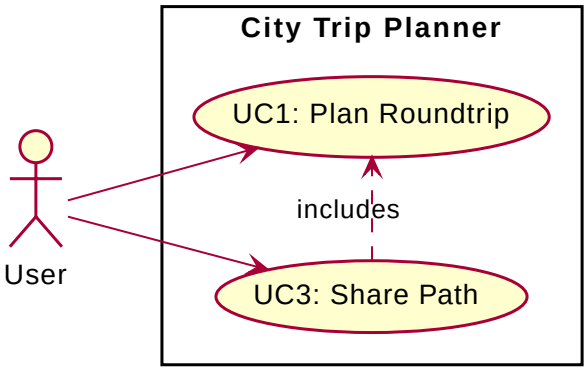


Abbildung 5. Anwendungsfälle Fokus Studienarbeit

UC3: Share Path hat eine Vorbedingung zu UC6: Explore Paths. Da aber UC6 nicht im Auslieferungsfenster der vorliegenden Studienarbeit liegt, wurde er im Diagramm nicht berücksichtigt.

5.3. Anwendungsfälle

5.3.1. UC1: Plan Roundtrip

Beschreibung	Der Anwender plant eine Stadtreise in der Schweiz. Vor seinem Aufenthalt plant er die Besichtigung von POIs basierend auf seinen persönlichen Interessen. Er gibt dafür die Kriterien für seinen individuellen Rundtrip ein. Basierend auf den eingegebenen Kriterien werden Points of Interest herausgesucht, welche zu seinem geplanten Vorhaben passen. Daraufhin werden passende Rundtrips zusammengestellt und dem Anwender angezeigt. Er kann nun die für ihn passendste Variante auswählen. Der Anwender landet am Schluss seines Rundgangs wieder am Ausgangspunkt. Dieser Anwendungsfall entspricht dem Hauptanwendungsfall.
Abdeckung von Anforderungen	Folgende Anforderungen wurden für diesen Anwendungsfall spezifiziert und liegen im Auslieferungsfenster: • [REQ1] • [REQ3] • [REQ3.3] • [REQ3.4] • [REQ4.1] • [REQ9] Nachstehende Anforderungen wurden spezifiziert, liegen aber nicht im Auslieferungsfenster der vorliegenden Studienarbeit. • [REQ3.1] • [REQ3.2] • [REQ5] • [REQ5.1]

5.3.2. UC2: Plan Trip

Beschreibung	Der Anwender plant eine Stadtreise in der Schweiz. Vor seinem Aufenthalt plant er die Besichtigung von POIs basierend auf seinen persönlichen Interessen. Er gibt dafür die Kriterien für seinen individuellen Trip ein. Basierend auf den eingegebenen Kriterien werden Points of Interest herausgesucht, welche zu seinem geplanten Vorhaben passen. Daraufhin werden passende Trips zusammengestellt und dem Anwender angezeigt. Er kann nun die für ihn passendste Variante auswählen. Der Anwender landet zum Schluss an seinem festgelegten Ziel.
Abdeckung von Anforderungen	Nachstehende Anforderungen wurden spezifiziert. Dieser UC2: Plan Trip liegt aber nicht im Auslieferungsfenster der vorliegenden Studienarbeit. • [REQ2] • [REQ3] • [REQ3.1] • [REQ3.2] • [REQ3.3] • [REQ3.4] • [REQ4.1] • [REQ5] • [REQ5.1] • [REQ9]

5.3.3. UC3: Share Path

Vorbedingungen	Diesem Anwendungsfall liegt UC1: Plan Roundtrip, UC2: Plan Trip und UC6: Explore Paths zugrunde. Das bedeutet, dass er nur angewendet werden kann, falls einer der gelisteten Anwendungsfälle bereits angewendet wurde.
Beschreibung	Der Anwender kann eine Route teilen. Der Anwender kann einen Rundtrip oder Trip mit einer anderen Person teilen. Er erhält dafür einen Link mit einer eindeutigen Identifizierung eines Rundtrips oder Trips. Personen, welche diese Identifizierung kennen, können direkt auf den Rundtrip oder Trip zugreifen und ihn auf dem eigenen Gerät anzeigen lassen.
Abdeckung von Anforderungen	Folgende Requirements werden mit diesem Use Case abgedeckt: • [REQ6]

5.3.4. UC4: Sign up User

Vorbedingungen	Dieser Anwendungsfall stellt keinen elementaren Anwendungsfall dar. Er stellt aber die Grundlage für erweiterte Anwendungsfälle wie UC5: CRUD Path, UC7: Rate Path und UC8: Categorize Interests dar.
Beschreibung	Ein Anwender, welcher erweiterte Funktionen brauchen will, kann sich registrieren und anmelden.
Abdeckung von Anforderungen	Dieser Use Case erfüllt keine konkreten Anforderungen, ist aber die Grundlage für erweiterte Funktionen.

5.3.5. UC5: CRUD Path

Vorbedingungen	Diesem Anwendungsfall liegt UC3: Share Path zugrunde. Das bedeutet, dass er nur angewendet werden kann, falls der aufgeführte Anwendungsfall bereits angewendet wurde.
Beschreibung	Ein Anwender kann eigene Routen speichern, anzeigen, aktualisieren und löschen.
Abdeckung von Anforderungen	Nachstehende Anforderungen wurden spezifiziert. Dieser UC5: CRUD Path liegt aber nicht im Auslieferungsfenster der vorliegenden Studienarbeit. • [REQ8]

5.3.6. UC6: Explore Paths

Vorbedingungen	Diesem Anwendungsfall liegt UC5: CRUD Path zugrunde. Das bedeutet, dass er nur angewendet werden kann, falls der aufgeführte Anwendungsfall bereits angewendet wurde.
Beschreibung	Ein Anwender kann aus einer Liste von vorgefertigten und veröffentlichten Routen eine Route aussuchen, wenn er keine eigene erstellen möchte.
Abdeckung von Anforderungen	Nachstehende Anforderungen wurden spezifiziert. Dieser UC6: Explore Paths liegt aber nicht im Auslieferungsfenster der vorliegenden Studienarbeit. • [REQ7]

5.3.7. UC7: Rate Path

Vorbedingungen	Diesem Anwendungsfall liegt UC3: Share Path zugrunde. Das bedeutet, dass er nur angewendet werden kann, falls der aufgeführte Anwendungsfall bereits angewendet wurde.
Beschreibung	Ein eingeloggter User kann veröffentlichte Routen bewerten und kommentieren.
Abdeckung von Anforderungen	Nachstehende Anforderungen wurden spezifiziert. Dieser UC6: Explore Paths liegt aber nicht im Auslieferungsfenster der vorliegenden Studienarbeit. • [REQ7.1]

5.3.8. UC8: Categorize Interests

Beschreibung	Der Anwender kann seine Interessen kategorisieren und für die Suche nach Points of Interest anwenden.
Abdeckung von Anforderungen	Nachstehende Anforderungen wurden spezifiziert. Dieser UC8: Categorize Interests liegt aber nicht im Auslieferungsfenster der vorliegenden Studienarbeit. • [REQ4]

5.4. Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen, kurz NFAs, richten sich nach ISO/IEC 25010 [13].

5.4.1. NFR1 - Functional Suitability

NFR1.1 - Functional Completeness	Die funktionale Vollständigkeit ist erfolgreich erfüllt, wenn alle funktionalen Anforderungen von Schweiz Tourismus erfolgreich umgesetzt wurden. Weiterentwicklungen sind nicht Teil dieses Kriteriums. Nicht erfüllte Anforderungen müssen konzeptionell festgehalten werden. Die Abnahme durch Schweiz Tourismus bestätigt diese Anforderung.
NFR1.2 - Functional Correctness	Die funktionale Korrektheit ist erfolgreich erfüllt, wenn eine berechnete Route die selektierte Laufdistanz nicht überschreitet und maximal um 20% unterschreitet. Die Abnahme durch Schweiz Tourismus bestätigt diese Anforderung.
NFR1.3 - Functional Appropriateness	Die funktionale Verhältnismässigkeit korreliert mit der Anforderung *Time Behaviour*. Angewendete Algorithmen müssen die Berechnung des Rundtrips unterstützen. Die funktionale Verhältnismässigkeit ist erfolgreich erfüllt, wenn Algorithmen angemessen eingesetzt werden und die Laufzeit des Gesamtsystems verbessern. Die Abnahme durch Schweiz Tourismus bestätigt diese Anforderung.

5.4.2. NFR2 - Performance

NFR2.1 - Time Behaviour	Diese Anforderung ist erfolgreich erfüllt, wenn die Berechnungsdauer einer Route maximal 5 Sekunden beträgt. Testing oder Log-Monitoring bestätigen diese Anforderung.

5.4.3. NFR3 - Compatibility

NFR3.1 - Interoperability	Interoperabilität ist erfolgreich erfüllt, wenn die verschiedenen Komponenten und Umsysteme der Applikation miteinander kommunizieren können und korrekt funktionieren. Ausfälle von Umsystemen sind nicht Teil dieser Anforderung. Testing bestätigt diese Anforderung. **Komponenten:** • Datenbank • Backend • Routing Engine • Frontend **Umsysteme:** • Geofabrik • Wikidata API

5.4.4. NFR4 - Usability

NFR4.1 - Operability	Die Bedienbarkeit ist erfolgreich erfüllt, wenn die gesamte Applikation auf *Docker* basiert und mit maximal 2 Befehlen ausgeführt werden kann. Entsprechendes *Docker*-Wissen ist vorausgesetzt. Eine fehlerhafte Applikation ist nicht Teil dieser Anforderung. Testing bestätigt diese Anforderung.
NFR4.2 - User Error Protection	Der Fehlerschutz ist erfolgreich erfüllt, wenn alle Eingaben des Anwenders validiert werden und bei Fehlern eine entsprechende Fehlermeldung ausgegeben wird. Testing bestätigt diese Anforderung.

5.4.5. NFR5 - Maintainability

NFR5.1 - Modularity	Die Modularität ist erfolgreich erfüllt, wenn alle Komponenten der Applikation in eigenen *Docker-Containern* laufen. Die einzige Verbindung zwischen Backend und Frontend ist eine *REST-API*. Code Reviews bestätigen diese Anforderung.
NFR5.2 - Modifiability	Die Veränderbarkeit ist erfolgreich erfüllt, wenn weitere Routing Engines ohne Änderung der bestehenden Architektur angebunden werden können. Die Schnittstelle zu den Routing Engines wird vereinheitlicht und aus interner Sicht als Interface angeboten. Code Reviews und Testing bestätigen diese Anforderung.

5.4.6. NFR6 - Portability

NFR6.1 Adaptability	Die Anpassungsfähigkeit ist erfolgreich erfüllt, wenn die Applikation in verschiedenen Umgebungen installiert und ausgeführt werden kann. Testing bestätigt diese Anforderung. **Umgebungen:** • Localhost • GitLab Pipeline • Server
NFR6.2 - Installability	Die Installierbarkeit ist erfolgreich erfüllt, wenn die Applikation auf den 3 gängigsten Betriebssystemen installiert und ausgeführt werden kann. Testing bestätigt diese Anforderung. **Betriebssysteme** • Linux • Windows • macOS

6. Analyse

In dieser Analyse werden die verschiedenen Lösungsstrategien für die Findung eines passenden Resultats eines Pfads und das Domänenmodell genauer untersucht. Diese Untersuchungen sollen unter Berücksichtigung der Anforderungen konzeptionellen Aufschluss über die Implementierungsmodelle geben.

6.1. Algorithmen

In diesem Abschnitt werden verschiedene Algorithmen betrachtet, welche für unsere Arbeit von Bedeutung sind. Sie dienen einem besseren Verständnis und helfen bei der Einschätzung der Komplexität.

6.1.1. Wegfindung

Zum Thema Wegfindung existieren bereits einige erprobte Algorithmen. Die wohl bekanntesten unter ihnen sind:

- Dijkstra-Algorithmus von Edsger W. Dijkstra
- A*-Algorithmus von Peter Hart, Nils J. Nilsson und Bertram Raphael
- Bellman-Ford-Algorithmus von Richard Bellman, Lester Ford und Edward F. Moore

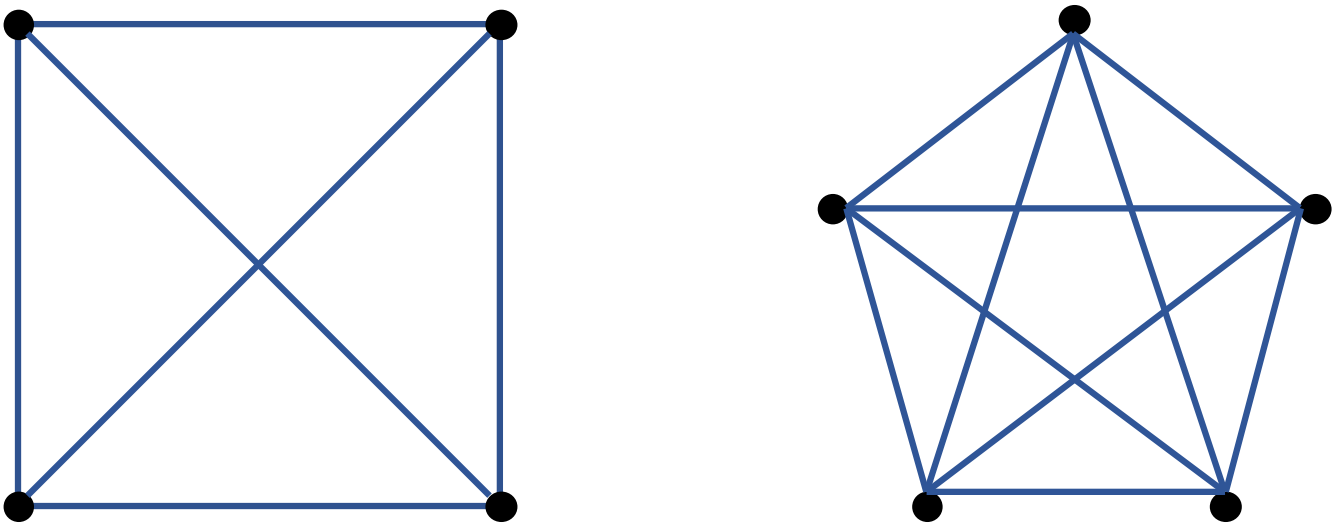


Abbildung 6. Graph mit 4 bzw. 5 Punkten und allen Verbindungen

Sie alle basieren auf kantengewichteten Graphen und suchen sich den kürzesten Weg durch ihn. Dabei kann der Dijkstra-Algorithmus nur mit positiven Kantengewichten rechnen, der Bellman-Ford-Algorithmus jedoch auch mit negativen. Hierbei ist aber zu beachten, dass es keine Zyklen mit negativen Gewichten geben darf. Der Algorithmus kann unter diesem Umstand nicht terminieren. Der A*-Algorithmus verwendet eine heuristische Schätzfunktion, kann aber auf den Dijkstra-Algorithmus reduziert werden.

6.1.2. Suche nach Route mit maximaler Distanz

Wir wissen, dass wir aufgrund des NP-schweren Problems des Handlungsreisenden nie ein optimales Ergebnis erhalten, ohne sämtliche Kombinationen durchgerechnet zu haben. Diese Kombinationen nehmen exponentiell mit der Anzahl Knoten zu. Für die Applikation City Trip Planner ist die Dauer, bis ein

passendes Resultat innerhalb der vorgegebenen Distanz gefunden wird, relevant. Dabei spielt die Optimalität eine zweitrangige Rolle.

Wir gehen von der natürlichen Intuition aus, dass man weiter gehen muss, wenn mehr Punkte auf dem Weg liegen sollen. gehen. Unser Ziel ist es, möglichst schnell einen passenden Weg zu finden, der möglichst nahe an die vorgegebene Distanz kommt. Dafür werden alle infrage kommenden Punkte nach aufsteigendem Abstand zum Startpunkt in einer Liste gespeichert. Aus dieser Liste werden jeweils unterschiedlich lange Sublisten an die Routing Engine gesendet. Dabei versuchen wir möglichst nahe an die vorgegebene Maximaldistanz zu kommen.

Dafür ziehen wir folgende Ansätze in Betracht:

- Brute Force für kurze Distanzen oder wenige Punkte
- Binäre Suche für mittlere Distanzen oder einige Punkte
- Schrittweises Schrumpfen für lange Distanzen oder viele Punkte

Die Schwellwerte für wenige, einige oder viele Punkte sollen konfigurativ gesteuert werden.

Strategie 1: Brute Force

Bei dieser Strategie wird schrittweise Punkt für Punkt in eine Subliste eingefügt und diese dann an die Routing Engine gesendet. Wenn ein Resultat die Maximaldistanz überschreitet, wird das letzte Resultat unterhalb des Maximums verwendet. Alle anderen Resultate werden verworfen. Diese Abfragen an die Routing Engine sind nebenläufig, um die Geschwindigkeit zu erhöhen. Beispiel:

```
concurrent_requests = 5
few_points_threshold = 20
max_iterations = ceil(few_points_threshold / concurrent_requests) = 4
```

Nach 4 Iterationen wird im gewählten Beispiel ein passendes Resultat, wenn eines existiert, gefunden. Die Laufzeitkomplexität ist `O(n)`.

Strategie 2: Binäre Suche

Bei der Strategie mit der binären Suche wird mit der Hälfte der Punkte begonnen. Sie sendet die Subliste an die Routing Engine und überprüft die errechnete Distanz. Wenn die Differenz negativ ist, wird in der unteren Hälfte weiter gesucht, wenn positiv entsprechend in der oberen. Beispiel:

```
number_of_points = 60
max_iterations = ceil(log2(number_of_points)) = 6
```

Die binäre Suche findet bei 60 Elementen in der Liste nach spätestens 6 Iterationen ein passendes Resultat, falls vorhanden. Die Laufzeitkomplexität entspricht `O(log2(n) + 1)`.

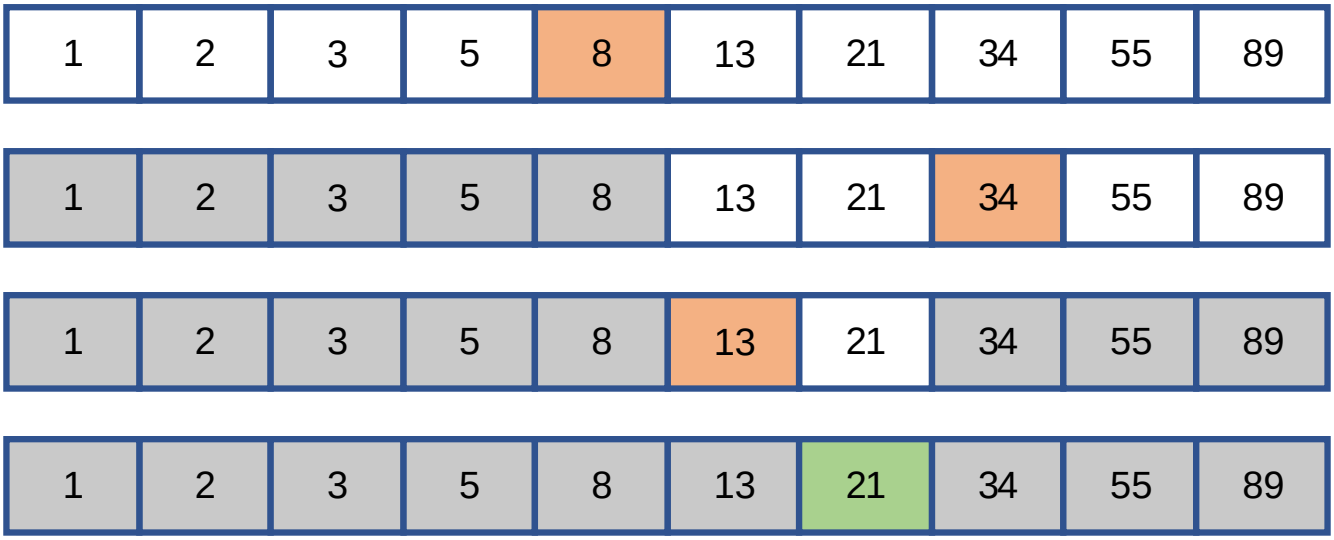


Abbildung 7. Beispiel einer binären Suche für gegebene Liste und gesuchtem Element 21

Strategie 3: Schrittweises Schrumpfen

Die letzte der drei Strategien verfolgt den Ansatz von schrittweisem Schrumpfen. Die Grösse der Subliste schrumpft dabei bei jedem Durchgang um einen konfigurierbaren Faktor. Die Anzahl der nebenläufigen Anfragen kann, wie in Strategie 1, konfiguriert werden. Beispiel:

```
concurrent_requests = 5
number_of_points = 80
shrink_factor = 0.9

sublist_1 = number_of_points * shrink_factor^0 = 80
sublist_2 = number_of_points * shrink_factor^1 = 72
sublist_3 = number_of_points * shrink_factor^2 = 64
sublist_4 = number_of_points * shrink_factor^3 = 58
sublist_5 = number_of_points * shrink_factor^4 = 52
```

Der Algorithmus endet, sobald die Routing Engine ein Resultat unter der Maximaldistanz gefunden hat. Je nach gewähltem Schrumpffaktor werden die Sublisten schneller oder langsamer kleiner. Die Laufzeitkomplexität ist im schlimmsten Fall `O(n)`.

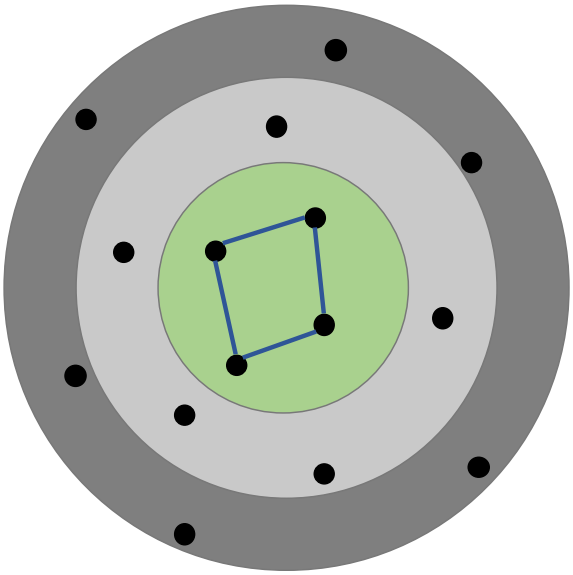


Abbildung 8. Schrittweises Schrumpfen der Punktewolke

6.2. Domäne

Im folgenden Abschnitt werden die Modelle definiert, mithilfe derer die Applikation umgesetzt werden soll. Der Hauptfokus liegt dabei auf einem Rundtrip und der Verknüpfung zu kategorisierten Interessen.

6.2.1. Domain Model

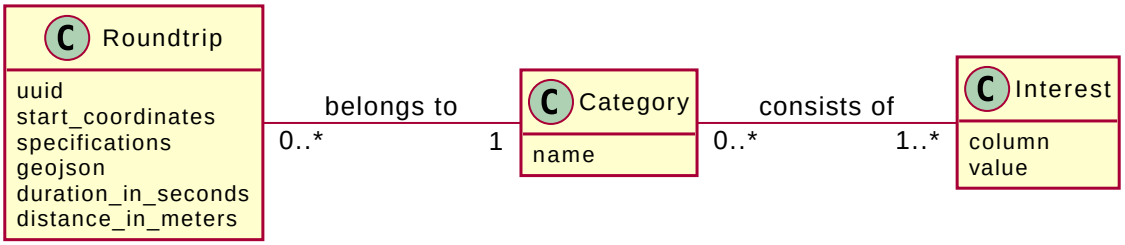


Abbildung 9. UML Klassendiagramm Domain Model City Trip Planner

Roundtrip

uuid	Der UUID wird als eindeutiger, nicht erratbarer Wert gebraucht, um Rundtrips erneut abzurufen.
start_coordinates	Dieses Feld speichert den Startpunkt eines Rundtrips.
specifications	In diesem Feld werden sämtliche Spezifizierungen für einen Rundtrip festgehalten. Dies ist für die erneute Anzeige, Analyse zum Benutzerverhalten und für die Fehlertoleranz von Bedeutung.
geojson	Das Resultat der berechneten Route inklusive gefundener Punkte werden in diesem Feld als GeoJSON abgespeichert. So muss ein Rundtrip bei erneutem Abruf nicht nochmals berechnet werden. Zusätzlich garantiert dieses Feld eine Rückwärtskompatibilität. Auch wenn sich die OpenStreetMap-Daten ändern, wird die berechnete Route mit den dazugehörigen POIs nach wie vor korrekt angezeigt.
duration_in_seconds	Die Dauer des Rundtrips wird als eigenes Feld geführt, um die Anzeige zu erleichtern.
distance_in_meters	Die Distanz des Rundtrips wird als eigenes Feld geführt, um die Anzeige zu erleichtern.

Category

name	Der Name einer Kategorie wird als Feld gespeichert.

Interest

column	Dieses Feld speichert den Spaltennamen einer Datenbanktabelle der importierten OpenStreetMap-Daten. Daraus werden die Filter für eine Kategorie erstellt.
value	Dieses Feld speichert den gesuchten Wert in einer Spalte der importierten OpenStreetMap-Daten. Daraus werden die Filter für eine Kategorie erstellt.

6.2.2. Design Model

Aus dem Domain Model lässt sich das folgendes Design Model ableiten.

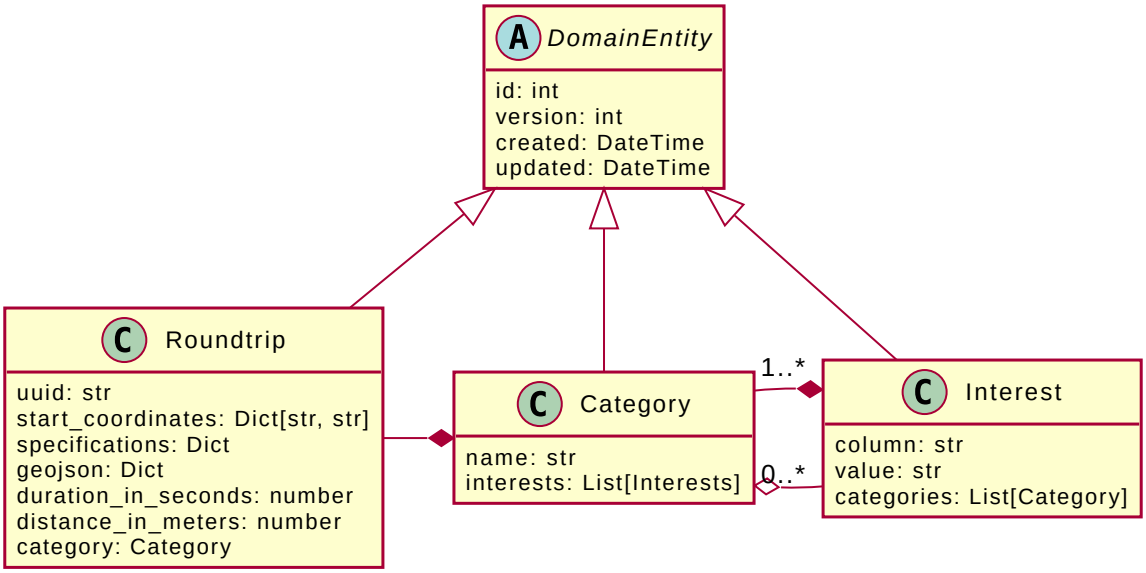


Abbildung 10. UML Klassendiagramm Design Model City Trip Planner

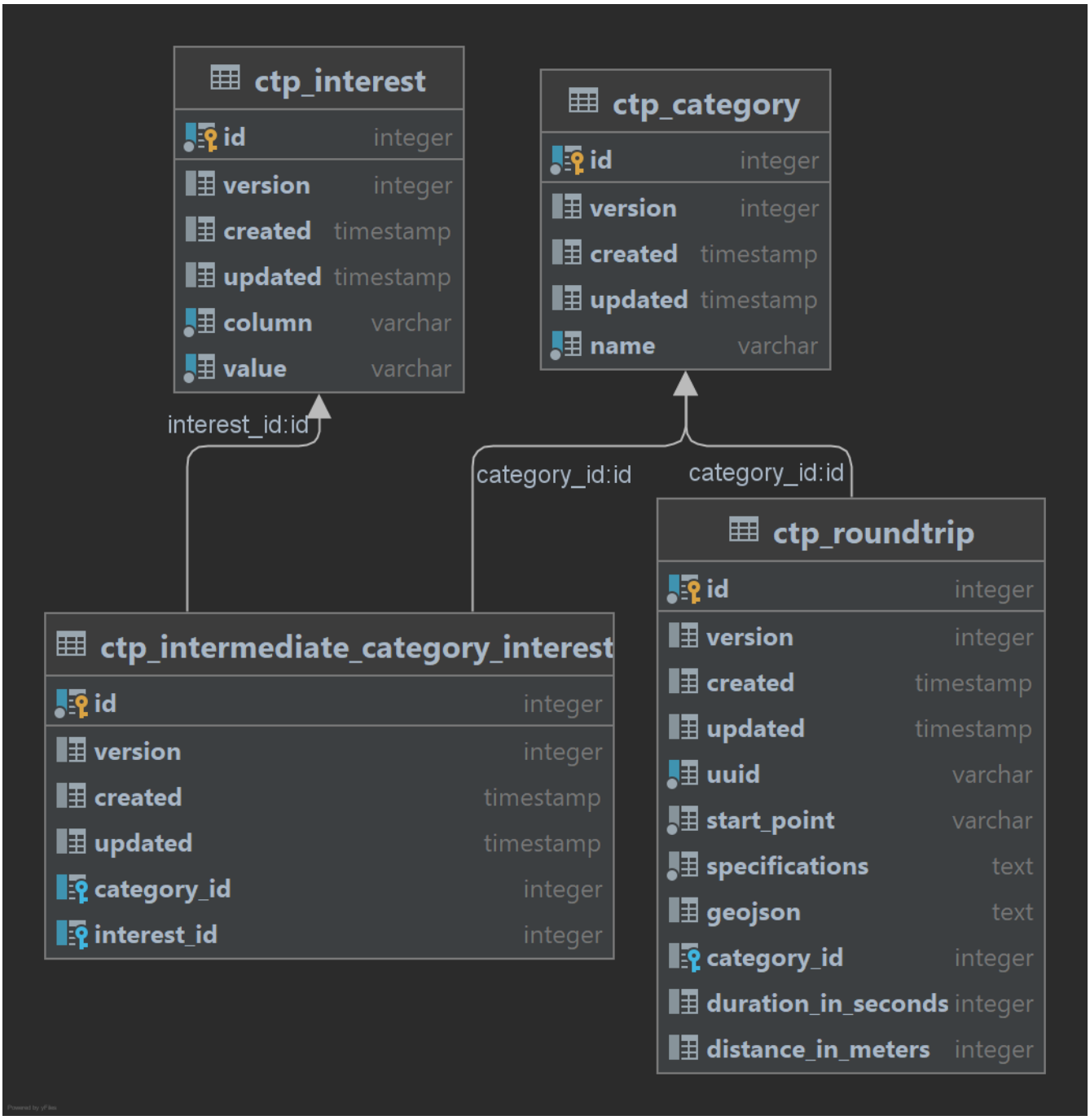
DomainEntity

Die Felder für die Persistenz werden in einer gemeinsamen Basisklasse gekapselt. Das bedeutet, dass jede Subklasse davon in der Datenbank gespeichert werden kann und sich nicht selber darum kümmern muss.

id	Generischer Wert für die eindeutige Identifizierung einer Entität der Domäne
version	Ganzzahliger Wert für optimistische Locking-Verfahren
created	Zeitpunkt, wann eine Entität angelegt wurde
updated	Zeitpunkt, wann eine Entität das letzte Mal aktualisiert wurde

6.2.3. Entity Relationship Diagram

Um die Objekte des Domain Models in der Datenbank abzuspeichern wurden die Tabellen folgendermassen erstellt.



Aufgrund des Object-relational Impedance Mismatch werden die geerbten Felder aus der objektorientieren Welt in der objektrelationalen Welt pro Tabelle geführt.

6.2.4. Objektkatalog

DomainEntity

Die abstrakte Klasse enthält ergänzende Felder für die konkreten Klassen aus der Domäne. Sie dienen für die Speicherung in einer Datenbank und bieten zusätzliche Informationen wie das Erstelldatum und das letzte Update einer Entität.

Category

Die Klasse Category steht für den Sammelbegriff von verschiedenen Interessen. Sie besitzt einen Namen und besteht aus mindestens einem Interest.

Interest

Ein Interest steht für eine Kombination von Spalte und dessen Wert der Datenbankeinträge in den OpenStreetMap-Tabellen. Sie werden für das Filtern von POIs verwendet.

Roundtrip

Ein Roundtrip steht für einen Rundtrip. Der Universally Unique Identifier wird als teilbare Information geführt, damit mehrere Personen auf den identischen Rundtrip zugreifen können. Im Startpunkt wird die Longitude und Latitude abgespeichert, von wo der Rundtrip startet. Die Spezifikation beinhaltet einen JSON-basierten Wert, welcher für das Erstellen eines Rundtrips relevant ist. Im Feld GeoJSON wird das Resultat der Routing Engine abgespeichert. Bei erneuten Zugriffen auf einen Rundtrip kann das Resultat direkt angezeigt und muss nicht mithilfe einer Routing Engine neu berechnet werden.

7. Architektur

Dieses Kapitel behandelt die Software Architekturen des Systems. Dabei wird unterschieden zwischen der Gesamtarchitektur, Backend-Architektur und der Frontend-Architektur. Die verschiedenen Architekturen wurden basierend auf den funktionalen und nicht-funktionalen Anforderungen erstellt. Die Architektur legt die Grundlage für die Entwicklung und ist somit von zentraler Bedeutung. Änderungen an der Software Architektur haben grosse Auswirkungen. Deshalb wurde die Architektur mit besonderer Sorgfalt erarbeitet und entworfen.

7.1. Kontextdiagramm

Das folgende Kontextdiagramm [14] dient als Einstieg. Dabei sind die wichtigsten Komponenten und Umsysteme aufgeführt. Sie müssen beim Entwurf der Architektur einbezogen werden und es müssen entsprechende Schnittstellen geschaffen werden.

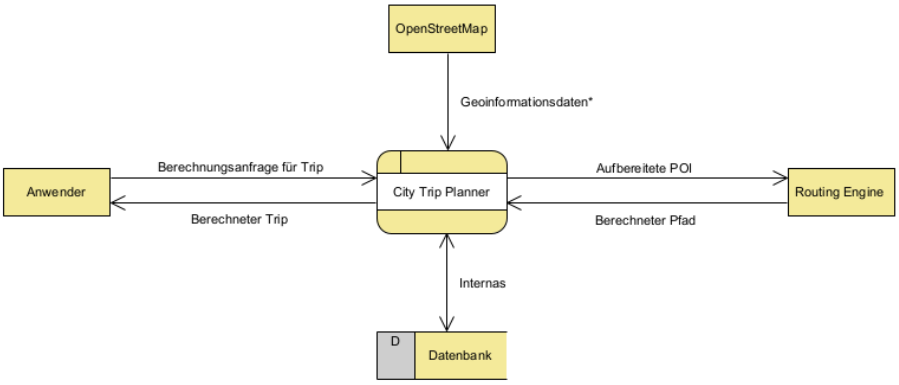


Abbildung 11. Kontextdiagramm [14] City Trip Planner

Als Grundlage für die Berechnung werden Geoinformationsdaten von OpenStreetMap verwendet. Nebst den Geoinformationsdaten wird für die Darstellung des berechneten Rundtrips auch Kartenmaterial von OpenStreetMap verwendet.

7.2. Architekturdiagramm Gesamtsystem

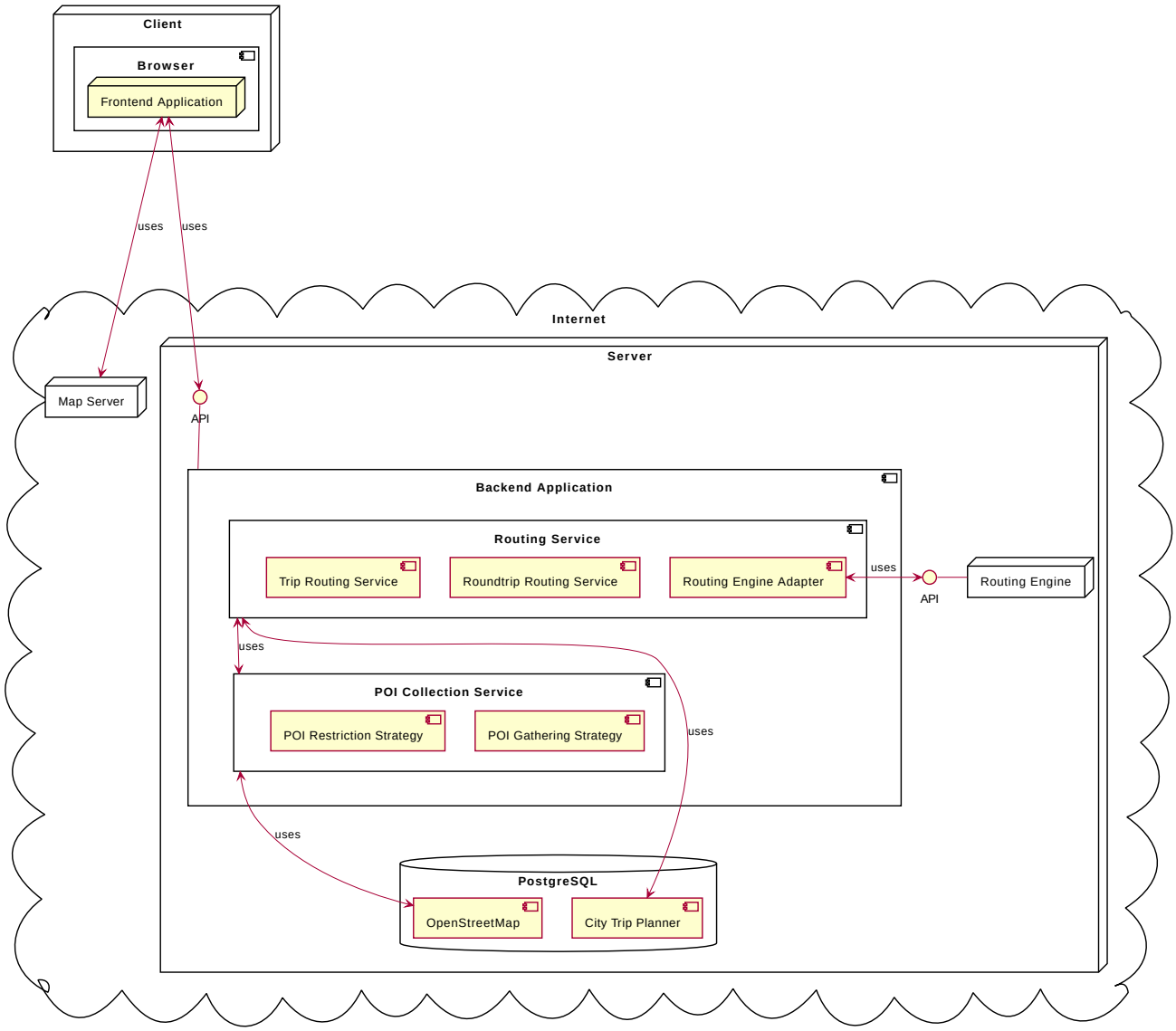


Abbildung 12. UML Komponentendiagramm [15] der Software Architektur für das Gesamtsystem City Trip Planner

Die Architektur des Gesamtsystems basiert auf einer Client-Server-Architektur. Die Frontend-Applikation läuft im Browser eines Anwenders. Diese Applikation kommuniziert mit der Backend-Applikation über eine REST-Schnittstelle und mit verschiedenen Kartenanbietern. Auf dem Server müssen folgende drei Komponenten betrieben werden:

Backend-Applikation

Die Backend-Applikation stellt die zentrale Einheit des Systems dar. Sie bietet eine Programmierschnittstelle an. Mithilfe dieser Schnittstelle können neue Rundtrips erstellt und die daraus entstehenden Daten bezogen werden.

Routing Engine

Die vom Backend verwendete Routing Engine wird auf dem Server betrieben. Sie ist eine eigenständige Installation und besitzt ihren eigenen Lebenszyklus. Die Kompatibilität der Versionen muss dabei stets geprüft werden.

PostgreSQL Server

Für die Datenhaltung muss ein PostgreSQL Server installiert werden. Dort werden sämtliche Daten von OpenStreetMap und die proprietären Daten des City Trip Planners gespeichert.

7.2.1. Ausführung zu Komponenten in der Backend-Applikation

Die Backend-Applikation besteht im Wesentlichen aus den beiden Hauptkomponenten für das Sammeln von POIs und das Erstellen von Routen. Es wurde im Diagramm auf Komponenten für Datenzugriffe zugunsten der Übersichtlichkeit verzichtet. Bei der Art von Routen wird aufgrund der Verschiedenartigkeit strikte zwischen Trip und Rundtrip unterschieden.

Der POI Collection Service besteht aus zwei Komponenten, welche die POIs filtern. Die POI Gathering Strategy nimmt eine Filterung aufgrund der Kategorisierung vor. Die POI Restriction Strategy filtert zusätzlich aufgrund der geografischen Lage und beispielsweise dem Namen. Das entstandene Set von POIs wird dann dem Routing Service überreicht.

Der Routing Service ist dafür zuständig, aus dem erhaltenen Set von POIs eine passende Route zu erstellen. Dazu wird ein Adapter verwendet, welcher die Aufbereitung der Daten so vornimmt, dass sie von einer Routing Engine verarbeitet werden können. Das entstandene Resultat wird dann in der Datenbank abgelegt, damit ein direkter Zugriff auf einen Trip oder Rundtrip möglich ist.

7.3. Referenzarchitektur Backend-Applikation

Robert C. Martin's Clean Architecture [16] ist die Referenzarchitektur für das Backend.

Der Architekturentwurf für das Backend von City Trip Planner sieht wie folgt aus:

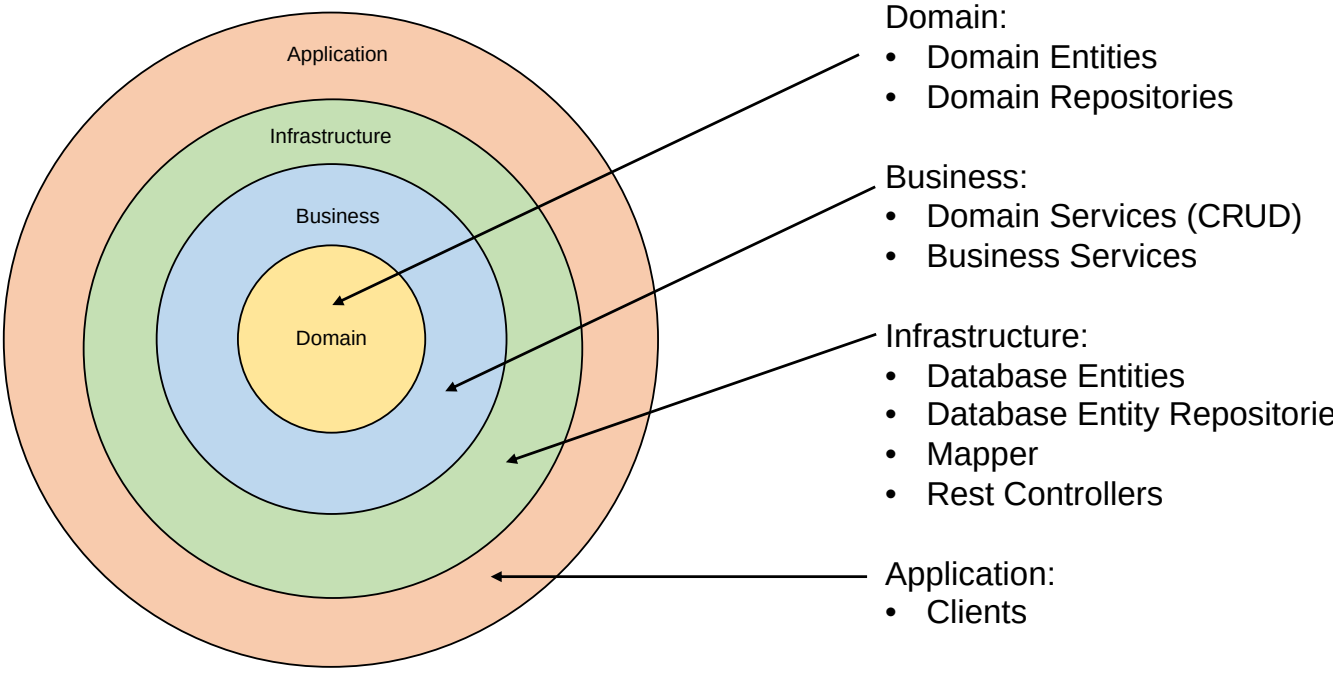


Abbildung 13. Architekturentwurf Backend-Applikation

7.3.1. Domain Layer

Das Domain Layer besteht lediglich aus den Domain Entities und dessen Repositories. Die Domain Entities sind ganz simple Datenhaltungsobjekte ohne weitere Geschäftslogik. Die Domain Repositories deklarieren den Vertrag, wie man auf die Domain Entities zugreifen kann.

Dieses Layer beinhaltet keine Third-Party-Libraries, sondern besteht nur aus sprachspezifischen Objekten.

7.3.2. Business Layer

Im Business Layer befindet sich sämtliche Geschäftslogik der Applikation. Einerseits sind das Services, welche CRUD-Operationen der Domäne anbieten und andererseits Services welche zwischen den Domain Entities vermitteln.

Das Business Layer besteht grösstenteils aus sprachspezifischen Objekten. Vereinzelt kann es vorkommen, dass einzelne Komponenten aus Third-Party-Libraries verwendet werden müssen. Beispielsweise für Transaction Handling, wenn die verwendete Programmiersprache keinen Standard dafür vorsieht.

7.3.3. Infrastructure Layer

Das Infrastructure Layer besteht nur aus Third-Party-Library Komponenten und Konfigurationen. Es ist frei von sämtlicher geschäftsbezogener Logik. Typischerweise wird hierzu ein Framework eingesetzt. Dieses Framework stellt beispielsweise Datenbankzugriffe und REST-Controller zur Verfügung. Da diese Komponenten nichts mit der eigentlichen Geschäftslogik zu tun haben, werden sie in einem eigenen Layer geführt.

7.3.4. Application Layer

Die äusserste Schale bezeichnen wir als Application Layer. In dieses Layer gehören sämtliche Applikationen und Services, welche mit der Applikation interagieren. In unserem Fall wird dies die Frontend-Applikation sein.

7.4. Referenzarchitektur Frontend-Applikation

Die Referenzarchitektur basiert auf den Angular Best Practices Lazy Loading Feature Modules [17].

Der Architekturentwurf für das Frontend von City Trip Planner sieht wie folgt aus:

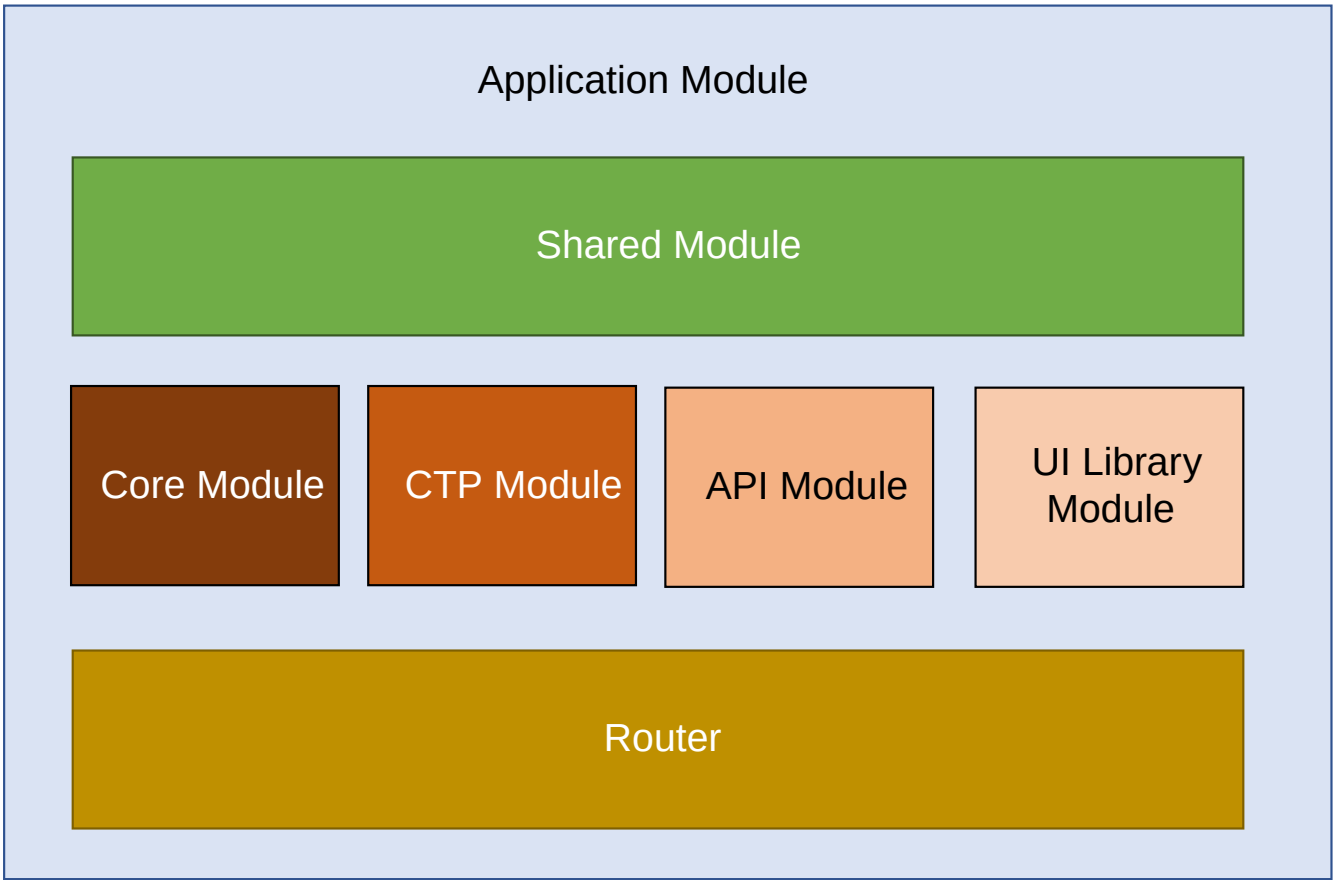


Abbildung 14. Architekturentwurf Frontend-Applikation

7.4.1. Application Module

Das Application Module ist das Grundgerüst der Applikation. Darin sind alle Framework-spezifischen Komponenten enthalten und es ermöglicht das Einbinden von weiteren Modulen.

7.4.2. Shared Module

Im Shared Module befinden sich Komponenten, welche in allen anderen Modulen verwendet werden. Typischerweise gehören selbstgebaute UI-Komponenten in dieses Modul.

7.4.3. Core Module

Das Core Module beinhaltet Interceptors, Guards und Services/Komponenten für die Authentisierung.

7.4.4. CTP Module

Das CTP Module enthält alle Komponenten und Services, welche für die Bedienung und Anzeige des City Trip Planners notwendig sind.

7.4.5. API Module

In diesem Modul befinden sich alle Models und Services für die Zugriffe auf die Backend-Applikation. Dieses Modul wird typischerweise mithilfe einer API Spezifikation generiert.

7.4.6. UI Library Module

In einem UI Library Module werden alle Komponenten einer UI Library deklariert. Damit erreicht man, dass sämtliche Importe einer Library an einem zentralen Ort vorgenommen werden und nicht über die gesamte Code-Basis gestreut sind.

Es ist möglich mehrere verschiedene UI Library Modules zu verwenden.

7.4.7. Router

Der Router stellt das Mapping von der URL zu den jeweiligen Modulen dar.

7.5. System Boundaries und Scopes

In diesem Abschnitt werden die Systemabgrenzungen und -bereiche definiert. Sie dienen dazu Fehlerquellen auf bestimmte Bereiche einzugrenzen. Dies dient der besseren Wartbarkeit, da schnell Rückschlüsse auf den Ursprung eines Fehlers gemacht werden können.

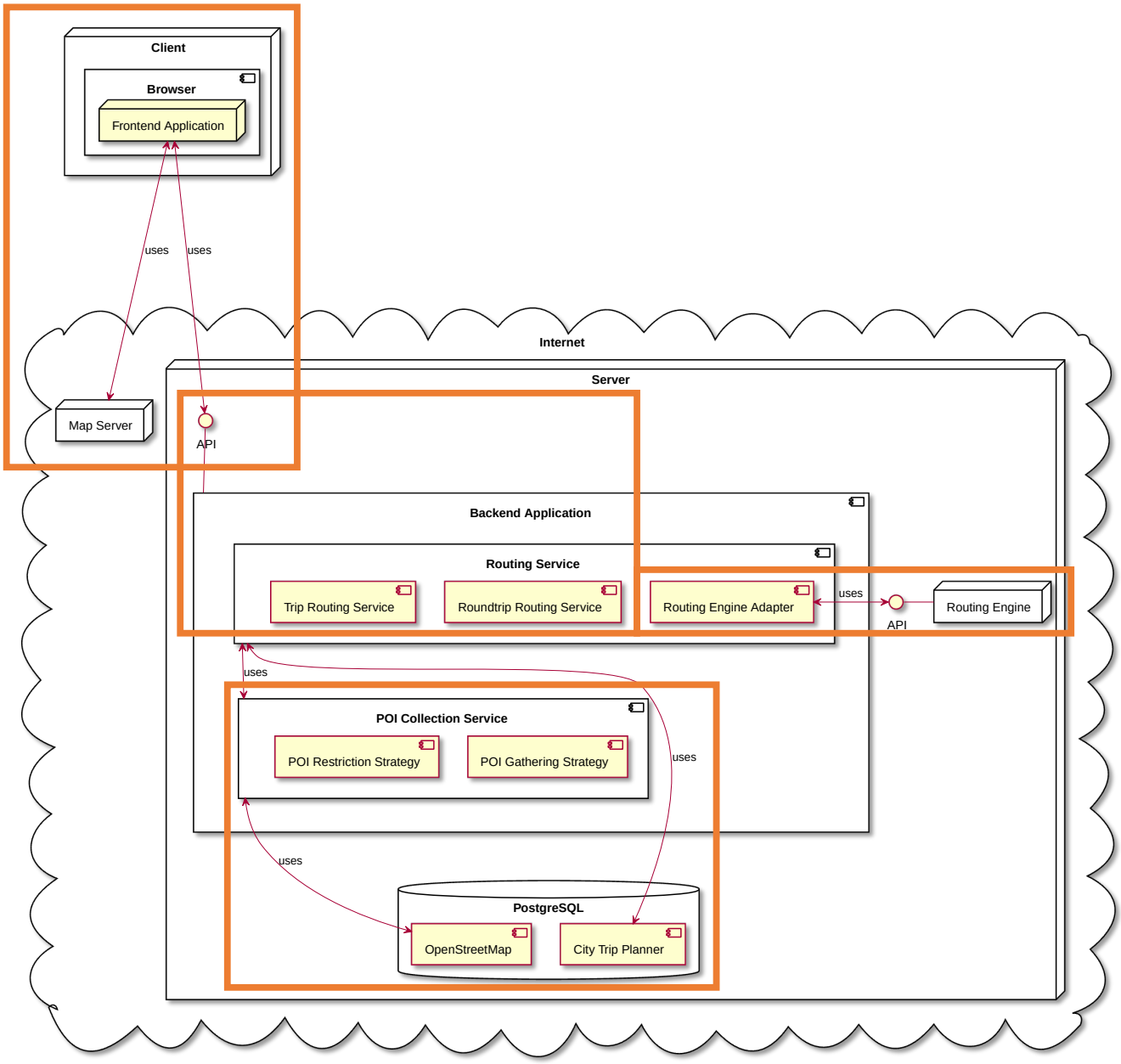


Abbildung 15. Systemabgrenzungen und -bereiche von City Trip Planner

7.5.1. Bereich Frontend

Die Frontend-Applikation ist eine eigene Komponente im System. Sie kommuniziert über HTTP mit verschiedenen Kartenservern und dem Backend. In diesem Bereich können folgende Fehler auftreten:

- Interne Programmierfehler
  - Falscher Ablauf bei der Erstellung eines Rundtrips
  - Fehlerhafte Darstellung der Karte
- Fehler bei der Kommunikation

    - Falsche Endpunkte
    - Fehlende Verbindung
- Probleme mit der Kompatibilität
    - Geräte
    - Browser

7.5.2. Bereich Backend API

Das Backend ist in drei Teilbereiche aufgeteilt. Ein Bereich ist die Bereitstellung einer Programmierschnittstelle. Dabei können folgende Fehler auftreten:

- Kompatibilität
    - API Version
    - Datentypen für Requests und Responses
- Angebotene Services
    - Falsche Verknüpfung
    - Validierung der Datenwerte

7.5.3. Bereich Backend Persistenz

Der Bereich Persistenz umfasst den Zugriff auf die Datenbank. Mögliche Fehler sind hierbei:

- Divergierendes Schema
- Transaktionsmanagement
- Datenintegrität und -aktualität

7.5.4. Bereich Backend Routing Engine

Der letzte Bereich befasst sich mit der Verbindung einer oder mehreren Routing Engines. Die Fehlerquellen belaufen sich hier auf:

- Fehlerhafte Aufrufe und Timeouts
- Datentransformationen von und zur Engine
- Software Lifecycle von Routing Engine

7.5.5. Relevanz für Studienarbeit

Die genannten vier Bereiche sind von zentraler Bedeutung für diese Studienarbeit. Fehlt eine Komponente, kann das System nicht funktionieren und die Anforderungen können nicht erfüllt werden. Der kritischste Punkt unter allen ist das Zusammenführen von Backend und Frontend. Um diese Herausforderung zu meistern, setzen wir auf OpenAPI [18]. Dadurch können wir die Kompatibilität zwischen den beiden Komponenten sicherstellen.

8. Design

Nachdem die Architektur nun feststeht, können die Entwürfe für Packages, Abfolgen und die Benutzeroberfläche gemacht werden. Der Fokus wird dabei auf das Backend gelegt, da diese Komponente die zentrale Rolle in der Architektur einnimmt. Die Referenzarchitektur dafür wurde bereits bestimmt und dieses Kapitel hält die erstellten Packages fest. Zudem wird auch der genaue Ablauf bestimmt, wie ein Rundtrip erstellt und erneut abgerufen wird. Die Mockups für die grafische Benutzeroberfläche finden sich im letzten Abschnitt.

8.1. Package Diagramme Backend

Das Ziel der gewählten Architektur ist es, dass alle äusseren Schalen auf der inneren - der Domäne - aufbauen. Ein weiterer Effekt ist, dass die Domäne keine Abhängigkeiten zu Frameworks oder Softwarebibliotheken von Drittanbietern hat. Demnach dürfen Abhängigkeiten nur gegen innen nicht aber gegen aussen zeigen.

8.1.1. Package Diagramm Gesamtübersicht

Das Package Diagramm orientiert sich an der Referenzarchitektur des Backend. Die Referenzarchitektur liest sich von innen gegen aussen. Aus Darstellungsgründen wird das Package Diagramm von oben nach unten gelesen.

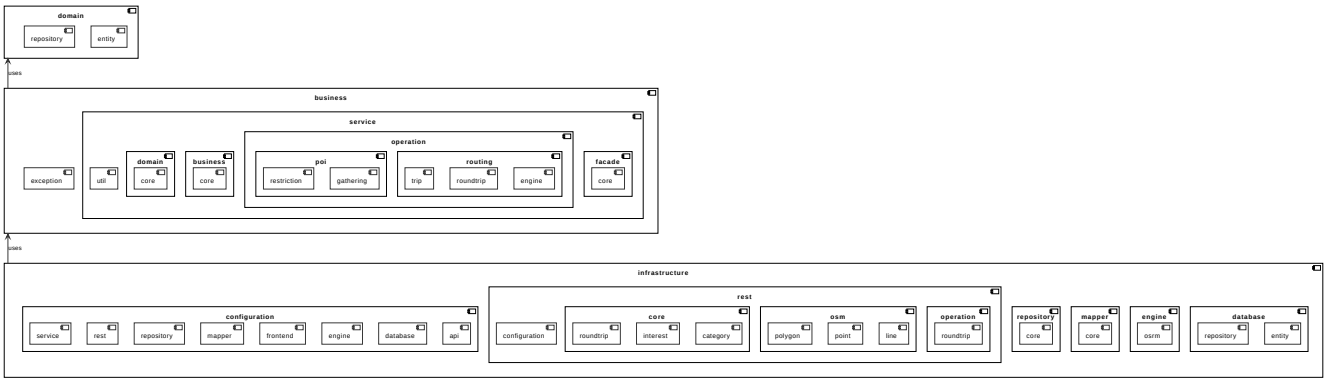


Abbildung 16. UML Komponentendiagramm [15] Package Diagramm City Trip Planner

Aufgrund der Komplexität und der Übersichtlichkeit verzichten wir auf die Verbindungen weiterer Abhängigkeiten.

8.1.2. Package Diagramm Domain Layer

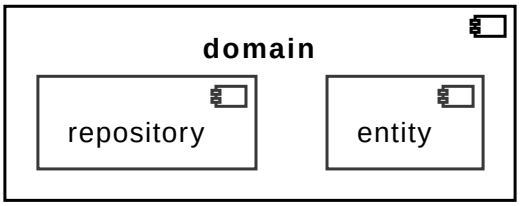


Abbildung 17. UML Komponentendiagramm [15] Package Diagramm City Trip Planner Domain Layer

Das Domain Layer stellt den Kern der Referenzarchitektur des Backend dar. In diesem Layer befinden sich die Klassen der Domäne und die Schnittstellen, welche definieren wie auf die Domäne zugegriffen werden kann. Diese Interfaces werden in der Infrastruktur-Schale implementiert, denn die Domäne ist frei von Softwarebibliotheken von Drittanbietern. In diesem Layer werden nur Standardfunktionalitäten der Programmiersprache angewandt.

8.1.3. Package Diagramm Business Layer

Das Business Layer stellt die erste Schale der Referenzarchitektur des Backend dar. Dieser Teil stellt die Grundlage sämtlicher Anwendungsfälle dar. Durch diese Wichtigkeit wurde sie bereits in der Architektur des Gesamtsystems abgebildet.

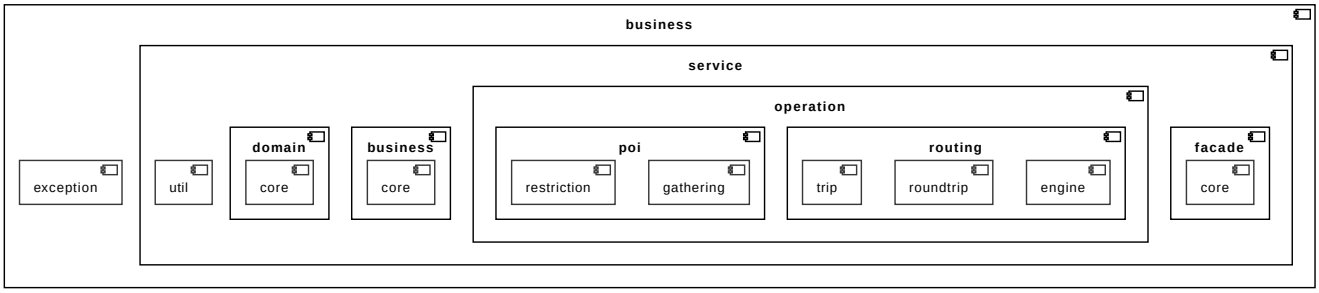


Abbildung 18. UML Komponentendiagramm [15] Package Diagramm City Trip Planner Business Layer

Dieses Layer beinhaltet sämtliche Geschäftsprozesse der Applikation. Unter anderem wird jeder Anwendungsfall in dieser Schicht implementiert. Dazu gehören beispielsweise sämtliche CRUD-Zugriffe auf die Domäne. Grundsätzlich sollte auch diese Schicht frei von Softwarebibliotheken von Drittanbietern sein. Es ist aber vereinzelt möglich, gegen diese Auflage aus guten Gründen zu verzichten.

Im Business Layer von City Trip Planner wurde von dieser Ausnahmeregelung Gebrauch gemacht. So werden die Kriterien für die SQL-Abfragen von geeigneten POIs bereits im Business Layer erstellt, aber erst in der Infrastruktur ausgeführt.

Diesen Spezialfall nahmen wir als Anlass, gegen die Vorgabe der Referenzarchitektur und zugunsten geringerer Komplexität zu verstossen. Die dabei eingegangene technische Schuld schätzen wir als gering ein. Es ist gängige Praxis, dass das Business Layer diesen Kompromiss eingeht, damit beispielsweise auch Transaktionen durch konkrete Geschäftsfälle gesteuert werden können.

8.1.4. Package Diagramm Infrastructure Layer

Das Infrastructure Layer stellt die zweite Schale der Referenzarchitektur des Backend dar.

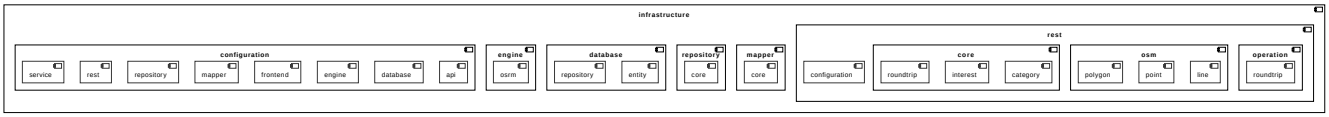


Abbildung 19. UML Komponentendiagramm [15] Package Diagramm City Trip Planner Infrastructure Layer

In dieser Schale befinden sich sämtliche Softwarekomponenten, welche von Drittanbietern stammen oder an diese angepasst werden müssen. Typischerweise sind das: Frameworks, Datenbanken, Adapter/Mapper

(ORM), REST-Controller und Konfigurationen für Dependency Injection und das Framework. Damit werden die Abhängigkeiten auf ein Minimum reduziert und können nachträglich ohne Anpassung der Domäne und der Geschäftslogik ausgetauscht werden.

8.2. Sequenzdiagramme

Die Sequenzdiagramme bestimmen den Grobablauf von Geschäftsprozessen. Der Hauptgeschäftsfall von City Trip Planner ist das Erstellen und Anzeigen von Rundtrips. Im Folgenden werden diese beiden Abläufe anhand von Diagrammen definiert.

8.2.1. Erstellen eines Rundtrips

Die Erstellung eines Rundtrips erfolgt nach folgendem Ablauf:

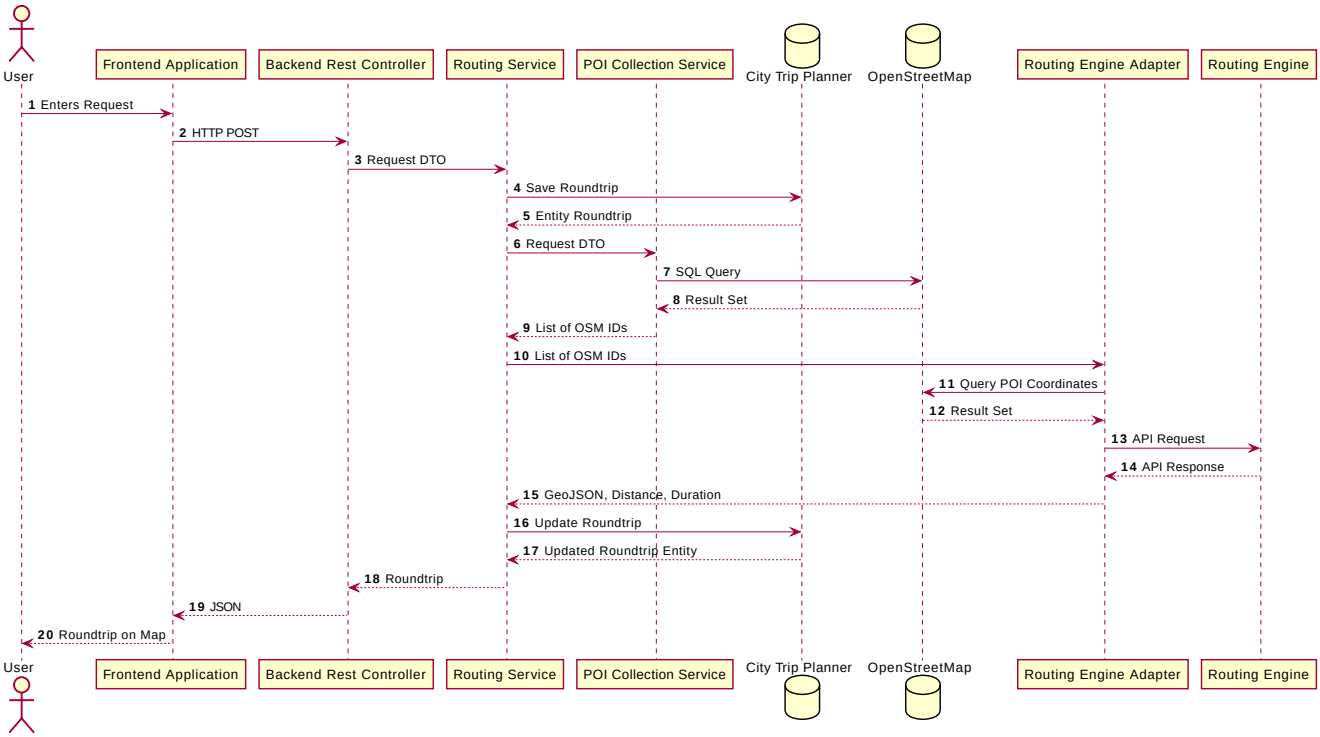


Abbildung 20. UML Sequenzdiagramm [19] der Erstellung eines Rundtrips

Der erstellte Rundtrip wird komplett in der proprietären Datenbank abgespeichert. Das ermöglicht einen erneuten Abruf desselben Rundtrips zu einem späteren Zeitpunkt, ohne dass die Routing Engine erneut einen Pfad berechnen muss. Dieser Ablauf ist im nächsten Abschnitt erklärt.

Das Abspeichern des GeoJSON wurde explizit so entworfen, damit bei Änderungen der OpenStreetMap-Daten eine Rückwärtskompatibilität garantiert werden kann!

8.2.2. Abfrage eines Rundtrips

Die erneute Abfrage eines bereits existierenden Rundtrips erfolgt nach folgendem Ablauf:

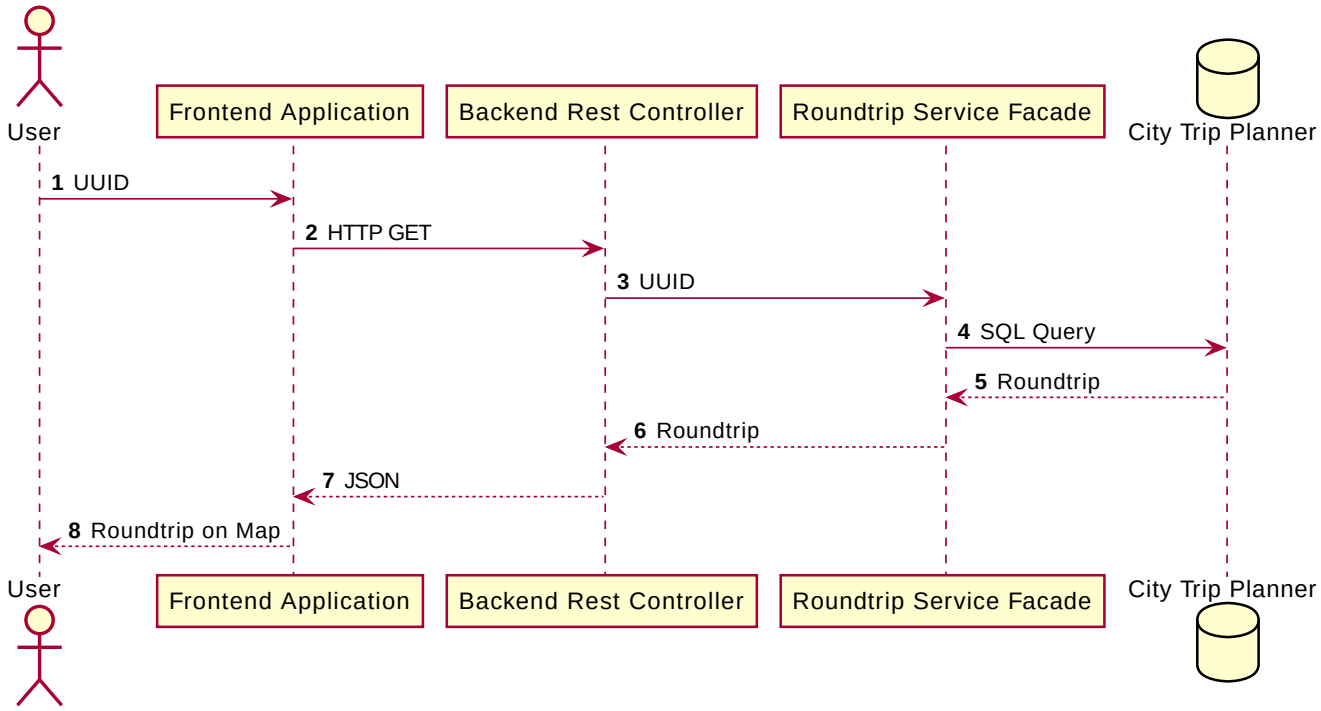


Abbildung 21. UML Sequenzdiagramm [19] der Abfrage eines existierenden Rundtrips

Dieser Vorgang wird für die erneute Darstellung eines Rundtrips gebraucht. Dabei entfällt die Anfrage an die Routing Engine, da das zuvor erstellte Resultat gespeichert wird. So kann auch ein anderer Benutzer, welcher im Besitz des Universally Unique Identifier ist, einen Rundtrip abfragen. Dies wird benötigt, wenn ein Rundtrip vom Desktop auf ein Mobilgerät übertragen oder zwischen Benutzer:innen geteilt wird.

Auf einen weiteren Detaillierungsgrad der Roundtrip Service Facade wurde zugunsten der Übersichtlichkeit verzichtet.

8.3. UI Design

Nachfolgend finden sich die erstellten Wireframes, für die wichtigsten Funktionalitäten unserer Applikation. Das Ziel ist ein intuitives User-Interface, dass an bereits existierende Routing- und Kartendienste angelehnt ist.

8.3.1. Anforderungen

- Mobile First: Die Anwendung sollte mit besonderem Augenmerk auf Kompatibilität von mobilen Geräten entwickelt werden.
- Die Eingabefelder sind sowohl in der Mobile wie auch der Desktop Version einfach bedienbar.
- Die Stationen auf dem Rundtrip werden mit zusätzlichen Informationen ergänzt.

8.3.2. Mockups

Die nachfolgenden Mockups wurden für die Anwendung auf mobilen Geräten und Desktops erstellt. Das Layout wurde demjenigen von Schweiz Tourismus [20] nachempfunden. Die vollständige Integration von City Trip Planner in das Corporate Design von Schweiz Tourismus ist nicht Bestandteil dieser Studienarbeit.

Übersicht

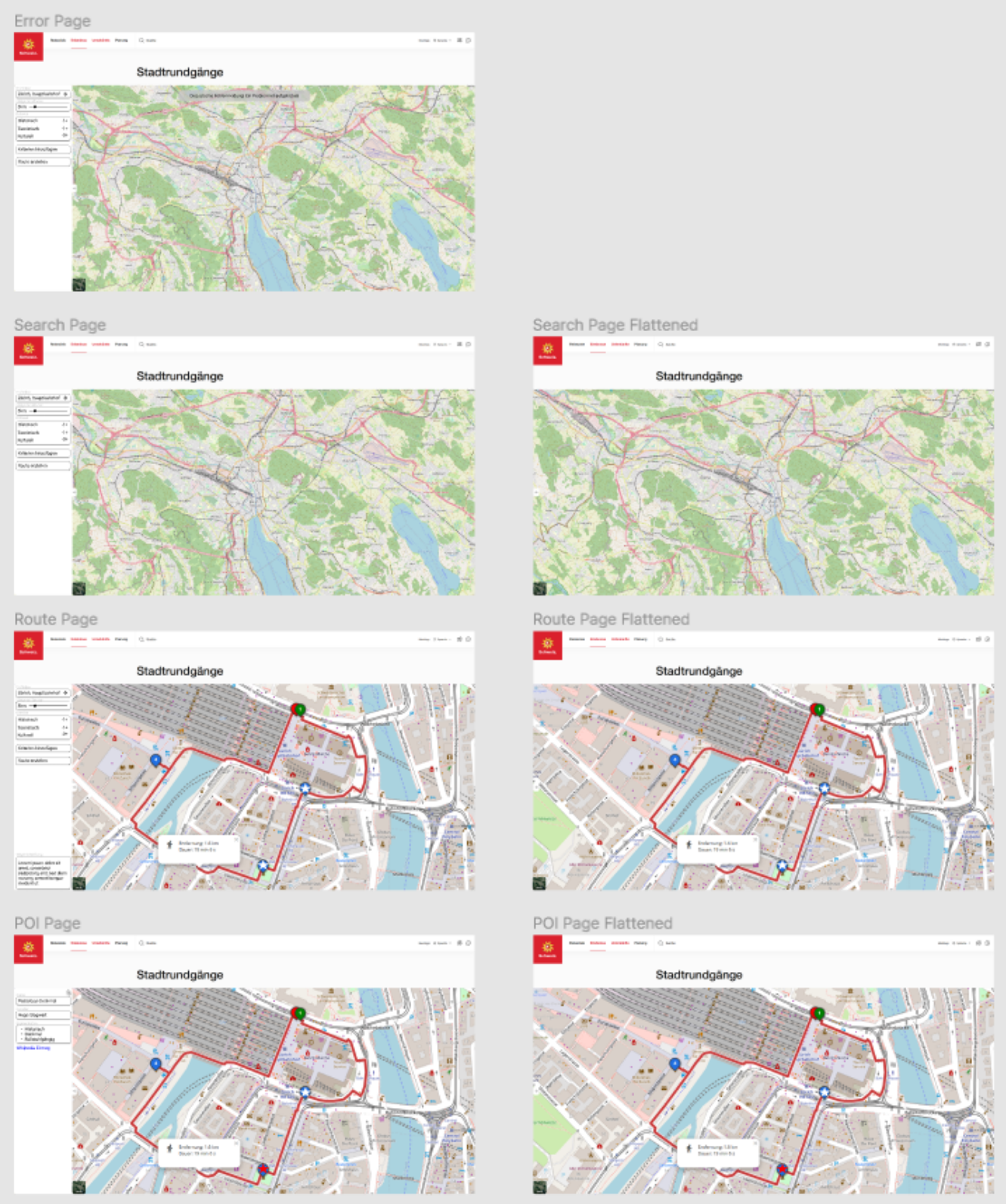


Abbildung 22. Alle Desktop Bildschirme

Routen-Suche

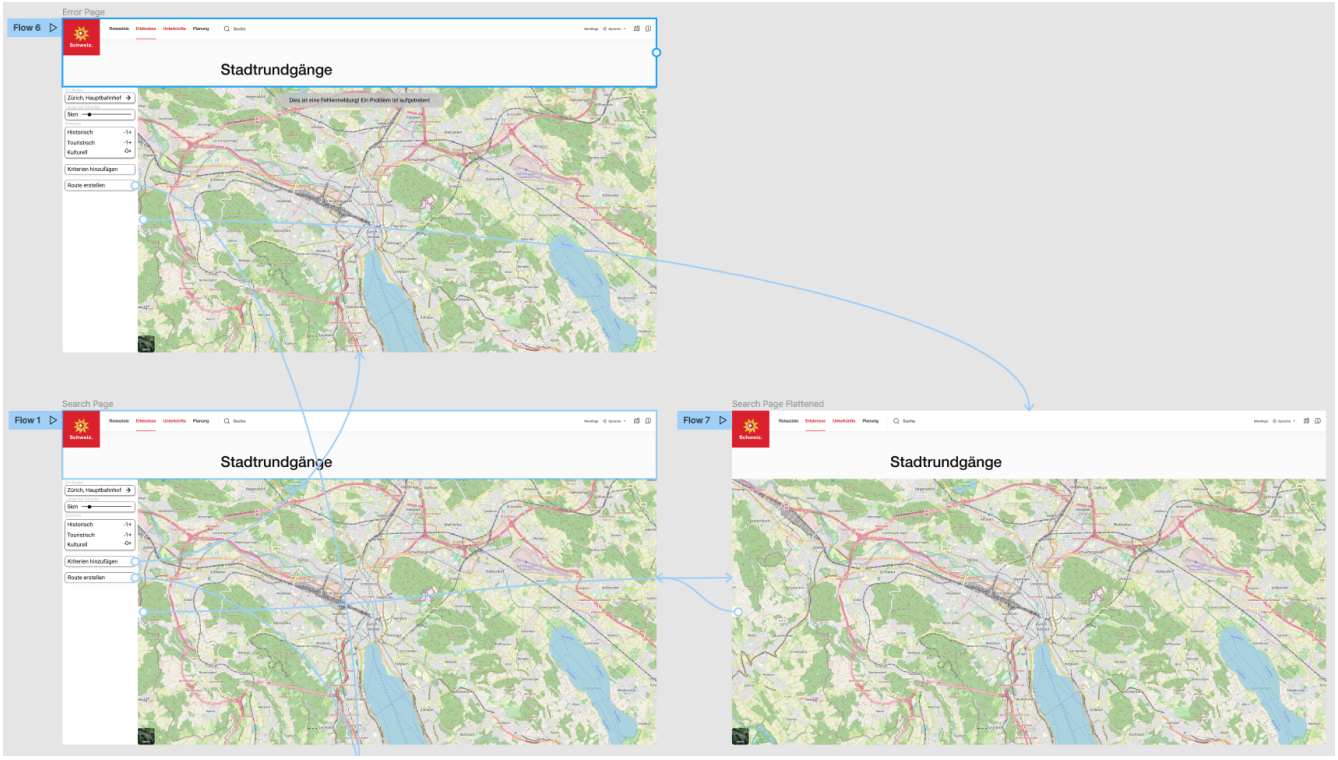


Abbildung 23. Routen-Suche Desktop

Routen-Darstellung Desktop

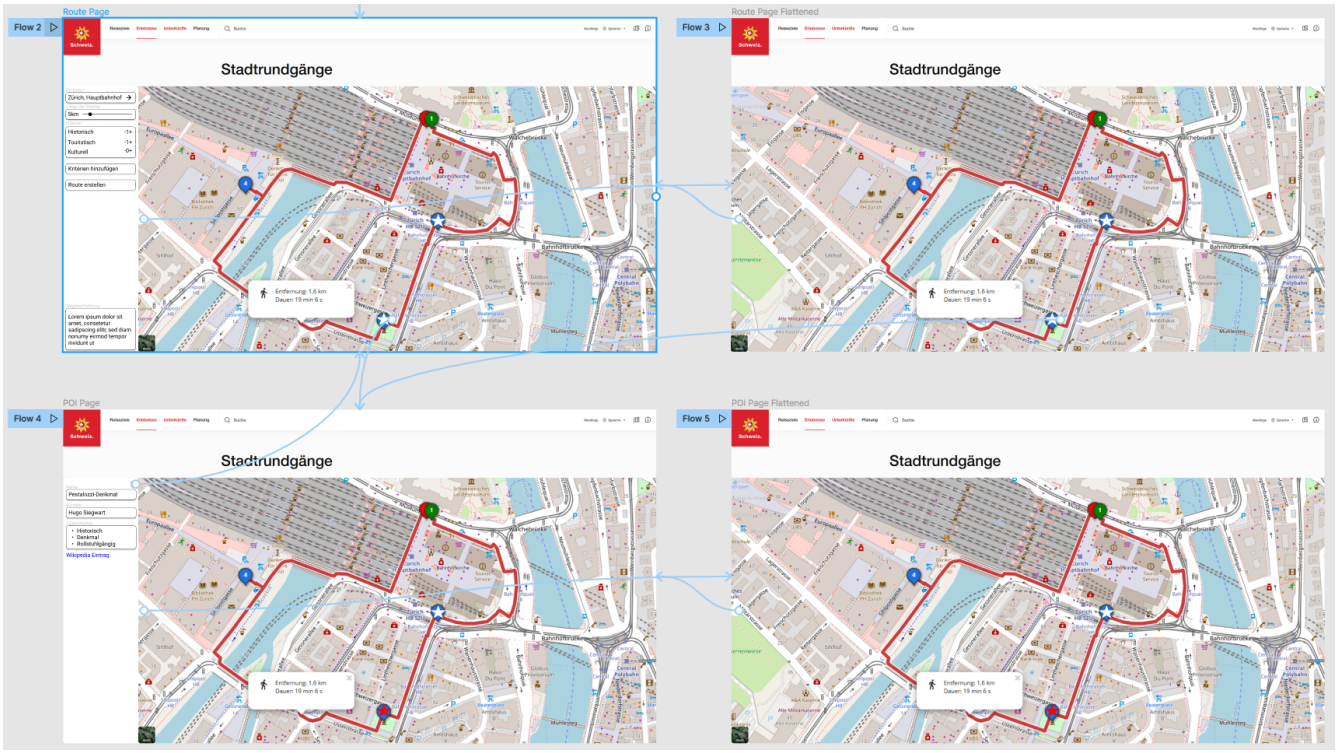


Abbildung 24. Routen-Darstellung Desktop

Übersicht

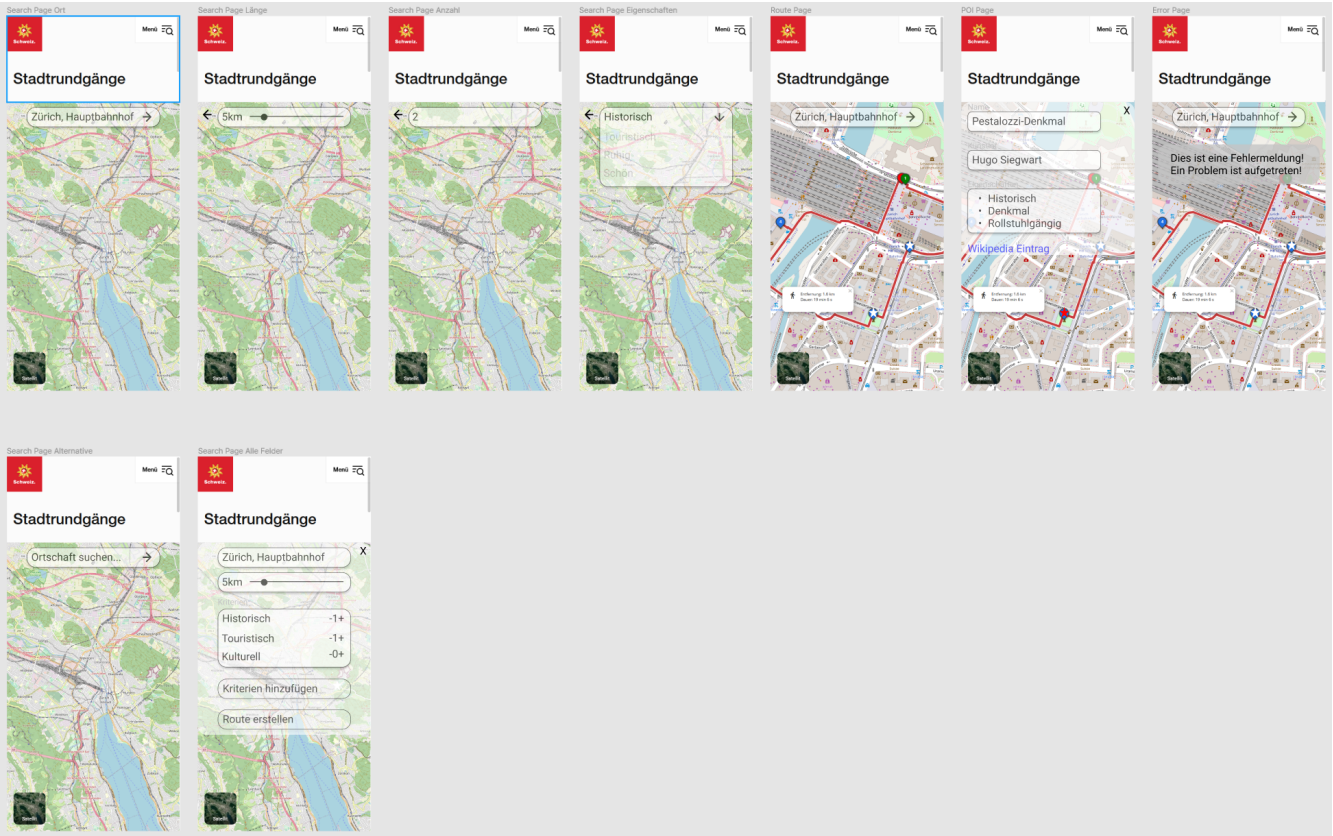


Abbildung 25. Alle Mobile Bildschirme

In der Mobile Variante erkennt man zwei verschiedene Ansätze. Einmal eine schrittweise Eingabe und einmal ein übersichtliches Menu mit allen Feldern.

Wir haben uns gemeinsam, mit Prof. Stefan Keller und Markus Dittli, für ein übersichtliches Menu entschieden.

Weitere Darstellungen der Mockup Screens finden sich im Anhang.

8.3.3. Verwendete Tools und Quellen

- Zur Erstellung dieser Mockups haben wir die Plattform Figma [21] genutzt.
- Die ersten Entwürfe haben wir mit diesem Google Maps UI Kit [22] angefertigt
- Der Kartenausschnitt auf dem Search-Screen kommt von OpenStreetMap [23]
- Der Map/Satellite Button kommt von Google Maps [24]
- Der Kartenausschnitt mit der eingezeichneten Route auf dem Screen der Routendarstellung kommt von Open Route Service [25]
- Die Beschreibungen der wichtigsten Punkte des Route-Screen kommen von Overpass Turbo [26]

9. Evaluation Routing Engine

9.1. Open Source Routing Machine - OSRM

Lizenz	OSRM läuft unter der BSD-Lizenz, einer reinen Open-Source-Lizenz. [27] [28] [29]
Konfiguration	Routing-Profile können mittels Lua-Scripts erstellt oder adaptiert werden. Dies erlaubt eine gewisse Flexibilität in der Benutzung von OSRM. In OSRM ist es nicht möglich, den Strassen und Wegen unterschiedliche Gewichtungen pro Request zu geben, Gewichtungen können nicht dynamisch zugewiesen werden. [27] [30]
Performanz	OSRM nutzt ausschliesslich Contraction Hierarchies, ist stark optimiert und sehr schnell. Auch lange Routen können innerhalb von Sekunden ausgerechnet werden. OSRM benötigt sehr viel Arbeitsspeicher für die Vorverarbeitung der Daten (250 GB pro Profil), aber auch zur Laufzeit (35 - 40 GB). [27] [30]
Fortbewegung	• Auto • Fussgänger • Fahrrad [28] [31]
Routing-Arten	Rundtrips sind mittels Trip-Service API möglich. • Routen • Alternative Routen • Rundtrips [27] [31] [32]

Zusätzliche Features	• Abbiegebeschränkungen beachten • Fahrspuren beachten • Wegweiser beachten • Matrix-Routing • Map-Matching / Track-Matching • Routen optimieren • Datentiles erzeugen und einzeln anzeigen • Routen-Kosten müssen nicht von der Zeit abhängig sein • Verkehrseinflüsse mittels CSV Dateien einberechnen [27] [31] [30]
Ausgabeformat	OSRM unterstützt diverse Formate. Zusätzlich gibt es ein weiteres öffentliches Projekt, die *Leaflet Routing Machine*. Diese kann genutzt werden, um die OSRM Response darzustellen. • polyline • polyline 6 • GeoJSON • komplette Übersicht • vereinfachte Übersicht [33] [34] [35]
Operation / API	OSRM wurde in C++ entwickelt und ist als Library verfügbar, sie ist direkt in einem Programm verwendbar. Man kann auch HTTP Requests an ein bestehendes Backend senden und es gibt ein optionales Web-Frontend. Das Projekt kann mit Docker aufgesetzt werden. [27] [36]
Dokumentation	Es gibt eine gute Dokumentation, diese ist aber nur in Englisch vorhanden. OSRM bietet eine grosse Menge an Informationen, Tutorials, Erfahrungsberichten und Erweiterungen. [27] [35] [30]

9.2. Graphhopper

Lizenz	Graphhopper läuft unter einer freien Apache-Lizenz. Teile davon sind aber nicht Open-Source und nur in der kostenpflichtigen Version verfügbar. [37] [38] [39]
Konfiguration	Routing-Profile können/müssen in Java geschrieben und kompiliert werden. Dies ist etwas komplexer und aufwändiger. Es können aber auch existierende Profile übernommen und angepasst werden. Personalisierte Profile können bei spezifischen Requests mitgegeben werden. [37] [40]
Performanz	Graphhopper nutzt Contraction Hierarchies, aber auch Dijkstra-/A*-Algorithmus, oder eine hybride Variante. Man kann schnelle Berechnungen machen, in denen man nicht flexibel ist, aber auch langsamere, flexiblere Berechnungen. Die Berechnungen mittels CH sind fast gleich schnell wie bei OSRM. Im flexiblen Modus dauert die Berechnung durchaus länger. Graphhopper benötigt sehr viel Arbeitsspeicher, aber nicht ganz so viel wie OSRM. Für die Vorverarbeitung der Daten ca. 64 - 128 GB pro Profil und zur Laufzeit bis zu 32 GB. [37] [40]
Fortbewegung	• Auto • Fussgänger • Fahrrad • Transit • Motorrad • Rollstuhl • Wanderer [31]
Routing-Arten	• Routen • Alternative Routen • Rundtrips [37] [31]

Zusätzliche Features	• Abbiegebeschränkungen (mittels A*-Algorithmus) • Wegweiser • Höhendaten • Map-Matching / Track-Matching • Routing-Kosten unabhängig von der Zeit • Isochrone API • Geocoding API • OSM-Daten Support • Flächen vermeiden • Dynamische Fahrzeug-Attribute [37] [38] [31]
Ausgabeformat	Graphhopper unterstützt diverse Formate. Zusätzlich gibt es ein weiteres öffentliches Projekt, die *Leaflet Routing Machine*. Diese kann genutzt werden, um die Graphhopper Response darzustellen. • Von der Routen API erhält man ein JSON mit einem Array von Objekten (RouteResponsePath) • Von der Matrix API erhält man ein JSON mit Arrays von Zahlen und Objekten • Von der Geocoding API erhält man ein JSON mit einem Array von Objekten (GeocodingLocation) • Von der Isochron API erhält man ein JSON mit einem Array von Objekten, eine Liste von Polygonen im GeoJSON-Format • Von der Cluster API erhält man ein JSON mit einem Array von Strings, Zahlen und einem Array von Objekten (Cluster) [41]
Operation / API	Graphhopper wurde in Java entwickelt und ist als Library verfügbar. Es gibt ein fertiges HTTP-Backend und ein optionales Web-Frontend. [37] [42]
Dokumentation	Es gibt eine gute Dokumentation, diese ist aber nur in Englisch vorhanden. Das Programm kann sehr schnell und einfach gestartet werden. [37]

9.3. Valhalla

Lizenz	Valhalla läuft unter der MIT-Lizenz, einer freizügigen Open-Source-Lizenz. [43] [44] [45]
Konfiguration	Die Profile müssen in Valhalla ebenfalls programmiert werden. Alle Profile nutzen den gleichen Graphen. Dies ist möglich, weil man zur Laufzeit, den Routen dynamische Kostenfaktoren zuweisen kann. Jede Routing-Anfrage kann mit anderen Kosten berechnet werden. Valhalla kann Abfahrt- und Ankunftszeiten berechnen, der Algorithmus nutzt zeitliche Einschränkungen und kann sogar historische Verkehrsdaten zur Routenberechnung verwenden. [43] [46]
Performanz	Valhalla nutzt einen eigenen A*-Algorithmus. Die Geschwindigkeit ist deutlich langsamer als mit Contraction Hierarchies, aber immer noch ziemlich schnell. Das Matrix-Routing dauert vergleichsweise sehr lange. Valhalla benötigt allerdings massiv weniger Arbeitsspeicher, 16 GB für alle Profile. Dies wird durch einen, in Kacheln unterteilten, hierarchischen Graphen gelöst. [43] [46] [47]
Fortbewegung	• Auto • Lastwagen • Fussgänger • Fahrrad • Transit • Taxi • Bus • Roller • Motorrad [31]
Routing-Arten	• Routen • Alternative Routen [31]

Zusätzliche Features	Valhalla besitzt eine komplexe Plugin-Architektur • Abbiegebeschränkungen • Wegweiser • Höhendaten (optional) • Tile-basierte Datenhaltung • Dynamische Kosten • Matrix-Routing • Isochron • Intermodal • Map-Matching • Optimierung (Traveling Salesman Problem) • Integriertes Python SDK • QGIS Plugin • Flächen vermeiden • Mobile / Offline nutzbar [43] [46] [31]
Ausgabeformat	Valhalla unterstützt diverse Formate. Zusätzlich gibt es ein weiteres öffentliches Projekt, die *Leaflet Routing Machine*. Diese kann genutzt werden, um die Valhalla Response darzustellen. • Von der Turn-by-Turn Route API erhält man ein JSON • Von der Isochrone API erhält man die Konturen als GeoJSON • Von der Map-Matching API erhält man ein JSON • Von der Locate API erhält man ein JSON • Mittels Mjolnir kann offenbar ein GeoJSON erzeugt werden [48] [49] [34]
Operation / API	Valhalla wurde in C++ entwickelt. Es gibt ein fertiges Ubuntu/Debian Paket, dass installiert werden kann. Es gibt ein Web-Frontend. [43] [44] [50]
Dokumentation	Es gibt eine gute Dokumentation, diese ist aber nur in Englisch vorhanden und kann manchmal etwas überwältigend sein. Das Programm kann sehr schnell und einfach gestartet werden. [43] [46]

9.4. Openrouteservice - ORS

Lizenz	Laut OpenStreetMap-Wiki läuft ORS unter einer freien Apache-Lizenz. In der Dokumentation von ORS selbst steht, dass alle Resultate von ORS unter die CC-BY 4.0 Lizenz fallen. [51] [39] [52] [53]
Konfiguration	ORS Routen können parametrisiert werden. Die Gewichtungen wirken sich direkt auf das Resultat aus, beinflussen allerdings auch die Performanz. Zusätzliche Profile werden in Java geschrieben und kompiliert. Meist müssen zwei Projekte angepasst werden: ORS und Graphhopper. [54]
Performanz	Die Performanz ist vergleichbar schnell mit Graphhopper, auch lange Routen benötigen nur hunderte Millisekunden. ORS benötigt allerdings sehr viel Arbeitsspeicher, mindestens 128 GB pro Profil. [54]
Fortbewegung	• Auto • Lastwagen • Fussgänger • Fahrrad • Rollstuhl • Wandern [31]
Routing-Arten	• Routen • Alternative Routen • Rundtrips [31]
Zusätzliche Features	• Isochron • Matrix-Routing • Flächen vermeiden • Dynamische Fahrzeug-Attribute • Höhendaten • Routen optimieren [31] [55]

Ausgabeformat	ORS unterstützt eine grosse Menge an Ausgabeformaten: • Von der Directions API erhält man ein JSON, GeoJSON, oder GPX • Von der Isochrones API erhält man ein GeoJSON • Von der Matrix API erhält man eine Matrix mit spezifizierten Werten • Von der Geocode API erhält man ein JSON • Von der POIs API erhält man ein GeoJSON • Von der Elevation API erhält man ein GeoJSON, Polyline 5 oder 6, oder einen Point • Von der Optimization API erhält man einen Antwortcode [56]
Operation / API	ORS ist in Java geschrieben und bietet kleinere Nebenprojekte und Microservices. Es gibt eine Höhendaten HTTP API, eine POI HTTP API, ein Routing-Problem-Löser, Geocoding, ein QGIS Plugin und eine Python / R Library für alle APIs. [54] [57] [58]
Dokumentation	Die Dokumentation ist nur in Englisch vorhanden, ist übersichtlich und kurz gehalten. Die API-Beispiele wurden mittels Swagger-Editor konfiguriert, man muss aber einen validen Key besitzen, um sie zu testen. [56]

9.5. pgRouting

Lizenz	pgRouting läuft unter mehreren Lizenzen. • GNU General Public License v2.0 or later • Boost Software License - Version 1.0 • MIT-X License • Creative Commons Attribution-Share Alike 3.0 License [59] [60] [61] [45] [62]
Konfiguration	Die Konfigurationen von pgRouting werden allesamt in SQL vorgenommen. Dies ermöglicht unglaublich variable Konfigurationen, bedeutet aber auch einen Mehraufwand. Alle Eigenschaften können zur Laufzeit, pro Request, verändert werden. pgRouting ist eine der flexibelsten Routing Engines, die es zurzeit gibt. [31] [63]
Performanz	pgRouting benötigt nicht viel Arbeitsspeicher, um die Graphen zu berechnen, allerdings dauern die Berechnung eher lange. Eine Abfrage sollte nicht mehr als 1 Million Zeilen umfassen, man muss das Resulat stark einschränken. Angeblich sollen Contraction Hierarchies ebenfalls verwendet werden können, diese bieten wohl eine höhere Performanz. [63]
Fortbewegung	Alle denkbaren Routings sind grundsätzlich möglich, müssen aber in Profilen definiert werden. • Auto • Lastwagen • Fussgänger • Fahrrad • Transit • und weitere ... [31]

Routing-Arten	pgRouting bietet keine vordefinierten Rundtrips oder alternative Routen, diese können allerdings sicherlich mittels Abfragen erreicht werden. • Routen • (Alternative Routen) • (Rundtrips) [31]
Zusätzliche Features	• Isochron • Matrix-Routing • Routen optimieren • Abbiegebeschränkungen • Flächen vermeiden • Dynamische Fahrzeug-Attribute • Höhendaten [31]
Ausgabeformat	Die Resultate von pgRouting können dank PostGIS in unterschiedlichen Formaten ausgegeben werden, auch GeoJSON. [64]
Operation / API	pgRouting wurde mehrheitlich mit PLpgSQL, C++ und C geschrieben und kann einfach als Erweiterung für PostgreSQL und PostGIS installiert werden. PostGIS stellt dabei geografische Objekte und Funktionen zur Berechnung von Daten zur Verfügung. Die Installation via Docker funktioniert sehr gut. [63] [65]
Dokumentation	Die Dokumentation ist nur auf Englisch vorhanden, aber das Setup ist sehr gut dokumentiert. [66]

9.6. Itinero

Lizenz	Itinero läuft unter der MIT-Lizenz, einer freizügigen Open-Source-Lizenz. [67] [68] [45]
Konfiguration	Es können Profile mittels Lua-Scripts erstellt werden. [67]
Performanz	Itinero nutzt Contraction Hierarchies, aber auch Dijkstra-/A*-Algorithmus. Die Performanz ist durchschnittlich gut, sie liegt bei ca. 2 - 4 Sekunden für längere Routen. Dies ist etwas langsam für Contraction Hierarchies, lässt sich aber noch nutzen. Möglicherweise ist die Performanz stark von der Konfiguration abhängig. [67]
Fortbewegung	• Auto • Grosser Lastwagen • Kleiner Lastwagen • Fussgänger • Fahrrad • Bus • Roller • Motorrad [69]
Routing-Arten	Wir haben keine genaueren Informationen über Itinero gefunden, gehen aber davon aus, dass Rundtrips möglich sind. • Routen • Mutmasslich Rundtrips [70]

Zusätzliche Features	• Abbiegebeschränkungen • Routen optimieren • Heatmaps • Intermodal • Bike Sharing • Zug und Bus Transit • Analyse Tools • Offline Routing [67] [68]
Ausgabeformat	Die API muss mittels *Content-Type: application/json* angefragt werden, um eine Route als GeoJSON zu erhalten. • GeoJSON [71]
Operation / API	Itinero ist in C# mit .NET geschrieben, ist für Windows geeignet, lässt sich mit Mono aber auch unter Linux nutzen. Itinero ist eigentlich als Library konzipiert, um in C# eingebunden zu werden. Es gibt auch ein optionales Web-Interface. [67] [72]
Dokumentation	Die Dokumentation ist Work in Progress (Stand 2017), nur in Englisch verfügbar und noch nicht sehr ausgereift. [67]

9.7. BRouter

Lizenz	BRouter lief unter der GPL-Lizenz, einer Share-Alike-Lizenz. Laut einem Kommentar auf GitHub, hat man auf eine MIT-Lizenz gewechselt. [73] [74] [45]
Konfiguration	Routing-Profile werden in einer eigenen Sprache geschrieben. Für das Fahrrad-Routing können sehr gute Profile erstellt werden. [73] [75]
Performanz	BRouter nutzt einen A*-Algorithmus. Die Rechenzeit war eher langsam, für 1000 km ca. 20 Sekunden. Für kürzere Routen ist die Rechenzeit bei wenigen Sekunden. Die Zeitdauer wird mehrheitlich von Konfigurationen und verwendeter Fortbewegung abhängig sein. Höhendaten werden standardmässig einberechnet. [73]
Fortbewegung	BRouter steht für BicycleRouter und hat sich einen guten Namen im Bereich Fahrrad-Routing gemacht. Fussgänger-Profile können wohl erstellt werden, sind aber umständlich. • Auto • Fahrrad • Fussgänger [73]
Routing-Arten	Rundtrips werden nicht als Funktion zur Verfügung gestellt, können aber eingezeichnet werden. • Routen • Alternative Routen • Rundtrips [73]
Zusätzliche Features	• Abbiegebeschränkungen • Höhendaten • Flächen vermeiden [73]

Ausgabeformat	Wir haben keine Angaben zu möglichen Ausgabeformaten gefunden.
Operation / API	BRouter wurde in Java geschrieben und ist als Library konzipiert. Es gibt auch ein optionales Web-Interface. [73] [76]
Dokumentation	Die Dokumentation ist nur in Englisch vorhanden und etwas unübersichtlich. Die Webseite von BRouter bietet wenige Informationen. [73]

9.8. Routino

Lizenz	Routino läuft unter der AGPL-Lizenz, einer Share-Alike-Lizenz. [77] [78] [79] [80]
Konfiguration	Profile werden als XML definiert. Es wird die schnellste oder kürzeste Route gesucht. Um das Routing zu optimieren, wird ein spezielles DB-Format verwendet. [77] [78]
Performanz	Routino nutzt einen eigenen A*-Algorithmus und ist daher eher langsam. Berechnungen dauern bei Distanzen von 1000 km ca. 10 Sekunden. [77]
Fortbewegung	• Auto • Lastwagen • Fussgänger • Fahrrad • Transit • Taxi • Bus • Roller • Motorrad • Rollstuhl • Pferd [81]
Routing-Arten	• Routen • Rundtrips [81]
Zusätzliche Features	• Abbiegebeschränkungen • Strassen-Eigenschaften einschränken • Geschwindigkeitslimit • Gewicht einschränken • Höhe, Breite und Länge einschränken [77] [78]

Ausgabeformat	Routino unterstützt 3 Formate, aber kein GeoJSON. • HTML • GPX XML • Text [82]
Operation / API	Routino wurde in C geschrieben und ist ein Standalone-Router, ein Binary das aufgerufen werden kann. Man kann auch ein CGI-Skript ausführen. Es gibt ein Web-Interface mit vielen Daten und einer guten Debug-Ansicht. [77] [83]
Dokumentation	Die Dokumentation ist gut, allerdings nur in Englisch verfügbar. [77]

9.9. OSMnx

Lizenz	OSMnx läuft unter einer MIT-Lizenz, einer freizügigen Open-Source-Lizenz. [84] [45]
Konfiguration	Man kann definieren, welche geografischen Bereiche benötigt werden. Zusätzlich können auch Filter angegeben werden, um nach bestimmten Tags zu suchen. [85]
Performanz	Diese Engine ist eher gedacht für die Verarbeitung von kleineren Datensätzen. Sie bezieht die Daten automatisch von der Overpass API. [85]
Fortbewegung	• Auto • Fahrrad • Fussgänger [84]
Routing-Arten	Wir haben keine weiteren Angaben zu alternativer Routenberechnung, oder Rundtrips gefunden. • Routen
Zusätzliche Features	• POIs • Gebäude-Umrisse • Höhendaten • Orientierung • Geschwindigkeit [84]
Ausgabeformat	OSMnx gibt die Daten in einem GeoPandas Objekt zurück. • GeoDataFrame [85]

Operation / API	OSMnx wurde in Python geschrieben. Die Installation ist einfach und gut beschrieben. Man kann OSMnx via Conda, Pip oder in einem Docker Container installieren. OSMnx verwendet NetworkX für die Arbeit mit Graphen und Netzwerken. [85] [84] [86]
Dokumentation	Die Dokumentation ist nur in Englisch verfügbar und nicht wirklich übersichtlich. [84]

9.10. Pyrosm

Lizenz	Pyrosm läuft unter einer MIT-Lizenz, einer freizügigen Open-Source-Lizenz. [87] [45]
Konfiguration	Der Bereich, in dem die Daten bezogen werden sollen, kann in Pyrosm nicht genau definiert werden. Man kann Kontinente, Länder, Regionen und Städte herunterladen und verwenden. Es gibt 4 verschiedene Kategorien, die verschiedene Daten für Strassennetzwerke mit einschliessen, oder eben weglassen. [85]
Performanz	Pyrosm kann grosse Datenmengen schnell verarbeiten, weil es die Daten in lokalen osm.pbf Dateien ablegt. Diese Engine ist gut geeignet, um grosse Datenmengen zu verarbeiten. [85]
Fortbewegung	• Auto • Fahrrad • Fussgänger [85]
Routing-Arten	Wir haben keine weiteren Angaben zu alternativer Routenberechnung, oder Rundtrips gefunden. • Routen
Zusätzliche Features	• PBF Daten herunterladen • Strassennetzwerke • Gebäude • POIs • Grenzen • Netzwerke als Graph exportieren [87]

Ausgabeformat	Pyrosm gibt ein GeoDataFrame mit MultilineStrings zurück. Es können auch Linestrings oder Points zurückgegeben werden. Damit kann ein Graph erzeugt werden. • MultilineStrings • Linestrings • Points [85]
Operation / API	Pyrosm wurde in Python geschrieben. Daten können zur Analyse an andere Libraries exportiert werden. Es gibt kein integriertes Tool, um Daten zu analysieren. Daten können aus Pyrosm exportiert und in OSMnx weiter verarbeitet werden. [85] [88]
Dokumentation	Die Dokumentation ist nur in Englisch verfügbar und etwas übersichtlicher als die von OSMnx. [87]

9.11. Kriterien

Die Kriterien basieren auf einem eigens erzeugten Excel, welches wir für die Berechnung erstellt haben. Wir haben die einzelnen Punkte unterschiedlich gewichtet und die Punkte in diese Tabelle übernommen.

Engine	Lizenz	Konfiguration	Performanz	Fortbewegung	Routing-Arten	Ausgabeformat	Operation / API	Dokumentation	Total
OSRM	2	3	2	3	3	3	2	2	**35**
Graphhopper	2	2	1	8	3	3	2	2	**34**
Valhalla	3	0	0	3	1	3	2	1	**18**
ORS	1	2	1	8	3	3	2	2	**33**
pgRouting	1	4	0	3	0	3	2	2	**28**
Itinero	3	3	1	3	0	3	0	0	**24**
BRouter	3	0	0	3	3	0	0	0	**9**
Routino	2	3	0	7	2	1	1	2	**26**
OSMnx	3	1	2	3	0	2	2	0	**23**
Pyrosm	3	3	1	3	0	2	2	1	**27**
Gewichtung	1	3	3	1	1	2	2	1	

9.12. Schlussfolgerungen

Anhand der Evaluation, der verschiedene Kriterien, multipliziert mit der Gewichtung ergibt sich ein eindeutiges Resultat.

- Platz 1: OSRM
- Platz 2: Graphhopper
- Platz 3: ORS

Wir entscheiden uns, anhand der Evaluation, OSRM für unsere Studienarbeit zu verwenden. Einige Punkte sprechen besonders stark für OSRM:

- OSRM lässt sich einfach konfigurieren und mittels Lua-Scripts können neue Profile erstellt werden.
- Die Contraction Hierarchies bieten eine hervorragende Performanz, auch für die Berechnung von langen Routen.
- Roundtrips können direkt berechnet werden.
- OSRM besitzt Matrix-Routing.
- GeoJSON wird als Ausgabeformat unterstützt.
- Die Dokumentation ist ausführlich und man erhält auch sonst sehr viele Informationen und Tutorials.

10. Implementation

10.1. Wichtige Klassen

Die wichtigsten Klassen wurden bereits in der Gesamtarchitektur des Systems entworfen. Sie werden in diesem Kapitel weiter erläutert.

10.1.1. POI Collection Service

Der POI Collection Service ist für das Zusammensammeln aller, für einen Rundtrip infrage kommenden, POIs zuständig. Er verwendet dazu Filter und Einschränkungen, um ein Set von POIs zu erhalten. Der POI Collection Service verwendet dazu die POI Gathering Strategy und POI Restriction Strategy.

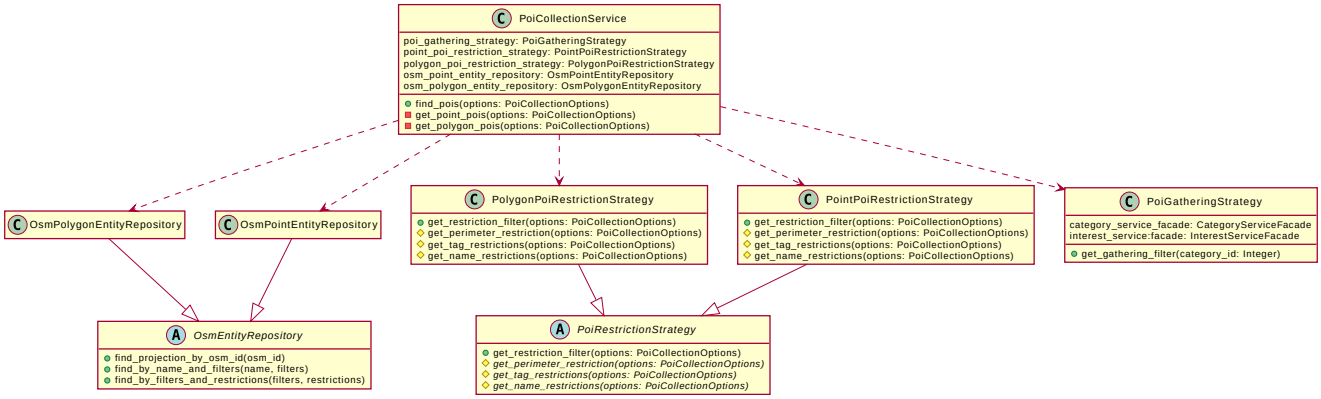


Abbildung 26. UML Klassendiagramm [89] POI Collection Service

Pseudo Code POI Collection Service

```
def find_pois(options) -> Dictionary:
    points = find_points(options)
    polygons = find_polygons(options)
    return transform_to_dictionary(points, polygons)

def find_points(options) -> List:
    gathering_filter = poi_gathering_strategy.get_filter(options.category_id)
    restriction_filter = point_restriction_filter.get_filter(options)
    return point_repository.find(gathering_filter, restriction_filter)

def find_polygons(options) -> List:
    gathering_filter = poi_gathering_strategy.get_filter(options.category_id)
    restriction_filter = polygon_restriction_filter.get_filter(options)
    return polygon_repository.find(gathering_filter, restriction_filter)
```

10.1.2. POI Gathering Strategy

Die POI Gathering Strategy erstellt den Filter, mit welchem die OpenStreetMap-Daten nach passenden POIs durchsucht werden. Dieser Filter besteht aus Key-Value-Paaren. Diese Datenstruktur nennt man in Python Dictionary. Der Key ist eine Zeichenkette und der Value ist eine Liste von Zeichenketten.

Die Keys repräsentieren die Spaltennamen der OpenStreetMap-Tabellen in der Datenbank. Die akzeptierten Werte in den jeweiligen Spalten werden in der Liste hinterlegt, welche den Value des Key-Value-Paares repräsentiert.

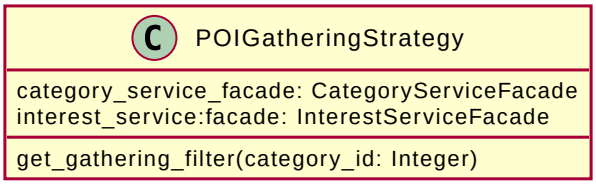


Abbildung 27. UML Klassendiagramm [89] POI Gathering Strategy

Der Filter kann für Punkte und Polygone angewendet werden und braucht keine zusätzliche Typisierung.

Pseudo Code POI Gathering Strategy

```
def get_filter(category_id) -> Dictionary:
    category = get_category(category_id)

    if not category:
        return Dictionary.empty()

    return interest_service_facade.find_filter_by_category(category)
```

Der Name dieser Klasse beinhaltet den Namen des Strategie-Entwurfsmusters. Aktuell wird aufgrund der nicht benötigten Typisierung keine algorithmische Kapselung des Typs vorgenommen.

10.1.3. POI Restriction Strategy

Die POI Restriction Strategy erstellt den Filter, mit welchem die OpenStreetMap-Daten basierend auf weiteren Kriterien eingeschränkt werden. Die verwendete Datenstruktur hierfür ist eine Liste von Kriterien, welche bei der Ausführung der Datenbankabfrage ausgewertet werden.

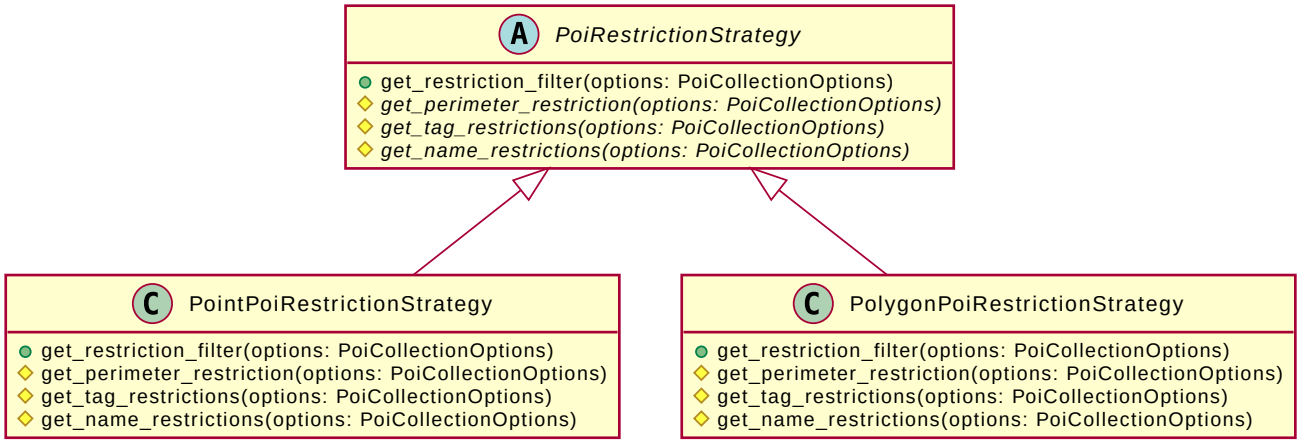


Abbildung 28. UML Klassendiagramm [89] POI Restriction Strategy

Die abgeleiteten Klassen für Punkte und Polygone kapseln die algorithmische Logik aufgrund der

unterschiedlichen Typisierung. Der Ablauf, um die Einschränkungen zu finden, verläuft sich für beide Typen gleich. Deshalb wurde hier eine Kombination von Strategy und Template Method Pattern gewählt. Im Pseudo Code wird dieses Template genauer beschrieben.

Pseudo Code POI Restriction Strategy

```
def get_filter(options) -> List:
    restrictions = List.empty()
    restrictions.extend(get_perimeter_restriction(options))
    restrictions.extend(get_tag_restrictions(options))
    restrictions.extend(get_name_restrictions(options))
    return restrictions
```

10.1.4. Roundtrip Routing Service

Der Roundtrip Routing Service ist für die Erstellung eines Rundtrips zuständig. Er ruft sämtliche anderen Services auf, um den gewünschten Rundtrip des Benutzers zu erstellen.

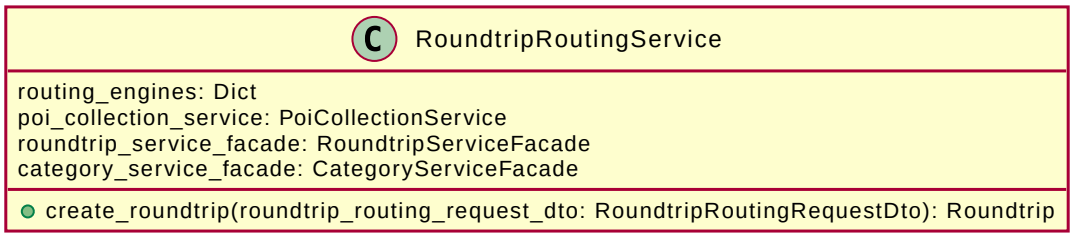


Abbildung 29. UML Klassendiagramm [89] Roundtrip Routing Service

Bei diesem Diagramm wurde auf die privaten Methoden zugunsten der Übersichtlichkeit verzichtet.

Pseudo Code Roundtrip Routing Service

```
def create_roundtrip(request) -> Roundtrip:
    category = category_service_facade.find(request.category_id)
    roundtrip = create_roundtrip(category, request)

    pois = poi_collection_service.find_pois(request.options)
    if len(pois) < 2:
        raise Error("Not enough POIs to create Roundtrip.")

    engine_response = call_routing_engine(request, pois)
    return update_roundtrip(roundtrip, engine_response)

def call_routing_engine(options) -> EngineResponse:
    engine = engines.get(options.engine_name)
    engine_request = transform(options)
    return engine.route(engine_request)
```

10.1.5. Routing Engine Adapter

Um die Abhängigkeit zu einer spezifischen Routing Engine zu lösen, wird ein Adapter verwendet. Dieser Adapter gibt einen Vertrag vor, was einer konkreten Implementation übergeben wird und was der erwartete Rückgabetyp ist.

Für den Namen dieses Interfaces wurde bewusst *Adapter* verwendet. Wir wollten dadurch vermeiden, dass es zu einer Verwechslung kommt, weil wir keine eigene Implementation einer Routing Engine entwickeln. Aus der Sicht von City Trip Planner handelt es sich um ein Interface und aus der Sicht der Routing Engine um einen Adapter.

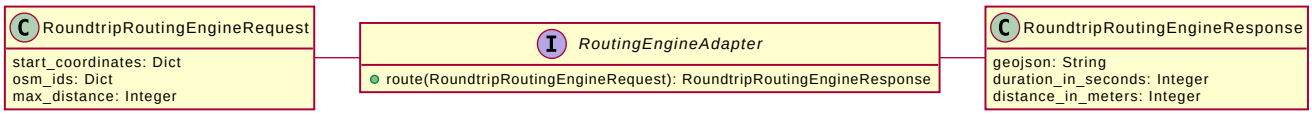


Abbildung 30. UML Klassendiagramm [89] Routing Engine Adapter

10.1.6. (Adaptierte) OSRM Engine

Der oben beschriebene Vertrag wird von dieser Klasse implementiert. Sie verwendet verschiedene Strategien zur Berechnung der Route. Diese Strategien können konfigurativ durch den Betreiber angepasst werden. Es gibt folgende Strategien:

- Short Distance Osrm Roundtrip Routing Strategy
- Binary Distance Search Roundtrip Routing Strategy
- Long Distance Osrm Roundtrip Routing Strategy

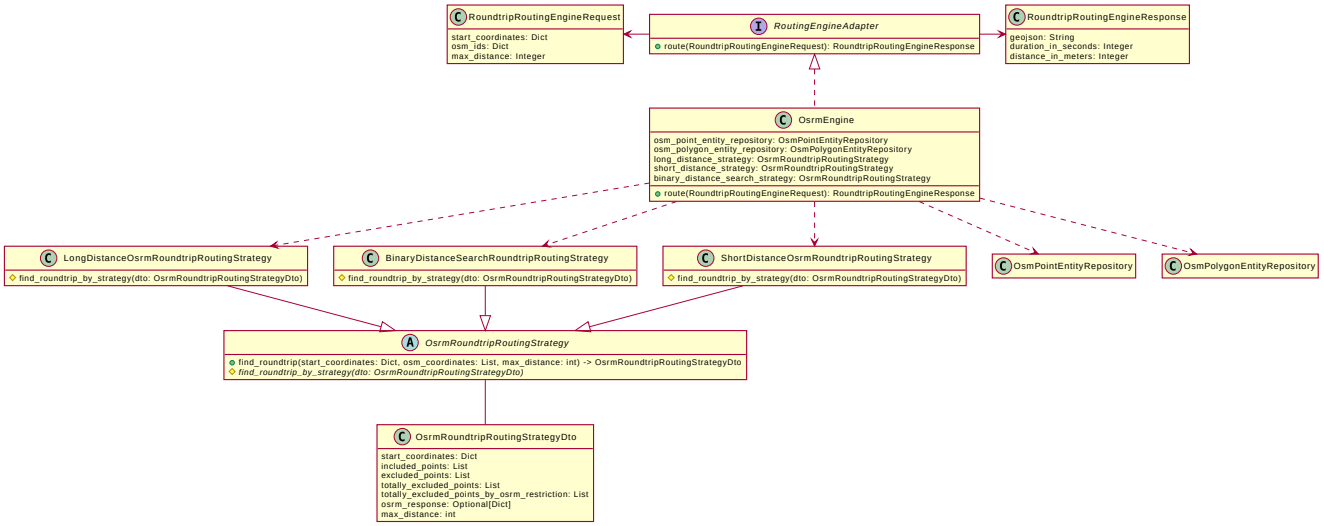


Abbildung 31. UML Klassendiagramm [89] OSRM Engine

Bei diesem Diagramm wurde auf die privaten Methoden zugunsten der Übersichtlichkeit verzichtet.

Pseudo Code OSRM Engine

```
def route(request) -> response:
    osm_coordinates = get_coordinates(request)
    strategy = get_strategy(len(osm_coordinates), request.max_distance)
    strategy_dto = strategy.find_roundtrip(osm_coordinates, request)

    if not strategy_dto.response:
        raise RoundtripCreationError()

    return transform_to_response(strategy_dto)
```

10.1.7. Short Distance Osrm Roundtrip Routing Strategy

Diese Strategie wird verwendet, um Routen mit kurzen Distanzen oder wenigen POIs zu finden. Die Schwellwerte können hierfür konfigurativ angepasst werden.

Die POIs sind nach der Distanz zum Startpunkt aufsteigend sortiert und werden als Liste übergeben. Der Algorithmus nimmt Punkt für Punkt vom Anfang der Liste und lässt die Routing Engine die Route berechnen bis der Schwellwert überschritten ist. Sobald dies geschieht, wird das letzte gültige Resultat verwendet. Dieser Algorithmus verfolgt den Ansatz *Brute Force*.

Um den Vorgang zu beschleunigen, werden simultan mehrere Requests ans OSRM-Backend gesendet. Dies funktioniert sehr gut für kurze Distanzen oder wenige Punkte.

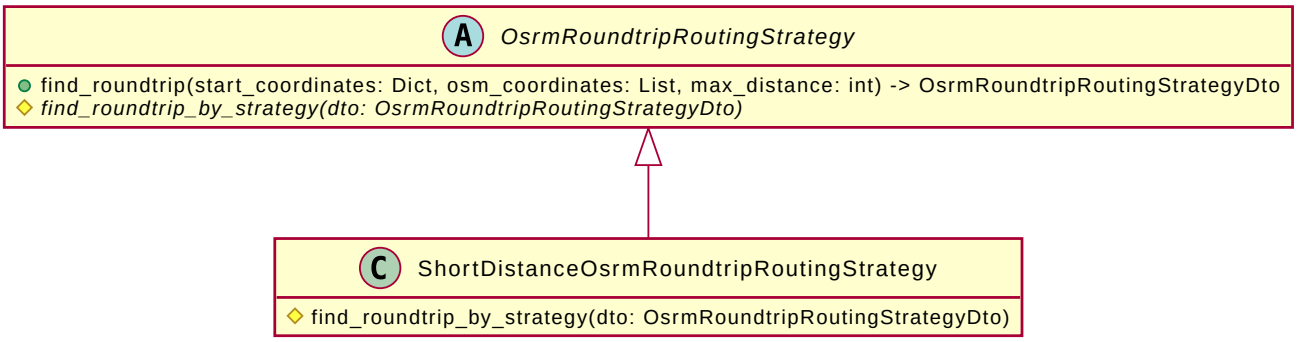


Abbildung 32. UML Klassendiagramm [89] Short Distance Osrm Roundtrip Routing Strategy

Pseudo Code Short Distance Osrm Roundtrip Routing Strategy

```
def find_roundtrip_by_strategy(dto) -> dto:
    last_result = undefined
    number_of_points = 1
    concurrent_requests = get_concurrent_requests()

    while True:
        poi_set = []
        for offset in range(0, concurrent_requests)
            index = number_of_points + offset
            if index > len(dto.included_points)
                break
            poi_set.append(dto.included_points[:index])

        if not poi_set:
            return update_dto(dto, last_result)

        intermediate_results = get_results(dto, poi_sets)

        foreach result in intermediate_result:
            distance = get_distance(result)
            if dto.max_distance < distance
                return update_dto(dto, last_result)
            else
                last_result = result

        number_of_points += len(sets)
```

10.1.8. Binary Distance Search Roundtrip Routing Strategy

Diese Strategy sucht mittels binärer Suche das Resultat, welches der maximalen Distanz am nächsten kommt. Sie beginnt mit der Hälfte der Punkte in der sortierten Liste und fährt dann basierend auf der Differenz fort.

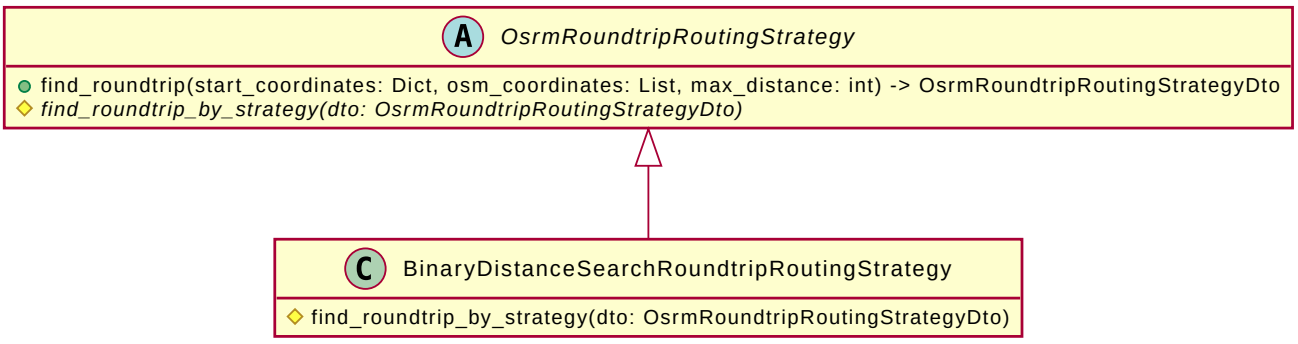


Abbildung 33. UML Klassendiagramm [89] Binary Distance Search Roundtrip Routing Strategy

Pseudo Code Binary Distance Search Roundtrip Routing Strategy

```
def find_roundtrip_by_strategy(dto) -> dto:
    iteration = 1
    size = len(dto.included_points)
    start_index = int(size * binary_fraction(iteration))
    last_index = 0
    current_index = start_index

    while True:
        points = dto.included_points[:current_index]
        result = get_result(dto, points)
        distance = get(distance)

        iteration += 1
        delta = binary_fraction(iteration) * size
        last_index = current_index
        if distance < dto.max_distance:
            current_index = int(current_index + delta)
        elif distance > dto.max_distance:
            current_index = int(current_index - delta)
        else:
            return update_dto(dto, dto.included_points[:last_index])

        if last_index == current_index:
            return update_dto(dto, dto.included_points[:last_index])
```

10.1.9. Long Distance Osrm Roundtrip Routing Strategy

Die letzte Variante der Strategien wird bei grossen Distanzen oder vielen Punkten angewendet. Bei dieser Strategie wird zuerst die Distanz eines Rundtrips berechnet, in der alle Punkte vorhanden sind. Die errechnete Distanz wird mit der Maximaldistanz ins Verhältnis gesetzt. Dieses Verhältnis wird als dynamischer Faktor zwischengespeichert. Nebst diesem dynamischen Faktor gibt es auch noch einen statischen Faktor. Der statische Faktor stammt aus der Konfiguration und gibt die Schrumpfrate vor. Der grössere Wert aus Distanzverhältnis und konfiguriertem Schrumpffaktor wird für die nächste Iteration mit der Anzahl POIs multipliziert. Dies ermöglicht eine Annäherung an die Maximaldistanz. Sobald ein Resultat gefunden wurde, welches innerhalb der vorgegebenen Maximaldistanz liegt, terminiert der Algorithmus.

Die Abfragen werden ebenfalls simultan an das Backend von OSRM gesendet. So können während der gleichen Wartedauer mehrere Resultate berechnet werden.

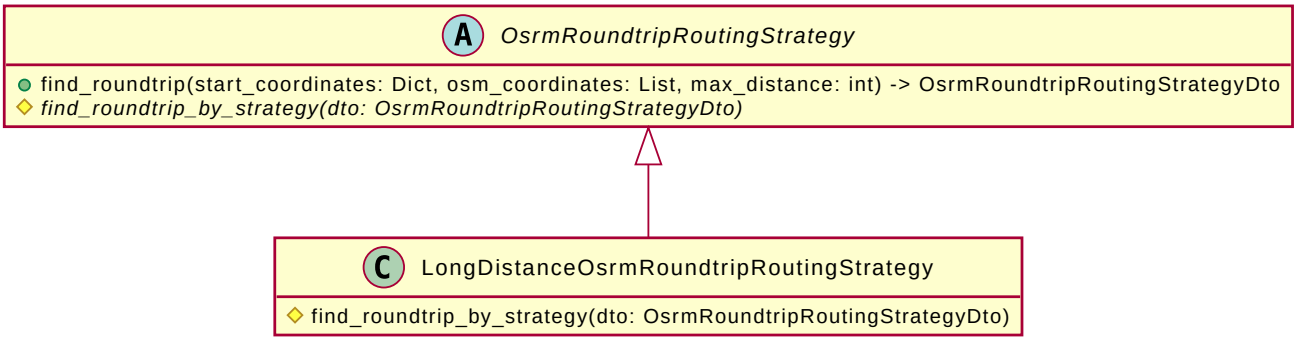


Abbildung 34. UML Klassendiagramm [89] Long Distance Osrm Roundtrip Routing Strategy

Pseudo Code Long Distance Osrm Roundtrip Routing Strategy

```
def find_roundtrip_by_strategy(dto) -> dto:
    iteration = 0
    distance = integer.max_size
    last_element = len(dto.included_points) -1
    shrink_factor = get_shrink_factor()
    concurrent_requests = get_concurrent_requests()

    while True:
        max_index = last_element
        distance_factor = dto.max_distance / distance
        factor = max(shrink_factor, distance_factor)

        sets = []
        for i in range(0, concurrent_requests)
            index = int(factor ** i) * max_index)
            if index < 1:
                break
            sets.append(dto.included_points[:index])
            last_index = index

        intermediate_results = get_results(dto, sets)

        for index, result in enumerate(intermediate_results):
            distance = get_distance(result)
            if distance < dto.max_distance:
                return update_dto(dto, result)

        if last_index < 1:
            return empty_result(dto)

        iteration += 1
```

10.2. Automatische Testverfahren

Für die Qualitäts- und Integritätsprüfung der Software wurden vier Stufen definiert.

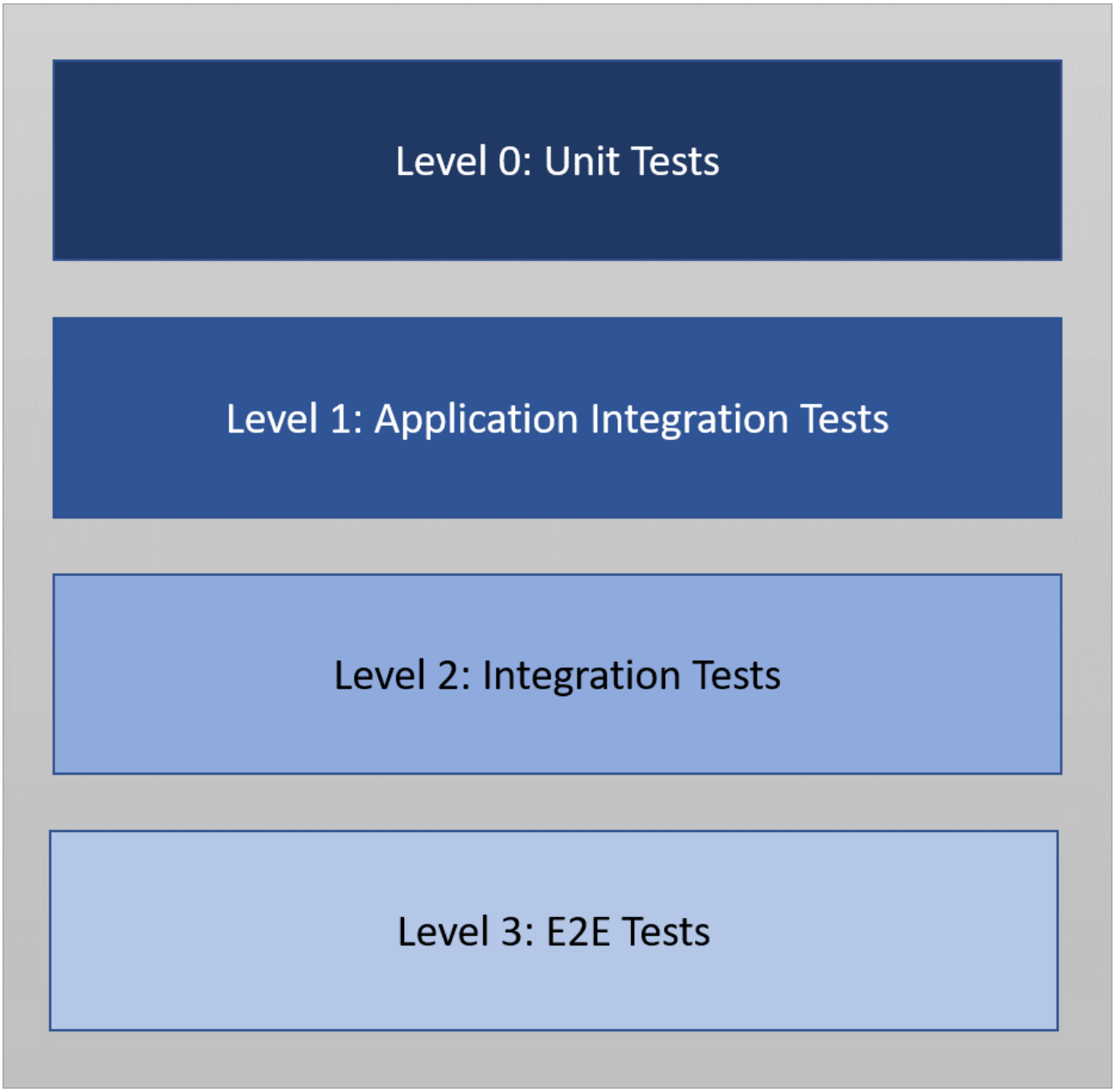


Abbildung 35. Stufen zur Prüfung von Qualität und Integrität der Software

In folgenden Abschnitten werden die Zuständigkeiten und Verantwortungen weiter erläutert.

10.2.1. Unit Tests

Mit Unit Tests werden die einzelnen Methoden der Klassen auf ihre Richtigkeit geprüft. Sie dienen in erster Linie für Regressionstests und sind besonders nützlich für Änderungen des Source Codes. Es ist sinnvoll, sie nach jedem Push eines Branch auszuführen. Die Pipeline soll so gebaut werden, dass kein Merge Request in den Master gefügt werden kann, wenn Tests fehlschlagen.

10.2.2. Application Integration Tests

Diese Tests stehen für die applikationsinternen Integrationstests. Sie überprüfen das korrekte Verhalten von mindestens zwei Komponenten der Applikation. Einen besonderen Mehrwert bieten diese Tests dann, wenn die Komponenten in verschiedenen Layers oder Schalen der Architektur (vgl. Referenzarchitektur Backend-Applikation) liegen. Diese Tests werden, wie die Unit Tests, bei jedem Push eines Branch und bei Merge Requests ausgeführt. Sie entsprechen den Smoke Tests.

10.2.3. Integration Tests

Die Integration Tests bilden die erweiterte Stufe der applikationsinternen Tests. Die Ausführung dieser Tests erfordert allerdings eine eigene Testumgebung. Das Ziel dieser Tests ist es nämlich, die Integration von realen Komponenten der Gesamtarchitektur (vgl. Architekturdiagramm Gesamtsystem) zu überprüfen und sicherzustellen. Diese Tests sollen mindestens wöchentlich, besser nächtlich ausgeführt werden.

10.2.4. End-to-End Tests

Die End-to-End Tests sind Testfälle, welche das komplette System über alle Komponenten der Gesamtarchitektur (vgl. Architekturdiagramm Gesamtsystem) prüfen. Sie dienen zur Überprüfung der nicht-funktionalen Anforderungen, wie beispielsweise der Performanz, oder zur Überprüfung der Integration aller Architekturkomponenten. Zusätzlich können ungewollte Änderungen an der Benutzeroberfläche festgestellt werden. Die End-to-End Tests entsprechen den Systemtests.

10.2.5. Automation Scope der Studienarbeit

Wir haben uns in Anbetracht der zur Verfügung stehenden Zeit dazu entschieden, auf die Automation für Integration Tests und End-to-End Tests zu verzichten. Dies sind sicherlich wichtige Verfahren der Qualitätssicherung, jedoch sind sie zu schwergewichtig und zu aufwändig für das Entwickeln eines Minimal Viable Product. Sie sollten aber bei der Produktentwicklung unbedingt eingesetzt werden, weshalb wir sie konzeptionell eingeführt haben.

Unverzichtbar sind allerdings Unit und Integration Tests, um Fehler bei der Entwicklung möglichst früh festzustellen. Wir legen ein besonderes Augenmerk auf die Application Integration Tests, damit wir automatisiert feststellen können, ob nach Änderungen im Backend das API Module im Frontend aktualisiert werden muss. Dies ermöglicht auch unabhängiges Arbeiten im Team.

11. Resultate und Weiterentwicklungen

11.1. Resultate

Das Resultat unserer Arbeit ist ein funktionierender Prototyp. Damit können Rundtrips mit kategorisierten Points of Interest, welche auf OpenStreetMap-Daten basieren, berechnet werden. Benutzer:innen können über eine Webseite den gewünschten Startpunkt, eine Kategorie, den Umkreis und die maximale Laufdistanz eingeben.

Anhand der genannten Eingaben wird in nützlicher Frist ein passender Rundtrip berechnet und dargestellt. Dies funktioniert für die gesamte Schweiz. Allerdings gibt es Regionen, in denen die Daten von OpenStreetMap nicht gleich gut erfasst sind wie in den Städten. Trotzdem ist es auch möglich, in ländlichen Regionen Rundtrips zu erstellen. Benutzer:innen müssen dafür lediglich die Eingaben, wie beispielsweise Suchradius, erhöhen. Wir haben einen Standard-Wert gewählt, welcher sich für städtisch-urbane Regionen eignet.

Jeder Rundtrip wird mit einem UUID abgespeichert und kann per QR-Code vom Desktop auf ein Mobilgerät übertragen werden. Der gleiche Rundtrip inklusive Parametern erscheint dann auch auf dem Mobilgerät. Zusätzlich können für alle Points of Interest Informationen und Bilder abgerufen werden. Man kann dafür die Details des gesamten Rundtrips anschauen oder auf einzelne POIs klicken.

Unsere Kontaktperson von Schweiz Tourismus, Markus Dittli, hat sich in einer Demonstration sehr positiv über das Minimal Viable Product geäussert. Er würde die Arbeit vorerst gerne im Auge behalten und allfällige Weiterentwicklungen begleiten.

11.2. Weiterentwicklungen

Die wichtigsten Weiterentwicklungen aus dem technischen Bericht sind nachfolgend konzeptionell erfasst. Diese dienen der Orientierung und als Anstoss für eine zukünftige Weiterentwicklung des Produktes.

11.2.1. Green-Routing

Das momentane Routing ist nicht schlecht, führt aber gerade bei längeren Strecken und ausserhalb von Städten gerne den Strassen entlang. Das soll korrigiert werden. Die Routen sollen explizit entlang von schönen und verkehrsarmen Wegen verlaufen. Diese Problemstellung war für unsere Studienarbeit zu komplex.

Vorgehen

In einer zukünftigen Arbeit kann für OSRM ein zusätzliches Routing Profil als Lua-Script erstellt werden. Als Hilfestellung kann das bereits existierende Foot-Script [90] verwendet werden. Die eigentliche Aufgabe ist es, eigene Restrictions und Gewichtungen zu definieren.

Das existierende Foot-Script berechnet Fussgänger-Routen auch über den Seeweg, nimmt also eine Schifffahrtslinie um den nächsten POI zu erreichen. Dies soll ausgeschlossen werden.

Bei erfolgreichem Abschluss kann das Script auch auf GitHub beigetragen werden.

11.2.2. Hindernisfreie Routen

Eine der Anforderung war es, Rundtrips auch hindernisfrei, für Menschen im Rollstuhl oder für Familien mit Kinderwagen, berechnen zu lassen. Die gewählte Routing Engine ist nicht in der Lage, solche Routen zu erstellen. Wo die Information, dass ein Point of Interest hindernisfrei ist, vorhanden ist, wird dies in den Details zum Roundtrip angezeigt.

Vorgehen

Unsere Applikation wurde so entwickelt, dass sie verschiedene Routing Engines unterstützt. Um diese Funktion anzubieten, kann eine passende Routing Engine an das System angeschlossen werden. Graphhopper [38] oder Routino [78] unterstützen diese Art von Routen.

11.2.3. Priorisierung von Points of Interest

Die Priorisierung von Points of Interest ist ein komplexes Thema. Es gibt viele zu berücksichtigende Faktoren, wie beispielsweise die Metriken anhand denen die Priorisierung vorgenommen wird. Ausserdem spielt es eine Rolle, ob innerhalb einer Kategorie oder kategorieübergreifend priorisiert werden muss.

OSRM hat die Einschränkung, dass Roundtrips mit nur maximal 100 Punkten erstellt werden können. Bei einer Abfrage, welche im Umkreis des Startpunkts mehr als 100 Punkte hat, werden nur die 100 Punkte berücksichtigt, welche am nächsten beim Startpunkt liegen.

Vorgehen

Das Wikidata QRank Projekt [12] bietet die Möglichkeit Entitäten, welche einen Wikidata-Eintrag besitzen, zu priorisieren. Man könnte dazu ein entsprechendes Datenbankfeld einführen und die Priorität darin pro Point of Interest führen. Bei der Abfrage müsste dann lediglich noch eine Sortierung vorgenommen werden und die Applikation käme ohne fundamentale Änderungen aus. Allerdings besitzen nicht alle, aus den OpenStreetMap-Daten stammenden Einträge, einen Wikidata-Eintrag. Für die Priorisierung dieser Points of Interest müsste ein anderes Verfahren bestimmt werden, welches bestenfalls mit QRank kompatibel ist.

11.2.4. Einschränkung von Anzahl gleicher Points of Interest

Bei gewissen POIs wie beispielsweise öffentlichen Toiletten oder Geldautomaten ist es nicht gewünscht, jede Toilette oder jeden Automaten im Umkreis abzuschreiten. Hier wäre es sinnvoll, dass solche Stationen nur einmal auf einem Rundtrip vorkommen.

Ein weiteres Beispiel sind Fahrradverleihe. Es wäre durchaus vorstellbar eine gewisse Strecke des Rundtrips per Fahrrad zurückzulegen. Hierbei müssten allerdings mindestens zwei Stationen des gleichen Fahrradverleihers auf der Route liegen.

Vorgehen

Um diese Erweiterung umzusetzen, braucht es die Möglichkeit von zusätzlicher Benutzerinteraktion. Wir stellen uns vor, dass nach der Darstellung Punkte ergänzt oder entfernt werden können, um diese Bedürfnisse abzudecken. Im Dialogfenster könnten dann auch Restaurants oder Einkaufsläden für Picknick angezeigt werden.

11.2.5. Alternativ-Routen

Zurzeit werden alternative Routen einfach berechnet, in dem wir unseren Startpunkt verschieben. Die Geschwindigkeit der Applikation lässt dies zu und es wurde von unserem Industriepartner akzeptiert. Eine Verbesserung wäre allerdings wünschenswert.

Vorgehen

Wir haben uns Gedanken gemacht, wie wir mit der bestehenden OSRM-Engine alternative Routen berechnen können. Man könnte Routen in verschiedene Himmelsrichtungen berechnen. Der Nachteil liegt klar bei der Performanz, da mehrere Routen berechnet werden müssten. Alternative Routen könnten mit verschiedenen Farben gemeinsam auf der Karte angezeigt werden. Der Nutzer kann sich eine selektieren und die Details dieser spezifischen Route anschauen.

Der Motorrad-Routenplaner kurviger.de [91] bietet das Erstellen von Motorradtouren in benutzerdefinierte Himmelsrichtungen an, jedoch keine Alternativen.

11.2.6. PrimeNG Accordion-Element

Das Accordion-Element hat einen bekannten Fehler, der auch in einem Issue [92] beschrieben wird. Das Issue ist geschlossen, allerdings scheint der Fehler weiterhin zu bestehen.

Vorgehen

Wird kein entsprechender Fix zum bekannten Problem geliefert, müsste für die Anzeige der Point of Interest-Details eine andere Komponente verwendet werden. Somit kann das Problem umgangen werden. Es ist nur die Anzeige betroffen, weshalb keine Neuimplementation der Anzeigelogik im Frontend notwendig ist.

12. Projektmanagement

12.1. Vorgehensmodell

Wir verwenden für die Studienarbeit das Vorgehensmodell SCRUM+. Dies wurde uns in den letzten Semestern mehrfach vermittelt.

SCRUM+ ist eine Mischung aus RUP und Scrum. Wir nutzen die Phasen aus RUP für die langfristige Planung und verwenden SCRUM in 1-wöchigen Sprints für die kurzfristige Planung.

12.2. Involvierte Personen

Person	**Rolle**	**Verantwortlichkeit**
Lukas Leuenberger	Student, Entwickler	Coding, Dokumentation, Kommunikation
Jan Ruch	Student, Entwickler	Coding, Dokumentation, Kommunikation
Prof. Stefan Keller	Betreuer FH OST	Betreuung, fachliche Unterstützung, Bewertung
Nicola Jordan	Wissenschaftlicher Mitarbeiter IFS FH OST	Betreuung, technische Unterstützung
Markus Dittli	Industriepartner Schweiz Tourismus	Anforderungen kommunizieren

12.3. Aufwandsschätzung

Das Hochschulreglement der Ostschweizer Fachhochschule (OST) vergibt für die Studienarbeit 8 ECTS Punkte pro Student. 1 ECTS Punkt wird mit 30 Arbeitsstunden gleichgesetzt. Somit entsteht für die vorliegende Studienarbeit ein Richtwert von 480 Arbeitsstunden, die zur Verfügung stehen. Das Projekt ist so ausgelegt, um die priorisierten Anforderungen innerhalb dieses Richtwerts zu erfüllen.

Wir haben nachfolgend die grössten Arbeitsschritte grob geschätzt. Die genaueren Schätzungen werden wir weitestgehend mit SCRUM in YouTrack durchführen. Ein Report ist unten angefügt.

12.4. Meilensteine

Phase	Meilenstein	Aufwand	Fälligkeit
Knowledge Gathering	-	4 Wochen	Sonntag, 17.10.2021
Inception	-	2 Wochen	Sonntag, 31.10.2021
Elaboration	M1: Architekturprototyp	1 Woche	Sonntag, 07.11.2021
Construction	M2: Evaluation Routing Engine	1 Woche	Sonntag, 14.11.2021
Construction	M3: Anbindung Routing Engine	2 Wochen	Sonntag, 28.11.2021
Construction	M4: Implementation Frontend	2 Wochen	Sonntag, 12.12.2021
Transition	M5: Abgabe SA	2 Wochen	Freitag, 24.12.2021

12.5. Prototypen und Releases

Die gesamte Arbeit ist ein Proof of Concept und soll ein Minimal Viable Product hervorbringen. Das Produkt wird unseren Betreuern und unserem Industriepartner präsentiert. Ein offizieller Release und das Deployment auf einem Server der Hochschule ist nicht geplant.

12.6. Risiken

Das Risikomanagement ist keine einmalige Arbeit. Während der Projektarbeit können neue Risiken auftauchen oder sich bekannte Risiken verändern.

12.6.1. R1: Unterschätzte Komplexität

Beschreibung	Die Komplexität einzelner Anforderungen wurde unterschätzt und die Arbeit wird deshalb nicht, oder nur fehlerhaft implementiert.
Eintrittswahr-scheinlichkeit	60%
Gewichteter Schaden	8
Vorbeugung	Modularisierung
Verhalten beim Eintreffen	Analysieren, Vereinfachen oder Weglassen der Anforderung und eine konzeptionelle Lösung erarbeiten

12.6.2. R2: Fehlerhaftes Exception Handling

Beschreibung	Exceptions werden nicht korrekt abgearbeitet, oder das Mapping auf HTTP Statuscodes ist fehlerhaft.
Eintrittswahr-scheinlichkeit	80%
Gewichteter Schaden	4
Vorbeugung	Testing, Modularisierung
Verhalten beim Eintreffen	Exception Handling überarbeiten, Registration der korrekten Statuscodes

12.6.3. R3: Neuer Technologie Stack

Beschreibung	Die neuen Technologien benötigen zu viel Einarbeitungszeit, Anforderungen können nicht umgesetzt werden.
Eintrittswahr-scheinlichkeit	40%
Gewichteter Schaden	6
Vorbeugung	Genaue Analyse, Prototypen, frühzeitiges Eingreifen der Betreuer
Verhalten beim Eintreffen	Besprechung mit den Betreuern, Prioritäten neu definieren

12.6.4. R4: Mangelhafte Kommunikation

Beschreibung	Die Kommunikation mit den Betreuern oder dem Industriepartner scheitert.
Eintrittswahr-scheinlichkeit	5%
Gewichteter Schaden	10
Vorbeugung	Frühzeitiges Eingreifen durch die Studenten
Verhalten beim Eintreffen	Besprechung mit den Betreuern, Notfalls ein Gespräch mit dem Studiengangleiter

12.6.5. R5: Personeller Ausfall

Beschreibung	Eine Person fällt aus gesundheitlichen Gründen aus, die Arbeit verzögert oder schmälert sich.
Eintrittswahr-scheinlichkeit	10%
Gewichteter Schaden	8
Vorbeugung	Gute Kommunikation im Voraus, das Risiko lässt sich kaum verhindern (höhere Gewalt)
Verhalten beim Eintreffen	Besprechung mit den Betreuern, Arbeitspakete und Anforderungen neu definieren

12.6.6. Neubewertung nach Meilenstein 1

Nach dem Erstellen des Architekturprototyps kann das Risiko R3 zumindest für das Backend ausgeschlossen werden.

Das Risiko R1 bleibt weiterhin bestehen und könnte besonders in der Construction Phase ein Problem werden. Das Risiko R4 war bislang kein Problem, die Kommunikation hat stets funktioniert.

Die übrigen Risiken werden als gering eingeschätzt.

12.6.7. Neubewertung nach Meilenstein 4

Das Risiko R3 ist mit weiteren Frontend-Technologien erneut aufgetreten. Es gab jedoch keine komplexeren Probleme.

Die programmatische Arbeit ist abgeschlossen, somit ist auch das Risiko R1 und R2 ausgeschlossen.

13. Projektmonitoring

Zu Beginn des Projektes wurde entschieden, in den ersten Wochen kein präzises Projektmanagement durchzuführen. Der administrative Mehraufwand, in dieser frühen Phase und mit nur 2 Personen, übersteigt den Nutzen von YouTrack. Der Koordinationsaufwand fällt gleichermassen an, wird aber bilateral in regelmässigen Meetings bewältigt.

In der sechsten Woche startet die Aufgaben- und Zeiterfassung in YouTrack. Die berechneten Stunden beziehen sich auf die letzten 9 Wochen der Arbeit. Der Arbeitsaufwand der ersten 5 Wochen befindet sich bei etwa 180 Stunden. Dies entspricht einem Arbeitsaufwand von rund 18 Stunden pro Woche und pro Person.

13.1. Effektiver Aufwand

		Sep.			Okt.				Nov.					Dez.		
Benutzer, gruppiert nach Arbeitstyp		18-19	20-26	27-03	04-10	11-17	18-24	25-31	01-07	08-14	15-21	22-28	29-05	06-12	13-19	20-26
Gesamtzeit	**482h 20m**	**00m**	**00m**	**00m**	**00m**	**00m**	**00m**	**26h 00m**	**38h 30m**	**31h 20m**	**61h 30m**	**70h 00m**	**58h 00m**	**39h 30m**	**112h 45m**	**44h 45m**
Dokumentation	**187h 35m**	00m	00m	00m	00m	00m	00m	18h 00m	8h 00m	3h 20m	4h 00m	30m	1h 00m	26h 15m	85h 45m	40h 45m
Jan Ruch	103h 00m							18h 00m	8h 00m	45m				13h 15m	37h 15m	25h 45m
Lukas Leuenberger	84h 35m									2h 35m	4h 00m	30m	1h 00m	13h 00m	48h 30m	15h 00m
Entwicklung	**259h 45m**	00m	00m	00m	00m	00m	00m	8h 00m	30h 30m	23h 00m	54h 30m	65h 30m	48h 00m	5h 15m	21h 00m	4h 00m
Jan Ruch	73h 30m									13h 00m	32h 15m	17h 45m	9h 30m		1h 00m	
Lukas Leuenberger	186h 15m							8h 00m	30h 30m	10h 00m	22h 15m	47h 45m	38h 30m	5h 15m	20h 00m	4h 00m
Projektmanagement	**35h 00m**	00m	00m	00m	00m	00m	00m	00m	00m	5h 00m	3h 00m	4h 00m	9h 00m	8h 00m	6h 00m	00m
Jan Ruch	17h 00m									2h 30m	1h 30m	2h 00m	4h 00m	4h 00m	3h 00m	
Lukas Leuenberger	18h 00m									2h 30m	1h 30m	2h 00m	5h 00m	4h 00m	3h 00m	

Abbildung 36. Zeit pro Woche nach Arbeitstyp und Benutzer (Stand: 22.12.2021)

Die Arbeitszeit aus den ersten 5 Wochen und die erfassten Aufwände in YouTrack ergeben eine Gesamtarbeitszeit von ca. 660 Stunden.

Die Differenz von rund 180 Stunden, zum Richtwert von 480 Arbeitsstunden, erklären wir uns durch das Lernen von neuen Technologien (Python, FastAPI, Alembic, SQLAlchemy, etc.). Wir haben einen grossen Anteil dieser Zeit dafür verwendet, uns an bestehenden Best-Practices zu orientieren. Für ein besseres Verständnis der Thematiken war das Einlesen in die Dokumentationen notwendig, was ebenfalls einiges an Zeit in Anspruch genommen hat.

Wir waren bereit mehr als das Minimum zu leisten, weil wir grosse Freude an diesem Projekt verspürten und viele Erfahrungen in einem neuen Bereich sammeln konnten. Wir sind mit dem Resultat überaus zufrieden, der zusätzliche Aufwand hat sich unserer Ansicht nach gelohnt.

13.2. Codestatistik

Die Codestatistik ist mit dem *Statistic*-Plugin ausgewertet. [93]

13.2.1. Backend

Extension	Count	Lines	Lines AVG	Lines CODE
gitignore *(GITIGNORE files)*	1x	207	207	101
md *(MD files)*	1x	105	105	62
py *(PY files)*	214x	5894	27	4715
sh *(SH files)*	3x	74	24	50
txt *(Text files)*	2x	114	57	57
yml *(YML files)*	4x	619	154	526

Abbildung 37. Anzahl Dateien und Codezeilen im Backend (Stand: 22.12.2021)

Die wichtigsten Dateien im Backend-Code sind die *.py* und die *.yml*. Der gesamte Python-Code umfasst 4715 Zeilen und ist über 214 Dateien verteilt. Die 4 YML-Dateien beinhalten in 526 Zeilen die komplette Konfiguration.

13.2.2. Frontend

Extension	Count	Lines	Lines AVG	Lines CODE
gitignore *(GITIGNORE files)*	1x	51	51	32
html *(HTML files)*	13x	274	21	247
md *(MD files)*	1x	81	81	45
scss *(SCSS files)*	13x	50	3	47
sh *(SH files)*	1x	46	46	27
svg *(SVG files)*	8x	196	24	196
ts *(TS files)*	75x	3755	50	2638
txt *(Text files)*	1x	31	31	22
yml *(YML files)*	1x	57	57	50

Abbildung 38. Anzahl Dateien und Codezeilen im Frontend (Stand: 22.12.2021)

Die wichtigsten Dateien im Frontend-Code sind die *.ts*, *.html* und die *.scss*. Der Code für die Darstellung der Webseite umfasst insgesamt 2932 Zeilen und ist über 101 Dateien verteilt.

13.3. Beschlussprotokolle

Die Beschlussprotokolle befinden sich im Journal. Für jedes Meeting werden die wichtigsten Beschlüsse notiert. Das Journal wird nur digital als PDF eingereicht.

14. Softwaredokumentation

14.1. Verwendeter Technologie Stack

- Für die Entwicklung der Backend-Applikation wird Python [94] verwendet.
  Als Framework wird FastAPI [95] eingesetzt.
- Die Datenhaltung wird mittels PostgreSQL [96] sichergestellt.
  PostgreSQL muss dafür mit entsprechenden Erweiterungen wie HSTORE [97] und PostGIS [98] ergänzt werden.
- Die benötigten Daten werden von OpenStreetMap [99] zur Verfügung gestellt.
- Die Bedienung der Applikation und Darstellung einer Route wird mittels Webapplikation sichergestellt. Sie wird mit Angular [100] implementiert.
- Als Routing Engine wird OSRM [101] eingesetzt.

Versionsübersicht:

Technologie	**Version**
Python	3.8+
FastAPI	0.70.0
PostgreSQL	13.3
PostGIS	3.1
Angular	13.1
OpenStreetMap-Daten für die Schweiz	Tagesaktuell von GeoFabrik: Download Switzerland [102]
OSRM Project	5.26.0

14.2. Installation und Setup

14.2.1. Installation

Die aktuellsten Installationsinstruktionen befinden sich im git-Repository Docker.

14.2.2. Setup

Um das Projekt weiterzuentwickeln, befindet sich in jedem Repository eine README-Datei mit den benötigten Befehlen.

- Backend
- Frontend
- Docker (virtuelle Infrastruktur)

Appendix A: Abgabeumfang

Die in der Aufgabenstellung geforderten Abgaben und die des Studiengangs werden folgendermassen abgegeben:

A.1. Archivierung

- Vom Studiengang geforderte Archivdatei als ZIP
- Eigenständigkeitserklärung als PDF
- Urheber- und Nutzungsrechte als PDF
- A0 Poster als PDF

A.2. E-Prints

- Plain-Text Abstract als TXT-Datei
- Bericht als PDF
- Einverständniserklärung zur Veröffentlichung des Berichts

A.3. Broschüre

- Abstract für Broschüre über absolvierte Studien- und Bachelorarbeiten

A.4. Code Repositories

- https://gitlab.ost.ch/lleuenberger/studienarbeit/frontend
- https://gitlab.ost.ch/lleuenberger/studienarbeit/backend
- https://gitlab.ost.ch/lleuenberger/studienarbeit/docker
- https://gitlab.ost.ch/lleuenberger/studienarbeit/documentation

A.5. Lauffähige Applikation

Die Webapplikation kann wie in https://gitlab.ost.ch/lleuenberger/studienarbeit/docker beschrieben installiert werden.

Aktuell haben nur Prof. Stefan Keller, Nicola Jordan, Jan Ruch und Lukas Leuenberger Zugriff auf die lauffähige Applikation. (Stand 23.12.2021)

A.6. Physische Version

Die vorliegende Studienarbeit wird als gebundene Version an Prof. Stefan Keller abgegeben.

Appendix B: Glossar und Abkürzungsverzeichnis

City Trip Planner

City Trip Planner ist der Name des Produkts, welches im Rahmen der vorliegenden Arbeit entwickelt wird.

Extract, Transform, Load

ETL beschreibt einen Prozess, bei dem Daten aus mehreren unterschiedlichen und verschieden strukturierten Quellen in ein Zielformat vereint und abgespeichert werden.

Geoinformationssystem

Ein Geoinformationssystem ist ein Informationssystem, indem räumliche Daten auf der Erde unterhalten werden.

Minimal Viable Product

Ein Minimal Viable Product steht für ein minimal existenzfähiges Produkt und repräsentiert die minimal funktionsfähige Iteration eines Produkts. Das Ziel eines MVP ist es, möglichst schnell Feedback zur Anwendung zu erhalten.

OpenStreetMap

OpenStreetMap ist ein Projekt, welches frei nutzbare Geodaten zur Verfügung stellt. Offizielle OpenStreetMap Webseite: www.openstreetmap.org [99]

Proof of Concept

An einem Proof of Concept wird die Durchführbarkeit belegt. Ein Proof of Concept ist ein Machbarkeitsnachweis.

Point of Interest

Ein Point of Interest (de: Ort von Interesse) stellt einen Punkt auf der Landkarte dar, welcher für den Betrachter von Bedeutung sein könnte.

PostgreSQL

PostgreSQL oder kurz Postgres ist ein objektrelationales Datenbanksystem. Offizielle PostgreSQL Webseite: www.postgresql.org [96]

Rundtrip

Der Begriff Rundtrip steht für einen Ausflug über diverse Punkte, welcher wieder am Ausgangspunkt endet.

Trip

Der Begriff Trip steht für einen Ausflug von einem Punkt A zu Punkt B.

Universally Unique Identifier

Ein UUID steht für eine Zahl, welche zur Identifikation eines Objekts verwendet wird.

Appendix C: Literatur- und Quellenverzeichnis

[1] Technischen Universität München. (2007). Ein Hybrid-Verfahren zur Bearbeitung Kombinatorischer Optimierungsprobleme. https://mediatum.ub.tum.de/doc/619515/539433.pdf S. 50-52

[2] Nathan Brixius, N. B. (2016, 15. Juni). Computing Optimal Road Trips Using Operations Research. Nathan Brixius. https://nathanbrixius.wordpress.com/2016/06/09/computing-optimal-road-trips-using-operations-research/

[3] Scala, F. (2015, March 26). POI Tour – Personalisierter Tourenplaner für Fussgänger - eprints. EPrints OST. Retrieved December 12, 2021, from https://eprints.ost.ch/id/eprint/408/

[4] Matter, J., & Suter, R. (2018, April 10). PlazaRoute: Fussgänger-Routing über offene Flächen im urbanen Raum - eprints. EPrints OST. Retrieved December 12, 2021, from https://eprints.ost.ch/id/eprint/625/

[5] Kielar, P. M., Biedermann, D. H., Kneidl, A., & Borrmann, A. (2016, December 11). A Unified Pedestrian Routing Model Combining Multiple Graph-Based Navi. SpringerLink. Retrieved December 12, 2021, from https://link.springer.com/chapter/10.1007/978-3-319-33482-0_31?error=cookies_not_supported&code=81c01d06-2c33-432b-9a50-8006b7a6d6f1

[6] Kammoun, S., Dramas, F., Oriolaand, B., & Jouffrais, C. (2010, December 17). Route selection algorithm for Blind pedestrian. IEEE Conference Publication | IEEE Xplore. Retrieved December 12, 2021, from https://ieeexplore.ieee.org/abstract/document/5669846

[7] Cohen, A., & Dalyot, S. (2020, July 8). SAGE Journals: Your gateway to world-class research journals. SAGE Journals. Retrieved December 12, 2021, from https://journals.sagepub.com/action/cookieAbsent

[8] Jacob, R., Zheng, J., Ciepłuch, B., Mooney, P., & Winstanley, A. C. (2009, December 7). Campus Guidance System for International Conferences Based on OpenStre. SpringerLink. Retrieved December 12, 2021, from https://link.springer.com/chapter/10.1007/978-3-642-10601-9_13?error=cookies_not_supported&code=28233a5d-9154-4bc4-a37f-dce13cb95cfb

[9] Novack, T., Wang, Z., & Zipf, A. (2018, November 6). A System for Generating Customized Pleasant Pedestrian Routes Based on OpenStreetMap Data. MDPI. Retrieved December 12, 2021, from https://www.mdpi.com/1424-8220/18/11/3794

[10] How Trail Router works - Trail Router. (2020, June 3). Trail Router. Retrieved December 21, 2021, from https://trailrouter.com/blog/how-trail-router-works/

[11] Scala, F. (2014, December 16). POI Tour – Personalisierter Tourenplaner für Fussgänger. EPrints OST. Retrieved December 12, 2021, from https://eprints.ost.ch/id/eprint/408/1/sa-poitour_fscala.pdf

[12] Brawer, S. (2021, February 23). GitHub - brawer/wikidata-qrank: A ranking signal for Wikidata entities. GitHub. Retrieved December 12, 2021, from https://github.com/brawer/wikidata-qrank

[13] AENOR present their Trust Platform: “Ensuring Software and Data Quality.” (n.d.). ISO 25000. Retrieved December 12, 2021, from https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

[14] What is Data Flow Diagram (DFD)? How to Draw DFD? (2012, January 27). Visual Paradigm. Retrieved December 19, 2021, from https://www.visual-paradigm.com/tutorials/data-flow-diagram-dfd.jsp

[15] PlantUML. (n.d.). Komponentendiagramm Syntax und Funktionen. PlantUML.com. Retrieved December 19, 2021, from https://plantuml.com/de/component-diagram

[16] Martin, R. C. (2012, August 13). Clean Coder Blog. The Clean Code Blog. Retrieved December 12, 2021, from https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

[17] Angular. (n.d.). Angular IO. Retrieved December 12, 2021, from https://angular.io/guide/lazy-

loading-ngmodules
[18] Home. (2021, October 18). OpenAPI Initiative. Retrieved December 15, 2021, from https://www.openapis.org/
[19] PlantUML. (n.d.-b). Sequenzdiagramm Syntax und Funktionen. PlantUML.com. Retrieved December 20, 2021, from https://plantuml.com/de/sequence-diagram
[20] Schweiz Tourismus. (n.d.). Schweiz Ferien & Reisen. Retrieved December 12, 2021, from https://www.myswitzerland.com/de-ch/
[21] Figma: the collaborative interface design tool. (n.d.). Figma. Retrieved December 12, 2021, from https://www.figma.com
[22] Carraway, K. (n.d.). Google Maps UI Kit. Figma. Retrieved December 12, 2021, from https://www.figma.com/community/file/775789656970237137
[23] OpenStreetMap. (n.d.). OpenStreetMap Routing with Open Source Routing Machine. Retrieved December 12
[24] Google. (n.d.). Google Maps Zurich. Google Maps Zurich. Retrieved December 12, 2021, from https://www.google.com/maps/place/Zürich/@47.3775499,8.4666755,12z/
[25] OpenRouteService. (n.d.). OpenRouteService route planner - directions, isochrones and places. Retrieved December 12, 2021, from https://maps.openrouteservice.org/#/directions/Burger%20King,Z%C3%BCrich,Schweiz/Alfred-Escher-Brunnen,Z%C3%BCrich,Schweiz/Usteristrasse,Z%C3%BCrich,Schweiz/Hiltl%20Sihlpost,Z%C3%BCrich,Schweiz/Burger%20King,Z%C3%BCrich,Schweiz/data/55,130,32,198,15,97,4,224,38,9,96,59,2,24,5,192,166,6,113,0,184,64,14,0,232,5,96,25,128,78,124,0,96,5,159,1,24,3,96,9,134,198,173,38,128,104,104,29,144,210,189,242,54,37,94,189,90,140,43,224,13,196,76,185,114,53,200,142,38,89,190,10,61,250,13,239,89,149,94,148,247,209,172,206,73,82,249,122,151,174,84,179,70,52,170,29,237,160,111,98,237,154,116,57,172,133,153,23,153,149,51,16,177,50,179,61,59,174,163,35,61,148,117,40,141,32,69,60,84,84,155,47,51,185,12,80,156,91,51,177,51,57,51,8,23,8,4,0,3,170,60,4,34,54,14,40,0,23,148,0,45,174,61,62,5,85,116,4,0,25,188,0,13,186,46,8,63,84,42,0,45,0,59,178,16,192,53,146,0,57,185,72,47,122,63,122,52,58,34,24,40,222,14,228,27,91,110,236,58,44,8,0,47,149,208,0
[26] overpass turbo. (n.d.). Overpass Turbo. Retrieved December 12, 2021, from https://overpass-turbo.osm.ch/
[27] Ramm, Frederik: Routing Engines für OpenStreetMap, FOSSGIS Konferenz 2017: Passau 22. - 25. März 2017. https://doi.org/10.5446/30524
[28] Project OSRM. (n.d.). OSRM. Retrieved October 29, 2021, from http://project-osrm.org/
[29] The 2-Clause BSD License | Open Source Initiative. (n.d.). Open Source Initiative (Opensource). Retrieved October 29, 2021, from https://opensource.org/licenses/BSD-2-Clause
[30] Nolde, N. (2020c, December 4). Open Source Routing Engines And Algorithms - An Overview ». GIS • OPS. Retrieved October 29, 2021, from https://gis-ops.com/open-source-routing-engines-and-algorithms-an-overview/#user-content-osrm
[31] Nolde, N. (2020b, December 4). Open Source Routing Engines And Algorithms - An Overview ». GIS • OPS. Retrieved October 29, 2021, from https://gis-ops.com/open-source-routing-engines-and-algorithms-an-overview/#user-content-tldr-overview
[32] OSRM API Documentation. (n.d.). Project-Osrm. Retrieved November 3, 2021, from http://project-osrm.org/docs/v5.24.0/api/#trip-service
[33] Patterson, D. (2017, November 1). OSRM response to geojson · Issue #4651 · Project-OSRM/osrm-backend [Comment on the article " OSRM response to geojson #4651"]. GitHub.

https://github.com/Project-OSRM/osrm-backend/issues/4651
[34] Liedman, P. (n.d.). GitHub - perliedman/leaflet-routing-machine: Control for routing in Leaflet. GitHub. Retrieved October 29, 2021, from https://github.com/perliedman/leaflet-routing-machine
[35] OSRM API Documentation. (n.d.). OSRM API Docs (Project-Osrm). Retrieved October 29, 2021, from http://project-osrm.org/docs/v5.10.0/api/#route-service
[36] Open Source Routing Machine - OpenStreetMap Wiki. (n.d.). OpenStreetMap Wiki (Openstreetmap). Retrieved November 7, 2021, from https://wiki.openstreetmap.org/wiki/Open_Source_Routing_Machine
[37] Ramm, Frederik: Routing Engines für OpenStreetMap, FOSSGIS Konferenz 2017: Passau 22. - 25. März 2017. https://doi.org/10.5446/30524
[38] Open Source. (n.d.). GraphHopper Directions API. Retrieved October 29, 2021, from https://www.graphhopper.com/open-source/
[39] Apache License, Version 2.0 | Open Source Initiative. (n.d.). Open Source Initiative (Opensource). Retrieved October 29, 2021, from https://opensource.org/licenses/Apache-2.0
[40] Nolde, N. (2020d, December 4). Open Source Routing Engines And Algorithms - An Overview ». GIS • OPS. Retrieved October 29, 2021, from https://gis-ops.com/open-source-routing-engines-and-algorithms-an-overview/#user-content-graphhopper
[41] Documentation - GraphHopper Directions API. (n.d.). Graphhopper. Retrieved October 29, 2021, from https://docs.graphhopper.com/#operation/getRoute
[42] GraphHopper - OpenStreetMap Wiki. (n.d.). OpenStreetMap Wiki (Openstreetmap). Retrieved November 7, 2021, from https://wiki.openstreetmap.org/wiki/GraphHopper
[43] Ramm, Frederik: Routing Engines für OpenStreetMap, FOSSGIS Konferenz 2017: Passau 22. - 25. März 2017. https://doi.org/10.5446/30524
[44] Introduction for Users - Valhalla. (n.d.). Valhalla. Retrieved October 29, 2021, from https://valhalla.readthedocs.io/en/latest/valhalla-intro/#valhalla-open-source-routing
[45] The MIT License | Open Source Initiative. (n.d.). Open Source Initiative (Opensource). Retrieved October 29, 2021, from https://opensource.org/licenses/MIT
[46] Nolde, N. (2020a, December 4). Open Source Routing Engines And Algorithms - An Overview ». GIS • OPS. Retrieved October 29, 2021, from https://gis-ops.com/open-source-routing-engines-and-algorithms-an-overview/#user-content-valhalla
[47] Tile structure - Valhalla. (n.d.). Valhalla. Retrieved October 29, 2021, from https://valhalla.readthedocs.io/en/latest/tiles/
[48] API Overview - Valhalla. (n.d.). Valhalla. Retrieved October 29, 2021, from https://valhalla.readthedocs.io/en/latest/api/
[49] Geojson - Valhalla. (n.d.). Valhalla. Retrieved October 29, 2021, from https://valhalla.readthedocs.io/en/latest/mjolnir/geojson/
[50] GitHub - valhalla/valhalla: Open Source Routing Engine for OpenStreetMap. (n.d.). GitHub. Retrieved November 7, 2021, from https://github.com/valhalla/valhalla
[51] DE:Openrouteservice - OpenStreetMap Wiki. (n.d.). OpenStreetMap Wiki. Retrieved October 29, 2021, from https://wiki.openstreetmap.org/wiki/DE:Openrouteservice
[52] openrouteservice. (n.d.-c). Terms of Service | Openrouteservice. Retrieved October 29, 2021, from https://openrouteservice.org/terms-of-service/
[53] Creative Commons — Attribution 4.0 International — CC BY 4.0. (n.d.). Creative Commons (Creativecommons). Retrieved October 29, 2021, from https://creativecommons.org/licenses/by/4.0/

[54] Nolde, N. (2020d, December 4). Open Source Routing Engines And Algorithms - An Overview ». GIS • OPS. Retrieved October 29, 2021, from https://gis-ops.com/open-source-routing-engines-and-algorithms-an-overview/#user-content-openrouteservice-ors
[55] openrouteservice. (n.d.). Openrouteservice. Retrieved October 29, 2021, from https://openrouteservice.org/
[56] openrouteservice. (n.d.-a). Dashboard | ORS. Retrieved October 29, 2021, from https://openrouteservice.org/dev/#/api-docs/
[57] GitHub - GIScience/openrouteservice: The open source route planner api with plenty of features. (n.d.). GitHub. Retrieved November 7, 2021, from https://github.com/GIScience/openrouteservice
[58] openrouteservice. (n.d.-c). Running with Docker. Openrouteservice Documentation. Retrieved November 7, 2021, from https://giscience.github.io/openrouteservice/installation/Running-with-Docker.html
[59] Introduction — pgRouting Manual (3.2). (n.d.). PgRouting. Retrieved October 29, 2021, from https://docs.pgrouting.org/latest/en/pgRouting-introduction.html#licensing
[60] GNU General Public License version 2 | Open Source Initiative. (n.d.). Open Source Initiative (Opensource). Retrieved October 29, 2021, from https://opensource.org/licenses/GPL-2.0
[61] Dawes, B., Frey, D., Abrahams, D., & Rivera, R. (n.d.). Boost Software License. Boost Software License (Boost). Retrieved October 29, 2021, from https://www.boost.org/users/license.html
[62] Creative Commons — Attribution-ShareAlike 3.0 Unported — CC BY-SA 3.0. (n.d.). Creative Commons (Creativecommons). Retrieved October 29, 2021, from https://creativecommons.org/licenses/by-sa/3.0/
[63] Nolde, N. (2020e, December 4). Open Source Routing Engines And Algorithms - An Overview ». GIS • OPS. Retrieved October 29, 2021, from https://gis-ops.com/open-source-routing-engines-and-algorithms-an-overview/#user-content-pgrouting
[64] ST_AsGeoJSON. (n.d.). PostGIS. Retrieved October 29, 2021, from https://postgis.net/docs/ST_AsGeoJSON.html
[65] GitHub - pgRouting/pgrouting: Repository contains pgRouting library. Development branch is “develop”, stable branch is “master.” (n.d.). GitHub. Retrieved November 7, 2021, from https://github.com/pgRouting/pgrouting
[66] Table of Contents — pgRouting Manual (3.2). (n.d.). PgRouting. Retrieved October 29, 2021, from https://docs.pgrouting.org/latest/en/index.html
[67] Ramm, Frederik: Routing Engines für OpenStreetMap, FOSSGIS Konferenz 2017: Passau 22. - 25. März 2017. https://doi.org/10.5446/30524
[68] Abelshausen, B. (2016, October 14). OpenLR. Itinero. Retrieved October 29, 2021, from https://www.itinero.tech/2016/10/14/openlr/
[69] Class Vehicle | Itinero - Documentation. (n.d.). Itinero API Documentation. Retrieved October 29, 2021, from https://docs.itinero.tech/itinero/Itinero.Osm.Vehicles.Vehicle.html
[70] Abelshausen, B., & Amenta, J. (n.d.). GitHub - itinero/routing: The routing core of itinero. GitHub. Retrieved October 29, 2021, from https://github.com/itinero/routing
[71] Routing API | Itinero - Documentation. (n.d.). Itinero. Retrieved October 29, 2021, from http://docs.itinero.tech/docs/itinero/routing-api.html
[72] Itinero - OpenStreetMap Wiki. (n.d.). OpenStreetMap Wiki (Openstreetmap). Retrieved November 7, 2021, from https://wiki.openstreetmap.org/wiki/Itinero
[73] Ramm, Frederik: Routing Engines für OpenStreetMap, FOSSGIS Konferenz 2017: Passau 22. - 25.

März 2017. https://doi.org/10.5446/30524
[74] (2019, May 4). BRouter License Change GPLv3 MIT · Issue #149 · abrensch/brouter [Comment on the article " BRouter License Change GPLv3 MIT #149 "]. GitHub. https://github.com/abrensch/brouter/issues/149
[75] BRouter: Freely configurable routing profile. (n.d.). BRouter. Retrieved October 29, 2021, from https://brouter.de/brouter/costfunctions.html
[76] BRouter - OpenStreetMap Wiki. (n.d.). OpenStreetMap Wiki (Openstreetmap). Retrieved November 7, 2021, from https://wiki.openstreetmap.org/wiki/BRouter
[77] Ramm, Frederik: Routing Engines für OpenStreetMap, FOSSGIS Konferenz 2017: Passau 22. - 25. März 2017. https://doi.org/10.5446/30524
[78] Bishop, A. M. (2020, December 30). Routino : Router for OpenStreetMap Data. Routino. Retrieved October 29, 2021, from https://www.routino.org/
[79] GNU General Public License version 3 | Open Source Initiative. (n.d.). Open Source Initiative (Opensource). Retrieved October 29, 2021, from https://opensource.org/licenses/GPL-3.0
[80] Free Software Foundation, Inc. . (2007, November 19). GNU Affero General Public License - GNU Project - Free Software Foundation. GNU Betriebssystem (GNU). Retrieved October 29, 2021, from http://www.gnu.org/licenses/agpl-3.0.html
[81] Routino : Route Planner for OpenStreetMap Data. (n.d.). Routino. Retrieved October 29, 2021, from http://www.routino.org/uk-openlayers2/router.html
[82] Bishop, A. M. (2021, January 1). Routino : Output. Routino. Retrieved October 29, 2021, from https://www.routino.org/documentation/output.html
[83] Routino - OpenStreetMap Wiki. (n.d.). OpenStreetMap Wiki (Openstreetmap). Retrieved November 7, 2021, from https://wiki.openstreetmap.org/wiki/Routino
[84] OSMnx 1.1.1 — OSMnx 1.1.1 documentation. (n.d.). OSMnx. Retrieved October 29, 2021, from https://osmnx.readthedocs.io/en/stable/
[85] Samadelli, M., & Keller, S. (2021, July). Programmier-Werkzeuge für das Analysieren von Netzwerken mit OpenStreetMap- und weiteren Datenquellen - HedgeDoc. HedgeDoc - Collaborative Markdown Notes. Retrieved October 29, 2021, from https://md.coredump.ch/s/sFsLf_coX#
[86] GitHub - gboeing/osmnx: OSMnx: Python for street networks. Retrieve, model, analyze, and visualize street networks and other spatial data from OpenStreetMap. (n.d.). GitHub. Retrieved November 7, 2021, from https://github.com/gboeing/osmnx
[87] Tenkanen, H., & Pyrosm Contributors. (2020). Pyrosm. Pyrosm. Retrieved October 29, 2021, from https://pyrosm.readthedocs.io/en/latest/
[88] GitHub - HTenkanen/pyrosm: Read OpenStreetMap data from Protobuf files into GeoDataFrame with Python, faster. (n.d.). GitHub. Retrieved November 7, 2021, from https://github.com/HTenkanen/pyrosm
[89] PlantUML. (n.d.-a). Klassendiagramm Syntax und Funktionen. PlantUML.com. Retrieved December 20, 2021, from https://plantuml.com/de/class-diagram
[90] H., D. J., Shea, K., Tin, E., Krasnyk, M., Lebedev, Y., P., M., Luxen, D., Ghosh, K., & Patterson, D. (2016, April 5). osrm-backend/foot.lua at master · Project-OSRM/osrm-backend. GitHub. Retrieved December 12, 2021, from https://github.com/Project-OSRM/osrm-backend/blob/master/profiles/foot.lua
[91] Kurviger - Dein Motorrad-Routenplaner. (n.d.). Kurviger.de. Retrieved December 12, 2021, from https://kurviger.de/
[92] Clawson, K. (2018, August 14). Display issue with accordion nested inside overlay · Issue #6318 ·

primefaces/primeng. GitHub. Retrieved December 12, 2021, from https://github.com/primefaces/primeng/issues/6318
[93] Topinka, T. (2021, February 6). Statistic - IntelliJ IDEs Plugin | Marketplace. JetBrains Marketplace. Retrieved December 16, 2021, from https://plugins.jetbrains.com/plugin/4509-statistic
[94] Welcome to. (n.d.). Python.Org. Retrieved December 12, 2021, from https://www.python.org/
[95] FastAPI. (n.d.). FastAPI. Retrieved December 12, 2021, from https://fastapi.tiangolo.com/
[96] PostgreSQL: The World's Most Advanced Open Source Relational Database. (n.d.). PostgreSQL. Retrieved December 12, 2021, from https://www.postgresql.org/
[97] Bartunov, O., & Sigaev, T. (2021, November 11). F.16. hstore. PostgreSQL Documentation. Retrieved December 12, 2021, from https://www.postgresql.org/docs/current/hstore.html
[98] PostGIS — Spatial and Geographic Objects for PostgreSQL. (n.d.). PostGIS Project Steering Committee (PSC). Retrieved December 12, 2021, from https://postgis.net/
[99] OpenStreetMap. (n.d.-a). OpenStreetMap. Retrieved December 12, 2021, from https://www.openstreetmap.org/
[100] Angular. (n.d.-b). Angular IO. Retrieved December 12, 2021, from https://angular.io/
[101] Project OSRM. (n.d.). Project OSRM. Retrieved December 12, 2021, from http://project-osrm.org/
[102] Geofabrik Download Server. (n.d.). Geofabrik. Retrieved December 12, 2021, from https://download.geofabrik.de/europe/switzerland.html
[103] PlantUML. (o. D.). Deployment Diagram syntax and features. PlantUML.com. Retrieved December 22, 2021, von https://plantuml.com/de/deployment-diagram
[104] Traefik. (o. D.). Traefik. Retrieved December 22, 2021, from https://doc.traefik.io/traefik/

Appendix D: Mockup Screens Bilder

Routen-Suche Desktop

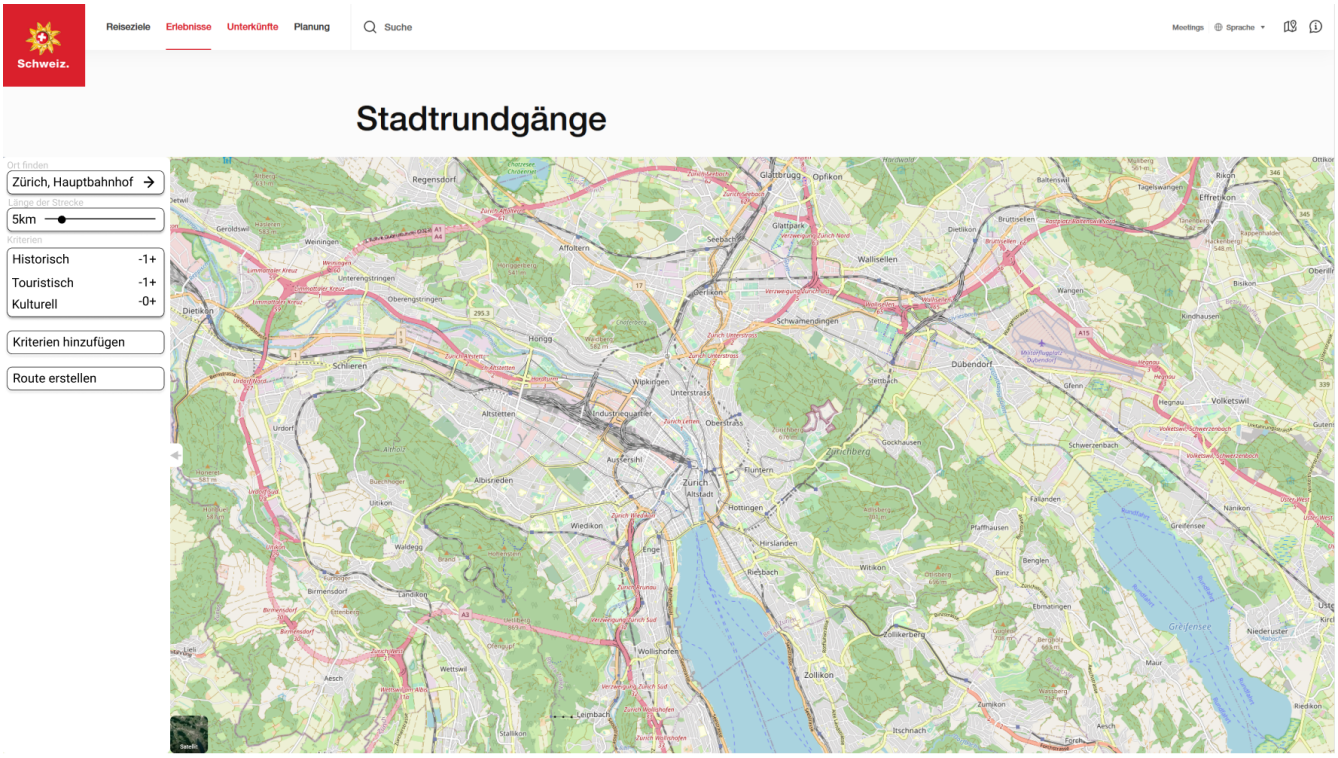


Abbildung 39. Routen-Suche Desktop

Error-Darstellung Desktop

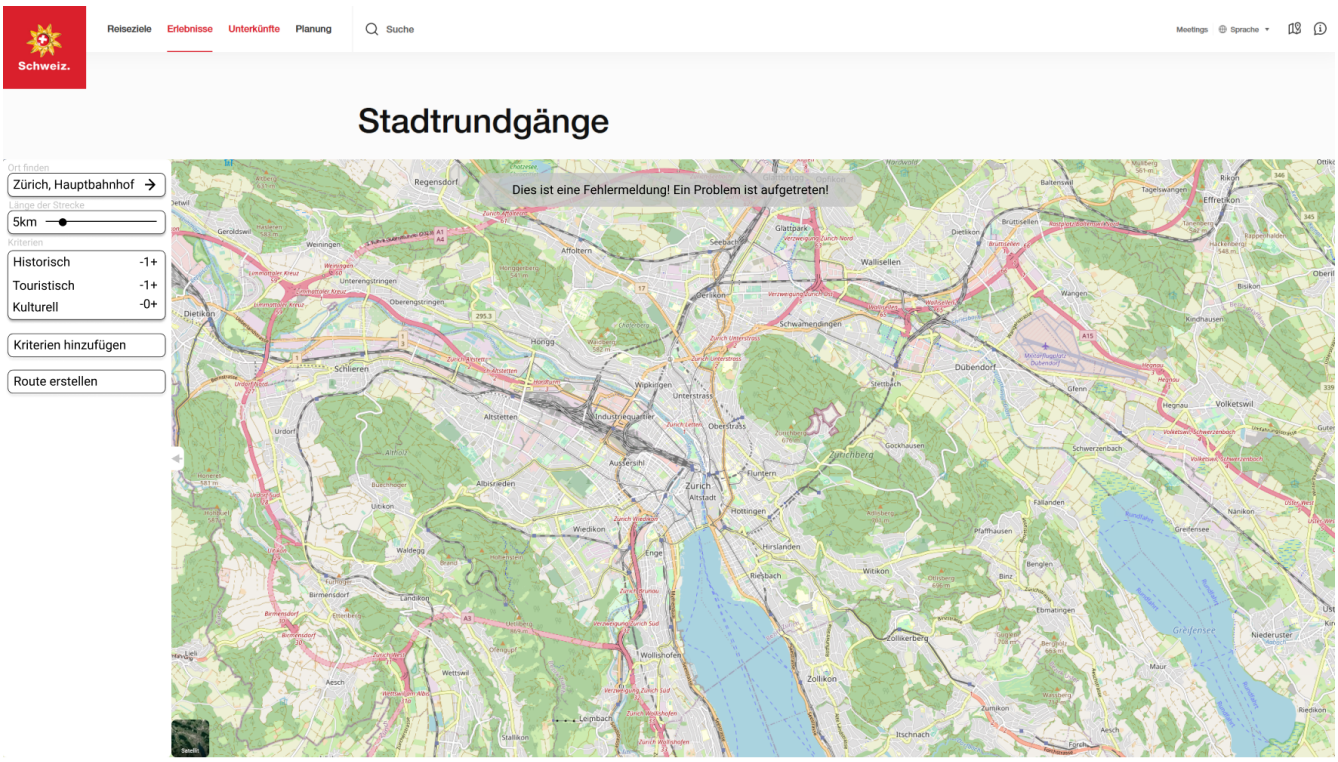


Abbildung 40. Error-Darstellung Desktop

Routen-Darstellung Desktop

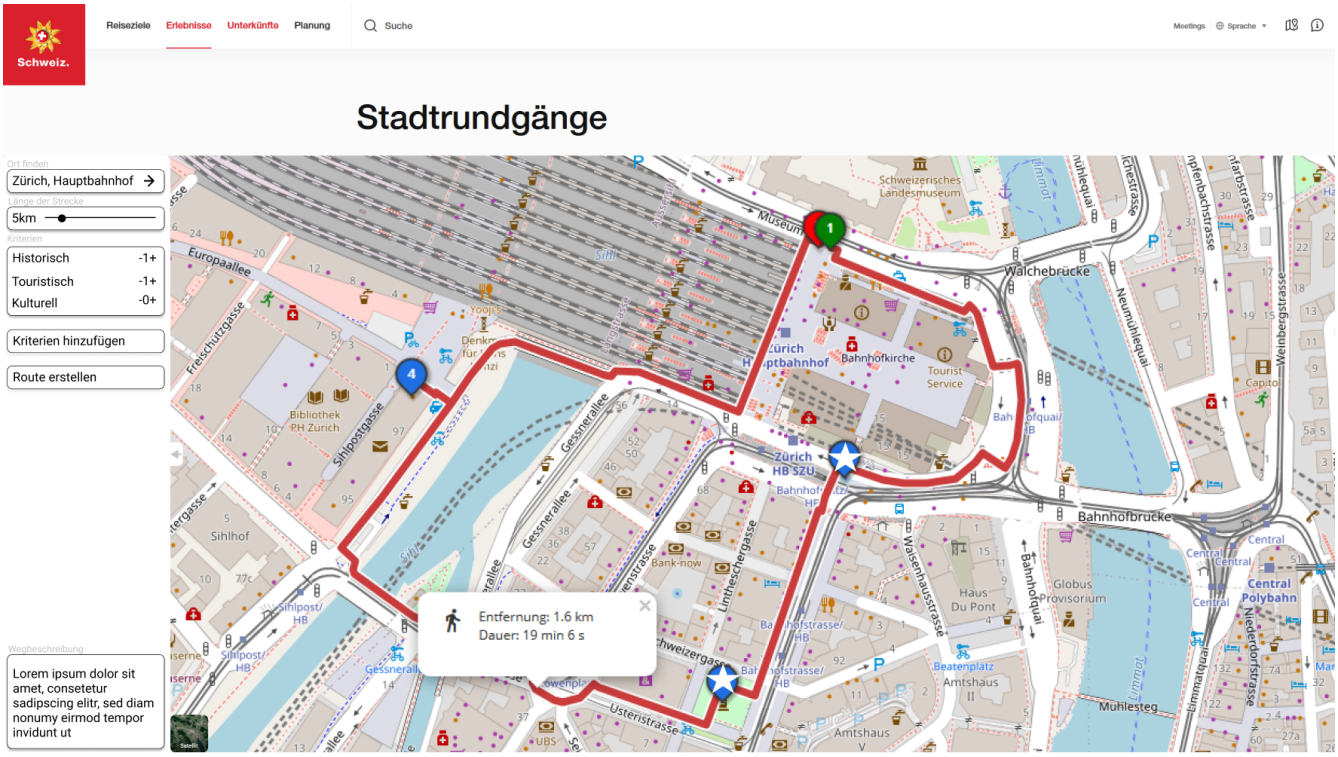


Abbildung 41. Routen-Darstellung Desktop

Eingeklapptes Menu Desktop

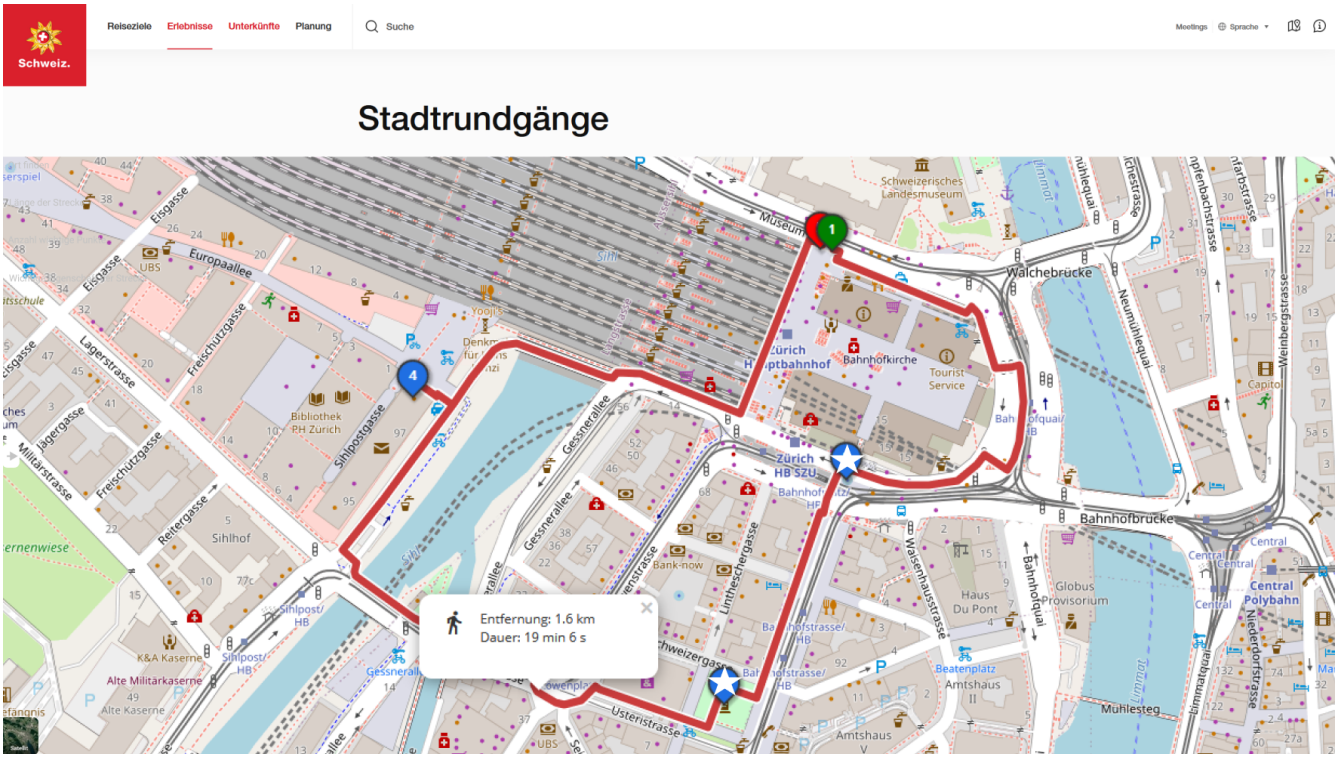


Abbildung 42. Eingeklapptes Menu Desktop

Wichtige Punkte Desktop

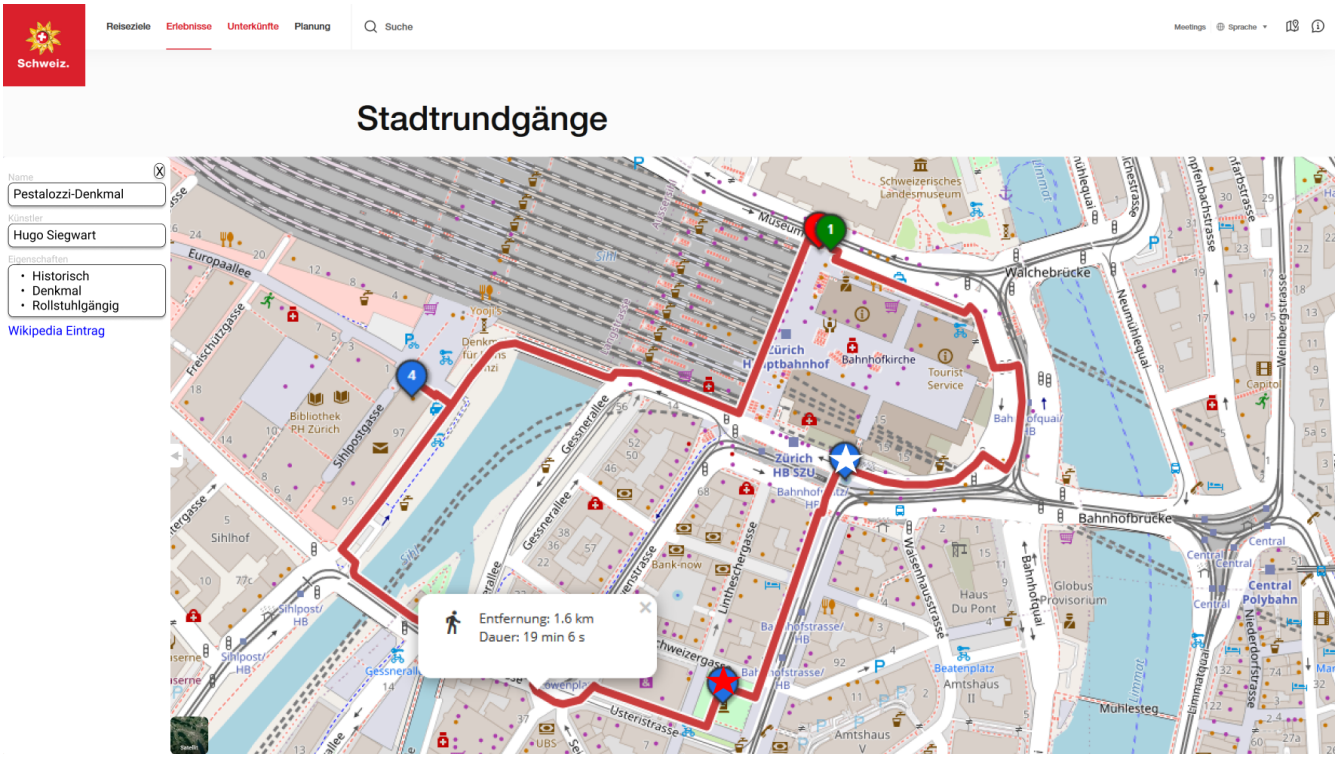


Abbildung 43. Wichtige Punkte Desktop

Routen-Suche Variante 1 Mobile

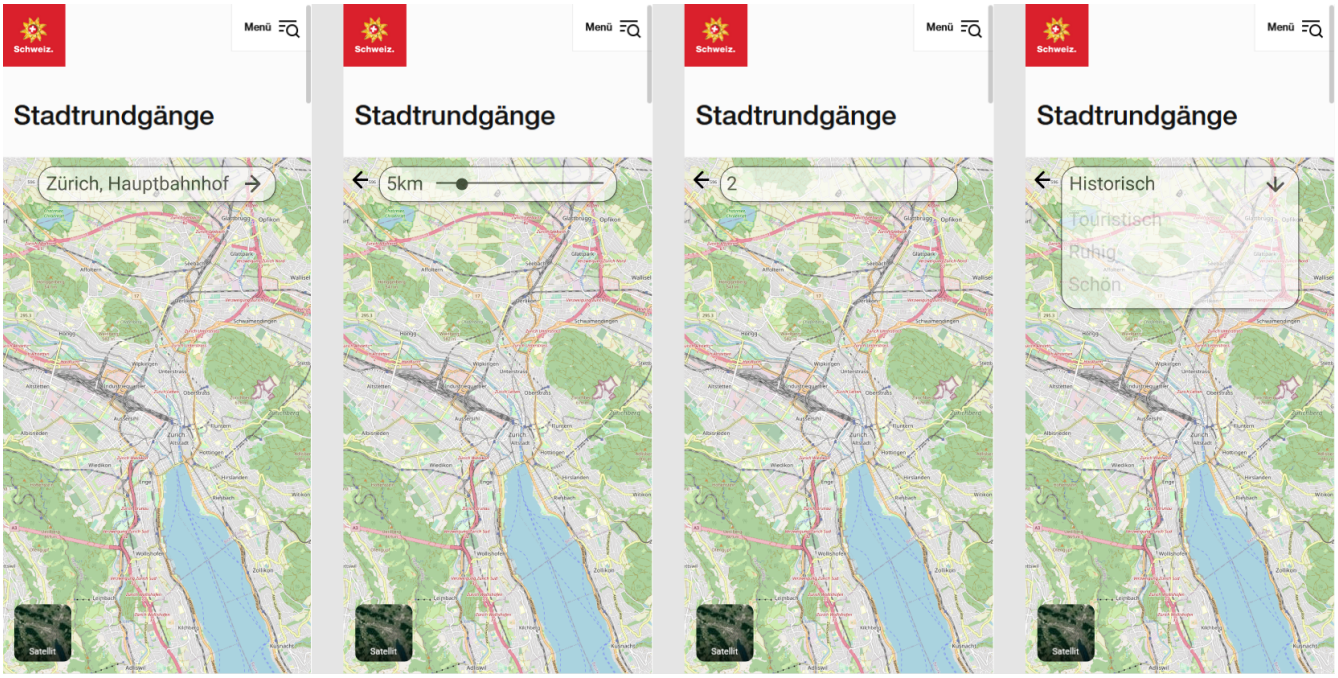


Abbildung 44. Routen-Suche Variante 1 Mobile

Routen-Suche Variante 2 Mobile

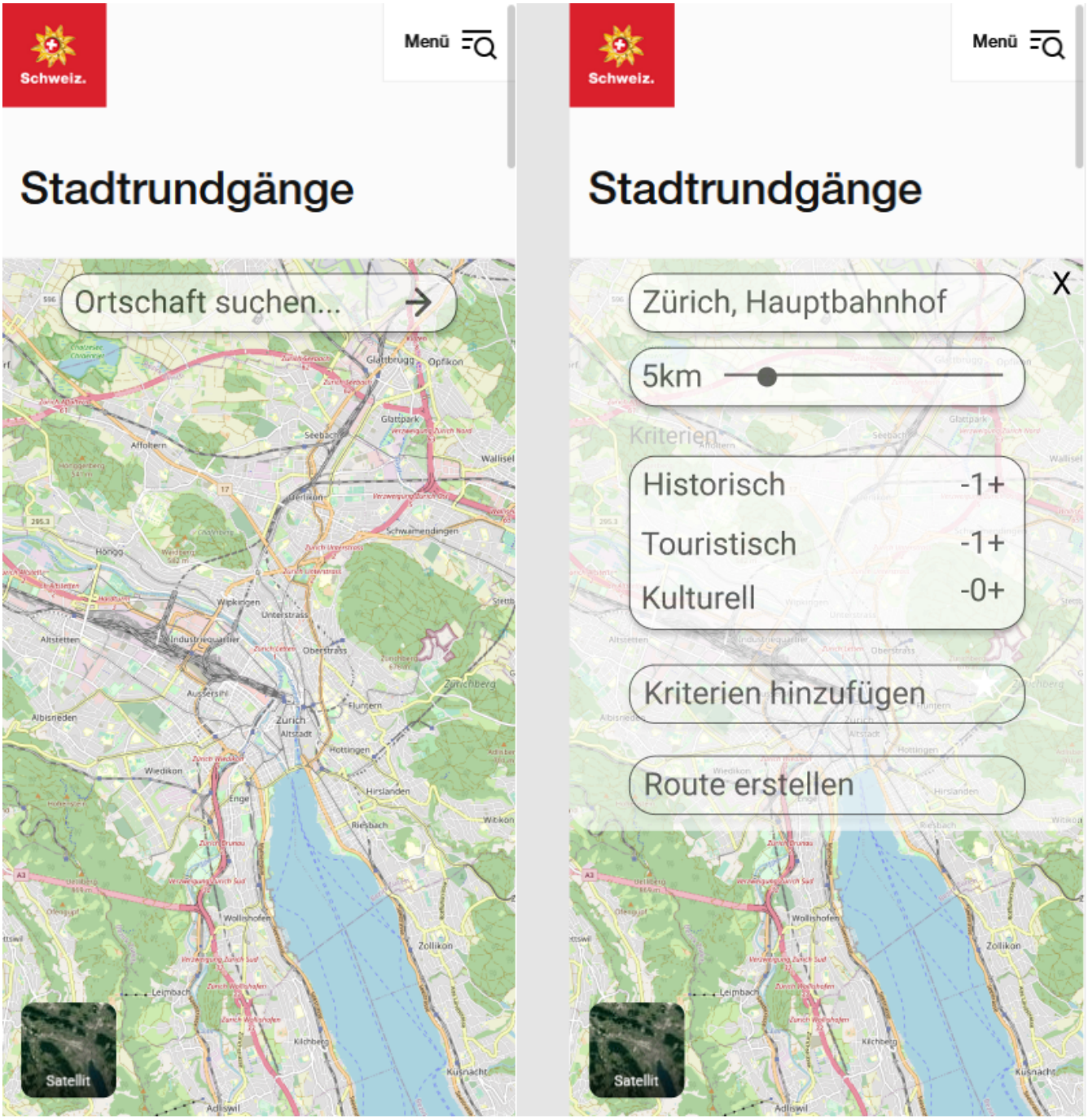


Abbildung 45. Routen-Suche Variante 2 Mobile

Routen-Darstellung Mobile

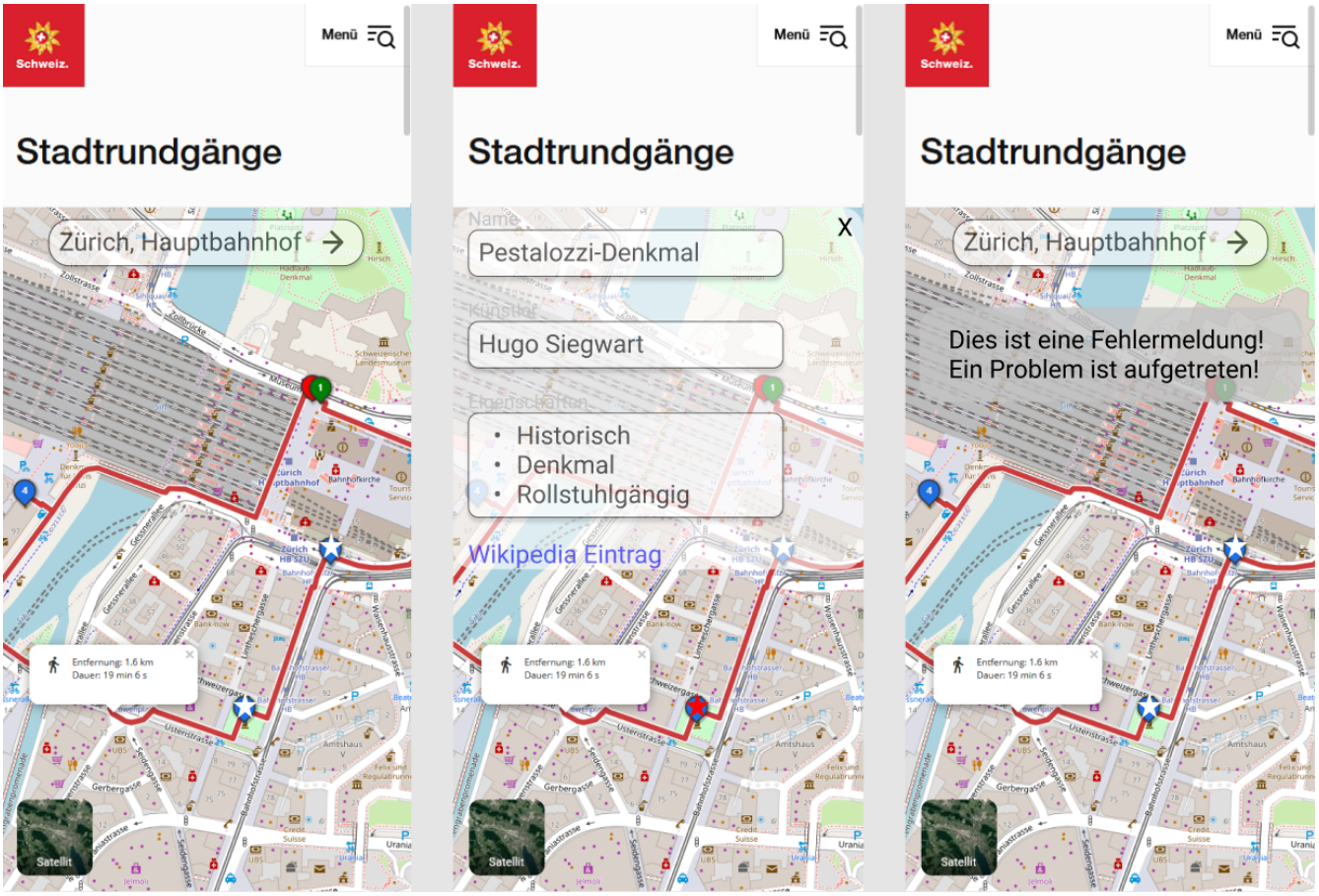


Abbildung 46. Routen-Darstellung Mobile

Appendix E: Dockerisierung der Applikation

Die gesamte Applikation wird mithilfe von Docker virtualisiert und kann dann in einer Container-Umgebung gestartet werden. Dies bietet eine sehr grosse Flexibilität hinsichtlich Konfiguration und Betrieb.

Die Umgebung für die Entwicklung sieht wie folgt aus:

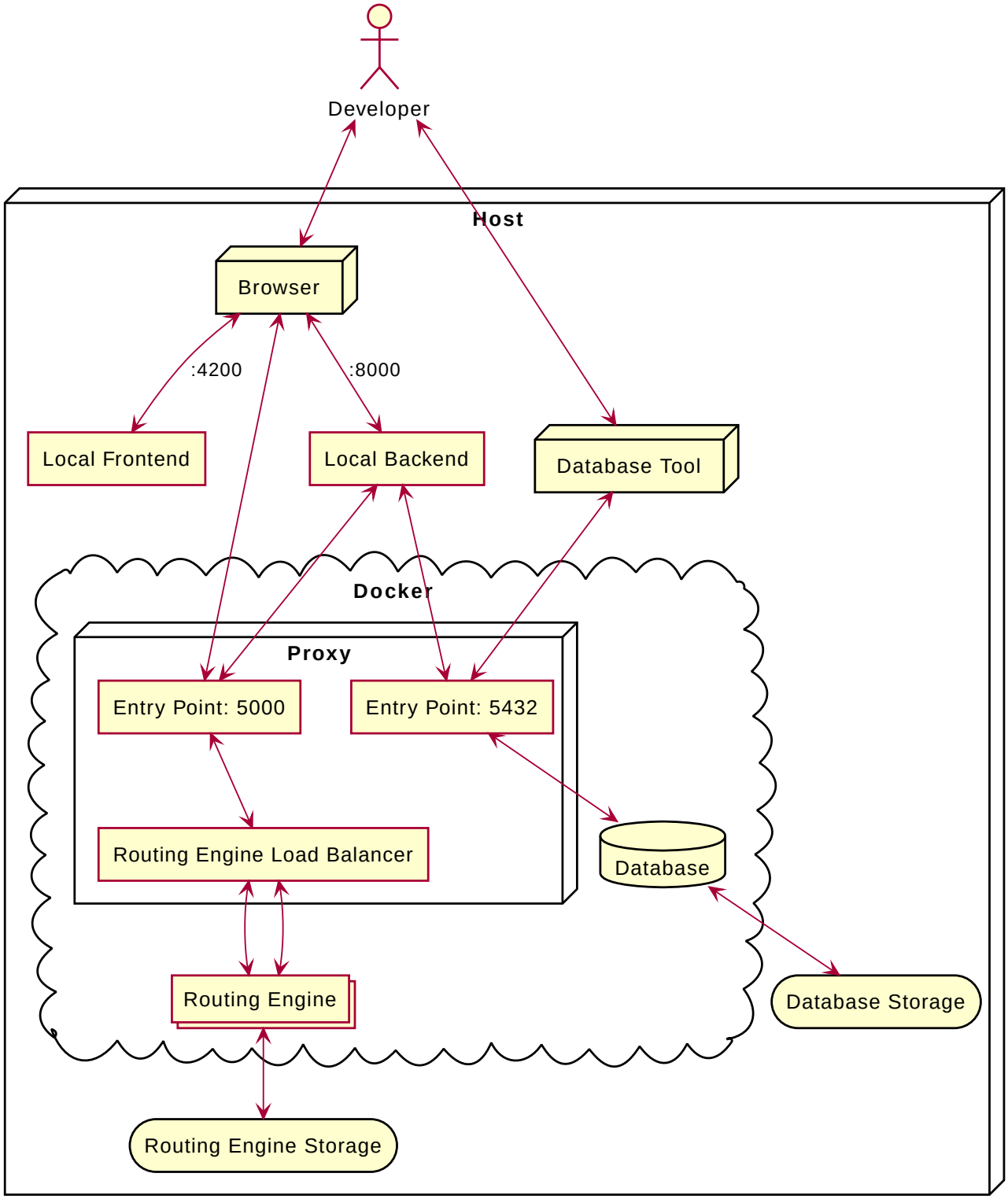


Abbildung 47. UML Deployment-Diagramm [103] Umgebung für die Entwicklung

Der Entwickler kann bei sich lokal das Docker Setup deployen und die zu entwickelnden Komponenten aus einer IDE oder in einem Terminal starten. Die Datenbank und die Routing Engine besitzen ein Volume auf dem Hostsystem, damit nach einem Neustart die Daten nicht verloren gehen. So können Upgrades eingespielt werden, ohne die Daten zu verlieren. Wichtig zu erwähnen ist, dass als Proxy ein Reverse Proxy eingesetzt wird.

Ein mögliches Deployment für die Produktion wurde im folgenden Diagramm abgebildet.

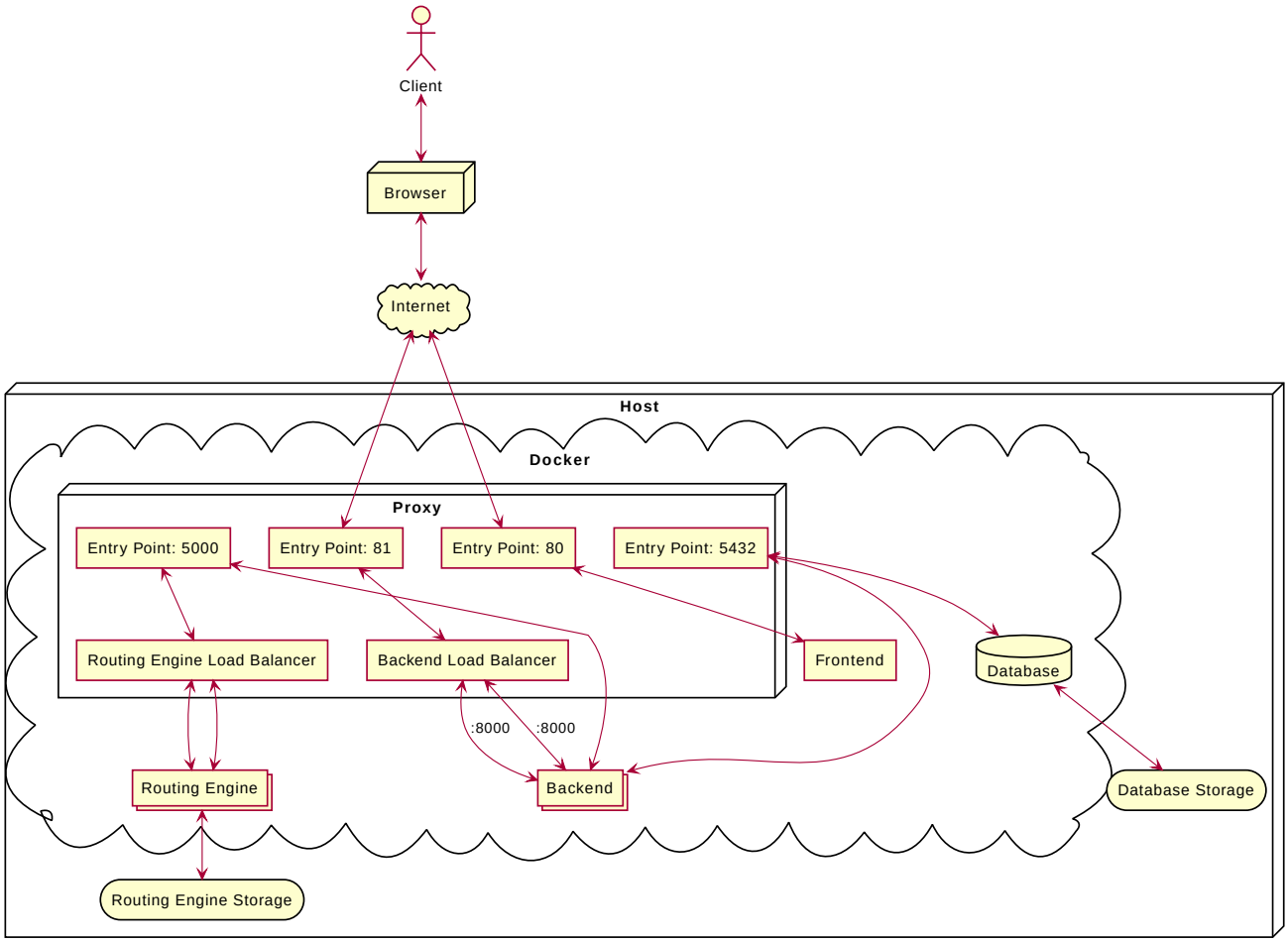


Abbildung 48. UML Deployment-Diagramm [103] Beispielumgebung für die Produktion

Sämtliche Zugriffe gehen in diesem Beispiel über den Proxy. In unserem Beispiel verwenden wir Traefik [104], welcher sehr viele nützliche Features, wie Load Balancing und TLS, enthält. Die gezeigten Diagramme dienen zur Veranschaulichung der Möglichkeiten, an der Applikation City Trip Planner weiterzuentwickeln respektive sie zu betreiben.

Appendix F: Persönliche Berichte

F.1. Jan Ruch

Nach meinen Erfahrungen startet man oft überhastet in ein Projekt. Man ist enthusiastisch und möchte am liebsten direkt mit dem Coding beginnen. In dieser Arbeit haben wir eine andere Herangehensweise gewählt. Zu Beginn dieser Arbeit hatte ich grosse Sorgen, dass mich die neuen Technologien überwältigen. Bislang hatte ich noch nie grössere Projekte mit Python durchgeführt, auch Spatial-Data und -Queries habe ich noch nie benutzt. Wir haben uns in den ersten knapp 5 Wochen nur mit der Sprache Python, dem Framework FastAPI, der DB-Erweiterung PostGIS und pgRouting auseinandergesetzt. Wir haben erst in der Woche 6 mit der eigentlichen Arbeit für unser Projekt begonnen.

Nach diesen ausführlichen Einführungen war das Programmieren selbst kein grosses Problem mehr. Besonders das Python-Framework war eingängig und verständlich. Mit den Spatial Queries hatte ich allerdings länger meine Mühen. Ich habe immer wieder viel Zeit benötigt, um herauszufinden welche Spalten und Funktionen für die Abfragen benötigt werden. Angular habe ich bereits einige Male entwickelt, aber ein neues Projekt in diesem Rahmen war auch für mich eine willkommene Auffrischung meines Wissens.

Die Arbeit hat mir grossen Spass gemacht und war eine wertvolle Erfahrung in meinem Studium. Ich habe gemeinsam mit Lukas viel gelernt, auch im Bereich Arbeitsteilung und Organisation. Das Projekt selbst ist zum Zeitpunkt der Abgabe in einem guten Zustand, aber selbstverständlich ausbaufähig. Ich hoffe, dass wir die Möglichkeit erhalten das Projekt voranzutreiben und in einer Bachelorarbeit weiter auszubauen.

Wenn ich etwas kritisieren müsste, dann hätte ich gerne mehr Zeit für die Implementation aufgewendet. Aufgrund von nebenläufigen Projekten war dies leider nicht möglich. Ich nehme mir diese persönliche Kritik als grosses Ziel für die Bachelorarbeit vor! Trotzdem habe ich das Gefühl, dass ich auch gerade im Bereich Implementation einiges dazu gelernt habe. Die vielen Code Reviews und Pair Programming Sessions haben sich, nach unseren Erfahrungen im EPJ, erneut ausgezahlt.

Um auf meine einleitenden Worte zurückzufinden, die unterschiedliche Herangehensweise hat sich in diesem Projekt sehr gelohnt. Ich denke, durch die lange Einarbeitungszeit hatten wir die Gelegenheit uns auch iterativ Gedanken über die Software zu machen. Alle Erkenntnisse konnten wir in Woche 6 direkt miteinfliessen lassen. Ich bin sehr zufrieden mit dem Abschluss des Projektes und bedanke mich bei allen Beteiligten für die spannende Erfahrung!

F.2. Lukas Leuenberger

In meiner bisherigen Studienzeit und in meinem Arbeitsumfeld bin ich noch nie mit Geoinformationssystemen in Berührung gekommen. Ich nutze rege Online-Kartendienste für Wanderungen und die Routenplanung mit dem Auto. Allerdings konnte ich mir nie vorstellen, wie ein solches System funktioniert. Diese Studienarbeit gab mir die Möglichkeit, in ein bisher unbekanntes Themengebiet der Informatik einzutauchen. Um ein Verständnis zu entwickeln, was alles möglich ist, habe ich mich im Bereich Spatial Queries mit PostgreSQL eingelesen. Ich bin fasziniert und begeistert, was alles möglich ist! Ich konnte die Schweiz mit SQL-Queries aus einem neuen Blickwinkel kennenlernen.

Die nächste Hürde war die Programmiersprache Python. Bislang habe ich noch nie ein Projekt in Python umgesetzt. Ich kenne Java und diverse Java-Frameworks sehr gut. Die Konzepte der objektorientierten Sprache und von objektrelationalen Mappern wie JPA/Hibernate wende ich in meinem Arbeitsalltag an. Es galt nun dieses Wissen mithilfe von Python umzusetzen. Ich erlebte eine grosse Lernkurve, da oftmals nicht das Was, sondern das Wie mich vor Herausforderungen gestellt hatte. Aus heutiger Sicht bin ich stolz, das Projekt mit einer mir bislang unbekannten Programmiersprache umgesetzt zu haben.

Besonders hervorheben möchte ich die Architektur des Gesamtsystems und der Backend-Applikation. Mir wurde auf eine schöne Art und Weise vor Augen geführt, wie wichtig dieses Thema ist und welche Auswirkungen es auf die Entwicklung hat. Tatsächlich können viele Probleme bereits erkannt und gelöst werden, bevor nur eine Zeile Code geschrieben oder getestet wird. Es freut mich deshalb umso mehr, dass wir keine Kompromisse eingehen mussten und das System wie entworfen implementieren konnten. Diese Erfahrung der eingehenden Anforderungsanalyse und -spezifikation, das Design eines neuen Systems und die dazugehörende Implementation zu erstellen sind meine persönlichen Highlights dieser Arbeit.

Wir haben während der ganzen Arbeit Lernjournale geführt, um Beispiele aus der Praxis und erlangtes Wissen festzuhalten. Diese Journale unterstützten während der ganzen Arbeit den Lernprozess. Das Niederschreiben von Problemen und deren Ursachen ist eine Arbeitstechnik, welche mir persönlich sehr geholfen hat, ein vertieftes Verständnis von Software Entwicklung zu erlangen. Ich werde dies auch für meine Bachelorarbeit in Betracht ziehen.

Die letzte Herausforderung war das Erlernen von Angular. Ich kannte das Framework nur oberflächlich aus vergangenen Projekten. In dieser Arbeit hatte ich die Möglichkeit, das erste Mal in meiner Karriere als Full Stack Developer zu arbeiten. Ich war eingangs skeptisch, all diese neuen Sachen zu meistern. Allerdings hat sich der Fleiss und die aufgewendete Zeit bezahlt gemacht. Ich nehme viele tolle Erfahrungen aus dieser Studienarbeit mit. Die wichtigste Erkenntnis für mich persönlich ist, dass die im Studium erlernten Konzepte funktionieren und dass ich in der Lage bin sie in die Praxis umzusetzen.