HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

IFS INSTITUTE FOR SOFTWARE

ÖV-Güteklassen 2018

Neue Methodik zur Beurteilung der Erschliessung durch den öffentlichen Verkehr

Bachelorarbeit

HSR Hochschule für Technik Rapperswil
Institut für Software

Frühlingssemester 2018

Autoren: Jonas Matter, Robin Suter
Betreuer: Prof. Stefan Keller
Experte: Claude Eisenhut

Wenden wir uns der Vergangenheit zu,
das wird ein Fortschritt sein

„Tornate all'antico e sarà un progresso", schrieb Giuseppe Verdi
am 5. Januar 1871 an Francesco Florimo

Abstract

ÖV-Güteklassen werden für die Beurteilung der Erschliessung mit dem öffentlichen Verkehr verwendet. Die heute anerkannte Spezifikation der ÖV-Güteklassen des Bundesamts für Raumentwicklung (ARE) basiert auf einer inzwischen ersetzten Schweizer Norm aus dem Jahre 1993. Diverse Kantone haben in Eigeninitiativen Anpassungen daran vorgenommen, um kantonalen Gegebenheiten gerecht zu werden. Die Implementationen dieser sind überholt und werden den aktuellen technischen Möglichkeiten nicht gerecht. So führen einige Kantone die Berechnung dieser in Handarbeit durch. Ebenfalls wird das Einzugsgebiet mit Luftlinien berechnet und die Topographie nicht konsequent in die Berechnung miteinbezogen. Die kantonalen Eigenlösungen zeigen, dass noch kein Konsens einer aktuellen Lösung vorhanden ist.

ÖV-Güteklassen 2018 kombiniert die Erkenntnisse einiger kantonalen Lösungen und verbessert diese mit den aktuellen technischen Möglichkeiten mit dem Ziel, eine allgemeingültige Spezifikation für die Schweiz zu erstellen. In der Spezifikation wird nun vorausgesetzt, dass das Einzugsgebiet auf einem Fussgänger-Routing-Graphen berechnet wird. Ebenfalls soll die Topographie konsequent berücksichtigt werden. Diese Anforderung wird mit der Berechnung von Leistungskilometern erreicht.

Der ÖV-Güteklassen 2018 Generator setzt die neu erstellte Spezifikation um. Er erzeugt einen schweizweiten Geodatensatz. So werden mithilfe von OpenStreetMap-Daten und mit der pgRouting-Software (PostgreSQL) auf einem Fussgänger-Graphen für jedes Einzugsgebiet Isochronen berechnet. Dadurch erhält man ein adäquates Verständnis über die Erreichbarkeit einer Haltestelle. Durch den Einsatz des hochaufgelösten digitalen Terrainmodells swissALTI^{3D} von Swisstopo wird die Topographie in die Berechnung miteinbezogen. Eine Haltestelle auf einem schwer begehbaren Gelände hat somit ein kleineres Einzugsgebiet. Die Berechnung des Kursintervalls einer Haltestelle wurde überdacht und eine neue Formel definiert. Der Generator erlaubt die automatisierte Berechnung der ÖV-Güteklassen für verschiedene Stichtage.

Zur Veranschaulichung werden die berechneten ÖV-Güteklassen 2018 in einer Webapplikation dargestellt. Dabei können diese den ÖV-Güteklassen des ARE überlagert werden.

ÖV-Güteklassen 2018 geht mit der Zeit und passt sich denn aktuellen technischen Möglichkeiten an. Durch die Konsolidierung einiger kantonaler Lösungen und Berücksichtigung der genauen Wegführung und Topographie wurde der erste Schritt in Richtung einer schweizweit anerkannten ÖV-Güteklassen-Spezifikation gemacht.

Abstract

Public transport quality gradings („ÖV-Güteklassen“) are used to measure the coverage radius of public transportation stations by distance and time. The currently accepted specification by the Swiss Federal Office for Spatial Development (ARE) is based on an outdated standard from 1993. Multiple cantons have since made adjustments to the methods to cater to their specific needs. The varied alterations show that there is a need to create a general solution for Switzerland. The cantonal adjustments and the current specification use linear distance for coverage radii and do not incorporate topographic data into the calculation.

This thesis introduces a new uniform specification for public transport quality gradings („ÖV-Güteklassen 2018“) which incorporates the varied approaches from the cantons and combines them with more modern technical capabilities. The specification requires the coverage areas to be calculated along a pedestrian routing graph that incorporates topography of the area as well as the frequency of service.

Additionally, we created a software that implements the newly designed specification. With the help of the routing engine „pgRouting“, we create coverage areas along a pedestrian routing graph that allows for a more detailed understanding of the public transport coverage radius. A high-resolution digital height model from Swisstopo integrates the topography into this radius allowing a public transport stop on a steep hill to show a smaller area of coverage. Furthermore, we define a new formula to calculate the time intervals of public transport stations and incorporate that into our quality measurement.

In order to evaluate the effectiveness of the calculation for the new specification, we created a web application which allows the new coverage areas to be examined and compared with the current public transport gradings from ARE. This web application allows for analysis of the differences between the two specifications and emphasizes the greater accuracy of the proposed new specification to form a unified public transit quality grading throughout the country.

Management Summary

Ausgangslage

In der Verkehrs- und Raumplanung werden ÖV-Güteklassen für die Beurteilung eines Standortes bezüglich der Erschliessung mit dem öffentlichen Verkehr verwendet. Durch diese können Entscheidungen betreffend der Standortentwicklung getroffen werden. Die gängige Strategie sieht vor, an bereits gut erschlossenen Standorten zu verdichten statt neues Bauland zu erschliessen, um nachträglich die ÖV-Infrastruktur hochziehen zu müssen. Der Nutzen der ÖV-Güteklassen beschränkt sich nicht auf Verkehrs- und Raumplaner. Privatpersonen können ebenso bei der Wohnungs- wie auch der Jobsuche davon Gebrauch machen. Ein Wohnort mit guter ÖV-Anbindung ist in Zeiten erhöhter Mobilität von hohem Stellenwert.

Die Methodik zur Erhebung von ÖV-Güteklassen wurde erstmals 1993 in einer inzwischen ersetzten Schweizer Norm festgelegt. Auf Grundlagen dieser hat das Bundesamt für Raumentwicklung (ARE) sowie verschiedene Kantone Berechnungsmethodiken entwickelt. Diese weichen unterschiedlich stark von der Norm ab.

Diese Berechnungsmethodiken sind für die heutigen technischen Möglichkeiten überholt. So wird unteranderem das Einzugsgebiet wie in Abbildung 1 sichtbar mit Luftlinien berechnet.

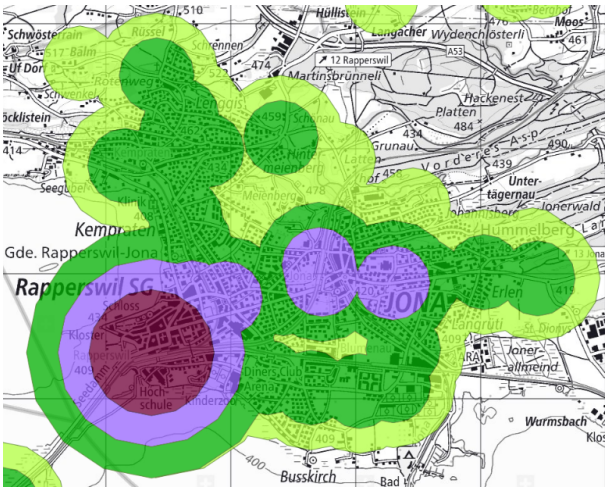

Abbildung 1: Luftlinie bei Einzugsgebieten [1]

Der Wegführung wird in den aktuellen Lösungen keine und der Topographie nur wage Beachtung geschenkt. So ist nicht klar, in welchem Mass eine anspruchsvolle Topographie zu Buche schlägt, denn eine Haltestelle in einem steilen Gebiet hat ein geringeres Einzugsgebiet als eine Haltestellen in einem flachen Gelände.

Erfahrungen aus der Praxis haben gezeigt, dass die bestehenden Lösungen einige Tücken in der Berechnung der Güteklassen aufweisen, welche zu einem falschen Schluss führen können.

Ziele, Vorgehen und Technologien

Im Kontext der Arbeit ÖV-Güteklassen 2018 wird in einem ersten Schritt eine neue Spezifikation entwickelt, welche auf dem Fundament der bestehenden Lösungen aufbaut, die Probleme und Learnings der kantonalen Anpassungen aufgreift und den aktuell technischen Möglichkeiten angleicht.

So soll unter anderem die genaue Wegführung beim Berechnen des Einzugsgebiet berücksichtigt und der Topographie Rechnung getragen werden. Dies ist mit freiverfügbaren Kartedaten von OpenStreetMap (OSM) und dem hoch aufgelöstem digitalen Terrainmodell (DTM) swissALTI^{3D} von Swisstopo möglich.

Die theoretisch erarbeitete Spezifikation *ÖV-Güteklassen 2018* wird direkt auf ihre Machbarkeit geprüft und als Generator umgesetzt, welche die Güteklassen berechnet.

Abgerundet wird Arbeit durch die Visualisierung der ÖV-Güteklassen und einem direkten Vergleich dieser basierend auf der neu erarbeitenden Spezifikation und der bestehenden Lösung des Bundesamts für Raumentwicklung (ARE).

Ergebnisse

Durch die Erarbeitung der *ÖV-Güteklassen 2018*-Spezifikation und deren technischer Umsetzung lassen sich Güteklassen generieren. Diese sind in Abbildung 2 ersichtlich.

Die Visualisierung bietet Raum- und Verkehrsplaner sowie Privatpersonen die Möglichkeit, Standorte auf die Qualität der ÖV-Erschliessung analysieren zu können, um so fundierte Entscheidungen in ihren Problemdomänen treffen zu können.

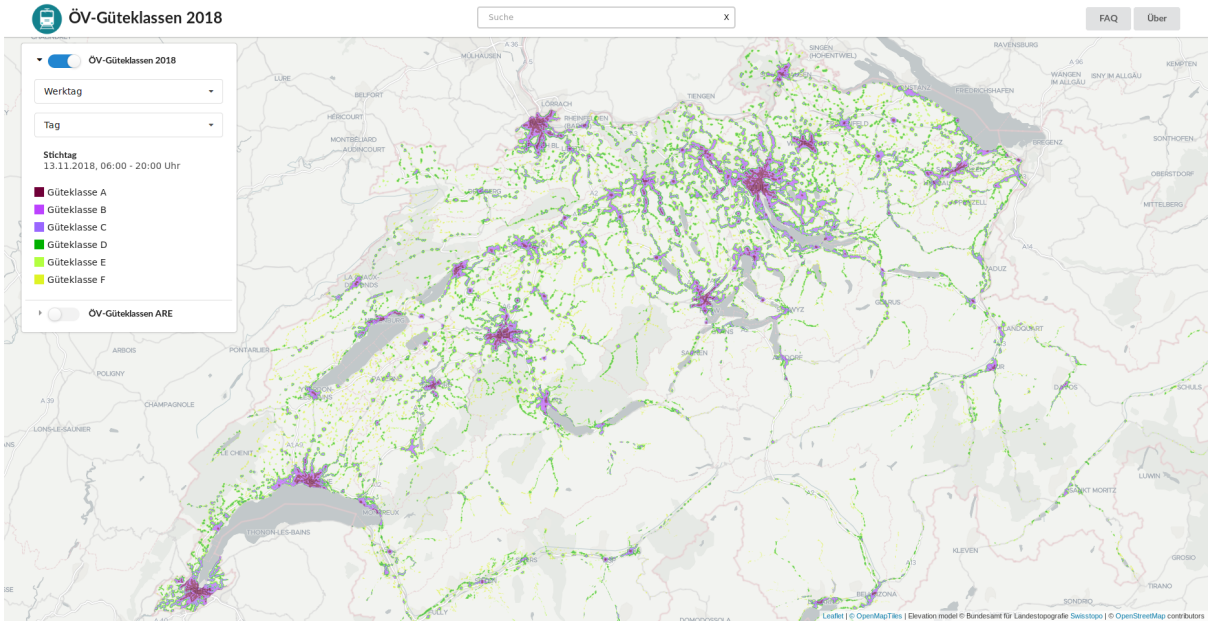

Abbildung 2: Darstellung der berechneten ÖV-Güteklassen in der Web-Applikation

Dabei wird unter anderem zusätzlich die Wegführung berücksichtigt, wie in Abbildung 3 ersichtlich

ist. Die Topographie wird durch die Verwendung von Leistungskilometern in der Berechnung des Einzugsgebiet miteinbezogen.

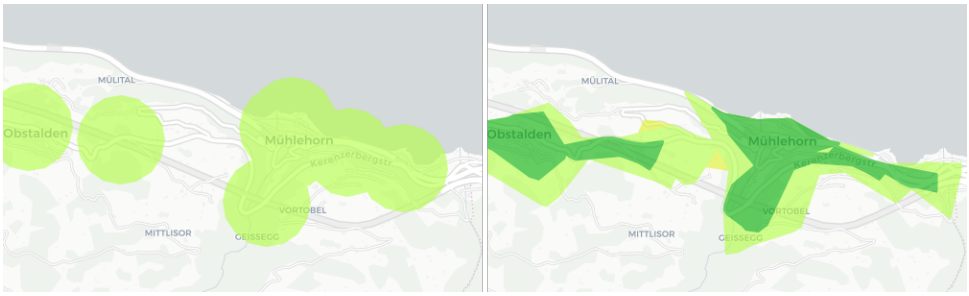

Abbildung 3: Vergleich zur Einfluss der Wegführung. Oben: ÖV-Güteklassen von ARE. Unten: ÖV-Güteklassen 2018.

Ausblick

In der vorliegenden Arbeit wurde beim Berechnen der Erreichbarkeit der Fokus auf eine routenbasierte Methode, namentlich Isochronen, gelegt. Ein weiterer spannender Ansatz, welcher in Betracht gezogen werden kann, ist ein grid- beziehungsweise raster-basierter Ansatz. In Abbildung 4 wurde dieser Ansatz für die Berechnung der Reisezeit zu einer Haltestelle eines Fussgängers verwendet. Dabei wird über ein Gebiet ein Raster gelegt und den Rastern eine Kategorie mit zugehörigen Kosten zugewiesen. Bei den Kosten handelt es sich im abgebildeten Fall um die Überwindungsgeschwindigkeit des jeweiligen Rasters. So erhält ein Raster mit einem Hindernis beispielsweise die Geschwindigkeit 0 km/h und eine Grünfläche 2 km/h. Dies hat unter anderem den Vorteil, dass man nicht direkt vom Routing-Netzwerk abhängig ist und auch offene Flächen wie Plätze und Grünflächen einbezogen werden.

Es lässt sich darüber streiten, ob alle Grünflächen von Fussgänger begehbar sind. Ebenfalls lassen sich offene Fläche für die bestehende Lösung durch eine Vorverarbeitung des Routing-Netzwerk beispielsweise durch *PlazaRoute* [2] ins Routing-Netzwerk integrieren.

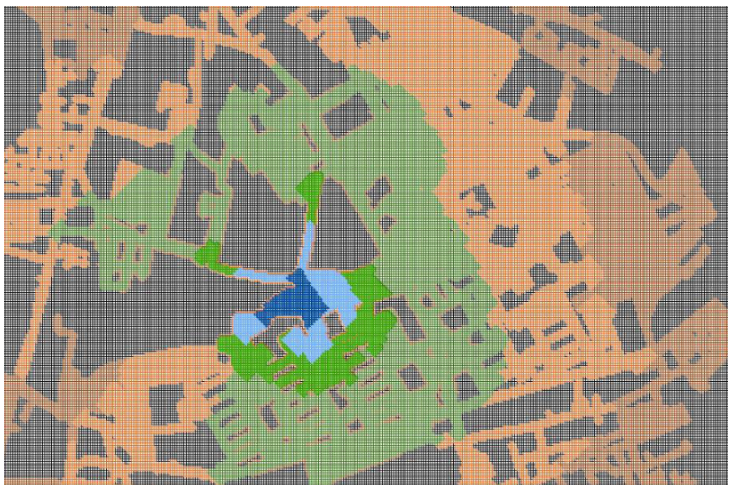

Abbildung 4: Grid- beziehungsweise raster-basierter Ansatz [3]

Inhaltsverzeichnis

I Technischer Bericht

1 Einführung

Diese Bachelorarbeit befasst sich mit der Analyse der Qualität der Erschliessung eines Standortes mit dem öffentlichen Verkehr und deren automatisierten Berechnung. Im Folgenden ist die Problemstellung erläutert und es wird eine Abgrenzung gezogen, was Teil dieser Arbeit ist.

1.1 Grundlagen und Begriffe

Nachfolgend werden Grundlagen und Begriffe eingeführt, welche für das grundlegende Verständnis der Arbeit relevant sind. Für zusätzliche Definitionen kann das Glossar zur Hand gezogen werden.

1.1.1 ÖV-Güteklasse

Definition
„ÖV-Güteklassen geben Auskunft darüber, wie gut ein Standort mit dem öffentlichen Verkehr erschlossen ist. Dies ist wichtig, wenn es darum geht, den öffentlichen Verkehr zu optimieren, Siedlungsverdichtung nach Innen an geeigneten Lagen voranzutreiben oder Standortentscheide für publikumsintensive Anlagen so zu treffen, dass sie möglichst wenig zusätzlichen Autoverkehr verursachen.“ [4]

OeVGK93
OeVGK93 steht kurz für ÖV-Güteklassen 93 und bezeichnet die Definition der ÖV-Güteklassen, welche im Jahre 1993 mit der SN 640 290 [5] verabschiedet wurde.

OeVGKARE
OeVGKARE steht kurz für ÖV-Güteklassen des Bundesamt für Raumentwicklung und bezeichnet die Spezifikation und Umsetzung der ÖV-Güteklassen basierend auf OeVGK93 mit Erweiterungen.

OeVGK18
OeVGK steht kurz für ÖV-Güteklassen 2018 und bezeichnet die neue Spezifikation der ÖV-Güteklassen, welche im Zuge dieser Arbeit erarbeitet wird.

1.2 Problemstellung

Mit ÖV-Güteklassen wird ermittelt, welche Gebiete wie gut mit öffentlichen Verkehrsmitteln erschlossen sind. Die Daten werden unter anderem von Raum- und Verkehrsplanern verwendet, um Entscheidungen bezüglich der Standortentwicklung zu treffen. Gemäss dem Raumplanungsgesetz ist eine Siedlungsentwicklung nach innen vorgeschrieben [6]. So soll verdichtet werden, bevor neues Bauland erschlossen wird.

Die Methodik zur Erhebung von ÖV-Güteklassen wurde 1993 erstmals in der Schweizer Norm (SN) 640 290 [5] festgelegt. Die Norm wurde damals zur Berechnung des Bedarfs an Parkplätzen eingeführt und 2006 durch die neue SN 640 281 [7] ersetzt. Darin sind ÖV-Güteklassen in dieser Form nicht mehr erhalten. Die alte Norm wird aber immer noch als Grundlage verwendet, um die aktuellen ÖV-Güteklassen zu berechnen [1].

Die Berechnungsmethodik, wie sie in der Norm festgehalten wurden, ist für die heutigen technischen Möglichkeiten überholt. Im Folgenden werden einige Probleme der Berechnung von Güteklassen im Bezug auf öffentliche Verkehrsmittel basierend auf der SN 640 290 [5] aufgezeigt. Diese Liste hat nicht den Anspruch, abschliessend und vollständig zu sein, da die Analyse der Probleme und Erarbeitung von Verbesserungen Teil dieser Arbeit ist und im Kapitel 2 aufgeschlüsselt wird. Sie verfolgt primär das Ziel, dem Leser einen Überblick über die zu untersuchenden Konzepte der bestehenden Berechnungsmethodik zu geben und den Handlungsbedarf aufzuzeigen.

1.2.1 Luftlinie bei Einzugsgebieten

Bei der Berechnung von ÖV-Güteklassen nach der SN 640 290 [5] werden um jede Haltestelle Kreise mit einem gewissen Radius gezogen. Diese bestimmen die Einzugsgebiete der Haltestellen mit unterschiedlicher Qualität. Diese Methode beachtet aber nicht die effektive Erreichbarkeit für Fussgänger. Es kann sein, dass die Haltestelle nur durch eine Strasse zugänglich ist und Fussgänger in der Umgebung einen Umweg laufen müssen. Das effektive Einzugsgebiet wird so also nicht akkurat repräsentiert.

1.2.2 Topographie

In der jetzigen Berechnung der ÖV-Güteklassen wird die Topographie der Umgebung von Haltestellen nicht konsequent einberechnet und nur wage definiert, wie weit die Berücksichtigung dieser gehen sollte. Je steiler der Zugang zu einer Haltestelle ist, desto kleiner wird das effektive Einzugsgebiet der Haltestelle, da Fussgänger weniger bereit sind, Distanzen zu laufen, die auf ebenem Grund akzeptabel sind.

1.2.3 Taktberechnung

Nach der verwendeten SN 640 290 [5] werden Haltestellen anhand der Anzahl der Verbindungen in einer bestimmten Zeit in Kategorien eingeteilt. Je höher diese Kategorie, desto grösser wird das Einzugsgebiet der Haltestelle bewertet.

Bei der Einteilung in die Kategorie wird aber nicht darauf geachtet, in welchem Takt die Verbindungen eintreffen. So kann es sein, dass an einer Haltestelle alle 15 Minuten genau ein Bus ankommt und bei einer anderen alle 30 Minuten gleich zwei Busse in gleicher Richtung hintereinander. Die Berechnung zählt aber nur die Anzahl Verbindungen insgesamt, womit beide Haltestellen danach gleichwertig wären, obwohl die Haltestelle mit dem 15 Minuten Takt offensichtlich einen besseren Takt hat.

In der Kategorisierung der Norm werden ausserdem Bushaltestellen, die halbstündlich oder stündlich bedient werden, in die gleiche Kategorie eingeteilt. Für ein Benutzer der Haltestelle ist dies allerdings ein markanter Unterschied.

1.2.4 Berechnung an einem Stichtag

Bei der Berechnung von ÖV-Güteklassen werden Verbindungen in einem gewissen Zeitraum berücksichtigt. Dabei werden nur Verbindungen zwischen 6.00 Uhr und 20.00 Uhr gezählt. Die Ansprüche an die ÖV-Güteklassen können aber auch andere Zeiten umfassen. So kann es interessant sein, die Verbindungen in der Nacht oder am Wochenende einzubeziehen.

1.2.5 Verschiedene Berechnungsmethodiken

Das Bundesamt für Raumentwicklung hat 2011 die Berechnungsmethodik ARE [1] herausgegeben, welche die ÖV-Güteklassen in einem automatisierten Prozess aus den Daten des elektronischen Fahrplans HAFAS [8] berechnet. Diese Berechnung basiert auf der SN 640 290 [5]. Anzumerken ist, dass Anpassungen an der Berechnungsmethodik der Norm vorgenommen werden, um eine automatisierte Berechnung mit verfügbaren Datenbeständen durchführen zu können.

Schwächen der SN 640 290 [5] und der Berechnungsmethodik ARE und regionale Einflüsse haben unterschiedliche Kantone dazu bewogen, Anpassungen vorzunehmen. Einige Kantone erweitern die Berechnungsmethodik des ARE, andere Kantone wiederum passen ihre Berechnungsmethodik basierend auf der SN 640 290 an. Dies ist ein starkes Indiz, dass der Wunsch nach einer einheitlichen Berechnungsmethodik existiert, welche national angewendet werden kann.

1.3 Vision

ÖV-Güteklassen 2018 ermöglicht es unterschiedlichen Stakeholdern die Qualität der öffentlichen Erschliessung eines Standortes unter Berücksichtigung des Terrainmodells und der Streckenführung in Kombination mit Verbesserungen und Erfahrungen der bestehenden Berechnungsmethodiken zu analysieren, um so Entscheidungen, welche den öffentlichen Raum betreffen, fundiert treffen zu können, sowie der allgemeinen Standortsfindung beistehen zu können.

1.4 Ziele und Unterziele

Nachfolgend sind die Ziele der Arbeit definiert. Dadurch ist ersichtlich, was Inhalt der Arbeit ist und wo Abgrenzungen gezogen werden.

1.4.1 Spezifikation OeVGK18

Es ist eine neue Spezifikation der Berechnungsmethodik von ÖV-Güteklassen mit dem Namen OeVGK18 zu definieren, welche auf den bestehenden Berechnungsmethodiken (siehe Abschnitt *Analyse bestehender Berechnungsmethodiken*) basiert und bei Bedarf optimiert (siehe Abschnitt *Evaluierung grundlegender Verbesserungen*). Dabei wir der Fokus darauf gelegt, dass mit einer verfügbaren Datenbasis gearbeitet werden kann (siehe Abschnitt *Abstimmung der Spezifikation mit verfügbaren Daten*).

Analyse bestehender Berechnungsmethodiken

Es existieren unterschiedliche Berechnungsmethodiken, welche ÖV-Güteklassen entweder auf nationaler oder kantonaler Ebene berechnen. Diese Methodiken berücksichtigen unterschiedliche Interessen und basieren auf verschiedenen Grundlagen.

In dieser Arbeit gilt es, bestehende Lösungen zu analysieren. Diese Analyse verfolgt das Ziel, eine Spezifikation zu entwickeln, welche einen Konsens auf nationaler Ebene erreichen kann. Darum sollen Erfahrungen und gemachte Verbesserungen von bestehenden Umsetzungen in die neue Spezifikation OeVGK18 fliessen.

Evaluierung grundlegender Verbesserungen

Die neue Spezifikation soll nicht nur eine reine kombinierte Sammlung von Verbesserungen und Erfahrungen aus bestehenden Lösungen sein, sondern auch eigene Optimierungen einbringen, wo es Sinn macht. Nachfolgend sind mögliche Verbesserungen beschrieben, welche auf eine Verwendung

geprüft werden soll, sei dies konzeptionell, aber auch technisch. Diese Liste ist nicht abschliessend, sondern umfasst den zu Beginn der Arbeit identifizierten Optimierungsbedarf.

Luftlinie bei Einzugsgebieten

Statt um jede ÖV-Haltestelle Umkreise mit fixem Radius zu ziehen, bietet es sich an, das effektive Einzugsgebiet zu analysieren, das ein Fussgänger durch Benutzung von Strassen und Gehwege von der Haltestelle aus erreichen kann.

Topographie

Um die Topographie bei der Berechnung vom Einzugsgebiet einer Haltestelle mit einzubeziehen, müssen die Höhenunterschiede beachtet werden. Zusätzlich zur Laufdistanz für einen Fussgänger werden die Höhenunterschiede in die Berechnung einbezogen und entsprechend gewichtet. Eine Haltestelle in einem steilen Gebiet hat so ein geringeres Einzugsgebiet als eine gleich gut angeschlossene Haltestelle in einem flachen Gebiet.

Ein digitales Terrainmodell der Schweiz für diesen Zweck wird von Swisstopo als swissALTI^{3D} angeboten. [9]

Taktberechnung

In der aktuellen Norm werden nur die Anzahl Verbindungen in einer gewissen Zeitspanne gezählt. Wie in 1.2.3 beschrieben, ergibt dies kein genaues Bild der Taktfrequenz der Verbindungen einer Haltestelle.

Es ist eine optimierte Methoden der Traktberechnung zu entwickeln, welche diese Problematik behebt.

Berechnung an einem Stichtag

Da die Fahrplandaten für das ganze Jahr zur Verfügung stehen und die Berechnung automatisiert erfolgen kann, spricht nichts dagegen, die ÖV-Güteklassen für mehrere Stichtage und Zeitintervalle zu ermitteln.

Abstimmung der Spezifikation mit verfügbaren Daten

Bei der Definition der OeVGK18-Spezifikation liegt der Fokus auf einer möglichst einfachen Verwendung. So soll die Spezifikation auf einem bereits verfügbaren Datenbestand aufsetzen. Dieses Ziel wird bei der Definition berücksichtigt, so dass nicht beispielsweise Transportmittelkategorien festgelegt werden, welche in den Fahrplandaten nicht ausgewiesen sind. Dies hat den Vorteil, dass die Spezifikation allgemeingültig eingesetzt werden kann.

1.4.2 Prototyp und Deliverables

Das Resultat der Arbeit besteht aus zwei Teilen. Für den ersten Teil wird die Spezifikation OeVGK18 definiert, die auf der Norm SN 640 290 [5] sowie der aktuellen Berechnungsmethodik des ARE [1] aufsetzt und diese unter anderem mit den oben genannten Punkten erweitert.

Im zweiten Teil wird die erarbeitete Spezifikation umgesetzt. Mit einem Generator werden die ÖV-Güteklassen basierend auf OeVGK18 für mehrere Parameter berechnet. In einem weiteren Schritt wird das Ergebnis der Berechnungen in einer Web-Applikation visualisiert. Darin können neben unseren Berechnungen auch die bisherigen ÖV-Güteklassen (Berechnungsmethodik ARE) angezeigt werden.

1.5 Rahmenbedingungen, Umfeld, Definitionen und Abgrenzungen

Die Arbeit befasst sich mit der Spezifikation und Berechnung der „ÖV-Güteklassen 2018“. Eine Analyse und ein quantitativer Vergleich mit der bisherigen Spezifikation OeVGK93 wird dabei bewusst ausgeklammert.

Für die Berechnung der ÖV-Güteklassen wird Python mit PostGIS und der Erweiterung pgRouting verwendet, sofern dies mit diesen Technologien möglich ist. Für die Web-Applikation stehen Vue.js oder React als Optionen zur Verfügung. Die Kartendaten werden von OpenStreetMap (OSM) bezogen, die Fahrplandaten werden uns von geOps und der SBB via der Open Data Platform [10] zur Verfügung gestellt. Für das Terrainmodell wird der Datensatz *swissALTI*^{3D} von Swisstopo [9] genutzt, welcher uns zu diesem Zweck von der HSR bereit gestellt wird.

1.6 Raison d'Être ÖV-Güteklassen

Die Verwendung von Güteklassen für die Beurteilung der Qualität der ÖV-Erschliessung erhält ihre Legitimation nicht von ungefähr. Im Grundlagenbericht des UVEK [11] über die „Normierte gesamtverkehrliche Erschliessungsqualitäten“ ist festgehalten, dass sich grundsätzlich drei methodische Ansätze zur Herleitung einer gesamtverkehrlichen Erschliessungsqualität unterscheiden lassen, ein modellbasierter, ein kategorialer und ein indikatorenbasierter Ansatz. Es wird das Fazit gezogen, dass mit dem kategorialen Ansatz, zu welchem die ÖV-Güteklassen gehören, sich inhaltliche Ansprüche am besten mit den Ansprüchen der Nachvollziehbarkeit in der Planungspraxis und Kommunizierbarkeit kombinieren lassen. Der kategorialen Ansatz ist demnach sehr praxisorientiert, transparent, sowie in der Branche etabliert.

1.7 Vorgehen und Aufbau der Arbeit

Diese Bachelorarbeit befasst sich mit dem Problem, eine den heuten technischen Möglichkeiten entsprechende ÖV-Güteklassen-Spezifikation (OeVGK18) und deren Berechnung umzusetzen. Die Arbeit wird somit in zwei von einander abhängigen Phasen umgesetzt. Die erste Phase ist theoretisch gestaltet. Dieser wird grosse Bedeutung beigemessen, da die zweite Phase auf den Ergebnissen dieser aufbaut. So verfolgt die erste Phase das Ziel, eine ÖV-Güteklassen 2018 Spezifikation zu erstellen, mit dem Fokus, schweizweit Akzeptanz zu erhalten. Dies hat die Implikation, dass bestehende Lösungen analysiert werden müssen, um so einen Konsens erreichen zu können. Bewährtes und Akzeptiertes wird kritisch hinterfragt.

In einem zweiten Teil wird die erarbeitete Spezifikation implementiert und die ÖV-Güteklassen für die gesamte Schweiz berechnet. Die Visualisierung erfolgt in Form einer Web-Applikation, worin die ÖV-Güteklassen auf einer interaktiven Karte dargestellt sind.

Die Erarbeitung sowie das Ergebnis der Spezifikation OeVGK18 wird im Teil I beschrieben. Die Berechnung und Visualisierung der Spezifikation wird im Teil II behandelt.

2 Stand der Technik

2.1 Aktuelle Situation

2.1.1 Schweizer Norm 640 290

ÖV-Güteklassen wurden erstmals mit der SN 640 290 [5] des Vereins Schweizerischer Strassenfachleute (VSS) im Jahre 1993 für die Schweiz definiert. Diese Norm enthält Richtwerte für die Bestimmung des Grenzbedarfs an Parkfeldern für Personenwagen und führte hierfür die Definition der ÖV-Güteklassen ein. Im Jahre 2006 wurde diese durch die SN 640 281 abgelöst, welche nicht mehr auf die ÖV-Güteklassen eingeht. Es wurde seither keine allgemeingültige Definition verabschiedet.

Im Folgenden ist aufgeschlüsselt, was genau die SN 640 290 [5] in Bezug auf ÖV-Güteklassen definiert.

Definition ÖV-Güteklassen

Grundsätzlich definieren ÖV-Güteklassen die Qualität der Erschliessung durch den öffentlichen Verkehr. Diese Qualität wird durch die Bedienungsqualität der Haltestelle und die Erreichbarkeit der Haltestelle festgesetzt. Beide Kriterien sind folgend definiert. Anmerken muss man, dass zur besseren Verständlichkeit und Übersichtlichkeit zusätzliche Terminologie wie Verkehrsmittelkategorie eingeführt wird.

Bedienungsqualität der Haltestelle

Die öffentlichen Verkehrsmittel werden zuerst in Verkehrsmittelkategorien eingeteilt:

- Verkehrsmittelkategorie 1
  - Bahnknoten: mehrere Bahnlinien in unterschiedliche Richtungen
- Verkehrsmittelkategorie 2
  - Bahnlinie
- Verkehrsmittelkategorie 3
  - Tram, Trolleybus, Autobus, Regionalbus, Ortsbus mit gutem Anschluss an Bahnlinie
- Verkehrsmittelkategorie 4
  - Ortsbus, lokaler Kleinbus

Die Verkehrsmittelkategorien werden dabei in die Gruppe A und B zusammengefasst, welche für die Berechnung des durchschnittlichen Kursintervalls berücksichtigt werden. Dabei werden für die Intervallberechnung alle Verkehrsmittel der Verkehrsmittelkategorie in einer Gruppe berücksichtigt. Daraus lassen sich die Haltestellenkategorien ermitteln (siehe Tabelle 1).

Kursintervall	**Verkehrsmittelkategorie**			
	Gruppe A		**Gruppe B**	
	1	**2**	**3**	**4**
< 5 min	I	I	II	III
5 – 9 min	I	II	III	IV
10 – 19 min	II	III	IV	V
20 – 39 min	III	IV	V	V
40 – 60 min	IV	V	V	-

Tabelle 1: Ermittlung der Haltestellenkategorie (SN 640 290)

Das Kursintervall ist der durchschnittliche Abstand zwischen Ankunft beziehungsweise Abfahrt (pro Linie jeweils in der Hauptlastrichtung) aller Verkehrsmittel in der gleichen Gruppe zwischen 06:00 – 20:00 Uhr jeweils von Montag bis Freitag. Die Norm macht hier Ausnahmen bei Verdichtungen in Hauptverkehrszeiten, Linienüberlagerungen und reinen Arbeitsplatzgebieten mit stark verdichtetem Fahrplan während Pendlerzeiten.

Erreichbarkeit der Haltestelle

Basierend auf den vorhin definierten Haltestellenkategorien (I bis V) kann die Erreichbarkeit pro Haltestellenkategorien definiert werden. Die Distanzen sind Luftlinien, wobei ein mittlerer Umwegfaktor von 20 bis 30% berücksichtigt wird. Daraus lassen sich die ÖV-Güteklassen ermitteln (siehe Tabelle 2)

Der Topographie wird Rechnung getragen, indem man bei schwierigen Gegebenheiten die nächste Klasse wählt oder die Distanz vergrössert.

Haltestellenkategorie	Distanz zur Haltestelle			
	< 300 m	**300 – 500 m**	**501 – 750 m**	**751 – 1000m**
I	Klasse A	Klasse A	Klasse B	Klasse C
II	Klasse A	Klasse B	Klasse C	Klasse D
III	Klasse B	Klasse C	Klasse D	-
IV	Klasse C	Klasse D	-	-
V	Klasse D	-	-	-

Tabelle 2: Ermittlung der Erreichbarkeit der Haltestelle (SN 640 290)

2.2 Lösungsansätze

Basierend auf der bestehenden Schweizer Norm 640 290 wurde im Jahre 2011 die Berechnungsmethodik des ARE veröffentlicht, welche in einer Weisung des UVEK vom 16.02.2015 als die Berechnugsmethodik definiert wird, welche bei der Beurteilung der Qualität der Erschliessung mit dem öffentlichen Verkehr verwendet werden soll.

In der Zwischenzeit haben einige Kantone Eigenlösungen entwickelt, welche Anpassungen an dieser Methodik und der Schweizer Norm 640 290 vornehmen. Folgend werden grob die Abweichungen zur Norm der Berechnungsmethodik ARE und zweier kantonaler Methodiken beschrieben, namentlich des Kanton Graubünden und Zürich. Ebenfalls wird ein Blick auf eine österreichische Umsetzung der ÖV-Güteklassen geworfen.

Im Anschluss werden die Methodiken zusammen betrachtet, Schwachstellen eruiert und Verbesserungen vorgeschlagen. Dieses Vorgehen hat das Ziel, Erfahrungen und etablierte Grundlagen in OeVGK18 im Kapitel 3 einfliessen zu lassen, um so eine allgemeine Akzeptanz erreichen zu können.

2.2.1 Berechnungsmethodik ARE

Das Bundesamt für Raumentwicklung (ARE) entwickelt in ihrem Grundlagenbericht für die Beurteilung der Agglomerationsprogramme Verkehr und Siedlung [1] eine Berechnungsmethodik der ÖV-Güteklassen basierend auf der bestehende Norm (siehe Kapitel 2.1.1). Die entwickelte Berechnungsmethodik basiert auf den Fahrplandaten von HAFAS [8]. Dies hatte die Konsequenz, dass die Berechnungsmethodik so wie sie in der Norm festgelegt ist, angepasst werden musste. Im Folgenden sind die Anpassungen aufgeführt.

Bedienungsqualität der Haltestelle

Die Verkehrsmittel lassen sich aus dem elektronischen Fahrplan [8] in der geforderten Tiefe nicht unterscheiden, wie es die Norm verlangt. Dies hat die Konsequenz, dass die folgende Verkehrsmittel-Einteilung verwendet wird:

- Verkehrsmittelgruppe A
  - Bahnknoten (mehrere Bahnlinien in verschiedenen Richtungen)
  - Bahnlinien
- Verkehrsmittelgruppe B
  - Tram, Busse, Postautos, Rufbusse oder Schiffe
- Verkehrsmittelgruppe C
  - Seilbahnen

Die Hauptlastrichtung kann aus den Fahrplandaten nicht extrahiert werden. Somit werden alle Abfahrten zwischen 6.00 und 20.00 Uhr gezählt und anschliessend halbiert. Bei Endhaltestellen und Linien, welche nur in eine Richtung verlaufen, erfolgen Korrekturen. Die genaue Art der Korrektur ist nicht weiter aufgeführt. Als Stichtag für die Auswertung wird ein Werktag ausserhalb der Ferienzeit und der touristischen Hochsaison definiert. Ebenfalls wir in reinen Arbeitsplatzgebieten mangels einheitlicher Definition keine Anpassung vorgenommen. Verdichtungen in den Hauptverkehrszeiten sind in den gezählten Abfahrten inbegriffen.

Erreichbarkeit der Haltestelle

Der Topographie wird keine Rechnung getragen, um eine landesweite Vergleichbarkeit sicherzustellen.

Die ÖV-Güteklassen sind folgend zu interpretieren:

- Güteklasse A: Sehr gute Erschliessung
- Güteklasse B: Gute Erschliessung
- Güteklasse B: Mittelmässige Erschliessung
- Güteklasse D: Geringe Erschliessung
- Keine Güteklasse: Marginale oder keine ÖV-Erschliessung

2.2.2 Berechnungsmethodik Kanton Graubünden

Im technischen Bericht „Definition ÖV-Struktur / Erhebung ÖV-Güteklassen Kanton Graubünden“ [12] werden Anpassungen der in der Norm beschriebenen Berechnungsmethodik der ÖV-Güteklassen an die speziellen Bedürfnisse des Kanton Graubündens, aber auch von allgemeiner Natur vorgenommen. Anpassungen, welche spezielle Gegebenheiten des Kanton Graubündens betreffen, werden in der folgenden Zusammenstellung ausgeklammert, da sie sich für eine automatische Berechnung und für eine nationale Betrachtung nicht eignen.

Bedienungsqualität der Haltestelle

Es werden zwei zusätzliche Haltestellenkategorien VI und VII (siehe Tabelle 3) ergänzt und eine feinere Gliederung der Kursintervalls eingeführt.

Auf die Unterscheidung zwischen „Tram, Trolleybus, Autobus, Regionalbus“ und „Ortsbus, lokaler Kleinbus“ wird verzichtet und in der ersteren Kategorie zusammengefasst. Ebenfalls werden Seilbahnen mit Erschliessungsfunktion wie Bushaltestellen behandelt.

Kursintervall	**Art des Verkehrsmittels**		
	Bahnknoten	**Bahnlinie**	**Tram/Bus**
< 5 min	I	I	II
5 – 10 min	I	II	III
11 – 19 min	II	III	IV
20 – 39 min	III	IV	V
40 – 60 min	IV	V	VI
> 60 min	-	VII	VII

Tabelle 3: Ermittlung der Haltestellenkategorie (Kanton Graubünden)

Ebenfalls werden nun Bushaltestellen, die halbstündlich und stündlich bedient werden, nicht mehr gleich kategorisiert. Auch werden Haltestellen erfasst, welche seltener als stündlich einen Anschluss haben. Durch die feinere Gliederung des Kursintervalls erhält man eine bessere Güteklasse bei einem 10-Minuten-Takt im Vergleich mit einem 15-Minuten-Takt.

Erreichbarkeit der Haltestelle

Die zusätzlichen Haltestellenkategorien (VI und VII) haben zur Folge, dass auch zwei zusätzliche Güteklassen E und F eingeführt werden (siehe Tabelle 4).

Haltestellenkategorie	**Distanz zur Haltestelle**			
	< 300 m	**300 – 500 m**	**501 – 750 m**	**751 – 1000m**
I	Klasse A	Klasse A	Klasse B	Klasse C
II	Klasse A	Klasse B	Klasse C	Klasse D
III	Klasse B	Klasse C	Klasse D	-
IV	Klasse C	Klasse D	-	-
V	Klasse D	-	-	-
VI	Klasse E	-	-	-
VII	Klasse F	-	-	-

Tabelle 4: Ermittlung der Erreichbarkeit der Haltestelle (Kanton Graubünden)

Bushaltestellen haben eine maximale Erschliessungswirkung von 500 Meter, da grössere Distanzen von Passagieren als zu lange empfunden wird. Somit werden nur Bahnknoten und Bahnlinien in den Kategorien > 500 Meter klassifiziert. Die Topographie wird manuell berücksichtigt.

Hauptverkehrszeiten werden nicht gesondert gehandhabt. Es zählt grundsätzlich das Angebot im gegebenen Zeitbereich.

Linien, welche grundsätzlich stündlich fahren, jedoch Taktlücken aufweisen um zu einen gewissen Zeitpunkt ein Angebot erfüllen zu können, werden nur in der Stundentakt-Gruppe klassifiziert, wenn sie nicht mehr als 2 Taktlücken aufweisen.

Es existieren Linienüberlagerungen mit einer ungleichen Verteilung. So können zwei Linien stündlich verkehren und eine Haltestelle kurz nacheinander bedienen, welche dann als Halbstundentakt klassifiziert werden. Der Passagier nimmt das Angebot aber als Stundentakt wahr. Jedoch ist nicht weiter ersichtlich, wie das Problem behoben wird.

Die Bestimmung der Hauptlastrichtung erfolgt analog zur Berechnungsmethodik ARE.

2.2.3 Berechnungsmethodik Kanton Zürich

Im Infoblatt „ÖV-Güteklassen“ [13] wird beschrieben, wie der Kanton Zürich die kantonalen ÖV-Güteklassen mit den Fahrplandaten des Zürcher Verkehrsverbundes (ZVV), angelehnt an die Berechnungsmethodik ARE, berechnet.

Bedienungsqualität der Haltestelle

Es werden nur Haltestellen berücksichtigt, die von Bahn, Tram und/oder Bus bedient werden. Seilbahnen und Schiffe werden explizit ausgeklammert. Somit ergibt sich folgende Kategorisierung:

- Bahnknoten (Bahnstationen mit S-Bahnlinien in min. 6 Richtungen und/oder IR-Anschluss)
- Bahnlinien
- Tram
- Bus

Der Kursintervall wird auf 05:30 bis 22:30 Uhr festgelegt (unter der Woche). Handelt es sich um Endhaltestellen oder Linien, welche nur in eine Richtung verkehren, werden alle Abfahrten gezählt. Zusätzlich werden wie im Kanton Graubünden Haltestellen mit einem Kursintervall von über 60 Minuten berücksichtigt (siehe Tabelle 5).

Kursintervall	**Art des Verkehrsmittels**			
	Bahnknoten	**Bahnlinie**	**Tram**	**Bus**
< 5 min	I	I	II	II
5 – 9 min	I	II	III	III
10 – 19 min	II	III	IV	IV
20 – 39 min	III	IV	-	V
40 – 60 min	-	V	-	VI
> 60 min	-	-	-	VII

Tabelle 5: Ermittlung der Haltestellenkategorie (Kanton Zürich)

Erreichbarkeit der Haltestelle

Auch im Kanton Zürich werden durch den zusätzlich abgebildeten Kursintervall (> 60 Minuten) zwei neue Güteklassen E und F eingeführt (siehe Tabelle 6).

Haltestellenkategorie	Distanz zur Haltestelle			
	< 300 m	**300 – 500 m**	**501 – 750 m**	**751 – 1000m**
I	Klasse A	Klasse A	Klasse B	Klasse C
II	Klasse A	Klasse B	Klasse C	Klasse D
III	Klasse B	Klasse C	Klasse D	-
IV	Klasse C	Klasse D	Klasse E	-
V	Klasse D	Klasse E	Klasse E	-
VI	Klasse E	Klasse E	-	-
VII	Klasse F	Klasse F	-	-

Tabelle 6: Ermittlung der Erreichbarkeit der Haltestelle (Kanton Zürich)

Der Topographie wird keine Rechnung getragen, es wird strikt auf die Luftlinie gesetzt. Sind Haltestellen nur durch Tram und/oder Bus erschlossen, werden analog zum Kanton Graubünden nur Distanzen bis zu 500 Meter berücksichtigt. Die Erschliessung durch Bahnknoten und Bahnlinien wird bis 750 Meter mit den Güteklassen B – E und bis 1000 Meter mit den Güteklassen C –D berücksichtigt. Diese Entscheidungen basieren auf ÖV-Angebotsverordnungen.

2.2.4 Berechnungsmethodik Österreich

Im Artikel „ÖV-Güteklassen – ein Werkzeug zur Analyse der Versorgung eines Standortes mit ÖV“ [14] ist die Umsetzung der Analyse der Versorgung eines Standortes mit öffentlichem Verkehr in Österreich beschrieben.

Bedienungsqualität der Haltestelle

Die Kursintervallberechnung erfolg analog zur Berechnungsmethodik ARE. Herauszuheben ist, dass zwei Stichtage definiert werden, ein gewöhnlicher Werktag und einen Tag, welcher ein besonders ungünstigen Verkehrstag wiederspiegelt.

Es wird keine grundsätzliche Unterscheidung zwischen Bahnknoten und Bahnlinien gezogen, sondern der Fernverkehr wird als das beste mögliche Verkehrsmittel angesehen.

Es werden 3 weitere Haltestellenkategorien eingeführt und eine erweiterte Gliederung der Kursintervalle vorgenommen (siehe Tabelle 7). So fliegen Haltestellen mit einem Intervall grösser 210 Minuten grundsätzlich bereits aus der Analyse.

Kursintervall	Art des Verkehrsmittels			
	Fernverkehr REX	**S-Bahn, U-Bahn, ...**	**Strassenbahn, Metrobus, ...**	**Bus**
< 5 min	I	I	II	III
5 – 9 min	I	II	III	III
10 – 19 min	II	III	IV	IV
20 – 39 min	III	IV	V	V
40 – 60 min	IV	V	VI	VI
60 – 120 min	V	VI	VII	VII
121 – 210 min	-	VII	VIII	VIII

Tabelle 7: Ermittlung der Haltestellenkategorie (Österreich)

Erreichbarkeit der Haltestelle

Durch die zusätzlichen Haltestellenkategorien und der feineren Kursintervallgliederung werden drei neue Güteklassen eingeführt, namentlich E, F und G. Die Evaluierung der ÖV-Güteklassen ist in Tabelle 8 ersichtlich.

Herauszuheben ist, dass die Wegführung berücksichtigt wird, was in den bisher analysierten Berechnungsmethodiken verwehrt blieb. Von einer Haltestelle aus werden die zu Fuss bewältigbaren Wege ermittelt.

Zusätzlich werden die gebildeten ÖV-Güteklasse-Polygone mit einem Raster, welches statistische Daten beinhaltet, verschnitten. So kann man beispielsweise zusätzlich die Bevölkerungsdichte evaluieren.

Haltestellenkategorie	Distanz zur Haltestelle				
	< 301 m	**301 – 500 m**	**501 – 750 m**	**751 – 1000m**	**1001 – 1250m**
I	Klasse A	Klasse A	Klasse B	Klasse C	Klasse D
II	Klasse A	Klasse B	Klasse C	Klasse D	Klasse E
III	Klasse B	Klasse C	Klasse D	Klasse E	Klasse F
IV	Klasse C	Klasse D	Klasse E	Klasse F	Klasse G
V	Klasse D	Klasse E	Klasse F	Klasse G	Klasse G
VI	Klasse E	Klasse F	Klasse G	-	-
VII	Klasse F	Klasse G	Klasse G	-	-
VIII	Klasse G	Klasse G	-	-	-

Tabelle 8: Ermittlung der Erreichbarkeit der Haltestelle (Österreich)

2.3 Verbesserungsmöglichkeiten

Folgend werden identifizierte Schwachstellen der bestehenden Berechnungsmethodiken (siehe Kapitel 2.2) aufgelistet und mögliche Optimierungen präsentiert. Dabei behält man das Ziel der automatischen Berechnung hinter dem geistigen Auge. Ob die definierten Kriterien mit der vorhandenen Daten-

basis umsetzbar sind und ob diese für eine schweizweite Umsetzung tauglich ist, ist Bestandteil dieser Analyse. Basierend auf diesen Verbesserungsmöglichkeiten, der Berechnungsmethodik ARE und den Learnings der beiden kantonalen Berechnungsmethodiken wird im Kapitel 3 eine neue Methodik vorgeschlagen.

2.3.1 Ermittlung der Erreichbarkeit der Haltestelle

Problem

In allen Lösungen wird die Luftlinie als massgebende Bewertung der Erreichbarkeit einer Haltestelle genommen. Dies berücksichtigt weder die zusätzliche Distanz durch verwinkelte Wege noch die Topographie. Der Kanton Graubünden behebt diese Problematik mit manuellen Anpassungen. Manuelle Eingriffe eigenen sich jedoch für eine automatisierte Lösung nicht.

Lösung

Um der Topographie und der Wegführung gerecht zu werden, wird einerseits der konkrete Weg entlang des Wege- und Strassennetzes für die Distanzberechnung berücksichtigt. Auf der anderen Seite wird die Topographie durch die Berechnung von Leistungskilometern miteinbezogen.

2.3.2 Wahl des Zeitintervalls

Problem

Die Methodiken verwenden zur Berechnung des Kursintervalls grundsätzlich einen Werktag ausserhalb der Ferienzeit und der touristischen Hochsaison und einen Bereich, welcher den Pendlerverkehr zu den Randzeit berücksichtigt. Der Kanton Zürich geht noch einen Schritt weiter und vergrössert ihren Einzugsbereich. Die Kurse am Wochenende und in der Nacht werden grundsätzlich ausgeklammert. Für Privatpersonen aber auch für Planer kann dieser Zeitraum relevant sein, um sich für einen Wohnort zu entscheiden oder um ein Angebot ausbauen zu können.

Lösung

Es werden drei Stichtage spezifiziert. Zusätzlich zum Standardbereich, welcher den Pendlerverkehr berücksichtigt, wird der Verkehr ausserhalb des Pendlerverkehrs am Abend und am Wochenende berücksichtigt. Die Gewichtung wird bei allen Zeitintervallen beibehalten, um einen quantitativen Vergleich zu ermöglichen.

2.3.3 Kursintervall

Problem

Für die Bestimmung des Kursintervalls an einer Haltestelle werden in den bisherigen Berechnungsmethodiken in einem bestimmten Zeitintervall alle Abfahrten gezählt. Das Kursintervall wird bestimmt als Quotient aus Anzahl Abfahrten und der Dauer dieses Zeitintervalls, was den Mittelwert an Abfahrten pro Stunde ergibt. Diese Methode berücksichtigt aber nicht die Verteilung der Abfahrten, was zu starken Verzerrungen führen kann.

Beachten wir als Beispiel eine fiktive Bushaltestelle, an der zu Beginn der Stunde gleich zwei Busse halten, dann aber für den Rest der Stunde kein einziger mehr. Der Mittelwert an Abfahrten an dieser Haltestelle berechnet sich zu zwei Abfahrten pro Stunde, was ein Kursintervall von 30 Minuten ergibt. Dies ist aber offensichtlich nicht ein realistischer Wert. Aus Sicht eines Benutzers dieser Haltestelle handelt es sich eher um ein 60-minütiges Kursintervall.

Lösung

Für die Berechnung des Kursintervall wird statt der Anzahl Abfahrten die Wartezeit zwischen zwei Abfahrten als Grundlage verwendet. Der Kursintervall τ einer Haltestelle wird definiert als das „Doppelte der erwarteten Wartezeit auf die nächste Abfahrt [...] bei zufälligem Zugang im Zeitintervall $I = [a, b)$“. [15]

Da die erwartete Ankunft bei zufälligem Zugang in der Mitte zwischen zwei regulären Abfahrten liegt, ergibt das Doppelte der erwarteten Wartezeit gerade das Kursintervall.

Wird das obige Beispiel mit diesem Ansatz erneut betrachtet, ergibt sich daraus ein Kursintervall von 58 Minuten (siehe Tabelle 9), was ein deutlich realistischeres Bild ergibt.

Berechnungsmethodik	Abfahrtszeiten	Berechneter Kursintervall
Bisherig	09:01, 09:02, 10:01, 10:02, ...	30 Minuten
Mittelwert aus Wartezeit	09:01, 09:02, 10:01, 10:02, ...	58 Minuten

Tabelle 9: Vergleich der Kursintervallberechnung mit bisheriger und neuer Methodik

2.3.4 Bestimmung der Bahnknoten

Problem

Bei der Unterscheidung von Bahnknoten und Bahnlinien ist bei der Berechnungsmethodik vom ARE lediglich spezifiziert, dass ein Bahnknoten Bahnlinien in mehrere Richtungen hat (siehe Kapitel 2.2.1). In den ÖV-Güteklassen des Kanton Zürichs werden Bahnknoten als „Bahnstationen in mindestens 6 Richtungen und/oder nut IR-Anschluss“ (Kapitel 2.2.3) definiert. Bei beiden Methodiken ist nicht klar, was unter einer Richtung genau zu verstehen ist.

Betrachtet man die resultierenden Daten der Berechnungsmethodik des ARE ist ersichtlich, dass beispielsweise Genève-Aéroport als Bahnknoten identifiziert wird. Diese Haltestelle bedient genau eine Richtung nach Genève. Auf den ersten Blick könnte man annehmen, das die Identifizierung als Bahnknoten dadurch zustande kommt, weil die Haltestelle ein Flughafen erschliesst. Dies wäre jedoch auf zwei Ebenen falsch, da in der Spezifikation dies nicht erwähnt wird und beispielsweise die Haltestelle Zürich Flughafen nicht als Bahnknoten identifiziert ist. Anzumerken ist, dass die Haltestelle Zürich Flughafen 5 unterschiedliche Richtungen bedient. Warum nun eine Haltestelle als Bahnknoten durchgeht oder eben nicht, entzieht sich unseren Kenntnissen. Die fehlende Transparenz in der Bestimmung der Bahnknoten ist zu kritisieren.

Lösung

Für die Bestimmung von Bahnknoten werden von einer gegebenen Bahnstation aus diejenigen anderen Bahnstationen gezählt, die mit einem beliebigen Zug ohne Zwischenhalt erreicht werden können. Konkret wird also für jeden (im definierten Zeitraum) an der Bahnstation haltenden Zug ermittelt, zu welcher Station er als nächstes fahren wird. Die Anzahl Richtungen wird dann definiert als die Anzahl von verschiedenen Bahnstationen, die dadurch gezählt werden.

3 Spezifikation OeVGK18

3.1 Zusammenhang zur Berechnungsmethodik ARE

In der Weisung des UVEK vom 16.02.2015 ist festgehalten, dass für die Beurteilung der Qualität der Erschliessung mit dem öffentlichen Verkehr die Berechnungsmethodik ARE verwendet wird [16]. Dies ist der Grund warum im Folgenden der Bogen zu dieser Berechnungsmethodik geschlagen wird und die vorgeschlagenen Änderungen dieser Grundlagen gegenüber gestellt wird. Die Berechnungsmethodik ARE wird im Original [1] zitiert und Anpassungen aus kantonalen Berechnungsmethodiken und eigene Verbesserungen werden, wo es Sinn macht, übernommen und gekennzeichnet.

3.1.1 Art der Verkehrsmittel

Die Art der Verkehrsmittel, die an einer Haltestelle abfahren, wird wie folgt unterschieden:

- *Verkehrsmittelgruppe A*
  - *Bahnknoten (mehrere Bahnlinien in verschiedenen Richtungen)*
  - *Bahnlinien*
- *Verkehrsmittelgruppe B*
  - *Tram, Busse, Postautos, Rufbusse oder Schiffe*
- *Verkehrsmittelgruppe C*
  - *Seilbahnen*

Die Verkehrsmittel werden weiterhin in drei Gruppen gehalten. Aufgrund der neuartigen Berechnung des Kursintervalls sind Bahnknoten und Bahnlinie in separaten Gruppen eingeteilt und Seilbahnen werden mit den restlichen Verkehrsmittel zusammen betrachtet.

- Verkehrsmittelgruppe A
  - Bahnknoten
- Verkehrsmittelgruppe B
  - Bahnlinie
- Verkehrsmittelgruppe C
  - Tram, Bus, Postauto, Rufbus, Schiff, Seilbahn

Als Bahnknoten gelten analog zu Berechnungsmethodik des Kanton Zürich Bahnstationen, die entweder einen Anschluss an den nationalen oder internationalen Fernverkehr haben und/oder Bahnlinien in mindestens 6 Richtungen verkehren.

3.1.2 Kursintervall

> *Als Stichtag für die Auswertung wird ein Werktag ausserhalb der Ferienzeit und der touristischen Hochsaison definiert.*

Die Einschränkung zur Wahl des Stichtags wird übernommen. Jedoch werden 2 weitere Zeiträume definiert, einer für das Wochenende und jeweils einer für die Nacht.

> *Zur Berechnung des Kursintervalls an einer Haltestelle werden aus dem elektronischen Fahrplan die Abfahrten auf allen Linien am Stichtag zwischen 6.00 und 20.00 Uhr gezählt. Um die durchschnittliche Anzahl Abfahrten in eine Richtung zu ermitteln, wird die Anzahl Abfahrten halbiert. Für Endhaltestellen sowie Haltestellen, die nur in einer Richtung befahren werden, erfolgen entsprechende Korrekturen. Anschliessend wird das Kursintervall für die beiden Verkehrsmittelgruppen A und B separat berechnet (840 Minuten geteilt durch die korrigierte Anzahl Abfahrten).*

Für die Kursintervallberechnung wird die statt der Anzahl Abfahrten die Wartezeit zwischen zwei Abfahrten als Grundlage verwendet. Der Kursintervall τ einer Haltestelle wird definiert als das „Doppelte der erwarteten Wartezeit auf die nächste Abfahrt [...] bei zufälligem Zugang im Zeitintervall $I = [a, b)$“ [15].

3.1.3 Haltestellenkategorie

Die Haltestellenkategorie wird nach folgender Tabelle ermittelt:

Kursintervall	***Verkehrsmittelgruppe***			
	A		***B***	***C***
	Bahnknoten	***Bahnlinien***	***Trams, Busse, Postautos, Rufbusse und Schiffe***	***Seilbahnen***
< 5 min	*I*	*I*	*II*	*III*
5 – 9 min	*I*	*II*	*III*	*IV*
10 – 19 min	*II*	*III*	*IV*	*V*
20 – 39 min	*III*	*IV*	*V*	*V*
40 – 60 min	*IV*	*V*	*V*	*V*

Bei der Gliederung des Kursintervalls wird die Einteilung der Berechnungsmethodik des Kanton Graubünden übernommen. Dies führt zu einer feineren und erweiterten Gliederung des Kursintervalls und zwei zusätzlichen Haltestellenkategorien VI und VII.

3.1.4 Distanz zur Haltestelle

Für die Distanz zur Haltestelle wird die Luftliniendistanz verwendet, d.h. die ÖV-Güteklassen bilden konzentrische Kreise um die Haltestelle. Die Radien der Kreise betragen 300 m, 500 m, 750 m und 1'000 m.

Als Einteilung werden neu Gehminuten mit der Einstufung 5, 7.5, 10 und 15 Minuten verwendet. Unter der Annahme, dass die Laufgeschwindigkeit $1.4m/s$ beträgt, wird die massgebende Gehdauer mithilfe der Laufgeschwindigkeit und der Anzahl zurückzulegende Leistungskilometer berechnet, welche die Topographie und die Wegführung berücksichtigt.

3.1.5 ÖV-Güteklassen

Die Güteklassen werden nach folgender Tabelle ermittelt:

Haltestellenkategorie	***Distanz zur Haltestelle***			
	< 300 m	***300 – 500 m***	***501 – 750 m***	***751 – 1000m***
I	*Klasse A*	*Klasse A*	*Klasse B*	*Klasse C*
II	*Klasse A*	*Klasse B*	*Klasse C*	*Klasse D*
III	*Klasse B*	*Klasse C*	*Klasse D*	*keine*
IV	*Klasse C*	*Klasse D*	*keine*	*keine*
V	*Klasse D*	*keine*	*keine*	*keine*

Durch die zusätzlichen Haltestellenkategorien VI und VII werden analog zur Berechnungsmethodik des Kanton Graubünden zwei weitere ÖV-Güteklassen E und F eingeführt.

3.2 Berechnungsmethodik OeVGK18

Folgend wird die Berechnungsmethodik OeVGK18 detailliert beschrieben. In Abbildung 5 ist zur Übersicht die Berechnung der ÖV-Güteklassen schematisch dargestellt. In einem ersten Schritt wird unter Berücksichtigung der Art der Verkehrsmittel (siehe Kapitel 3.2.1) und dem Kursintervall (siehe Kapitel 3.2.2) die Haltestellenkategorie (siehe Kapitel 3.2.3) eruiert. Abschliessend werden mithilfe der Haltestellenkategorie und der Gehzeit zur Haltestelle (siehe Kapitel 3.2.4) die ÖV-Güteklassen (siehe Kapitel 3.2.5) berechnet.

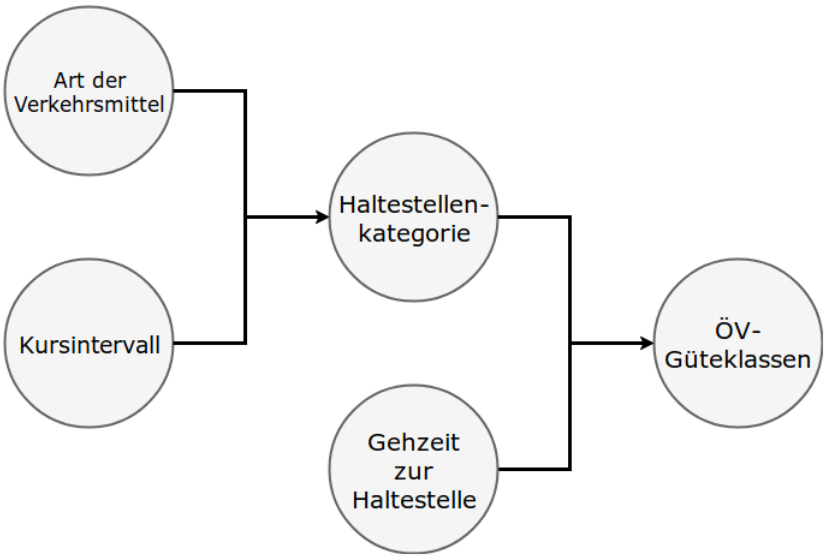

Abbildung 5: Schema ÖV-Güteklassen Berechnung

3.2.1 Art der Verkehrsmittel

Verkehrsmittel werden in folgende Verkehrsmittelgruppen eingeteilt:

- Verkehrsmittelgruppe A
  - Bahnknoten
- Verkehrsmittelgruppe B
  - Bahnlinie
- Verkehrsmittelgruppe C
  - Tram, Bus, Postauto, Rufbus, Schiff, Seilbahn

Bahnknoten

Bahnknoten sind Bahnstationen, die entweder einen Anschluss an den nationalen Fernverkehr haben und/oder Bahnlinien in mindestens 6 Richtungen verkehren.

Die Anzahl Richtungen einer Bahnstation wird gezählt als die Anzahl umliegender Bahnstationen, die mit einem beliebigen Zug innerhalb des definierten Zeitbereichs ohne Zwischenhalt erreicht werden können.

Schiffe und Seilbahnen

Schiffe und Seilbahnen werden nur berücksichtigt, wenn sie ein besiedeltes Wohngebiet erschliessen und nicht ausschliesslich für touristische Zwecke verwendet werden.

3.2.2 Kursintervall

Es sind 3 Stichtage mit jeweils zwei Zeitintervallen zu definieren, welche ausserhalb der Ferienzeit und der touristischen Hochsaison liegen.

Stichtag	Zeitintervall
Werktag	06:00 – 20:00 Uhr
Werktag	20:00 – 00:00 Uhr
Samstag	06:00 – 20:00 Uhr
Samstag	01:00 – 04:00 Uhr
Sonntag	06:00 – 20:00 Uhr
Sonntag	01:00 – 04:00 Uhr

Tabelle 10: Stichtag und Zeitintervall

Der Kursintervall τ einer Haltestelle wird definiert als das „Doppelte der erwarteten Wartezeit auf die nächste Abfahrt [...] bei zufälligem Zugang im Zeitintervall $I = [a, b)$“ [15]. Es werden nur Abfahrten betrachtet, die innerhalb des definiertem Zeitintervall an der Haltestelle abfahren.

Die Umsetzung dafür geschieht wie folgt:

Abfahrtszeitpunkte im Zeitintervall I: $x_{1...n}$
Erste Abfahrt nach Zeitpunkt b (nach Fahrplan): x'
Fiktive Abfahrt nach b bei zyklischer Fortsetzung: $x'' = x_1 + (b - a)$

Für die Wartezeit am Zeitintervallende wird die Fahrt $x_{n+1} = min\{x', x''\}$ verwendet.

Kursintervall:

$$\tau^{a,b} = \frac{1}{b-a} \sum_{i=0}^{n} \Delta_i$$

wobei: $\Delta_i = (x_{i+1} - x_i)^2$ für $i \in \{1, \ldots, n-1\}$
$\Delta_0 = (x_1 - a)^2$
$\Delta_n = (x_{n+1} - x_n)^2 - (x_{n+1} - b)^2$

3.2.3 Haltestellenkategorie

Die Haltestellenkategorie I bis VII wird mit folgender Tabelle eruiert:

Kursintervall [min]	**Verkehrsmittelgruppe**		
	A	**B**	**C**
≤ 5	I	I	II
(5, 10]	I	II	III
(10, 20]	II	III	IV
(20, 40]	III	IV	V
(40, 60]	IV	V	VI
> 60	-	VII	VII

Tabelle 11: Haltestellenkategorie

3.2.4 Gehzeit zur Haltestelle

Bei der Berechnung der Gehzeit zu einer Haltestelle ist eine Laufgeschwindigkeit von $1.4m/s$ anzunehmen und die Strecke entlang des Wege- und Strassennetzes zu berücksichtigen, welche im Folgenden als Horizontaldistanz bezeichnet wird.

Damit man der Topographie gerecht wird, ist als massgebende zu laufende Distanz Leistungsmeter zu verwenden:

$$x = \begin{cases} a + b/0.1 + c/0.15, & \text{wenn } c/d > 0.22 \\ a + b/0.1, & \text{sonst} \end{cases}$$

wobei: $x =$ Leistungsmeter [m]
$a =$ Horizontaldistanz [m]
$b =$ positive Steigung in Höhenmeter [m]
$c =$ negative Steigung in Höhenmeter [m]
$d =$ Horizontaldistanz mit negativer Steigung [m]

Die Gehzeit t zur Haltestelle ergibt sich nun aus:

$$t = \frac{x}{1.4m/s}$$

3.2.5 ÖV-Güteklassen

Die Kombination aus Haltestellenkategorie und Gehzeit zur Haltestelle liefert folgende ÖV-Güteklassen-Gruppierung:

Haltestellenkategorie	Gehzeit zur Haltestelle [s]			
	≤ 300	**(300, 450]**	**(450, 600]**	**(600, 900]**
I	Klasse A	Klasse A	Klasse B	Klasse C
II	Klasse A	Klasse B	Klasse C	Klasse D
III	Klasse B	Klasse C	Klasse D	Klasse E
IV	Klasse C	Klasse D	Klasse E	-
V	Klasse D	Klasse E	-	-
VI	Klasse E	-	-	-
VII	Klasse F	-	-	-

Tabelle 12: ÖV-Güteklassen

Die Grenze wird bei einem Angebot schlechter als Stundentakt und einem Einzugsgebiet von 300 Sekunden gezogen, da man bei einer weiteren Betrachtung nicht mehr von einer Erschliessung reden kann und so auch nicht ausgewiesen werden soll.

4 Umsetzungskonzept

In einer ersten Phase wurde die ÖV-Güteklassen 2018 Spezifikation in Kapitel 3 erstellt. Dabei wurde die Spezifikation so gestaltet, dass diese den aktuellen technischen Möglichkeiten gerecht wird. Dies impliziert unter anderem die Verwendung eines Routing-Graphen für die Berechnung des Einzuggebiets und das Einbeziehen eines hoch aufgelösten digitalen Terrainmodells. Dabei werden zusätzliche Fehler in den bisherigen Lösungen ausgemerzt.

In einer zweiten Phasen wird inkrementell die Spezifikation umgesetzt. Dabei wird ein Generator entwickelt, welcher die ÖV-Güteklassen aufgrund der in der ersten Phase erarbeiteten Spezifikation berechnet. Parallel dazu wird eine Web-Applikation mit zugehörigem Backend erarbeitet, welche die berechneten ÖV-Güteklassen 2018 darstellt und visuell dem Platzhirsch (Spezifikation des Bundesamt für Raumentwicklung) gegenüberstellt. Diese Web-Applikation verfolgt das Ziel, Verkehrs-, Raumplaner sowie Privatpersonen einen Einblick in die Qualität der ÖV-Erschliessung an einem Standort geben zu können. Die zweite Phase ist im Teil I beschrieben.

5 Resultate

In diesem Kapitel werden die Resultate der Arbeit präsentiert und mit den im Kapitel 2.1 vorgestellten bestehenden Berechnungsmethoden verglichen. Die technische Beschreibung zur Implementation befindet sich im Teil II Kapitel 5.

5.1 Zielerreichung

In einer Evaluation (Teil 2) wurden bestehende Ansätze zur Berechnung von ÖV-Güteklassen analysiert und verglichen. Es zeigte sich, dass die Methoden auf einer alten Norm aufbauen, die von einigen Kantonen erweitert wurden. Für die Berechnung der ganzen Schweiz gibt es aber keine Spezifikation, die moderne Analysen wie die Berücksichtigung des Strassennetzes vorsehen.

Unter Zuhilfenahme der verschiedenen Berechnungsmethodiken erstellten wir anschliessend eine eigene Spezifikation, die OeVGK18. Dabei haben wir neben einer genaueren Ermittlung der Erreichbarkeit (Kap. 2.3.1) und die Berücksichtigung des Terrains auch Punkte konkretisiert, die in bisherigen Methoden nicht genau spezifiziert sind, wie etwa die Berechnung des Kursintervalls (Kap. 2.3.3) oder die Bestimmung von Bahnknoten (Kap. 2.3.4).

In einem nächsten Schritt wurde die Berechnung anhand der OeVGK18-Spezifikation umgesetzt und automatisiert. So lassen sich die ÖV-Güteklassen auch für nachfolgende Jahre neu berechnen. Alle Parameter, die in der Spezifikation festgelegt wurden, sind in einer Konfigurationsdatei festgehalten und können beliebig verändert werden. Dadurch kann die Implementation auch als Basis für Weiterentwicklungen der Berechnungsmethoden dienen.

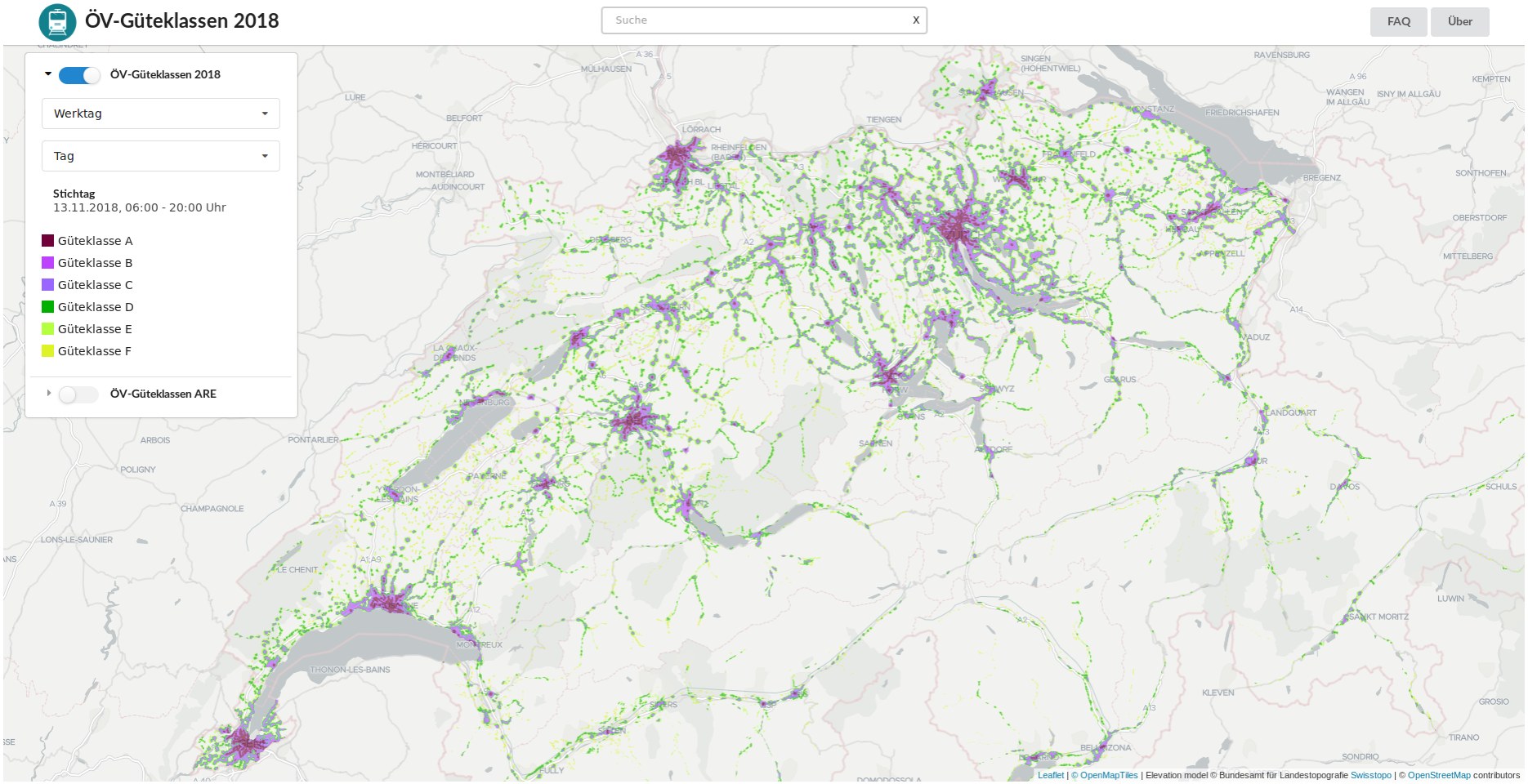

Abbildung 6: Darstellung der berechneten ÖV-Güteklassen in der Web-Applikation

Zusätzlich zur automatisierten Berechnung wurde eine Web-Applikation entwickelt, die es ermöglicht, die berechneten ÖV-Güteklassen im Browser darzustellen (siehe Abbildung 6). Ebenfalls wurden darin die bisherigen ÖV-Güteklassen vom Bundesamt für Raumentwicklung (ARE) integriert, was einen visuellen Vergleich der beiden Resultate erleichtert.

5.2 Vergleich mit bisherigen ÖV-Güteklassen

Im Folgenden wird an einigen Beispielen gezeigt, wie die Resultate unserer Berechnung der OeVGK18 von bisherigen Methoden abweichen. Als Vergleich werden dazu die ÖV-Güteklassen von ARE aus dem Jahre 2018 [1] verwendet.

5.2.1 Einfluss der Wegführung

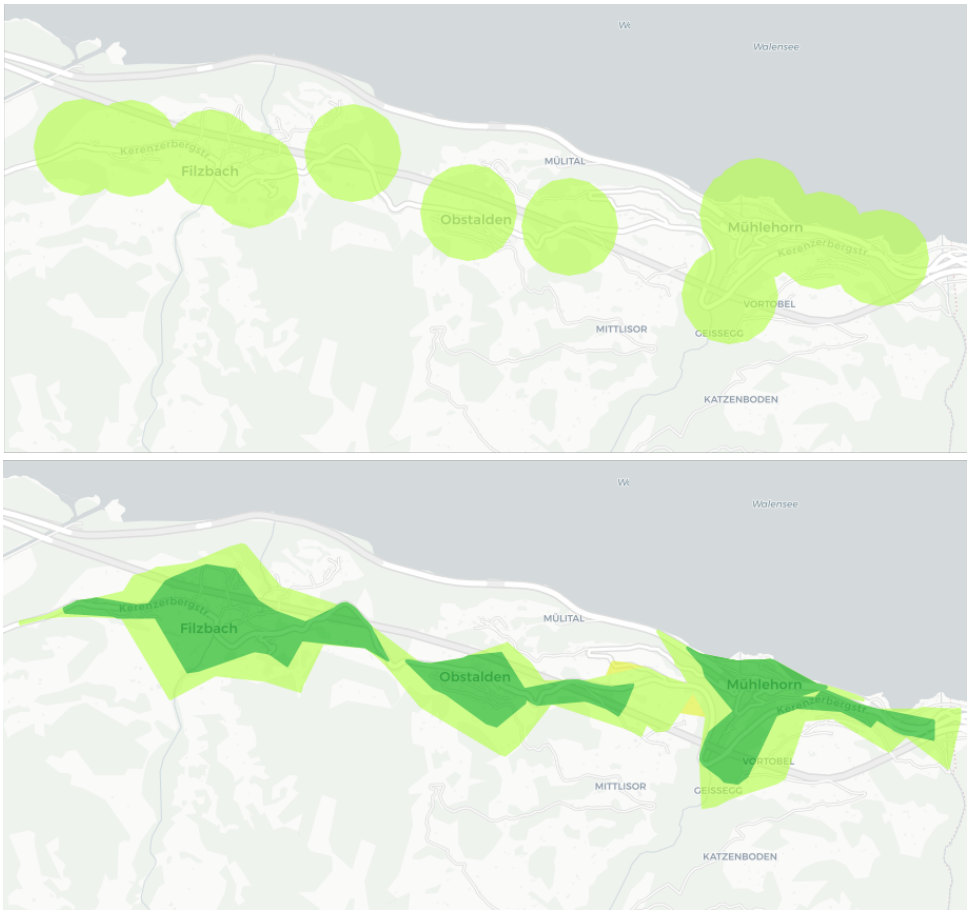

Abbildung 7: Vergleich zur Einfluss der Wegführung. Oben: ÖV-Güteklassen von ARE. Unten: ÖV-Güteklassen 2018.

Wie in 1.2.1 besprochen wird bei den bisherigen Ansätzen das Einzugsgebiet einer Haltestelle mit einem Kreis, und somit als Luftlinie, beschrieben. Mit OeVGK18 wird nun das Einzugsgebiet durch eine

Isochrone beschrieben. Die Fläche zeigt das Einzugsgebiet, das von einem Fussgänger von der Haltestelle aus in der definierten Zeit erreichbar ist.

Wie in Abbildung 7 zu sehen ist, entstehen dadurch Geometrien, die dem Strassen- und Wegnetz folgen. Dies ergibt einen realistischeren Eindruck über die effektiven Gebiete, die von einer Haltestelle erschlossen werden.

Die Berücksichtigung der Wegführung ist im Direktvergleich in Abbildung 8 noch deutlicher sichtbar.

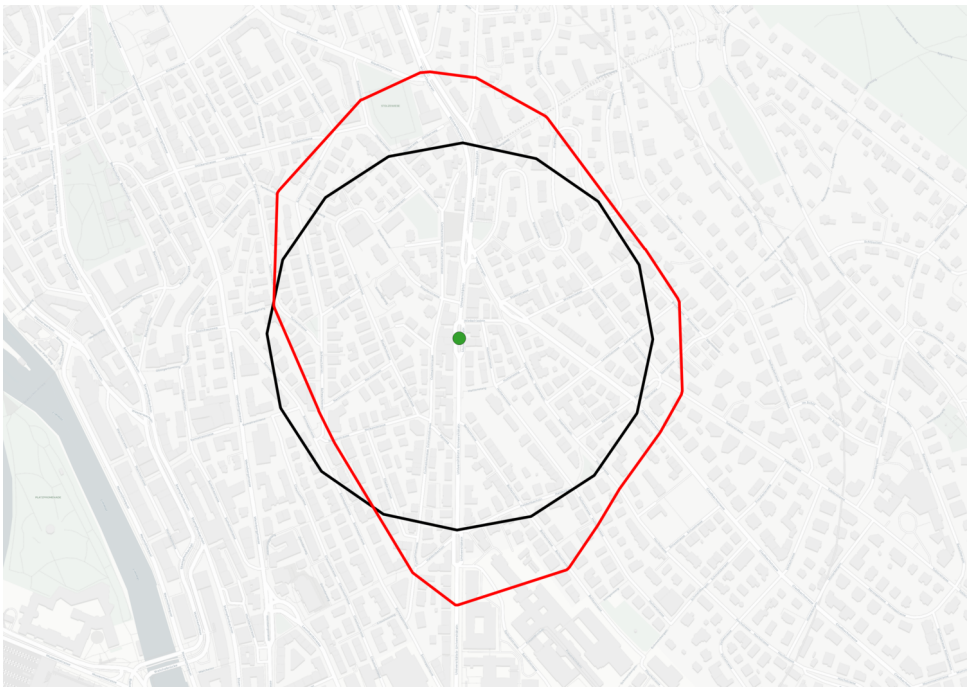

Abbildung 8: Direktvergleich Einzugsgebiet von ÖV-Güteklassen von ARE (schwarz) und ÖV-Güteklassen 2018 (rot) ohne Berücksichtigung der Topographie; Haltestelle Zürich, Winkelriedstrasse

5.2.2 Einfluss der Topographie

Der Stand der Technik berücksichtigt die Topographie nicht konsequent. Somit hat eine anspruchsvolle Topographie keinen Einfluss auf das Einzugsgebiet einer Haltestelle. Das sieht man gut, wenn man die ÖV-Güteklasse des Bundesamt für Raumentwicklung in Abbildung 8 betrachtet. Auf der gut sichbaren Strecke (Universitätsstrasse) werden im Durchmesser des schwarzen Kreises über eine Distanz von 300 Metern 9 Höhenmetern zurückgelegt. Nach der Definition der Berechnung der Leistungskilometern (siehe Kapitel 3.2.4) sind das zu den 300 Metern zusätzliche 90 Leistungsmeter, welche zurückgelegt werden müssen. Der Einfluss der Topographie ist nun im Direktvergleich in Abbildung 9 sichtbar. Die Haltestelle hat mit Einbezug der Topographie ein kleineres Einzugsgebiet. Man sieht, dass sich dies gut im unteren Teil bemerktbar macht. Die Strecke han von unten nach oben gesehen einen starken Anstieg, welcher ins Gewicht fällt. Beachtet man die Fläche von oben nach unten, sieht man nur

geringe Unterschiede, da das Gefälle den Grenzwert von 22% nicht überschreitet.

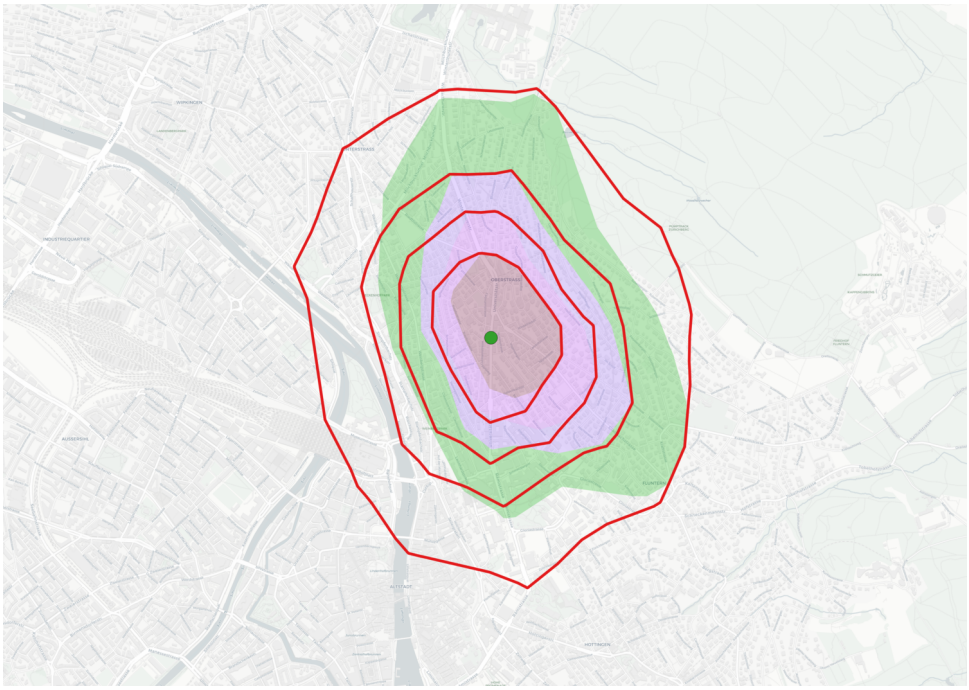

Abbildung 9: Direktvergleich Einzugsgebiet Isochronen mit (Fläche ausgefüllt) und ohne (rot) Berücksichtigung der Topographie; Haltestelle Zürich, Winkelriedstrasse

5.2.3 Verbesserte Berechnung des Kursintervalls

Der Unterschied in der Berechnung des Kursintervalls ist gut sichtbar, wenn man die Haltestelle „Aarau, Buchenhof“ betrachtet. Damit wir die Berechnungsmethodiken OeVGK18 und OeVGKARE vergleichen können, nehmen wir für die Auswertung von OeVGK18 den Stichtag 21.03.2018. Die Fahrplandaten des OeVGKARE stammen vom 21.03.2017. Dieser Vergleich ist in diesem Zusammenhang zulässig, da beide einen Werktag repräsentieren und keine Änderung im Fahrplan erkannt wurde. Ebenfalls sieht man für diese Haltestelle die Korrektur, welche das ARE für Verbindungen vornimmt, welche nur in eine Richtung befahren werden. So wird die Anzahl Verbindungen in diesem Fall verdoppelt, um sie anschliessend bei der Berechnung wieder zu halbieren.

An diesem Tag werden im Intervall 06:00 bis 20:00 Uhr 56 Fahrten an dieser Haltestelle registriert. Folgt man der Berechnungslogik des Bundesamts für Raumentwicklung, erfolgt daraus folgender Intervall: $840/((56*2)/2) = 15\text{min}$

Betrachtet man nun einen Auszug der Abfahrten in Tabelle 13, sieht man, dass beide Buslinien 4 Minuten versetzt alle 30 Minuten verkehren. Die Aussage, dass man an dieser Haltestelle einen 15-Minuten-Takt hat, ist somit nicht ganz korrekt. Kommt man um 06:10 Uhr an die Haltestelle, wartet man 23 Minuten auf die nächste Verbindung.

Abfahrtszeit	Bus-Nr
06:03	7
06:07	5
06:33	7
06:37	5
07:03	7
07:07	5
07:33	7
07:37	5
...	...

Tabelle 13: Abfahrtszeiten Bus 5 und 7, Haltestelle Aarau, Buchenhof

Die neue Intervall-Berechnung gemäss Kapitel 3.2.2 ergibt nun für die gleiche Haltestelle den Intervall 23.1 Minuten. Der Unterschied macht sich in der Bestimmung der Haltestellenkategorie bemerkbar. Da es sich bei dieser Haltestelle um eine Busstation handelt, fällt man nach der OeVGK18-Spezifikation von der Haltestellekategorie IV (OeVGKARE) in die Kategorie V.

Abschliessend kann man sagen, dass 23.1 Minuten eine bessere Annäherung an den eigentlichen Kursintervall als die 15 Minuten, die vom ARE berechnet wurde, ist. Dadurch wird das wahrgenommene Angebot an der Haltestelle besser wiederspiegelt.

5.2.4 Verbesserte Ermittlung von Bahnknoten

Bei der Ermittlung der Bahnknoten werden nur Haltestellen berücksichtigt, welche Verbindungen in 6 verschiedene Richtungen oder Zugang zu einer Fernverkehrsverbindung haben. Aufgrund der Daten des Bundesamt für Raumentwicklung nehmen wir an, dass bei der Berechnung der OeVGKARE nur Haltestellen als Bahnknoten identifiziert werden, welche mindestens drei verschiedene Richtungen bedienen. Diese Annahme kommt zustande, wenn man für die Bahnknoten des OeVGKARE die Anzahl verschiedenen Richtungen mit unserer Implementation eruiert. Dabei gibt es mit „Genève-Aéroport“ einen Ausreiser, welcher nur eine Richtung anbietet. Die Problematik ist in Kapitel 2.3.4 beschrieben.

Ein direkter Vergleich ist in Tabelle 14 ersichtlich.

	OeVGKARE	OeVGK18
Anzahl Bahnknoten	150	106

Tabelle 14: Vergleich Anzahl Bahnknoten

Da wir mindestens doppelt so viele verschiedene Richtungen wie OeVGKARE verlangen, schränken wir so die Identifikation als Bahnknoten ein, was sich in der Anzahl identifizierter Bahnknoten bemerkbar macht.

	Anzahl Bahnknoten
OeVGKARE ∩ OeVGK18	92
OeVGKARE \ OeVGK18	58
OeVGK18 \ OeVGKARE	14

Tabelle 15: Unterschiede in der Identifikation der Bahnknoten

Dass durch die höhere Anzahl verlangter verschiedener Richtungen eine kleinere Menge als Bahnknoten identifizierte Haltestellen resultiert, scheint logisch. Tabelle 15 nimmt die Unterschiede in den resultierenden Bahnknoten unter die Lupe. Überraschend ist, dass die Bahnknoten, welche durch OeVGK18 identifiziert werden, nicht komplett in OeVGKARE vorkommen. Es handelt sich um 14 Haltestellen, welche OeVGK18 im Unterschied zu OeVGKARE als Bahnknoten bestimmt werden. Dabei handelt es sich unter diesen um 2 Haltestellen, welche eine Fernverkehrsverbindung haben und so als Bahnknoten gezählt werden. Warum die restlichen 12 Haltestellen nicht ebenfalls als Bahnknoten identifiziert sind, entzieht sich unseren Kenntnissen. Darunter fällt beispielsweise die Haltestelle Flüelen mit 9 verschiedenen Richtungen.

5.3 Vergleich ÖV-Güteklassen mit unterschiedlichen Stichtagen und Zeitintervallen

Bisher hat man sich auf einen Stichtag ausserhalb der touristischen Hochsaison beschränkt, um so ein allgemeingültiges Resultat zu erhalten. Dadurch schränkt man die Auswertungsmöglichkeiten stark ein, da die Qualität der ÖV-Erschliessung beispielsweise an einem Wochenende oder am Abend/in der Nacht genauso interessant auszuwerten ist wie tagsüber. Durch die Konfigurationsmöglichkeit des „ÖV-Güteklassen 2018 Generators“ ist es eine Leichtigkeit, in einem Schritt die ÖV-Güteklassen für unterschiedliche Stichtage und Zeitintervalle zu berechnen.

In Abbildung 10 sieht man den Vergleich für einen Stichtag mit unterschiedlichen Zeitintervallen, links tagsüber und rechts in der Nacht. Anzumerken ist, dass beide mit denselben Ellen gemessen werden. Dadurch ergeben sich in der Nacht deutlich tiefere Güteklassen als am Tag.

5.4 Ausblick: Weiterentwicklung

Durch die neue Spezifikation und Web-Applikation ist die Grundlage für weitere und fortgeschrittene Auswertungen geschaffen.

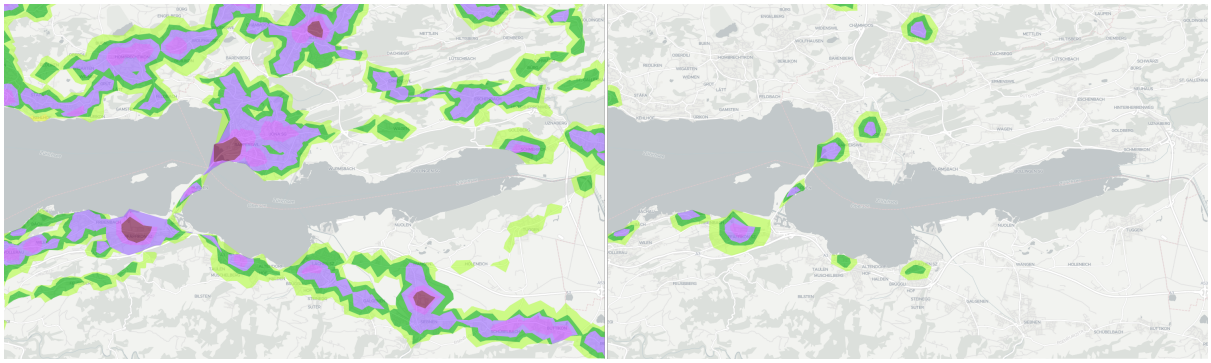

Abbildung 10: Vergleich Stichtag und Zeitintervall: 10.11.2018, 06:00 - 20:00 Uhr (links) und 10.11.2018, 01:00 - 04:00 Uhr (rechts)

Verknüpft man die neuen ÖV-Güteklassen mit der Bevölkerungsdichte, ist es möglich, Verbesserungspotential für den ÖV zu eruieren. Dadurch wären Auswertungen analog zu „95% der wohnhaften Personen in einem bestimmten Gebiet müssen mindestens in der ÖV-Güteklassen XYZ sein“ möglich.

Aber auch für den privaten Gebrauch ist eine Weiterentwicklung von Interesse. So ist ein Wohnortfinder denkbar, welcher die Qualität des ÖV-Anschlusses und anderen Faktoren (Nähe zu einer Schule oder Krankenhaus, etc.) berücksichtigt.

Raster-Ansatz

Für die Berechnung der Einzugsgebiete wird in dieser Arbeit der Ansatz von Isochronen verwendet, die entlang des Strassen- und Wegnetzes verlaufen. Die Grundannahme dahinter ist, dass sich Fussgänger immer entlang des Wegnetzes fortbewegen. Dabei gibt es Probleme bei offenen Fussgängerflächen, wie sie in [2] behandelt werden. Ein anderer Ansatz basiert statt auf Routen auf einem Raster [3]. Dieser geht davon aus, dass sich Fussgänger überall fortbewegen, wo keine Hindernisse (Gebäude, Flüsse, Barrieren, etc.) sie daran hindern. Dabei wird ein Raster über die Karte gelegt und jeder Kachel einen Kostenwert zugeordnet. Für die Erzeugung der Einzugsgebiete werden dann vom Startpunkt aus in alle Richtungen die Kacheln eingefärbt und bei jedem Schritt die Kosten der Kachel addiert. Dies wird so lange wiederholt, bis die gewünschten Maximalkosten erreicht werden. In Abbildung 11 ist ein Ergebnis einer solchen Berechnung abgebildet.

Mit dieser Methode ist die Qualität des Routing-Graphen und Problematiken wie bei Fussgängerflächen weniger relevant. Für die Berechnung der ÖV-Güteklassen ist dies ein spannender Ansatz für einen Vergleich mit der Lösung in dieser Arbeit.

Multimodaler Verkehr

In allen bisherigen Ansätzen zur Ermittlung von ÖV-Güteklassen wird das Einzugsgebiet von Haltestellen über die Laufdistanz charakterisiert. Dabei werden die Möglichkeiten des Individualverkehrs (wie etwa Auto, Fahrrad, etc.) nicht mit eingerechnet. Die Maximaldistanz eines Erschliessungsgebiets liegt bei Laufdistanzen bei 15 Minuten Gehzeit. Mit einem Fahrrad etwa wäre die Distanz deutlich grösser,

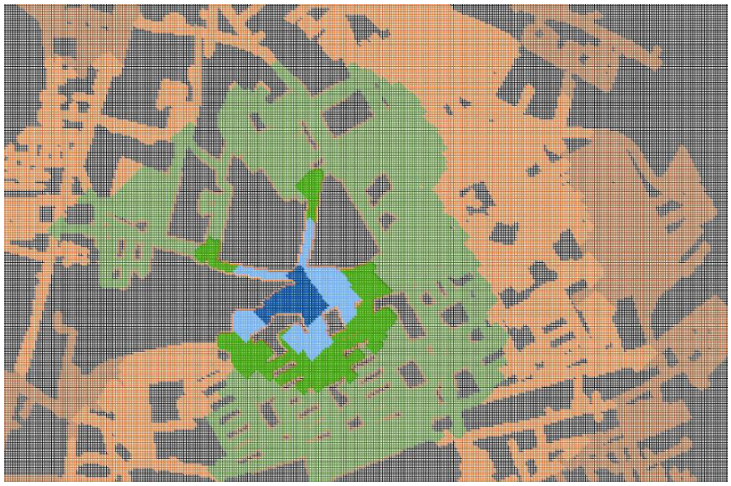

Abbildung 11: Grid- beziehungsweise raster-basierter Ansatz [3]

die in 15 Minuten zurückgelegt werden können. Für andere Verkehrsmittel bräuchte es daher jeweils andere Massstäbe.

Die jetzige Einschränkung auf Fussdistanzen führt auch daher, dass die ÖV-Güteklassen nicht zu komplex werden sollen, damit eine einfache und schnelle Interpretation möglich bleibt. Daher ist hier abzuwägen, inwiefern es sinnvoll ist, andere Arten des Individualverkehrs miteinzubeziehen.

5.5 Dank

Wir möchten folgenden Personen für ihre Unterstützung und Mitwirkung bei dieser Arbeit danken:

Prof. Stefan Keller, IFS Institut für Software, für die Zeit, Ressourcen, Kontakte, Know-How und Unterstützung, von welcher wir jederzeit profitieren konnten.

Prof. Claudio Büchel, IRAP Institut für Raumentwicklung, für die initiale Idee, Know-How über Verkehrsplanung und Fachunterstützung über das gesamte Projekt.

Mitarbeiter, IFS Institut für Software, für den regen Know-How-Austausch und die Unterstützung bei der Produktivsetzung.

II SW-Projektdokumentation

1 Überblick

Der Teil I Technischer Bericht verfolgt das Ziel, dem Leser einen Überblick über die Problemstellung und über den Inhalt der Arbeit zu geben. Dabei wurde dem Stand der Technik besondere Beachtung geschenkt und die Spezifikation „ÖV-Güteklassen 2018“ (Kapitel 3) erstellt. Der Teil SW-Projektdokumentation legt den Fokus auf deren konkreten Umsetzung.

2 Anforderungsspezifikation

2.1 Use Cases

Im Folgenden sind die funktionalen Anforderungen an das System mit all seinen Komponenten, welche im Kapitel 4 aufgeführt sind, als Use Cases im Brief-Format beschrieben. Zur Übersicht ist das Use Case Diagramm in Abbildung 12 zu betrachten.

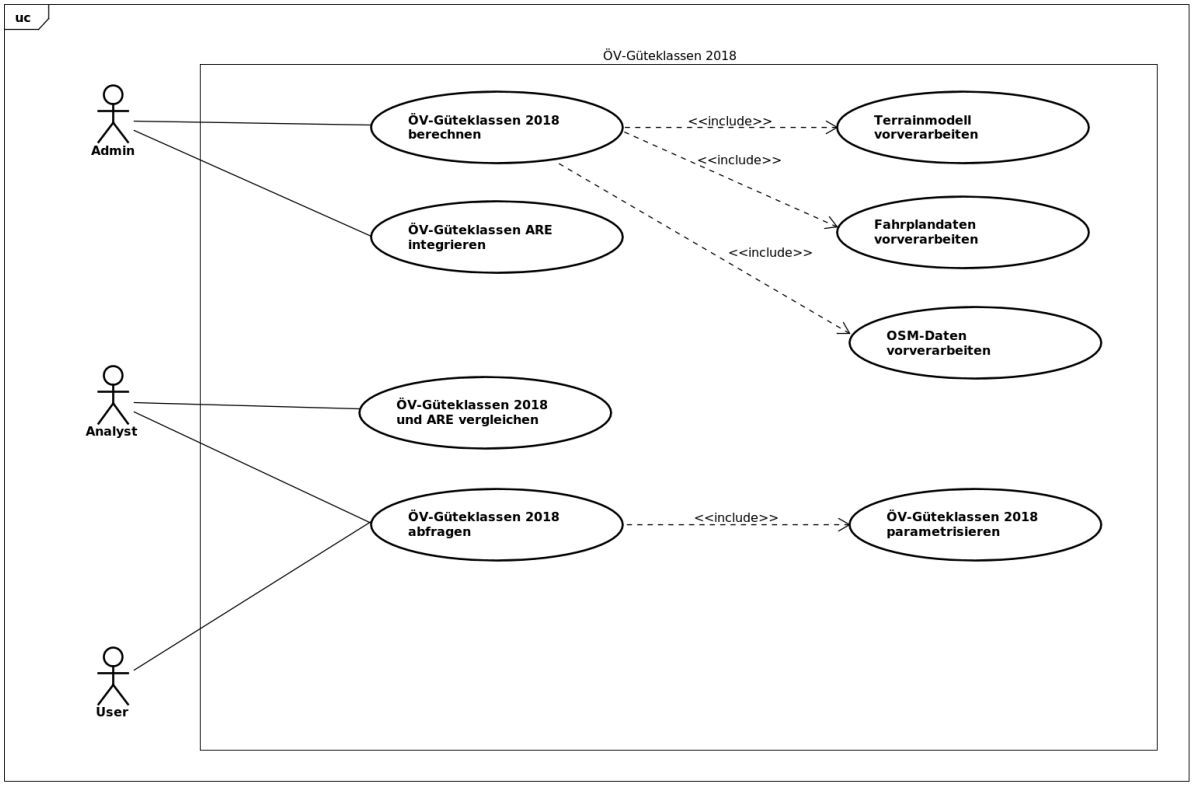

Abbildung 12: Use Case Diagramm

2.1.1 Aktoren

Um die Anforderungen akkurat evaluieren zu können, ist es essentiell, die Aktoren des Systems zu identifizieren. In der Tabelle 16 sind die Aktoren mit ihren Interessen aufgelistet.

Aktor	Beschreibung und Interesse
Admin	Ein *Admin* ist für die Bereitstellung von akkuraten ÖV-Güteklassen-Daten verantwortlich, welche er den Aktoren *Analyst* und *User* für die weitere Auswertung zur Verfügung stellt. So möchte er die erforderlichen Daten sammeln und verarbeiten.
Analyst	Unter dem Aktor *Analyst* sind Raum-/Verkehrsplaner, Transportgesellschaften und weitere Personen zusammengefasst, welche die ÖV-Güteklassen-Daten für analytische Zwecke und als Planungsinstrument verwenden möchten.
User	Ein *User* ist grundsätzlich am aktuellen Stand der ÖV-Güteklassen in einer bestimmten Region interessiert, welche er als Basis für weitere Entscheidungen bezieht.

Tabelle 16: Aktoren

2.1.2 UC01: ÖV-Güteklassen 2018 berechnen

Aktoren: *Admin*

Include: UC02: Terrainmodell vorverarbeiten, UC03: Fahrplandaten vorverarbeiten, UC04: OSM-Daten vorverarbeiten

Der *Admin* startet die Berechnung der ÖV-Güteklassen 2018 basierend auf einer Konfiguration (Stichtage und Zeitspannen).

2.1.3 UC02: Terrainmodell vorverarbeiten

Aktoren: *Admin*

Der *Admin* führt das manuelle Beziehen des Terrainmodell durch und übergibt diese dem System „ÖV-Güteklassen 2018“. Das System bereit die Daten so vor, dass sie in UC01: ÖV-Güteklassen 2018 berechnen für die „ÖV-Güteklassen 2018“-Berechnung einbezogen werden können Da diese Daten nicht einem stetigen Wandel unterstehen und von Swisstopo [9] nur alle 6 Jahren aktualisiert werden, steht es dem *Admin* frei, diesen Schritt auszulassen und bereits im System vorhandene, verarbeitete Höhendaten zu verwenden.

2.1.4 UC03: Fahrplandaten vorverarbeiten

Aktoren: *Admin*

Der *Admin* nutzt das System „ÖV-Güteklassen 2018“ für das Beziehen der Fahrplandaten über eine externe Schnittstelle. Das System bereit die Daten so vor, dass sie in UC01: ÖV-Güteklassen 2018 berechnen für die „ÖV-Güteklassen 2018“-Berechnung einbezogen werden können.

2.1.5 UC04: OSM-Daten vorverarbeiten

Aktoren: *Admin*

Der *Admin* nutzt das System „ÖV-Güteklassen 2018“ für das Beziehen der OSM-Daten über eine externe Schnittstelle. Das System bereit die Daten so vor, dass sie in UC01: ÖV-Güteklassen 2018 berechnen für die „ÖV-Güteklassen 2018“-Berechnung einbezogen werden können.

2.1.6 UC05: ÖV-Güteklassen ARE integrieren

Aktoren: *Admin*

Der *Admin* nutzt das System „ÖV-Güteklassen 2018“ für die Integration der bereits berechneten ÖV-Güteklassen ARE [1].

2.1.7 UC06: ÖV-Güteklassen 2018 und ARE vergleichen

Aktoren: *Analyst*

Der *Analyst* nutzt das System „ÖV-Güteklassen 2018“ für den Vergleich der ÖV-Güteklassen ARE und 2018. Das System präsentiert eine grafische Überlagerung der beiden ÖV-Güteklassen.

2.1.8 UC07: ÖV-Güteklassen 2018 abfragen

Aktoren: *Analyst, User*

Include: UC08: ÖV-Güteklassen 2018 parametrisieren

Die Aktoren *Analyst* und *User* nutzt das System „ÖV-Güteklassen 2018“, um die berechneten ÖV-Güteklassen 2018 anzuzeigen.

2.1.9 UC08: ÖV-Güteklassen 2018 parametrisieren

Aktoren: *Analyst, User*

Die Aktoren *Analyst* und *User* übergibt dem System „ÖV-Güteklassen 2018“ eine Konfiguration der anzuzeigenden ÖV-Güteklassen 2018. Das System „ÖV-Güteklassen 2018“ liefert auf Basis dieser die zugehörigen, vorgerechneten Daten.

2.2 Nicht-funktionale Anforderungen

Im Folgenden sind die nicht-funktionalen Anforderungen an das System „ÖV-Güteklassen 2018“ aufgelistet. Bei der Erarbeitung wird darauf geachtet, dass die Kriterien *Specific* und *Measurable* aus den *SMART*-Kriterien [17] eingehalten werden. Als Spezifikations-Template wird das bewährte Quality Attribute Scenario (QAS) des Software Engineering Institute (SEI) [18] eingesetzt. In der Kategorie *Stimulus Source* werden die Aktoren aus der Tabelle 16 wiederverwendet. Die erwähnten Komponenten sind im Kapitel 4 genauer beschrieben.

2.2.1 NFA01: Auslieferung der Komponenten als Docker-Image

Scenario	Auslieferung der Komponenten als Docker-Image
Business Goals	Deployment und Portierbarkeit gewährleisten
Relevant Quality Attributes	Maintainability, Manageability, Portability, Deployability
Stimulus Source	Admin
Stimulus	Die Generierung des Docker-Image wird gestartet
Environment	Normaler Betrieb (unabhängig vom laufenden Web-App und Backend)
Artifact	Das ganze System „ÖV-Güteklassen 2018“ mit all seinen Komponenten
Response	Die Komponenten sind als Docker-Images verfügbar, welche auf Docker Hub hochgeladen werden könnten
Response Measure	Einzelne Komponenten lassen sich als Docker-Image starten

Tabelle 17: QAS NFA01

2.2.2 NFA02: Berechnung der ÖV-Güteklassen 2018

Scenario	Berechnung der ÖV-Güteklassen 2018
Business Goals	Aktualisierte ÖV-Güteklassen 2018 für weitere Auswertungen
Relevant Quality Attributes	Performance (response time), Usability, Manageability
Stimulus Source	Admin
Stimulus	Ausführung UC01: ÖV-Güteklassen 2018 berechnen
Environment	Normaler Betrieb (unabhängig vom laufenden Web-App und Backend)
Artifact	ÖV-Güteklassen 2018 Generator und seine abhängige Fremdsysteme
Response	Berechnete ÖV-Güteklassen 2018 als GeoJSON
Response Measure	Die ÖV-Güteklassen 2018 lassen sich innerhalb von 12 Stunden nach Absetzen des Befehls über ein CLI für 6 Stichtage berechnen

Tabelle 18: QAS NFA02

2.2.3 NFA03: Periodische Aktualisierung der ÖV-Güteklassen 2018

Scenario	Periodische Aktualisierung der ÖV-Güteklassen 2018
Business Goals	Möglichkeit zur automatischen Aktualisierung der ÖV-Güteklassen 2018 für weitere Auswertungen
Relevant Quality Attributes	Administrability
Stimulus Source	Cron-Job
Stimulus	Periodische Ausführung UC01: ÖV-Güteklassen 2018 berechnen
Environment	Normaler Betrieb (unabhängig vom laufenden Web-App und Backend)
Artifact	ÖV-Güteklassen 2018 Generator und seine abhängige Fremdsysteme
Response	Berechnete ÖV-Güteklassen 2018 als GeoJSON
Response Measure	ÖV-Güteklassen 2018 Generator lässt ein automatisiertes Starten zu, um so periodisch aktualisierte GeoJSONs bereitstellen zu können, welche von anderen Komponenten verwendet werden können

Tabelle 19: QAS NFA03

2.2.4 NFA04: Dauer des Unterbruchs bei Integration aktualisierter Daten

Scenario	Dauer des Unterbruchs bei Integration aktualisierter Daten
Business Goals	Kurzer Unterbruch während des Einspielens aktualisierter Daten
Relevant Quality Attributes	Availability
Stimulus Source	Admin
Stimulus	Nach Ausführung von UC01: ÖV-Güteklassen 2018 berechnen und UC05: ÖV-Güteklassen ARE integrieren durch manuelle Interaktion des Admin
Environment	Web-App und Backend ist heruntergefahren
Artifact	Web-App und Backend
Response	Web-App ist wieder verfügbar und Backend ist mit aktualisierten „ÖV-Güteklassen 2018“-Daten gestartet
Response Measure	Web-App ist nicht länger als eine Stunde vom Zeitpunkt des Herunterfahrens nicht verfügbar

Tabelle 20: QAS NFA04

2.2.5 NFA05: Antwortzeit ÖV-Güteklassen 2018 anzeigen

Scenario	Antwortzeit ÖV-Güteklassen 2018 anzeigen
Business Goals	Nutzerzufriedenheit
Relevant Quality Attributes	Performance (response time), Scalability, Usability
Stimulus Source	User, Analyst
Stimulus	Ausführung UC06: ÖV-Güteklassen 2018 und ARE vergleichen oder UC07: ÖV-Güteklassen 2018 abfragen
Environment	Normaler Betrieb
Artifact	Web-App und Backend
Response	ÖV-Güteklassen 2018 werden für den gewünschten Bereich angezeigt
Response Measure	ÖV-Güteklassen 2018 werden nach Verändern des Zooms innerhalb von 5 Sekunden angezeigt

Tabelle 21: QAS NFA05

3 Analyse

3.1 Routing Engine

In den Rahmenbedingungen (Kapitel 1.5) wurde festgehalten, für die Implementation der „ÖV-Güteklassen 2018“ PostGIS [19] und die Routing-Engine pgRouting [20] zu verwenden, soweit diese die gewünschte Funktionalität unterstützt. Nachfolgend soll kurz aufgezeigt werden, welche alternativen Technologien in Frage kommen und ob die technische Machbarkeit mit PostGIS/pgRouting gegeben ist.

Für die Wahl einer Routing-Engine gibt es drei grosse Open-Source Lösungen, die sich bewährt haben: OSRM [21], GraphHopper [22] und pgRouting. Für unsere Zwecke sind folgende Kriterien massgebend:

1. Das Erstellen von Isochronen

2. Die Integration von Höhendaten

Für das Kriterium 1 bringen GraphHopper und pgRouting bereits eingebaute Funktionen mit, für OSRM gibt es ein experimentelles Tool von Mapbox [23].

Für das zweite Kriterium, die Integration von Höhendaten, gibt es bei GraphHopper und OSRM eingeschränkte Möglichkeiten. GraphHopper bietet zwar an, das weltweit abdeckende Modell von Shuttle

Radar Topography Mission (SRTM) direkt einzubinden, dessen Genauigkeit ist mit 90 Metern aber deutlich tiefer als bei unserem swissALTI^{3D}-Modell mit 2–10 Metern [9]. Die Integration eines eigenen Modells würde bei beiden Routing-Engines eine Anpassung der Implementation benötigen.

Bei pgRouting bringt die Implementation selbst keine direkte Unterstützung von Höhendaten mit. Im Unterschied zu den anderen Varianten kann die Kosten-Funktion zur Berechnung von Wegzeiten aber einfach selbst angepasst werden. Durch die Benutzung von PostgreSQL mit PostGIS bietet dieser Ansatz deutlich mehr Flexibilität. Das Terrainmodell kann in einer separaten Datenbank effizient als Raster persistiert und für die Kostenberechnung der einzelnen Kanten abgerufen werden.

Da die Fahrplandaten ebenfalls in relationaler Form in einer PostgreSQL-Datenbank gehalten werden, wird auch die Integration dieser Daten mit dem Routing-Graphen erleichtert.

Abschliessend kann man sagen, dass die Kriterien mit pgRouting erfüllt werden und sich von den drei evaluierten Routing-Engines am besten für die Integration mit den restlichen Daten (Fahrplan- und Höhendaten) eignet.

3.2 Evaluation Frontend-Framework

Wie in den Rahmenbedingungen (Kap. 1.5) vorausgesetzt, stehen für die Technologie der Web-Applikation (Frontend) die Optionen React [24] oder Vue.js [25] zur Auswahl. Die Wahl viel auf diese zwei Frameworks, da sich beide momentan schnell verbreiten und zum Design von Single Page Applications (SPA) ausgelegt sind, was für unsere Zwecke zum Darstellen einer Karte gut geeignet ist. Ein grösseres Framework wie Angular ist für unsere Anforderungen zu mächtig, weshalb die Evaluation auf diese zwei leichtgewichtigeren Varianten eingeschränkt wird.

Vue.js und React werden im Folgenden anhand mehrerer Kriterien verglichen. Als Demonstration wird mit beiden Varianten eine identische kleine Web-Applikation entwickelt. Diese besteht aus einer Web-Karte und einem kleinen Kontrollelement, um einen zusätzlichen Layer einzublenden. Der Code ist auf Github [26] zu finden.

Zum Schluss wird ein Fazit gezogen und entschieden, welches Framework für das Frontend eingesetzt wird.

3.2.1 Funktionsumfang

Sowohl React als auch Vue.js bauen auf sehr ähnlichen Prinzipien auf. Applikationen werden in einzelne Komponenten zerteilt und aufeinander aufgebaut. In einer Komponente ist jeweils die Darstellung (Template) und die UI-Logik miteinander gekoppelt. Komponenten halten intern Daten, die von einer anderen Komponente bedingt verändert werden können.

Ein Unterschied liegt in der Defintion von HTML-Templates. Vue.js verwendet reines HTML mit einer erweiterten Template-Syntax, ähnlich wie bei herkömmlichen Template-Engines. React dagegen benutzt JSX [27], eine spezielle Syntax, bei der JavaScript und HTML gemischt werden können. In einem Zwischenschritt wird dies zu JavaScript kompiliert, das schlussendlich HTML-Elemente erstellt. Dies macht React etwas angenehmer, da in den Templates direkt beliebiges JavaScript eingebunden werden kann. Bei Vue.js sind nur einzelne Expressions nötig, und der Rest der Syntax muss HTML-kompatibel sein. So werden z.B. JavaScript-Variablen wie `myVar` in die HTML-kompatible Form `my-var` umgewandelt, was bei der Benutzung verwirrend sein kann.

React wird von Facebook entwickelt und ist etwas älter (2013) als Vue.js (2014), das unabhängig unterhalten wird. Dadurch gibt es beim Paket-Repository von NPM mehr vorgefertigte Komponenten für React (ca. 50'000 verglichen mit ca. 10'500 für Vue.js). Die Unterstützung und Wartung von Facebook spricht dafür, dass React auch in den nächsten Jahren stets weiter entwickelt wird.

Beide Frameworks sind aber schlank gehalten und bieten nur die Kernfunktionalitäten. Erweiterte Komponenten wie Routing oder State-Management sind in andere Frameworks ausgelagert, welche jeweils eingebunden werden können.

Insgesamt haben React und Vue.js einen sehr ähnlichen Funktionsumfang und verwenden fast identische Prinzipien.

3.2.2 Integration Leaflet

Für das Einfügen einer Web-Karte mit Raster-Kacheln in eine Web-Applikation hat sich die Library *Leaflet* [28] bewährt. Es wird analysiert, wie fortgeschritten die Leaflet-Unterstützung in den jeweiligen Frameworks ist.

Für React bietet *react-leaflet* [29] die Funktionalität von Leaflet an, das Equivalent für Vue.js bildet *Vue2Leaflet* [30].

Beide Komponenten bieten eine rudimentäre Dokumentation an, die Bedienung wird aber erst mit den zahlreichen Beispielen klar. Bezüglich der Funktionalität ist kein grosser Unterschied festzustellen. *Vue2Leaflet* bildet zwar nicht die komplette Leaflet-Library ab, die für diese Arbeit benötigten Funktionen sind aber alle vorhanden.

3.2.3 Integration Vector Tiles

Vector Tiles sind Kacheln, die Ausschnitte aus Karten als Vektoren darstellen, anstatt wie bei Raster-Kacheln mit simplen Bildern [31]. Dies hat unter anderem den Vorteil, dass nicht bei jeder Zoomstufe ein neues Bild geladen werden muss, sondern die Vektoren beliebig vergrössert werden können.

Für die Integration solcher Vector Tiles in Webkarten hat die Firma Mapbox ein Format spezifiziert und mit *Mapbox GL JS* [32] eine JavaScript-Library veröffentlicht, um diese einzubinden. Diese Library bietet deutlich mehr Möglichkeiten als Leaflet, um die Darstellung von Kacheln individuell anzupassen. Die Vektor-Daten selbst sind bei Mapbox bis zu einem gewissen Abrufvolumen kostenlos erhältlich. Es können aber auch andere Datenquellen wie z.B. von OpenMapTiles [33] verwendet werden.

Für die Integration mit React und Vue.js gibt es mehrere vorgefertigte Komponenten von Dritten. Wir haben jeweils die beliebtesten (Anzahl Github-Stars) davon angeschaut. Für React ist dies *react-mapbox-gl* [34], für Vue.js *vue-mapbox-gl* [35].

Die React-Komponente ist gut dokumentiert und relativ einfach zu benutzen. Die einzelnen Komponenten sind sauber in React-Komponenten abgebildet, was das Handling mit mehreren Layern angenehm macht.

Dagegen ist die Vue.js-Komponente nur ein einfacher Wrapper um die API von *Mapbox GL JS*. So kann z.B. ein zusätzlicher Layer nicht direkt im Template eingebunden werden, sondern muss über Event-Handling nach dem Laden der Karte hinzugefügt werden. Dies macht das dynamische Hinzufügen und Entfernen von Karten-Elementen, was wir in unserer Applikation viel benötigen, sehr mühsam.

3.2.4 Lernkurve

Die Lernkurve beschreibt under anderem, wie viel Zeit benötigt wird, sich in die neue Technologie einzuarbeiten. Dies ist durchaus ein subjektives Kriterium und hängt von unserem bestehenden Erfahrung mit anderen Web-Frameworks ab.

Die beiden sehr ähnlichen Philosophien von React und Vue.js weichen in gewissen Punkten stark von uns vertrauten Praktiken ab, so etwa die inhärente Kopplung von Darstellung und Logik, die in anderen Modellen wie MVC (Model-View-Controller) aufgespalten wird. Insofern benötigen beide Varianten eine gewisse Einarbeitungszeit, um sich mit diesen Konzepten vertraut zu machen.

Bei React kommt zusätzlich die neue Syntax JSX [27] dazu, die zwar etwas angenehmer zu lesen ist, aber in vielen kleinen Punkten von bereits Bekanntem abweicht. So mussten z.B. HTML-Attribute wie `for` umbenannt werden, um nicht eine Namenskollision mit JavaScript zu verursachen.

Die Template-Syntax von Vue.js ist deutlich vertrauter, auch weil sie sehr dem Stil von anderen Template-Engines ähnelt. Vue.js erlaubt auch einiges an „Syntactic Sugar“ in den Templates. So ist simples Data-Binding mit Event-Handling direkt im Template möglich, während bei React dies separat behandelt werden muss.

Aufgrund dieser Einschätzung kann man sagen, dass der Lernerfolg bei Vue.js schneller eintritt.

3.2.5 Tooling

Da sowohl React wie auch Vue.js auf NodeJS aufsetzen und den Paketmanager NPM benutzen, ist das Tooling sehr ähnlich.

Um eine neue Applikation zu erstellen, bietet React das Tool *create-react-app* [36] an. Beim Ausführen kann man angeben, welche zusätzlichen Libraries (z.B. für Unit-Testing) man verwenden möchte. Während der Entwicklung kann ausserdem ein lokaler Server gestartet werden, der jeweils automatisch das Browser-Fenster neu lädt, wenn sich der Source-Code geändert hat.

Vue.js bietet mit *vue-cli* [37] ein ähnliches Tool an. Um eine neue Applikation aufzusetzen, kann zwischen verschiedenen Templates gewählt werden, welche sich aufgrund der geplanten Grösse der Applikation unterscheiden. Während der Entwicklung bietet *vue-cli* die praktisch identischen Möglichkeiten wie das Tool für React.

Insgesamt kann zwischen den beiden Varianten in Sachen Tooling kein markanter Unterschied festgestellt werden.

3.2.6 Auswertung

Für die Bewertung wird jedem Kriterium einen Wert von 1 bis 6 zugewiesen, wobei 6 die Bestnote ist. Jedes Kriterium wird anhand der Relevanz und Wichtigkeit in unserem Projekt gewichtet.

ID	Kriterium	Relatives Gewicht (0-1)	Vue.js	React
1	Funktionsumfang	0.2	5 / 1.0	5 / 1.0
2	Integration Leaflet	0.1	5 / 0.5	4 / 0.4
3	Integration Vector Tiles	0.3	2 / 0.6	5 / 1.5
4	Lernkurve	0.2	4 / 0.8	3 / 0.6
5	Tooling	0.2	5 / 1.0	5 / 1.0
Total			**21 / 3.9**	**22 / 4.5**

Tabelle 22: Resultat der Frontend-Framework-Evaluation

Wie in der Tabelle 22 zu sehen ist, sind React und Vue.js vor der Gewichtung der Kriterien sehr ähnlich bewertet. Das Kriterium Integration Vector Tiles sorgt für den Ausschlag zu Gunsten von React. Dieses Kriterium haben wir stark gewichtet, da die Web-Karte die Hauptkomponente unserer Web-Applikation bildet und mit *Mapbox GL JS* deutlich mehr Flexibilität als Leaflet anbietet. In diesem Punkt bieten die vorhandenen Komponenten für Vue.js nicht die gewünschte Qualität.

3.3 ÖV-Güteklassen 2018 Generator

Der Container ÖV-Güteklassen 2018 Generator (siehe Kapitel 4) ist für die Berechnung der „ÖV-Güteklassen 2018“ verantwortlich. Umgesetzt wird diese Komponente als Python-Applikation. Für die Kommunikation und Integration mit den Umsystemen sind einige Libraries nötig, die im Folgenden kurz diskutiert werden.

3.3.1 Integration mit PostgreSQL

Für die Integration der Fahrplandaten und der Routing-Engine für die Berechnung der Isochronen ist eine Integration mit PostgreSQL nötig. Die Routing-Engine läuft auf pgRouting, eine Erweiterung von PostGIS, das wiederum eine Erweiterung von PostgreSQL ist. Insofern braucht es für beide Integrationen lediglich eine Möglichkeit, mit PostgreSQL zu kommunizieren.

Dazu bietet sich bei Python die relativ neue Library *records* [38] an.

Für unsere Zwecke braucht es nur simple SQL-Abfragen, da ein Grossteil der Logik mit Stored Procedures auf dem SQL-Server selbst umgesetzt wird.

3.3.2 Umgang mit Geometrien

In der Business-Logik des Generators wird viel mit Geometrien gehandhabt, die schlussendlich zu den „ÖV-Güteklassen 2018“ verabeitet werden. Hierzu hat sich die Python-Library *Shapely* [39] bewährt. Diese unterstützt die GEOS-Funktionen, die PostGIS ebenfalls implementiert. Dadurch können mit den Geometrien die gleichen Berechnungen wie auf der Datenbank vorgenommen werden, falls dies benötigt wird. Dies spricht somit für eine reibungslose Integration.

Ebenfalls können Geometrien mit *Shapely* direkt in GeoJSON exportiert werden. Dies erlaubt eine einfache Serialisierung, was wiederum eine einfache Weiterverarbeitung der Resultate ermöglicht.

3.4 Design Frontend

Für das Frontend (Web-Applikation mit React) wird für das Design des User Interface ein Entwurf in Form von Wireframes erstellt. Das Wireframe der Standard-Ansicht ist in Abbildung 13 dargestellt. Die restlichen Wireframes sind im Anhang Wireframes abgebildet.

Grundsätzlich besteht das UI aus einer interaktiven Web-Karte, die sich über das gesamte Browser-Fenster erstreckt. Darauf befindet sich ein Steuerelement, um die Anzeige der ÖV-Güteklassen zu

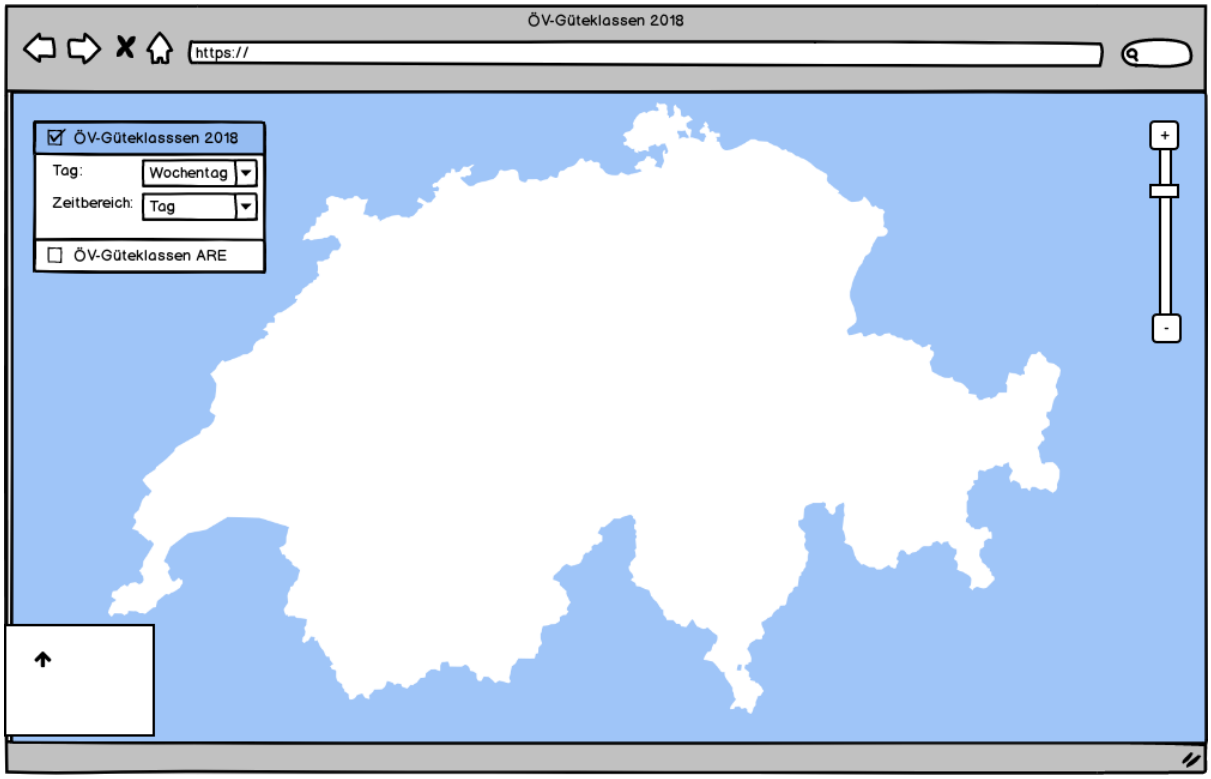

Abbildung 13: Wireframe der Standardansicht in der Web-Applikation

konfigurieren. Die beiden Karten-Layer der Datensätze OeVGK18 und OeVGKARE können ein- oder ausgeblendet werden. Bei der OeVGK18 können zusätzlich Parameter wie der Tag (Wochentag oder Wochenende) sowie der Zeitbereich (Tag, Abend oder Nacht) konfiguriert werden.

4 Architektur

In diesem Kapitel wird die Architektur unserer Applikation und die Schnittstellen zu den Umsystemen besprochen. Als Anhaltspunkt wird das C4 Modell [40] von Simon Brown verwendet. In einem ersten Schritt wird unsere Applikation „ÖV-Güteklassen 2018“ in den Kontext des grösseren Systems gesetzt. Anschliessend teilen wir das System „ÖV-Güteklassen 2018“ in einzelne Container und den zentralen Container „ÖV-Güteklassen 2018 Generator“ in einzelne Komponenten auf.

4.1 Systemkontext

In Abbildung 14 ist gestrichelt eingerahmt das System „ÖV-Güteklassen 2018“ zusammen mit den dazugehörigen Umsystemen dargestellt. Die Pfeile bedeuten, dass ein System in Richtung der Pfeilspitze Anfragen an ein anderes System sendet. Die Daten fliessen dementsprechend in die entgegengesetzte Richtung.

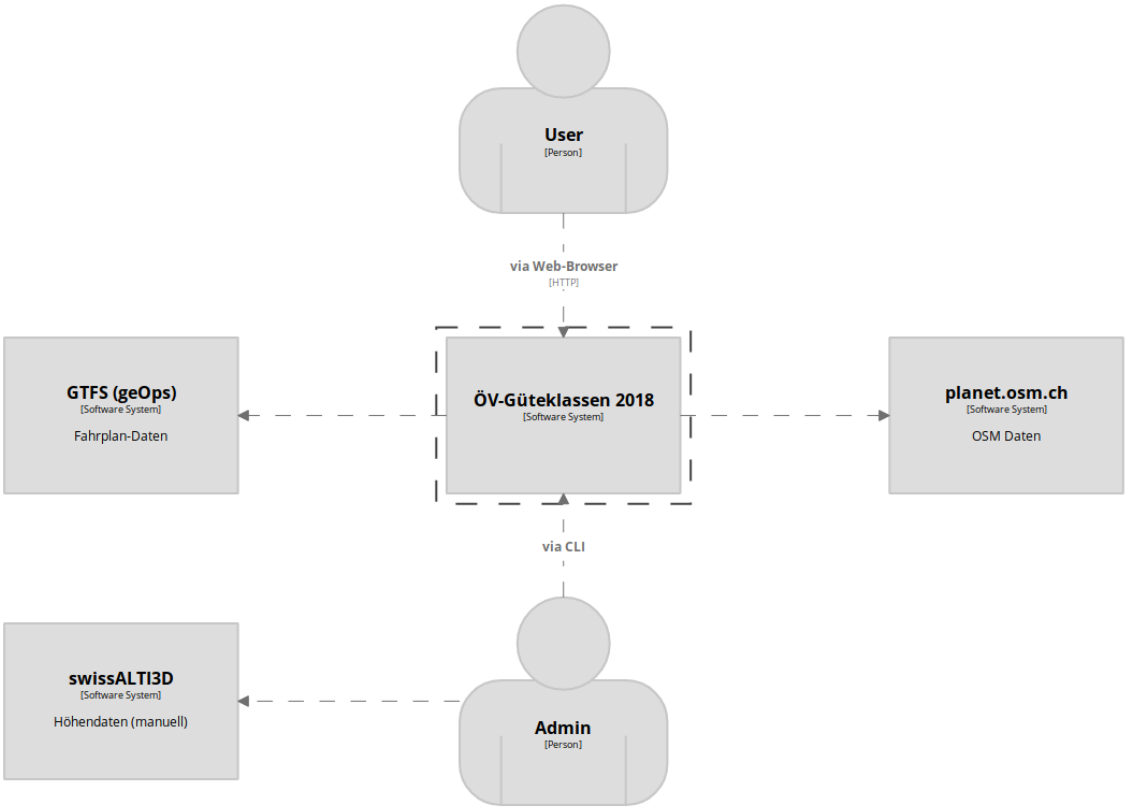

Abbildung 14: Systemkontextdiagramm ÖV-Güteklassen 2018

Im Folgenden werden die Umsysteme sowie die Beziehungen zu unserem System „ÖV-Güteklassen 2018“ genauer beschrieben.

4.1.1 swissALTI^{3D}

Als Terrainmodell wird swissALTI^{3D}, ein Produkt von Swisstopo [9], verwendet. Dieses wird aus lizenztechnischen Gründen und aufgrund der Grösse (ca. 120 GB) manuell vom Admin bezogen. Ebenfalls sind die Daten keinem stetigem Wandel unterstellt. Sie werden von Swisstopo [9] in einem Nachführungszyklus von 6 Jahren aktualisiert.

Das Terrainmodell wird anschliessend in die Routing-Engine integriert, wo die Höhenunterschiede für die Kostenberechnung einer Route verwendet werden können.

4.1.2 GTFS

Die Fahrplandaten werden von der SBB regelmässig im HAFAS-Format [8] publiziert. geOps [41] konvertiert diese in das offene und einfacher handhabbare Format General Transit Feed Specification (GTFS) [42] und publiziert die Daten kostenfrei. Diese können im CSV-Format für eine Speicherung

in einer relationalen Datenbank bezogen werden. Das Datenmodell ist in Abbildung 15 ersichtlich.

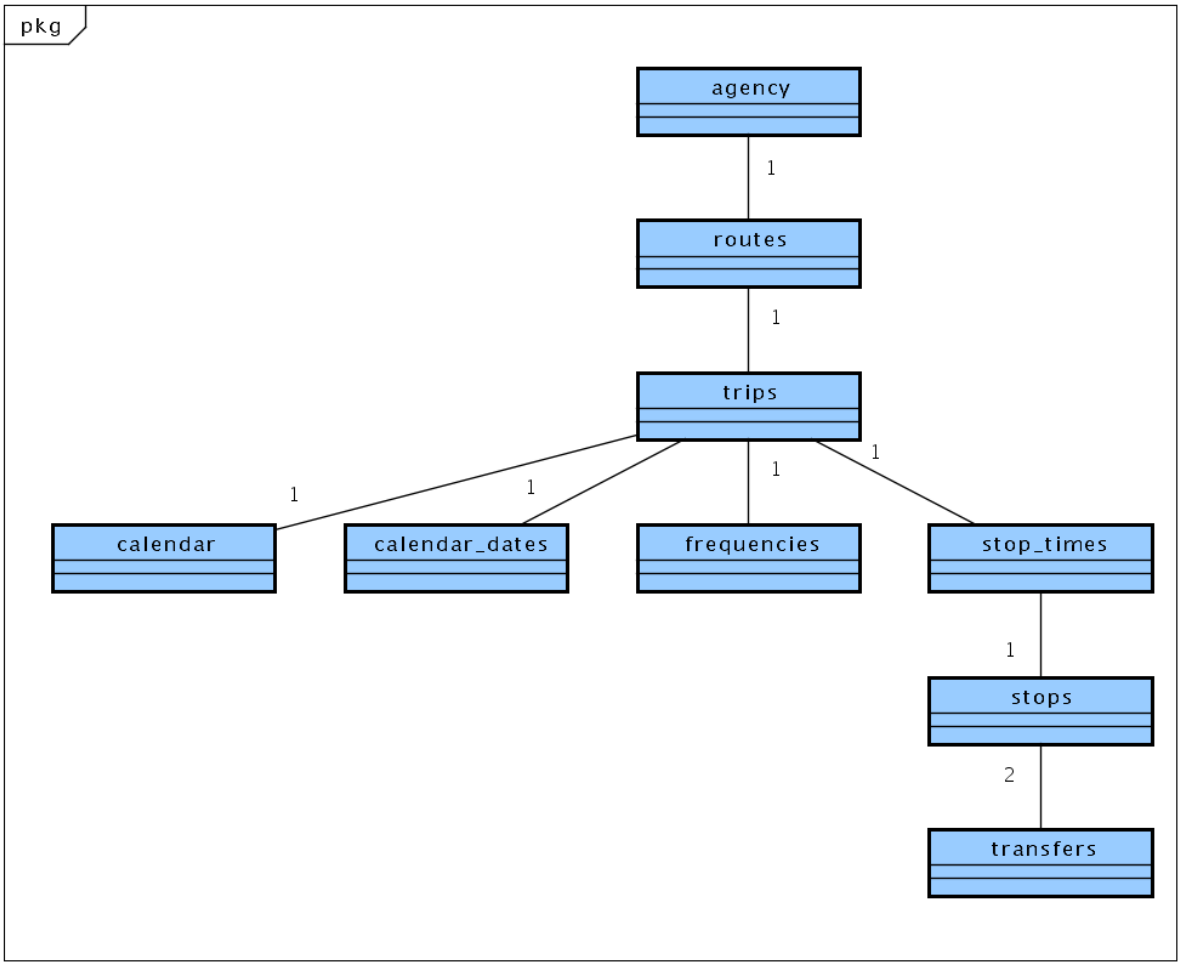

Abbildung 15: GTFS Datenmodell

4.1.3 planet.osm.ch

Für die Berechnung der ÖV-Güteklassen braucht es aktuelle Kartendaten, um die Erreichbarkeiten von Haltestellen entlang dem Strassennetz zu analysieren. Dazu bieten sich Daten von OpenStreetMap ideal an, da sie frei verfügbar sind und kontinuierlich aktualisiert werden.

Unter planet.osm.ch [43] werden täglich aktualisierte OSM-Daten der ganzen Schweiz bereit gestellt. Die Daten sind im Protocolbuffer Binary Format (PBF) und werden mit entsprechenden Tools in eine relationale Datenbank importiert, wo sie von einer Routing-Engine genutzt werden können.

4.2 Container

In der Abbildung 16 ist das System „ÖV-Güteklassen 2018“, wie es in im Systemkontext vorkam, in einzelne Container aufgespalten. Im Folgenden werden die einzelnen Container genauer beschrieben.

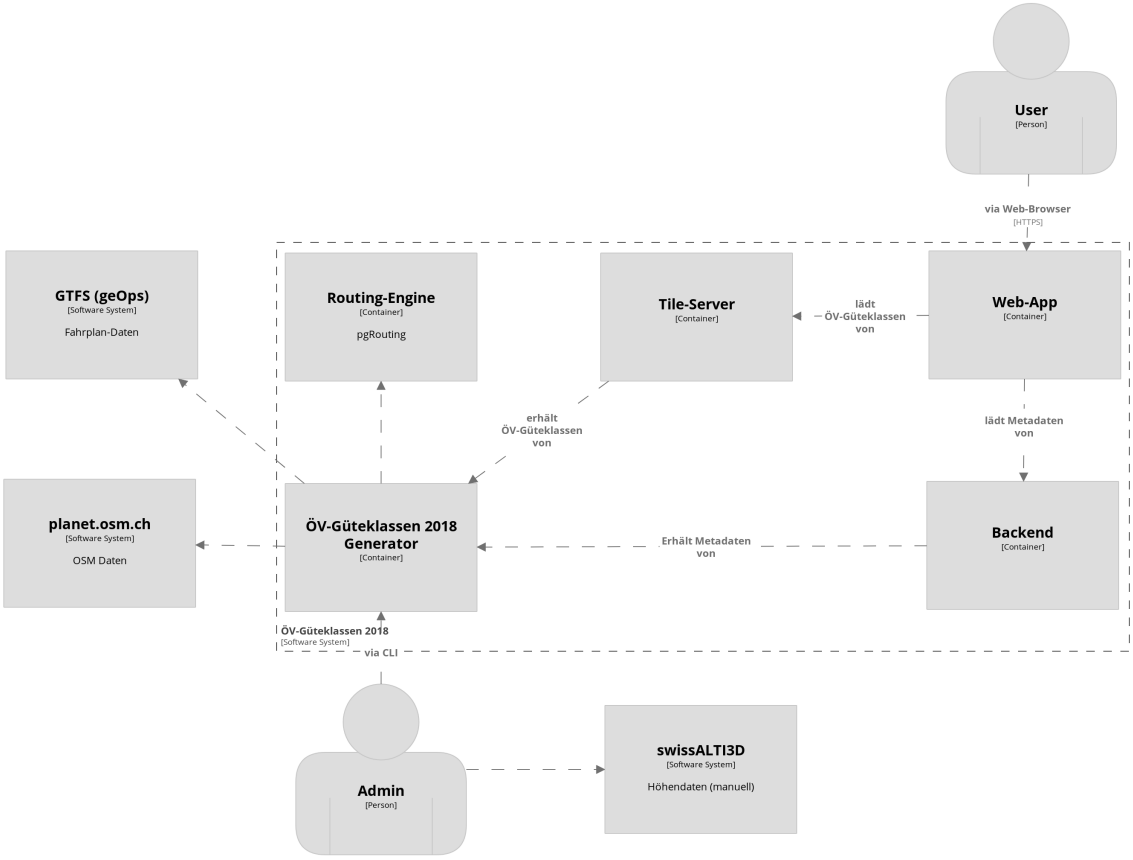

Abbildung 16: Containerdiagramm ÖV-Güteklassen 2018

4.2.1 ÖV-Güteklassen 2018 Generator

Der „ÖV-Güteklassen 2018 Generator“ stellt das Kernstück der Berechnung der ÖV-Güteklassen 2018 aufgrund der Spezifikation dar. In der Abbildung 17 ist das Schichten-Diagramm ersichtlich. Die Kernlogik der Berechnung befindet sich in der Schicht *business*. Der Generator agiert auf unterschiedlichen Datenquellen, so werden Fahrplandaten (GTFS), ein Routing-Graphen (planet.osm.ch und Routing-Engine) und ein Terrain-Modell (swissALTI^{3D}) verarbeitet und genutzt. Die Komponente setzt voraus, dass vor dem Starten diese Daten im System vorhanden sind. Die Integration dieser Daten ist in der Schicht *integration* angesiedelt. Die Interaktionen mit dem User sind in der Schicht *ui* und die Resultate werden in der Schicht *output* aufbereitet.

Die Softwarearchitektur ist an die „Clean Architecture“-Prinzipien von [44] angelehnt. Diese Architektur hat den Vorteil, dass die Kernlogik, die *business*-Schicht, keine Abhängigkeiten auf die die anderen Schichten hat. Diese Schicht ist somit absolut unabhängig. Durch das Exponieren von Interfaces für

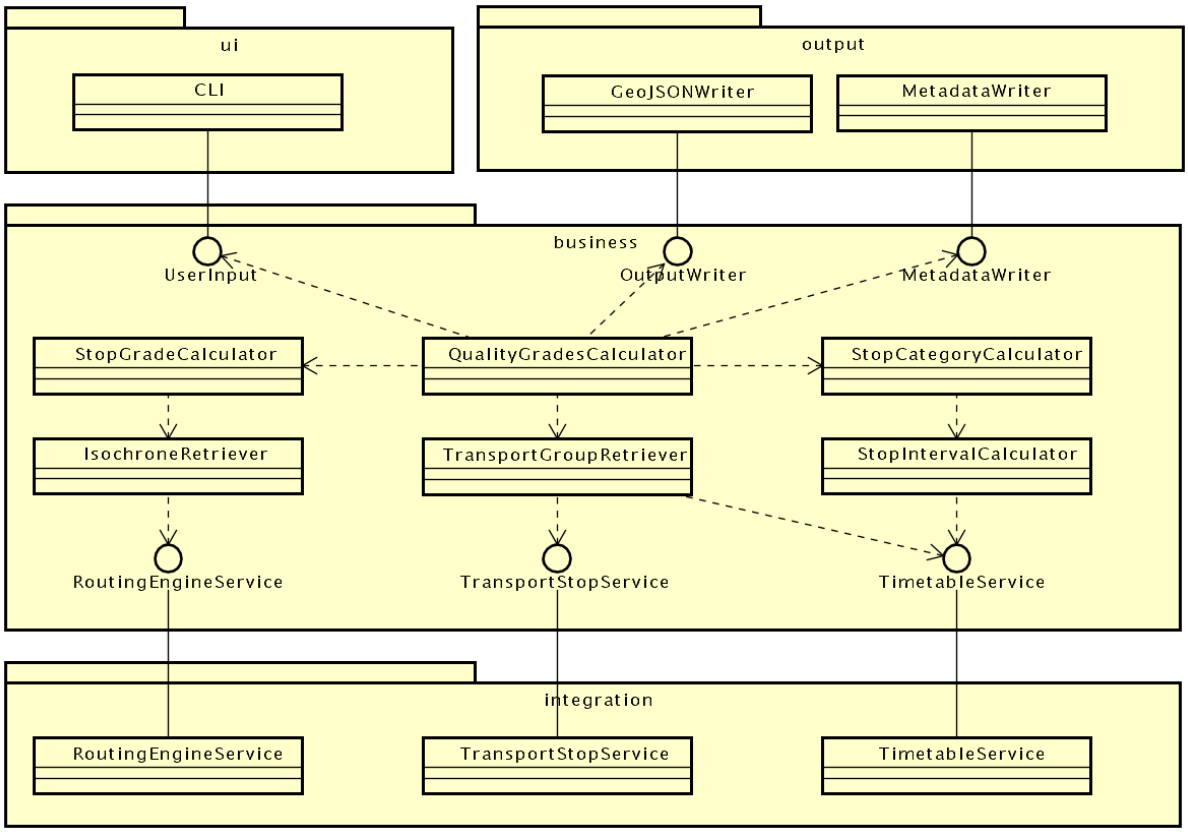

Abbildung 17: Schichten-Diagramm ÖV-Güteklassen Generator

die Schichten darüber und darunter können diese separat mit unterschiedlichen Dynamiken entwickelt werden. So entsteht beispielsweise der Anspruch nach einem zusätzlichen Output-Format öfters als nach einer Anpassung der Berechnung der ÖV-Güteklassen. Durch diese Trennung kann nun die Schicht output um einen zusätzlichen Service ergänzt werden, ohne dabei Gefahr zu laufen, dass die Kernlogik durch die Änderung unbemerkt vom bisherigen Verhalten abweicht. Die Kernlogik erwartet grundsätzlich nur eine Implementation, welche das bereitgestellte Interface implementiert. Wie sie dieses Interface implementiert ist der *business*-Schicht prinzipiell egal. Im Folgenden werden die einzelnen Schichten beschrieben.

business

In dieser Schicht ist die Kernlogik des Generators angesiedelt. Hier wird die Berechnung der ÖV-Güteklassen koordiniert. So wird die Konfiguration des Users entgegen genommen und die nötigen Vorverarbeitungen und Berechnungen angestossen. Durch die Natur der Berechnungslogik (siehe Abbildung 5) lässt sich die Logik analog strukturieren. Dabei wird die Haltestellekategorie, welche sich aus der Art des Verkehrsmittels und dem Kursintervall zusammensetzt, über den Service *StopCategoryCalculator* berechnet. Die ÖV-Güteklassen lassen sich in einem weiteren Schritt mithilfe des *TransportGroupRetriever* und *StopGradeCalculator* berechnen. *QualityGradesCalculator* übernimmt hier die Koordination der Berechnung.

integration

Alle Datenbankzugriffe erfolgen über die *integration*-Schicht. Es ist bewusst kein Object-Relational Mapping angedacht. Dies aus dem Grund, da intensiv mit Stored Procedures gearbeitet wird und vorallem Berechnungen auf der Datenbank durchgeführt werden und nicht direkt ein Datenmodell existiert, welches im Generator verwendet und in der Datenbank persistiert wird.

ui

In der ui-Schicht sind die Interaktionen mit dem User angedacht. So ist hier ein Command Line Interface (CLI) angesiedelt, welche Konfigurationen über eine einfache Bedienung entgegen nimmt.

output

Die Ergebnisse der ÖV-Güteklassen werden über diese Schicht dem User exponiert. Der Vorteil der oben beschriebenen „Plugin-Architektur“ macht sich hier stark sichtbar. Entsteht das Verlangen nach einem zusätzlichen Output der ÖV-Güteklassen, etwa als GeoPackage, kann problemlos ein Service *GeoPackageWriter* erstellt werden, welche das Interface *OutputWriter* implementiert und analog zum *GeoJSONWriter* das Resultat als GeoPackage ausliefert. Die Kernlogik in der business-Schicht muss so nicht angepasst werden.

Der *MetadataWriter* ist gesondert hervorzuheben. Durch diesen werden Konsumenten des Resultats des „ÖV-Güteklassen 2018 Generator“ darüber informiert, welche ÖV-Güteklassen generiert wurden, sprich für welche Stichtage und Zeitintervalle ein Ergebnis vorliegt. Dabei handelt es sich um eine JSON-Datei.

4.2.2 Backend

Das Backend nimmt die Ergebnisse der Berechnungen vom Container ÖV-Güteklassen 2018 Generator entgegen. Konkret heisst das, dass das Backend die Metadaten, welche vom Generator zusätzlich zu den ÖV-Güteklassen geschrieben werden, einliest und diese Daten über ein Web-API möglichen Clients zur Verfügung stellt. Das Web-API exponiert somit die Information, über welche Stichtage und Zeitintervalle ÖV-Güteklassen vorhanden sind.

Über das Backend werden nicht direkt die generierten ÖV-Güteklassen beispielsweise als GeoJSON ausgeliefert. Dies aus dem Grund, da diese Dateien eine Grösse haben, welche eine Auslieferung über ein Web-Application Programming Interface (API) nicht zumutbar machen. Für die Visualisierung auf einer Karte eignen sich andere Varianten, welche in den Container Web-App und Tile-Server angesprochen werden.

Web-API

Die Services, welche über das Web-API exponiert werden, sind in Tabelle 23 aufgeführt.

Route	Argument	Zweck
/typesOfDays	-	Retourniert die Stichtage für welche ÖV-Güteklassen vorhanden sind.
/gradings	typeOfDay	Retourniert die Metadaten für die ÖV-Güteklassen, welche für den übergebenen Stichtag (typeOfDay) vorhanden sind.

Tabelle 23: Web-API

4.2.3 Web-App

Die Web-App dient als Frontend des Systems „ÖV-Güteklassen 2018“. Die Applikation wird als SPA direkt an den Browser des Users ausgeliefert. Die Web-App nutzt das Web-API des Backend-Containers für das Beziehen von Metadaten, welche die verfügbaren ÖV-Güteklassen beschreiben. Mithilfe der Metadaten können die ÖV-Güteklassen vom Container Tile-Server bezogen und visualisiert werden.

Dafür wird, wie in der Evaluation (siehe Kapitel 3.2) festgehalten, *React* [24] eingesetzt.

4.2.4 Tile-Server

Der Tile-Server ist ein wichtiger Container, welcher von der Web-App genutzt wird. Durch diesen und die damit verbundenen Verwendung von Vector Tiles [31] wird eine enorme Performanzsteigerung und somit eine bessere Usability erreicht. Der Tile-Server war in einer ersten Version nicht angedacht. In dieser hat die Web-App die ÖV-Güteklassen direkt als GeoJSON vom Backend bezogen. Dies war auf zwei Ebenen problematisch. In einem ersten Schritt musste das GeoJSON vom Server geladen werden. Das GeoJSON für die ÖV-Güteklassen 2018 am Stichtag 13.11.2018 tagsüber hat eine Grösse von 65 MB. Dieses muss in voller Grösse geladen werden und das für jeden verfügbaren Stichtag, welcher ein Benutzer auswählt. Die Visualisierung des GeoJSON in dieser Grössenordnung ist extrem unflüssig und man wartet beim Verschieben der Karten mehrere Sekunden auf die Darstellung der ÖV-Güteklassen.

Diese Problematik ist mit der Verwendung von Vector Tiles und einem Tile-Server obsolet. Der Tile-Server liefert genau die Daten aus, welcher der Benutzer gerade betrachten möchte. Liegt der Fokus beispielsweise auf Rapperswil-Jona, werden die ÖV-Güteklassen nur in diesem Gebiet geladen und nicht für die ganze Schweiz. Die ÖV-Güteklassen werden gekachelt, was das Ausliefern eines Auszugs der Daten möglich macht. Je nach Zoomstufe ist es möglich, darzustellende Features zusammenzufassen und weniger oder mehr Detail in die Darstellung zu bringen. Somit kann die zu ladende Grösse der Vector Tiles und dadurch die Antwortzeit vom Server optimiert werden.

4.2.5 Routing-Engine

Die Routing-Engine ist für die Berechnung von Isochronen verantwortlich. Dazu werden zuerst die Kartendaten vom System planet.osm.ch eingelesen und zusammen mit dem Terrainmodell von swissALTI^{3D}, das zuvor vom Admin manuell besorgt wurde, einen Routing-Graphen erstellt.

Sobald der Admin die Berechnung der ÖV-Güteklassen auslöst, wird die Routing-Engine vom ÖV-Güteklassen 2018 Generator nach Isochronen abgefragt, die mit Haltestellen als Startpunkt und einer gewissen Distanz berechnet werden. Diese Isodistanzen bilden Polygone, die anschliessend vom Generator weiter verarbeitet werden.

5 Implementation

In diesem Kapitel wird die Implementation der einzelnen Container beschrieben, wie sie im Kapitel 4.2 definiert wurden.

5.1 ÖV-Güteklassen 2018 Generator

5.1.1 Datenaufbereitung

Der ÖV-Güteklassen 2018 Generator operiert auf unterschiedlichen Daten von externen Datenquellen. Diese Daten kommen in den verschiedensten Formaten daher und müssen für eine weitere Verarbeitung gefiltert, aufbereitet und optimiert werden. Für eine einfache Handhabung wurde ein Docker-Setup gewählt. Die Bedienung dieses Setups ist im Kapitel 10 ausführlich beschrieben.

Grundsätzlich kann man sagen, dass zwei Docker-Container existieren, namentlich *tooling* und *db*. Der Docker-Container *tooling* nimmt Daten entgegen oder bezieht diese von externen Diensten und bereitet sie so vor, dass diese vom Docker-Container *db* in die Datenbank *oevgk18* für eine Weitereverwendung des ÖV-Güteklassen 2018 Generators geladen werden können. Der Docker-Container *db* sorgt ebenfalls dafür, dass beim Starten die nötigen Extensions (pgrouting, hstore, ...) aktiviert sind und zuerst die Schemas und danach die Stored Procedures erstellt werden. Zusätzlich müssen nach den einzelnen Imports, welche nachfolgend beschrieben werden, zusätzliche Modifaktionen am Schema vorgenommen werden. Diese Modifikationen werden in einem Post-Import-Schritt durchgeführt, weil die Terrain- sowie die OSM-Daten durch Tools importiert werden, welche ein Schema vorgeben und somit bereits vorhandene Schemas überschrieben werden. Ebenfalls werden zusätzliche Indizes erstellt.

Im weiteren sind die Services, welche die beiden Docker-Container bereitstellen, anhand eines Daten-

flussdiagramms aufgeschlüsselt. Der ÖV-Güteklassen 2018 Generator operiert grundsätzlich auf drei externen Datenquellen, namentlich sind das die Fahrplandaten (nachfolgend *gtfs data*), das Terrainmodell (nachfolgend *terrain data)* und die OSM-Daten (nachfolgend *osm data*). Für jede Datenquelle existiert ein Service im Docker-Container *tooling* und ein Pendant im Docker-Container *db*, welche die vorverarbeiteten Daten des Docker-Containers *tooling* entgegennimmt und in die Datenbank spielt.

OSM-Daten

OSM-Daten werden von der Routing-Engine für das Berechnen von Isochronen und für das Identifizieren von ÖV-Haltestellen verwendet. In Abbildung 18 ist ersichtlich, wie der Docker-Container *tooling* die PBF-Datei der Schweiz von einem externen Dienst [43] bezieht und für die zwei vorhin erwähnten Zwecke vorbereitet. Es besteht ebenfalls die Möglichkeit, eine lokale OSM-Datei zu verwenden. Dies ist vorallem für Testing-Zwecke vorteilhaft, da auf kleineren Ausschnitten leichter getestet werden kann.

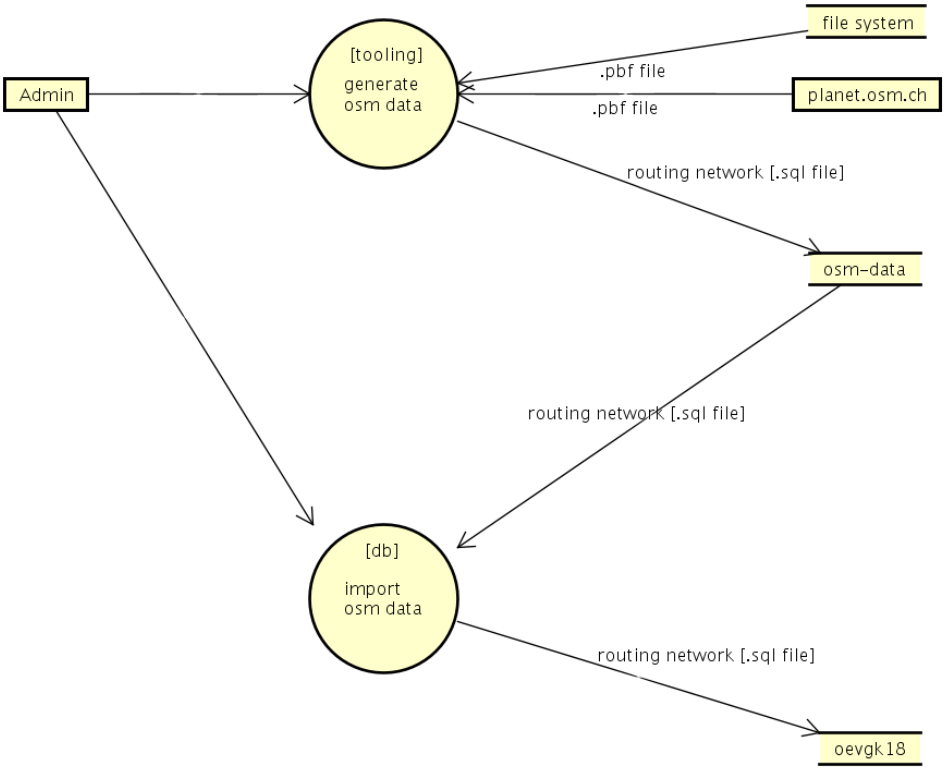

Abbildung 18: Datenfluss Setup OSM-Daten

Der Routing-Graph wird mithilfe des Tools *OSM2PO* [45] für pgRouting aufbereitet und als SQL-Datei bereitgestellt. *OSM2PO* wird durch eine separate Konfiguration für unsere Zwecke konfiguriert, so dass die Routen für Fussgänger optimiert werden.

GTFS-Daten

Die Fahrplandaten werden im Docker-Container *tooling* automatisch von geOps [41] im GTFS-Format [42] heruntergeladen und im CSV-Format als TXT-Dateien bereitgestellt. Diese wiederum

können in einem nächsten Schritt durch den Docker-Container *db* in die Datenbank gespielt werden. Der Datenfluss ist in Abbildung 19 ersichtlich.

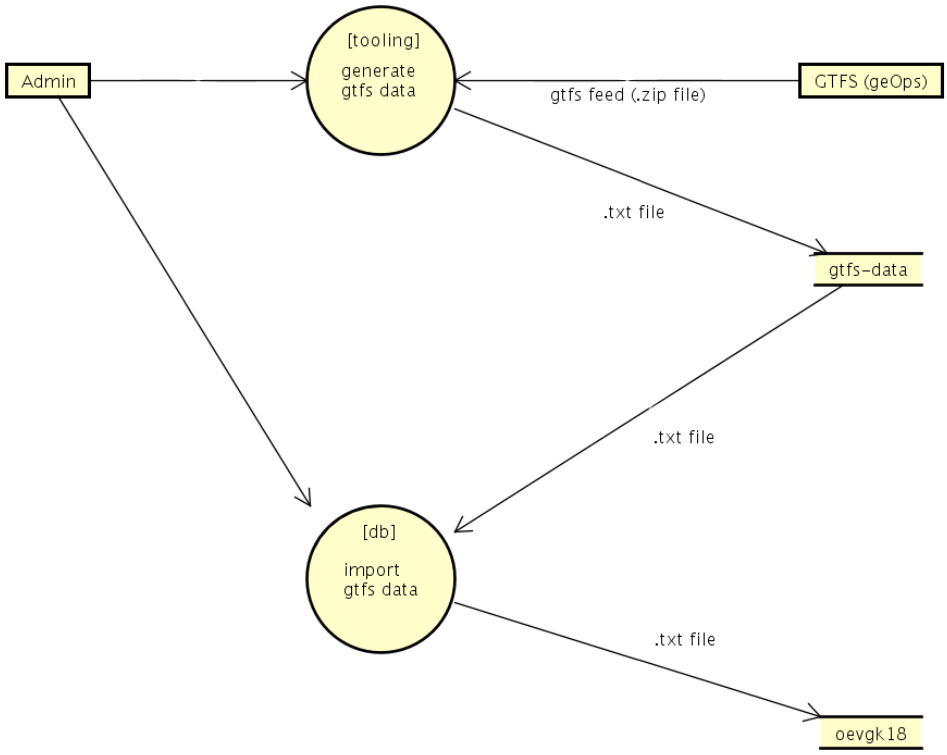

Abbildung 19: Datenfluss Setup GTFS-Daten

Terrain-Daten

Das Terrainmodell kann aufgrund der Grösse der Datei und aus lizenztechnischen Gründen nicht von einem externen Dienst bezogen und muss somit dem Service manuell als TIF-Datei bereitgestellt werden. Diese Datei wird mit raster2pgsql [46] verarbeitet, einem Tool von PostGIS, welches Raster-Formate in ein Format konvertiert, welches in eine PostGIS-Tabelle geladen werden kann. Dabei wird von der üblichen Struktur (ein Service im Docker-Container *tooling* und ein Pendant im *db*) abgewichen. raster2pgsql ist darauf ausgelegt, direkt in eine PostgreSQL-Datenbank zu schreiben. Der Weg über eine Intermediate-Datei wäre somit eine überflüssige Indirektion. Somit entfällt hier der Service im Docker-Container *tooling*. Der Datenfluss ist in Abbildung 20 ersichtlich.

5.1.2 Umsetzung Spezifikation

Datenfluss

Durch die unterschiedliche Natur der Datenquellen und der Menge dieser ist es hilfreich, zu visualisieren, zu welchem Zeitpunkt welche Daten zu welchem Zweck benötigt werden. Wie in Abbildung 5 ersichtlich ist, werden die ÖV-Güteklassen in fünf Schritten berechnet. In Abbildung 21 ist nun schematisch die Raison d'Être der Datenquellen ersichtlich. Die Pfeilrichtung beschreibt den Datenfluss.

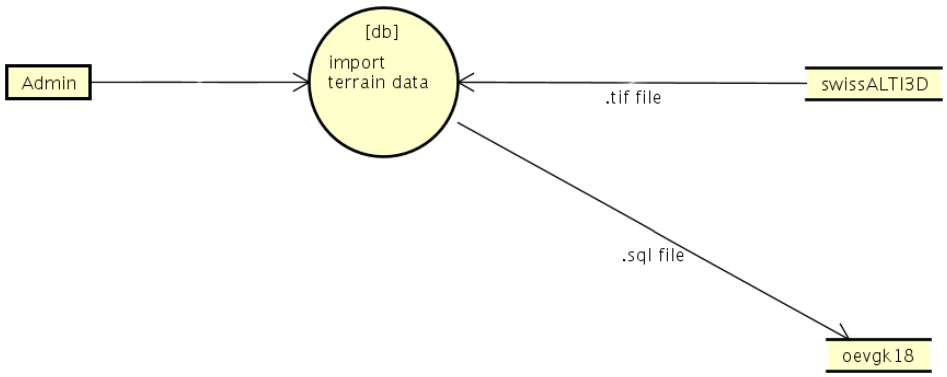

Abbildung 20: Datenfluss Setup Terrain-Daten

Die Datenquellen *GTFS*, *OSM* und *Terrain-Daten* entsprechen an sich keinen separaten Datenquellen, sondern werden nur zum besseren Verständnis getrennt dargestellt. Wie wir in Kapitel 5.1.1 gesehen haben, werden diese aufbereitet und in einer Datenbank *oevgk18* gehalten.

Nachfolgend werden die fünf Berechnungsschritte separat beschrieben.

Berechnung der Art der Verkehsmittel

Die öffentlichen Verkehrsmittel werden in drei Verkehrsmittelgruppen gruppiert (siehe Kapitel 3.2.1 im Teil I). Wie in Kapitel 4.1.2 beschrieben werden die Fahrplandaten im GTFS-Format [42] gehalten. Es werden dabei 8 Verkehrsmittel-Typen unterschieden. In Tabelle 24 ist ein Mapping der definierten GTFS Route Typen zu den Verkehrsmittelgruppen ersichtlich.

GTFS Route Type	Verkehrsmittelgruppe
0 (Tram, Streetcar, Light rail)	C
1 (Subway, Metro)	A/B
2 (Rail)	A/B
3 (Bus)	C
4 (Ferry)	C
5 (Cable car)	C
6 (Gondola, Suspended cable car)	C
7 (Funicular)	C

Tabelle 24: Mapping GTFS Route Type Verkehrsmittelgruppe

Die Herausforderung beim Berechnen der Art der Verkehrsmittel ist die Bestimmung der Bahnknoten. Die Definition des Bahnknoten ist im Kapitel 3.2.1 genau spezifiziert. Für die Klassifizierung eines Bahnknoten ist die Anzahl der Richtungen, in welche man von einer Bahnstation aus verkehren kann, relevant.

In Listing 1 ist ersichtlich, wie für eine spezifizierte Auswahl an Haltestellen (*relevant_stops*) die Anzahl erreichbaren Haltestellen gezählt wird. Im abgebildeten Query wird geprüft, ob der aktuelle Stop bei einem anderen *Trip* als vorherige Haltestelle vorkommt. Die Anzahl wird mit der Window Function

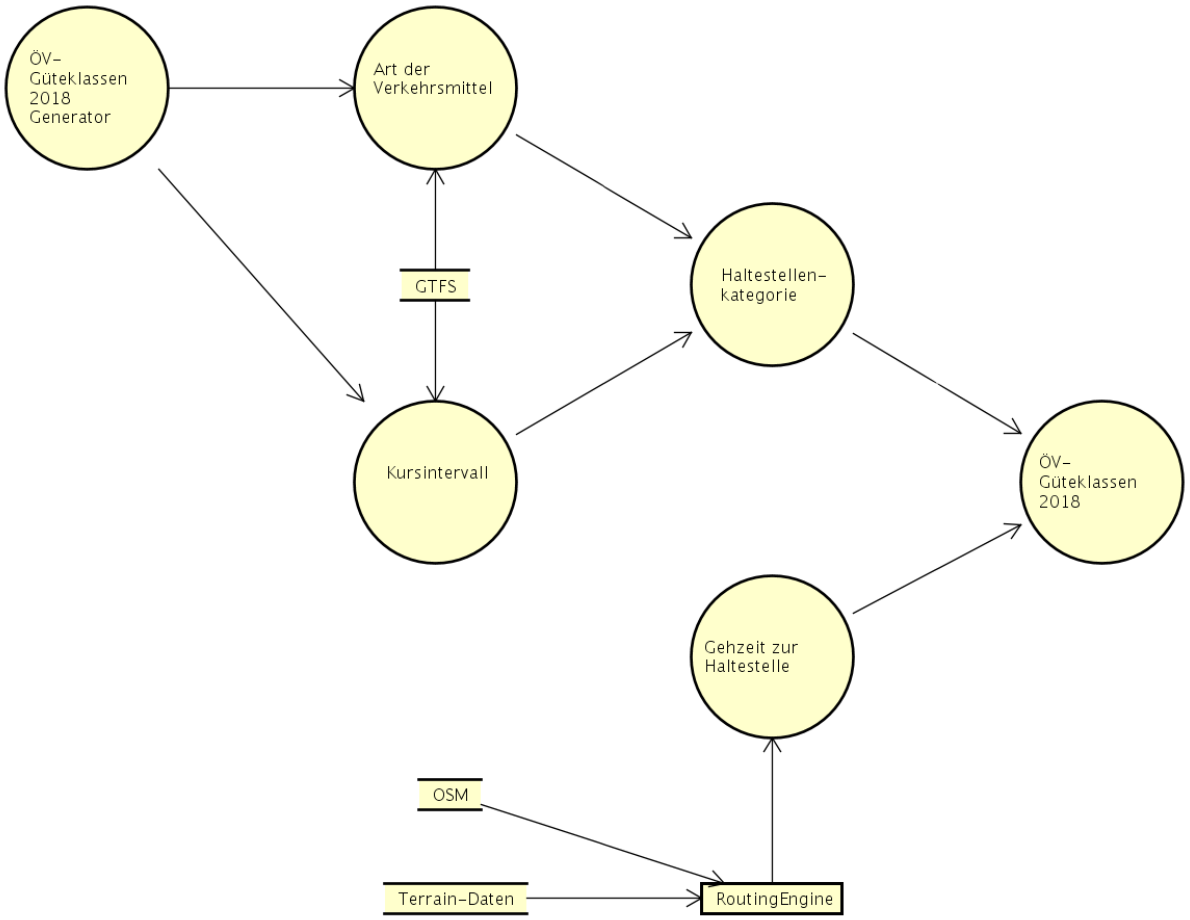

Abbildung 21: Datenfluss ÖV-Güteklassen 2018 Generator

gesammelt und gruppiert für jede Haltestelle retourniert.

Die abgebildete Berechnung hat mehrere Ver- und Verschlimmbesserungen durchlaufen. Es lohnt sich einen ausführlicheren Blick auf die Evolution der Abfragetechnik zu legen, da diese Hoch und Tiefs durchlebt hat und einige Erkenntnisse daraus gezogen wurden.

1. Version In einer ersten Version war angedacht, dass das Berechnen der Anzahl erreichbaren Haltestellen für jede einzelne Haltestelle separat aufgerufen wird.

Das Datenmodell von GTFS zwingt einem zu einem Join über mehrere grosse Tabellen, welcher mehrmals durchgeführt werden muss. In Listing 2 ist ersichtlich, wie in einer Common Table Expression dieser für jede Haltestelle durchgeführt wird. Dieser Join bildet für eine Station alle *Trips* mit der Nummer des akutellen *Stops* auf einem *Trip* ab. Dieser Join ist sehr zeit- und ressourcenintensiv. Diese Abfrage dauert pro Haltestelle 11.9 Sekunden, was bei momentan akuellen 1815 Bahnhaltestellen extrem ins Gewicht fällt und nicht zumutbar ist.

```sql
WITH relevant_stops AS (
    SELECT unnest(:relevant_stops) AS uic_ref
),
next_station_mapping AS (
    SELECT DISTINCT s.stop_name, t.trip_id, st.stop_sequence, s.uic_ref
    FROM stops s
        INNER JOIN stop_times st ON s.stop_id = st.stop_id
        INNER JOIN trips t ON st.trip_id = t.trip_id
        INNER JOIN routes r ON t.route_id = r.route_id
    WHERE r.route_type = 2 OR r.route_type = 1
)
SELECT
    DISTINCT nsm1.uic_ref,
    COUNT(nsm2.stop_name) OVER (PARTITION BY nsm1.uic_ref)
    FROM relevant_stops
    LEFT JOIN next_station_mapping nsm1 ON relevant_stops.uic_ref = nsm1.uic_ref
    INNER JOIN next_station_mapping nsm2 ON nsm1.trip_id = nsm2.trip_id
WHERE
    nsm1.stop_sequence = (nsm2.stop_sequence - 1)
GROUP BY nsm1.uic_ref, nsm2.stop_name;
```

Listing 1: SQL-Query zur Bestimmung der Anzahl erreichbaren Haltestellen (finale Version)

2. Version In einer zweiten Version war die Überlegung, den Join zu persistieren, damit nicht jede Abfrage diese aufwändigen Join durchführen muss und Indizes darauf zu generieren.

Der Grund, warum in diesem Fall nicht eine temporäre Tabelle in Frage kommt, ist in der PostgreSQL-Dokumentation beschrieben: „The autovacuum daemon cannot access and therefore cannot vacuum or analyze temporary tables.“ [47] Es ist auch nicht möglich den Daemon manuell anzustossen, da das Erstellen der temporären Tabelle und das Starten des Daemons aus einer PostgreSQL-Funktion nicht in der selben Session durchgeführt werden kann, was den Nutzen der temporären Tabelle zunichte macht, denn damit die Indizes greifen, ist dieser Prozess relevant. Das Persistieren des Join in einer persistenten Tabelle (*next_station_mapping*) und das Anlegen der Indizes bringen in diesem Fall eine enorme Performanzsteigerung, wie in Tabelle 25 ersichtlich ist, mit der Annahme, dass Autovacuum lief.

	temporäre Tabelle	**Tabelle**
Erstellen der Tabelle	6.7 s	7.3 s
Erstellen der Indizes	-	3.9 s
Abfrage	n * 4.8 s	n * 0.062 s

Tabelle 25: Performanzvergleich Temp Table und Table mit Index

3. Version Um die Datenbank nicht mit fast 2000 Abfragen für alle Bahnhaltestellen zu belasten, entstand die Idee, für alle Haltestellen in einer Abfrage die erreichbaren Haltestellen zu zählen.

```
CREATE OR REPLACE FUNCTION get_next_stations(uic_ref INTEGER)
    RETURNS TABLE(stop_name text) AS $$
BEGIN
RETURN QUERY
    WITH next_station_mapping AS (
        SELECT DISTINCT s.stop_name, t.trip_id, st.stop_sequence, s.uic_ref
            FROM stops s
                INNER JOIN stop_times st ON s.stop_id = st.stop_id
                INNER JOIN trips t ON st.trip_id = t.trip_id
                INNER JOIN routes r ON t.route_id = r.route_id
            WHERE r.route_type = 2 OR r.route_type = 1
    )
    SELECT nsm2.stop_name FROM next_station_mapping nsm1
                  INNER JOIN next_station_mapping nsm2
                      ON nsm1.trip_id = nsm2.trip_id
                  WHERE nsm1.uic_ref = rev_uic_ref AND
                      nsm1.stop_sequence = (nsm2.stop_sequence - 1)
                      GROUP BY nsm2.stop_name;
END;
$$ LANGUAGE PLPGSQL
```

Listing 2: SQL-Query zur Bestimmung der Anzahl erreichbaren Haltestellen (Version 1)

Dazu wurde die Abfrage in die Version umgebaut, welche in Listing 3 ersichtlich ist. Die Abfrage läuft in 9.8 Sekunden durch und ist im Vergleich zur Lösung aus Version 2 für momentan 1815 vorhandenen Bahnhaltestellen um über 100 Sekunden schneller.

Finale Version Da nun die Berechnung nicht mehr für alle Bahnhaltestellen einzeln durchgeführt wird, lohnt sich die Analyse, ob sich die Erstellung der Tabelle (*next_station_mapping*) und der Indizes im Vergleich zum Join in einer Common Table Expression noch lohnt.

	Join in Tabelle	**Join in CTE**
Erstellen der Tabelle	7.3 s	-
Erstellen der Indizes	3.9 s	-
Abfrage	9.8 s	n * 0.062 s
Total	**21 s**	**17.5 s**

Tabelle 26: Performanzvergleich Join in Tabelle und Join in CTE

In der Tabelle 26 ist das Ergebnis ersichtlich. Die Performanzverbesserung ist nicht immens, jedoch wird die Verbesserung übernommen, da in dieser Variante nicht zusätzliche eine Tabelle angelegt werden muss, welche nur für einen Join benötigt wird und sonst nichts mit der Domain zu tun hat.

Die finale Version war bereits in Listing 1 ersichtlich.

```
WITH relevant_stops AS (
    SELECT unnest(:relevant_stops) AS uic_ref
)
SELECT
    DISTINCT nsm1.uic_ref,
    COUNT(nsm2.stop_name) OVER (PARTITION BY nsm1.uic_ref)
    FROM
        relevant_stops
    LEFT JOIN next_station_mapping nsm1 ON relevant_stops.uic_ref = nsm1.uic_ref
    INNER JOIN next_station_mapping nsm2 ON nsm1.trip_id = nsm2.trip_id
WHERE
    nsm1.stop_sequence = (nsm2.stop_sequence - 1)
GROUP BY
    nsm1.uic_ref, nsm2.stop_name;
```

Listing 3: SQL-Query zur Bestimmung der Anzahl erreichbaren Haltestellen (Version 3)

Fazit Optimierungen, welche in der 2. und 3. Version Sinn gemacht hatten, waren in der finalen Version nicht mehr notwendig. Es hat sich gezeigt, dass es sich lohnt, vergangene Annahmen zu überdenken und zu prüfen, ob diese auf die aktuelle Situation immer noch zu treffen. In Tabelle 27 sind die Resultate der Optimierungen zusammengefasst.

Version	Dauer
Version 1	360 min
Version 2	112.53 s
Version 3	21 s
Finale Version	**17.5 s**

Tabelle 27: Performanzvergleich der verschiedenen Version

Integration der Bahnhöfe mit Anschluss an den Fernverkehr In der Spezifikation zur Ermittlung von Bahnknoten (Teil I Kap. 3.2.1) wird verlangt, dass auch diejenigen Bahnhöfe als Bahnknoten zu identifizieren sind, die einen Anschluss an den nationalen Fernverkehr haben. Aus den Fahrplandaten ist nicht zu erkennen, welche Züge als Fernverkehr gelten.

Stattdessen werden die Fernverkehrskonzessionen genutzt, die vom Bundesamt für Verkehr (BAV) vergeben werden. Da zum Zeitpunkt dieser Arbeit (Juni 2018) allein die SBB den Auftrag zur Erschliessung des Fernverkehrs in der Schweiz besitzt, ist die Konzession PBK 0584 [48] relevant. Dort sind alle Bahnhöfe aufgelistet, die garantiert mit dem Fernverkehr erschlossen werden. Damit werden sie mit unserer Spezifikation als Bahnknoten identifiziert.

Der Datensatz der Fernverkehrskonzessionen ist leider nicht in einem maschinen-lesbaren Format vorhanden. Die Bahnhöfe wurden daher aus der Liste manuell in ein CSV extrahiert [49], das in die Datenbank eingelesen wird.

Berechnung des Kursintervalls

Für die Berechnung des Kursintervalls wurde die Formel, wie sie in der Spezifikation (siehe Kapitel 3.2.2) definiert wurde, direkt umgesetzt. Um alle Abfahrtszeiten einer Haltestelle zu erhalten, wurde zuerst eine SQL-Query für die GTFS-Datenbank formuliert, in der nach der UIC-Referenz gefiltert wird. Die Ausführung dieser Query dauerte allerdings ca. 1.7 Sekunden pro Haltestelle, was bei einer Gesamtzahl von 28'000 Haltestellen etwa 11 Stunden dauern würde.

Viel effizienter ist es, die Abfahrtszeiten für alle Haltestellen in einer einzelnen Abfrage zu aggregieren. Diese ist in Listing 4 abgebildet. Die optimierte SQL-Query dauert ca. 7 Sekunden für alle 28'000 Haltestellen, was eine drastische Verbesserung darstellt.

```
WITH calendar_trip_mapping AS (
    SELECT st.departure_time, s.uic_ref
    FROM stop_times st
    INNER JOIN stops s ON st.stop_id = s.stop_id
    INNER JOIN trips t ON st.trip_id = t.trip_id
    INNER JOIN calendar_dates c ON t.service_id = c.service_id
    WHERE c.date = :date
)
SELECT uic_ref, array_agg(departure_time) AS departure_times
FROM calendar_trip_mapping
GROUP BY uic_ref
```

Listing 4: Effiziente SQL-Query zur Abfrage aller Abfahrtszeiten an einem bestimmten Tag

Berechnung der Haltestellenkategorie

Die Haltestellenkategorie setzt sich zusammen aus der Einteilung einer Haltestelle in die Verkehrsmittelgruppe (Art der Verkehrsmittel) und dem Kursintervall. Die Übersetzung dieser Werte in die Haltestellenkategorien I – VII erfolgt nach der Tabelle in der Spezifikation (Kapitel 3.2.3 im Teil I). In der Konfigurationsdatei können diese Parameter allerdings frei angepasst werden.

Berechnung der ÖV-Güteklassen

Die ÖV-Güteklassen einer einzelnen Haltestelle wird durch Isochronen beschrieben, die jeweils eine Fläche bilden, von der die Haltestelle innerhalb einer maximalen Gehzeit erreichbar ist. Jeder dieser Isochronen wird eine ÖV-Güteklasse von A – F zugewiesen. Welche Isochronen für eine Haltestelle erzeugt werden und welche Güteklasse ihnen zugewiesen wird, kann aus der Tabelle 12 der Spezifikation (Kap. 3.2 im Teil I) entnommen werden.

Snapping von der Haltestelle auf den Routing-Graph Die Haltestellen stammen aus dem Datenstamm der SBB [8] und sind somit nicht Teil des Routing-Graphen. Um die Isochronen berechnen zu können, muss auf dem Routing-Graphen zuerst ein Startpunkt ermittelt werden, von dem das Routing entlang der Strassen und Wege ausgehen kann. Dieser Punkt soll möglichst nahe an der eigentlichen

Haltestelle liegen. Es ist anzumerken, dass unsere Fahrplandaten lediglich ein geographischer Punkt pro Haltestelle definiert und so Buskanten oder Gleise nicht genau abbilden.

Ein erster Ansatz dieses „Snappings“ sucht mithilfe einer „Nearest Neighbor Suche“ den geographisch nächstliegenden Vertex des Routing-Graphen zu den Koordinaten der Haltestelle. Dies ist bei regulären Bushaltestellen mit zwei Kanten ein guter Ansatz, da diese Koordinaten meist neben oder direkt auf der Strasse liegen. Problematisch wird dies aber etwa bei grossen Bahnhöfen wie dem Zürcher Hauptbahnhof. Die Koordinate des Bahnhofs liegt in der Mitte der Bahnhofshalle, welche als Fussgängerfläche erfasst ist. Da solche Flächen nicht im Routing-Graphen abgebildet werden, liegt der nächste Vertex des Routing-Graphen auf einer Rolltreppe, die nicht mit dem restlichen Graphen verbunden ist (siehe Abbildung 22). Dieses Snapping auf einen isolierten Graphen führt dazu, dass die Routing-Engine nie ausserhalb des Bahnhofs routen kann, dadurch können auch keine Isochronen erzeugt werden.

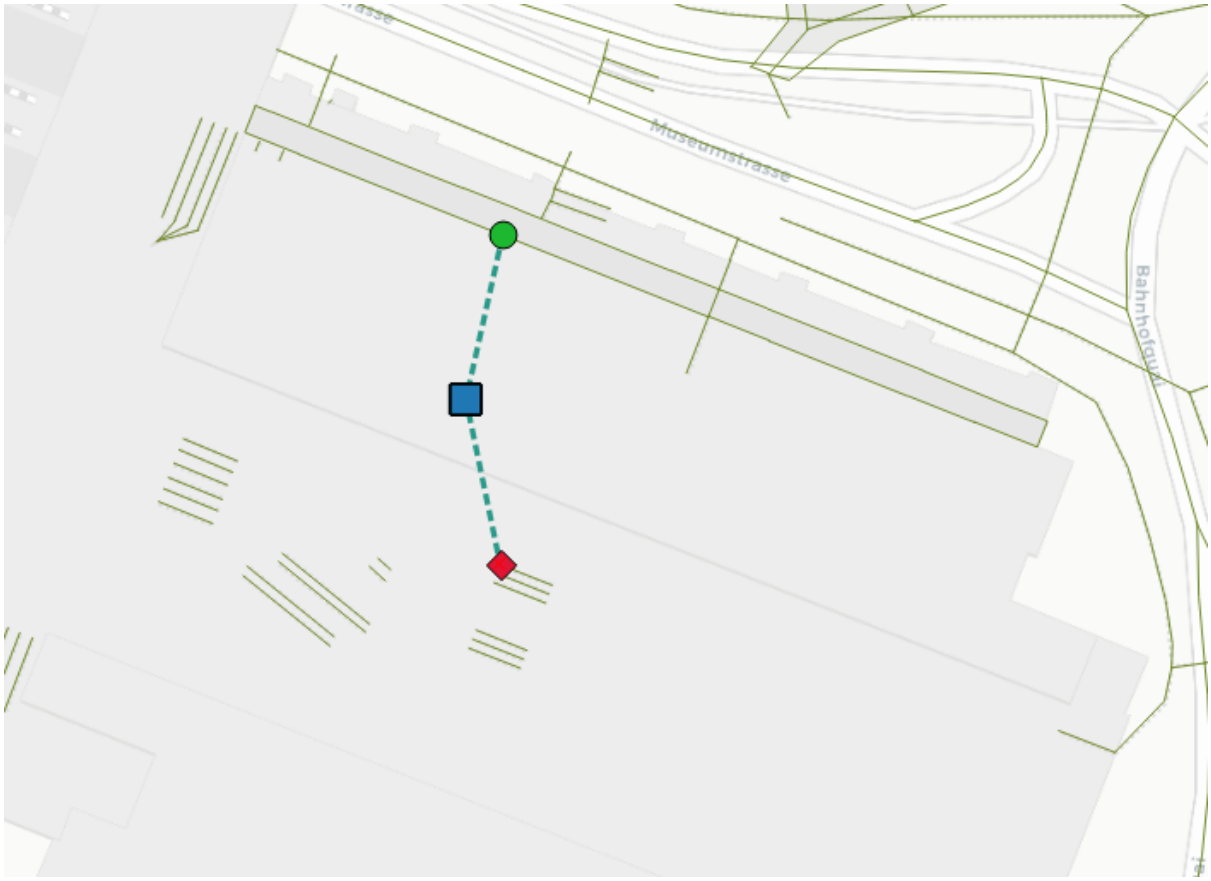

Abbildung 22: Die Haltestelle (blaues Quadrat) wird mit dem einfachen Ansatz zur Rolltreppe „gesnapped“ (Roter Diamant), während die Optimierung (Grüner Punkt) ein Ergebnis erzeugt, dass mit dem restlichen Routing-Netz verbunden ist.

Ein optimierter Ansatz nimmt nicht direkt den nächstliegenden Punkt auf dem Routing-Graphen, sondern prüft, ob dieser ermittelte Vertex auf einem isolierten Segment des Graphen liegt. Dafür werden mit Hilfe der Routing-Engine alle Punkte gesucht, die von diesem Vertex aus in 200 Metern Fussweg erreichbar sind. Wenn es einen Punkt gibt, der mindestens 150 Meter entfernt vom Startpunkt ist, wird dies als genügend bewertet, um den Startpunkt für die Erzeugung von Isochronen zu verwenden. Wenn diese Bedingung nicht erfüllt ist, werden iterativ andere Vertices geprüft (sortiert nach

der Distanz zur Haltestelle), bis ein passender Vertex gefunden wird. Der Code dazu ist in Listing 5 ersichtlich.

```
FOR nearest_vertex IN
  WITH nearest_bboxes AS (
      SELECT id, the_geom
      FROM routing_segmented_vertices_pgr v
      -- get nearest points in approximation
      ORDER BY the_geom <#> ST_SetSRID(ST_MakePoint(stop_lon, stop_lat), 4326)
      LIMIT 100
  )
  SELECT id
  FROM nearest_bboxes
  -- order by nearest points (precise)
  ORDER BY ST_Distance(
    the_geom,
    ST_SetSRID(ST_MakePoint(stop_lon, stop_lat), 4326))
LOOP
  SELECT max(agg_cost)
  INTO max_distance
  -- Get all points reachable in 200 Meters
  FROM pgr_drivingDistance(
      'SELECT id, source, target, cost FROM edge_preselection',
      nearest_vertex,
      200,
      FALSE);

  IF max_distance >= 150
  THEN
    RETURN nearest_vertex;
  END IF;
END LOOP;
```

Listing 5: SQL Stored Procedure für das „Snapping“ der Haltestelle auf den Routing-Graph

Das Ergebnis für den Zürcher Hauptbahnhof ist in Abbildung 22 zu sehen. Dieser Startpunkt ist noch immer nicht optimal, da ein Fussgänger auch ein Ausgang auf der anderen Seite des Bahnhofes wählen könnte. Für eine optimale Bestimmung müsste der Routing-Graph mit zusätzlichen Wegen ergänzt werden. Fussgängerflächen könnten beispielsweise durch eine Vorverarbeitung durch *PlazaRoute* [2] in den Routing-Graph integriert werden. Dabei werden zusätzliche Graph-Kanten eingefügt.

Vorbereitung des Routing-Graphen Zum Zeitpunkt des Imports mit *OSM2PO* besteht ein vollständiger Routing-Graph, auf dem bereits Routen berechnet werden können. Für die Berechnung von Isochronen ist dieser Routing-Graph aber nicht geeignet, da jeder Strassenabschnitt von einer Verzweigung zur nächsten als eine einzige Kante abgebildet wird. Werden nun Isochronen erstellt, beachtet der Algorithmus nur Vertices, die komplett in der definierten Zeit erreichbar sind. Dadurch werden Strassen ausgeschlossen, die innerhalb der Zeit nur bis zu einem Teil der Strecke erreichbar sind. Dies

führt zu ungenauen Isochronen, wie in Abbildung 23 ersichtlich ist.

Um diesen Effekt zu vermindern, wird vor der Berechnung der komplette Routing-Graph segmentiert. Dazu werden alle Kanten in einzelne Segmente geteilt, die maximal 30 Meter lang sind. Anschliessend muss die Topologie erneut berechnet werden, um die Verbindungen von Kanten und Vertices zu aktualisieren. Die daraus resultiertenden Isochronen sind deutlich genauer und bilden ein realistischeres Einzugsgebiet ab (siehe Abbildung 23).

Abbildung 23: Mit der Segmentierung des Routing-Graphen (unten) ergeben sich gegenüber des regulären Graphen (oben) höher aufgelöste Isochronen

Als weiteren Schritt der Vorbereitung werden die Terrain-Daten in die Kosten der Kanten eingerechnet. Dazu wird die Formel aus der Spezifikation für Leistungskilometer (siehe 3.2.4) verwendet. Für jede Kante werden die Kosten in beide Richtungen berechnet, da Steigung und Gefälle unterschiedlich gewichtet werden.

Berechnung der Isochronen In einem ersten Schritt wird ermittelt, welche Isochronen für die gegebene Haltestelle erzeugt werden müssen. Dabei wird die Gehzeit mithilfe der konfigurierten Laufgeschwindigkeit in Meter umgerechnet, da die Kosten des Routing-Graphen ebenfalls in Meter abgebildet sind.

Anschliessend werden die Isochronen berechnet. Dieser Algorithmus besteht aus zwei Teilen. Als erstes werden mit dem Dijkstra-Algorithmus [50] alle Vertices ermittelt, die von der Haltestelle aus in der

festgelegten maximalen Laufdistanz im Routing-Graph erreichbar sind (siehe Listing 6). Die Richtung der Kanten wird dabei umgekehrt, da der Dijkstra-Algorithmus die Strecke ausgehend der Haltestelle berechnet. Für unsere Zwecke ist es aber realistischer, den Laufweg zu berechnen, wenn man zur Haltestelle hin läuft.

```
CREATE TEMP TABLE distances ON COMMIT DROP AS (
SELECT
    vertices.id,
    vertices.geom_vertex AS point,
    isochron.agg_cost as distance
-- get all vertices that are reachable within the max boundary
FROM pgr_drivingDistance(
            'SELECT id, source, target, cost, reverse_cost FROM edge_preselection',
            start_vertex,
            max_boundary,
            TRUE
        ) AS isochron
    INNER JOIN vertex vertices ON isochron.node = vertices.id
);
```

Listing 6: Mit einem Dijkstra-Algorithmus werden bis zur Maximaldistanz alle erreichbaren Vertices gesucht (Auszug)

Vorgängig werden mit einer Query alle Kanten gesucht, die sich in einem Radius von 1300 Metern der Haltestelle befinden (siehe Listing 7). Dies, weil keine Isochronen berechnet werden, die einen grösseren Radius als 1300 Meter einschliessen. Dadurch verringert sich die Anzahl Kanten enorm, die pgRouting mit dem Dijkstra-Algorithmus abarbeiten muss.

```
CREATE TEMP TABLE edge_preselection ON COMMIT DROP AS (
SELECT
    id,
    source,
    target,
    effort AS reverse_cost,
    -- reverse directionality to simulate walking towards the public transport stop
    reverse_effort as cost
FROM routing r
-- select edges within ~1300 meters from the start vertex
WHERE ST_intersects(ST_Buffer(start_vertex_geom, 0.013), r.geom_way)
);
```

Listing 7: Mit einer effizienten Index-Suche werden alle Kanten in der Nähe der Haltestelle ermittelt (Auszug)

Im zweiten Schritt wird mit diesen Vertices eine Alpha Shape [51] erzeugt (siehe Listing 8). Dieser Algorithmus bildet das kleinst mögliche Polygon, wobei alle gefunden Vertices darin enthalten sein müssen. Diese Alpha Shape stellt die Isochrone dar.

```
RETURN QUERY
-- only select results with at least 3 points to form a polygon
WITH relevant_bounderies AS (
    SELECT boundary
    FROM unnest(boundaries) boundary, distances d
    WHERE d.distance <= boundary
    GROUP BY boundary
    HAVING count(d.distance) >= 3
)
SELECT
boundary,
polygon_geom AS polygon
FROM relevant_bounderies,
    -- create an alpha shape with a buffer
    LATERAL st_buffer(
        pgr_pointsAsPolygon(
            'SELECT id::integer, ST_X(point)::float AS x, ST_Y(point)::float AS y '
            || 'FROM distances WHERE distance <= ' || boundary || ';',
            0.00005
        ), 0.0001) AS polygon_geom
WHERE polygon_geom IS NOT NULL;
```

Listing 8: Mit den erreichbaren Vertices wird eine Alpha Shape erzeugt (Auszug)

Der erste Schritt muss pro Haltestelle nur ein Mal für die grösste zu berechnende Distanz durchgeführt werden, für die Erzeugung der Isochronen für kürzere Distanzen wird das Ergebnis des Dijkstra-Algorithmus gefiltert und nur die Vertices beachtet, die innerhalb dieser Distanz von der Haltestelle erreichbar sind. Auf die einzelnen Isochronen wird ein zusätzlicher Buffer angewendet, um die scharfen Ecken etwas abzurunden. Dadurch wird bewusst etwas Unschärfe hinzugefügt, um nicht die Illusion zu erwecken, dass die Isochrone das Einzugsgebiet exakt abbildet, sondern nur eine Approximation dessen darstellt.

Ausgabe des Ergebnisses

Nach der Berechnung der ÖV-Güteklassen liegt für jede Haltestelle, die mindestens eine Basiserschliessung hat, ein oder mehrere Isochronen vor. Diese gilt es nun im GeoJSON Format auszugeben. Dabei wird für jeden der unterschiedlichen Stichtage eine separate Datei erstellt.

Für jede Isochrone wird ein GeoJSON-Feature erstellt, wobei die ÖV-Güteklasse und die *uic_ref* der Haltestelle als zusätzliche Attribute mitgegeben werden. Anschliessend werden alle Features sortiert nach ÖV-Güteklassen, wobei die tieferen Klassen zuerst eingefügt werden. Dies hat später für die Visualisierung eine Auswirkung. Weil die Isochronen der höheren Klassen diejenigen der tieferen überlagern, sollten bei der Visualisierung die tieferen Klassen zuerst gerendert werden. Durch die Reihenfolge wird dies bereits in diesem Schritt sichergestellt.

Zusätzlich zu den ÖV-Güteklassen wird eine GeoJSON-Datei mit den Punkten aller Haltestellen erstellt,

um sie später mit der Web-Applikation auf der Karte anzeigen zu können. Ebenfalls wird eine JSON-Datei geschrieben mit Metadaten zu den Parametern der einzelnen Stichtage, die für die Berechnung verwendet wurden. Mit dieser JSON-Datei wird dem Backend ebenfalls mitgeteilt, welche Stichtage angeboten werden können.

5.2 Backend

In diesem Abschnitt wird die Implementation des Containers Backend erläutert. Primäres Ziel der Komponente ist es, Nutzer Informationen über verfügbare ÖV-Güteklasse bereitzustellen. Es handelt sich dabei um eine schlanke Flask-Applikation [52] mit sehr geringer Logik.

Das Backend wird beim Starten initialisiert. Dabei wird eine generierte Metadaten-Datei des Container ÖV-Güteklassen 2018 Generator eingelesen. Genauer Sinn und Zweck dieser Datei ist im Abschnitt 4.2.1 beschrieben. Diese Datei beinhaltet Informationen, welche ÖV-Güteklassen verfügbar sind. Dadurch kann die Web-App entscheiden, welche ÖV-Güteklassen der Benutzer auswählen kann und wie diese vom Container Tile-Server bezogen werden können.

Diese Informationen werden über ein Web-API exponiert. Diese besteht aus Representational State Transfer-Services. Die verfügbaren Services sind in der Tabelle 23 beschrieben. Im Bezug auf das Maturity Model [53] von Leonard Richardson befinden wir uns auf Level 2.

Mithilfe von Flasgger [54] wird eine OpenAPI Specification (OAS) [55] erstellt. Dies hat den Vorteil, dass das API sauber dokumentiert ist und Clients der API in unterschiedlichen Sprachen generiert werden können.

Das konkrete Web-API, welches das Backend exponiert, kann mit ReDoc (OpenAPI/Swagger-generated API Reference Documentation) [56] oder mit dem klassischen Swagger UI [57] betrachtet werden.

5.3 Tile-Server

Als Tile-Server bietet sich TileServer GL [58] an. Wie die GeoJSON in ein vom Tile-Server unterstütztes Format gelangen, ist im Abschnitt 5.3.1 beschrieben.

Der Tile-Converter ist als Docker-Container verfügbar. Dieser wird im Kapitel 7.2 genauer beschrieben.

5.3.1 Tile-Converter

Der Container ÖV-Güteklassen 2018 Generator generiert die ÖV-Güteklassen aktuell im GeoJSON-Format. Damit der Tile-Server Vector Tiles ausliefern kann, müssen die ÖV-Güteklassen in ein entsprechendes Format umgewandelt werden. Der oben erwähnte Tile-Server nutzt MBTiles [59]. Dabei handelt es sich um eine Spezifikation von Mapbox, um beliebige gekachelte Karten-Daten zu speichern. Dieses Format ist extrem effizient für Transfers, was in unserer Problemdomain relevant ist. Für die Konvertierung bietet sich das Tool *tippecanoe* [60], ebenfalls von Mapbox, an. Dieses erlaubt die Konvertierung von verschiedenen Dateiformaten, so auch GeoJSON in MBTiles und die Konfiguration der zu generierenden Vector Tilesets.

Die Indirektion über das GeoJSON-Format ist bewusst. Die ÖV-Güteklassen können als GeoJSON breiter verwendet werden, etwa in QGIS. Das MBTiles ist speziell auf Vector Tilesets ausgelegt, was bei uns nicht der Hauptfokus ist.

Der Tile-Converter ist als Docker-Container verfügbar. Dieser wird im Kapitel 7.2 genauer beschrieben.

5.4 Web-App

In einer Evaluation, welche im Kapitel 3.2 beschrieben ist, wurde entschieden, dass der Fokus auf React [24] gelegt wird. Die Web-App wurde dementsprechend mit React entwickelt. Grafisch wurde das Frontend aufgrund der Wireframes umgesetzt. Ein Auszug ist in Abbildung 24 ersichtlich.

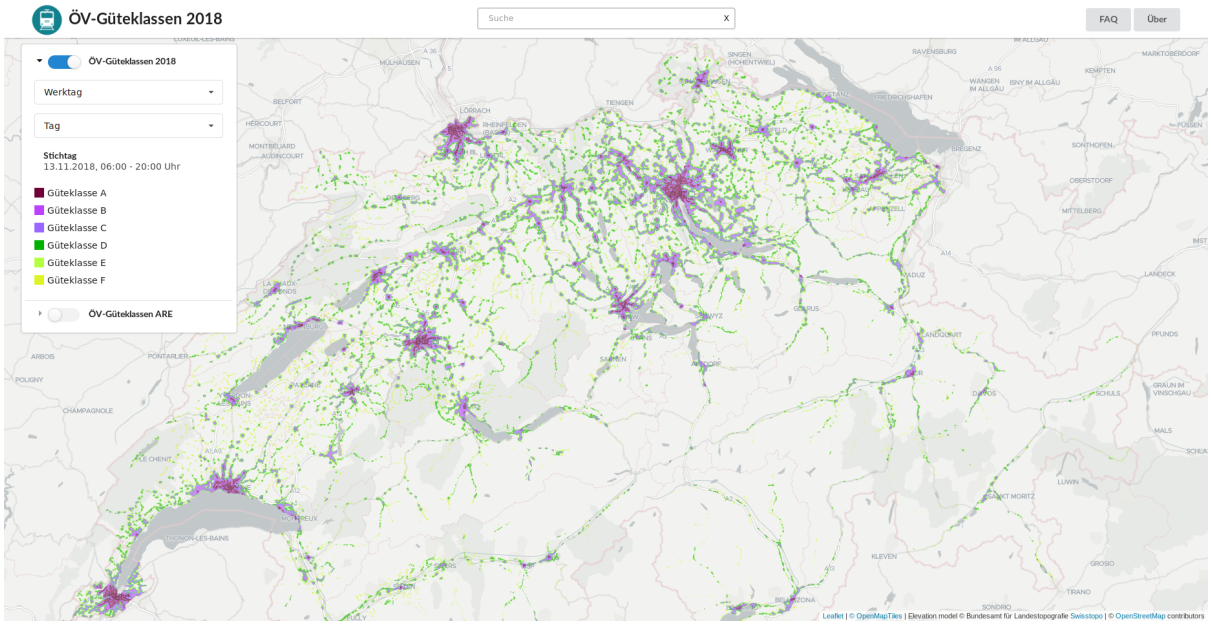

Abbildung 24: Web-App

Zur Übersicht sind die React-Komponenten in einer Baumstruktur in Abbildung 25 aufgeschlüsselt.

Dabei handelt es sich im engeren Sinn nicht um ein klassisches UML-Komponenten-Diagramm. Die Verantwortlichkeiten werden nachfolgend kurz beschrieben.

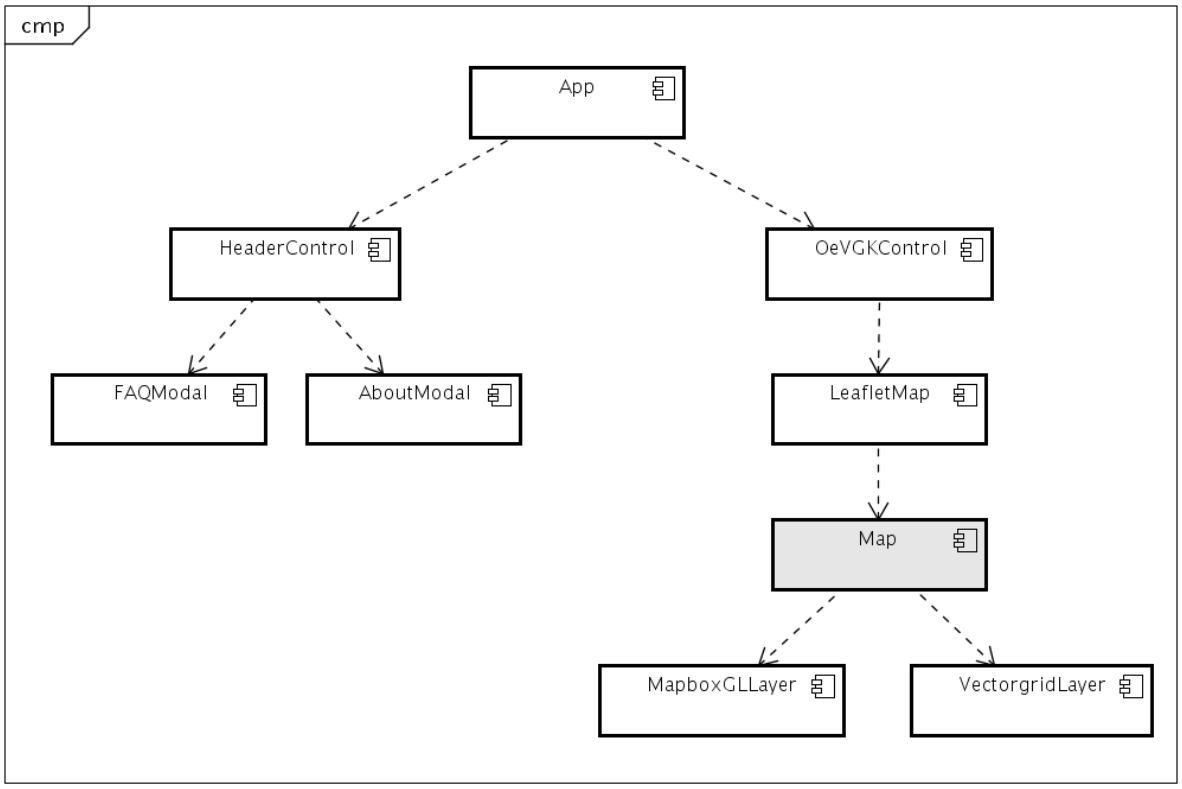

Abbildung 25: Web-App React Component Übersicht (grau: externe Komponente)

App Der zentrale Einstieg der React-Applikation ist die *App*-Komponente.

HeaderControl Der ganze Header wird über den *HeaderControl* verwaltet, welche aus zwei Modals (=Dialogen) besteht, die geöffnet werden können. Zur einfachen Bedienung ist ebenfalls ein Geocoding als Suchfunktion eingebunden.

OeVGKControl Interessant ist die Haupt-Komponente *OeVGKControl*, welche die Benutzereingaben und die Karte kontrolliert. Diese Komponente nutzt das Web-API vom Backend. Der Ablauf ist stark vereinfacht in Abbildung 26 sichtbar. Beim Mounten dieser Komponente werden vom Backend die verfügbaren Stichtage angefragt (Schritt 1 und 2). Anschliessend werden für einen Stichtag die zugehörigen Zeitintervalle geladen (Schritt 3 und 4). Mit diesen beiden Informationen werden vom Tile-Server dann die gewünschten ÖV-Güteklassen geladen (Schritt 7). Wählt der User einen neuen Stichtag oder Zeitintervall, wiederholt sich der Ablauf.

Es sei angemerkt, dass auf dem Tile-Server für einen Satz von ÖV-Güteklassen nicht nur einmal zugegriffen wird, wie man aufgrund der Sequenz annehmen könnte. Der Vorteil der Vector Tiles und dem

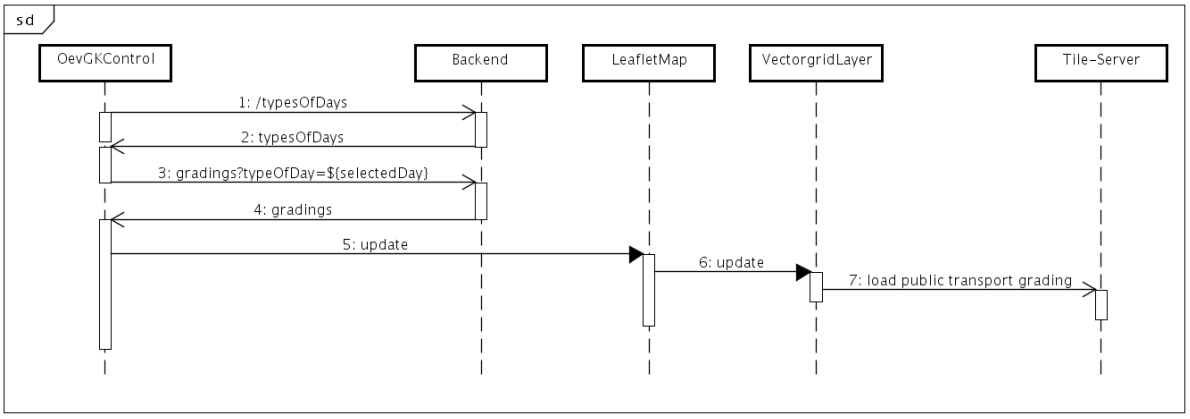

Abbildung 26: Auszug Zusammenspiel React Components

damit verbundenen Tile-Server liegt darin, dass gerade nur die Tiles geladen werden, welche benötigt werden. Verändert man Auszug und Zoomstufe, löst man somit Zugriffe auf den Tile-Server aus.

LeafletMap *LeafletMap* ist für die Darstellung der Karte verantwortlich. Dabei wird die Komponente *Map* von react-leaflet [29] verwendet. Mithilfe von Mapbox GL JS (MapboxGLLayer) [61] kann die Basiskarte ebenfalls mit Vector Tiles abgebildet werden, was standardmässig von Leaflet nicht unterstützt wird.

VectorgridLayer Die Kartendaten, welche über die Basiskarte gerendert werden, werden mit der Komponente *VectorgridLayer* verwaltet. So existiert jeweils ein Layer für die ÖV-Güteklassen 2018, für die ÖV-Güteklassen des Bundesamt für Raumentwicklung und eine für die ÖV-Haltestellen. Somit werden alle Kartendaten mit Vector Tiles abgebildet. Alle diese Layer sind in Abbildung 27 ersichtlich.

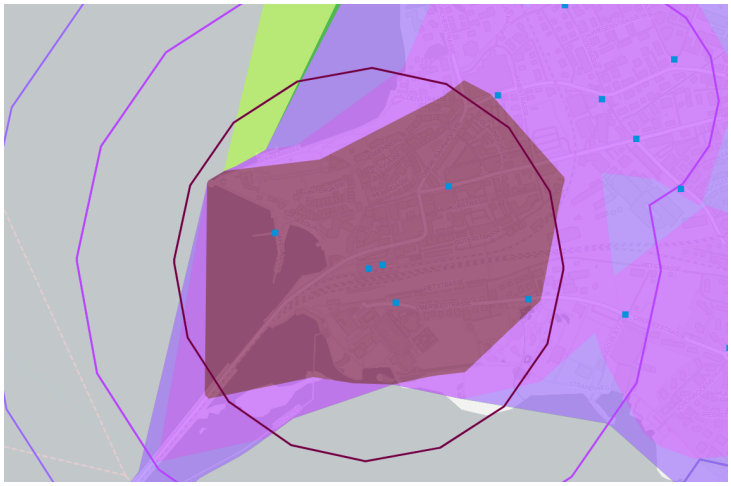

Abbildung 27: Vector-Grid Layers

Beim Layer, welcher die ÖV-Haltestellen anzeigt, ist der Vorteil der Vector Tiles stark sichtbar. So wurden diese Vector Tiles so konfiguriert, dass sie erst ab einer gewissen Zoomstufe angezeigt werden.

In der Abbildung 28 sieht man den Unterschied zwischen Zoomstufe 13 (links) und 14 (rechts). Dadurch müssen auf einer tieferen Zoomstufe weniger grosse Vector Tiles geladen werden und der Fokus bleibt auf den ÖV-Güteklassen.

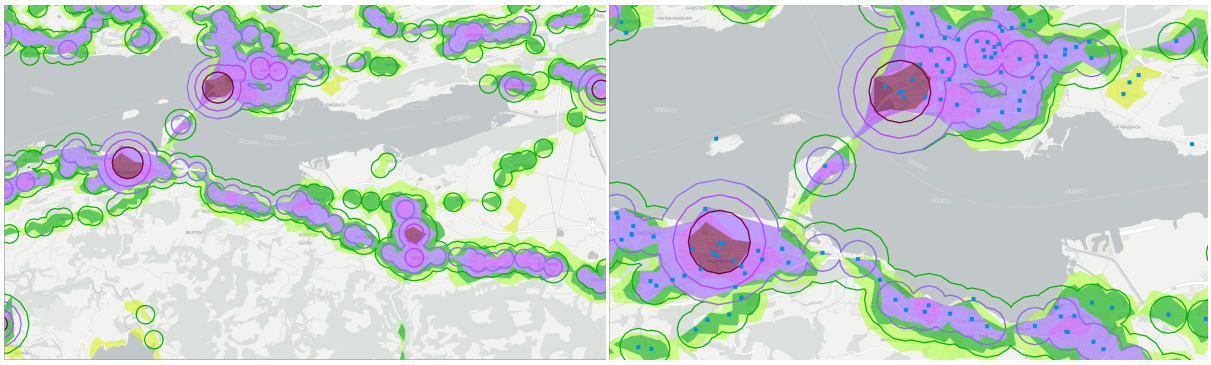

Abbildung 28: Vergleich Zoomstufe ÖV-Haltestelle Layer: Zoomstufe 13 (links) und 14 (rechts)

5.4.1 Wechsel von Mapbox GL auf Leaflet

In einer Analyse wurden die beiden Frontend-Frameworks Vue.js und React evaluiert (siehe Kapitel 3.2). Die Unterstützung von Mapbox GL JS [32] war das ausschlagende Kriterium für den Entscheid zugunsten von React. Der Grund für den Einsatz von Mapbox GL JS gegenüber Leaflet [28] war die gute Unterstützung von Vector Tiles. Trotz alldem hat sich während der Entwicklung gezeigt, dass der Wechsel zu Leaflet praktisch unausweichlich ist. Im Folgenden ist dieser Schritt begründet.

Der ausschlaggebende Nachteil von Mapbox GL JS zeigte sich bei der Integration der ÖV-Güteklassen, um sie auf der Webkarte anzeigen zu können. Wenn sich mehrere Haltestellen nah beieinander befinden, überschneiden sich ihre Isochronen. Da wir hinter den Geometrien der ÖV-Güteklassen auch die Basiskarte im Hintergrund anzeigen möchten, muss die Deckkraft der Geometrien entsprechend verringert werden. Mapbox GL JS unterstützt die Einstellung der Deckkraft aber ausschliesslich auf Feature-Ebene, sprich auf einzelnen Geometrien. Bei sich überlagernden Features entsteht damit der Effekt, dass die Farben sich mischen und einander verstärken (siehe Abbildung 29). Dies ist irreführend, da ein Gebiet mit zum Beispiel Güteklasse A durchgehend gleich kategorisiert werden soll, egal von wie vielen verschiedenen Haltestellen mit der gleichen Güteklasse das Gebiet erschlossen wird.

Für die Lösung dieses Problems gibt es zwei Ansätze:

1. Die Deckkraft der einzelnen Geometrien auf 100% setzen und dabei die Deckkraft des kompletten Layers verringern

2. Die Geometrien voneinander ausschneiden, so dass keine Überlagerungen mehr existieren

Der erste Ansatz ist mit Mapbox GL JS momentan nicht möglich. Die Unterstützung dafür wird aktuell auf Github diskutiert und steht noch offen [62]. Der zweite Ansatz wurde während der Entwicklung

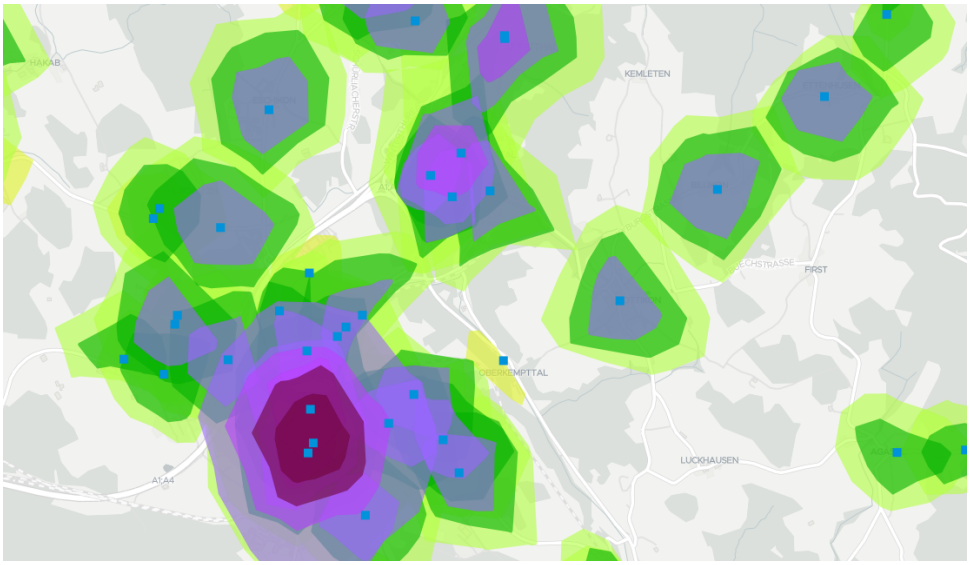

Abbildung 29: Überlappende Geometrien mit gleicher Deckkraft ergeben gemischte und irreführende Farbflächen

implementiert. Da alle Geometrien von allen anderen ausgeschnitten werden, ist dies eine komplexe Operation in der Grössenordnung $\mathcal{O}(n^2)$. Mit einem R-Tree-Index kann diesem Problem entgegen gesteuert werden. Es entstanden ausserdem Probleme mit der Topologie der ausgeschnittenen Geometrien sowie Render-Fehler im Browser.

Diese Problematiken haben uns dazu bewegt, von Mapbox GL JS auf Leaflet umzusteigen. Die erste Option, die Verringerung der Deckkraft auf dem kompletten Layer, ist dort kein Problem. Damit wurde auch das Ausschneiden der einzelnen Geometrien überflüssig.

Vector Tiles mit Leaflet

Der Wechsel zu Leaflet bedeutet auch, dass Vector Tiles nicht mehr nativ unterstützt werden. Allerdings gibt es mit *Leaflet.VectorGrid* [63] eine Erweiterung, die genau dies erlaubt. Dafür gibt es zwar keine vorgefertigten React-Komponenten, mit etwas Handarbeit können diese aber leicht selbst erstellt werden.

Ein Spezialfall ist das Einbinden der Basiskarte, die von OpenMapTiles [33] bezogen wird. Diese wird von *VectorGrid* nicht unterstützt. Es gibt aber wiederum von Mapbox selbst eine Library, um Mapbox GL JS in einer Leaflet-Karte einzubinden [61]. Somit haben wir schlussendlich einen Hybrid aus Mapbox GL JS und Leaflet, wobei Mapbox GL JS nur für die Basiskarte verwendet wird, wo die oben genannten Probleme mit der Deckkraft nicht relevant sind.

6 Tests

6.1 ÖV-Güteklassen 2018 Generator

Der Generator für die ÖV-Güteklassen 2018 wurde in Python implementiert. Für Unit-Tests hat sich dafür in vorhergehenden Projekten das Testing-Framework *pytest* [64] bewährt.

Die Architektur wurde so gestaltet, dass die Business-Schicht keine direkte Abhängigkeit auf die Integrations-Schicht hat, die für die Kommunikation mit der Datenbank verantwortlich ist, und so isoliert getestet werden kann. Die Abhängigkeiten wurden stattdessen dynamisch in einer „Registry", analog zu einer Dependency Injection, gehalten. Dies erlaubte es uns, in den Unit-Tests die Module in der Integrations-Schicht durch Mocks auszutauschen.

Da einige Logik der Berechnung in Stored Procedures auf der Datenbank implementiert wurden, wäre es wünschenswert, diese ebenfalls automatisiert zu testen. Zwar gibt es dafür Ansätze wie *pgTAP* [65], um Unit-Tests für Datenbank-Funktionen zu schreiben, bei uns war aber neben der Logik auch die Performance ein entscheidender Faktor. Diese konnte nur manuell auf dem kompletten Datensatz der Schweiz getestet werden, denn erst dann wurde klar, welche Indizes benötigt werden und wie die Datenbank eine bestimmte Query ausführt.

Vorstellbar wären hingegen Integrations-Tests, die auf einem Test-Set die kompletten ÖV-Güteklassen berechnen. Die Schwierigkeit daran ist das aufwändige Setup mit Fahrplan-, Routing- und Höhendaten, die alle in einer laufenden Datenbank aufgesetzt werden müssten. Dabei wäre es natürlich wünschenswert, das ganze Setup automatisiert in einer Continuous Integration Umgebung zu betreiben.

6.2 Backend

Für das Backend wurde das automatische Testing auf minimale Tests beschränkt, die prüfen, dass der Server auf API-Requests keine unerwartete Antworten liefert. Während der Entwicklung wurde das Backend immer schlanker. Wurden zuerst die kompletten GeoJSON-Daten für die Web-Applikation ausgeliefert, wird seit dem Wechsel zu Vector Tiles nur noch ein JSON mit Metadaten zu den Parametern eingelesen und über ein API angeboten. Dementsprechend ist neben dem Exponieren von Metadaten keine mehr Logik vorhanden.

6.3 Web-Applikation

Unit-Tests

Für das in React geschriebene Frontend wurde ursprünglich geplant, Unit-Tests mit *Jest* [66] zu schrei-

ben. Allerdings kam schon früh zur Entwicklung das Problem auf, dass die *Mapbox GL* Komponente, die für die Basiskarte verwendet wird, nicht in einem Test geladen werden kann, weil dafür ein Browser benötigt wird. Da die einzige Logik in der Applikation in der Verarbeitung der Metadaten vom Backend besteht, wären reine Unit-Tests aufwändig. Sinnvoller wäre es, ein Integration-Test zusammen mit dem Backend-API zu machen. Dies erfordert aber ein aufwändiges Setup, um dies automatisiert mit einer Continuous Integration zu verbinden. Dieser Aufwand wäre aufgrund der Anforderungen der Web-Applikation nicht gerechtfertigt.

Type-Checking

Da JavaScript eine schwach typisierte Sprache ist, macht es Sinn, während der Entwicklung statisches Type-Checking zu verwenden. So können bereits zu Beginn des Entwicklungsprozesses Fehler mit Typisierungen vermieden werden.

Dafür bietet sich die Library *Flow* [67] an, die wie React ebenfalls von Facebook entwickelt wird. Mit dieser ist es möglich, bei allen Deklarationen die Typen anzugeben. Kontinuierlich wird dann geprüft, dass die Typisierungen korrekt verwendet werden.

7 Infrastruktur

7.1 Continuous Integration

In unserem Projekt wird Continuous Integration (CI) einerseits für das automatische Erstellen dieser Dokumentation mit LaTeX verwendet und andererseits, um automatische Tests für die Implementation auszuführen. Beides wird im Folgenden kurz beschrieben.

7.1.1 Erstellen der LaTeX-Dokumentation

Bei jedem Git Commit in das Repository der Dokumentation [68] wird mit Continuous Integration ein PDF erstellt. So wird sichergestellt, dass die Dokumentation keine syntaktischen Fehler aufweist. Das PDF wird in unser internes JIRA hochgeladen und direkt dem Task angehängt, der zum Branch gehört.

Für diese Continuous Integration haben wir uns für Travis CI [69] entschieden, da die Integration mit Github gut funktioniert und das Produkt für Open-Source-Projekte kostenlos ist. Allerdings bietet Travis CI keine direkte Unterstützung für eine LaTeX-Umgebung an. Die LaTeX-Pakete aus den Repositories sind ausserdem veraltet. So muss bei jedem Build die LaTeX-Umgebung kompiliert werden, damit immer die neueste Version aller Pakete verwendet werden kann.

7.1.2 CI für die Unit Tests

Für die Continuous Integration der Web-Applikation braucht es eine Umgebung mit Node JS, um die Applikation zu kompilieren und die Tests auszuführen. Für den ÖV-Güteklassen 2018 Generator wird eine entsprechende Umgebung für Python benötigt. Dazu bietet sich ebenfalls Travis CI [69] an. Die entsprechenden Umgebungen sind bereits vorhanden, womit der Konfigurationsaufwand sehr klein bleibt. Ebenfalls existiert ein Cron-Job, welcher die Builds und somit die Tests in regelmässigen Abständen ausführt.

7.2 Deployment

Für das Deployment wurde eine Docker-Umgebung eingerichtet, die für die einfache Handhabung der Berechnung der ÖV-Güteklassen 2018 dient. Die Details dazu sind im Kapitel 5.1 beschrieben.

Um die berechneten ÖV-Güteklassen in der Web-Applikation zu visualisieren, wurde eine zusätzliche Docker-Umgebung vorbereitet. Diese kombiniert die Web-App, Backend sowie der Tile-Server, damit man die komplette Applikation ohne zusätzliche Konfiguration lokal oder auf einem Server deployen kann. Dies hat den Vorteil, dass das Setup, welches deployed wird, analog lokal gestartet und getestet werden kann. Um die Services in einem gemeinsamen Endpoint zu verbinden, wird eine Nginx-Instanz als Reverse-Proxy verwendet, der die API-Requests an die entsprechenden Endpoints weiterleitet. Die Umgebung ist in Abbildung 30 dargestellt.

Zusätzlich wurde ein Service eingerichtet, um die errechneten ÖV-Güteklassen im GeoJSON-Format in Map-Tiles, sprich in MBTiles (siehe Kapitel 5.3.1) umzuwandeln, damit der Tile-Server damit umgehen kann. Der Benutzer kann die GeoJSON-Dateien in einem Ordner ablegen und die Docker-Umgebung mit Docker Compose starten. Die Aufbereitung der Map-Tiles geschieht dann automatisch im Hintergrund.

7.3 Version Control

Alle Artefakte, welche in dieser Arbeit erstellt wurden, sind öffentlich auf Github verfügbar. Es wurde dazu eine Github Organisation mit dem Namen „public-transport-quality-grades“ [70] erstellt.

In der Tabelle 28 sind die einzelnen Repositoriers aufgelistet und kurz der Inhalt erläutert. Diese soll als Übersicht dienen und einen raschen Einstieg in die Version Control ermöglichen.

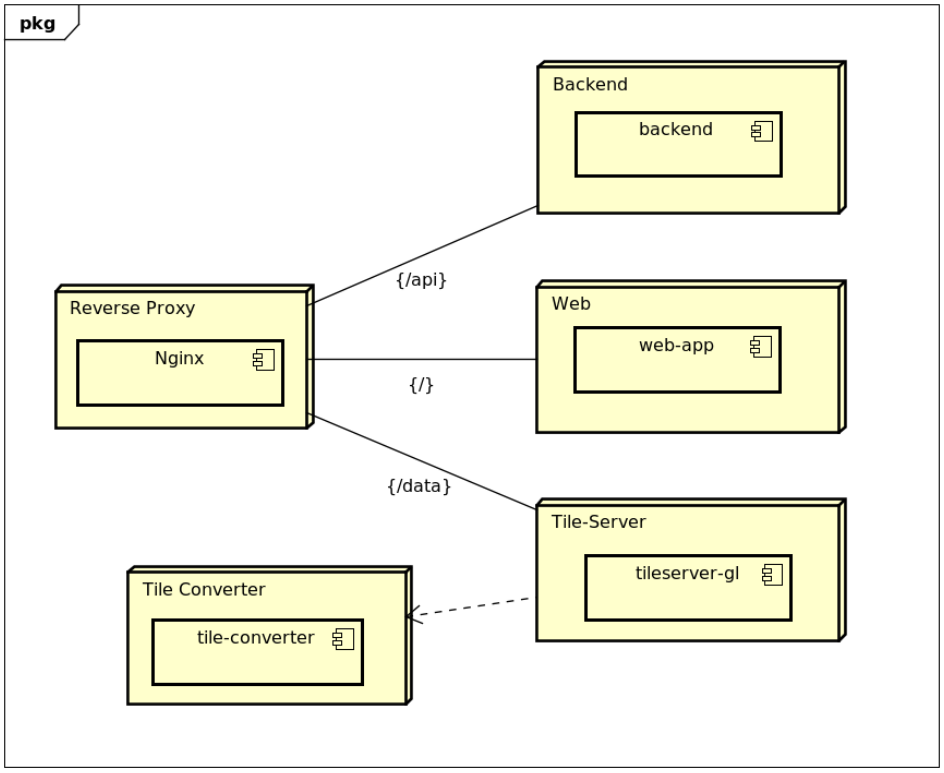

Abbildung 30: Deployment-Diagramm der Web-Applikation mit den entsprechenden Backend-Services

Name	Inhalt
thesis [68]	Bachelorarbeit
oevgk18-specification [71]	Aus der Bachelorarbeit extrahierte ÖV-Güteklassen 2018 Spezifikation
oevgk18-generator [49]	ÖV-Güteklassen 2018 Generator
web-app [72]	Web-Applikation zur Visualisierung der ÖV-Güteklassen 2018
backend [73]	Backend der ÖV-Güteklassen 2018
backend-api [74]	Dokumentation des Web-API des Backend
admin [75]	Sitzungsprotokolle und Entscheidungen
infrastructure [76]	Interne Projekt-Infrastruktur (Jira-Setup, ...)
playground [26]	Spielwiese für die Evaluation von Frameworks

Tabelle 28: Github Repository Übersicht

8 Resultate und Weiterentwicklung

8.1 Resultate

Mit unserer Spezifikation OeVGK18 sowie deren Implementation ist die Grundlage für eine einheitliche Ermittlung von ÖV-Güteklassen für die gesamte Schweiz geschaffen. Mithilfe des entwickelten Generators können die ÖV-Güteklassen jederzeit und automatisiert mit den aktuellsten Datensätzen für unterschiedliche Stichtage und Zeitintervalle erzeugt werden. Die Web-Applikation erlaubt eine Visualisierung der errechneten Geodaten auf einer interaktiven Webkarte. Zusätzlich können die ÖV-Güteklassen des ARE den ÖV-Güteklassen 2018 visuell überlagert werden, um sie zu vergleichen. Durch das erstellte Docker-Setup wird ein einfaches Deployment erreicht.

8.1.1 Laufzeit

Um die Laufzeit für das Berechnen der ÖV-Güteklassen 2018 auszuwerten, wird zuerst für die Kernoperation die theoretische Komplexität bestimmt. Mit einer Messung wird dann die reale Laufzeit ermittelt und ausgewertet.

Theoretische Laufzeit

Für die Bestimmung der theoretischen Laufzeit wird lediglich das Berechnen der Isochronen betrachtet, da dies für die eigentliche Berechnung der ÖV-Güteklassen die Kernfunktion darstellt, die am meisten Laufzeit benötigt.

Wie in 5.1.2 beschrieben wird für jede der N Haltestellen eine oder mehrere Isochronen erzeugt. Dieser Algorithmus besteht aus zwei Teilen. Als erstes werden mit dem Dijkstra-Algorithmus [50] alle Vertices ermittelt, die von der Haltestelle aus in einer festgelegten Zeit im Routing-Graphen erreichbar sind. Im zweiten Schritt wird mit diesen Vertices eine Alpha Shape [51] erzeugt, die alle gefundenen Punkte in einem Polygon einschliesst. Der erste Schritt muss pro Haltestelle nur ein Mal für die grösste Distanz durchgeführt werden, für die Erzeugung der Isochronen für kürzere Distanzen wird das Ergebnis des Dijkstra-Algorithmus gefiltert.

Der Dijkstra-Algorithmus wird in pgRouting mit der „Boost Graph"-Library [77] implementiert und hat eine Laufzeit von $\mathcal{O}(V \log V + E)$, wobei E die Anzahl der Kanten und V die Anzahl Vertices im Routing-Graphen bezeichnen. Für eine effizientere Berechnung werden in einem vorhergehenden Schritt alle Kanten $E\prime$ ermittelt, die maximal 1300 Meter von der Haltestelle entfernt sind. Diese Vorselektion der Kanten ist dank einem R-Tree-Index mit $\mathcal{O}(\log E)$ möglich. Insgesamt ergibt dies eine Laufzeit von $\mathcal{O}(\log E) + \mathcal{O}(V \log V + E\prime)$. Da die Vorselektion der Kanten $E\prime$ nur sehr klein ist ($E\prime \ll V$) und in konstanter Zeit läuft, kann dies vernachlässigt werden.

Der zweite Schritt, das Erstellen der Alpha Shape, wird nur noch mit den vorher ermittelten Punkten $V\prime$ durchgeführt. Da dies im Vergleich zum kompletten Graphen nur sehr wenige Punkte sind ($V\prime \ll V$), ist dieser Schritt für die Laufzeit ebenfalls zu vernachlässigen.

Diese Berechnung der Isochronen wird für alle N Haltestellen durchgeführt. Insgesamt ergibt das folgende Laufzeit:

$$\mathcal{O}(N(\log E + V \log V))$$

wobei: N = Anzahl Haltestellen
V = Anzahl Vertices im Routing-Graph
E = Anzahl Kanten im Routing-Graph

Reale Messung der Laufzeit

Um einen Richtwert für die effektive Laufzeit der Berechnung zu haben, wird auf einem Test-Computer der komplette Ablauf vom Datenimport bis zur Ausgabe der ÖV-Güteklassen 2018 als GeoJSON durchlaufen und die Zeit gemessen.

Testumgebung Die Messung wird auf einem Desktop-Computer mit einem Intel Xeon E3-1245 mit 3.4 GHz und 16 GB Arbeitsspeicher durchgeführt. Als Betriebssystem wird Linux verwendet, das Terrain-Modell ist auf einer externen Festplatte im TIF-Format abgelegt.

Ergebnis Die gemessene Laufzeiten sind in Tabelle 29 ersichtlich. Es fällt auf, dass die Vorbereitung zur eigentlichen Berechnung die meiste Zeit beansprucht. Dies liegt unter anderem daran, dass diese Vorbereitungsschritte externe Funktionen aufrufen, die nicht spezifisch optimiert werden können. So braucht etwa das Berechnen der Topologie nach der Graph-Segmentierung fast zwei Stunden. Dabei wird lediglich die eingebaute Funktion von pgRouting aufgerufen. Als Optimierung ist vorstellbar, die Topologie direkt während der Routing-Graph-Segmentierung zu aktualisieren, was aber deutlich komplexer als die jetzige Lösung wäre.

Die eigentliche Berechnung der ÖV-Güteklassen für 6 Stichtage konnte mit Indizes auf der Datenbank und Optimierung der Algorithmen auf eine Laufzeit von rund 70 Minuten gebracht werden. Dafür wurde während der Entwicklung kontinuierlich optimiert, in früheren Iterationen dauerte die Berechnung noch mehrere Tage. Als Beispiel dafür sei die Optimierung zu erwähnen, den Dijkstra-Algorithmus für die Berechnung der Isochronen für jede Haltestelle nur ein Mal für die weiteste Distanz zu verwenden statt für jede Isochrone einzeln. Allein durch diese Optimierung konnte die Laufzeit zur Berechnung einer Haltestelle von ca. 450 ms auf 30 ms reduziert werden, womit die totale Laufzeit von 18 Stunden auf rund 70 Minuten gekürzt wurde. Die nicht-funktionale Anforderung NFA02: Berechnung der ÖV-Güteklassen 2018 wurde somit klar eingehalten und übertroffen.

Berechnungsschritt	Benötigte Zeit
Datenimport	**1 h 45 min**
Fahrplandaten importieren	4 min
Routingdaten importieren	16 min
Terrain-Modell importieren	1 h 25 min
ÖV-Güteklassen berechnen	**4 h 56 min**
Routing-Graph segmentieren	38 min
Topologie neu berechnen	1 h 53 min
Terrain-Daten in Routing-Graph rechnen	1 h 13 min
ÖV-Güteklassen für 6 Stichtage berechnen	1 h 12 min
Gesamt	**6 h 41 min**

Tabelle 29: Ergebnis der Laufzeitmessung

Auswertung mit theoretischer Laufzeit Im Absatz Theoretische Laufzeit wurde die theoretische Laufzeit zur Berechnung von Isochronen ermittelt. In der Messung hat sich ergeben, dass für einen einzelnen Stichtag die Berechnung dieser Isochronen ca. 12 Minuten benötigt.

Mit der theoretischen Laufzeit von $\mathcal{O}(N(\log E + V \log V))$ gibt es $N = 24'318$ Haltestellen sowie $E = 7'040'965$ Kanten und $V = 6'640'003$ Vertices im Routing-Graphen. In der Messung dauerte die Berechnung für jede der N Haltestellen durchschnittlich ca. 30 ms, was einer Gesamtlaufzeit von ca. 12 Minuten entspricht. Diese Berechnung wurde für 6 verschiedene Stichtage und Zeitintervalle durchgeführt. Dabei ist anzumerken, dass in der Messung dieser Operation einige Variablen vernachlässigt wurden, wie etwa die Zeit für die Anfrage und Antwort des Datenbank-Servers.

8.2 Möglichkeiten der Weiterentwicklung

Im Teil I Kapitel 5 wurde bereits auf mögliche Weiterentwicklung eingegangen, die alternative Ansätze zur Ermittlung von ÖV-Güteklassen beinhalten. In diesem Abschnitt soll der Fokus auf der Implementation unserer Spezifikation OeVGK18 liegen.

Während der Arbeit sind folgende Ansätze zur Weiterentwicklung und Optimierung aufgekommen:

Parallelisierung Die Berechnung der ÖV-Güteklassen 2018 wurde während der Entwicklung stets optimiert, um eine akzeptable Laufzeit zu erzielen. Der Grossteil der Optimierungen involvierte Datenbank-Indizes oder Vorberechnungen, um die Komplexität zu Verringern. Da die Berechnung der ÖV-Güteklassen für jede Haltestelle unabhängig erfolgt, könnte zusäztlich der ganze Prozess parallelisiert werden, um Multi-Core-Systeme auszunutzen. In der jetzigen Implementation wurde dies

für die Berechnung der Leistungskilometer bereits umgesetzt. Mit der Parallelisierung des gesamten Prozesses könnte etwa auf einem 4-Core System eine Laufzeitverbesserung von annähernd Faktor 4 erzielt werden. Aufgrund der Natur der Berechnungen eignet sich besonders der Ansatz „Data parallelism“ [78]. Die Berechnungen gehen von Haltestellen aus. Somit kann man diese problemlos in einzelne Bereiche aufteilen und unabhängig voneinander parallel berechnen.

Effizientere Segmentierung des Routing-Graphen In der jetzigen Implementation wird der Routing-Graph segmentiert und anschliessend mit einer integrierten Funktion von pgRouting die komplette Topographie neu berechnet. Diese Funktion dauert knapp 2 Stunden. Eine Optimierung könnte sein, die Topographie des Graphen bereits während der Segmentierung neu zu berechnen. Da jede Kante in mehrere Kanten zerstückelt wird, müssten dazu jeweils neue Vertices eingefügt und entsprechend verbunden werden. Dies bedeutet etwas höhere Komplexität bei der Segmentierung, dafür entfällt der Schritt zur Neuberechnung des kompletten Graphen durch pgRouting.

Zusätzliche Ausgabeformate Momentan werden die errechneten ÖV-Güteklassen im GeoJSON-Format ausgegeben. Für eine weitergehende Analyse der Resultate kann es sinnvoll sein, die Resultate zusätzlich in zum Beispiel einer PostGIS-Datenbank auszugeben. Dabei könnten auch beliebige Metadaten zu den Haltestellen und Geometrien eingebunden werden, die während der Berechnung entstehen. Durch unsere Architektur benötigt ein zusätzliches Ausgabeformat lediglich eine weitere Implementation von *OutputWriter* (siehe Kapitel 4.2.1).

9 Projektmanagement

9.1 Vorgehen

Für die Bachelorarbeit wurde das agile Vorgehen SCRUM in Kombination mit Elementen von Rational Unified Process (RUP) gewählt. Gründe für diese Entscheidung sind, dass das agile Vorgehen der noch zu Beginn offenen Spezifikation der ÖV-Güteklassen und der theoretischen Aufarbeitung entgegenkommt, welche dann iterativ finalisiert werden kann und dass der theoretische Fokus der Arbeit so besser gehandhabt und schneller reagiert werden kann. Die wöchentlichen Besprechungen und Reviews mit dem Betreuer ist ein weiterer Grund für diese Entscheidung. Die Kombination mit Elementen von RUP ermöglicht es, dass Projekt in einzelne Phasen aufzuteilen, um so das Ziel und die Zeit nicht aus den Augen zu verlieren und den theoretischen Teil und die Spezifikation zu einem Abschluss bringen zu können.

9.1.1 Entwicklung

Der Source-Code der Implementation wie auch diese Arbeit wird mit Git verwaltet und ist auf Github [70] abgelegt. Die Entwicklung und das Dokumentieren erfolgt nach dem Github-Flow. Der Master-Branch ist auf allen Repositories während der ganzen Zeit gesperrt, so dass er nur über Pull-Requests bearbeitet werden kann. Für jedes Arbeitspaket wird ein Branch erstellt. Ist das Arbeitspaket implementiert, wird ein Pull-Request erstellt und dem anderen Projekt-Mitglied zum Review übergeben. Wird der Pull-Request akzeptiert, wird der Feature-Branch in den Master gemerged. Dieses Vorgehen hat den Vorteil, dass alle Änderungen, welche in den Master gelangen, ein Review durchlaufen müssen und so die Qualität hochgehalten werden kann.

9.2 Zeitplanung

Die Arbeitspakte und Zeit wird mithilfe von Jira verwaltet. Für alle Tätigkeiten werden Arbeitspakete im Backlog erfasst, priorisiert und geschätzt. Die Schätzung der Arbeitspakete erfolgte mit Story Points. Die Arbeitszeitverbuchung wurde auf Arbeitspaket-Stufe mit Stunden gemacht.

9.2.1 Phasen / Iterationen und Meilensteine

Die Bachelorarbeit wird in die RUP-Phasen (Inception, Elaboration, Construction, Transition) aufgeteilt. Dabei wird jedoch eine von der gängigen Norm abweichende Aufteilung gewählt. Durch den theoretischen Fokus der Arbeit und der zu erarbeitenden Spezifikation wird der Elaboration das grösste Zeitbudget zugeordnet. Dies ist auch der Grund, warum mit einwöchigen Sprints gearbeitet wird. In den letzten zwei Wochen des Projekts wird der Aufwand pro Sprint verdoppelt (40h pro Person), da in dieser Zeit Vollzeit am Projekt gearbeitet werden kann. Die in Tabelle 30 definierten Meilensteine werden in Jira als Epics definiert und die Arbeitspakete diesen zugewiesen. Dies aus dem einfachen Grund, da Jira das Konzept Milestones nicht einfach unterstützt.

Sprint	**Sprint 1**	**Sprint 2**	**Sprint 3**
Phase	Inception	Elaboration	Elaboration
Milestones	*Projektantrag genehmigt*	*Projektplan erstellt, Scope abgesteckt*	*Stand der Technik evaluiert*
Inhalt	1. Projektantrag erstellen 2. Grobplanung erstellen 3. LaTex und CI aufsetzen	1. Projektplan erstellen 2. FA/NFA erarbeiten 3. Abgrenzungen definieren	1. in Norm 640 290 einarbeiten 2. aktuelle Berechnungsmethodik aufschlüsseln 3. Fremdsysteme & Datenquellen eruieren

Sprint	**Sprint 4**	**Sprint 5**	**Sprint 6**
Phase	Elaboration	Elaboration	Elaboration
Milestones	*Stand der Technik evaluiert*	*technische Machbarkeit geprüft*	*Spezifikation umsetzungsbereit*
Inhalt	1. Anbindung Fremdsysteme & Datenquellen evaluieren 2. Framework für Frontend evaluieren 3. Auslieferung Kartendaten an Frontend evaluieren	1. In PostGIS einarbeiten 2. Machbarkeitsanalyse pgRouting durchführen	1. Verbesserungen der Berechnungsmethoden erarbeiten 2. Spezifikation erstellen

Sprint	**Sprint 7**	**Sprint 8**	**Sprint 9**
Phase	Elaboration	Elaboration	Construction
Milestones	*Spezifikation umsetzungsbereit*	*End of Elaboration*	*Spezifikation implementiert*
Inhalt	1. Verbesserungen der Berechnungsmethoden erarbeiten 2. Spezifikation erstellen	1. Architektur definieren 2. Infrastruktur aufsetzen 3. Frontend Design Entwurf erstellen	1. Konfigurationsmöglichkeiten definieren 2. Spezifikation umsetzen

Sprint	**Sprint 10**	**Sprint 11**	**Sprint 12**
Phase	Construction	Construction	Construction
Milestones	*Spezifikation implementiert*	*Spezifikation implementiert*	*Spezifikation implementiert*
Inhalt	1. Spezifikation umsetzen	1. Spezifikation umsetzen	1. Berechnung automatisieren

Sprint	**Sprint 13**	**Sprint 14**	**Sprint 15**
Phase	Construction	Construction	Construction
Milestones	*Web-Applikation entwickelt*	*Web-Applikation entwickelt*	*Web-Applikation entwickelt*
Inhalt	1. Backend für Auslieferung der Kartendaten erstellen 2. Frontend-Entwurf umsetzen	1. Backend für Auslieferung der Kartendaten erstellen	1. Frontend fertig stellen

Sprint	**Sprint 16**
Phase	Transition
Milestones	*BA abgegeben*
Inhalt	1. Präsentation erstellen 2. Plakat erstellen 3. Arbeit finalisieren 4. Arbeit abgeben

Tabelle 30: Phasen / Iterationen und Meilensteine

Fazit Projektverlauf

Im Anschluss an das Projekt lohnt es sich, einen kurzen Blick auf die Abweichungen zwischen dem Geplanten und der Realität zu werfen. Bis und mit Sprint 8 ist das Projekt mit einigen wenigen Arbeitspaketsverschiebungen analog zum Projektplan verlaufen. Ab Sprint 9 und mit der eigentlichen Implementierung wurde uns relativ schnell bewusst, dass sich die Entwicklung der einzelnen Komponenten parallel durchführen lässt. Der Hauptfokus und somit auch der grösste Aufwand lag jedoch immer noch bewusst auf dem ÖV-Güteklassen 2018-Generator, da das Projekt mit diesem steht und fällt. Nebenbei konnte jedoch Zeit in den Aufbau der Web-App und des Backends gesteckt werden. Auch einen Frontend-Entwurf stand relativ rasch, um so keine Überraschungen gegen Ende des Projekts zu erleben.

Im Nachhinein würden wir dieses Vorgehen wieder einschlagen. Dies aus dem Grund, dass der Generator der Kern des Projekts ist und man dadurch bei Problemen und somit Verzögerungen Abstriche bei der Web-App und dem Backend machen.

9.3 Stakeholder

Im folgenden werden die Stakeholder identifiziert, genauer analysiert und in Anspruchsgruppen eingeteilt. In Abbildung 31 ist ersichtlich, in welchem Mass diese Interesse an einem erfolgreichen Projektabschluss haben und Einfluss auf die Gestaltung des Projektes nehmen können. Das Interesse und der jeweilige Einfluss ist detailliert in Tabelle 31 aufgeführt.

Durch die Identifizierung und Gruppierung kann ein unterschiedliches Stakeholder Management durchgeführt werden. Die *Key Player*, welche sich im oberen rechten Quadrant befinden, werden bei den Entscheidungen einbezogen, regelmässig auf den neusten Stand gebracht und aktiv ins Projekt einbezogen. Die weniger relevanten Stakeholder im Bezug auf den Einfluss werden für das Projekt motiviert und das Interesse auf das Projektergebnis wird geweckt.

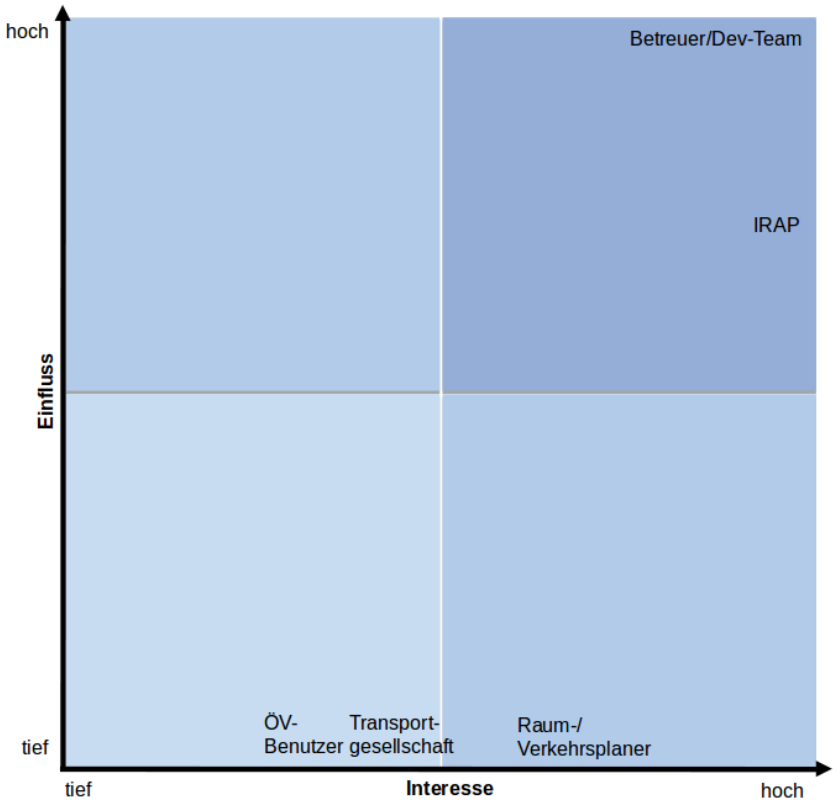

Abbildung 31: Stakeholder Map

Stakeholder	Interesse	Einfluss
Betreuer (Prof. Stefan Keller)	Erfolgreiche Bachelorarbeit, welche Themengebiete des Geometa Labs tangiert und nach Abschluss einen konkreten Nutzen erbringt.	Einfluss auf die Definition der Aufgabenstellung, Anforderungen, Rahmenbedingungen, laufende Steuerung des Projekts in enger Zusammenarbeit mit dem Dev-Team.
Dev-Team	Erfolgreiche Bachelorarbeit, in welcher das erlernte Wissen des Studiums konkret angewendet werden kann und das Thema „Netzwerkanalyse“ im OSM-Umfeld aufgreift. Die BA soll ein Produkt hervorbringen, welches von unterschiedlichen Personengruppen eingesetzt werden kann.	Einfluss auf die Definition der Aufgabenstellung, Anforderungen, Rahmenbedingungen, laufende Steuerung des Projekts in enger Zusammenarbeit mit dem Betreuer.
IRAP (Prof. Claudio Büchel)	Interessen analog zu Raumplaner/Verkehrsplaner. Durch die praktische Erfahrung mit der bisherigen Norm existiert ein aktives Interesse, diese neu zu gestalten und für die Zukunft fit zu machen.	Nimmt die OeVGK18-Spezifikation ab und gestaltet die Spezifikation aktiv mit.
Raumplaner/Verkehrsplaner	Gemäss Verordnung werden nur noch Gebiete mit einer guten ÖV-Erschliessung verdichtet. Mit den Güteklassen lässt sich prüfen, an welchen Standorten eine Verdichtung stattfinden kann.	Hat keinen direkten Einfluss. Die Interessen fliessen in die Anforderungen.
Transportgesellschaften	Transportgesellschaften überprüfen mit den Güteklassen die aktuelle Erschliessung der Schweiz und können dadurch Erweiterungspotential eruieren.	Hat keinen direkten Einfluss. Die Interessen fliessen in die Anforderungen.
Dev-Team	ÖV-Benutzer, welcher auf der Wohnungssuche ist, prüft mit den Güteklassen die Erschliessung eines potentiellen Wohnortes.	Hat keinen direkten Einfluss. Die Interessen fliessen in die Anforderungen.

Tabelle 31: Stakeholder

9.4 Risiken

ID	Risiko	max. Schaden [h]	WSK	gewichteter Schaden
1	Definition der OeVGK18-Spezifikation findet kein Ende.	20	30%	6
2	PostGIS und pgRouting eignet sich nicht für die ÖV-Güteklasse-Berechnung.	40	25%	10
3	PostGIS und pgRouting sind technologisches Neuland.	24	30%	7.2
4	Es existieren viele Abhängigkeiten auf Fremdsysteme.	8	20%	1.6
5	Fremdsysteme bieten nicht die geforderte Datenqualität.	24	20%	4.8
6	Frontend-Framework eignet sich nicht für Karten-Integration.	16	30%	4.8

Tabelle 32: Risiken

In der Tabelle 32 sind die identifizierten Projektrisiken aufgelistet. Grundsätzlich lässt sich sagen, dass sich alle Risiken in einem vertretbaren Rahmen befinden. Das Hauptrisiko der Bachelorarbeit ist das technische Neuland und die fehlende Erfahrung mit PostGIS und pgRouting. Der Umgang und die Konsequenzen, welche sich aus der Risikoanalyse ableiten, sind in den folgenden Unterkapitel beschrieben. Die Risiken werden in einem zwei-wöchentlichen Rhythmus diskutiert und bei Unstimmigkeiten revidiert und neue Massnahmen getroffen.

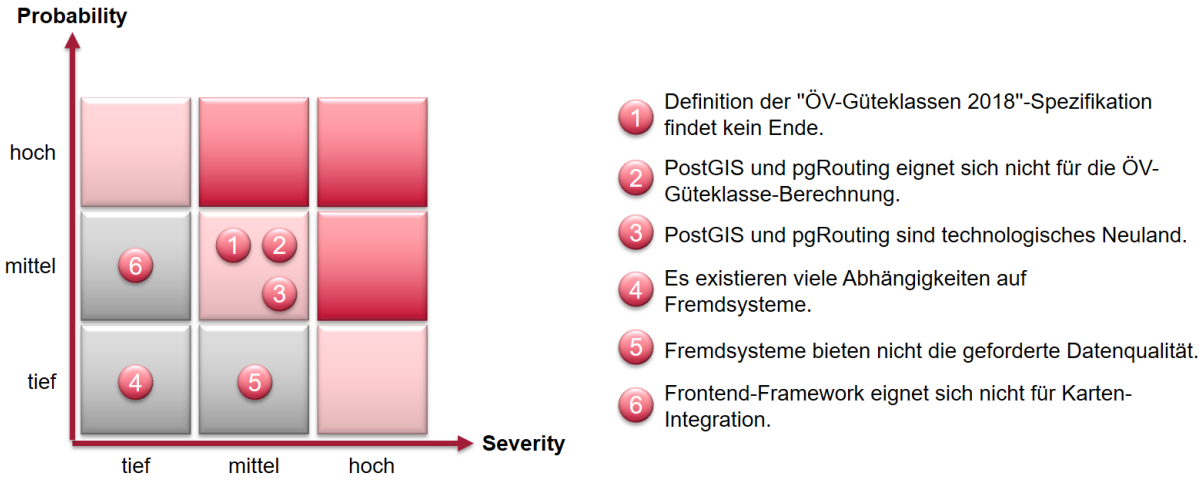

Abbildung 32: Risiko-Analyse

9.4.1 Umgang mit Risiken

Die Risikoanalyse hat einen erwarteten Mehraufwand von *34.4* Stunden ergeben. Den in Tabelle 32 identifizierten technischen Risiken (*ID 2, 3 und 6*) wird mit einer geplanten Einarbeitung und Prototypen

entgegen gewirkt. Damit die Spezifikation der OeVGK18 ein definitives Ende findet und mit der Umsetzung begonnen werden kann, wird ein fixer Milestone „Spezifikation umsetzungsbereit“ definiert, welcher zwei Überarbeitungsrunden beinhaltet.

9.4.2 Konsequenz

Falls die geplanten Massnahmen nicht den erwarteten Nutzen zeigen oder unerwartete Stolpersteine auftreten, wird der Use Case UC05: ÖV-Güteklassen ARE integrieren aus dem Scope gestrichen.

9.4.3 Risiko-Refinement

Alle zwei Wochen findet ein Risiko-Refinement statt. Zweck dieses Meeting ist es, dass die Risiken mit der aktuellen Situation abgeglichen werden. Bei Bedarf kann so rasch auf eintretende Risiken reagiert und mit den betroffenen Stakeholder einen Konsens über das weitere Vorgehen gefunden werden.

10 Softwaredokumentation

Grundsätzlich besteht die Software aus den Komponenten ÖV-Güteklassen 2018-Generator (siehe Kapitel 5.1), Backend (siehe Kapitel 5.2) und Web-App (siehe Kapitel 5.4).

Die Installations- und Benutzeranleitungen zu den Komponenten sind in den jeweiligen Github-Repositories gehalten.

Für einen raschen Einstieg sind hier die Verweise auf die einzelnen Anleitungen der verschiedenen Komponenten aufgeführt.

10.1 ÖV-Güteklassen 2018 Generator

Die Installations- und Benutzeranleitungen für den *ÖV-Güteklassen 2018-Generator* befindet sich auf Github [49].

10.2 Backend

Die Installations- und Benutzeranleitungen für das *Backend* befindet sich auf Github [73].

Die Web-API, welche das Backend exponiert, kann mit ReDoc (OpenAPI/Swagger-generated API Reference Documentation) [56] oder mit dem klassischen Swagger UI [57] betrachtet werden.

10.3 Web-App

Die Installations- und Benutzeranleitungen für das *Web-App* befindet sich auf Github [72]. Der *Tile-Server* sowie der *Tile-Converter* sind ebenfalls in dieser Anleitung beschrieben.

Glossar

Dijkstra-Algorithmus Sucht den kürzesten Pfad zwischen einem Start- und Endvertex auf einem Routing-Graphen.

Driving Distance Sucht mithilfe des Dijkstra-Algorithmus auf einem Routing-Graphen alle Vertices, welche innerhalb eines Kostenmaximums (z.B. Distanz) sind.

Geocoding Der Prozess, einer Postadresse eine Koordinate zuzuordnen. Der umgekehrte Weg, das Bestimmen einer Postadresse aus einer Koordinate, nennt sich *Reverse Geocoding*.

GeoJSON Ein spezifiziertes Format (RFC 7946) für die serialisierte Repräsentation von geografischen Daten. Dafür wird die JSON-Notation verwendet.

GeoJSON-Feature Beschreibt eine Geometrie im GeoJSON-Format. Dabei kann es sich um einen Punkt, Multipolygon, Linie, etc. handeln.

GEOS Steht für "'Geometry Engine, Open Source"', eine Software-Bibliothek, die ein Objektmodell für geometrische Daten bietet und geometrische Funktionen implementiert.

Haltestelle Wird, wenn nicht anders erwähnt, als eine ÖV-Haltestelle verstanden, an der ein öffentliches Verkehrsmittel hält. Dies kann z.B. ein Bahnhof, eine Bus- oder Tramhaltestelle oder auch eine Berg- oder Talstation einer Seilbahn sein.

Isochrone Eine Darstellung der Erreichbarkeit auf einer Karte von einem Standort aus. Definiert wird dies als geschlossene Linie um einen Standort, die alle Punkte miteinander verbindet, die mit gleicher Laufzeit von diesem Standort entfernt sind.

Kante Beschreibt eine Verbindung zwischen zwei Vertices.

Leistungskilometer Der Leistungskilometer berücksichtigt die Horizontaldistanz sowie die Werte, die sich aus Steigung und starker Gefälle errechnen lassen und ist ein Mass zur Abschätzung des Zeit- und Energieaufwands.

Nearest Neighbor Suche Sucht auf einem Routing-Graphen Vertices, welche am nächsten zu einem bestimmten Vertex sind.

OeVGK18 Steht kurz für ÖV-Güteklassen 2018 und bezeichnet die neue Spezifikation, welche im Zuge dieser Arbeit erarbeitet wird.

OeVGK93 Steht kurz für ÖV-Güteklassen 93 und bezeichnet die Definition der ÖV-Güteklassen, welche im Jahre 1993 mit der Schweizer Norm 640 290 verabschiedet wurde.

OeVGKARE Steht kurz für ÖV-Güteklassen des Bundestamt für Raumentwicklung und bezeichnet die Spezifikation und Umsetzung der ÖV-Güteklassen basierend auf OeVGK93 mit Erweiterungen.

OpenStreetMap Ein Community-Projekt mit dem Ziel, eine frei verfügbare Karte der Erde zu erstellen, die von jedem bearbeitet und ergänzt werden kann.

QGIS Ein frei verfügbares Geoinformationssystem für den Desktop für die Anzeige, Bearbeitung und Analyse von geografischen Daten.

Routing-Engine Eine Software, die aus Kartendaten einen Routing-Graphen aufbereitet und Funktionalitäten anbietet, um auf diesen Routen zu berechnen.

Routing-Graph Ein Verbund von Vertices, welche über Kanten miteinander verbunden ist und ein Strassen- und Wegenetzwerk abbildet, über das Routen berechnet werden können.

Stored Procedure Ist eine Funktion in einen Datenbankmanagementsystem, welche aufgerufen werden kann. Dadurch können häufig verwendete Abläufe, die sonst durch den Client ausgeführt werden müssen, auf das Datenbanksystem ausgelagert werden.

Terrainmodell Ein digitales Terrainmodell (DTM) beschreibt die Geländeform ohne Bewuchs und Bebauung.

Vertex Beschreibt den Anfang und das Ende einer Graph-Kante.

ÖV-Güteklassen Geben Auskunft darüber, wie gut ein Standort mit dem öffentlichen Verkehr erschlossen ist.

Abkürzungsverzeichnis

OSM OpenStreetMap

RUP Rational Unified Process

API Application Programming Interface

UML Unified Modeling Language

REST Representational State Transfer

CI Continuous Integration

ARE Bundesamt für Raumentwicklung

ÖV Öffentlicher Verkehr

PBF Protocolbuffer Binary Format

GTFS General Transit Feed Specification

SN Schweizer Norm

SPA Single Page Applications

UVEK Eidgenössisches Departement für Umwelt, Verkehr, Energie und Kommunikation

UIC International Union of Railways

SRTM Shuttle Radar Topography Mission

ORM Object-Relational Mapping

OAS OpenAPI Specification

CLI Command Line Interface

Wireframes

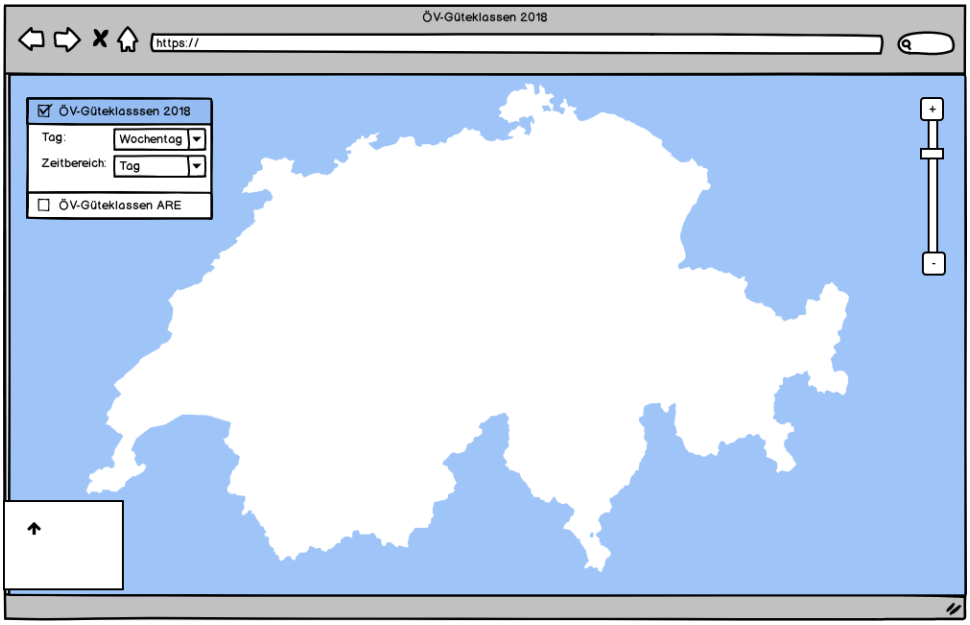

Abbildung 33: Wireframe der Standard-Ansicht

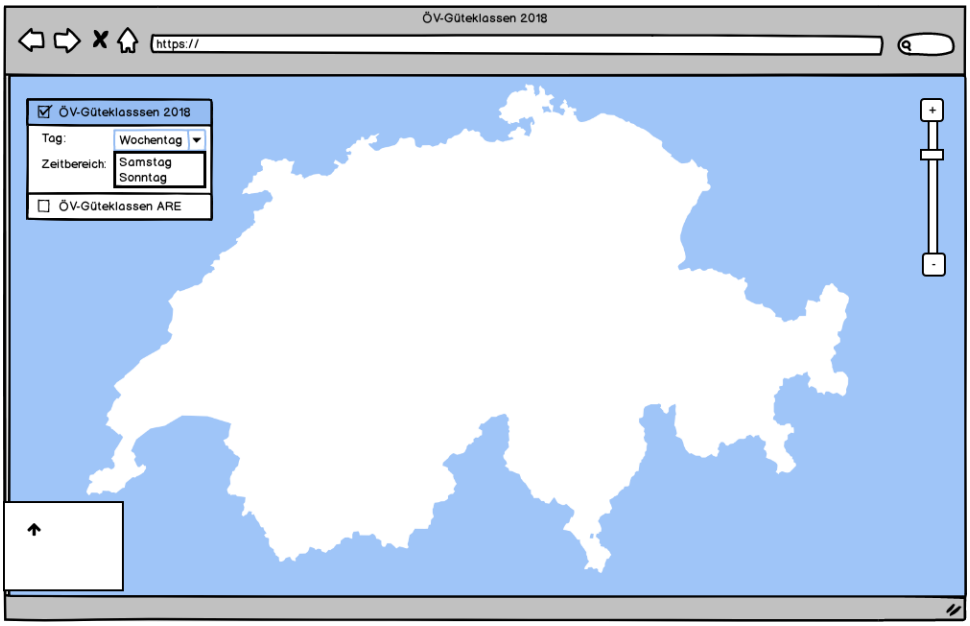

Abbildung 34: Wireframe zur Auswahl des Tages

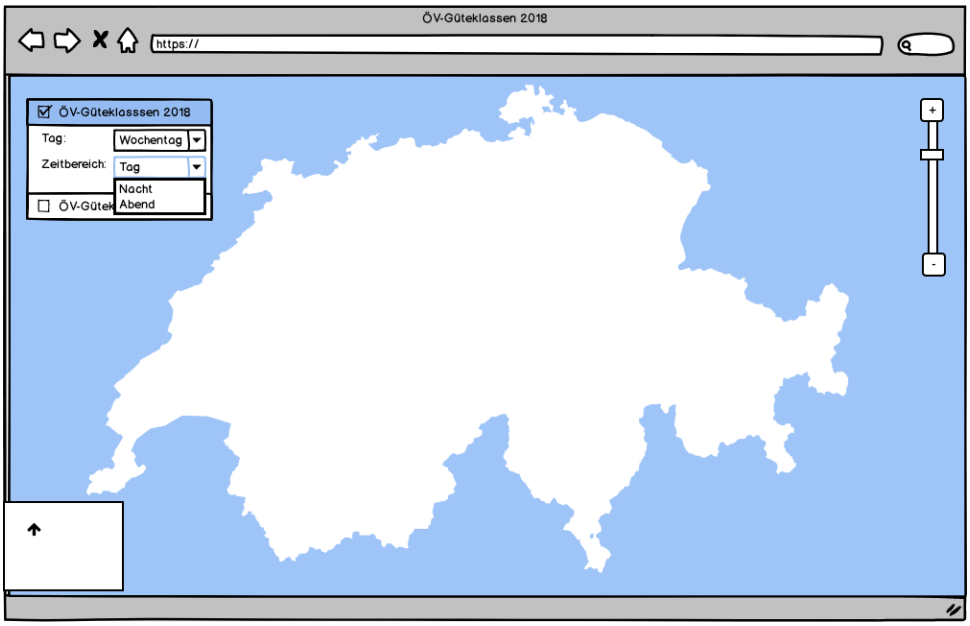

Abbildung 35: Wireframe zur Auswahl des Zeitbereichs

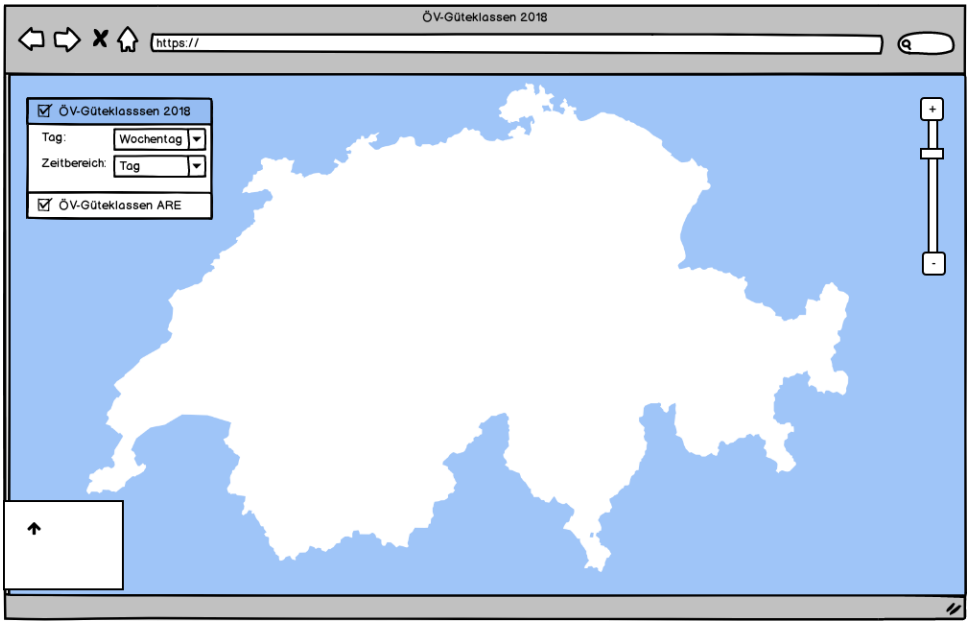

Abbildung 36: Wireframe, wenn ÖV-Güteklassen ARE ausgewählt wird

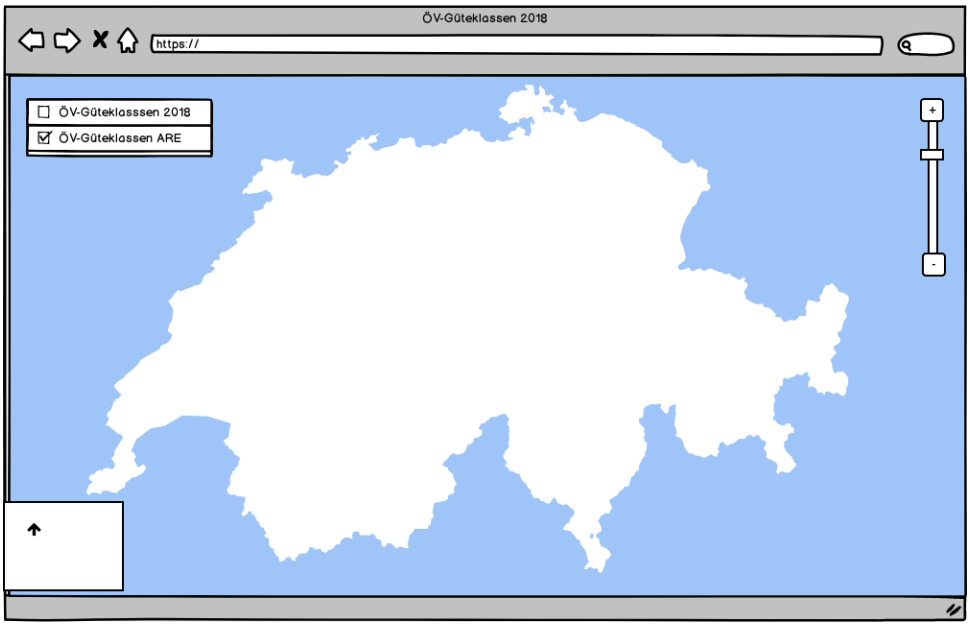

Abbildung 37: Wireframe, wenn nur die ÖV-Güteklassen ARE angewählt sind

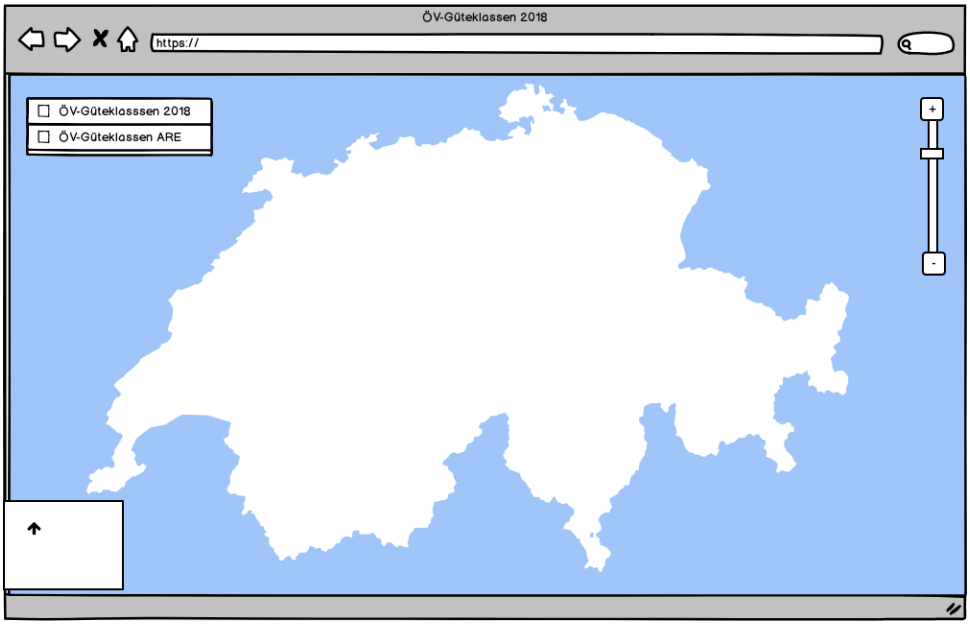

Abbildung 38: Wireframe, wenn alle Ansichten ausgeblendet werden

Literaturverzeichnis

[1] Bundesamt für Raumentwicklung ARE, "ÖV-Güteklassen — Berechnungsmethodik ARE," 2017, [Online; accessed 27-February-2018]. [Online]. Available: https://www.are.admin.ch/are/de/home/verkehr-und-infrastruktur/grundlagen-und-daten/verkehrserschliessung-in-der-schweiz.html

[2] J. Matter and R. Suter, "PlazaRoute: Fussgänger-Routing über offene Flächen im urbanen Raum," 2018. [Online]. Available: https://eprints.hsr.ch/625/

[3] S. Rossetti et al., "Pedestrian mobility and accessibility planning: some remarks towards the implementation of travel time maps," *CSE-City Safety Energy*, vol. 1, pp. 67–78.

[4] "ÖV-Güteklassen Kanton Graubünden," 2018, [Online; accessed 27-February-2018]. [Online]. Available: https://www.gr.ch/DE/institutionen/verwaltung/bvfd/aev/oev/oevgueteklassen/Seiten/default.aspx

[5] VSS, "Grenzbedarf, reduzierter Bedarf, Angebot," Vereinigung Schweizer Strassenfachleute VSS, Zürich, SN 640 290 Parkieren, 1993.

[6] Bundesamt für Raumentwicklung ARE, "RPG 1: Umsetzung der Siedlungsentwicklung nach innen," 2014, [Online; accessed 04-April-2018]. [Online]. Available: https://www.are.admin.ch/are/de/home/raumentwicklung-und-raumplanung/raumplanungsrecht/revision-des-raumplanungsgesetzes--rpg-/rpg-1--anspruchsvolle-umsetzung-der-siedlungsentwicklung-nach-in.html

[7] VSS, "Angebot an Parkfeldern für Personenwagen," Vereinigung Schweizer Strassenfachleute VSS, Zürich, SN 640 281 Parkieren, 2006.

[8] *INFO+ HRDF-Exportschnittstelle 5.20.39*, 2016, [Online; accessed 07-March-2018]. [Online]. Available: http://www.fahrplanfelder.ch/fileadmin/fap_daten_test/hrdf.pdf

[9] "swissALTI3D — swisstopo," 2017, [Online; accessed 05-March-2018]. [Online]. Available: https://shop.swisstopo.admin.ch/de/products/height_models/alti3D

[10] "Open Data Platform Swiss Public Transport," 2017, [Online; accessed 05-March-2018]. [Online]. Available: https://opentransportdata.swiss

[11] "Normierte gesamtverkehrliche Erschliessungsqualitäten - Grundlagenbericht," 2015, [On-

line; accessed 10-April-2018]. [Online]. Available: http://www.mobilityplatform.ch/de/shop/show-item/product/16057

[12] "Definition ÖV-Struktur / Erhebung ÖV–Güteklassen Kanton Graubünden," 2014, [Online; accessed 22-February-2018]. [Online]. Available: https://www.gr.ch/DE/institutionen/verwaltung/bvfd/aev/dokumentation/VDokumente/PDF_Bericht%20_OEV_Gueteklassen_GR_140314.pdf

[13] "ÖV–Güteklassen Infoblatt," 2017, [Online; accessed 13-March-2018]. [Online]. Available: https://www.geolion.zh.ch/geodatensatz/1989/downloadPDF

[14] "ÖV-Güteklassen – ein Werkzeug zur Analyse der Versorgung eines Standortes mit ÖV," *AGIT – Journal für Angewandte Geoinformatik*, no. 3, 2017.

[15] R. Rickborn, *VISUM 12 Grundlagen*, ser. Ptv vision. epubli GmbH, 2012, pp. 167–168.

[16] Eidgenössisches Departement für Umwelt, Verkehr, Energie und Kommunikation, "Programm Agglomerationsverkehr - Agglomerationsprogramme Verkehr und Siedlung," 2015, [Online; accessed 27-March-2018]. [Online]. Available: https://www.are.admin.ch/are/de/home/verkehr-und-infrastruktur/programme-und-projekte/agglomerationsprogramme-verkehr-und-siedlung.html

[17] G. T. Doran, "There's a s.m.a.r.t. way to write managements's goals and objectives." *Management Review*, vol. 70, no. 11, pp. 35–36, 1981.

[18] L. Bass et al., *Software Architecture in Practice*, 3rd ed. Addison-Wesley Professional, 2012.

[19] "PostGIS," 2018, [Online; accessed 14-May-2018]. [Online]. Available: https://postgis.net/

[20] "pgRouting," 2018, [Online; accessed 14-May-2018]. [Online]. Available: https://pgrouting.org/

[21] "Project OSRM — Open Source Routing Machine," 2018, [Online; accessed 14-May-2018]. [Online]. Available: http://project-osrm.org/

[22] "GraphHopper," 2018, [Online; accessed 14-May-2018]. [Online]. Available: https://www.graphhopper.com/

[23] "Github: mapbox/osrm-isochrone," 2018, [Online; accessed 14-May-2018]. [Online]. Available: https://github.com/mapbox/osrm-isochrone

[24] "React — A JavaScript library for building user interfaces," 2018, [Online; accessed 13-March-2018]. [Online]. Available: https://reactjs.org/

[25] "Vue.js — The Progressive JavaScript Framework," 2018, [Online; accessed 13-March-2018]. [Online]. Available: https://vuejs.org/

[26] "Github: public-transport-quality-grades/playground," 2018, [Online; accessed 18-March-2018]. [Online]. Available: https://github.com/public-transport-quality-grades/playground

[27] "JSX," 2018, [Online; accessed 13-March-2018]. [Online]. Available: https://jsx.github.io

[28] "Leaflet," 2018, [Online; accessed 13-March-2018]. [Online]. Available: http://leafletjs.com

[29] "Github: PaulLeCam/React-Leaflet," 2018, [Online; accessed 13-March-2018]. [Online]. Available: https://github.com/PaulLeCam/react-leaflet

[30] "Github: KoRiGaN/Vue2Leaflet," 2018, [Online; accessed 13-March-2018]. [Online]. Available: https://github.com/KoRiGaN/Vue2Leaflet

[31] "Vector Tiles — Die Webkarten-Technologie der Zukunft?" 2018, [Online; accessed 18-March-2018]. [Online]. Available: http://geometalab.tumblr.com/post/169805467537/vector-tiles-die-webkarten-technologie-der

[32] "Mapbox GL JS," 2018, [Online; accessed 18-March-2018]. [Online]. Available: https://www.mapbox.com/mapbox-gl-js/api/

[33] "OpenMapTiles," 2018, [Online; accessed 18-March-2018]. [Online]. Available: https://openmaptiles.org/

[34] "Github: alex3165/react-mapbox-gl," 2018, [Online; accessed 18-March-2018]. [Online]. Available: https://github.com/alex3165/react-mapbox-gl

[35] "Github: phegman/vue-mapbox-gl," 2018, [Online; accessed 18-March-2018]. [Online]. Available: https://github.com/phegman/vue-mapbox-gl

[36] "Github: facebook/create-react-app," 2018, [Online; accessed 18-March-2018]. [Online]. Available: https://github.com/facebook/create-react-app

[37] "Github: vuejs/vue-cli," 2018, [Online; accessed 18-March-2018]. [Online]. Available: https://github.com/vuejs/vue-cli

[38] "records – sql for humans," 2018, [Online; accessed 05-April-2018]. [Online]. Available: https://pypi.python.org/pypi/records/

[39] "Shapely," 2018, [Online; accessed 05-April-2018]. [Online]. Available: https://github.com/Toblerity/Shapely

[40] "C4 model," 2018, [Online; accessed 20-February-2018]. [Online]. Available: https://c4model.com/

[41] "geops — Public Transporation Feed for Switzerland," 2018, [Online; accessed 07-March-2018]. [Online]. Available: http://gtfs.geops.ch/

[42] "General Transit Feed Specification," 2018, [Online; accessed 07-March-2018]. [Online]. Available: http://gtfs.org/

[43] "Osm Data for Switzerland," 2018, [Online; accessed 07-March-2018]. [Online]. Available: https://planet.osm.ch/

[44] R. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*, ser. Robert C. Martin series. Prentice Hall, 2017, ch. 22.

[45] "OSM2PO," 2017, [Online; accessed 24-April-2018]. [Online]. Available: http://osm2po.de/

[46] "raster2pgsql," 2017, [Online; accessed 24-April-2018]. [Online]. Available: https://postgis.net/docs/using_raster_dataman.html

[47] "PostgreSQL 10.3 Documentation," 2018, [Online; accessed 09-May-2018]. [Online]. Available: https://www.postgresql.org/docs/current/static/

[48] "Bundesamt für Verkehr BAV: TU-Verzeichnis," 2018, [Online; accessed 14-June-2018]. [Online]. Available: https://www.bav.admin.ch/bav/de/home/themen-a-z/verzeichnisse/tu-verzeichnis.html

[49] "Github: public-transport-quality-grades/oevgk18-generator," 2018, [Online; accessed 13-June-2018]. [Online]. Available: https://github.com/public-transport-quality-grades/oevgk18-generator

[50] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec 1959. [Online]. Available: https://doi.org/10.1007/BF01386390

[51] N. Akkiraju, H. Edelsbrunner, M. Facello, P. Fu, E. Mucke, and C. Varela, "Alpha shapes: definition and software," in *Proceedings of the 1st International Computational Geometry Software Workshop*, vol. 63, 1995, p. 66.

[52] "Flask," 2018, [Online; accessed 10-June-2018]. [Online]. Available: http://flask.pocoo.org

[53] "Maturity model," 2018, [Online; accessed 10-June-2018]. [Online]. Available: https://martinfowler.com/articles/richardsonMaturityModel.html

[54] "Github: Flasgger," 2018, [Online; accessed 10-June-2018]. [Online]. Available: https://github.com/rochacbruno/flasgger

[55] "Openapi specification," 2018, [Online; accessed 10-June-2018]. [Online]. Available: https://swagger.io/specification

[56] "Oevgk18 backend api spezifikation," 2018, [Online; accessed 10-June-2018]. [Online]. Available: https://public-transport-quality-grades.github.io/backend-api/

[57] "Oevgk18 backend api swaggerui," 2018, [Online; accessed 10-June-2018]. [Online]. Available: https://public-transport-quality-grades.github.io/backend-api/swagger-ui

[58] "Github: TileServer GL," 2018, [Online; accessed 11-June-2018]. [Online]. Available: https://github.com/klokantech/tileserver-gl

[59] "Github: MBTiles Specification," 2018, [Online; accessed 11-June-2018]. [Online]. Available: https://github.com/mapbox/mbtiles-spec

[60] "Github: tippecanoe," 2018, [Online; accessed 11-June-2018]. [Online]. Available: https://github.com/mapbox/tippecanoe

[61] "Github: mapbox/mapbox-gl-leaflet," 2018, [Online; accessed 11-June-2018]. [Online]. Available: https://github.com/mapbox/mapbox-gl-leaflet

[62] "Github: mapbox-gl-js — Diskussion zur Einführung der Deckkraft für Ebenen," 2018, [Online; accessed 11-June-2018]. [Online]. Available: https://github.com/mapbox/mapbox-gl-js/issues/4090

[63] "Github: Leaflet.VectorGrid," 2018, [Online; accessed 11-June-2018]. [Online]. Available: https://github.com/Leaflet/Leaflet.VectorGrid

[64] "pytest," 2018, [Online; accessed 07-June-2018]. [Online]. Available: https://pytest.org/

[65] "pgtap," 2018, [Online; accessed 07-June-2018]. [Online]. Available: https://pgtap.org/

[66] "Jest – Delightful JavaScript Testing," 2018, [Online; accessed 08-April-2018]. [Online]. Available: https://facebook.github.io/jest/

[67] "Flow," 2018, [Online; accessed 07-April-2018]. [Online]. Available: https://flow.org/

[68] "Github: public-transport-quality-grades/thesis," 2018, [Online; accessed 20-February-2018]. [Online]. Available: https://github.com/public-transport-quality-grades/thesis

[69] "Travis CI," 2018, [Online; accessed 20-February-2018]. [Online]. Available: https://travis-ci.org/

[70] "Github: public-transport-quality-grades," 2018, [Online; accessed 20-February-2018]. [Online]. Available: https://github.com/public-transport-quality-grades

[71] "Github: public-transport-quality-grades/oevgk18-specification," 2018, [Online; accessed 13-June-2018]. [Online]. Available: https://github.com/public-transport-quality-grades/oevgk18-specification

[72] "Github: public-transport-quality-grades/web-app," 2018, [Online; accessed 13-June-2018]. [Online]. Available: https://github.com/public-transport-quality-grades/web-app

[73] "Github: public-transport-quality-grades/backend," 2018, [Online; accessed 13-June-2018]. [Online]. Available: https://github.com/public-transport-quality-grades/backend

[74] "Github: public-transport-quality-grades/backend-api," 2018, [Online; accessed 13-June-2018]. [Online]. Available: https://github.com/public-transport-quality-grades/backend-api

[75] "Github: public-transport-quality-grades/admin," 2018, [Online; accessed 13-June-2018]. [Online]. Available: https://github.com/public-transport-quality-grades/admin

[76] "Github: public-transport-quality-grades/infrastructure," 2018, [Online; accessed 13-June-2018]. [Online]. Available: https://github.com/public-transport-quality-grades/infrastructure

[77] "The Boost Graph Library," 2018, [Online; accessed 06-June-2018]. [Online]. Available: https://www.boost.org/doc/libs/1_67_0/libs/graph/doc/index.html

[78] B. Barney, "Introduction to Parallel Computing," 2018. [Online]. Available: https://computing.llnl.gov/tutorials/parallel_comp/

[79] "Geometa Lab at HSR," 2018, [Online; accessed 20-February-2018]. [Online]. Available: https://www.ifs.hsr.ch/index.php?id=12520

[80] "Structurizr," 2018, [Online; accessed 20-February-2018]. [Online]. Available: https://structurizr.com/

[81] "CircleCI," 2018, [Online; accessed 20-February-2018]. [Online]. Available: https://circleci.com/

[82] "Isodistance and Isochrone Maps," 2016, [Online; accessed 06-March-2018]. [Online]. Available: https://www.gislounge.com/isodistance-isochrone-maps/

[83] "Github: Osmosis," 2017, [Online; accessed 24-April-2018]. [Online]. Available: https://github.com/openstreetmap/osmosis

[84] "Github: Manual zur Vorbereitung der Datenbank," 2018, [Online; accessed 02-May-2018]. [Online]. Available: https://github.com/public-transport-quality-grades/oevgk18-generator/tree/master/docker

Abbildungsverzeichnis

Tabellenverzeichnis

Code Listings