Evaluation of Open Data Using the Example of a Public Transport Navigation Website

BACHELOR THESIS

Daniel Langbein

Submitted on 24 May 2023

Friedrich-Alexander-Universität Erlangen-Nürnberg
Faculty of Engineering, Department Computer Science
Professorship for Open Source Software

Supervisor:
Philip Heltweg, M.S.
Prof. Dr. Dirk Riehle, M.B.A.

Friedrich-Alexander-Universität
Faculty of Engineering

Declaration of Originality

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, 24 May 2023

License

Erlangen, 24 May 2023

Abstract

This thesis deals with the implementation of an open source trip planner for the public transport association VGN, the results of which take into account both parking the bicycle at a stop and taking the bicycle on public transport until the destination. In principle, software already exists that enables multimodal navigation with public transport, and the timetable and road data sets required for this are freely available for many regions. For the VGN, however, there is no trip planner yet that prioritizes bicycle-assisted navigation with public transit. During the implementation of such a project, the difficulties that arise when integrating open data sources were investigated. Even though there are limitations due to poor quality of and missing information in the data sources, it was achieved to develop a trip planner that suggests bicycle-assisted journeys, which are often faster and involve fewer transfers. In addition, the software solution allows for easy configuration and reuse in other regions. While the open source components used offer several possibilities to further improve the trip planner in the future (e.g., by integrating rental bicycles), data sets covering these aspects are currently missing.

Contents

List of Figures

List of Tables

Acronyms

API	application programming interface
BEG	Bayerische Eisenbahngesellschaft mbH
CC BY	Creative Commons Attribution License
CC BY 4.0	Creative Commons Attribution 4.0 International License
CC BY 3.0 DE	Creative Commons Namensnennung 3.0 Deutschland
CEN	Comité Européen de Normalisation
CSV	comma-separated values
DB	Deutsche Bahn AG
DELFI	Verein zur Förderung einer durchgängigen elektronischen Fahrgastinformation
DEM	digital elevation model
ES5	ECMAScript 2009
EUPL	European Union Public Licence
GBFS	General Bikeshare Feed Specification
GNU AGPLv3	GNU Affero General Public License v3.0
GTFS	General Transit Feed Specification
GTFS-RT	GTFS Realtime
HSL	Helsinki Regional Transport Authority
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ITS	Intelligent Transport Systems
JSON	JavaScript Object Notation

LGPL	GNU Lesser General Public License
MMTIS	EU-wide multimodal travel information services
NeTEx	Network Timetable Exchange
ODbL	Open Data Commons Open Database License
OSM	OpenStreetMap
OTP	OpenTripPlanner
REST	representational state transfer
SIRI	Service Interface for Real-time Information
SRTM	Shuttle Radar Topography Mission
TEN-T	Trans-European Transport Network
Transmodel	Public Transport Reference Data Model
TriMet	Tri-County Metropolitan Transportation District of Oregon
UI	user interface
UML	Unified Modelling Language
URL	Uniform Resource Locator
VAG	VAG Verkehrs-Aktiengesellschaft
VGN	Verkehrsverbund Großraum Nürnberg
WebGL	Web Graphics Library
WOF	Who's On First
XML	eXtensible Markup Language

1 Introduction

Statewide as well as regional public transportation trip planners are often based on proprietary or non-public data and most of them are closed source. This makes it impossible to customize or extend these software solutions, which, with few exceptions, only support walking as personal navigation mode.

A search as a pedestrian for travel connections by public transport has its limitations: If there are no stops near the origin or destination, travel connections either take a long time due to walking distances or are not found at all. Both can result in people taking the car instead of public transport in order to get to their destination more quickly and without being tied to a timetable.

In Germany, car journeys are to be reduced by improving alternative offerings. In its coalition agreement, the German government set itself the goal to expand the public transport network and realize more frequent connections (Franz, 2021). At the same time, it sees the expansion of bicycle traffic as part of achieving its climate goals (MDR.DE, 2023).

The use of public transport and cycling can go hand in hand, because by bike distant stops can be reached faster, which increases the area that can be reached within the same travel time. However, planning trips by bicycle and public transportation requires local knowledge to choose bicycle-friendly routes and to know which public stop provides the shortest connection to the destination.

The goal of this work is to develop a web application that can be used to find multimodal itineraries that combine bicycle segments with public transport. The search will be limited to the Nuremberg metropolitan region, or more precisely the operational area of the Verkehrsverbund Großraum Nürnberg (VGN), which is the largest transportation association in Bavaria (Bayerisches Staatsministerium für Wohnen, Bau und Verkehr, n.d.).

To enable further use of the application, it wil be implemented as open source, based solely on open data. There will be a focus on working with open data: It will be investigated how well open data sources can be found, how they can be integrated into a data pipeline and what their quality is.

The following chapters describe the development of the open data based web application in detail. As an introduction, common open data standards are presented. Based on this, various open data sources are compared and two of them are selected as basis for the project. Complementary, the following chapter gives an overview of already existing trip planners. After these general explanations, the realization of the project is described. For this purpose, requirements are first defined that are to be fulfilled by the web application. Chapter 5 explains the structure and components of the application, including the pipeline used to import data. Subsequently, important implementation details are highlighted. Finally, it is reflected whether the end result fulfills the requirements and which problems arise from the selected data sources.

2 Data Sources and Their Quality

2.1 Preliminaries

This section introduces the terminology used in this thesis when talking about public transit networks.

2.1.1 Public Transit Network

Baum et al. (2019, pp. 3–4) give a technical definition in which a public transit network is a 4-tuple consisting of stops, trips, routes and transfers. We follow their definition, but remain less technical.

A passenger can board or disembark vehicles (such as trains, subways or buses) at *stops*. *Trips* are sequences of at least two stops which are served by the same vehicle one after the other. Trips with the same stop sequence are grouped as *routes*. A trip leg is the part of a trip between a passenger's boarding and disembarking. Passengers *transfer* with the help of private transportation (such as walking or cycling). This may be during the initial transfer from their starting point to the first stop, from the last stop to the destination, or during intermediate transfers between trip legs.

2.1.2 Transport Modes and Multimodality

Modes of transportation can be partitioned as follows (2017/1926/EU, 2017, p. 11):

- *Scheduled*: Modes of transportation that run on a schedule such as train, subway or bus.
- *Demand-responsive*: Non-scheduled transportation such as taxi, car-sharing or bike-hire.
- *Personal*: Personal transportation options such as walking, biking or driving.

A route planning is called *multimodal* when different modes of transportation are combined.

To refer to the combination of certain transport modes, the following terms are used:

- *Transit*: Travel by public transport. Usually, this involves the combination of all scheduled modes of transport. Strictly speaking, only public transport stops are possible as the start and destination of such journeys. But often arbitrary addresses can be searched, from which nearby stops are selected as entry and exit points to public transportation.
- *Walk and Transit*: Travel by public transportation, which may include longer walks at the beginning, between trip legs, or at the end. Any start and destination locations are possible.
- *Bike and Transit*: Travel by public transportation, where, unlike *Walk and Transit*, transfers are made by bicycle.
- *Bike and Ride*: This term is analogous to *Park and Ride*. The initial transfer to the first stop is done by bicycle, which is then parked there. Thereafter, the journey is continued with *Transit* or *Walk and Transit*.

2.2 Common Data Standards

The data structures of schedule-based and demand-responsive public transportation are significantly more complex than the graphs used for private navigation modes. In the following, we therefore present three formats commonly used for public transportation.

2.2.1 GTFS and GTFS-RT

Due to the complexity of time-based public networks, a clear standard is needed for the exchange of transit data. Probably the most widely used is the General Transit Feed Specification (GTFS) with Google (2020) listing multiple thousand agencies around the world providing a GTFS feed for them.

The specification originated in 2005 as a result of the transit agency Tri-County Metropolitan Transportation District of Oregon's (TriMet) desire to integrate their public transit connections with external trip planners, one of which was Google Maps (Roth, 2010). At present, the data standard is maintained by the non-profit organization MobilityData (MobilityData, n.d.-a).

GTFS "defines a common format for public transportation schedules and associated geographic information" (Google, 2022b) required to do trip planning. This includes the operating transit agencies, the location of stops, stations and platforms as well as the departure and arrival times of vehicles at stops. It is optional to include fare information and vehicle travel paths (Antrim et al., 2023). The latter is necessary to represent the route of a vehicle more precisely than just given by its stop locations.

GTFS datasets consist of text files in the comma-separated values (CSV) format[1] and should be published as one zip file under a public, permanent URL (Antrim et al., 2023). This technically simple format has probably also led to its widespread use. Nevertheless, there is still a problem-inherent workload for the transit agencies to export GTFS from their transit system. This is because the data model of GTFS, as shown in Figure 2.1, is by no means simple.

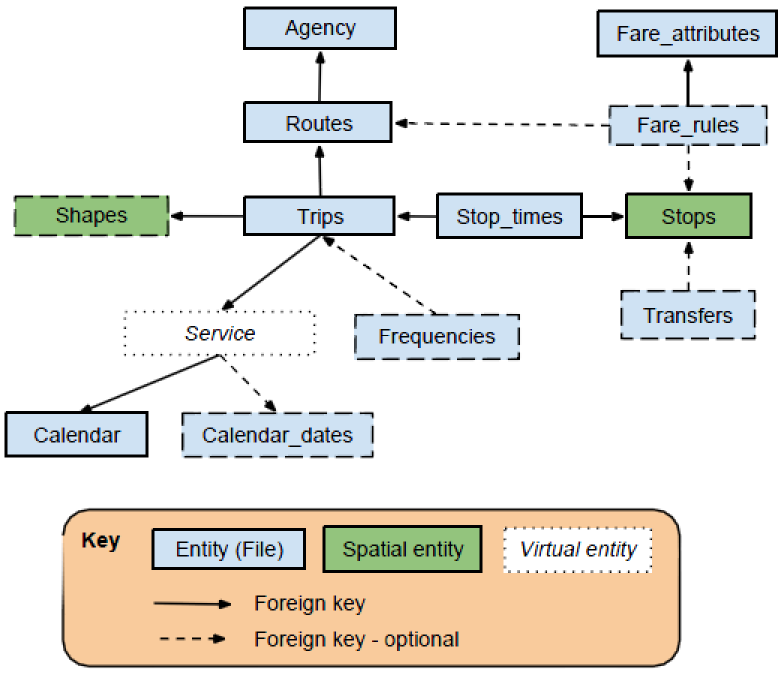


Figure 2.1: GTFS data model. From Davis, M. (2011, September 27). *Data model diagrams for GTFS* [Lin.ear th.inking]. Retrieved April 24, 2023, from https://lin-ear-th-inking.blogspot.com/2011/09/data-model-diagrams-for-gtfs.html

GTFS Realtime (GTFS-RT) is an extension to GTFS, which allows transit agencies to provide the current departures and arrival times, occupancy rates of vehicles as well as schedule or stop changes (MobilityData, 2022).

[1]https://datatracker.ietf.org/doc/html/rfc4180

2.2.2 NeTEx and SIRI

Transmodel is the European reference data model for public transport (5T s.r.l., n.d.-a). It is a high-level conceptual model documented as entity-relationship model and in the Unified Modelling Language (UML) that is standardised by the European Committee for Standardization (Comité Européen de Normalisation; CEN) and covers "public transport concepts across a wide range of functional areas" (Data4PT, 2020). It defines common terminology and fundamental concepts in an implementation independent way (Data4PT, 2020) which facilitates interoperability across different organisations and systems (5T s.r.l., 2019). The fact that such an exchange of travel information between many different IT systems is necessary to support travellers on their journeys is symbolically shown by Figure 2.2.

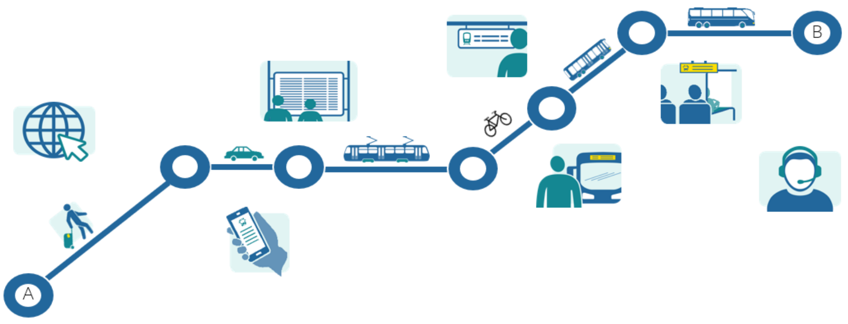


Figure 2.2: Consistent travel information is needed before and during a trip at various stages. From 5T s.r.l. (2019, September). Transmodel at a glance. Retrieved April 29, 2023, from https://www.transmodel-cen.eu/wp-content/uploads/2015/01/Transmodel_at_a_-glance-1.pdf

The Network Timetable Exchange (NeTEx) and the Service Interface for Real-time Information (SIRI) are both XML based implementations of the Transmodel that are standardized by CEN. The relation between the conceptual model Transmodel and NeTEx as one of its implementations is visualized in Figure 2.3.

Thereby NeTEx forms a standard for the exchange of public transport schedules and related data (5T s.r.l., n.d.-c). It has a much wider scope than GTFS which is only intended for provisioning journey planning systems (5T s.r.l., n.d.-b). NeTEx on the contrary, is intended for use in back office (5T s.r.l., n.d.-b). The more technical specification with an XML schema allows to transfer a complete dataset in only one XML file and to perform automatic validation (5T s.r.l., n.d.-b; Arneodo, 2015).

Using the SIRI framework it is additionally possible to exchange real-time information about public transport services with NeTEx (Arneodo, 2015).

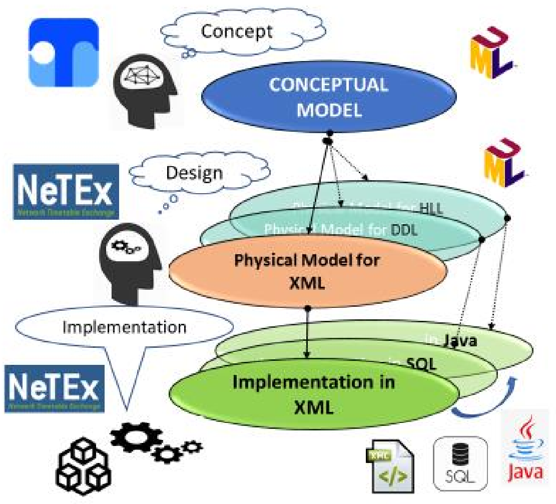


Figure 2.3: Transmodel and NeTEx cover different levels of data specification. From Data4PT. (2020, January 31). Methodology for comparing data standards. Retrieved April 29, 2023, from https://data4pt-project.eu/wp-content/uploads/2021/03/Data4PT-Methodology-for-comparing-data-standards.pdf

With its capabilities, NeTEx forms a superset, of which the individual member countries implement only a subset in national profiles. The NeTEx website lists currently four profiles (5T s.r.l., n.d.-c). However, on the NeTEx GitHub wiki of Knowles (2020b), one of the members of the NeTEx working group (Knowles, 2020a), several more can be found. These variations complicate the use of NeTEx in software projects. For example, OpenTripPlanner presented in the following chapter 3 so far only supports data import of the nordic profile used by Norway, Finland, Sweden and Denmark (Knowles, 2020b; Thomas Gran et al., 2022).

Due to the larger scope, both static NeTEx data and real-time information via SIRI can be mapped to GTFS and GTFS-RT respectively.[2] In the opposite direction, a complete NeTEx dataset cannot be generated from GTFS (5T s.r.l., n.d.-b), though the NeTEx wiki (2023) lists converters for some of the profiles.

2.2.3 GBFS

In addition to GTFS, MobilityData (n.d.-b) is also developing the General Bike-share Feed Specification (GBFS), a data standard enabling the exchange real-time

[2]NeTEx has a UML package that shows how a mapping to GTFS is possible (5T s.r.l., n.d.-b). And also with SIRI, thanks to the XML schema, it is possible to generate for example Java classes of the data model, thus reducing the effort to create a converter to GTFS-RT (CEN TC 278 Working Group 3 Sub Group 7, 2005; MobilityData, n.d.-c).

data on demand-responsive mobility. This includes rental bikes and e-scooters.

2.3 Available Open Data Sources

In the following, we show which open data sources are available that can be used by our project for journey planning in the VGN area. Thereby, the data sources can be divided into scheduled public transport, a pedestrian and bicycle network, and demand-responsive mobility. In this process, we evaluate the data quality of suitable sources and decide to take the GTFS feed of the VGN for schedule-based navigation and OpenStreetMap (OSM) as a basis for the pedestrian and bicycle road graphs. Finally, we conclude that there are still no open data sources for demand-responsive mobility and real-time updates in our region.

2.3.1 National Access Points

In the EU, a framework for Intelligent Transport Systems (ITS) has been in place since 2010 (2010/40/EU, 2010). With Regulation 2017/1926/EU (2017), this was expanded to include the provision of EU-wide multimodal travel information services (MMTIS). In it, the member states were given a timetable for providing open access to data needed for, among other things, multimodal travel planning. The data is to be available at a central location for each country, the so-called National Access Points. In addition, concrete data formats are prescribed. Static public transport data for instance shall use the NeTEx and dynamic public transport data the SIRI format. So far, the provision of real-time data is still optional.

Since December 2021, the National Access Points must include the following static travel data for the comprehensive Trans-European Transport Network (TEN-T):

- Address identifiers and points of interests for location search.
- Data required to perform trip planning with scheduled and private modes. Besides the timetable-based public transport data this includes road, pedestrian and cycle networks. The latter including details about the bike track, such as whether it is a lane on the road or a shared pedestrian and bike path, as well as details about surface quality and weather it is a scenic route.
- Bike parking, park and ride stops and bike sharing locations required for demand-responsive shared-mobility modes.
- Booking information including fare zones, passenger classes, where to pay for car parking and possible payment methods.

This amount of data with the details mentioned above would be perfect to implement a trip planner with scheduled, personal and demand-responsive modes.

However, the comprehensive TEN-T network only covers long-distance routes and does not provide nearly enough details in the VGN area to enable walking, cycling or bus navigation. This can be seen on the map depicted in Figure 2.4 where the road and rail networks of TEN-T are displayed as red, green and purple overlay. Furthermore, there is no separate bicycle network as part of TEN-T as the European Cyclists' Federation (2020) criticises. The data criteria listed in the regulation should therefore be understood more as a recommendation to the member states on what they can additionally publish to improve the capabilities of multimodal travel planning.

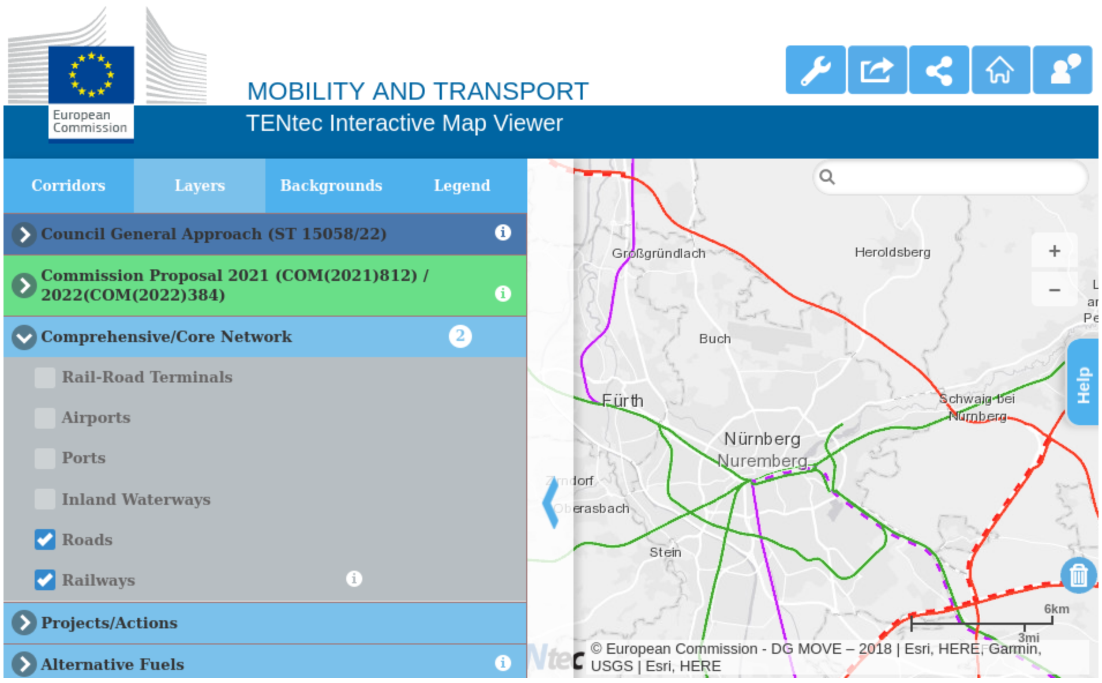


Figure 2.4: The TENtec Interactive Map Viewer.

The Mobility and Transport website of the European Commission contains a regularly updated document listing the current National Access Points.[3] In this document, the column *MMTIS National Access Point* is of particular interest to us.

For Germany, the Mobility Data Marketplace is listed there (European Comission, 2022), which will be migrated to the new Mobilithek platform by the end of 2023 (Bundesanstalt für Straßenwesen, n.d.). If we search on it[4] for scheduled public transport with a free-of-charge license, we get three results. Two of them include timetable data: The first one is a GTFS feed covering bus in the district

[3]https://transport.ec.europa.eu/transport-themes/intelligent-transport-systems/road/action-plan-and-directive/national-access-points_en

[4]https://service.mdm-portal.de/mdm-portal-application/publicationSearch.do

Fulda and the second one a NeTEx feed for public transport in whole Germany provided by DELFI e.V., a registered association providing seamless passenger information in Germany. To access the nationwide dataset we are redirected to the open data website of the transport association Rhein-Ruhr[5]. There we can see that the data is also available as GTFS feed. Both are licensed under CC BY, but the version number of the license is missing and a free account is required for a download from the website. Other means of authenticated download are not mentioned. Even though these are two open data sources, the registration and web login make it difficult to actually use them in an automated data pipeline – which would make it far easier to integrate the weekly published updates.

To get a rough overview of the geographic coverage of the DELFI dataset we decided to inspect the GTFS format and marked all stops included in it in red on a map. We chose the GTFS zip file because with 300 MB it is less than a quarter of the size of the NeTEx dataset[6] and the format is technically simpler while still containing the same timetable information that is relevant for us. Two sections of the resulting map view can be seen in Figure 2.5. Within Germany, the density of stops at the selected scale is so high that its area is colored red without any visible gaps. Larger cities in neighboring countries, for example Paris, show that the dataset also includes long-distance train or bus connections across national borders up to their final stop. But we also see that some stops are incorrectly located. For example, there is one point east of the Caspian Sea, several points south-west of Western and Central Africa in the Atlantic Ocean, and one far below Antarctica. We did not investigate further what the stop names of the wrongly located stops are, or which transit agency they belong to. But since the DELFI data set is a combination of the data from 12 country systems with long-distance traffic information (Transport association Rhein-Ruhr, n.d.), it is possible that the errors all originate from just one of the underlying data sources.

For our use case, a small slice of the GTFS data is enough, which we can obtain either by intersecting it with the outline of the VGN[7] or by filtering it by the 135 transit agencies of the VGN association. In the end, we decided to use the GTFS feed offered directly by the VGN instead, eliminating the need for processing. More about this is covered in subsection 2.3.2.

If we search the Mobility Data Marketplace for scheduled public transport data, regardless of license terms, then there are no further results that intersect with the VGN area. Moreover, no real-time data sources are listed apart from one GTFS-RT feed of the FlixBus inter-city bus service when searching for real-time traffic data.

[5]https://www.opendata-oepnv.de/ht/en/organisation/delfi/start

[6]300 versus 1317 MB as of April 24, 2023.

[7]With e.g. gtfs-filter (https://github.com/twalcari/gtfs-filter) it is possible to extract a bounding box of a larger GTFS file.

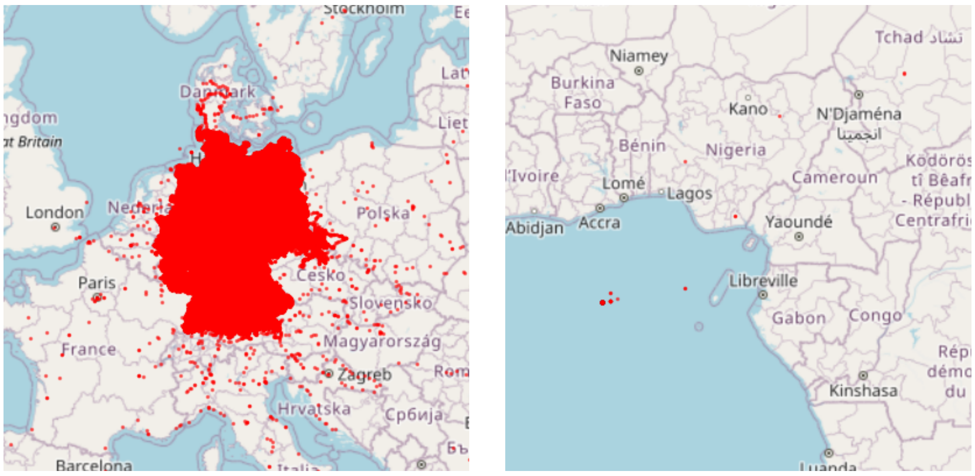


Figure 2.5: Stop locations included in the DELFI GTFS dataset. Copyright of tiles by OSM contributors.

In addition to data for schedule-based public transportation, we also need a data source for the personal modes walking and cycling. The bicycle network category provides only 12 Germany-wide bicycle routes. This level of detail is roughly on par with the long-distance TEN-T network, but unsuitable for our regional area. Therefore, another data source is needed, which we found with OSM and discuss in more detail in subsection 2.3.3. This leaves only demand-responsive services which are covered separately in subsection 2.3.5.

2.3.2 Data Sources From Public Service Providers

The German National Access Point lists by far not all data sets and open APIs from the transport sector. It is therefore worthwhile to additionally check directly with the transit agencies as they often have a webpage dedicated to open data. This is also the case for the VGN and the VAG Verkehrs-Aktiengesellschaft (VAG), a municipal company providing train, tram, bus and rental bikes in Nuremberg that is part of the VGN association (VAG, n.d.-a).

The only data set offered by the VGN is an annually updated GTFS feed of the entire transport network licensed under CC BY 3.0 DE (VGN, 2022). The infrequent updates are a clear disadvantage compared to the DELFI dataset. A replacement bus that runs additionally during construction work on the rails, for example, is potentially not covered – even if such construction work is announced well in advance. On the other hand, the dataset has the advantage that it is tailored exactly to our target region. We therefore decided to use the VGN dataset to drive the schedule-based navigation modes. This allows us to eliminate one data pipelining step from the scope of this work: We do not need to filter the GTFS dataset by region.

When calculating the convex hull around all stops of the VGN source we get the result visible in Figure 2.6. This gives us a good approximation of the area of the VGN and we see that there are no such large location errors as in the DELFI GTFS feed. However, when we take a look inside the zip file, we notice that the feed is very stripped down. The agency table contains only one record about the VGN transport association itself. The individual trips are thus not assigned to the individual transport operators of the VGN as it is the case with the DELFI data set which lists e.g. the VAG agency. Furthermore, the optional shape and fare information are missing.

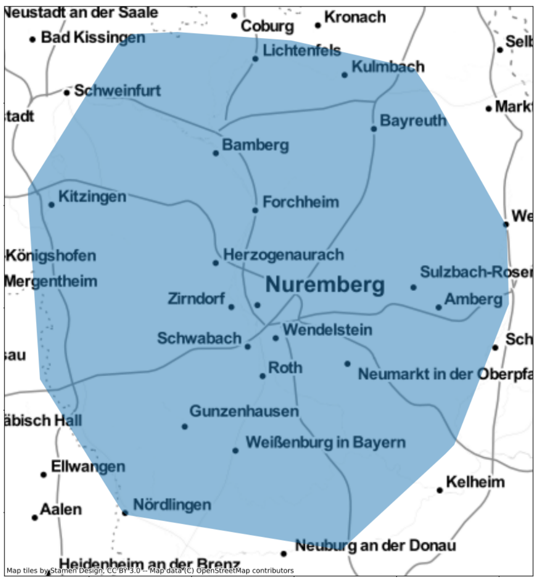


Figure 2.6: Convex hull around the stops locations of the VGN GTFS dataset. Map tiles by Stamen Design. Copyright of map data by OSM contributors.

Unfortunately, there are no supplementary real-time updates for the GTFS feed on the VGN's open data site although their own app includes real-time information which means that the data is available in their system. This would solve the problem of the infrequent updates and could also cover short-term schedule changes, delays and cancellations.

Similarly, there is no published GBFS feed by the VAG for their rental bike service either (VAG, n.d.-b).

It is also worth mentioning that the Deutsche Bahn AG (DB), a nationwide operating transit agency, offers a realtime application programming interface (API) with information about train departure and arrival times licensed under CC BY 4.0 (DB, n.d.). To use this interface, a free account is required and requests are limited to 60 per minute. This and the fact that a custom REST API is used, complicate the integration of the data source making it unsuitable for this project.

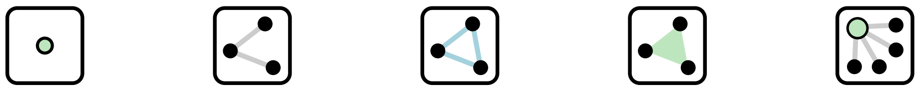
2.3.3 OpenStreetMap

OSM is an ODbL licensed, worldwide geographic database maintained by a community of mappers who are in many places organized in groups (OpenStreetMap, n.d.). The project is supported by the non-profit OpenStreetMap Foundation, which formally operates some of the services without intending to directly control it (OpenStreetMap Foundation, 2023).[8]

The conceptual data model consists of only three basic components: Nodes, ways, and relations (Verdy p et al., 2023). All of them can have multiple key-value pairs (tags) assigned.

- Nodes represent points on the surface of the earth and can be used to, among others, map the location of bike lockers, the entrance of a train station or a bus stop.
- Ways are a list of nodes that define a polyline. They can be used to describe linear features as *open way* (e.g. roads or rails), the boundary of areas as *closed way* or *areas* (e.g. the ground plan of a railway station building or a residential house). In the last two cases, they form a solid polygon.
- A relation describes the relationship between two or more basic components. It can be used, for example, to model cycle routes or large administrative boundaries (Stevea et al., 2022).

Figure 2.7 illustrates different forms of these basic components.



(a) Node **(b)** Open way **(c)** Closed way **(d)** Area **(e)** Relation

Figure 2.7: Icons of the basic components in OSM. (b), (c) and (d) represent different types of ways. The images are in the public domain.

Based on this model, a large variety of geographical aspects can be mapped up to a very fine level of detail. How much detail is actually available, however, varies greatly from region to region. In figure 2.8, we see the difference in quality between a map section from the city of Nuremberg and the rural village of Pfeifferhütte. Both images are at the same scale. The houses in the visible area of Pfeifferhütte do not yet have tags with their addresses (street name and house number) and therefore a geocoder can only estimate their location. Furthermore, the non-rectangular house gives an example that occasional mapping errors do occur. Figure 2.9 shows this house in the online map editor of OSM. There we

[8]Transparency note: Independent of this bachelor thesis, I am a voting member of the OpenStreetMap Foundation.

see that the upper right corner of the building was mistakenly combined with a fire hydrant. In addition, we can see from the underlying satellite image that the building is rectangular shaped in reality.

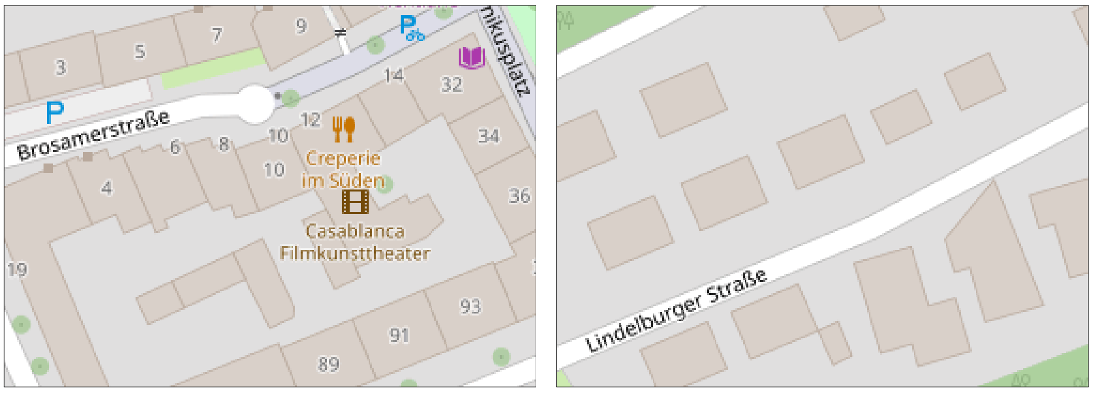


Figure 2.8: Two map sections with the default OSM map style. Adapted from https://osm.org/. Copyright of tiles by OSM contributors.

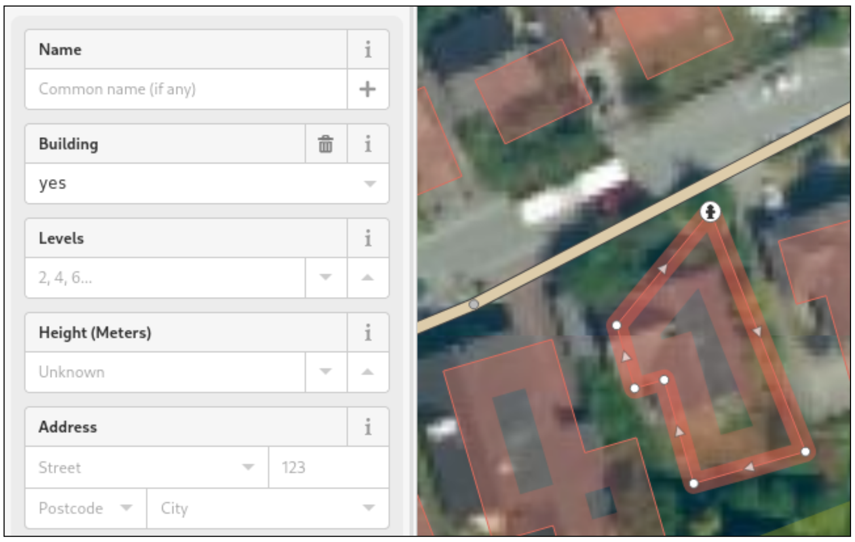


Figure 2.9: Part of the OSM online editor. Copyright of map data by OSM contributors. Copyright of satellite imagery by Bayerische Vermessungsverwaltung.

Of the many uses of OSM, we are most interested in the pedestrian and bicycle network. Even if there are minor errors and we can certainly find paved roads that are still missing in the dataset, it still has a very high density of road, bike and footpaths in the area of the VGN making it a god fit for our use case. In particular, the many signposted bicycle paths stored are helpful for bicycle-friendly navigation. Figure 2.10 shows these highlighted in color for the city of

Erlangen. In the background of the same image we also see the road network. All of this information comes from the OSM database.

We have therefore decided to use the daily updated OSM database extract for Germany provided by Geofabrik GmbH[9] which covers the entire area of the VGN.

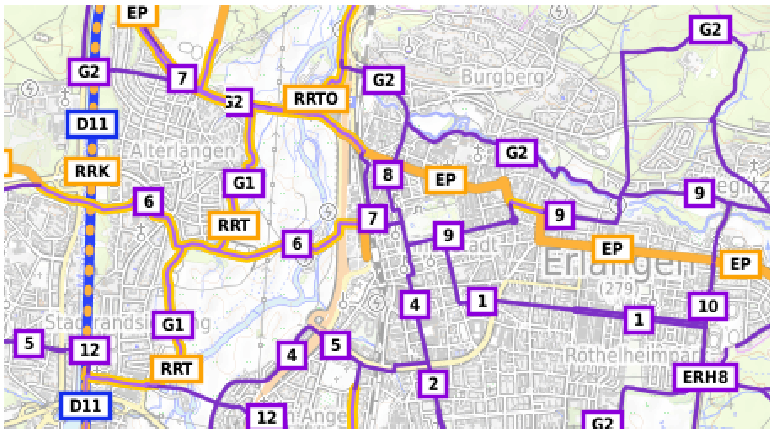


Figure 2.10: Sign-posted cycling routes overlayed on the OpenTopoMap. Adapted from https://cycling.waymarkedtrails.org/. Copyright of map data by OSM. Copyright of base map by OpenTopoMap with height data from SRTM.

2.3.4 Community Lists

Finding suitable data sources is not easy. For example, the German National Access Point misses many sources and while the Finnish one provides helpful visualizations of the geographic coverage of individual datasets, the short descriptions of them are not available in English making it difficult for non-native speakers. Furthermore, it requires individual research to discover open data offers from transport operators and other institutions such as e.g. the OpenData portal of the Bavarian State Office for Survey and Geoinformation which includes a CC BY 4.0 licensed shapefile covering all sign-posted bike paths in Bavaria.[10]

One potential remedy for this could be the European open data portal, which currently contains about 1.6 million data sets (Publications Office of the European Union, n.d.-a). But with the amount of results, the currently few filter options become a problem. For example, if we filter by the GTFS format, we get 174 results and can't narrow them down geographically, e.g. by country.[11]

[9]https://download.geofabrik.de/europe.html

[10]https://geodaten.bayern.de/opengeodata/OpenDataDetail.html?pn=bvv_radwege

[11]Interestingly, we found a wrongly categorized GTFS-RT feed of the transport association Rhein-Sieg in the GTFS category, which was not listed in the German National Access Point (Publications Office of the European Union, n.d.-b).

Another way to find data sources are the many lists from the open data and public transit communities. There, important aspects of about data sets are often highlighted, visually represented, or machine-readable.

The *European transport feeds* website provides a good overview of existing nationwide GTFS and NeTEx feeds by visualizing them on two maps of Europe and summarizes license information in a table. It does furthermore offer permanent links to the latest versions of the datasets (Tens et al., 2023) and if they are behind a login, they are self-hosted and made available for direct download.

From their website you also reach another community project: The GTFS-Publikationen is a tabular listing of all German GTFS feeds with their license, download link and reports (Holger Bruch et al., 2023). Whenever a GTFS feed changes, it is automatically checked in form and content and its report is updated. This table shows that the feed provided by the VGN contains 4089 errors and even more warnings. The high number is somewhat misleading, since if a faulty route is served at different times, it will appear several times in the data set. Nevertheless, we see that the quality of the VGN data set could still be improved.

Lastly, we would like to mention the project *list of transport APIs*, which contains a schema for describing such APIs and uses it to give details about many (also proprietary) interfaces in a machine-readable way (Jannis R et al., 2023). Among those are the HAFAS interface of the DB and the EFA interface of the VGN which are used internally by those companies and also by the Transportr app as presented in section 3.1.

These projects are maintained on a best effort basis and are by no means always up to date nor do they fully cover existing data sources. Nevertheless, it is often easier to find suitable data sources through some of these projects. The project about German GTFS feeds provides with 46 results in any case much more information than the German National Access Point which gave us just one result during our investigations in subsection 2.3.1.

2.3.5 Data of Demand-Responsive Services and Real-Time Updates

While the goal of this work is primarily to combine personal cycling with public transit, this would be very well complemented by a rental bike mode. We did, however, not find any suitable data sources for our target region. This may be because such sources are inherently realtime sources.[12] For example, to integrate

[12]In the German National Access Point, there is a car and bike sharing category listing 21 data sets. However, all of these cover only car rentals. And while OSM has tags for bike rental stations (Pieren et al., 2023), it is not intended for dynamic data such as the current availability

rental bicycles into a multimodal search, it is not enough to know the static locations of bike stations, but it requires additional knowledge about the current availability of bicycles at these stations.

Such real-time feeds as well as updates for timetable-based navigation are still rare in Germany. As a result, we were not able to integrate a rental bike mode or real-time updates (delays and cancellations) into our project, even though the selected software modules would have supported this out of the box.

of rental bikes.

3 Existing Software

In this chapter, we provide an overview of existing multimodal trip planners with a focus on support for personal bicycle modes. Even though Google Maps is not open source, it was still reviewed due to its widespread use.

3.1 Transportr and Navitia

Transportr is a community-developed open source Android application licensed under GNU AGPLv3 or later. The multimodal journey planner combines walking and schedule based public transportation (Transportr, n.d.-b). It supports different public transit networks worldwide whose coverages are visualized either by a marker or by red coloring on a map in Figure 3.1.

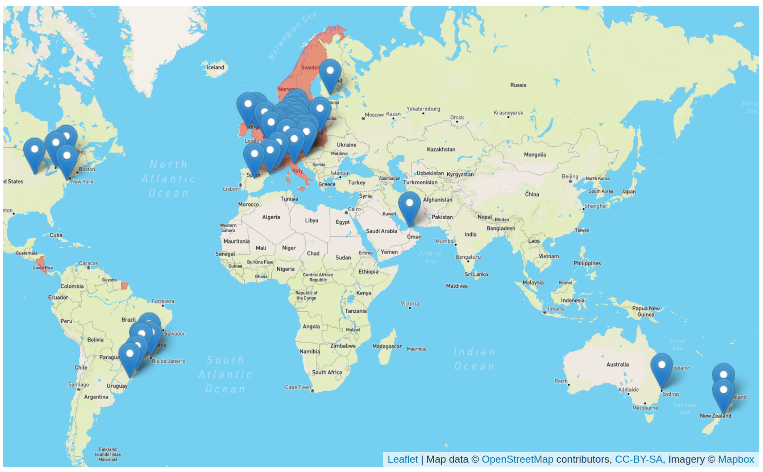

North Atlantic Ocean
South Atlantic Ocean
Indian Ocean
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © Mapbox

Figure 3.1: Overview of transport regions supported by Transportr. Adapted from Transportr. (n.d.-b). *Free public transport assistant without ads or tracking* [Transportr]. Retrieved April 12, 2023, from https://transportr.app/

Simultaneous navigation in multiple networks is not possible with Transportr, even if they are adjacent or overlap. This is due to the fact that depending on the transit network different web services are queried for possible journeys. For instance, the HAFAS interface of the DB is used for the nationwide DB network and the EFA interface of the VGN is used for the regional VGN network. (Schildbach, 2023a, 2023b; Transportr, n.d.-a)

The webservice Navitia is being developed by the French company Kisio Digital as open source software with the main repository being licensed under GNU AGPLv3 (Antoine D, 2014; COSSON, n.d.-a, n.d.-c). It supports queries for multimodal journeys by public transport, including private and rental bikes (COSSON, n.d.-b). Supported data sources are GTFS, NTFS (own extension of GTFS) for scheduled services as well as OSM for the road network (Simart et al., 2023). Real-time information can also be imported, but there is a lack of documentation about this. Kisio Digital runs an instance of Navitia and offers its API for a fee, with the first 5000 requests per month being free (COSSON, n.d.-d). While there is a Docker Compose file to run the service ourselves, it is currently considered unstable (Antoine D et al., 2023).

New transit areas with a public GTFS feed can be added to their instance upon request. This allows Transportr to support new regions without programming adapters for their different interfaces. Instead, the Navitia API has only been added once as a routing provider, and it now only takes a small change to add a new region after it was added to the Navitia API (Transportr, n.d.-a).

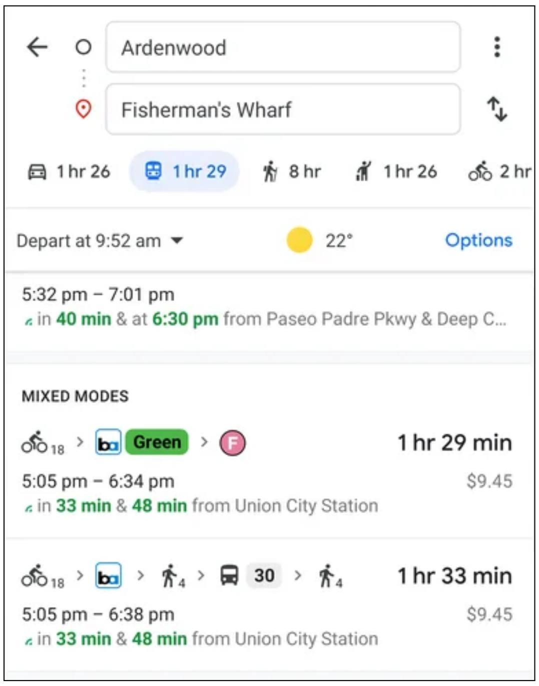
The option to use Navitia with GTFS and OSM data as backend and Transportr as frontend makes it possible to search for public connections using only free software based on open data. However, Transportr only displays trips with the two modes walking and public transit.

3.2 Google Maps

The Google Maps platform is the most widely used mapping software in terms of total Google Play application downloads (AndroidRank, n.d.) as well as the US charts of the Apple App Store for iPhones (Apple Inc., 2023).

Its features include multimodal navigation with biking, ridesharing and public transportation, as Dutta (2019) announced on the Google Blog in 2019. Figure 3.2 shows how the Google Maps mobile application suggests *Bike and Ride* connections whose first trip legs consist of cycling followed by a combination of public transportation and ridesharing.

However, if we look at the example depicted in Figure 3.3, we find that Google Maps is only of limited use for finding bike and transit connections in the VGN. When searching for trips from the university library in Erlangen to a sports center in Nuremberg, the application does not suggest any connections with bicycle legs even if we enabled the *Bicycle* mode in the *Connections to transit* preferences.



Ardenwood

Fisherman's Wharf

1 hr 26 | 1 hr 29 | 8 hr | 1 hr 26 | 2 hr

Depart at 9:52 am | 22° | Options

5:32 pm – 7:01 pm
in 40 min & at 6:30 pm from Paseo Padre Pkwy & Deep C...

MIXED MODES

18 > Green > F — 1 hr 29 min
5:05 pm – 6:34 pm — $9.45
in 33 min & 48 min from Union City Station

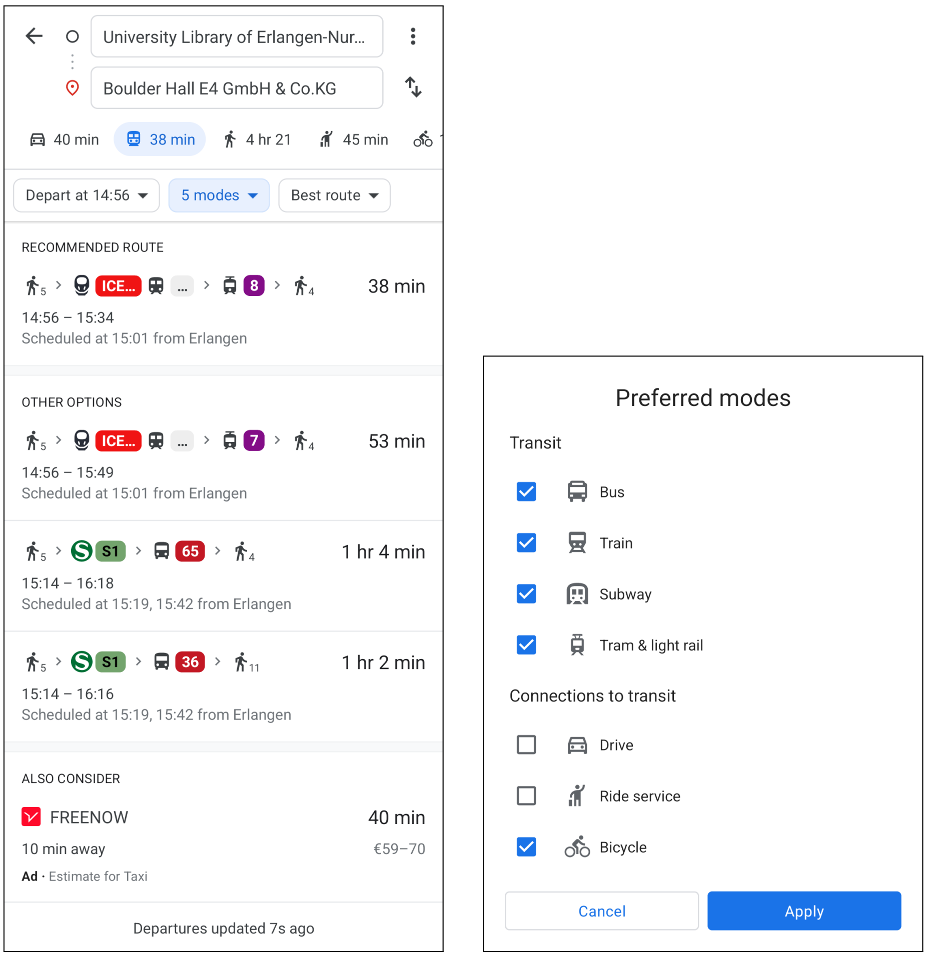
18 > > 4 > 30 > 4 — 1 hr 33 min
5:05 pm – 6:38 pm — $9.45
in 33 min & 48 min from Union City Station

Figure 3.2: The *transit* tab of Google Maps includes *Bike and Ride* connections. Adapted from Dutta, V. (2019, August 27). *Travel your first and last mile with google maps* [Google]. Retrieved April 13, 2023, from https://blog.google/products/maps/travel-your-first-and-last-mile-google-maps/

This is not because the first leg of the journey is a short 5-minute walk. When we placed the starting point further from the train station, all results still started with a walk, this time of 13 minutes. Furthermore, some results contain high-speed trains (ICE) that are not included in VGN tickets[1] and which cannot be disabled in the settings.

Finally, it should be noted that Google Maps is a closed platform. Outsiders cannot see which data the navigation is based on, and the use of the map services is only permitted within the scope of Google's license terms. As such it is prohibited to "create or augment any other mapping-related dataset" from Google Maps or to "use any part of Google Maps/Google Earth with other people's products or services for or in connection with real-time navigation" (Google, 2022a).

[1]Tickets for the long distance trains ICE, IC, EC can't be bought in the VGN shop (https://www.vgn.de/verbindungen/). But for some season tickets there are upgrades available at a different transport agency (https://www.vgn.de/tickets/db-zuschlag/#fernverkehr-ic-ec).



University Library of Erlangen-Nur...

Boulder Hall E4 GmbH & Co.KG

40 min | 38 min | 4 hr 21 | 45 min

Depart at 14:56 | 5 modes | Best route

RECOMMENDED ROUTE

5 > ICE... ... > 8 > 4 — 38 min
14:56 – 15:34
Scheduled at 15:01 from Erlangen

OTHER OPTIONS

5 > ICE... ... > 7 > 4 — 53 min
14:56 – 15:49
Scheduled at 15:01 from Erlangen

5 > S1 > 65 > 4 — 1 hr 4 min
15:14 – 16:18
Scheduled at 15:19, 15:42 from Erlangen

5 > S1 > 36 > 11 — 1 hr 2 min
15:14 – 16:16
Scheduled at 15:19, 15:42 from Erlangen

ALSO CONSIDER

FREENOW — 40 min
10 min away — €59–70
Ad · Estimate for Taxi

Departures updated 7s ago

Preferred modes

Transit

- ✓ Bus
- ✓ Train
- ✓ Subway
- ✓ Tram & light rail

Connections to transit

- ☐ Drive
- ☐ Ride service
- ✓ Bicycle

Cancel | Apply

Figure 3.3: A Google Maps query whose resulting journeys do not include bike trip legs (left) even though the bicycle option is active (right).

3.3 OpenTripPlanner

OpenTripPlanner is an open source multimodal trip planner licensed under LGPL v3.0 or later. It was initiated in 2009 by TriMet as a collaborative project between several other transit agencies and developers from major open source journey planners such as OneBusAway and Graphserver (Byrd & Gran, 2022). Among other things, this was done with the purpose of sharing the development costs of complex navigation software, fostering innovation and competition, as well as gaining wider adoption (TriMet, 2011).

The project has since become a member of the non-profit charity Software Freedom Conservancy, which supports the Project Leadership Committee by taking care of all matters outside of software development and documentation such as financial and legal concerns. This allows developers to focus on improving the software (Byrd & Gran, 2022; Software Freedom Conservancy, n.d.).

Since early stages of development, OpenTripPlanner supports “true multi-modal trips combining walking, biking and public transit” (TriMet, 2011, p. 12). Thereby not only *Bike and Ride* is supported, but also *Bike and Transit* (Welker, 2011).

Since the release of version 2.2 in 2022, the supported data sources include GTFS and NeTEx for schedule based public transit, GTFS-RT and SIRI “to apply realtime updates on top of schedule data” (Thomas Gran et al., 2022), GBFS for rental bikes and OSM to create pedestrian and bicycle road networks. Two years earlier, the routing algorithm used with scheduled transportation was replaced with a multi-criteria variant of the round-based RAPTOR algorithm wich improves performance and is better suited for dynamic real-time data as it does not require preprocessing (Byrd & Gran, 2022; Delling et al., 2012). The new algorithm returns all pareto optimal journeys, that are as such optimal in at least one of the considered criteria for a given time range. Possible criteria are, among others, shortest travel time, fewest transfers and a generalized cost function which can e.g. be used to consider ticket prices (Gran et al., 2023).

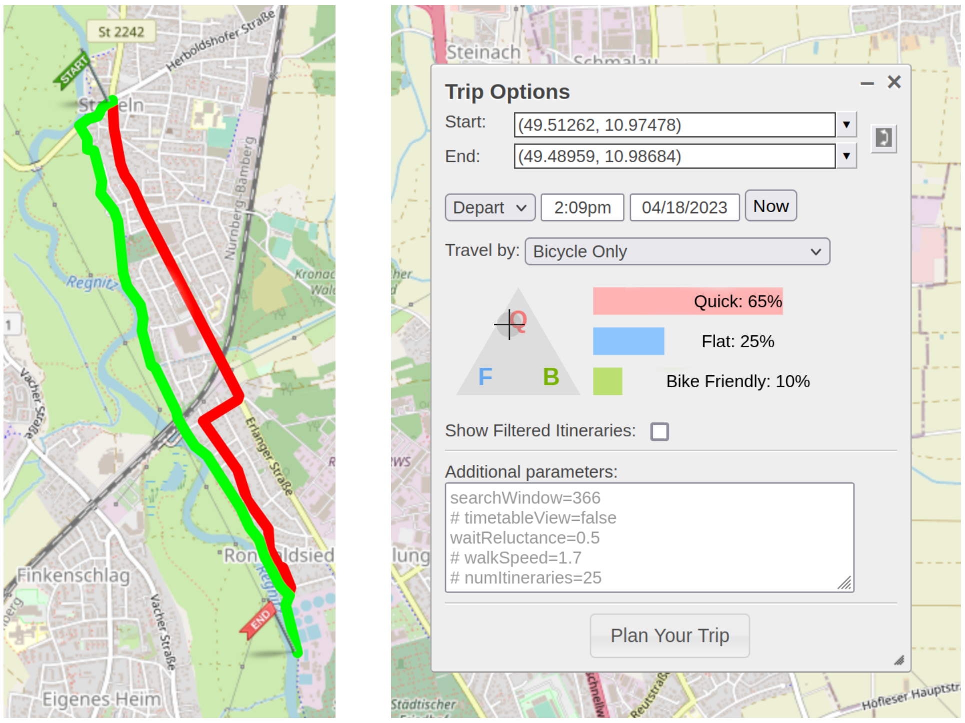
Routing of personal transportation modes is done with the A* algorithm (Gran, 2023). There are options for customization here as well. Figure 3.4 shows how a control triangle in the OpenTripPlanner web interface affects the returned bike routes. Choosing a high safety value yields the green route that runs along a bike path in the river valley. Whereas if more weight is placed on speed and flatness, one is guided along the red route following the main road instead.

OpenTripPlanner has an integrated web client, but the focus is on using it as a backend service. As such, it offers a RESTful and, via extension, a GraphQL API (Gran et al., 2022). The web interface, with its pop-ups visible in Figure 3.5, is not aimed at end users but rather offers the operators of an OpenTripPlanner server the opportunity to try out and adjust the configuration parameters for the target region.

Overall, with OpenTripPlanner there is an open source route planner with strong multimodal functionality that supports common open data standards and can be configured well with regard to cycling. Multiple transit agencies around the world use OpenTripPlanner as backend service in their productive environment in combination with custom frontends (Byrd et al., 2023). In the next section we look at one of these software solutions.

3.4 Digitransit

The Digitransit platform is developed by Helsinki Regional Transport Authority (HSL). It “combines open-source components into a route planning ser-



Trip Options

Start: (49.51262, 10.97478)

End: (49.48959, 10.98684)

Depart 2:09pm 04/18/2023 Now

Travel by: Bicycle Only

Quick: 65%

Flat: 25%

Bike Friendly: 10%

Show Filtered Itineraries: ☐

Additional parameters:

```
searchWindow=366
# timetableView=false
waitReluctance=0.5
# walkSpeed=1.7
# numItineraries=25
```

Plan Your Trip

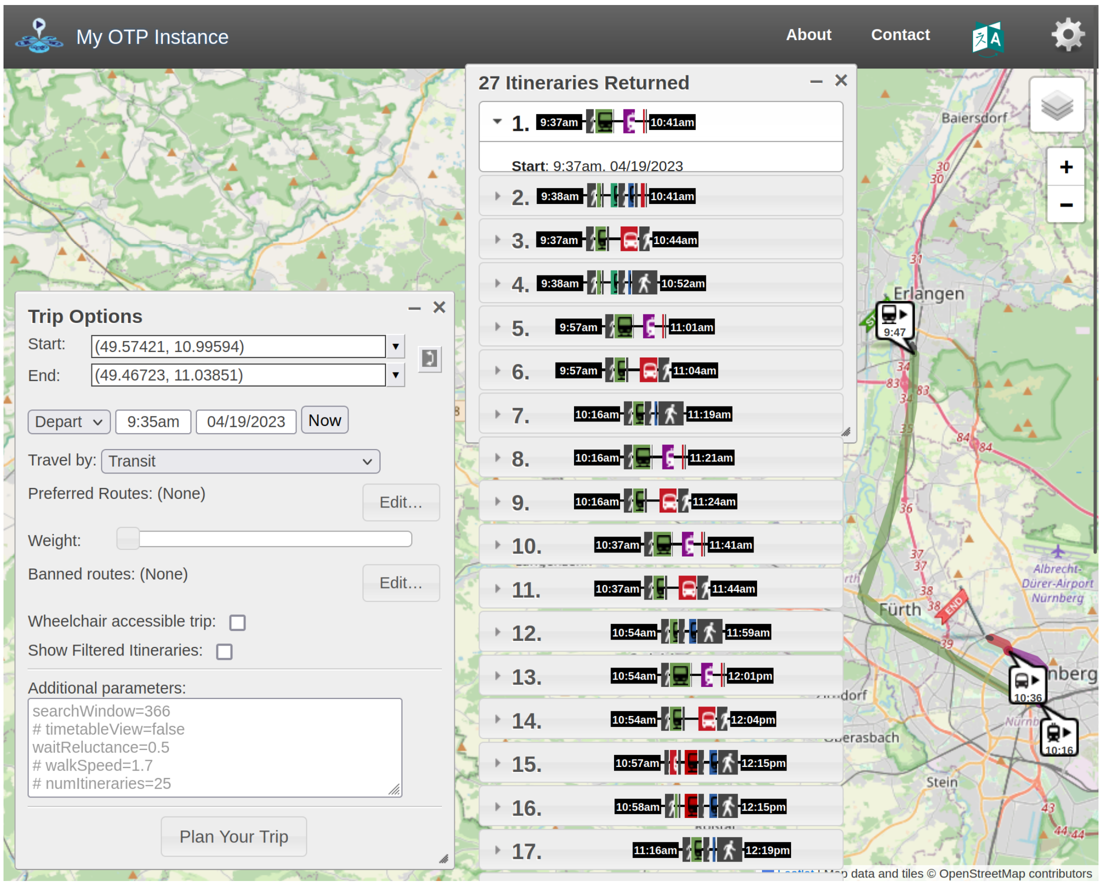
Figure 3.4: Two bicycle connections resulting from different search parameters in OpenTripPlanner. Copyright of map data and tiles by OSM contributors.

vice” (Malkki & Lappalainen, 2022). The source code of the entire project, including the build process as well as the resulting Docker images used in production, is public. But in some places, such as the code repository *OpenTripPlanner data container*, license information is missing (HSL, 2023).

The core components of the platform are OpenTripPlanner, Pelias, a map server and Digitransit UI. The geocoder Pelias is used for searching and resolving addresses and places into coordinates in order to pass them to OpenTripPlanner for navigation requests. Digitransit UI is a self-developed responsive web interface that offers multimodal navigation with private or rented bike, walking and public transit based on the APIs of the other two services. It is dual licensed under EUPL v1.2 and GNU AGPLv3.

In the settings, walking and cycling speeds can be adjusted and the different modes of transportation can be selected. Furthermore, additional information can be displayed on the map, such as public stops or bike rental locations with their current capacity.

Journeys with rental bikes are integrated into the normal results. On the other hand, trips that include legs with private bicycles are grouped separately. Figure 3.6 shows a search query with no stops in the immediate surroundings. The


My OTP Instance
About
Contact
Baiersdorf
Erlangen
Fürth
Stein
Albrecht-Dürer-Airport Nürnberg
9:47
10:36
10:16
Map data and tiles © OpenStreetMap contributors

Trip Options

Start: (49.57421, 10.99594)

End: (49.46723, 11.03851)

Depart 9:35am 04/19/2023 Now

Travel by: Transit

Preferred Routes: (None) Edit...

Weight:

Banned routes: (None) Edit...

Wheelchair accessible trip: ☐

Show Filtered Itineraries: ☐

Additional parameters:

```
searchWindow=366
# timetableView=false
waitReluctance=0.5
# walkSpeed=1.7
# numItineraries=25
```

Plan Your Trip

27 Itineraries Returned

1. 9:37am – 10:41am
   Start: 9:37am, 04/19/2023
2. 9:38am – 10:41am
3. 9:37am – 10:44am
4. 9:38am – 10:52am
5. 9:57am – 11:01am
6. 9:57am – 11:04am
7. 10:16am – 11:19am
8. 10:16am – 11:21am
9. 10:16am – 11:24am
10. 10:37am – 11:41am
11. 10:37am – 11:44am
12. 10:54am – 11:59am
13. 10:54am – 12:01pm
14. 10:54am – 12:04pm
15. 10:57am – 12:15pm
16. 10:58am – 12:15pm
17. 11:16am – 12:19pm

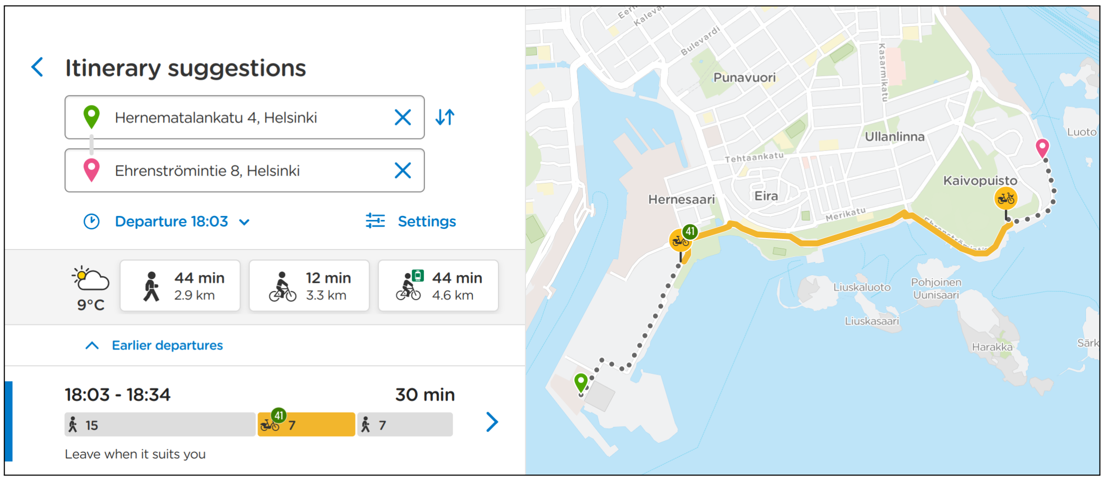
Figure 3.5: Search results in the web interface provided by OpenTripPlanner.

fastest connection by public transport takes 30 minutes, which includes renting a bike from a dock and returning it at another node. Cycling the entire route takes only 12 minutes. In this example, the *Bike and Ride* trips are not suitable. They are accessible via the bike icon with the green train, but take 44 minutes to reach the destination – the same amount as if one would walk.

Overall, Digitransit is an open source platform that uses OpenTripPlanner for multimodal routing. The web interface is user-friendly and supports multiple bike modes including *Bike and Ride* and *Bike and Transit*.

3.5 Stadtnavi and Bbnavi

stadtnavi is a model project of the city Herrenberg that was developed as part of the *Clean Air* program of the German Federal Ministry for Digital and Transport (Reith, n.d.). It is based on Digitransit UI which has been extended by several features, such as map layers for car and bike parking, charging stations, and bike repair locations.


Itinerary suggestions

Hernematalankatu 4, Helsinki

Ehrenströmintie 8, Helsinki

Departure 18:03

Settings

9°C | 44 min 2.9 km | 12 min 3.3 km | 44 min 4.6 km

Earlier departures

18:03 - 18:34 30 min

15 | 41 7 | 7

Leave when it suits you

Figure 3.6: Multimodal search results in Digitransit UI. Copyright of map data by OSM contributors.

As project leader Jana Zieger (2022) wrote in a post on Urban Digital at the end of 2022, the long-term plan is to create an operator model including participating municipalities and transit agencies. However, since federal funding ended in 2021 (Reith, n.d.), further development has been slow. There are rarely new commits in the code repository of the Digitransit UI fork, which is now 1500 changes behind (GitHub, Inc., 2023; stadtnavi, 2023).

Based on stadtnavi, there is the pilot project bbnavi in Brandenburg. It is currently funded by the DigitalAgentur Brandenburg GmbH (DigitalAgentur Brandenburg GmbH, n.d.-b). Whether it will later become a regular project and how it will be financed is still open (DigitalAgentur Brandenburg GmbH, n.d.-a). At present, it is being actively developed and many of the changes in Digitransit UI have been integrated into their fork (bbnavi, 2023).

Both projects show that there are efforts in Germany to offer open-source alternatives to the established proprietary journey planning solutions and to boost public transport through innovation. In doing so, the data situation is one of the difficulties the municipalities face. For example, bbnavi does not yet display real-time public transit information (DigitalAgentur Brandenburg GmbH, n.d.-b), even though this is supported by Digitransit UI and OpenTripPlanner. The reason for this is that real-time information from the Berlin-Brandenburg transport association is not published as open data. Instead, you can either use a proprietary HAFAS API or get access to a REST API on request (VBB Verkehrsverbund Berlin-Brandenburg GmbH, n.d.). But also in local matters, such as the allocation of charging stations, care must be taken by municipalities to ensure that these have open interfaces with data about their location and occupancy in order to visualize them on a map (RESET gemeinnützige Stiftungs-GmbH, 2023).

4 Requirements

This chapter presents the functional and non-functional requirements that shall be fulfilled by the multimodal journey planner created as part of this thesis.

4.1 Functional Requirements

To be able to refer to functional requirements, these are numbered and prefixed with “F”.

F1: Door-to-Door Journey Search

As a user that needs to visit a new place, I want to use an application to search for journeys by public transport to get there, so that I don’t have to manually shift through timetables.

The search is supposed to navigate me from the start address to the destination address, so I don’t have to manually select a start and destination stop.

F2: Bike and Ride

As a user who likes to bike, I prefer faster connections that start with a bicycle transfer to the next train station over slower connections which begin with a bus ride to the station, so that I spent less time commuting.

F3: Bike and Transit

As a user who can take their folding bike on public transport, I am searching for journeys whose first and last transfers are by bike, so that I can make use of the speed advantage of cycling over walking.

F4: Journey Overview

As a user who has searched for journeys by bike and public transit, I would like to have an overview where the most important information of each proposed journey is visible, so that I can identify suitable connections.

This should include at least the departure time, the duration of the journey and the sequence of vehicles.

F5: Journey Details

As a user that has chosen a journey, I need additional information about it, so that I can be at the right stop or platform on time for each trip leg.

F6: Bike Route Details

As a user who has selected a journey which includes bicycle routes unfamiliar to me, I need the directions of the individual bike routes, so that I can find my way.

To follow longer bike routes either my location should be visible on a map or there should be an option to share the bike route with a navigation app of my choice.

4.2 Non-Functional Requirements

To be able to refer to non-functional requirements, these are numbered and prefixed with "N".

N1: Data Sources

The application should use the chosen data sources as basis for multimodal navigation.

Necessary pre-processing steps of data from the data sources as well as their import into the application should be part of a data pipeline.

N2: Web Application

The user interface shall be realized as a web application. This allows the application to be used on most modern platforms in the browser of choice.

N3: Mobile Friendliness

The user interface should be designed in such a way that it is also possible to use it on mobile devices with narrow screens.

N4: Documentation and Code Style

The documentation should enable others to use and adapt the application.

It should be documented how the individual components and the application as a whole can be configured and executed. A uniform code style should be used within each component and identifiers such as variable and function names should be well-chosen to improve readability.

Test cases and examples can be part of the documentation.

N5: Testing

New functionalities should be tested using suitable test types. Especially for important aspects of the data pipeline, test cases should be created and automatically executed on changes.

N6: Configuration and Operation

It should be possible to configure the application for different geographic regions with other data sources (of the same format) and to run it on Linux.

5 Architecture

This chapter presents the technical structure of the developed journey planner, that was chosen to fulfill the requirements. First, the division into a web-frontend and several computationally intensive back-end services is motivated, and their responsibilities are thereafter explained. Next, the structure of the data pipeline necessary for importing data from the data sources is described. Finally, an example is given of how a public instance of the trip planner can be operated and how it can be scaled.

5.1 Underlying Architecture

The main architecture of the application results from the requirements as defined in the previous chapter.

Since the calculation of multimodal travel suggestions is computationally intensive and the necessary data from the data sources is quite large (3.8 GB for the German OSM extract (Geofabrik GmbH, n.d.)), an implementation as a pure client-side web application (N2) does not make sense in this case. Instead, a split into a client-side user interface that manages the application state coming from user interaction and several backend services for the compute- and memory-intensive tasks is appropriate. The front-end communicates with the other services via APIs and uses them to answer search queries and represent the results. The services include calculating multimodal travel options (F2, F3), address search (F1), and providing a map (F6).

For this purpose, a data pipeline is used to download raw data from OSM and the GTFS feed of the VGN, prepare it for the individual services and import it into them (N1).

5.2 Runtime Services

Figure 5.1 shows the data flow in the final application, which reflects the previously described division into a frontend, several backend services and a data pipeline.

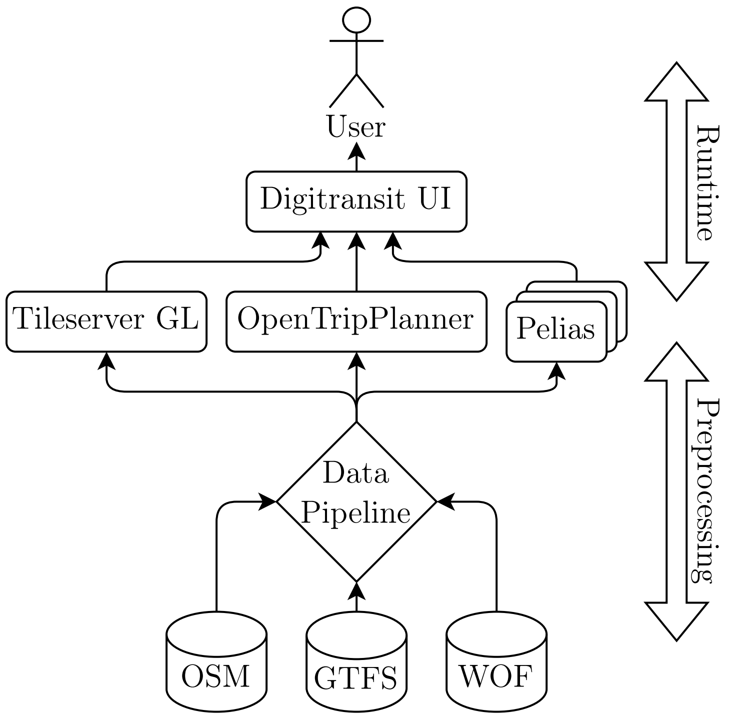


Figure 5.1: Flow of information from data sources until the user.

For the individual components, existing software from various open source projects was used. Which component is responsible for which part of the travel planner is explained in the following subsections.

5.2.1 Digitransit UI as Frontend

The web interface Digitransit UI is used as front end. It is a part of Digitransit platform, which HSL operates for various regions in Finland. The possibilities for searching multimodal travel routes with Digitransit have already been explained in more detail in section 3.4.

By reconfiguring Digitransit UI, most of the platform's services have been omitted, resulting in the interface requiring only three more services: OpenTripPlanner is used for route calculation, Pelias is used to answer address queries, and a raster and vector tile server is used for displaying the interactive map. For the latter, the map server tessera used by Digitransit was replaced with Tileserver GL.

5.2.2 OpenTripPlanner as Router

OpenTripPlanner is used by Digitransit UI via the GraphQL interface for routing requests between two points given as coordinates and for loading a transit stop map overlay. More details about the supported navigation modes of OpenTripPlanner have already been given in section 3.3.

5.2.3 Pelias as Geocoder

In the trip planner user interface there is a possibility to search for start and destination addresses. While entering these, possible addresses and stop names are suggested. Alternatively, a start or destination can be selected directly on the map, whereupon a nearby address is inserted into the corresponding input field.

All these three aspects are implemented with forward geocoding (converting an address to coordinates), backward geocoding (mapping nearby addresses to coordinates) and the address autocompletion of Pelias.

Figure 5.2 shows how the individual services of Pelias are integrated into the overall data flow of the journey planner.

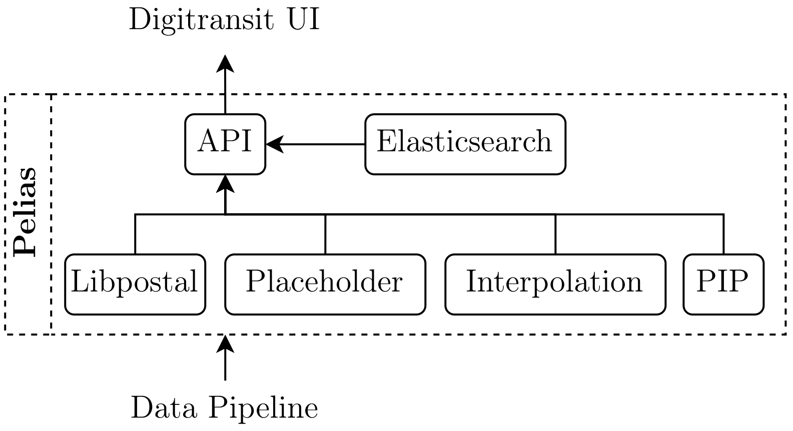


Figure 5.2: Microservices of Pelias.

The Geocoder is composed of several microservices in a modular way: The interface is provided by the API service, which answers queries using an Elasticsearch database and the four additional services Libpostal (parsing addresses), Placeholder (relationship between administrative areas), Point-in-Polygon (PIP; reverse geocoding related calculations) and Interpolation (estimating locations of addresses missing in the database; see Figure 5.3).

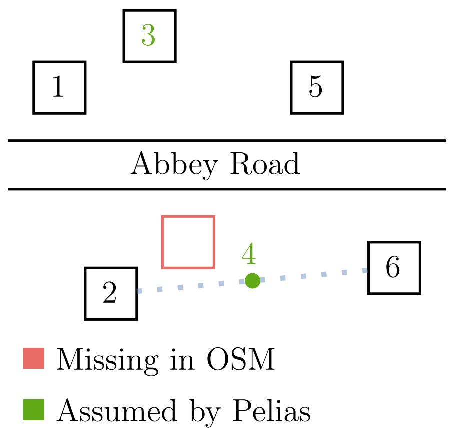


Figure 5.3: Visualization of address interpolation done by the Pelias Interpolation service.

5.2.4 Tileserver GL as Map Service

Because this map covers a large region, it is divided into a uniform grid of square tiles and only the tiles needed for the visible map section at the current zoom level are requested from Tileserver GL.

If the client supports WebGL, vector tiles and an accompanying map style are requested. These are then used to render the map section on the client side, which enables continuous zooming between the predefined zoom levels of the tiles. Alternatively, Digitransit UI requests the tiles as rendered images. These raster tiles can be displayed by the client with less computational effort.

5.3 Structure of the Data Pipeline

The services of the journey planner operate on the basis of timetable, address and road data. In order to prepare raw data from the data sources for the individual services, to convert it into a suitable data format and to import it into the services, a special process is required. Since this procedure is quite comprehensive, it has been divided into several steps, which together form the so-called data pipeline.

The steps of the pipeline can be grouped into three phases: First, current data is downloaded from the data sources, then it is processed, and finally, filesystem images are created for each service containing the application and required files from the pipeline.

The individual steps of the processing phase all follow the same scheme and can be supplemented or replaced by further steps and in some cases also omitted. The following types of pipeline steps can be distinguished, each mapping input files of one format to output files of the same or another format:

- Files a of one format $\mathbb{A}$ are converted to another format $\mathbb{B}$, or in other words files b are generated from files a.

  $f\colon \mathbb{A} \mapsto \mathbb{B}$ with $\mathbb{A} \neq \mathbb{B}$

- Files of one format are filtered and stay in the same format.

  $g\colon \mathbb{A} \mapsto \mathbb{A}$

- By combining files a with additional information b, they are extended or enhanced and stay in the same format $\mathbb{A}$.

  $h\colon (\mathbb{A}, \mathbb{B}) \mapsto \mathbb{A}$ with $\mathbb{A} \neq \mathbb{B}$

- Multiple files $(a_1, a_2, \dots)$ of the same format $\mathbb{A}$ are merged into one single file a.

  $i\colon \{(a_1, a_2, \dots) \mid a_i \in \mathbb{A}\} \mapsto \mathbb{A}$

Figure 5.4 shows the data pipeline as a whole. On the left it is visible that

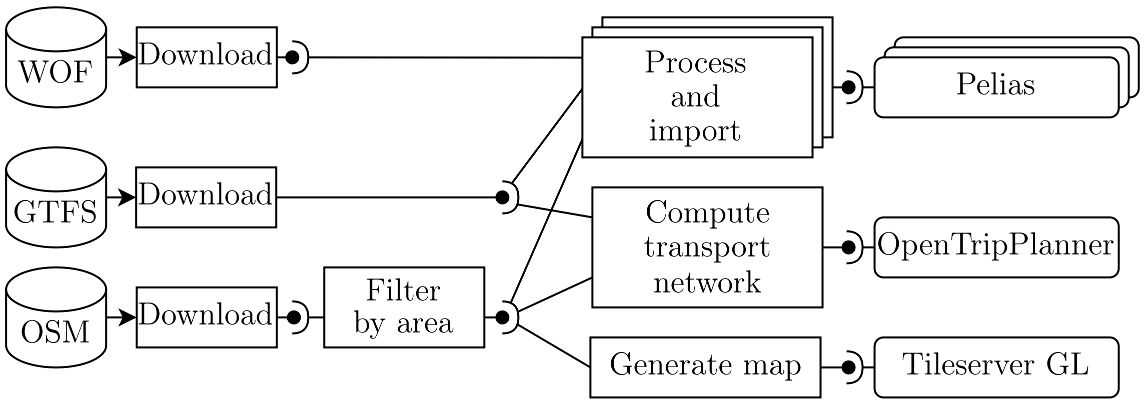


Figure 5.4: Key steps in the data pipeline.

in addition to the OSM extract for Germany and the GTFS feed of the VGN, raw data from the Who's On First project[1] is also downloaded. This is necessary for importing OSM data into Pelias, which is discussed in more detail in subsection 6.3.2.

Furthermore, it is shown that in a filter step the OSM data is trimmed down to the region of the VGN, which results in fewer computations to be performed in subsequent pipeline steps and smaller files generated for the three backend services.

[1]https://www.whosonfirst.org/

5.4 Reverse Proxy

The interfaces offered by the individual services are all based on unencrypted HTTP. To offer a public instance of the journey planner that can be reached securely over the Internet, a reverse proxy can be placed in front of the services of the back end, which accepts encrypted requests via HTTPS and forwards them to the right service of the journey planner based on their destination address.

Figure 5.5 shows how the trip planner can be operated on a server in this way. When a user accesses the website in their browser, the web application files are first transferred (1). Then, the interface is executed client-side and, based on the user's interaction, further requests are generated, such as loading vector tiles of the map (2) and reverse-geocoding coordinates after a user has selected a starting point on the map (3).

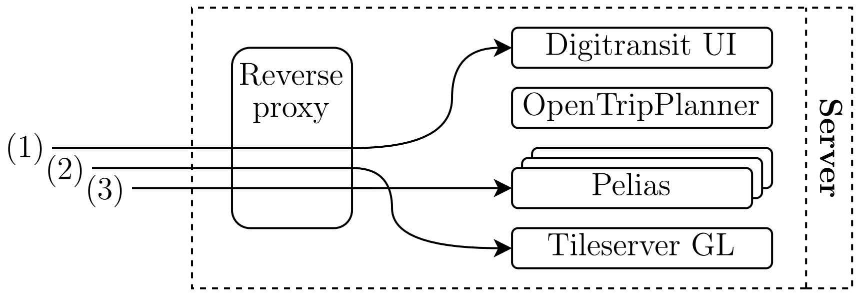


Figure 5.5: A reverse proxy forwarding requests to different services.

5.5 Scalability

Since the multimodal route calculations of OpenTripPlanner are computationally intensive and requests to the back-end increase in proportion to the number of users, it is worthwhile to consider the scalability of the trip planner.

The state of the web application is managed solely by the front-end, which runs in the user's browser. Without any adjustments to the individual services, this allows for vertical scaling – adding more processing power to the back-end server – as well as horizontal scaling: Multiple servers can be run and the load distributed between them.

At first, it is usually easiest to scale the backend vertically as a whole. When running on a virtual server with Google Cloud Run, for example, this can be done with few steps on the management interface.[2] Next, the individual services can be scaled out to dedicated servers, and as soon as adding more processing power

[2]https://cloud.google.com/run/docs/configuring/cpu#setting

to a service is no longer economical or possible, multiple instances of that service can be created and the requests split among them. In the example of Google Cloud Run, this becomes necessary once the maximum of 8 virtual CPU cores[3] is reached.

Figure 5.6 gives an example where two OpenTripPlanner instances are running on dedicated servers separately from the other services. It only requires configuration of the reverse proxy to split incoming routing requests between the two instances. Because the reverse proxy is still the central point for receiving requests from the front end, such a restructuring is completely transparent to the front end.

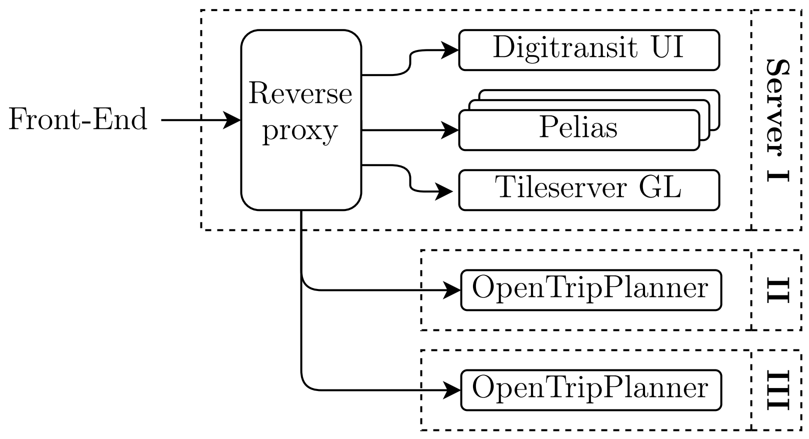


Figure 5.6: Possible setup with load balanced OpenTripPlanner.

[3]https://cloud.google.com/run/quotas

6 Design and Implementation

This chapter presents important aspects of the implementation of the journey planner and the technologies used. It also explains the difficulties that arose during the configuration of Digitransit UI and the additional handling of the data sources that was necessary due to technical shortcomings of the latter.

6.1 Codebase

The source code of the journey planner is distributed across several Git repositories, which are publicly available on GitHub.

Since this project is largely powered by the routing functionality of OpenTripPlanner while placing a focus on bike modes, it has been named BikeTripPlanner and the associated demo instance has been released at https://biketripplanner.de. The main repository[1] contains the files necessary to run the data pipeline and launch the application.

The interface was initially customized by forking stadtnavi/digitransit-UI. In the further course, however, a switch was made to the more recent version of the Digitransit platform itself, on whose v3 branch the final result[2] is based. Furthermore, to create a tileset for the map, a bicycle-specific map style was forked[3] and a Python script[4] was created for a pipeline step that makes corrections to the GTFS feed.

6.2 Containerization

The services of the BikeTripPlanner and the tools used in the data pipeline are written in different programming languages and require different runtime environments. To simplify the installation of the individual software components, they

[1]https://github.com/langbein-daniel/BikeTripPlanner

[2]https://github.com/langbein-daniel/digitransit-ui

[3]https://github.com/langbein-daniel/cyclo-bright-gl-style

[4]https://github.com/langbein-daniel/gtfs-modifications

were containerized with executable images that contain the programs with all the dependencies they require. With the help of the Docker Engine[5] either the official Docker images of the services are used for this purpose or, in the case of Digitransit UI, they are created from a customized Dockerfile.

Containerization allows, for example, some of the Pelias services to be based on Node.js 12[6] while Digitransit UI will soon be updated to Node.js 14[7] or 16[8], regardless of which version is installed on the host system. Furthermore, it brings isolation of the services from each other, which need to be explicitly connected in order to communicate with each other, e.g. via shared volumes or by attaching them to common networks.

To manage image creation and container startup, Docker Compose[9] is used. This allows defining the containerized services in a Compose file (`docker-compose.yml`) including the steps necessary to build the associated container images. Thereby variables from an additional `.env` file are taken into account, which forms the basis for the central configuration of the BikeTripPlanner and is described in more detail in section 6.6.

Due to the service and variable definitions, controlling the journey planner is possible with fewer and shorter commands. To start all of its services, for example, only the command

```
sudo docker compose up -d --wait
```

is needed.

The data pipeline utilizes containerization as well, as described in the following section.

6.3 Realization of the Data Pipeline

The download of raw data from the data sources, the individual processing steps and the creation of data images, each containing the application of a service with required files, has been implemented with several Dockerfiles. These contain instructions to create a new image from one or more existing Docker images and are referenced in the Compose file.

[5]https://docs.docker.com/engine/

[6]https://github.com/pelias/docker-baseimage/blob/ea19fa55219782ca6c5f2e638ce5fea5f7feab79/Dockerfile#L32

[7]https://github.com/HSLdevcom/digitransit-ui/pull/4545/files#diff-dd2c0eb6ea5cfc6c4bd4eac30934e2d5746747af48fef6da689e85b752f39557

[8]https://github.com/HSLdevcom/digitransit-ui/pull/4776/files#diff-dd2c0eb6ea5cfc6c4bd4eac30934e2d5746747af48fef6da689e85b752f39557

[9]https://docs.docker.com/compose/

In the download phase of the pipeline, the Dockerfile command

```
ADD <URL> <DESTINATION_PATH>
```

downloads raw data and stores it in a container image. The processing steps of the pipeline are implemented as multi-stage Docker image builds that use input files from the previous pipeline steps (Docker build stages) with the command

```
COPY --FROM=<IMAGE> <SOURCE_PATH> <DESTINATION_PATH>
```

and whose output files are stored in the resulting Docker images. Last, the files required by a service are added to the application image, again using the `COPY` command. In some cases, as for example with OpenTripPlanner, this last step was integrated into the last processing step specific to a service as this way a short Dockerfile with only one `COPY` command can be omitted.

The advantages of this approach of implementing the data pipeline by building multiple Docker images is explained in the following subsection based on the generation of the bicycle friendly map. This is followed by an explanation of why it was necessary to deviate from a pure Docker multi-stage build in the pipeline step for the Pelias geocoder. Finally, the realized data pipeline is shown with its extension possibilities.

6.3.1 Map Generation

Digitransit UI uses map tiles according to the Mapbox Vector Tile Specification[10] and renders them to images with a Mapbox style[11]. The generation of a vector tileset and the map style files consists of several intermediate steps. For this purpose, a fork of the bicycle-specific *OSM Cyclo Bright* map style was created and supplemented with a Makefile, allowing it to be used independently of this project as well.

The vector tiles are created with Tilemaker[12] from the OSM map section of the VGN. Tilemaker is included in the Debian 12 and Debian Unstable releases, but it is not yet available in the Debian Docker image (*debian:stable-slim*), which is still based on Debian 11. Furthermore, the program spritezero used to render the map style icons is not available in any Debian release.

To generate the map on a Debian system without containerization, or based on the Debian Docker image, a compilation of both tools would be necessary, which brings additional maintenance effort. By implementing this pipeline step with multiple stages in one Dockerfile, this is avoided: First, the icons are rendered

[10]https://github.com/mapbox/vector-tile-spec
[11]https://docs.mapbox.com/mapbox-gl-js/style-spec/
[12]https://packages.debian.org/sid/tilemaker

based on the openmaptiles-tools Docker image[13] where spritezero is pre-installed. Then Tilemaker is compiled with the Dockerfile included in its source code[14] and the ready rendered icons are copied from the previous build stage with `COPY --from=sprites /build build/sprites`. In the following execution of the Makefile to generate the tileset, the presence of spritezero is no longer necessary.

The ability to run different tools in different Docker images as used in this pipeline step shows the advantages of the containerized data pipeline. Running it natively on the host system would require a more steps to install dependencies and then compile both tools. In addition, Docker caches the individual build steps, which means that when the pipeline is run again, e.g. after an update to the OSM region, the static icons are not rendered again.

6.3.2 Pelias Data Import

As we showed in subsection 2.3.3, some buildings are missing in OSM or are not provided with complete addresses including house numbers. To solve this problem Pelias offers the possibility to add OpenAddresses as a collection of authoritative address data. However, in contrast to the neighboring state of Thuringia, there is no official data set for Bavaria available in OpenAddresses, which is why we did not include this data source in the pipeline.[15]

Apart from OpenAddresses, the data source Who's On First is a prerequisite for data import into Pelias. It is used to complete geographic hierarchies between places. Such administrative information as the associated locality or zip code of places is often missing in OSM.

To import addresses from OSM augmented with Who's On First into the geocoder, Pelias has a script that uses Docker Compose to start an Elasticsearch database container and then runs commands in the microservices containers and in special import containers to download, process and import the data.

We did not find a way for commands executed while building a Docker image to access another locally running container (here: Elasticsearch). Therefore, this pipeline step deviates from a Docker multi-stage build. Instead, analogous to Pelias' helper script, an Elasticsearch database container is started first and then only the download and import commands of the script are executed in different containers, which are responsible for importing from OSM and Who's On First. By mounting a folder in the containers, they operate on shared data and the output files are stored on the host system. In doing so, we avoided using the help

[13]https://github.com/openmaptiles/openmaptiles-tools#docker-images

[14]https://github.com/systemed/tilemaker/blob/master/Dockerfile

[15]https://github.com/openaddresses/openaddresses/blob/master/sources/de/th/statewide.json

scripter directly because, according to the associated README, it needs to be installed on the Docker host[16] and for Digitransit UI, an additional import step for stop names from the GTFS dataset[17] was required anyway.

The possibility to implement individual pipeline steps also in another way shows how flexible the data pipeline is. It is sufficient that one or more output files result after a step, which can be copied back into an image with the `ADD` command of a Dockerfile to be used as input by further pipeline steps.

However, the alternative approach chosen for Pelias has the disadvantages that the individual steps are not cached and are therefore executed again on each pipeline run. Also, this pipeline step consists of 32 shell commands vs. a single

```
sudo docker compose build <NAME>
```

command.

6.3.3 Extension Possibilities

Due to the alternative approach of the Pelias data import, which prepares data for all microservices of the geocoder in a single pipeline step, as well as necessary corrections of the GTFS feed, a representation slightly differing from the architectural perspective results when considering the realized pipeline steps. This is illustrated with further steps that can be added to the modular pipeline in Figure 6.1.

For the data of the VGN it was necessary to add the GTFS-modifications step, because some values in the CSV tables of the GTFS feed contain unescaped special characters ("), which lead to errors during the data import into Pelias. The Python script of the GTFS-modifications step corrects them automatically, but the fact that such technically simple format errors exist surprised us.

In the same step, the field bikes_allowed is added and set to true since it is not included in the VGN dataset. This field marks for the individual vehicles whether carrying a bike is allowed and is taken into account by OpenTripPlanner during routing, which has the consequence that no *Bike and Transit* connections are suggested if this field is missing.

If these modifications are not needed when using other data sources, the step as a whole can be omitted, because its input format is the same as the output file format.

Furthermore, it can be seen on the diagram that an input option is not used so far when generating the navigation graph of OpenTripPlanner: By using an

[16]https://github.com/pelias/docker#installing-the-pelias-helper-script
[17]https://github.com/HSLdevcom/pelias-gtfs

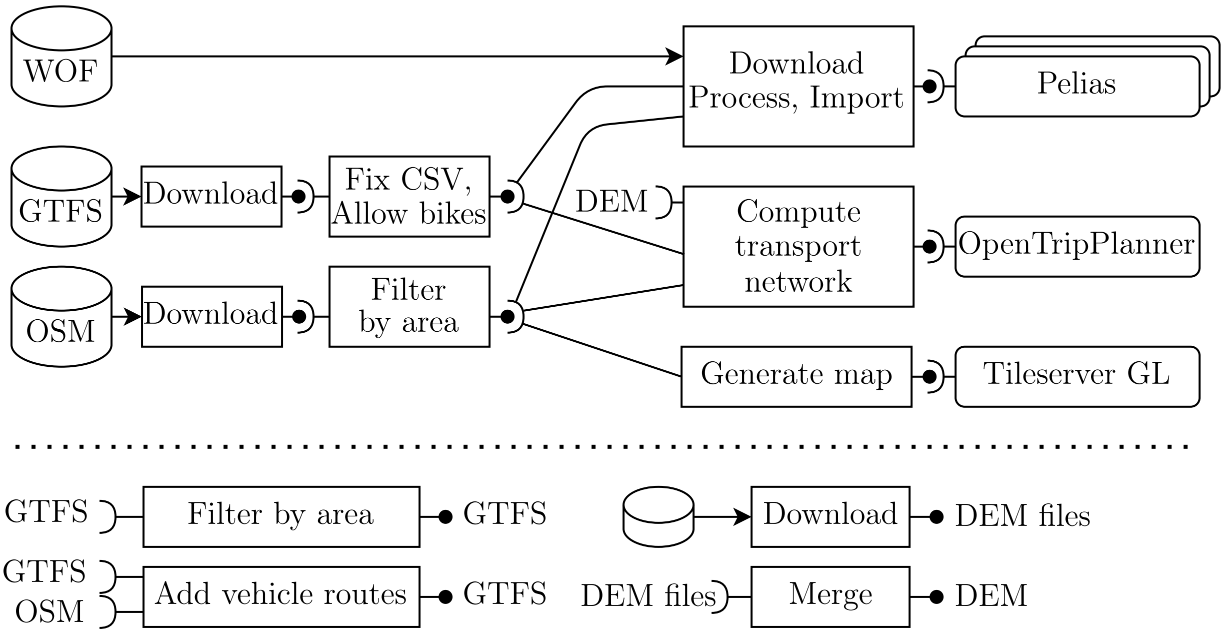


Figure 6.1: Implementation of the data pipeline (top) with possible extensions (bottom).

optional GeoTIFF file that contains elevation profile data, the slope in the terrain can be taken into account when generating the graph. With available elevation information, OpenTripPlanner can calculate more accurate travel times and avoid climbs for foot and bike paths. Such digital elevation models are available as open data e.g. from SRTM[18] or from the European Space Agency (Copernicus DEM[19]).

In the lower half of the figure extension possibilities are illustrated. Besides downloading and merging individual tiles of a DEM, two further modification steps for the GTFS feed are shown there: By filtering by the area of the VGN, the dataset can be reduced and by adding detailed routes of individual vehicles, their trip legs can be better visualized in Digitransit UI.

Such route shapes can also be part of a GTFS feed in the first place, but are missing in the VGN dataset. Since they are usually well mapped in OSM, there are special tools that add this information from OSM to a GTFS dataset. How big the visual difference can be is shown in Figure 6.2, which compares the displayed route of a bus line in BikeTripPlanner with the OSM based ÖPNVKarte.

[18]https://www.earthdata.nasa.gov/sensors/srtm
[19]https://spacedata.copernicus.eu/collections/copernicus-digital-elevation-model

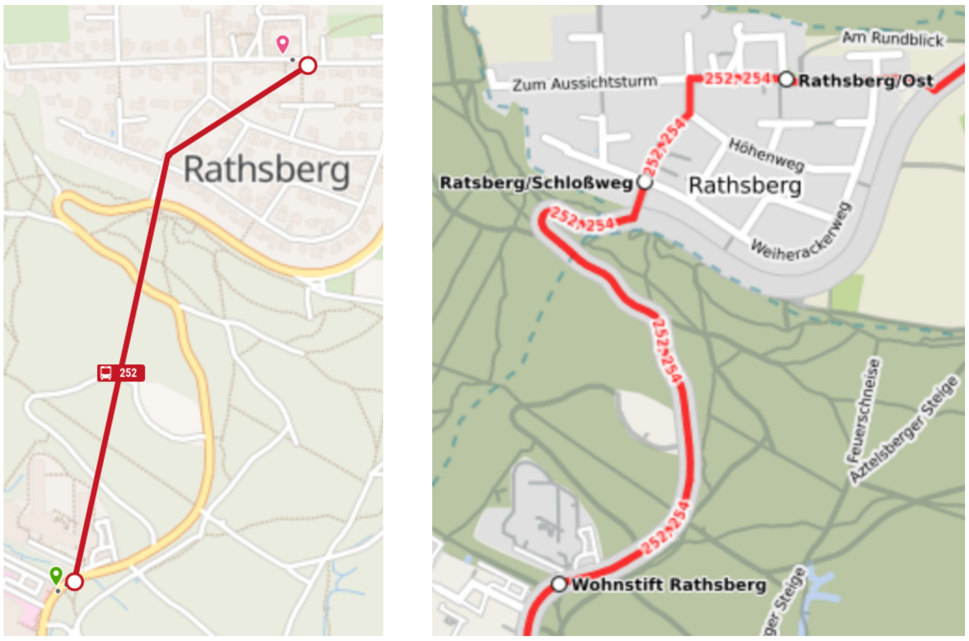


Figure 6.2: Visualization of a bus route in BikeTripPlanner (left) and on ÖPNVKarte (right). Copyright of ÖPNVKarte map tiles by MeMoMaps. Copyright of map data by OSM contributors.

6.4 Container Healthchecks

With Docker's HEALTHCHECK[20] command, it is possible to store a command to be executed regularly in containers, whose return value and execution duration gives Docker information about the state of the container.

Healtchecks were applied to all BikeTripPlanner services so that Docker can detect overload or failure of them. In addition, a healthcheck was needed for the Elasticsearch database so that when Docker Compose starts the geocoder, it waits to create the Pelias API service until Elasticsearch is not only created and started, but also responding to queries. Otherwise, the Pelias API service queries the database for existing layers at startup and if it is not yet accessible, it does not learn about the layer with the stop information from the GTFS import.

The healthchecks are designed to be a bit more extensive than strictly necessary for checking the state of a service, as it would be possible for OpenTripPlanner e.g. with a simple call of the `/health` endpoint[21]. Instead, in the case of OpenTripPlanner, a GraphQL query is made every 15 seconds for the names of all transit agencies from the GTFS dataset and it is asserted that the VGN is included in the result. In this way, the healtchchecks additionally serve as E2E

[20]https://docs.docker.com/engine/reference/builder/#healthcheck
[21]https://docs.opentripplanner.org/en/dev-2.x/sandbox/ActuatorAPI/

tests of the live system, detecting gross errors originating from data import (or a too early start of Pelias API).

6.5 Digitransit UI Customizations

Digitransit UI is the central component of the Finnish Digitransit platform. In order to use the web interface for BikeTripPlanner, it was reconfigured for a different region with different map, geocoder and routing services and some of its options were disabled. This section gives an overview of the configuration possibilities of Digitransit UI and describes how adjustments were difficult to make due to lack of documentation.

Digitransit UI is an npm package.[22] In package.json various scripts are defined for developing, building, testing and starting the web interface productively. How to add a new configuration to the source code with the add-theme script is explained in the `docs` folder[23] of the project. Using it, a configuration abbreviated with btp was created, which at its core consists of the two files `app/configurations/btp.js` and `sass/themes/btp/_theme.scss`.

Further instructions on how to adjust values in these configuration files are not provided.

6.5.1 The Configuration Object

The JavaScript configuration file exports an object, with properties about:

- Localization (languages and time zone)
- The geographical area
- The base URLs of required services
- Possible transport modes and which of them are active by default
- Menu subitems with information about the service itself (link to source code, listing of data sources, privacy policy, ...)

The properties of the configuration object override the predefined values of the `config.default.js` file, which contains almost no explanatory comments. This leads to the need to infer the meaning of configuration options just by the names of the properties.

[22]https://docs.npmjs.com/about-packages-and-modules

[23]https://github.com/HSLdevcom/digitransit-ui/blob/68e0ca0ccd2eb57f0258496ede8b88bf07d581c3/docs/Themes.md

6.5.2 The Property OTPTimeout

An example where this is almost sufficient is the property OTPTimeout. This has as default value the number 12000 and we assumed that it is the timeout for routing requests to OpenTripPlanner in milliseconds. To confirm the actual unit used, the source code had to be searched for usage of the property. Either a short comment or a more precise naming of the property, e.g. to OTPTimeoutMs, would be necessary here to avoid possible misunderstandings and subsequent programming errors.

The default value of this property is defined by the following program lines:

```
const OTP_TIMEOUT = process.env.OTP_TIMEOUT || 12000;
export default {
  OTPTimeout: OTP_TIMEOUT
  // ...
};
```

Attempts to pass a higher timeout via the OTP_TIMEOUT environment variable when starting Digitransit UI resulted in errors. Upon subsequent review of the source code, it was determined that this is caused by a programming error: The function call `process.env.OTP_TIMEOUT` returns the value of the environment variable as a string, but the OTPTimeout property is reused elsewhere as number. Therefore, it is necessary to parse the read value as number first. This has been fixed in the BikeTripPlanner's configuration and a patch for this bug will be provided to the developers of Digitransit soon.

6.5.3 The Property FeedIds

The most difficult to adapt was the property feedIds, whose value in the existing configuration files is usually set to the name of the respective configuration (e.g. `['OULU']` for the region around the Finnish city of Oulu).

In the btp configuration the value is set to `['1']`. This causes Digitransit UI to request from OpenTripPlanner the stop map overlay of the transit feed with ID 1. Correspondingly, OpenTripPlanner is configured to set the ID of the imported GTFS feed to 1. If the two values do not match, no stops can be selected on the map and no transit lines can be displayed. Furthermore, when searching for stop names with Pelias, Digitransit UI filters for results originating from the `gtfs1` data source. In order to have such results, the `--prefix 1` option must be used beforehand when importing GTFS data into Pelias because the data source listed in Pelias is otherwise just named `gtfs`.

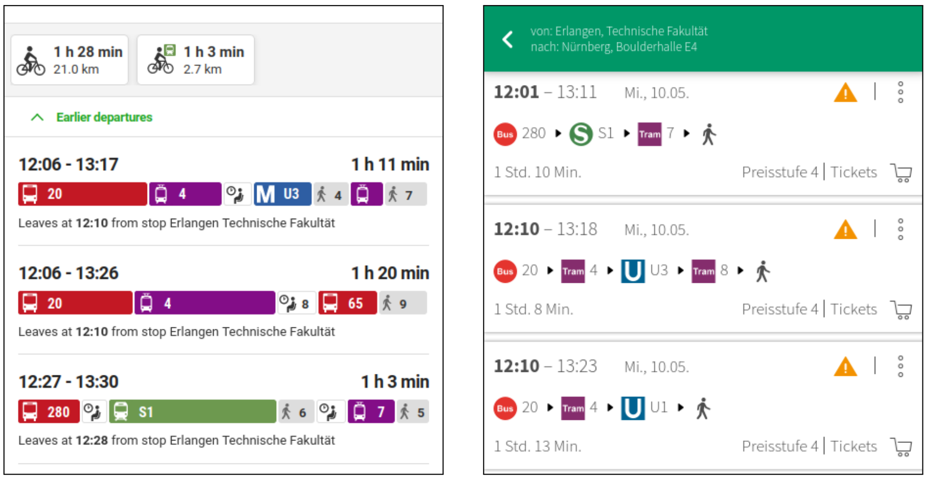
Filtering OpenTripPlanner and Pelias by transit feeds is not required by this project which has only one Digitransit UI instance. However, since there are multiple local instances running in the Digitransit Platform, such a distinction makes it possible to use common OpenTripPlanner and Pelias services with different frontends.

To make our Digitransit UI configuration usable for regions other than the VGN without any source code modification and recompilation, we read several values from environment variables at application startup. How this is implemented in detail is described in the later subsection 6.6.1.

6.5.4 Theming

In order to support users of the journey planner in recognizing the different types of vehicles, the Digitransit UI theme was adapted to the colors contained in the VGN GTFS feed. This was done by setting the colors property in the configuration object and adjusting the color values of the style sheet in `sass/themes/btp/_theme.scss`.

This causes that in the itinerary overview and on the map e.g. buses are shown in the red and commuter trains in the green tone, which is also used by signage at public transport stops. Figure 6.3 shows the result in comparison with the web version of the VGN journey planner.


1 h 28 min
21.0 km
1 h 3 min
2.7 km
Earlier departures
12:06 - 13:17 1 h 11 min
20 4 U3 4 7
Leaves at 12:10 from stop Erlangen Technische Fakultät
12:06 - 13:26 1 h 20 min
20 4 8 65 9
Leaves at 12:10 from stop Erlangen Technische Fakultät
12:27 - 13:30 1 h 3 min
280 S1 6 7 5
Leaves at 12:28 from stop Erlangen Technische Fakultät

von: Erlangen, Technische Fakultät
nach: Nürnberg, Boulderhalle E4
12:01 – 13:11 Mi., 10.05.
Bus 280 S1 Tram 7
1 Std. 10 Min. Preisstufe 4 | Tickets
12:10 – 13:18 Mi., 10.05.
Bus 20 Tram 4 U3 Tram 8
1 Std. 8 Min. Preisstufe 4 | Tickets
12:10 – 13:23 Mi., 10.05.
Bus 20 Tram 4 U1
1 Std. 13 Min. Preisstufe 4 | Tickets

Figure 6.3: Comparison between a themed Digitransit UI (left) and the VGN journey planner (right).

6.6 Central Configuration

It takes substantial effort to adapt the Digitransit Platform with its various services as well as the tooling to run the data pipeline and build the containers to a region outside Finland, because the individual components are configured in different ways and it is necessary to make changes to the source code in multiple of the associated Git repositories.

The BikeTripPlanners uses only two components developed by Digitransit Platform itself, Digitransit UI and the Pelias GTFS Importer, so the configuration scope is smaller. But as shown in the previous section, a configuration is still not easy.

Therefore, in the process of customizing to the VGN region, care was taken to provide good documentation of the parameters used and to bundle important aspects of the configuration. For this, a top-level `.env` file is used whose variable definitions are taken into account by Docker Compose during the data pipeline, for creating the application and data images, and for starting the services. This allows that only by modifying one configuration file, the trip planner and all its components can be used with other data sources (of the same format) and in a different geographic region.

6.6.1 Bounding Box

One configuration option that affects several components of the journey planner is the bounding box defined in the `.env` file. It consists of four coordinates describing the vertices of a rectangular geographical area and encloses the area of the VGN with the following values:

```
MIN_LON=10.011636032586688
MAX_LON=12.223993889052613
MIN_LAT=48.70792025947608
MAX_LAT=50.25793688217101
```

Digitransit uses the bounding box, among other things, to set the center of the VGN as the initial cutout for the map. To get the information there, the variables in the Compose file are passed as arguments to the Dockerfile used for building the Digitransit UI Docker image:

```
version: '3.9'
services:
  digitransit-ui:
    build:
      # Git repository containing Digitransit UI Dockerfile
      context: ${DOCKERFILE_DIGITRANSIT_UI}
      args:
        MIN_LON: ${MIN_LON}
        MAX_LON: ${MAX_LON}
        MIN_LAT: ${MIN_LAT}
        MAX_LAT: ${MAX_LAT}
```

Subsequently, four environment variables are set to the values of the bounding box arguments in the Dockerfile of Digitransit UI:

```
ARG MIN_LON
ARG MAX_LON
ARG MIN_LAT
ARG MAX_LAT
ENV MIN_LON=${MIN_LON}
ENV MAX_LON=${MAX_LON}
ENV MIN_LAT=${MIN_LAT}
ENV MAX_LAT=${MAX_LAT}
```

The environment variables set in the Dockerfile exist in containers created of the resulting Docker image and can be read by the application at runtime. In the btp JavaScript configuration of Digitransit UI, this is done by the following commands:

```
const MIN_LON = parseFloat(process.env.MIN_LON);
const MAX_LON = parseFloat(process.env.MAX_LON);
const MIN_LAT = parseFloat(process.env.MIN_LAT);
const MAX_LAT = parseFloat(process.env.MAX_LAT);
```

Analogous to the bounding box used by Digitransit UI, the assertion about the name of a transport company described in section 6.4 is also configurable via a variable in the `.env` file.

6.7 Reconfiguration to Finland

That we managed to make the trip planner configurable for another region with little effort is shown by the nationwide example for Finland. It can be applied by replacing the top-level `.env` file with `examples/Finland/.env`. The fact that the Finnish GTFS dataset is also neither error-free nor comprehensive, and how the healthchecks were adapted, is described in the following two sections.

6.7.1 Other Data Sources – Similar Problems

In the configuration for Finland, the two primary data sources were replaced by the Finland OSM export and the nationwide GTFS dataset from Fintraffic, which is provided under CC BY 4.0.[24] With the latter, there were problems when downloading it with Docker, as the certificate used by the download server was expired and there were further problems with intermediate certificates. These errors were quickly fixed after personal contact and in the meantime the latest GTFS dataset can be automatically downloaded from the pipeline and further used.

[24]https://developer.matka.fi/pages/en/home.php

Furthermore, we were surprised to find that the Finnish GTFS dataset also contains CSV format errors: Lines of the CSV files that contain a field that has a comma in it have one comma too few elsewhere. This causes them to contain one field less than other lines, which violates the CSV format: "Each line should contain the same number of fields throughout the file" (Shafranovich, 2005, p. 3). This bug has also been fixed after a notice from us.

The optional bikes_allowed information is not included in the Finnish dataset either, so the GTFS-modifications pipeline step is used.

The geographical area was adjusted to match the data sources. To do this, we used an online tool[25] to select a rectangle on a map that encloses Finland as a whole and then entered the coordinates of the corner points in the configuration file.

6.7.2 Adapting the Healthchecks for Different Sources

The main effort of adapting BikeTripPlanner to a different region consists of changing the variables used in the healthchecks of Pelias microservices. For each service of the geocoder, these specify a request and an expected snippet of the JSON response, which is used to detect gross data import errors. New values for the variables can be tested, for example, by starting Pelias locally and accessing the request URLs in the web browser Firefox, which then displays the resulting JSON responses.

In the case of the Libpostal microservice, the address *Itäinen 5, Helsinki, Finland* was selected for the Finland configuration on https://osm.org and inserted percent-encoded into the request URL:

```
HEALTHCHECK_LIBPOSTAL_REQUEST=
  localhost:4400/parse?address=
  It%C3%A4inen%205%2C%20Helsinki%2C%20Finland
```

The percent encoding is done by Firefox when copying the request URL from the address bar.

To match the new address, the variable containing a snippet of the expected JSON response has also been adjusted. Now *nürnberg* (Nuremberg) is no longer expected, but *helsinki* instead:

```
HEALTHCHECK_LIBPOSTAL_RESPONSE=
  '"label":"city","value":"helsinki"'
```

[25]https://boundingbox.klokantech.com/

7 Evaluation

This chapter evaluates the implementation of the multimodal journey planner with regard to the functional and non-functional requirements that were defined in chapter 4. For this purpose, each requirement is examined individually to determine whether it has been met or which aspects are still missing. Subsequently, it is reflected how well the selected open data sources are suited for solving the problem and which difficulties were encountered when working with them.

7.1 Evaluation of Functional Requirements

This section evaluates whether the functional requirements as defined in section 4.1 are met.

F1: Door-to-Door Journey Search

Requirement F1 expresses the need for a search for public transport, whereby it should be possible to enter a start and destination address and receive itineraries that cover the complete travel route from door to door.

On the BikeTripPlanner homepage, it is possible to search for a journey by public transport between any two locations in the area of the VGN, using all types of public transportation operated by the VGN by default. Either addresses can be entered or points on the map can be selected as the start and destination of a journey.

Unfortunately, as described in subsection 2.3.3, not all buildings in OSM are tagged with their full address. In Pfeifferhütte, as an example, there are whole blocks of houses with missing house numbers. This means that the geocoder Pelias cannot estimate a location for many addresses in Pfeifferhütte – even with the use of interpolation (see subsection 5.2.3) – and shows the center of the municipality (Gemeinde Schwarzenbruck) as the result instead.

To summarize, the aspects described in F1 are covered, but with the limitation that not every address is located correctly. After all, in that case it is still possible to manually select the exact location on the map.

Thus, requirement F1 is fulfilled, even if there are local restrictions due to the underlying data.

F2: Bike and Ride

Requirement F2 describes the user preference for shorter connections: It is preferred to bike to a nearby train station if this reduces the total travel time.

After a search query the BikeTripPlanner's overview of suggested itineraries starts with a row containing up to three icons. The first two lead to direct *Walk* and *Bike* connections, the third one leads to a separate overview of itineraries for the *Bike and Ride* and *Bike and Transit* modes. The icons are only displayed if at least one connection has been found for the respective transport modes. Below this row, found *Walk and Transit* connections are listed.

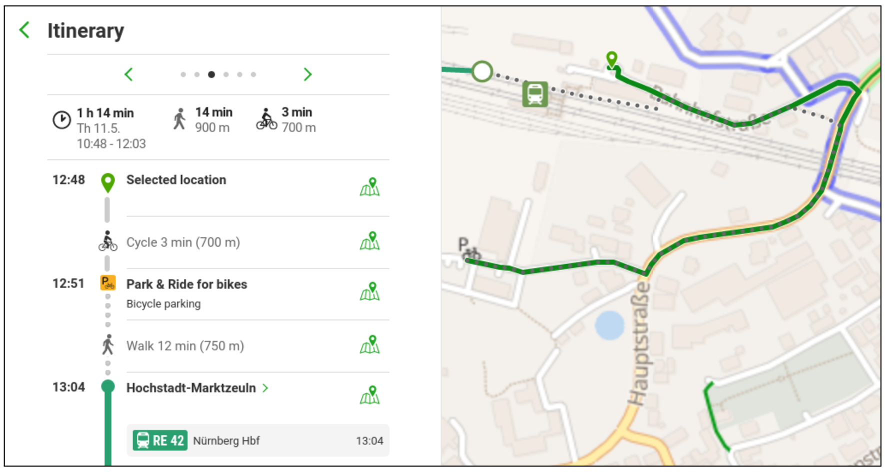
In Digitransit UI, *Bike and Ride* connections are implemented in a way that they start with a bike ride to a parking facility near public stops. From there *Walk and Transit* is used to get to the destination.

By providing both, *Walk and Transit* and *Bike and Ride*, itineraries, the user can compare them and choose a fast connection. If there are parking spaces near train stations, the *Bike and Ride* results are often faster than the corresponding *Walk and Transit* itineraries.

However, not all train stations have bicycle parking facilities and not all of them are mapped in OSM. This leads to results in the BikeTripPlanner that are unlikely to be useful to users. One example is given by the train station Hochstadt-Marktzeuln which according to the station database (Stationsdatenbank Bayern) of the Bavarian Railways Company (BEG, 2019) has no bicycle parking at all. If a starting point close to this train station is chosen, the suggested *Bike and Ride* itineraries lead back to the nearest bicycle parking area in the city before returning back to the station on foot and continuing with the *Transit* part of the journey.

As Figure 7.1 shows, this takes approximately 17 minutes longer than starting directly by train. Here, a more naive implementation of the *Bike and Ride* mode would be preferable, where the journey starts by cycling straight to the first public stop, leaving it up to the user where to park their bike.

Digitransit UI does not offer such an alternative, but since bike parking is available at most train stations, requirement F2 is met – with location-specific restrictions either due to non-existent bicycle parking or due to missing data in OSM.


Itinerary

1 h 14 min
Th 11.5.
10:48 - 12:03

14 min
900 m

3 min
700 m

12:48 Selected location

Cycle 3 min (700 m)

12:51 Park & Ride for bikes
Bicycle parking

Walk 12 min (750 m)

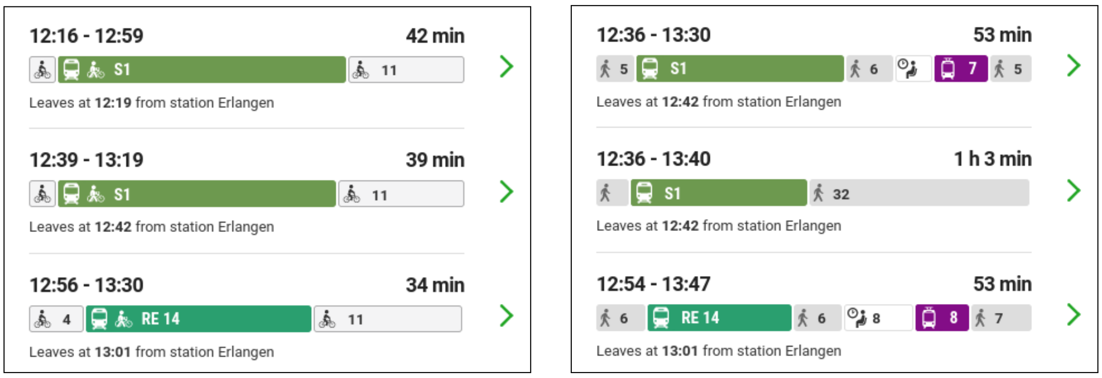
13:04 Hochstadt-Marktzeuln

RE 42 Nürnberg Hbf 13:04

Figure 7.1: Part of a *Bike and Ride* journey in Digitransit UI. Copyright of map data and tiles by OSM contributors.

F3: Bike and Transit

For longer distances, the fastest suggested connections of BikeTripPlanner are those of the *Bike and Transit* mode. Figure 7.2 gives an example where the suggested itineraries from the public library in Erlangen to a bouldering gym in Nuremberg are compared for the *Bike and Transit* and *Walk and Transit* modes.


12:16 - 12:59 42 min
S1 11
Leaves at **12:19** from station Erlangen

12:39 - 13:19 39 min
S1 11
Leaves at **12:42** from station Erlangen

12:56 - 13:30 34 min
4 RE 14 11
Leaves at **13:01** from station Erlangen

12:36 - 13:30 53 min
5 S1 6 7 5
Leaves at **12:42** from station Erlangen

12:36 - 13:40 1 h 3 min
S1 32
Leaves at **12:42** from station Erlangen

12:54 - 13:47 53 min
6 RE 14 6 8 8 7
Leaves at **13:01** from station Erlangen

(a) Bike and Transit (b) Walk and Transit

Figure 7.2: Comparison of suggested routes by Digitransit UI for different transport modes.

However, users must independently verify whether bicycle transport is permitted on the proposed itineraries as this information is not included in the GTFS dataset of the VGN.

By helping users find bicycle-assisted transit connections, requirement F3 is met. This said, there are limitations due to the missing data.

F4: Journey Overview

In the evaluation of requirement F2, it was already described that journey overviews for the *Walk and Transit*, the *Bike and Ride* and the *Bike and Transit* mode are part of the BikeTripPlanner. In addition, the previous figure (Figure 7.2) shows that for each of the suggested itineraries, the departure time, arrival time, and trip duration are displayed. The results are sorted by their departure time and include the number and type of transport modes encountered, from which the number of transfers can also be derived.

In this way, all aspects of requirement F4 on the details of a journey overview, and therefore requirement F4 itself, are fulfilled.

F5: Journey Details

Requirement F5 states that information on finding the correct stops and platforms of a trip is expected from the journey planner.

When an itinerary is selected in BikeTripPlanner, the user is taken to its detailed view, where the time spent walking and biking is summarized in addition to the total trip duration. This can be seen in Figure 7.1 of the evaluation of requirement F2.

In addition, there is information about the journey segments whose list of intermediate stops can be unfolded. Transfers are divided into a walking distance and the subsequent waiting time. Both can be seen in Figure 7.3 (a).

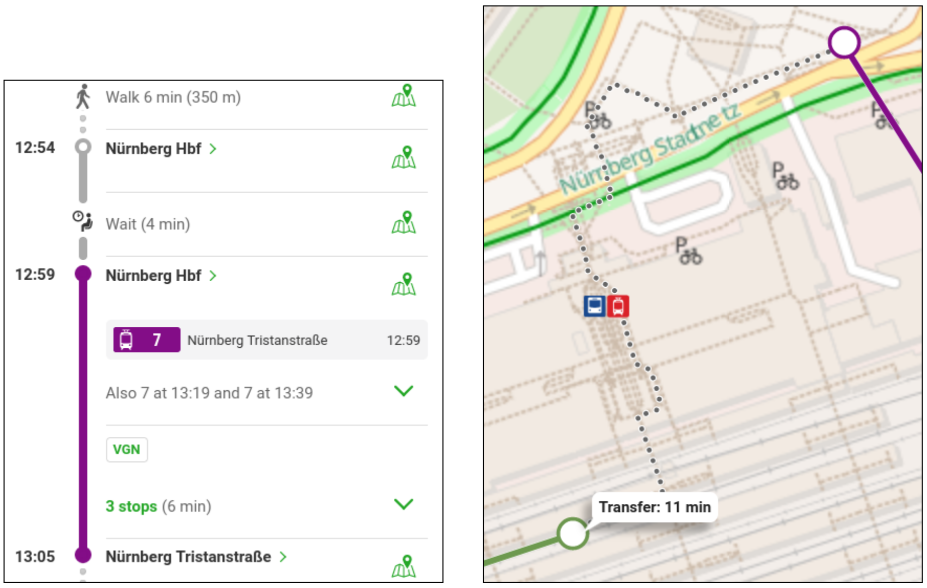
Although Digitransit UI can display platform numbers in the trip details, this information is not included in the VGN GTFS dataset. While Digitransit UI also includes a map view that can show the current location of the device, even with this it can be difficult to find the correct departure point at larger train stations – especially due to less accurate localization in buildings and underpasses.

Figure 7.3 (b) shows the transfer of the connection from Figure 7.3 (a) in the map view. In this case, the footpath leads north out of the station building. If it went the other way, finding the right platform would only be possible via the destination boards on site.

Therefore, requirement F5 is not fulfilled due to missing data.

F6: Bike Route Details

While using a smartphone, for example, it is possible to switch from the detailed view of an itinerary to its visualization on an interactive, bicycle friendly map that highlights bike specific features such as bicycle parking and roads with bike


Walk 6 min (350 m)
12:54 Nürnberg Hbf >
Wait (4 min)
12:59 Nürnberg Hbf >
7 Nürnberg Tristanstraße 12:59
Also 7 at 13:19 and 7 at 13:39
VGN
3 stops (6 min)
13:05 Nürnberg Tristanstraße >

(a) (b)

Figure 7.3: Itinerary details presented as text (a) and on an interactive map (b) in Digitransit UI. Copyright of map data and tiles by OSM contributors.

lanes. In addition, the current location can be shown by touching the circle icon at the lower right edge of the map, which can be seen in Figure 7.4.

This, together with a cell phone mount, supports navigation along the bike routes suggested by BikeTripPlanner and fulfills one of the two alternatives of requirement F6. Nevertheless, the additional option to export a journey would be helpful, e.g. to use voice guidance in an app specialized on bicycle navigation.

However, this does not affect that requirement F6 is fulfilled.

7.2 Evaluation of Non-Functional Requirements

This section examines whether the non-functional requirements as defined in section 4.2 are met.

N1: Data Sources

The data sources selected were the GTFS feed of the VGN for the schedule-based modes and the German OSM export for the personal transport modes. In addition, a data pipeline was developed to process data from these sources and import them into the different components of the application.

This means that all aspects of requirement N1 are fulfilled.

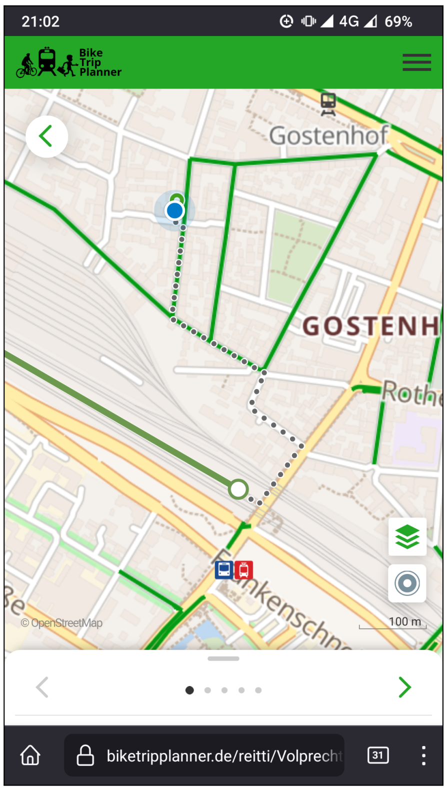


Figure 7.4: Mobile map view of Digitransit UI.

N2: Web Application

With Digitransit UI a responsive web application was chosen as frontend for the BikeTripPlanner. It is ECMAScript 2009 (ES5) compliant, which is supported by all major browsers (Refsnes Data, n.d.) and the use of WebGL is optional.

By realizing the user interface as a web application, requirement N2 is fulfilled.

N3: Mobile Friendly

While on a wide screen the map is visible next to the connection search, the web application adapts to narrow screens.

The previous Figure 7.4 shows a screenshot of a mobile phone displaying the map view of an itinerary. It is possible to control the map via touchscreen and mouse-specific inputs, such as right-clicking, are not necessary. The green smaller sign at the top left returns to the textual representation of the trip.

Since all functions of the web application can also be used in the mobile version, requirement N3 is fulfilled.

N4: Documentation and Code Style

The source code repository of the BikeTripPlanner contains a README file that gives a rough overview of the different components and data sources used. It refers to the configuration file, explains how to run the data pipeline and start the BikeTripPlanner locally. Additionally, a demo instance is referenced to try out the application and there are examples on how to customize it for a different region and run a public instance.

Newly added functionalities such as the GTFS-modifications Python script and configuration via the project-wide `.env` file are provided with meaningful identifiers and explanatory comments.

Attention has also been paid to a consistent style within the various components, e.g. the GTFS-modifications script uses the camel_case naming convention common to Python (Rossum et al., 2023) and optional type annotations have been added.

In this way, all aspects of requirement N4 are achieved.

N5: Testing

For the web interface configuration developed, no tests have been created, because only existing functionalities are used and instead the existing unit and end-to-end test cases of Digitransit UI can be extended.

To generate the map in the OSM Cyclo Bright style, there are two examples for Finland in addition to the map section of the VGN. However, these have to be tested manually, executable test cases do not exist.

Also for the Python script, which offers a good opportunity for unit testing, no test cases have been created yet.

For the data pipeline, the focus was on ensuring that it could be easily automated and integrated into automated workflows, e.g. as part of continuous deployment. That being said, automated execution and subsequent testing of the pipeline results have not yet been implemented.

At least basic end-to-end tests have been created through container healthchecks, which check the presence of certain values in the imported data. With the command `make build test` a new run of the pipeline is started, and after it completed new containers are created and their healthchecks verified. This allows updating the data with a subsequent rough system test.

Overall, important aspects of the data pipeline are already checked, but an automatic execution of it is not yet done and the Python script is missing unit tests. Therefore, requirement N5 is not fulfilled.

N6: Configuration and Operation

The BikeTripPlanner can be adjusted for other regions and data sources (of the same format) through a top-level `.env` file. By creating data images for each of its services and providing Docker Compose project files, it is possible to start the application either locally or publicly accessible through domain and HTTPS certificates with a few commands on most Linux systems.

This satisfies all aspects of requirement N6.

7.3 Evaluation of Chosen Data Sources

The conclusion on the selected data sources is mixed and is described in more detail in the following two subsections. This is followed by a summary of how more of the functional requirements could be met with better data.

The import of data from OSM worked seamlessly and stands out due to its high level of detail. However, the quality of the community-maintained database is strongly location-dependent. The GTFS feed on the other hand comes directly from the transport association and therefore has the advantage of completeness and accuracy of the included data, but there is important information for route planning that is not part of the dataset and we had to fix technical format errors.

OpenStreetMap

Integrating the daily updated German data extract from OSM into the route planner worked without any problems.

Regarding this application, OSM stands out as a data source with many details that are helpful for route planning. Due to available information about driving directions, different types of bike paths and signposted bike routes, a bike-friendly navigation is possible. The navigation is furthermore enhanced by highlighted bike paths on the generated map as well as by marked parking spaces for bicycles. Pedestrian navigation also benefits from the details and takes into account, for example, entrances to buildings and paths within larger station facilities.

With the geocoder, however, the weaknesses of the community-maintained database were also apparent: The data quality is strongly location-dependent. Especially in rural areas, many addresses are incomplete or missing completely. As a result, some addresses can only be located very roughly or not at all when searching for them.

More dynamic information, such as temporary construction sites, can also be entered in OSM, but the database is never completely up-to-date here, because community members must first become aware of something and then add it to OSM.

There are many possible use cases for OSM, but ideally it should be used in conjunction with other data to detect possible mistakes or to fill larger gaps of the data set. For this purpose, the geocoder uses the additional data source Who's On First to complete the geographic hierarchies of the included places. For address data from OSM, this would be possible analogously with OpenAddresses, but this data set does not cover Bavaria.

GTFS Feed of the VGN

On the positive side, the VGN's GTFS feed can be accessed without a login, which allowed for easy integration into the data pipeline using the `ADD` command in a Dockerfile. However, the dataset contains format errors that need to be fixed before further processing is possible.

Unlike OSM, the feed's data comes directly from the producer, the VGN transport association. Therefore, the dataset can score with high completeness and accuracy: All public stops of the VGN are included and the stops of the means of transport are precisely located. Only the infrequent annual update could be improved at this point to include temporary schedule changes.

However, the data set lacks important information about bicycle transport in the individual means of transport and the platform numbers, which limits the fulfillment of the functional requirements, as well as price information that could also be taken into account during route calculation.

Overall, we see it as very positive that the VGN already makes some static data openly available and would like to see the GTFS dataset being expanded even further as well as updated more frequently.

Impact of Higher Quality Data

Although the two data sources still have room for improvement, it was already possible to develop a journey planner that is very suitable for finding bicycle-supported connections in the VGN area. Still, route planning could be improved if more information was available. As Table 7.1 shows, all but one of the functional requirements are already met with the current data sources. However, 4 of the 6 requirements have limitations due to the data situation. With improved data quality and the addition of missing information, all but one requirement could be met by the BikeTripPlanner.

Requirement	**Status**	**Status with improved data**
F1	✓$*_1$	✓
F2	✓$*_{1,2}$	✓$*_2$
F3	✓$*_1$	✓
F4	✓	✓
F5	✗$*_1$	✓
F6	✓	✓

$*_1$: Restrictions due to data quality or missing data.
$*_2$: Restrictions due to non existent bike lockers and missing unsafe parking option.

Table 7.1: Fullfillment of functional requirements with and without improved data.

8 Conclusions

This work shows that it is possible to create a multimodal journey planner using open data, which not only demonstrates as a prototype that bicycle connections can be integrated into public transport, but even realizes this in a user-friendly way.

After first selecting OSM and the GTFS feed of the VGN as open data sources for this purpose, functional and non-functional requirements were then defined to be met by the web application covering the region of the VGN. For the implementation, the existing open source components OpenTripPlanner for route calculation, Digitransit UI as web interface, Pelias as geocoder and Tileserver GL as map service were selected. Using a containerized data pipeline, the current data from the data sources can be processed and imported into the respective services. The resulting Docker images contain the application of the different services bundled with all required data and can be published via a container registry. Through containerization, it is possible to launch the trip planner with Docker Compose on different systems. Finally, it was evaluated to what extent the finished system meets the requirements. In this context, it turned out that one functional and one non-functional requirement were not met. This is due to the fact that platform numbers are missing from the GTFS dataset and within the scope of this work, test cases were not created anymore.

At this point, it should be noted that several problems have been encountered while working with the open data, ranging from technical format errors to incorrect data and missing information. If more of the existing data, such as the platform information available in the system of the VGN, were available as open data, all functional requirements would be met.

In addition to test cases, future integration of the data pipeline into automation tools is advisable: On the one hand, this simplifies daily updates of the data, e.g. to take into account temporary construction sites during routing. On the other hand, this makes it possible to automatically run tests after changes have been made to detect programming and data errors. However, there is also a lot of potential with regard to the data situation itself, which makes it possible to improve the travel planner, because the software used already supports the integration of rental bikes, real-time updates on schedule changes and fare information. Here, it would be desirable that more data sets are included in the German National Access Point and that additionally the address data of the Bavarian State Office for Survey (Bayerische Vermessungsverwaltung) is made available for OpenAddresses in order to be able to realize a reliable address search within Bavaria with Pelias.

As mentioned at the beginning of this thesis, the federal government wants to strengthen public transportation as well as bicycle traffic. With the trip planner developed, there are many examples that show that bicycle-assisted navigation with public transit offers more travel options that are often faster or involve fewer transfers. These connections may already provide an alternative to the car for some. Nevertheless, it is to be hoped that the government invests more in the achievement of its goals and, accordingly, continues to expand public transit and cycling infrastructure so that even more travel connections can be found in the future.

In this context, open projects such as stadtnavi and bbnavi, which are important sources of inspiration for this work, should also be funded further in the future. These projects are also based on the Digitransit platform, which was extended for the present work by a central configuration and bundled as a Docker Compose project. Due to the use of open data and the publication as an open source project, the application developed in this thesis is now available to others for further adaptation, extension and improvement.

References

5T s.r.l. (n.d.-a). *CEN reference data model for public transport* [Transmodel]. Retrieved April 25, 2023, from https://transmodel-cen.eu/

5T s.r.l. (n.d.-b). *How does NeTEx compare with GTFS?* [NeTEx]. Retrieved April 25, 2023, from https://netex-cen.eu/faq/how-does-netex-compare-with-gtfs/

5T s.r.l. (n.d.-c). *Welcome* [NeTEx]. Retrieved April 25, 2023, from https://netex-cen.eu/

5T s.r.l. (2019, September). Transmodel at a glance. Retrieved April 29, 2023, from https://www.transmodel-cen.eu/wp-content/uploads/2015/01/Transmodel_at_a_-glance-1.pdf

Admin, Anasfou, Brede.dammen, Aurige, Andtry & Skinkie. (2023, March 20). *NeTEX* [DATA4pt knowledge base]. Retrieved April 26, 2023, from https://data4pt.org/w/index.php?title=NeTEX#Conversion_of_NeTEx_between_other_formats

AndroidRank. (n.d.). *List of android most popular google play apps* [AndroidRank]. Retrieved April 13, 2023, from https://www.androidrank.org/android-most-popular-google-play-apps?category=all&sort=4&price=all

Antoine D. (2014, April 25). *COPYING* [GitHub. hove-io/navitia] [original-date: 2013-06-19T15:34:30Z]. Retrieved April 24, 2023, from https://github.com/hove-io/navitia/blob/9c86460533ce6900cee1063b0986d98441b41f8f/COPYING

Antoine D, SIMARD, J., Jacquin, A., pbench, Georget, O., Berard, N., & AZIME, A. (2023, March 15). *README.md* [GitHub. hove-io/navitia-docker-compose] [original-date: 2016-12-13T11:32:39Z]. Retrieved April 22, 2023, from https://github.com/hove-io/navitia-docker-compose/blob/177e825289db4f15b438442ef7fc89e526cc
README.md

Antrim, A., Frachet, L., Millet, T., dbabramov, Ababilov, A., heidiguenin, isabelle-dr, Barbeau, S., zhsh, Campagna, G., omar-kabbani, Gräbener, T., Morgan, R., Swartz, P., Paun, N., Ehrenfried, L., Grégoire, L., bhallil ilias, i., Fleischer, G., … bboissin. (2023, March 14). *GTFS schedule reference* [General transit feed specification]. Retrieved April 24, 2023, from https://gtfs.org/schedule/reference/

Apple Inc. (2023). *Top iPhone navigation apps on the app store - apple* [App store preview]. Retrieved April 13, 2023, from https://apps.apple.com/us/charts/iphone/navigation-apps/6010

Arneodo, F. (2015, October). Public transport network timetable exchange. introduction. Retrieved April 25, 2023, from https://www.netex-cen.eu/wp-content/uploads/2015/12/01.NeTEx-Introduction-WhitePaper_1.03.pdf

Baum, M., Buchhold, V., Sauer, J., Wagner, D., & Zündorf, T. (2019). UnLimited TRAnsfers for multi-modal route planning: An efficient solution, 16 pages. https://doi.org/10.4230/LIPIcs.ESA.2019.14

Bayerisches Staatsministerium für Wohnen, Bau und Verkehr. (n.d.). *Verkehrs- und tarifverbünde* [Bayerisches staatsministerium für wohnen, bau und verkehr]. Retrieved January 31, 2023, from https://www.stmb.bayern.de/vum/handlungsfelder/management/verkehrsverbuende/index.php

bbnavi. (2023, April 22). *Digitransit-ui* [GitHub. bbnavi/digitransit-ui] [original-date: 2023-01-26T11:08:46Z]. Retrieved April 22, 2023, from https://github.com/bbnavi/digitransit-ui

BEG. (2019, February). *Stationssteckbrief hochstadt-marktzeuln* [Stationsdatenbank bayern]. Retrieved May 23, 2023, from https://stationsdatenbank.bayern-takt.de/StationsdatenbankBEG/Steckbrief.html?lang=de&efz=8002878

Bundesanstalt für Straßenwesen. (n.d.). *Migrating to the mobilithek* [MDM portal]. Retrieved April 30, 2023, from https://www.mdm-portal.de/migrating-to-the-mobilithek/?lang=en

Byrd, A., & Gran, T. (2022, March 31). *OpenTripPlanner project history* [OpenTripPlanner 2]. Retrieved April 15, 2023, from https://docs.opentripplanner.org/en/dev-2.x/History/

Byrd, A., Gran, T., Begerad, S., Barbeau, S., Shuster, T., Barslett, T., & Vivek. (2023, March 9). *OpenTripPlanner deployments worldwide* [OpenTripPlanner 2]. Retrieved April 26, 2023, from https://docs.opentripplanner.org/en/dev-2.x/Deployments/

CEN TC 278 Working Group 3 Sub Group 7. (2005). SIRI white paper. Retrieved April 25, 2023, from https://www.vdv.de/siri-white-paper08.zipx?forced=true

Commission Delegated Regulation (EU) 2017/1926 of 31 May 2017 supplementing Directive 2010/40/EU of the European Parliament and of the Council with regard to the provision of EU-wide multimodal travel information services (Text with EEA relevance.) (2017, May 31). Retrieved January 23, 2023, from http://data.europa.eu/eli/reg_del/2017/1926/oj/eng

COSSON, N. (n.d.-a). *Digital services to boost mobility* [Navitia]. Retrieved April 14, 2023, from https://navitia.com/en/entreprise/

COSSON, N. (n.d.-b). *Getting started* [Navitia.io documentation]. Retrieved April 22, 2023, from https://doc.navitia.io/#journey_planning

COSSON, N. (n.d.-c). *Mentions légales* [Navitia]. Retrieved April 14, 2023, from https://navitia.com/mentions-legales/

COSSON, N. (n.d.-d). *Navitia.io API prices* [Navitia.io]. Retrieved April 24, 2023, from https://navitia.io/en/tarifs/

Data4PT. (2020, January 31). Methodology for comparing data standards. Retrieved April 29, 2023, from https://data4pt-project.eu/wp-content/uploads/2021/03/Data4PT-Methodology-for-comparing-data-standards.pdf

Davis, M. (2011, September 27). *Data model diagrams for GTFS* [Lin.ear th.inking]. Retrieved April 24, 2023, from https://lin-ear-th-inking.blogspot.com/2011/09/data-model-diagrams-for-gtfs.html

DB. (n.d.). *Timetables* [DB API marketplace]. Retrieved May 3, 2023, from https://developers.deutschebahn.com/db-api-marketplace/apis/product/timetables

Delling, D., Pajor, T., & Werneck, R. F. (2012). Round-based public transit routing, 11. Retrieved November 24, 2022, from https://www.microsoft.com/en-us/research/wp-content/uploads/2012/01/raptor_alenex.pdf

DigitalAgentur Brandenburg GmbH. (n.d.-a). *bbnavi* [DigitalAgentur Brandenburg GmbH]. Retrieved April 22, 2023, from https://www.digital-agentur.de/schwerpunkte/digitale-arbeit-und-mobilitaet/bbnavi

DigitalAgentur Brandenburg GmbH. (n.d.-b). *Mobilitätsplattform für Kommunen in Brandenburg* [bbnavi]. Retrieved April 22, 2023, from https://bbnavi.de/

Directive 2010/40/EU of the European Parliament and of the Council of 7 July 2010 on the framework for the deployment of Intelligent Transport Systems in the field of road transport and for interfaces with other modes of transport Text with EEA relevance (2010, July 7). Retrieved January 31, 2023, from http://data.europa.eu/eli/dir/2010/40/oj/eng

Dutta, V. (2019, August 27). *Travel your first and last mile with google maps* [Google]. Retrieved April 13, 2023, from https://blog.google/products/maps/travel-your-first-and-last-mile-google-maps/

European Comission. (2022, November 7). National access points. a mechanism for accessing, exchanging and reusing transport related data. under delegated acts of the ITS directive (2010/40/EU). Retrieved January 22, 2023, from https://transport.ec.europa.eu/system/files/2022-11/its-national-access-points.pdf

European Cyclists' Federation. (2020, November 19). *TEN-t, EuroVelo and cycling* [ECF]. Retrieved April 30, 2023, from https://ecf.com/what-we-do/ten-t-eurovelo-and-cycling

Franz, M. (2021, November 26). *Ampel-Koalitionsvertrag zum Thema Mobilität: Signale für eine Verkehrswende* [heise online]. Retrieved May 14, 2023, from https://www.heise.de/hintergrund/Koalitionsvertrag-zum-Thema-Mobilitaet-Signale-fuer-eine-Verkehrswende-6277748.html

Geofabrik GmbH. (n.d.). *Europe* [Geofabrik download server]. Retrieved May 25, 2023, from https://download.geofabrik.de/europe.html

GitHub, Inc. (2023, April 22). *Code frequency over the history of stadtnavi/digitransit-ui* [GitHub. stadtnavi/digitransit-ui]. Retrieved April 22, 2023, from https://github.com/stadtnavi/digitransit-ui/graphs/code-frequency

Google. (2020, November 8). *Cities covered* [Google maps transit]. Retrieved April 24, 2023, from https://web.archive.org/web/20201108133310/https://maps.google.com/landing/transit/cities/index.html

Google. (2022a, July). *Google maps/earth additional terms of service* [Google]. Retrieved April 13, 2023, from https://www.google.com/help/terms_maps/?hl=en

Google. (2022b, September 7). *GTFS static overview | static transit* [Google developers]. Retrieved April 24, 2023, from https://developers.google.com/transit/gtfs

Gran, T. (2023, March 24). *Fix: Make sure the default streetRoutingTimeout is used* [GitHub. opentripplanner/OpenTripPlanner@43ff451]. Retrieved April 18, 2023, from https://github.com/opentripplanner/OpenTripPlanner/commit/43ff4517ba276788202cc112f0b5f1c26c142469

Gran, T., Ehrenfried, L., & Byrd, A. (2022, March 31). *OTP interfaces (APIs) and data sources* [OpenTripPlanner 2]. Retrieved April 19, 2023, from https://docs.opentripplanner.org/en/dev-2.x/Interfaces-Data-Sources/?h=interfaces

Gran, T., Junnila, H., & Ehrenfried, L. (2023, March 30). *Raptor/package.md* [GitHub. opentripplanner/OpenTripPlanner] [original-date: 2011-07-18T16:15:38Z]. Retrieved April 15, 2023, from https://github.com/opentripplanner/OpenTripPlanner/blob/2f3b10347d82bcc3860fbafe45864a44fe85856a/src/main/java/org/opentripplanner/raptor/package.md

Holger Bruch, Jannis R, Leonard Ehrenfried, Falco Nogatz & Julius Tens. (2023, May 2). *GTFS-publikationen.* Retrieved May 3, 2023, from https://gtfs.mfdz.de/

HSL. (2023, April 19). *OpenTripPlanner data container* [GitHub. HSLdevcom/OpenTripPlanner-data-container]. Retrieved April 21, 2023, from https://github.com/HSLdevcom/OpenTripPlanner-data-container

Jannis R, Krause, V., derf, Buchholz, M., & Cornu, N. (2023, March 19). *List of transport APIs* [GitHub. public-transport/transport-apis] [original-date: 2020-12-11T18:02:25Z]. Retrieved May 3, 2023, from https://github.com/public-transport/transport-apis

Knowles, N. (2020a, August 12). *Nicholas knowles GitHu page* [GitHub. nick-knowles/nick-knowles] [original-date: 2020-08-12T12:01:52Z]. Retrieved April 26, 2023, from https://github.com/nick-knowles/nick-knowles

Knowles, N. (2020b, August 21). *NationalProfiles* [GitHub. nick-knowles/NeTEx.wiki]. Retrieved April 26, 2023, from https://github.com/nick-knowles/NeTEx/wiki/NationalProfiles

Malkki, J., & Lappalainen, J. (2022, September 20). *Terms of use* [Digitransit]. Retrieved April 21, 2023, from https://digitransit.fi/en/developers/apis/6-terms-of-use/

MDR.DE. (2023, May 11). *Mehr Radwege: Was tut der Bund dafür?* [MDR.DE]. Retrieved May 14, 2023, from https://www.mdr.de/nachrichten/deutschland/politik/fahrradweg-strassenverkehr-kilometer-100.html

MobilityData. (n.d.-a). *Data standards* [MobilityData]. Retrieved April 25, 2023, from https://mobilitydata.org/data-standards/

MobilityData. (n.d.-b). *GBFS: A common language for shared mobility* [General bikeshare feed specification]. Retrieved April 25, 2023, from https://gbfs.mobilitydata.org/

MobilityData. (n.d.-c). *SIRI* [General transit feed specification]. Retrieved April 25, 2023, from https://gtfs.org/resources/siri/

MobilityData. (2022, May 25). *GTFS realtime overview* [General transit feed specification]. Retrieved April 25, 2023, from https://gtfs.org/realtime/

OpenStreetMap. (n.d.). *Communities* [OpenStreetMap]. Retrieved May 2, 2023, from https://www.openstreetmap.org/communities

OpenStreetMap Foundation. (2023, January 19). *OpenStreetMap foundation.* Retrieved May 25, 2023, from https://wiki.osmfoundation.org/wiki/Main_Page

Pieren, Mateusz Konieczny, Esperanza, Extremecarver, Florimondable, Phobie, Alv, Chrabros, Emkey08, Mentor, M!dgard, Sim6, Richard, Olo81, Ulamm, Achadwick, Redrat, Corriere, Trip4you, ... Motp. (2023, April 26). *Bicycle* [OpenStreetMap wiki]. Retrieved May 2, 2023, from https://wiki.openstreetmap.org/wiki/Bicycle

Publications Office of the European Union. (n.d.-a). *Data.europa.eu - the official portal for european data* [European comission]. Retrieved May 3, 2023, from https://data.europa.eu/data/datasets?locale=en

Publications Office of the European Union. (n.d.-b). *Ist-data bus and rail public transport realtime* [European comission]. Retrieved May 3, 2023, from https://data.europa.eu/data/datasets/d1ee4c33-dd6b-47bd-b7a6-39181196f0fe?locale=en

Refsnes Data. (n.d.). *JavaScript ES5* [W3schools online web tutorials]. Retrieved May 23, 2023, from https://www.w3schools.com/js/js_es5.asp

Reith, N. (n.d.). *stadtnavi. Gemeinsam Mobilität neu denken. Das Modellprojekt für vernetzte Mobilität in Herrenberg* [stadtnavi]. Retrieved April 21, 2023, from https://stadtnavi.de/

RESET gemeinnützige Stiftungs-GmbH. (2023, March 27). *Interview: stadtnavi Herrenberg - lokal, multimodal und Open Source zur nachhaltigen Mobilität* [RESET. Digital for Good]. Retrieved April 22, 2023, from https://reset.org/interview-stadtnavi-herrenberg-lokal-vernetzt-fuer-eine-gruene-mobilitaet/

Rossum, G. v., Warsaw, B., & Coghlan, N. (2023, April 30). *PEP 8 – style guide for python code* [Python enhancement proposals]. Retrieved May 23, 2023, from https://peps.python.org/pep-0008/

Roth, M. (2010, January 5). *How google and portland's TriMet set the standard for open transit data* [Streetsblog san francisco]. Retrieved April 24, 2023, from https://sf.streetsblog.org/2010/01/05/how-google-and-portlands-trimet-set-the-standard-for-open-transit-data/

Schildbach, A. (2023a, March 13). *DbProvider.java* [GitHub. schildbach/public-transport-enabler] [original-date: 2013-05-27T15:31:24Z]. Retrieved April 14, 2023, from https://github.com/schildbach/public-transport-enabler/blob/2d8a1a7e84a1455d5eb2f601b98ef8e48ac3a2e3/src/de/schildbach/pte/DbProvider.java

Schildbach, A. (2023b, March 13). *VgnProvider* [GitHub. schildbach/public-transport-enabler] [original-date: 2013-05-27T15:31:24Z]. Retrieved April 14, 2023, from https://github.com/schildbach/public-transport-enabler/blob/2d8a1a7e84a1455d5eb2f601b98ef8e48ac3a2e3/src/de/schildbach/pte/VgnProvider.java

Shafranovich, Y. (2005, October). *Common format and MIME type for comma-separated values (CSV) files* (Request for Comments No. RFC 4180) (Num Pages: 8). Internet Engineering Task Force. https://doi.org/10.17487/RFC4180

Simart, S., Antoine D, Georget, O., Brisset, B., Guillaume P., AZIME, A., SIMARD, J., Bougué, P.-E., l-vincent-l, Haméon, G., & Jacquin, A. (2023, April 14). *Readme.rst* [GitHub. hove-io/navitia] [original-date: 2013-06-19T15:34:30Z]. Retrieved April 14, 2023, from https://github.com/hove-io/navitia/blob/fdb4f4d8f3be007a70cf5827b7ab4583b44fc964/readme.rst

Software Freedom Conservancy. (n.d.). *Member project services* [Software freedom conservancy]. Retrieved April 15, 2023, from https://sfconservancy.org/projects/services/

stadtnavi. (2023, April 22). *Digitransit-ui* [GitHub. stadtnavi/digitransit-ui] [original-date: 2019-06-20T13:54:26Z]. Retrieved April 22, 2023, from https://github.com/stadtnavi/digitransit-ui

Stevea, Gravitystorm, Spesh, ChrisB, Damian, Mzajac, Extremecarver, PeterIto, Eimai, Nguyen, M., NE2, IL vitto, Richard, Dryke, Roberto, F., MacLondon, Ashimema, Lulu-Ann, GercoKees, ... Milliams. (2022, June 23). *Cycle routes* [OpenStreetMap wiki]. Retrieved May 2, 2023, from https://wiki.openstreetmap.org/wiki/Cycle_routes

Tens, J., kemkim & astatio. (2023, May 2). *European transport feeds*. Retrieved May 3, 2023, from https://eu.data.public-transport.earth/

Thomas Gran, Andrew Byrd, Leonard Ehrenfried, Zsombor Welker & vpaturet. (2022, October 12). *Using european data standards* [OpenTripPlanner 2]. Retrieved April 23, 2023, from https://docs.opentripplanner.org/en/dev-2.x/Netex-Norway/

Transport association Rhein-Ruhr. (n.d.). *Deutschlandweite sollfahrplandaten (GTFS)* [OpenData öPNV]. Retrieved May 1, 2023, from https://www.opendata-oepnv.de/ht/en/organisation/delfi/start?tx_vrrkit_view%5Bdataset_name%5D=deutschlandweite-sollfahrplandaten-gtfs&tx_vrrkit_view%5Bdataset_formats%5D%5B0%5D=ZIP&tx_vrrkit_view%5Baction%5D=details&tx_vrrkit_view%5Bcontroller%5D=View

Transportr. (n.d.-a). *Contributing* [Transportr]. Retrieved April 12, 2023, from https://transportr.app/contribute/

Transportr. (n.d.-b). *Free public transport assistant without ads or tracking* [Transportr]. Retrieved April 12, 2023, from https://transportr.app/

TriMet. (2011, July 13). *OTP 2009-2011 RTO grant final report*. Retrieved April 15, 2023, from https://raw.githubusercontent.com/wiki/opentripplanner/OpenTripPlanner/History/2011-07-OTP-Workshop/OTP%202009-2011%20RTO%20Grant%20Final%20Report.pdf

VAG. (n.d.-a). *Bike sharing in nuremberg - easy bike rental - everywhere* [VAG-rad]. Retrieved May 2, 2023, from https://www.vagrad.de/en/nuernberg/

VAG. (n.d.-b). *Datensatz* [VAG-OpenData]. Retrieved May 3, 2023, from https://opendata.vag.de/dataset/

VBB Verkehrsverbund Berlin-Brandenburg GmbH. (n.d.). *API mit Fahrplaninformationen* [VBB. Mobilität mit Zukunft]. Retrieved April 22, 2023, from https://www.vbb.de/vbb-services/api-open-data/api/

Verdy p, PeterIto, Tordanik, Boppet, Harry Wood, LarsF, Aseerel4c26, Richard, Chrabros, Ck3d, Murray, Push-f, Alexa ingerasul 94, Ashimema, OnTour, Joto, Mmd, Multimodaal, GoodClover, ... Deelkar. (2023, May 12). *Elements* [OpenStreetMap wiki]. Retrieved May 25, 2023, from https://wiki.openstreetmap.org/w/index.php?title=Elements&oldid=2525333

VGN. (2022). *Open Data: VGN-Soll-Fahrpläne im GTFS-Format* [VGN]. Retrieved November 24, 2022, from https://www.vgn.de/web-entwickler/open-data/

Welker, Z. (2011, April 18). *Add support for route_bikes_allowed/trip_bikes_allowed* [GitHub. opentripplanner/OpenTripPlanner@6d892ac]. Retrieved April 15, 2023, from https://github.com/opentripplanner/OpenTripPlanner/commit/6d892ace874ce75d740bffc943add2e04d861cf3

Zieger, J. (2022, December 22). *Stadtnavi vorgestellt – Kommunale Mobilität multimodal, nutzerfreundlich und ganzheitlich gedacht* [urban-digital.de]. Retrieved April 21, 2023, from https://urban-digital.de/stadtnavi-kommunale-mobilitaet-multimodal-nutzerfreundlich-ganzheitlich/