Semantic Technologies and Linked Data – A.Y. 2019/2020 – June 2020

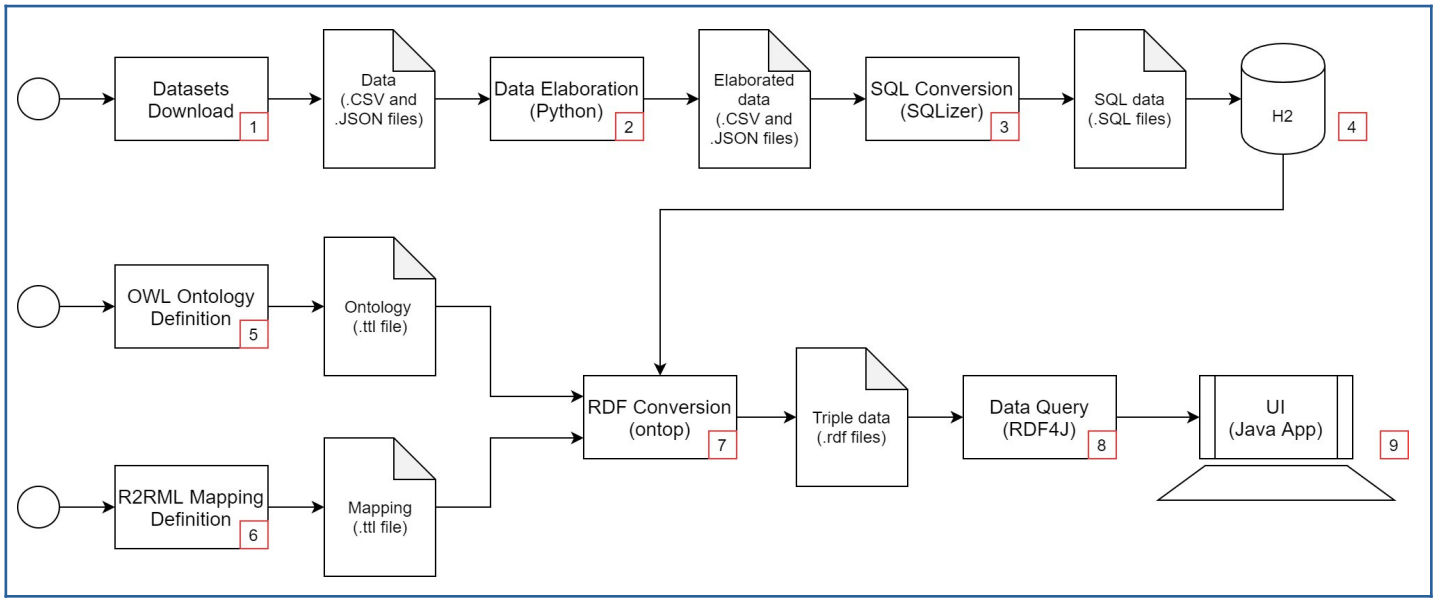
“Explore Zurich”: Project Report

Silvia Fracalossi – Student ID: 13180

“Explore Zurich” is an offline application to discover the tourist attractions of Zurich. Without indicating a specific location, the user can look for bars, restaurants, museums, shops and tourist attractions by district. Furthermore, a brief description of the points of interest is displayed, along with name, opening hours and address. Moreover, public transportation data is included, allowing the user to discover the location of the train stations and the stops offered by the city. The information are visualized in a map in the homepage, which contains also a filter panel on the side, to indicate which district and what types of facilities to display.

Data Preparation

In order to serve the data to the Java application, the pipeline shown in the picture below was followed.



The main goals of the pipeline were to convert all geographical information in the same coordinate system, split data to better process it, upload data in a H2 database to access it, create mapping and ontology to covert data from table and tree structure to a graph structure. More details on the steps is explained in the following paragraphs.

1. Datasets Download

The datasets are derived from different sources and were downloaded and stored in the folder "resources/data/01_original". Below, a table indicating the entities, along with the website name and the link to the see or download the dataset.

Entity No.	Name	Source Name	Link
1	District	Stadt Zürich	https://opendata.swiss/en/dataset/stadtkreise1
2	Neigbourhood	Wikipedia	https://en.wikipedia.org/wiki/Subdivisions_of_Z%C3%BCrich
3	Bus stop	OpenData.swiss	https://data.stadt-zuerich.ch/dataset/ktzh_haltestellen_des_oeffentlichen_verkehrs_
4	Bus line	OpenData.swiss	"
5	Train station	Transport Open Data	http://transport.opendata.ch/v1/locations?query=Zurich
6	Car parking	OpenData.swiss	https://data.stadt-zuerich.ch/dataset/geo_oeffentlich_zugaengliche_parkhaeuser
7	Bike parking	OpenData.swiss	https://data.stadt-zuerich.ch/dataset/geo_zweiradparkierung
8	Restaurants	Stadt Zurich	https://www.zuerich.com/en/data/gastronomy/restaurants
9	Bars	Stadt Zurich	https://www.zuerich.com/en/data/gastronomy/nightlife/bars+%7C%7C+lounges
10	Museums	Stadt Zurich	https://www.zuerich.com/en/data/place/culture/museums
11	Attractions	Stadt Zurich	https://www.zuerich.com/en/data/place/attractions
12	Shopping	Stadt Zurich	https://www.zuerich.com/en/data/place/shopping

2. Data Elaboration (Python)

One of the first step in the project implementation was the creation of python scripts to elaborate the data. The goals of the code are to define in which district the different entities are located and to covert all the geographical coordinates into a single coordinate system, namely "ESPG:2056". The instances, along with the newly-computed district identification number and coordinates, are stored in .CSV and .JSON files in a different folder, namely "resources/data/02_python_elaborated".

3. SQL conversion

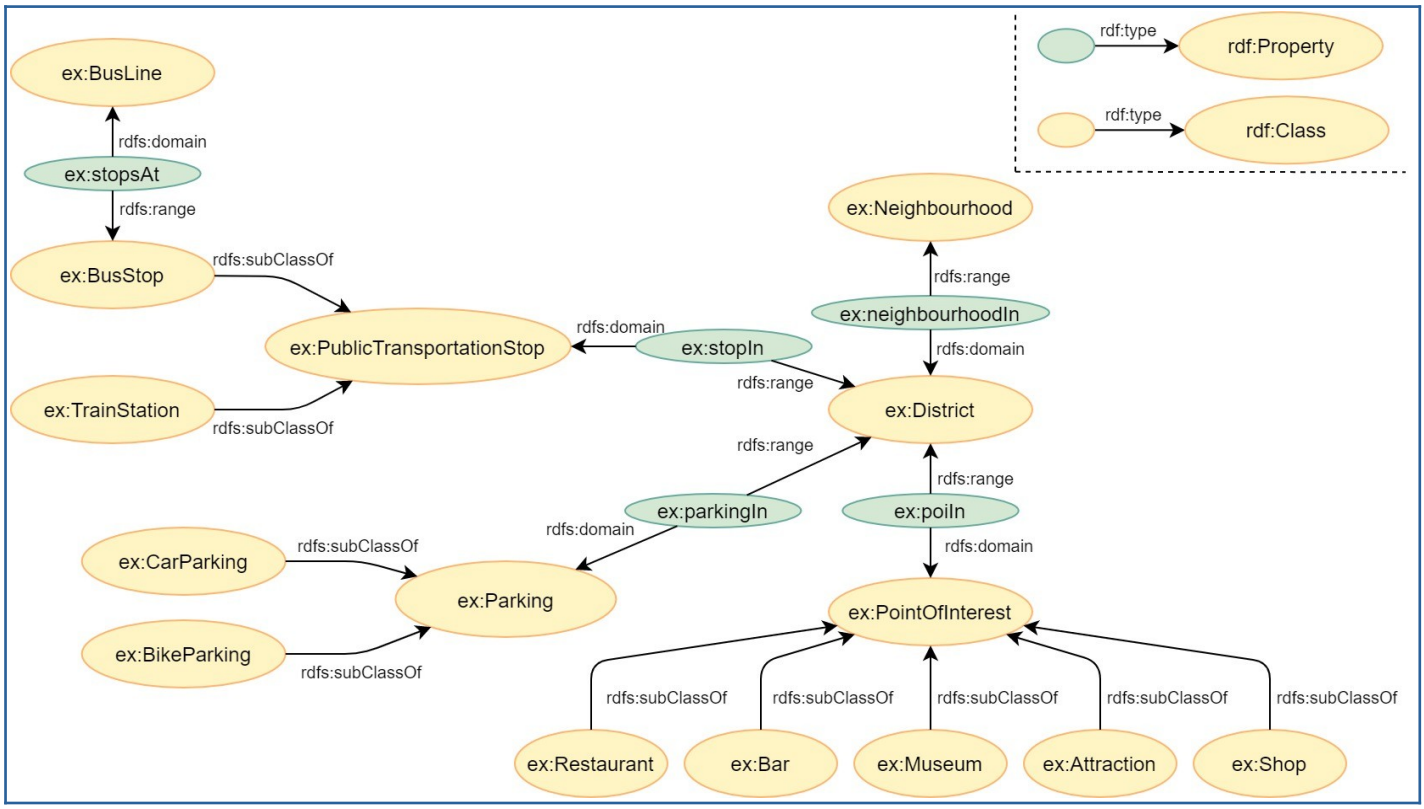
In order to upload the data into the H2 database, the .CSV and .JSON files have to be converted into SQL insertion scripts, and the headers into SQL table creation scripts. To do this, the online tool called SQLizer has been utilized. The link to the website is the following: https://sqlizer.io/#/. The data was then saved in a different folder, called “resources/data/03_sql”.

4. H2 database

An H2 database has been created. The SQL files generated by the previous step have been executed in the DBMS. The database file has been saved in the folder “resources/database”.

5. OWL Ontology Definition

Based on the application idea and on the datasets, the ontology has been defined. The following image shows a graphical representation of the ontology.



The data properties were not represented in the diagram; they include the “:area” property, for the definition of the polygon areas for the districts, and the “:location” property, for the definition of the point location of the other instances types. The final .TTL file has been saved in the folder “resources/ontology”.

6. R2RML Mapping Definition

To define the mappings, R2RML has been utilized as RDB to RDF mapping language. The generated file has been saved into the “resources/mapping” folder.

7. RDF Conversion

In order to convert the SQL tables into triples, a RDF conversion was necessary. To do so, Ontop has been utilized as RDF data processor. The file containing the database location, driver, username and password has been created and saved in the folder “resources/database”. The configuration of the Ontop execution includes the location of the database properties file, the mapping file and the ontology file, plus the name of the output file. The command was stored as “generate_rdf.sh” in the main folder, and the RDF file in the “resources/data/04_RDF” folder.

8. Data Query (RDF4J)

To query the data triples, RDF4J has been chosen as the Java framework for processing and handling RDF data. A file to manage the data requests was created, called “DataLoader”. It is called by the other Java classes in the application, loads the data from the RDF file and contains methods utilizing SPARQL queries to retrieve data. The java file is located in the “src/main/java” folder.

9. UI (Java App)

The User Interface has been implemented in Java. As explained before, it relays on the “DataLoader” file to query the RDF data. In order to handle and display geographical data in a map, the library ArcGIS has been utilized. Moreover, to show graphical components like buttons and labels, the library JavaFX was chosen. More on the appearance of the application can be found in the chapter “User Interface”.

How to use the system

Below, the content of the README file, which can be found in the main folder of the repository.

Requirements – Java Application

- IntelliJ (or any other IDE that support Maven projects)
- Java 11

Requirements – Python Script (not necessary to run the application)

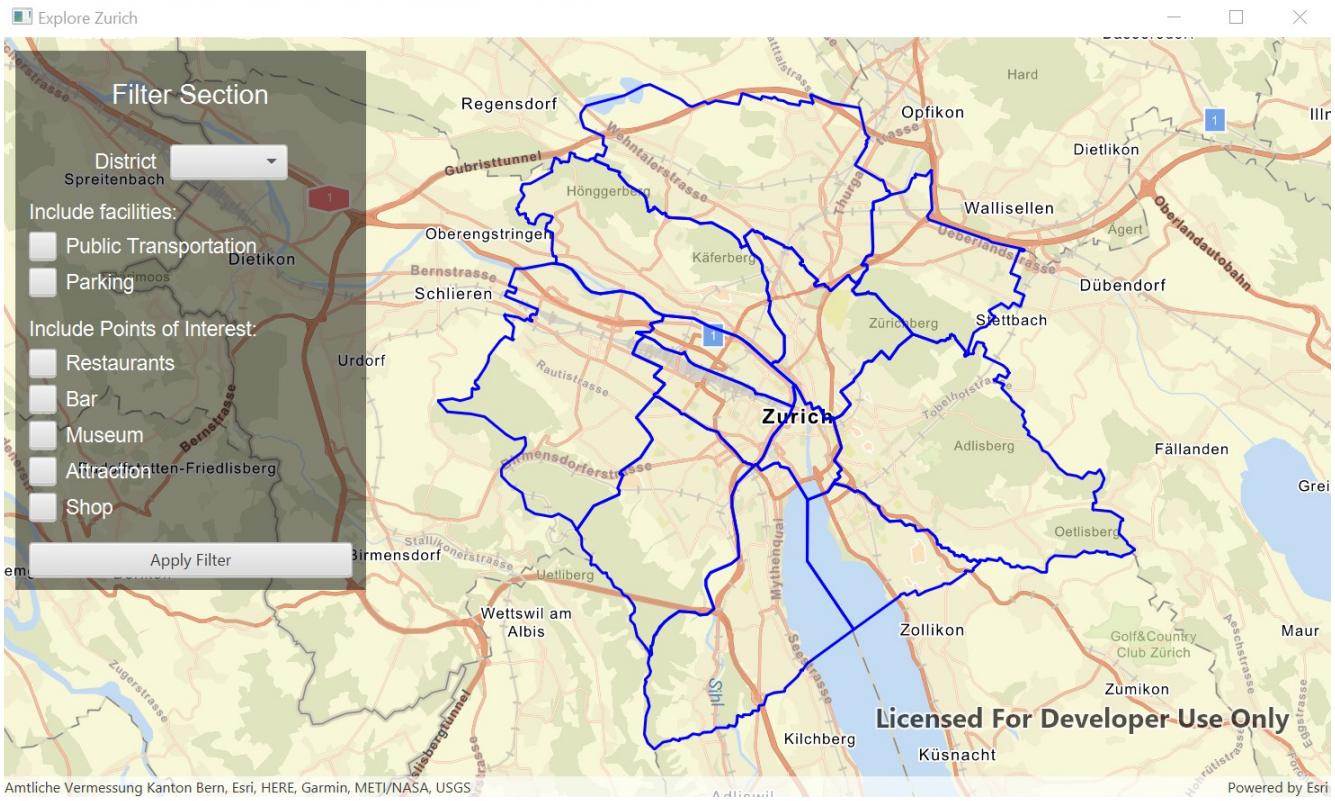
- csv library
- json library
- shapely library
- geopandas library

Installation steps

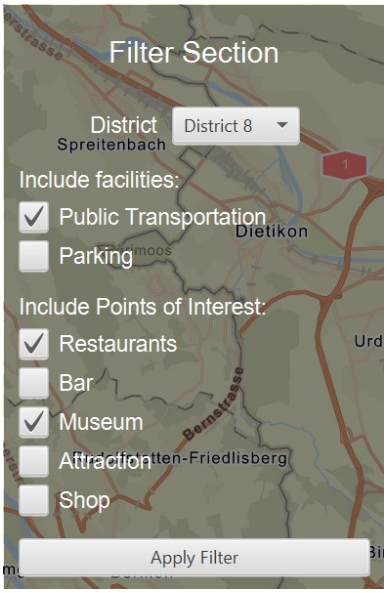
- Open the project in Intellij IDEA
- Select *File* > *Project Structure...* and ensure that the Project SDK and language level are set to use Java 11.
- Open the Maven view with *View* > *Tool Windows* > *Maven*.
- In the Maven view, under *Plugins* > *dependency*, double-click the `dependency:unpack` goal. This will unpack the native libraries into $USER_HOME/.arcgis.
- In the Maven view, run the `compile` phase under *Lifecycle* and then the `exec:java` goal to run the app.
- Run the main method in the src/main/java/Main.java file

User Interface

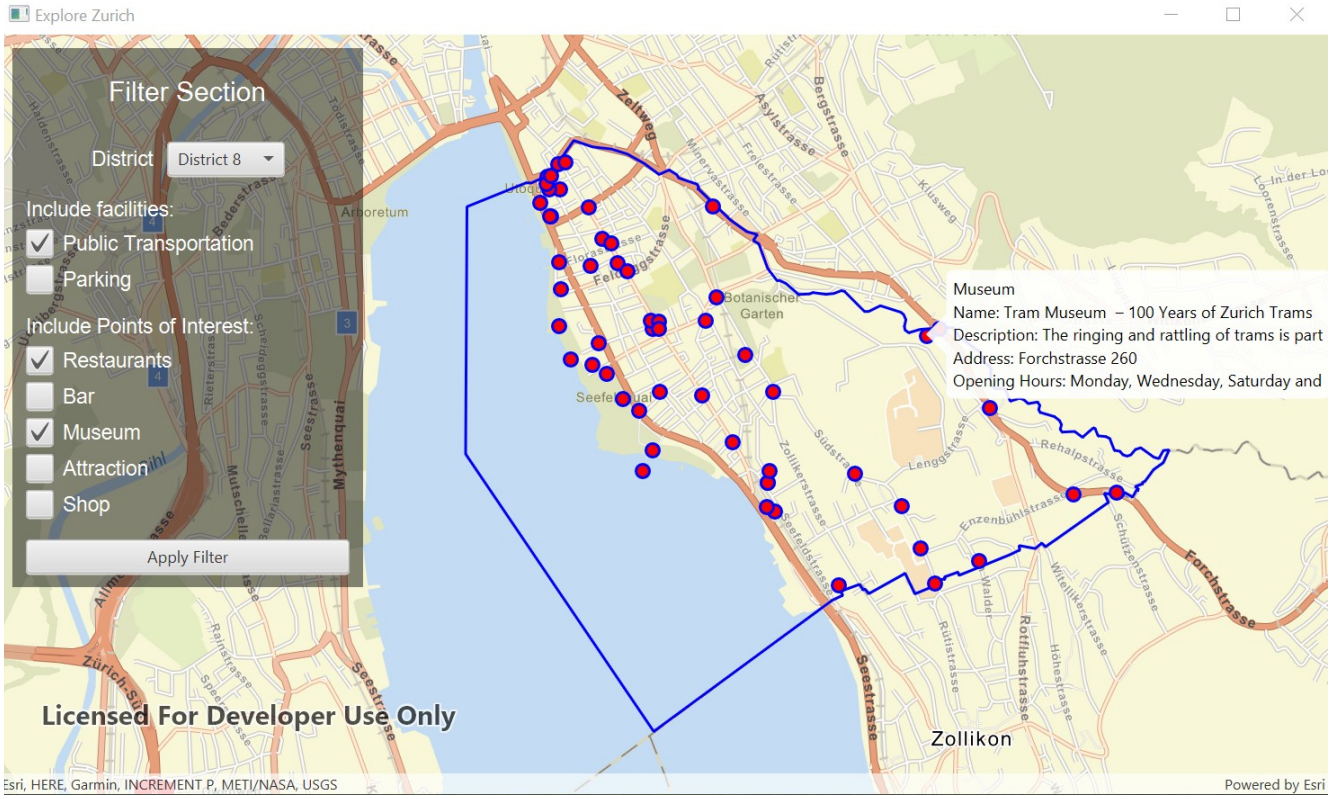
After running the main method in the src/main/java/Main.java file, the application window will be displayed. The window shows the map of Zurich, along with blue polygons, representing the 12 districts of the city. On the left side, there is a panel to filter the facilities and the point of interest. In the image, no facility nor point of interest type is selected; therefore, no data other than the data is displayed.



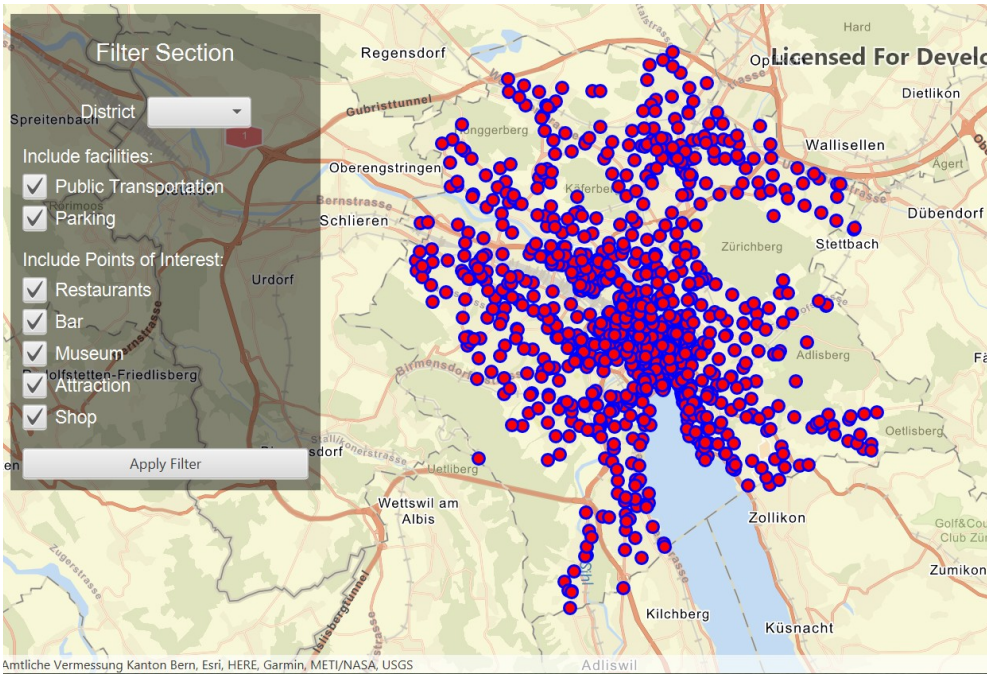
The filter panel contains check boxes and option boxes in order to select the district and to decide whether to display facilities like Public Transportation and Parking spots and the possibility to show Points of Interest, such as Restaurants, Bars, Museums, Attractions, Shops.



After the display request has been entered in the filter section, the user has to click the “Apply Filter” button. The window will close, but another one will immediately appeare, displaying the filtered data. By clicking a point in the map, a pop-up will open, containing information about the selected point. By right-clicking away, the pop up will close.



In order to see all the datapoints of the city, the user has to check all the facilities and points of interest check boxes in the filter section. Moreover, no district has to be selected. The result can be seen in the image below.



Lessons Learned

There are several lessons that I have learned by developing the project. Creating the ontology and the mapping manually allows the developer to truly understand the information that he/she is handling and that will be available to query, removing the risk of saving data that will not find a practical application in the final result. Moreover, querying graph data is different than querying table data, but it is easier to express certain concepts in SPARQL without having to merge multiple tables in the classic DBMS. Furthermore, loading data into the main memory makes the loading faster.

However, there are certain aspects that could have been done in an easier way. For example, it would have been useful if certain steps were automatized, like setting the data elaboration and the conversion into SQL in the same script. Moreover, the online tool “SQLizer” did not support files larger than 5000 lines. Therefore, I had to split the restaurant dataset into 7 files in order to process and convert it.