Semantic Web Project

Public Bicycle Sharing Stations Query System

-

Reinforced by Semantic Web Technology

2019 - 2020

Team members:
Jiawei XU
Malshani RANCHA GODAGE

Supervisor:
Antoine ZIMMERMANN

Contents

1 Introduction

Semantic web technologies are becoming more famous and popular in the field of web development and IoT fields which provide concise solutions for existing problems of the internet and communication world. It enables computers and people to store and access the meaningful information linked together consortiums of standard organizations.

This report explains how we utilize semantic web technologies to solve a problem which copes with day to day lives of current society. This project is discussing transport domain which talks about bicycle sharing stations, some other public and private transport methods, tourism, prices of transport, weather and etc. In the beginning, we talk about bicycle sharing systems and furthermore other mentioned problems.

Bicycle sharing stations provide service to people who want to use bicycles and travel for a while and drop it in any station, but not to keep them. The system is fully automated and data is updated over the network, but there is no one portal to access all the data. Open data of different scenarios are published in various public and private websites separately. They are not linked together. They may be linked as documents, but not as entities. Data on each website follow their own standards, own format and own vocabularies. To reduce the mentioned issue, in this project we are given a system to design and implement using semantic web technologies.

2 Technologies

Basically this application uses Semantic Web technologies starting from extracting open data to displaying them on the website. We use Protege to build our OWL ontology for the domain of bicycle sharing system. Apache Jena is using to create and access RDF triples with model.

Apache Jena Fueski is a server which uses SPARQL. We use Fueski server to persist the data set and SPARQL to query it. We use RDFa when representing data on the website. Since the system is web based we are using web development frameworks and technologies such as SpringBoot, HTML, Maven, AJAX, jQuery and BootStrap.

3 System Design

Following figure illustrates the basic architecture of the project. Flow is depicted in arrows with numbers which described below.

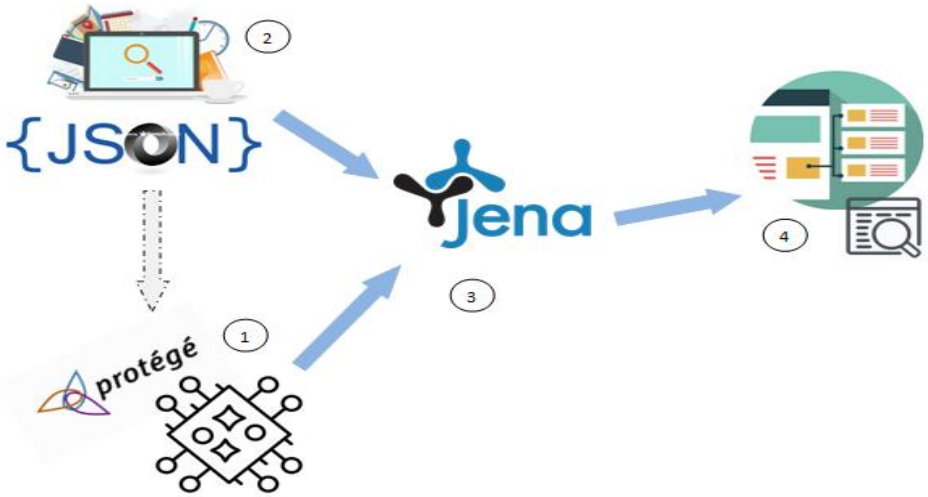


Figure 1: Implementation flow

Item (1) > existing ontologies and ontology we created in Protege by analyzing JSON data of website
Item (2) > the JSON data of static and dynamic web pages
Item (3) > the RDF model is saving in Fueski server which is generated using (1), (2)
Item (4) > the website visualizing the data from Fueski server

When we design the whole project, it consists of three major parts, which will be described in this section.

1. Design the Model Structure: One of the most important parts of this project is to model the scenario. We checked the format and content of data we are going to use such as real time and static data of bicycle sharing stations. We applied knowle
2. Extract data and generate the model: Then we decided how to extract static and dynamic data. After extracting static data, RDF triplets are generated according to model we designed.
3. Visualizing the data: We design this as Website and REST API by providing search options over the data set.

4 Implementation

Two java projects are developed as extractData and bicycleSharingStations (website). Following explains the implementation of two projects.

4.1 Design the model structure

After analyzing the data, we model our scenario. We identified what are the entities and properties of this specific domain. Since, we could not find a suitable domain ontology for the bicycle sharing stations, we proposed and developed and OWL ontology in Protege. Following figure 2 depicts the ontology we created.

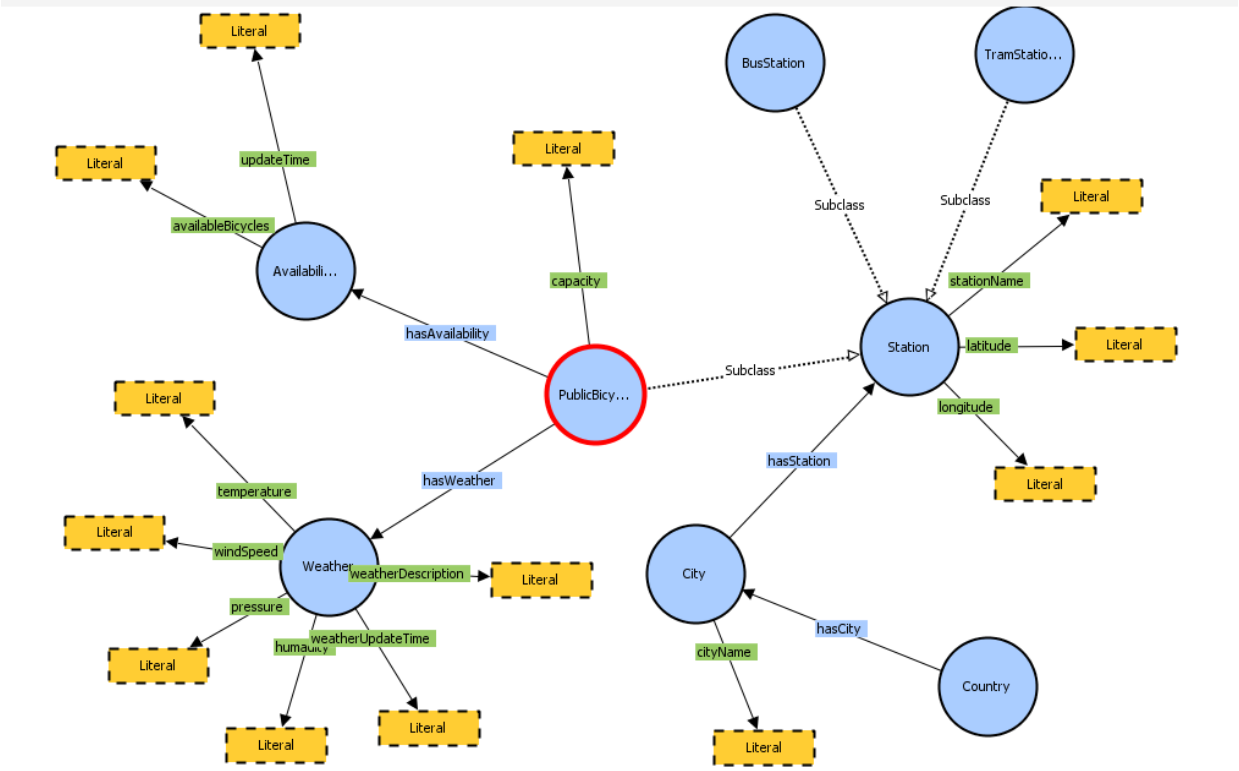


Figure 2: OWL ontology

4.2 Create the model

We extracted open data published in websites given by service providers. All of them (for now three cities) are written in JSON. We developed a program to extract all the static data of selected cities, then create RDF triplets. All the triplets are linked to generate the RDF model according to the Ontology we created. Then save this model in Fueski server. Now we can add dynamic data of stations only running the according program specific to the city. Each time you run the program for a city, these dynamic data will be added to the model. Model has been expanded.

About blank nodes

When the availability of bikes is added, we save the history of data. Each time dynamic data extracting process running, we create a blank node for that and add new nodes for the blank node for available bikes and updated date time.

We have saved history of weather as well. Each time weather process is running we generated new blank node and give identity for each history data.

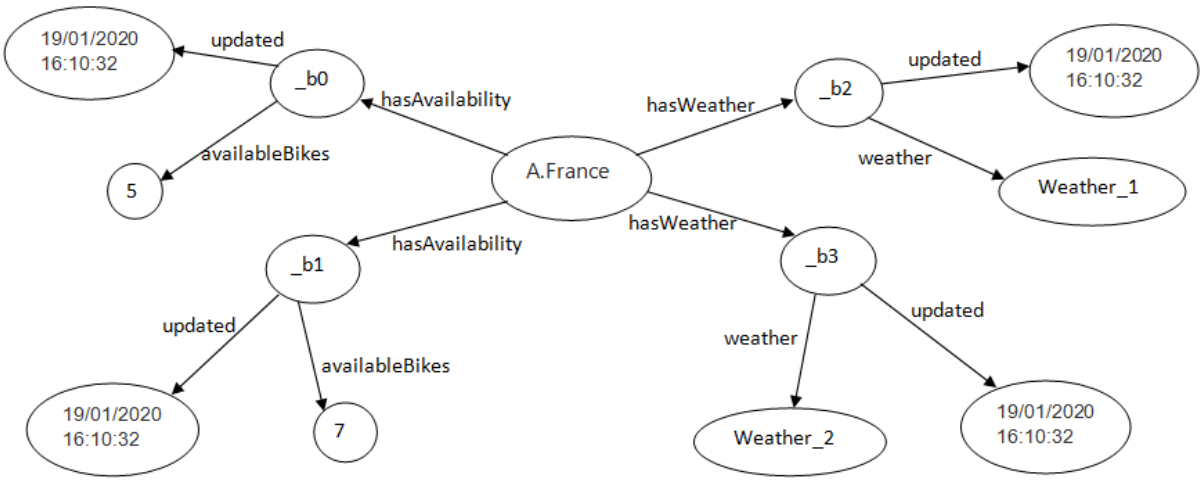


Figure 3: Blank nodes

Static data	URL
Saint Etienne	https://saint-etienne-gbfs.klervi.net/gbfs/en/station_information.json
Lyon with Key (We used this method)	https://api.jcdecaux.com/vls/v1/stations?contract=lyon&&apiKey=b5c059fa1b8e115f157e20cfa797e01b7650f0a7
Lyon	https://download.data.grandlyon.com/ws/grandlyon/pvo_patrimoine_voirie.pvostationvelov/all.json?maxfeatures=100&start=1
Paris	https://opendata.paris.fr/api/records/1.0/search/?dataset=velib-emplacement-des-stations
Nantes	https://api.jcdecaux.com/vls/v1/stations?contract=nantes&&apiKey=b5c059fa1b8e115f157e20cfa797e01b7650f0a7
Toulouse	https://data.toulouse-metropole.fr/api/records/1.0/search/?dataset=velo-toulouse&rows=1000
Dynamic data	URL
Saint Etienne	https://saint-etienne-gbfs.klervi.net/gbfs/en/station_status.json
Lyon	https://download.data.grandlyon.com/wfs/rdata?SERVICE=WFS&VERSION=1.1.0&outputformat=GEOJSON&request=GetFeature&typename=jcd_jcdecaux.jcdvelov&SRSNAME=urn:ogc:def:crs:EPSG::4171
Nantes	https://api.jcdecaux.com/vls/v1/stations?contract=nantes&&apiKey=b5c059fa1b8e115f157e20cfa797e01b7650f0a7
Toulouse	https://transport.data.gouv.fr/gbfs/toulouse/station_status.json

Table 1: Data extract URL

Table 1 has the URLs of API we used to extract data. To access some stations APIs and weather API we have generated API keys.

4.3 Display on a web site

To display the data on the web site, we created SpringBoot web application. it has a dropdown to select a city, then user press the search button, AJAX request is sent REST API, there we have a service when the city is given, choose the relevant information such as stationName, latitude, longitude, capacityOfBicycles, availableBicycles.

4 Planning of realization

We have two iterations. By the first iteration we have extracted bicycle sharing stations data to generate the RDF triples. We have created the model and save in as a data set in the fueski server.

By the final submission, we are going to update the ontology. Because the ontology we created is basic. Even though the ontology is not validated by domain experts, we are going to use and develop the ontology for the better use of RDF graph we generated.

5 How to run the project

Two projects have to compile and run. One project is to create a triple store and second one is for display the data. To create the Data store,

1. Run Fueski server
2. Add new data store, name: bicycle_stations
3. Open the project extractData and open the package, complete (extractdata\src\main\java\complete)
4. Run 'CreateModel.java' program to extract static data of SAINT-ETIENNE, LYON, TOULOUSE and NANTES cities, Wait until the process completes, it gives a success message on console.
5. Run DynamicSaintEtienne.java, DynamicLyon.java, DynamicToulouse.java and DynamicNantes.java programs to extract dynamic data. Wait until the process completes. Each step will give success message on console.

Now you have extracted necessary data and saved RDF triplet and Fueski triple store. If you see the http://localhost:3030/bicycle_stations, you can see the data. Web project should be working properly now. Web project is developed using Spring boot and Maven. First enter command *mvn clean install*, then run the project with *mvn spring-boot:run* command. Now browse the http://localhost:8080 to see the web site. Select a city in dropdown list, you can see the data.

6 Discussion

When we add the language tag for literal, it gives us issues when querying the result, due to the limited time, we are not going to fix this issue. We use literals without language tag.

In the Flueski data set, we have saved the availability history of stations and weather history of each station. Following table depicts how we divided the workload. We design the project as the agreement of both team members. We did pair programming but one of us is responsible for each task.

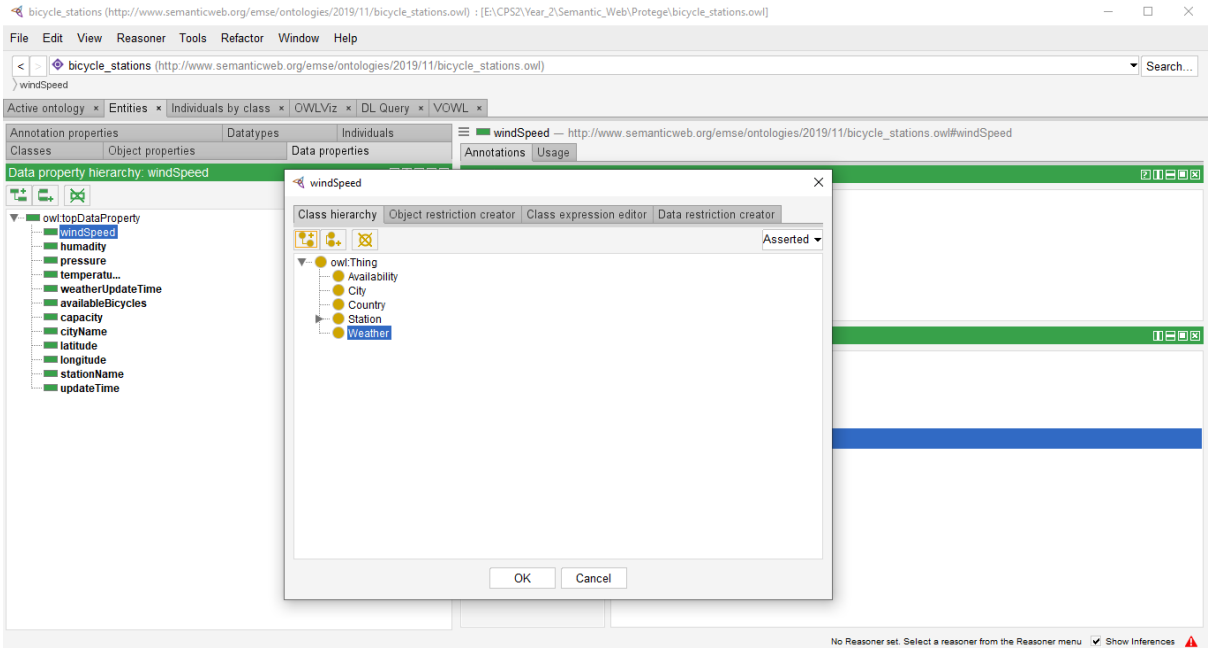
Task	Responsible member
Design the model	Jiawei XU, Malshani R.G
Generate OWL ontology in Protege	Jiawei XU
Extract Static data, create the model and save it in the server	Jiawei XU
Extract Dynamic data, create statement and save it in the server as update the model	Malshani R. G
Extract weather data and added to Model	Jiawei XU, Malshani RG
Setup website and REST API	Malshani R. G
Add RDFa in HTML	Malshani R. G
Testing	Jiawei XU, Malshani R.G

7 References

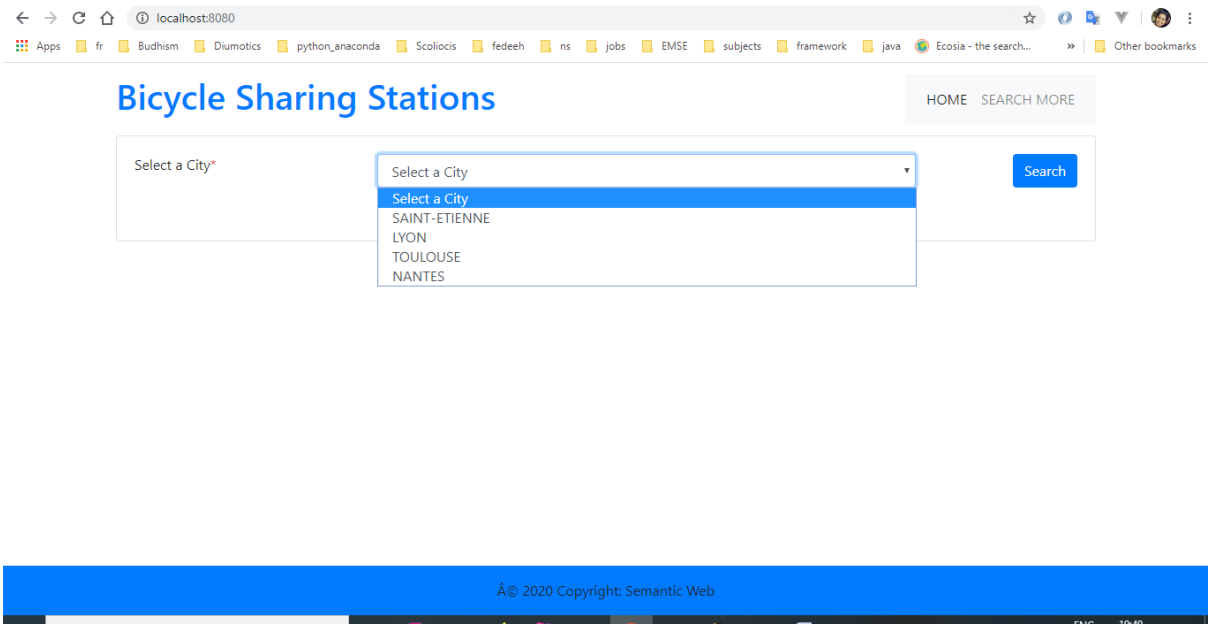
[1] https://www.w3.org/TR/rdf-sparql-query/
[2] https://spring.io/guides/gs/spring-boot/
[3] https://jena.apache.org/tutorials/rdf_api.html
[4] https://spring.io/guides/gs/rest-service/

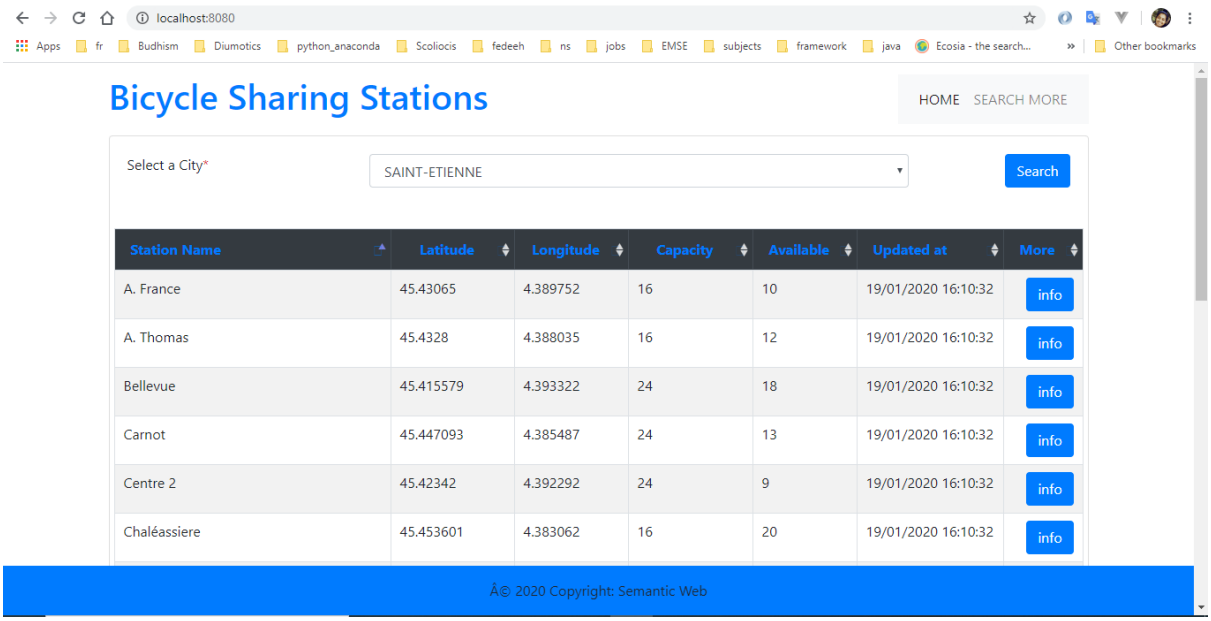
Appendix

1)OWL model implementation in Protege.



2)Web site for search bicycles




localhost:8080
Apps
fr
Budhism
Diumotics
python_anaconda
Scoliocis
fedeeh
ns
jobs
EMSE
subjects
framework
java
Ecosia - the search...
Other bookmarks
Bicycle Sharing Stations
HOME
SEARCH MORE
Select a City*
SAINT-ETIENNE
Search
Station Name
Latitude
Longitude
Capacity
Available
Updated at
More
A. France 45.43065 4.389752 16 10 19/01/2020 16:10:32 info
A. Thomas 45.4328 4.388035 16 12 19/01/2020 16:10:32 info
Bellevue 45.415579 4.393322 24 18 19/01/2020 16:10:32 info
Carnot 45.447093 4.385487 24 13 19/01/2020 16:10:32 info
Centre 2 45.42342 4.392292 24 9 19/01/2020 16:10:32 info
Chaléassiere 45.453601 4.383062 16 20 19/01/2020 16:10:32 info
Â© 2020 Copyright: Semantic Web