MOBIDATALAB

Labs for prototyping future mobility data sharing solutions in the cloud

D4.6 Data Access Services (V2)

Description of Data access interfaces

30/11/2023

Author(s): Johannes LAUER (HERE), Huy Minh NGUYEN (HERE),
Hiba MECHYAKHA (HOVE), Benjamin PLOT (HOVE),
Maroua-Dorsaf DJELOUAT (AKKODIS), Renée OBREGON-GONZALEZ (AKKODIS)

MobiDataLab is funded by the EU under the H2020 Research and Innovation Programme (grant agreement No 101006879).

Summary sheet

Deliverable Number	D4.6
Deliverable Name	D4.6 - Data Access Services V2 – Description of data access interfaces
Full Project Title	MobiDataLab, Labs for prototyping future Mobility Data sharing cloud solutions
Responsible Author(s)	Johannes LAUER (HERE)
Contributing Partner(s)	AKODIS, HOVE, HERE
Peer Review	POLIS, F6S
Contractual Delivery Date	30-11-2023
Actual Delivery Date	30-11-2023
Status	Final
Dissemination level	Public
Version	V1.0
No. of Pages	36
WP/Task related to the deliverable	WP4/T4.3
WP/Task responsible	HERE
Document ID	MobiDataLab-D4.6-DataAccessServicesV2_v1.0.docx
Abstract	The deliverable is a text document that describes the demonstrator for D4.6 Data Access services v2 – Description of Data Access interfaces. The prototype constitutes the practical integration of available data access services and data channels. The task is also related to Task 4.2 Reference Data Catalogue, Task 4.4 Data Processors, and Task 4.5 Anonymization and Data Privacy.

Funded by the European Union

Legal Disclaimer

MOBIDATALAB (Grant Agreement No 101006879) is a Research and Innovation Actions project funded by the EU Framework Programme for Research and Innovation Horizon 2020. This document contains information on MOBIDATALAB core activities, findings, and outcomes. The content of this publication is the sole responsibility of the MOBIDATALAB consortium and cannot be considered to reflect the views of the European Commission.

Project partners

Organisation	Country	Abbreviation
AKKODIS	France	AKKODIS
AETHON SYMVOULI MICHANIKI MONOPROSOPI IKE	Greece	AETHON
CONSIGLIO NAZIONALE DELLE RICERCHE	Italy	CNR
HOVE	France	HOVE
HERE GLOBAL B.V.	Netherlands	HERE
F6S NETWORK IRELAND LIMITED	Ireland	F6S
POLIS - PROMOTION OF OPERATIONAL LINKS WITH INTEGRATED SERVICES, ASSOCIATION INTERNATIONALE	Belgium	POLIS

Document history

Version	Date	Organisation	Main area of changes	Comments
0.1	15.08.2023	HERE	ToC	Initial Version
0.4	17.11.2023	HERE, HOVE, AKKODIS	All	Draft Version
0.5	21. 11. 2023	F6S, POLIS	All	Peer Review
0.6	24.11.2023	HERE	All	TL Rework
1.0	29-30.11.2023	AKKODIS	All	Coordinator Quality Check + Submission

Executive Summary

This is the second version of the prototype integrating new data access services and data channels developed in WP4 and tested in the Living Lab instances. The deliverable is a demonstrator for the provided services. This document provides an overview of the demonstrators and further details on the components. A service catalogue with self-registration options is one of the fundamental developments. Further, access options via Jupyter Notebooks, QGIS and Navitia are described and their interaction with the transport cloud and external data and services are demonstrated. The document is written for practitioners in the mobility ecosystem to get hands-on experience with mobility data sets. Further it is of interest for decision makers to demonstrate the possibilities on how mobility data and services can support the solution of challenges in mobility data depending on domains.

Table of contents

List of figures

Funded by the European Union

Abbreviations and acronyms

Abbreviation	Meaning
CSW	Catalogue Service for the Web
FAIR	Findable Accessible Interoperable Reusable
POI	Point of interest
REST	Representational Stare Transfer
URV	Universitat Rovira I Virgili

1. Introduction

Data access and service is an important part of the mobility data sharing concept. They will enable the usage of mobility data and play a key role in an efficient ecosystem. Since mobility stakeholders are rather a distributed system of independent actors, than a single monolithic system, standards, interoperability and usability of their data, systems and services are mandatory for mobility data sharing. Within this document, examples for data access and services are provided. An extendable, open-source-based catalogue to make services FAIR (findable, accessible, interoperable, and reusable) is implemented. Demonstrators and examples for data and service access are given and the integration within the Transport Cloud as well as within the broader set of mobility data services is described. The usage of the components within the Living- and Virtual Labs enabled direct feedback loops and did influence further deliverables throughout the project.

2. Standards

Relevant standards are described in detail in deliverable D2.4 State of the Art on Mobility Data Sharing Standards. Within the following sub chapters, additional updates on the D4.5 Data Access Services v1 – Pilot set of data provided via services and initial interfaces will be given.

2.1. Data Access

With the ongoing discussions on mobility data spaces, data access layers and their interfaces are evolving. Regarding geospatial data, the OGC API is evolving and establishing new standards for geospatial data access. However, there is still a large diversity of interfaces within the mobility data ecosystem and a gap of common standards.

2.2. Service Access

With the development of web technology, many services are developed and delivered to end-user using standardized REST API (Representational state transfer application programming interface). Although many services in mobility space are well caught up with this trend, the challenges of service discovery and awareness persist. Service access layer is introduced to address these challenges, to increase services exposure and discoverability in the mobility data space, and to promote seamless integration of services API into user applications.

Funded by the European Union

2.3. Interface description

Improving the accessibility of data and services starts with a standardized documentation and a common access layer. With the feedback of the Living Labs and the experience gained by implementing the Transport Cloud, many sharable insights have been explored within MobiDataLab.

3. Data Access Demonstrator

3.1. Access via Jupyter Notebook

In the interest of living labs, several tests have been carried out on the datasets proposed in the catalogue and concocted different combinations to best help participants and give concise and varied examples of the possible analyses.

To achieve this, Jupyter Notebooks are introduced as a tool. Jupyter Notebook – an open-source web application that allows to create and share documents that contain live code, equations, visualizations, and narrative text – provides an interactive environment where the user can write and execute code in small, manageable chunks called cells. This interactivity is especially useful for data analysis, experimentation, and rapid prototyping.

Note that these Notebooks were inspired by the work done within the framework of the “First International Summer School on Data Science for Mobility 2022, Santorini, Greece”.

The functionality of our Notebooks is explained as follows:

‘CKAN datasets standards and accessibility.ipynb’: a non-exhaustive list of data standards available on the CKAN catalogue of MobiDataLab and information on how to access and manipulate them in Jupyter Notebook. The Notebook is cut into different parts:

Packages Installation: Installing Python packages in Anaconda is a simple process that can be done through various methods, such as using the conda command, pip, or the Anaconda Navigator. For more details about these methods, please refer to this tutorial: https://www.tutorialspoint.com/how-do-i-install-python-packages-in-anaconda

Datasets Reading with Pandas: The Pandas packages has been chosen to deal with different Data formats in CKAN, such as text formats (CSV/TXT), XML files (XSD, XLSX, ODS, etc.). Pandas is a Python library for data analysis. It introduced two new types of objects for storing data that make analytical tasks easier and eliminates the need to switch tools: Series, which have a list-like structure, and DataFrames, which have a tabular structure.

Funded by the European Union

Datasets Reading with GeoPandas: GeoPandas is an open-source Python library, that provides easy-to-use tools and data structures for working with geospatial data. It is built on top of other popular Python libraries like Pandas, Matplotlib, and Shapely, and it simplifies the manipulation and analysis of geospatial data, including vector data in the form of shapefiles, GeoJSON, etc.

***'Data visualization with OSMnx. ipynb'*:** This Jupyter Notebook is a concrete example using the municipality of Leuven (and some more generic use cases) on how to use the OSMnx library to extract and visualize data from OpenStreetMap. OSMnx is a Python package that simplifies downloads of geospatial data from OpenStreetMap and model, project, visualise, and analyse real-world street networks and any other geospatial geometries.

In this example, the focus is on micro mobility in Leuven. The modal shift is part of Leuven's long term policy plans and can be seen as a challenge on its own. Modal splits and monitoring of the modal splits can be difficult, which translates in many projects and many plans aiming to get more people on public transport, on the bike, on new mobility and to encourage the combination of modes. The modal share of shared mobility services is still low, compared to walking or private vehicles (motorized and non-motorized).

Each city has its own characteristics, and Leuven has a lot of students with high bike use for internal movements. Additionally, most inhabitants have their own bikes. The city of Leuven has been working on mobility plans regarding cycling lanes and cycling streets, but there is still room for improvement.

The challenge in this case is to study, based on the available data, the relevant criteria (Housing, real estate, etc.) to decide where traffic lanes are needed. The aim is to provide a general framework helping decision-making on very objective data.

As for the procedure followed, Leuven's street network has been imported from OSM. A function that simplifies graph topology by removing all nodes that are not intersections or dead-ends has been used. The plotted data is shown below.

Funded by the European Union

```
In [2]: # Specify the name that is used to seach for the data
place_name = "Leuven, Belgium"

# Fetch OSM street network from the location
graph = ox.graph_from_place(place_name, network_type='bike', simplify=True) #Simplifies graph topology by removing all nodes that

# Plot the streets
# the plot will get a toolbar menu at the bottom left, in which you can enable mouse zooming
fig, ax = ox.plot_graph(graph, node_size=0.5, edge_color='grey')
```

Figure 1: Example of Leuven's street network from OSM

After that, graph edges (street network) and the place name (Leuven, Belgium) have been loaded into a GeoDataFrame. Building geometries with tags and place names have been loaded into a GeoDataFrame. Further tags can be loaded by using the presented library (e.g.: cafes, markets, parking lots). The improved rendering is shown in the screenshot below.

```
In [14]: fig, ax = plt.subplots(figsize=(3.5,3.5))

# Plot the footprint
area.plot(ax=ax, facecolor='black', zorder=0)

# Plot street edges
edges.plot(ax=ax, linewidth=0.5, edgecolor='grey', zorder=1)

# Plot buildings
buildings.plot(ax=ax, facecolor='yellow', alpha=0.7, zorder=2)

# Plot bicycle parkings
parks.plot(ax=ax, color='green', alpha=0.7, markersize=10, zorder=3)


plt.tight_layout()
plt.axis('on')
```

Figure 2: Plotting data with GeoDataFrame

'Data visualization with Matplotlib .ipynb' : This notebook contains examples on how to use the Matplotlib Python library to filter, manage and visualise statistical data. Matplotlib provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble the one of MATLAB.

In this example, the environmental impact of the Low Emission Zones (LEZ) in the city of Milan, Italy has ben analysed. The low emission zone in Milan, called "area B", is covering 90% of the city, that in coming years will progressively become more restrictive with certain categories of vehicles being banned from it.
The challenge is to contribute to the analysis of LEZ (thanks to e.g., air quality reports), perform cross-domain analysis and make these reports available as a service with interoperable interfaces. After processing the raw data (translation of metadata from Italian into English, datetime conversion, columns filtering, etc.), the following graph shows the pollutant emission by month and station ID:

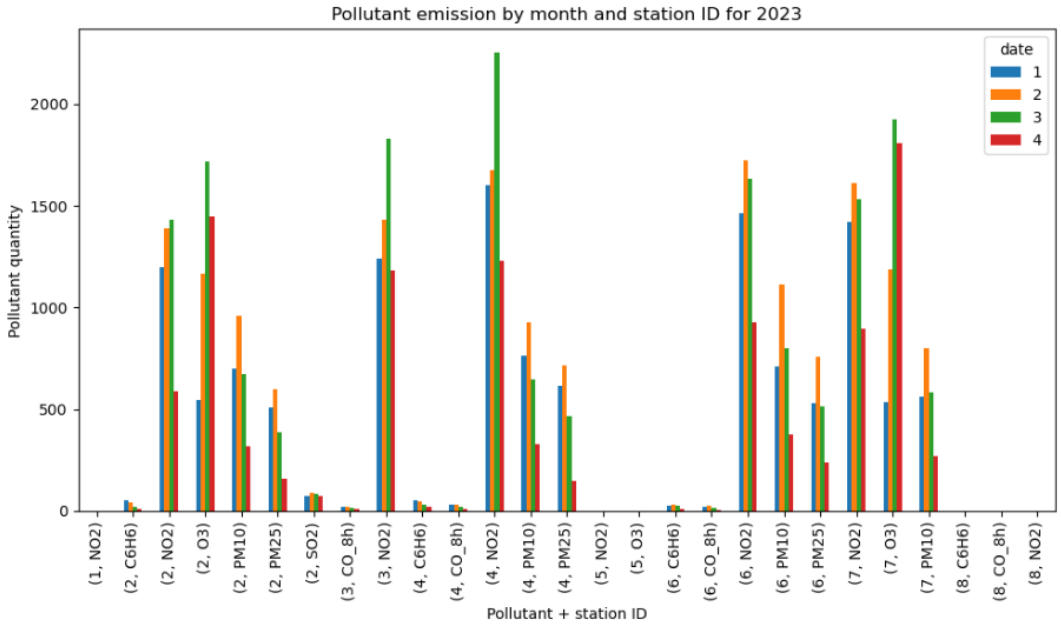

Figure 3: Result of a cross-domain analysis

'Data visualization with Folium .ipynb': A tutorial on how to use Folium and its different functionalities to manipulate your datasets. Folium is a Python library for visualizing geospatial data on an interactive leaflet map. It enables the integration of a base map with specified width and height and either default map styles or customised ones. As default map styles Folium provides OpenStreetMap, Mapbox, CartoDB (incl. positron and dark_matter) and further.

In this tutorial, an example with data provided by the BikeMi service of Milan, Italy is described. The used dataset consists of location, characteristics and number of stalls offered of the BikeMI Bike Sharing stations.

A screenshot of the resulting map, made with Folium. The map can be customized as examples are given in the Notebook:

```
In [11]: # Create a Map instance
         m = folium.Map(location=[45.4641943, 9.1896346], zoom_start=10, control_scale=True)

         m
```

Out[11]:

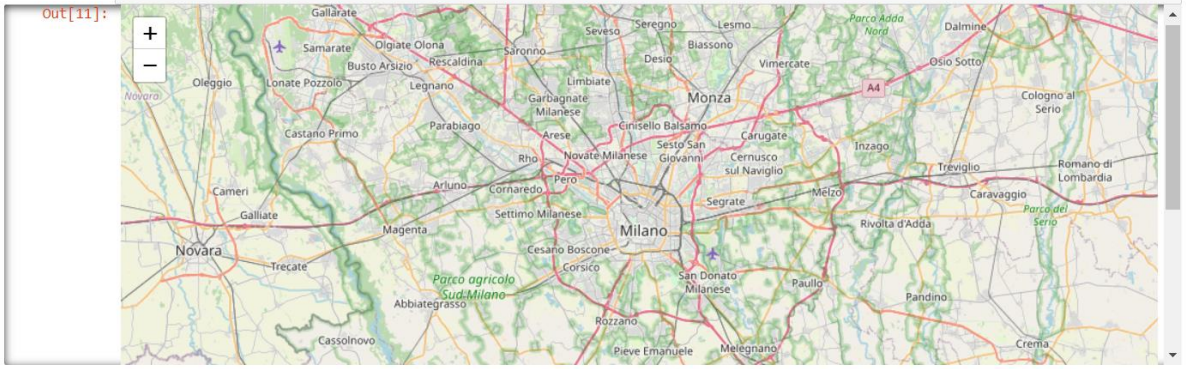

Figure 4: Example of Data visualization with Folium

The processed BikMi stations,are displayed as markers on the map:

```
In [11]: # Convert points to GeoJSON
         points_gjson = folium.features.GeoJson(df, name="BikMi stations")

In [12]: # Create a Map instance
         m = folium.Map(location=[45.4641943, 9.1896346], tiles = 'cartodbpositron', zoom_start=13.5, control_scale=True)

         # Add points to the map instance
         points_gjson.add_to(m)

         # Alternative syntax for adding points to the map instance
         #m.add_child(points_gjson)

         #Show map
         m
```

Out[12]:

Figure 5: Example of displayed BikMi Stations

3.2. Access via QGIS

QGIS[1] is a “Free and Open Source Geographic Information System” in which we can visualize, treat, modify, analyze and publish geospatial data. It provides access to many platforms with many extensions/plugins to connect to data stores (local and cloud-based) and execute external/cloud-based processing (e.g. routing). It also proposes algorithms to do spatial analysis. QGIS is a tool that allows to find and access the metadata collected in the MobiDataLab CKAN and GeoNetwork portals easily. It is easy to install, the only particular recommendation for new users when downloading the program is to choose the most stable version of QGIS.

GeoNetwork[2] is an open-source reference tool to manage geo data catalogues. Besides the support of catalogue service for the web (CSW), which is a common standard in the spatial metadata ecosystem, the software also supports further catalogue formats and has import capabilities for self-defined metadata structures. With its metadata harvester capabilities, it can also load larger sources automatically and scheduled over the web.

CKAN[3] is an open-source open data platform used by many governments around the world to publish and manage their open data. The basis of CKAN is to make data more discoverable for users and re-users, for citizens, and for data consumers at large. CKAN is a highly extensible product with several useful external extensions that can be installed separately (e.g., data storage, harvesting, CSW support, etc.).

The CKAN and GeoNetwork catalogues were created as they are quite accessible through different environments and coding languages. Their API functions and their Catalogue Service for the Web (CSW) allow to expose, search and get data or metadata through Geographical Information Systems (GIS) such as QGIS. Further information on these catalogues is provided in the deliverable D4.2 and D4.4, which are part of Task 4.2.

This section demonstrates an end-to-end workflow from data discovery to data visualization, using the MetaSearch tool for accessing GeoNetwork and other catalogues and the CKAN plugin for CKAN catalogues.

3.2.1. MetaSearch tool – GeoNetwork

MetaSearch is a QGIS built-in tool to interact with metadata catalog services, supporting the OGC Catalog Service for the Web (CSW) standard. MetaSearch provides an easy and intuitive approach and user-friendly interface to searching metadata catalogs within QGIS. In this example, the CSW API of the GeoNetwork catalogue is used.

[1] https://qgis.org/fr/site/
[2] https://geonetwork-opensource.org
[3] https://ckan.org/

Funded by the European Union

To launch the MetaSearch tool consult the “Web Menu” (on top of the QGIS menu), select MetaSearch and select again MetaSearch. Then, add the Catalog Service for the Web (CSW) by selecting the “Services” tab and “New”. The next step is to add the CSW URL (https://geonetwork.mobidatalab.eu:8443/geonetwork-4.0.5-0/srv/eng) on the “Service” tab, assign a name to the portal, save it and click on “OK” (the steps can be seen in Figure 6).

After setting up the CSW connection (Figure 6), the user can proceed to query the data of interest via keywords or spatial extent. The screenshot, Figure 7, shows some results from the reference data sets, denoted with the keyword “mobidatalab”.

To search for specific keyworks (type them on the search bar and click on search) or to see all the available datasets on the portal (leave empty the search bar and click on search).

To filter data on a particular area of the world, change the map coverage according to the area on which your QGIS project is focused with the button “map extent” or to access the metadata all over the world by using the “set global” button.

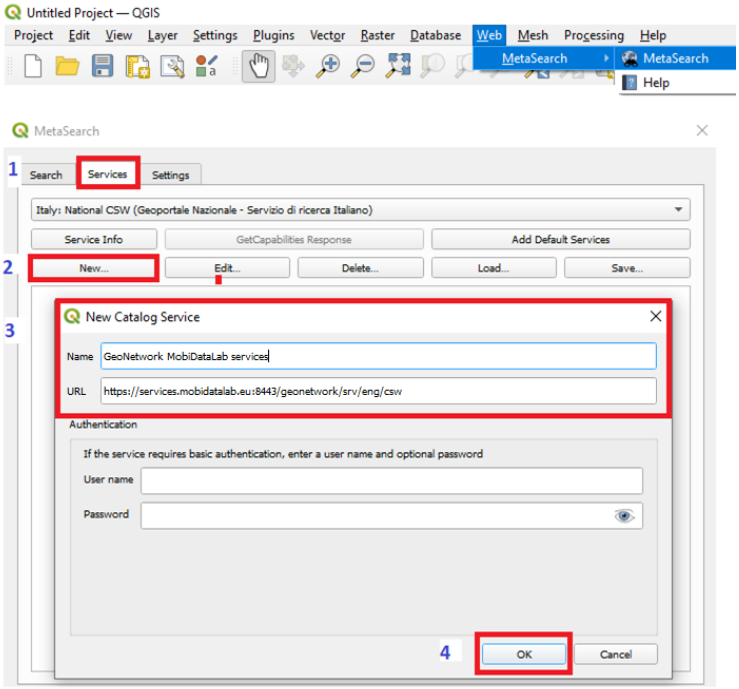

Figure 6: Setup CSW catalogue source in QGIS MetaSearch

When selecting a service or a dataset, it is possible to get the metadata record, which provides the link to access the source of the data. It is also possible to download the results as XML.

If the service or the dataset is in the right format to be read by QGIS, it will be possible to load the data.

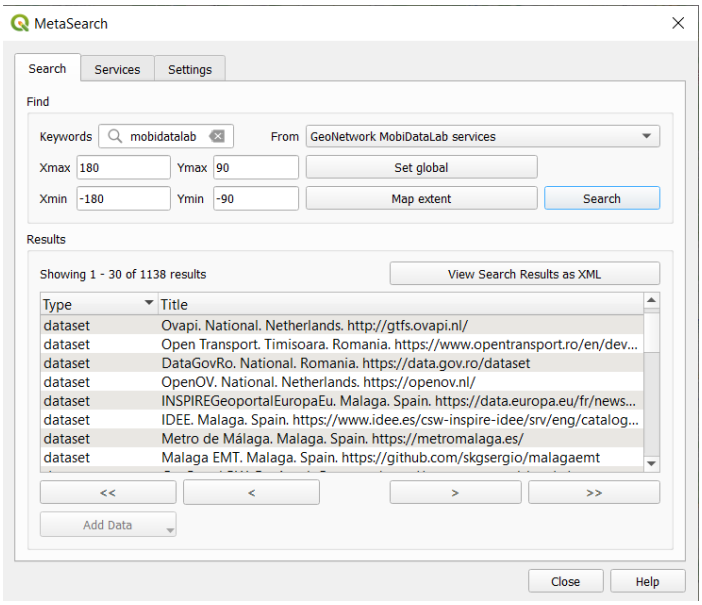

Figure 7: Query for keywords in CSW catalogue in QGIS MetaSearch

Figure 8 shows more details of the selected dataset. In the links section, the user can navigate through the resources to obtain the published data. For supported resources like WMS or WFS data sources, they can be loaded directly in QGIS without intermediate steps.

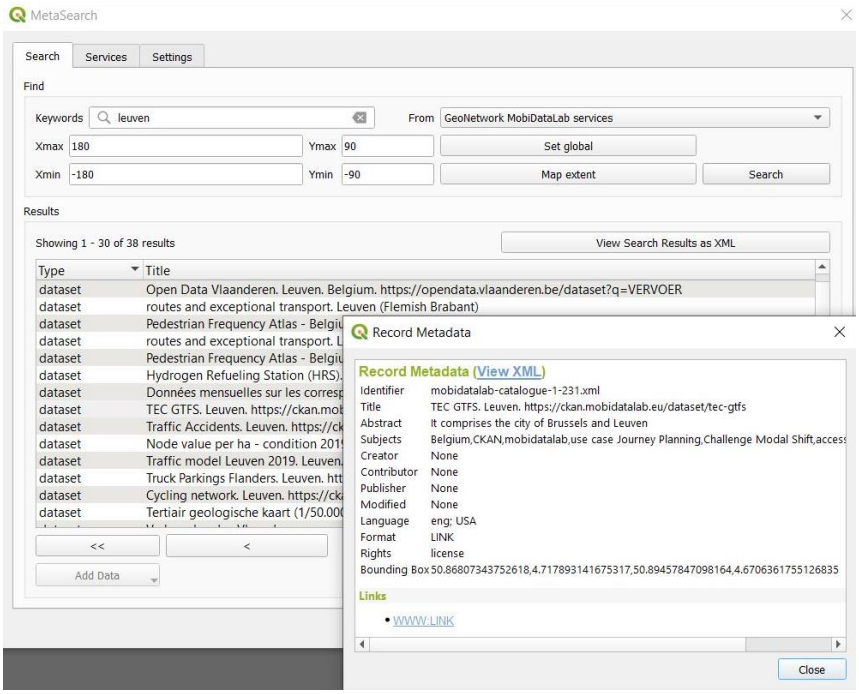

Figure 8: Detail of a metadata CSW record in QGIS MetaSearch

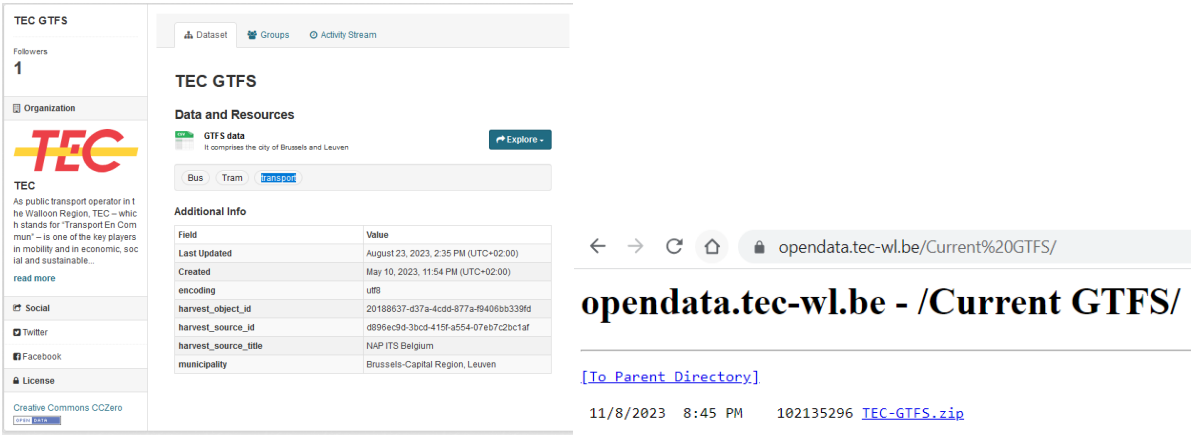

Figure 9: Detail of a metadata CSW record in the linked resource in the web browser

In this example, the link to the publisher's website is listed (Figure 9). Therefore, to access the dataset the user needs to view and download the data in the web browser and then load it into QGIS. The data in Figure 9 refers to the TEC GTFS and can be found in the MobiDataLab GeoNetwork catalogue (link).

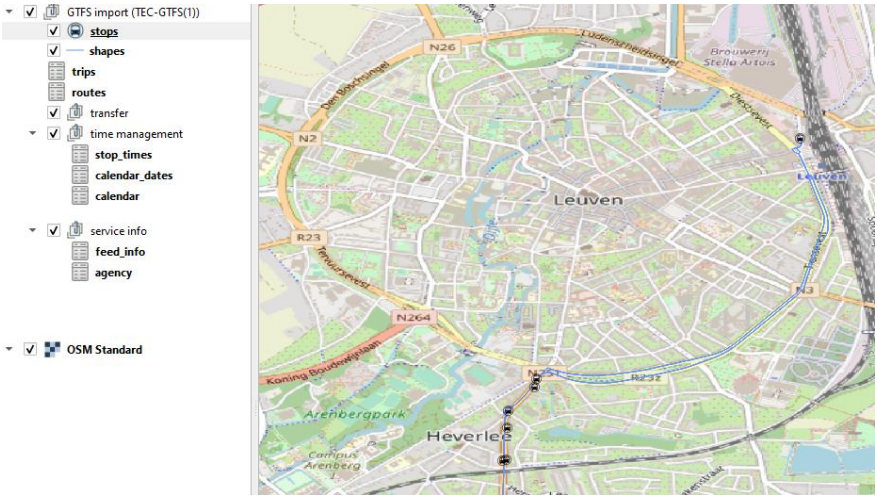

Figure 10: QGIS display of a GTFS of the public transport operator TEC in Leuven

This example entails loading a GTFS file into QGIS with the plugin “GTFS Go”. The dataset visualization can be seen in Figure 10, which show the public transport operator lines and stops in Leuven.

This end-to-end example demonstrates a typical data acquisition, loading and analysis workflow which is part of many challenges about mobility data. On the one hand, it shows that a tool like QGIS with its many plugins allows users to harmonize data and combine external datasets to fulfil the analytics and visualization. However, this is only enabled in the single client tool given the fact that the specialized plugins are integrated and are fitting to the available data (formats).

To avoid this, and achieve better interoperability, it is proposed to use open standards for tool independent interoperability. Bringing together data providers to solve the challenges within the mobility domain is one of the main goals of MobiDataLab.

To add a Web Map Service (WMS) layer directly from the MetaSearch, when the service is offered, it is just necessary to follow the following steps (see Figure 11: Steps to add a WMS/WMTS via the Metasearch). Open the Metadata search, select the desired CSW and search for a keyword.

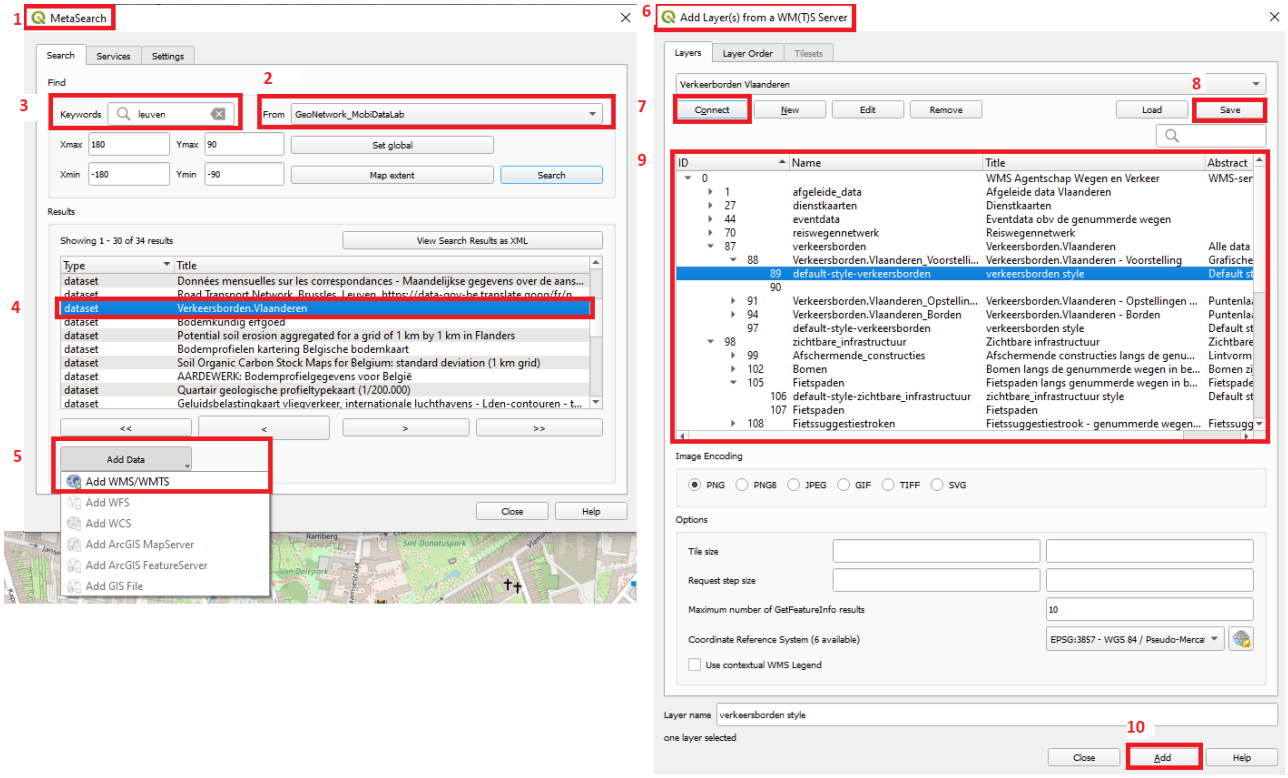

Figure 11: Steps to add a WMS/WMTS via the Metasearch

Then, select an available WMS and click on “Add data”. An “Add Layer(s) from a WM(T)S server” window will open, click on “connect” and “save” if you wish to save the information of the server. One or more layers will be displayed, from this list one or more can be select it and add it into the QGIS project.

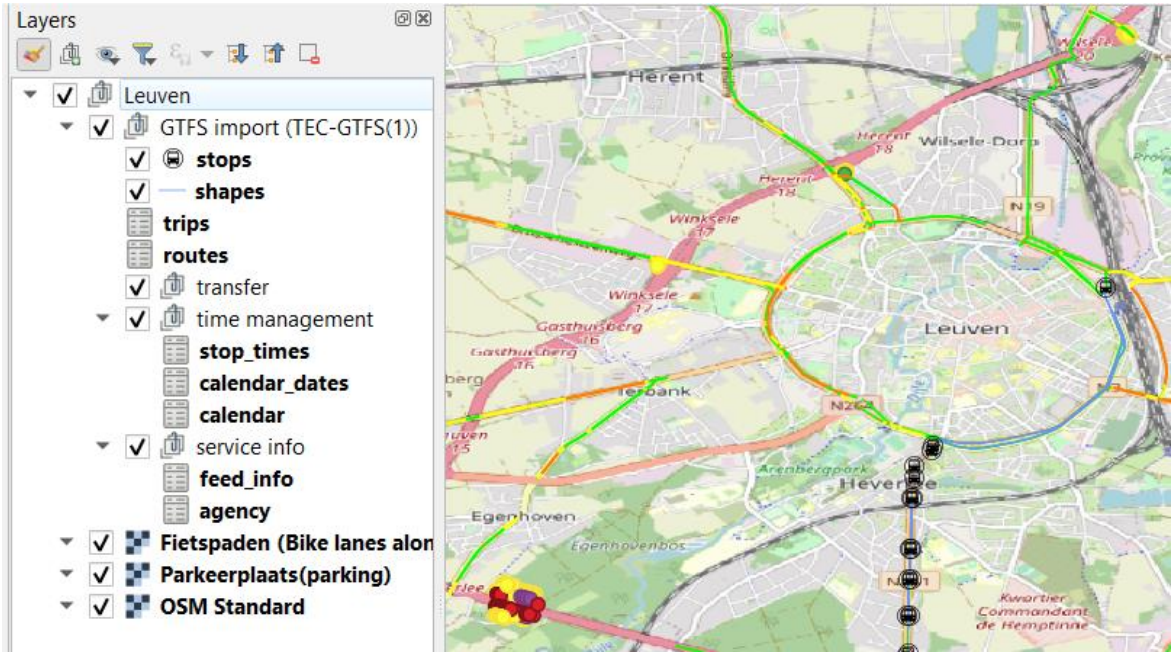

Figure 12: WMS bike lanes and parking around Leuven

3.2.2. CKAN plugin – metadata access

To install the CKAN plugin it is necessary to go to the “Plugins” menu (on top the QGIS menu bar) and select “Manage and Install Plugins” and search for the CKAN-Browser plugin and click on install.

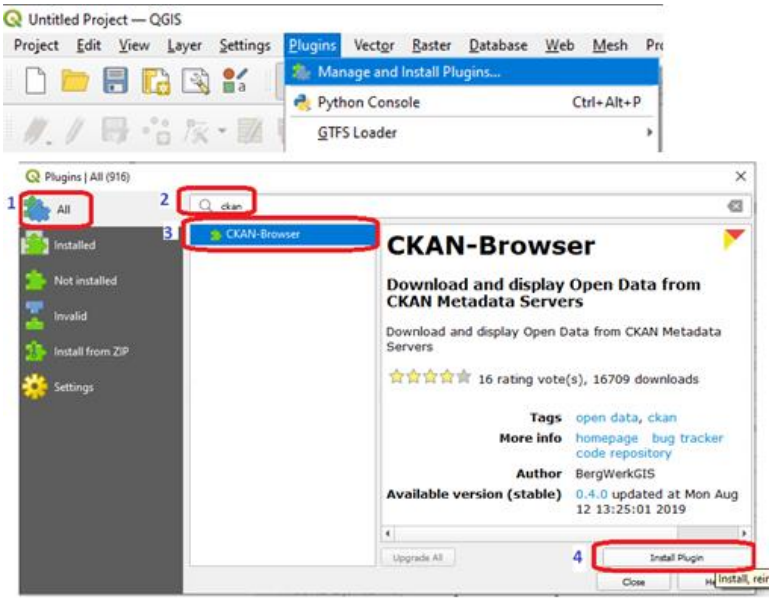

Figure 13: CKAN Plugin installation

To browse and access the CKAN metadata, it is necessary to click on the CKAN icon that appears of the QGIS toolbar and enter the URL of the CKAN instance (in this case https://ckan.mobidatalab.eu/api/3/) to add to the QGIS CKAN servers list (see Figure 14). By default, there are some CKAN portals already on the list. To verify that the right URL was entered, the connection can be tested. If the connection is successful, it is possible to save the link to the server on the list to find quickly when using the plugin.

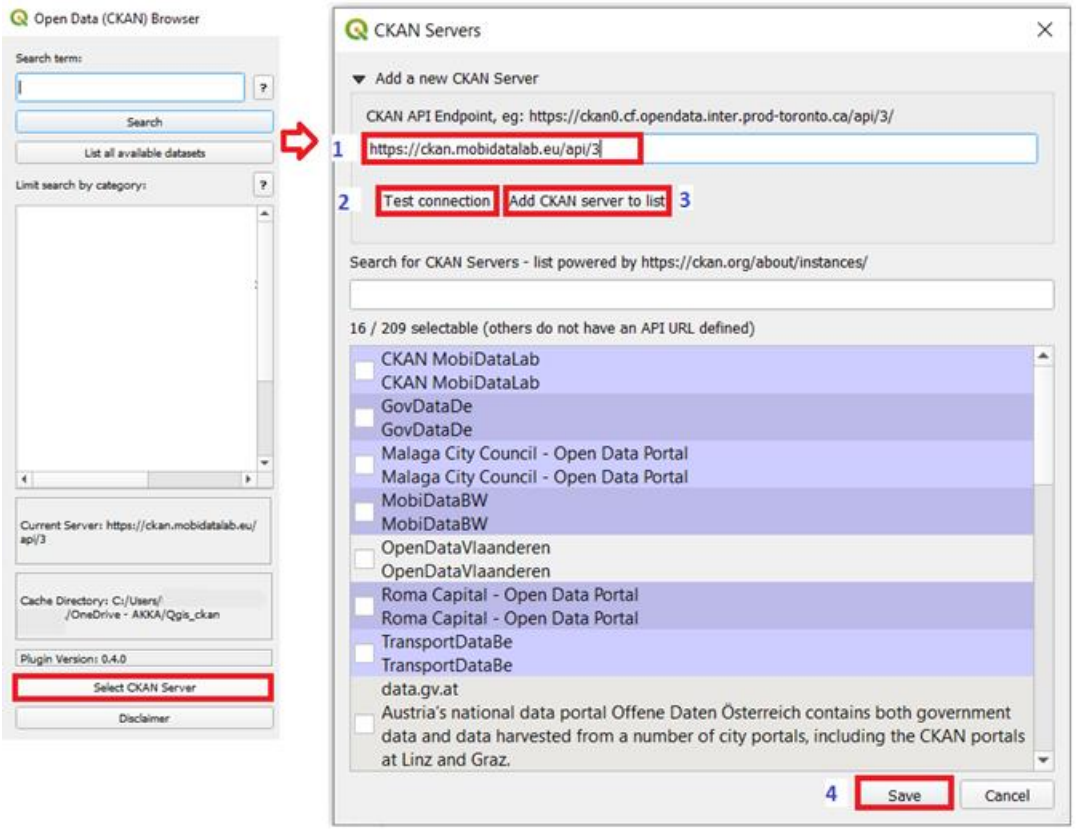

Figure 14: Adding a CKAN catalogue into the plugin list

The next step is to select the desired CKAN instance and start searching for a particular term or for all the datasets on the instance see Figure 15 (by leaving the search box in blank and just clicking on list all the available datasets).

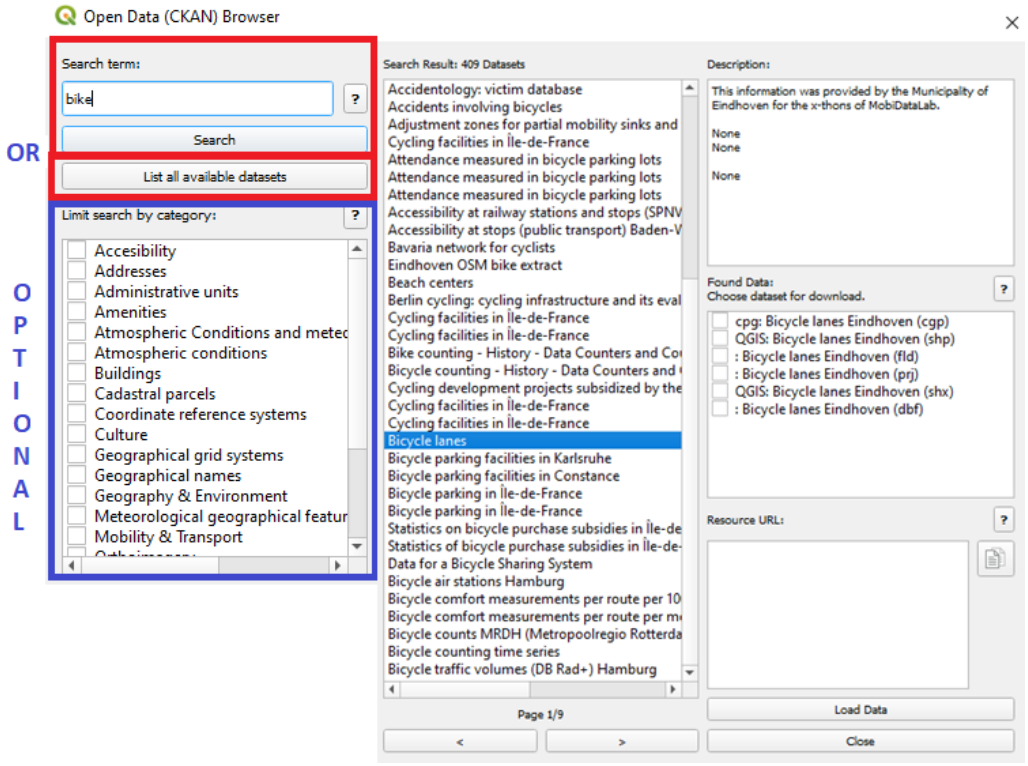

Figure 15: CKAN metadata browser

Datasets can be discovered and filtered through categories (also known as groups on the CKAN interface) proposed on the left side or by selecting one or more of these categories (however, it is important to mention that not all the datasets have been assigned into these categories), cf. Figure 15.

When selecting a particular dataset, the description and the format will be showed on the right boxes.

When the datasets are in a format that can be read directly by QGIS, it will be possible to select them and load them.
Otherwise, there is often the possibility to download the data to consult it by other means. To save downloadable datasets, it is necessary to create a folder “cache directory” where we will be able to find them easily. This can be set by going to the CKAN plugin settings, Figure 16.

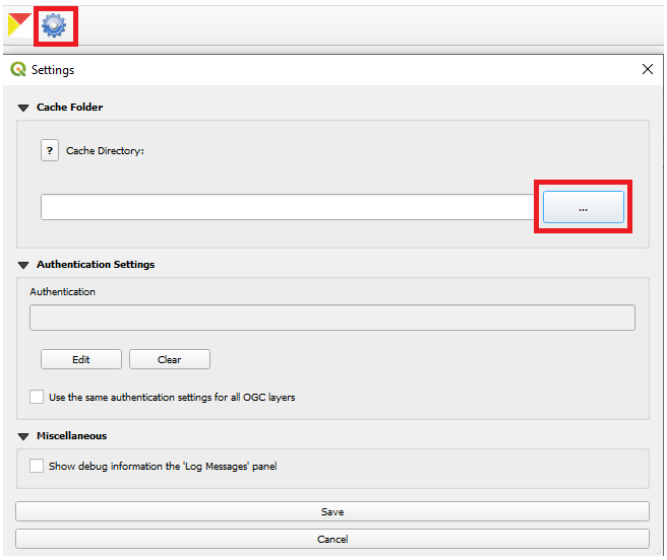

Figure 16 CKAN Plugin settings

If you wish to add or access a web service manually after finding a link in the browser, the connection can be established by going to the “Layer” Menu, selecting “Add layer” and selecting the type of layer wished to be added, see Figure 17.

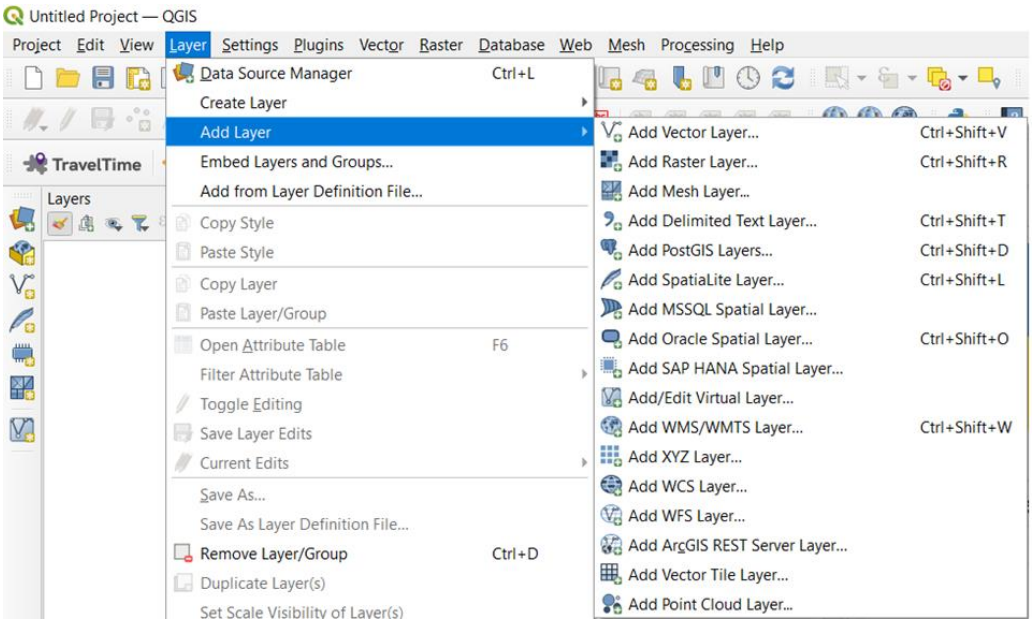

Figure 17: Add manually a layer

To get more information about the steps to follow to add different data formats in QGIS we created a quick guide for new users. This guide is available in the GitHub repository of MobiDataLab: https://github.com/MobiDataLab/QGIS-guide-CKAN-GeoNetwork-access.

To load the data downloaded from the CKAN browser, it is possible to slide layers from the “Cache folder” into the QGIS project or add the layers via the “Layer” menu by selecting “Add Vector Layer” and searching on the cache directory for the d downloaded file. In the example below a shape file will be loaded (see Figure 18 and Figure 19).

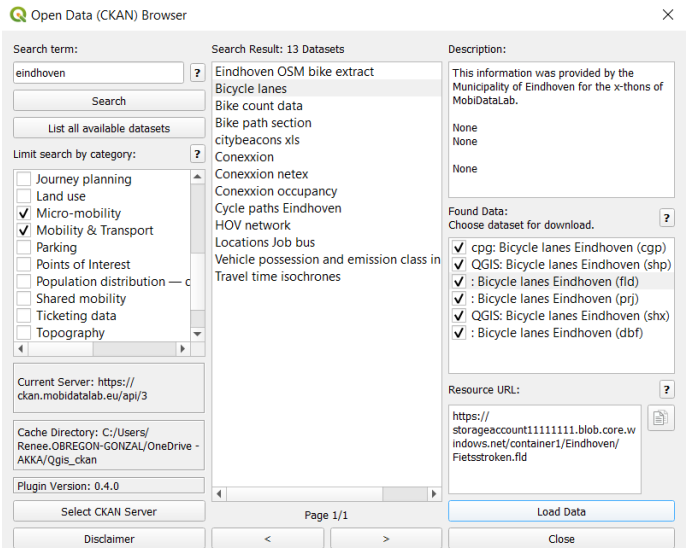

Figure 18: Load data found in the CKAN plugin browser

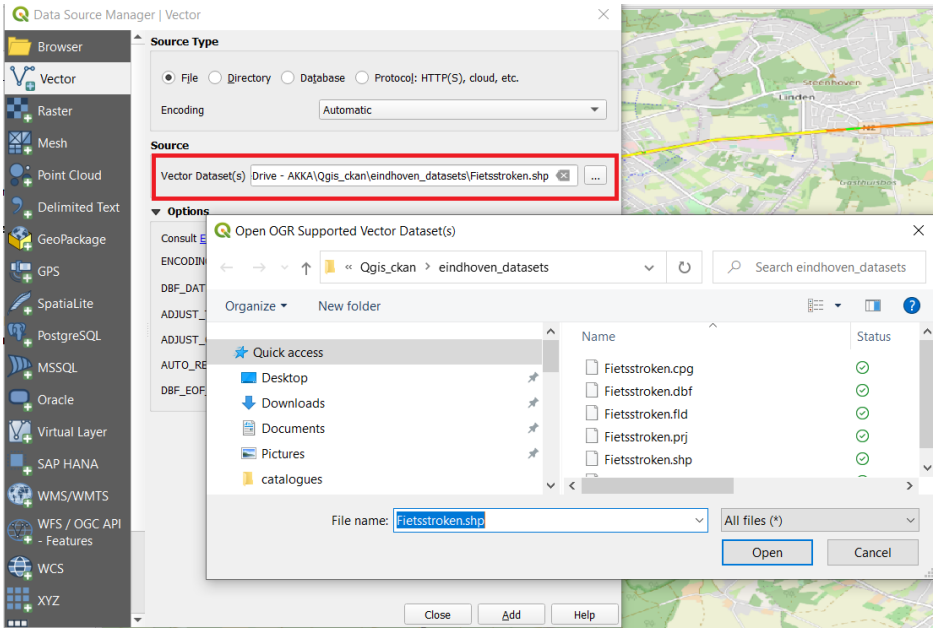

Figure 19: Load downloaded data into QGIS

An advantage of the MobiDataLab catalogues is that we have pre-filtered data, so all the datasets related to this project and reference groups are federated in only two portals, which makes data more accessible, findable and linkable.

With this data it is possible to analyse different use cases. For example, if we explore the datasets of the Municipality of Eindhoven, we can display the safe cycling areas, train, bus and taxi stations. This information can be then analysed with the processing tools of QGIS (see Figure 20) so we can find if bike and ride is possible for people coming from outside the city or identify how accessible the city is for cyclist and people with reduced mobility arriving to the city.

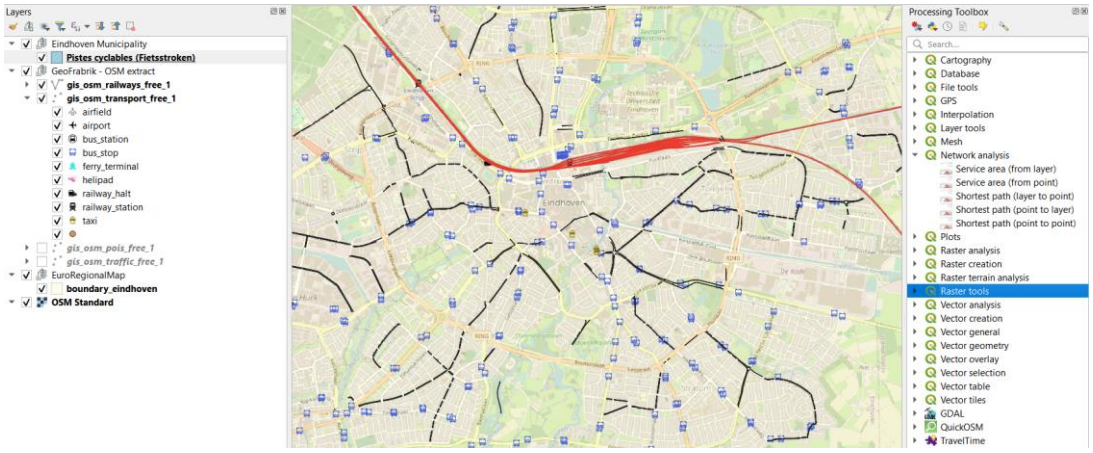

Figure 20: Processing toolbox of QGIS and network analysis

There are many plugins to discover, manipulate, display and analyse data. To get some ideas consult the annex at the end of this deliverable (page 35).

3.3. Access via Navitia

Navitia, a multimodal passenger information system is a part of the living labs' available tool stack. With a given URL and token, the participants can access its APIs (Multimodal journey planning API, autocompletion API, Isochrone API…) that uses multiple injected data sources such as transport schedule data (GTFS, NetexFR, NTFS…), Real-time data (Siri…), disruptions' data (Specific disruption object in JSON format), geographic data (OSM data, POIs csv files, Polygon files for geographic outlines…), accessibility data and so on.
Navitia Playground https://playground.navitia.io/index.html enables also accessing Navitia services for testing or exploration purposes as well. It simplifies selecting parameters and lists several examples of the different call types.
Here are a few examples of the data found using Navitia:

- Accessing Transport planning Data: The user can search for public transport schedules of a specific service for example: In the Figure 21 below, to get the timetable of a running bus service in Berlin, we use:

URL	https://api.navitia.io/v1/coverage/de/lines/line%3AOBE%3A17283_700/route_schedules?from_datetime=20231115T070000&

Figure 21: Timetable of a bus service in Berlin

Customizable journey planning (using transport data and accessibility data): we can customize the journey planning by specifying the traveler's type (slow_walker, wheelchair and so on). Here is an example of a customized journey in Eindhoven for a wheelchair user:

URL	https://api.navitia.io/v1/coverage/nl/journeys?from=stop_area%3AOAM%3Astoparea%3A345454&to=stop_area%3AOAM%3ANavitia%3A2402786&language=en-GB&traveler_type=&datetime=20231108T080000&

Accessing disruptions: transportation disruptions can also be accessed using the URL.

URL	https://api.navitia.io/v1/coverage/fr-idf/disruptions?

The example in the figure below describes an incident at a stop point, the objects impacted which could describe the transportation services, the application period, and the severity of the incident.

```
{
    "application_periods": [
        {
            "begin": "20231106T101200",
            "end": "20231106T105800"
        }
    ],
    "category": "Incidents",
    "cause": "perturbation",
    "contributor": "shortterm.tr_idfm",
    "disruption_id": "00134510-7c85-11ee-bc68-0a58a9feac02",
    "disruption_uri": "00134510-7c85-11ee-bc68-0a58a9feac02",
    "id": "0013efce-7c85-11ee-bc68-0a58a9feac02",
    "impact_id": "0013efce-7c85-11ee-bc68-0a58a9feac02",
    "impacted_objects": [
        {
            "pt_object": {
                "embedded_type": "stop_point"
                "id": "stop_point:IDFM:463144",
                "name": "Chaussée d'Antin - la Fayette (Paris)",
                "quality": 0,
                "stop_point": {9 items},
            }
        }
    ],
    "messages": [4 items],
    "severity": {
        "color": "#EF662F",
        "effect": "SIGNIFICANT_DELAYS",
        "name": "perturbée",
        "priority": 30
    },
    "status": "past",
    "tags": [
        "Actualité"
    ],
    "updated_at": "20231106T101517",
    "uri": "0013efce-7c85-11ee-bc68-0a58a9feac02",
}
```


Accessing POIs of a specific OSM category or a defined category assigned in the Navitia_poi format:

URL	https://api.navitia.io/v1/coverage/fr-idf/poi_types/poi_type%3Aamenity%3Ahospital/pois?

Using playground, we can visualize the distribution of the 380 hospitals in the Île-de-France region by selecting the category: amenity=hospital.

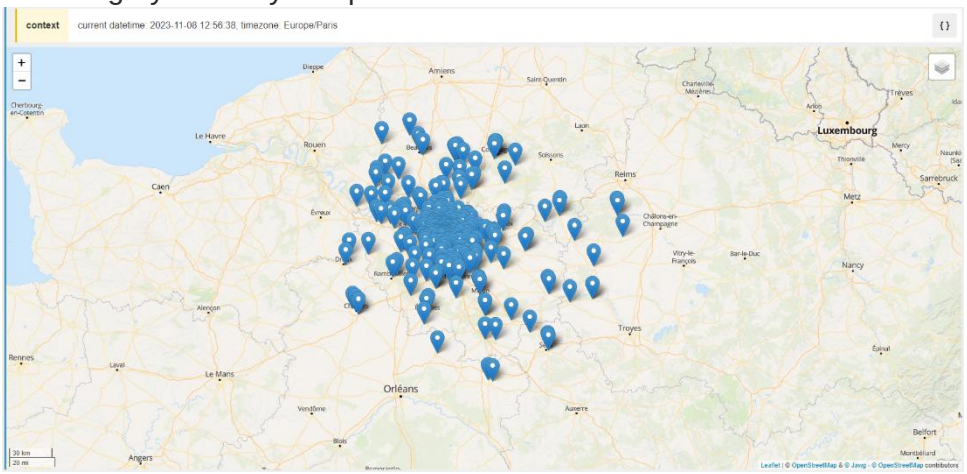

Figure 22: Hospital POIs in tje Île-de-France region

In “properties”, we can also find “accessibility” and other useful pieces of information about the pois.

```
{
    "address": {5 items}
    "administrative_regions": [1 item],
    "coord": {
        "lat": "48.9409347"
        "lon": "2.8730666",
    },
    "id": "poi:osm:node:1744066136",
    "label": "Clinique Saint-Faron (Mareuil-lès-Meaux)",
    "name": "Clinique Saint-Faron",
    "poi_type": {
        "id": "poi_type:amenity:hospital",
        "name": "Hôpital"
    },
    "properties": {
        "amenity": "hospital",
        "healthcare": "hospital",
        "name": "Clinique Saint-Faron",
        "ref:FR:FINESS": "770813400",
        "type:FR:FINESS": "365"
    },
}
```


Accessibility heatmap using the isochrone API: indication reachable areas within a time frame: The example below is the isochrone from a hospital “Universitair Kinderziekenhuis Koninging Fabiola” in Brussels

URL	https://api.navitia.io/v1/coverage/be/isochrones?from=poi%3Aosm%3Anode%3A493193906&datetime=20231108T130141&boundary_duration%5B%5D=600&boundary_duration%5B%5D=1200&boundary_duration%5B%5D=1800&boundary_duration%5B%5D=2400&boundary_duration%5B%5D=3000&boundary_duration%5B%5D=3600&

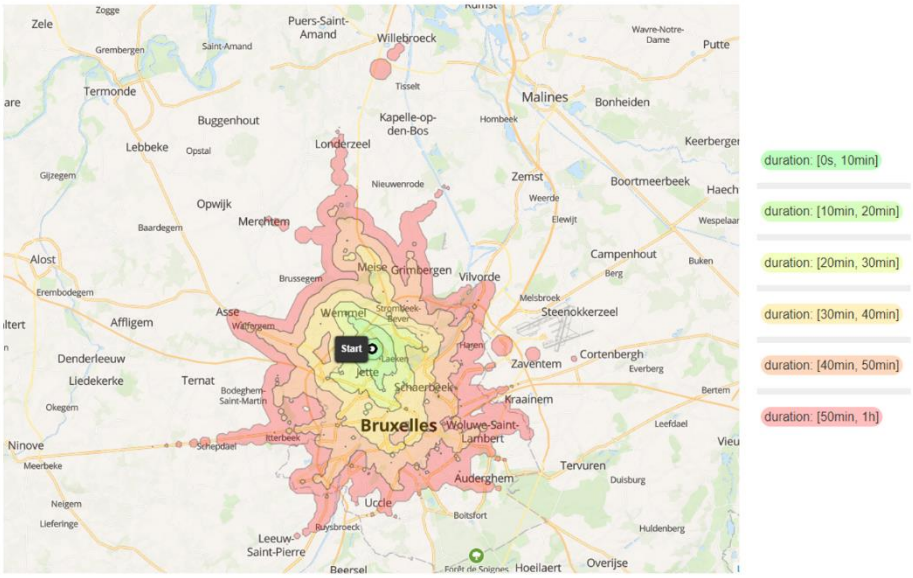

Figure 23: Isochrone map for Bruxelles region

4. Service Access Demonstrator

4.1. Design

To promote discoverability of service API, the MobiDataLab Service Catalogue is developed around the OpenAPI Specification (OAS), an open standard for describing and defining RESTful APIs. It was previously called the Swagger Specification and is currently maintained by the OpenAPI Initiative, which is a group of industry experts and companies. The OAS is a machine-readable format that enables developers to outline the details of an API in a structured and standardized way, including information about the endpoints, operations, parameters, responses, and other relevant data. Since the OAS is widely used by many web services to define their documentations or complement to their documentations, the MobiDataLab Service Catalogue acts as web client that consumes OAS to onboard, index and manage service APIs.

Considering the FAIR principle, the MobiDataLab Service Catalogue are developed as 3 open-source components: the catalogue website, the collection storage of OpenAPI specification document, and the GitHub automation pipeline. Inspired by the openapi-directory project[4], the MobiDataLab Service Catalogue shares the same interfaces and data structures, enabling seamless integration between these two any to any other catalogue, client that consume OAS. The GitHub automation pipeline realizes contribution process and deployment process on top of the GitHub native infrastructure and process, which play an important role in the sustainability and engagement in the open-source and geospatial communities.

In addition, the MobiDataLab Service Catalogue supports indexing of services without OAS, based on the input description and category. Listing of these services serves as a starting point to encourage provision of an OAS from the service contributors for better usability and interoperability.

4.2. Browse and Query for Service API

User can simply browse through the services shown in the webpage. Alternatively, user can type in the search box keywords, for example name service provider, service category, to filter the results.

[4] https://github.com/APIs-guru/openapi-directory

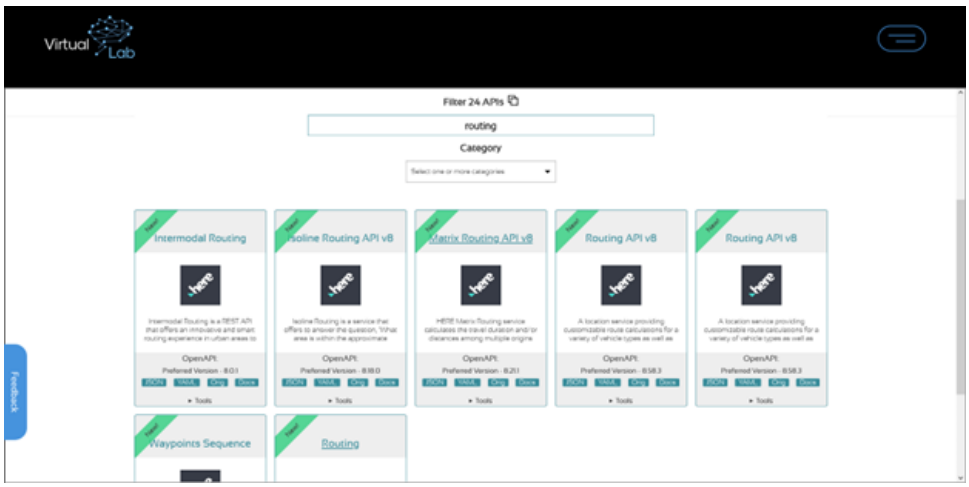

Figure 24: Mobidatalab Service Catalog Website integrated in the Living Labs

In each card item, there are buttons and links with the following functionalities:

- View OpenAPI specification in raw format using the JSON, YAML or Orig buttons.
- View OpenAPI specification in visualized interface, with Docs or Swagger UI button (the submenu Tools needs to be expanded to show these buttons)
- Edit OpenAPI specification with buttons Swagger Editor or OpenAPI GUI. The changes are not saved into the catalogue. To submit these changes, please create a GitHub issue or Pull Request for review.
- View reference to APIs, whose OAS not provided or available.

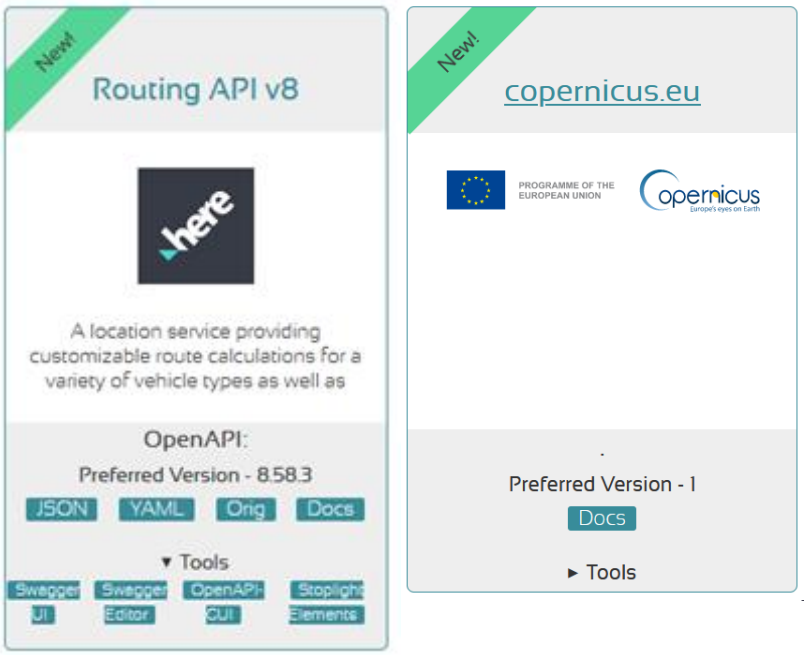

Figure 25: Service with OAS (left) and without OAS (right) shown in the Mobidatalab Service Catalog Website

The API specification can be viewed by any supported web client or desktop clients, namely Swagger UI, redoc, Postman, etc.

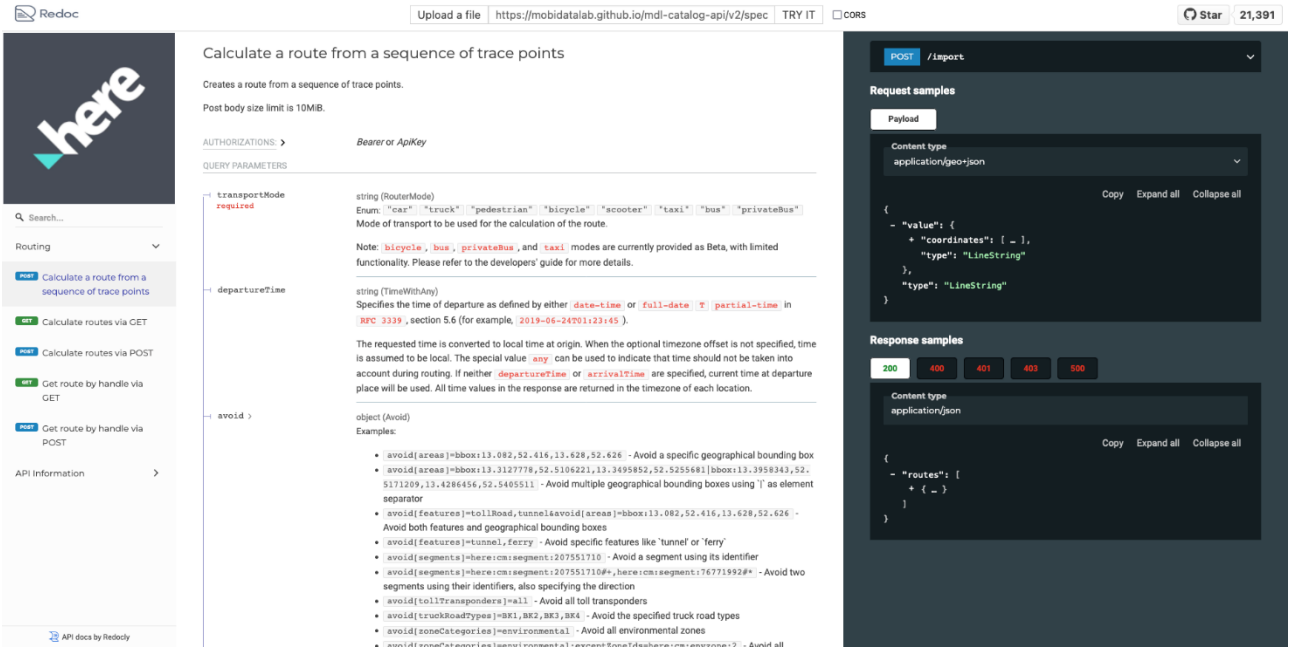

Figure 26: OpenAPI specification shown in redoc web interface

4.3. Test and use the Service API

OpenAPI compliant service API and endpoints can be executed directly in some web clients like Swagger UI, all desktop clients and can be integrated seamlessly into almost every programming language. As sending requests to service endpoints from Swagger UI hosted in third party website is no longer supported in many modern service implementations due to security implication related Cross-Origin Resource Sharing, our recommendation for trying out service endpoints is to use the desktop client Postman[5], an API platform for building and using APIs. This section demonstrates how an API specification document can be downloaded from MobiDataLab Service Catalogue Website and imported into Postman for further exploration and development.

The API specification document can be downloaded either as JSON or YAML file, using the button shown in Figure 25. In the Postman application, the specification documents can be imported via the menu button, then fill in the URL to the JSON or YAML file (Figure 27). The specification will then be imported into Postman and available for browsing and properly sending requests to the service API (Figure 28). Thanks to high interoperability of OAS, the request made in Postman can easily be transformed into different programming languages, via the button “Code” directly in Postman (Figure 29).

[5] https://www.postman.com/

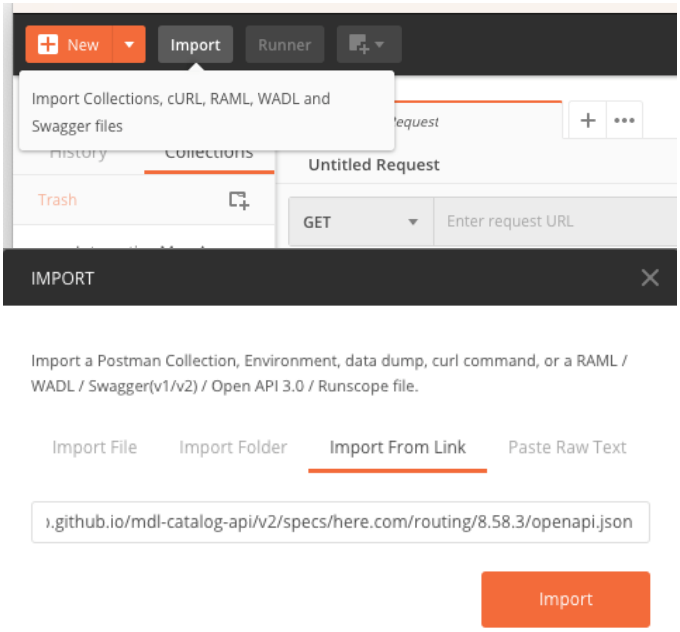

Figure 27: Postman interface to import OpenAPI specification document

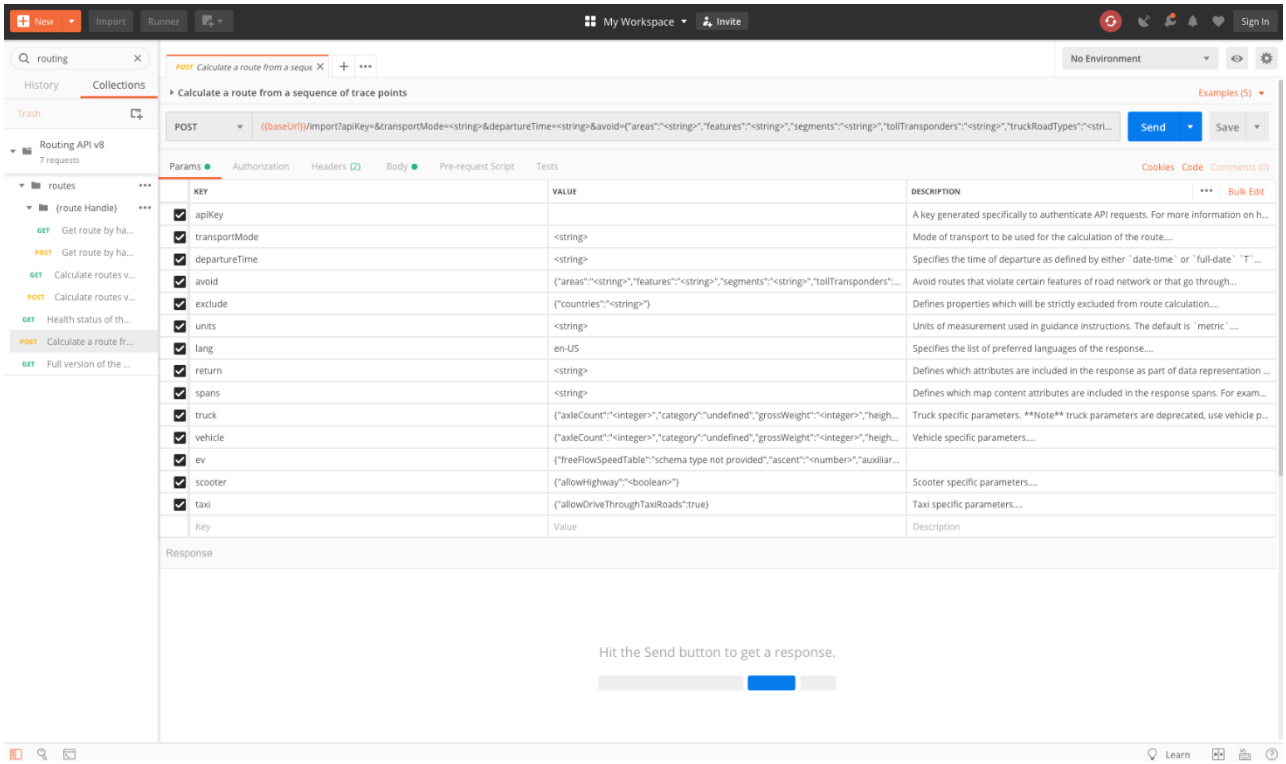

Figure 28: Postman interface to show OpenAPI endpoints specification

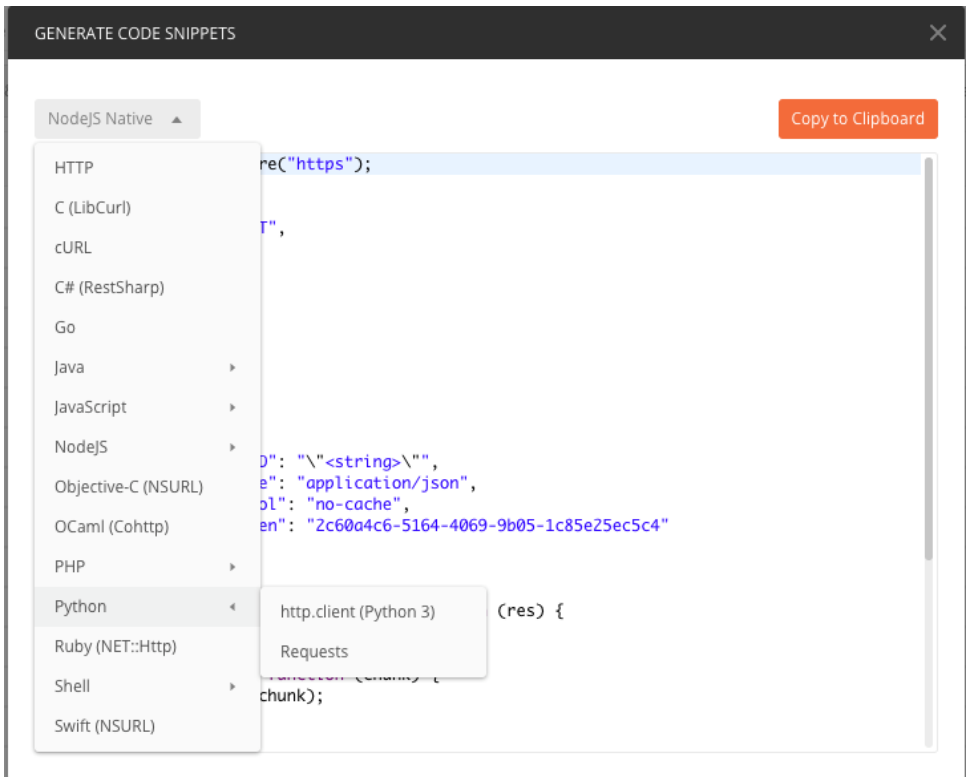

Figure 29: Postman interface to generate code from an OpenAPI endpoints

4.4. Add new Service API

Adding new API into the catalogue can be done by clicking the Add API menu in top bar and fill in the form.

Once done, clicking the Add API button at the end will open a GitHub page where user can create a GitHub Issue with all the required information.

The submitted GitHub Issue will be processed, reviewed and will reference the changes in the catalogue.

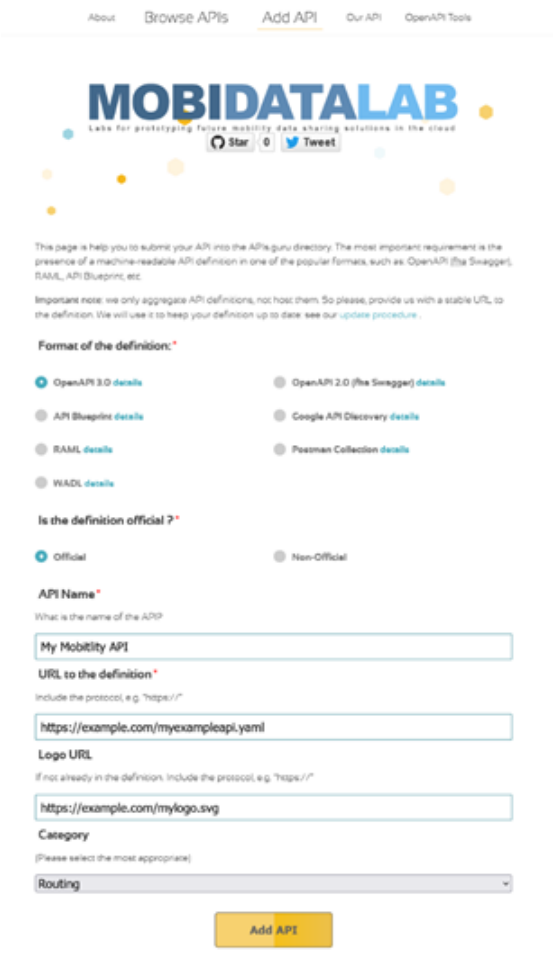

Figure 30: Add new Service API from MobiDataLab Service Catalog Website

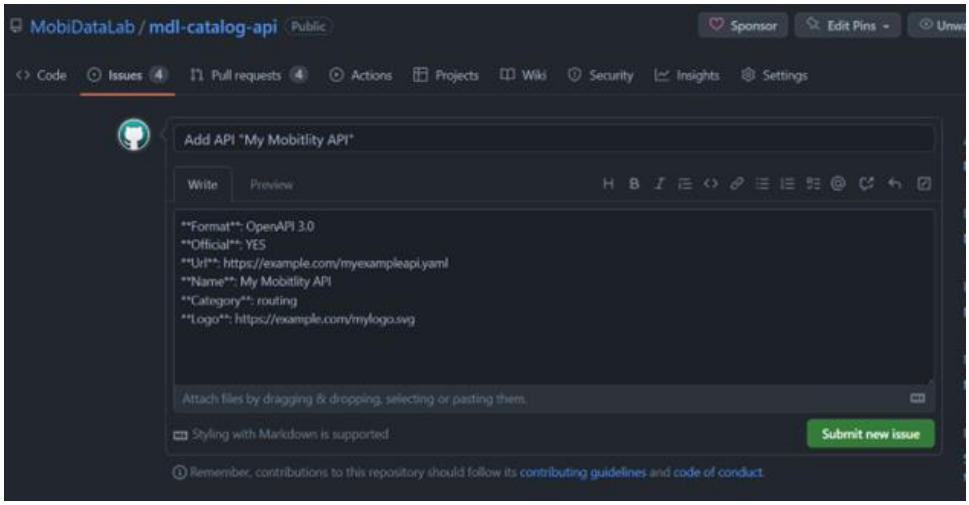

Figure 31: GitHub issue referencing new service request

4.5. Contribution Process

The contribution process of the services in the catalog is designed on top of GitHub automation and open-source contribution process, onboards and validates contributed OpenAPI specifications with minimal input from human actors, increasing service discoverability and accessibility. An overview diagram of the process is demonstrated in Figure 32.

The automated contribution process begins when a user, contributor create a request for new Service API from the catalogue website. This action would create a GitHub issue and trigger the GitHub Action pipeline, which tries to add the specification document of the new API in the following steps:

1. Download resources specified in the GitHub issue: API specification document, logo URL.
2. Validate the API specification and fix syntax errors if applicable.
3. Create a Git branch corresponding to the issue and commit the changes.
4. Create a Git Pull Request (PR) to merge the branch into main branch.

After the PR is created, the added API specification can be reviewed, adjusted if required and approved by human reviewers. Once approved, the PR with the new API will be merged into main branch, and another Gitlab Action pipeline will start automated deployment of the new changes to the service catalogue website.

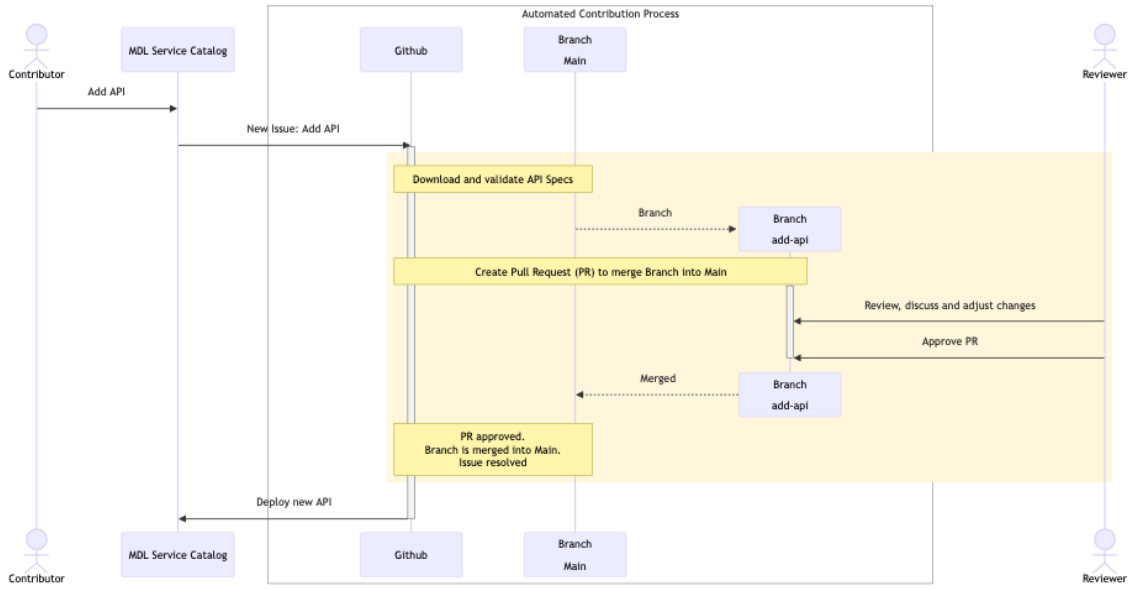

Figure 32: Diagram of contribution process of MobiDataLab Service Catalog

5. Data and Services in the Living Labs

The described data- and service catalogues and the tools to access them have been introduced in the living labs and continuously improved. For the relevant changes, the communication with the users provided an important feedback channel to adjust and improve the tools. Further, in the course of the project, new tools and technologies have been explored and introduced.
The labs did provide a practical perspective from the end-users. Within the pre-organized webinars, the tools and services have been introduced and described. This has resulted in a more user-friendly tool stack, awareness for a broad set of technologies and potential connections between the heterogeneous set of participants from challenge providers and x-athon attendees.

6. Conclusions

Within the D4.6 V2, an update of the V1 document and demonstrator is given. The idea to integrate the user feedback throughout the project and split of the deliverable in two versions was very useful. With the user interaction through living labs and the possibility to update the tools and services within the project lifetime, it is possible to deliver an up-to-date environment. This is important, since the project results will be an important trigger and a seeding for the further development of mobility data sharing. The usage of open-source components and the development and provisioning of relevant services for interaction with mobility data help to enable successor projects and influence the ongoing product development in industry.

7. Future Work and post Project Potentials

The developed services, service and data catalogues and further compontents of the project are great demonstrations how collaborative work results in aligned understanding of an ecosystem and its assets and interfaces. However, the mobility ecosystem is extremely divers and the MobiDataLab project can only be a seeding for further initiatives, that connect and build on top of the shared knowledge and developments. Software and services are based to large parts on open source software. The architecture and many components are provided within the open source repository of the project. The probably most sustainable part of the project is, and will be the mindset, that has been shared from the beginning on and which has been been grown to a exemplantory path for enabling mobility data sharing. With the wide set of project stakeholders – project members, reference group, living lab participants, conference audiences, partnering projects, … - the tools and services have the potential and the possibility to be reused and grow to further enable mobility data sharing. This will be realized within the strong network of research partnerships, industries, start-ups and individuals, which has been build from the MobiDataLab project and leveraged through the provisioning of software under an open source license and the sharing via the reports and the open knowledge base.

8. Annexes

8.1. QGIS Plugins to explore

Flemish geoportal : geopunt4QGIS
GeoPortal.rlp Metadata Search
GTFS Loader plugin
Opendatasoft plugin
Here Route API Plugin
Travel Time
Mobility Areas
PosiView
GBFS-NOW
QGIS Cloud Plugin
Networks
OpenTripPlanner
Tempus
MapTiler
QuickOSM
OSMDowloader
Pelias Geocoding
Valhalla
OSM place search
QBan(o)
AdressesFr
Google Maps geocoder
OS search for addresses
OS Translator II
GeoPortal.rlp Metadata Search
Qweather
HERE Maps for QGIS: Connect QGIS to your personal space on HERE Data Hub and to your Interactive Map Layer inside the HERE Platform.
Greek Data

Funded by the European Union

MobiDataLab consortium

The consortium of MobiDataLab consists of 10 partners with multidisciplinary and complementary competencies. This includes leading universities, networks and industry sector specialists.

@MobiDataLab
#MobiDataLab

https://www.linkedin.com/company/mobidatalab

For further information please visit **www.mobidatalab.eu**

MobiDataLab is co-funded by the EU under the H2020 Research and Innovation Programme (grant agreement No 101006879).

The content of this document reflects solely the views of its authors. The European Commission is not liable for any use that may be made of the information contained therein. The MobiDataLab consortium members shall have no liability for damages of any kind that may result from the use of these materials.