Resume of Semantic Web Project

Céline BUFFAT and Sébastien RICO, 13 December 2019
M2 DSC University Jean Monnet

GitHub link : https://github.com/SebastienRico/SemWeb

Introduction

In the context of the Master 2 DSC of the University Jean-Monnet in Saint-Etienne, we had to create a web application that gives information about bike stations using web semantic.

About the application.

Highlights

- Inserting data into Fuseki

To insert the data into our database (we chose Fuseki), we used Sparql-Generate. We learned how to iterate on a json or xml source. We also modified the name of the city to have a capital letter at the beginning. The reason why is that the url we chose to represent the city is a dbpedia url and it needed a uppercase to work. The sparql requests are located in the folder *SPARQLGenerateTurtle* located in the *resources* folder.

- Sparql request to Fuseki

Once we had our data into Fuseki we were able to do some request from our application. Some request were basic like finding all the cities but some were more complicated.

- Search bar:

The idea behind the request for the search bar was to retrieve all the stations that contains the string the user searched. To do this we used a filter with a regex. We request the base in *StationDAO* with the method *findStation.*

- Search Station near me:

The first problem was to locate the user, we used a CURL request and managed to retreive the approximate location. Then in the request, we used some math xpath function to calculate the distance between the stations and the location. Then we order by this distance and take only the first ten. We request the base in *StationDAO* with the method *findStationNearMe.*

All of our web pages contains RDFa. For the *city.html* page we used blank node to represent the cities, each one has a type (from wikidata the entity city) and a label the name of the city.

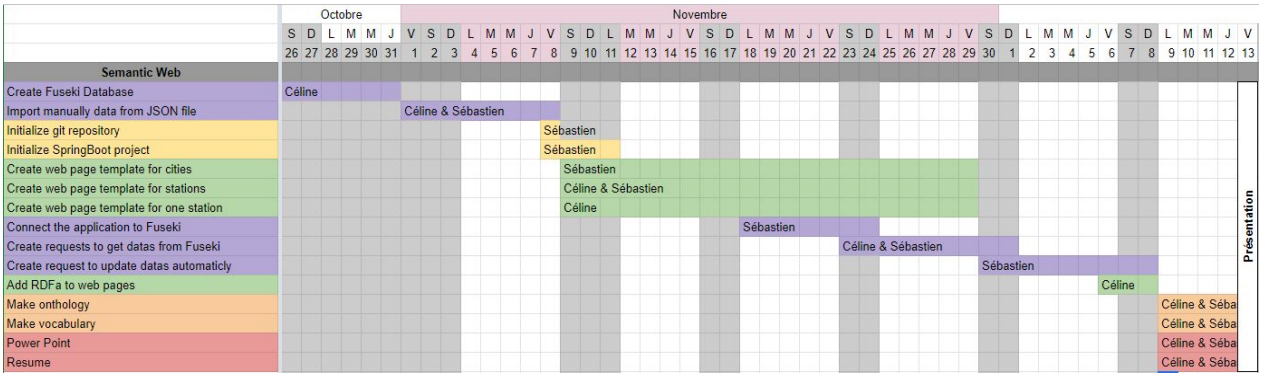
Limitations

Our application could be improved as it is hard to add a new city to it. To do this we could for example use a csv configuration file to store the city and their url as well as the translation from the data of the source to the generic data we use (for instance, say that

‘id_station’ in the source correspond to the generic ‘id’). The user just have to fill a line with a new city and the application will read it and then generate the new city. It could also be done with a form directly on the website. We could parse the result of the URL and ask the user which node corresponds to which station’s property.

Managing of the project

We tried to split the project in two equal parts. At first, Sébastien was working on the project more than Céline. After the holidays of October, Céline had more time to work on the project and Sebastien less. Either way, we always tried to know what the other was doing. We did the request with Sparql-Generate together so that we could both understand it. We both worked on every part of the project at some point, either when implementing or debugging the program. We tried to work in the same place as much as possible so that we could help each other easily.



Ontologie