Université de Genève

Technologies du web sémantiques
D400009

TP 1: Enriched Trails

Author: Sajaendra Thevamanoharan

E-mail: sajaendra.thevamanoharan@etu.unige.ch

Author: Deniz Sungurtekin

E-mail: deniz.sungurtekin@etu.unige.ch

31 Octobre 2020

UNIVERSITÉ
DE GENÈVE
FACULTÉ DES SCIENCES
Département d'informatique

Introdution

Le but de ce TP est de créer une application qui permet d'afficher les points d'intérets sur une trajectoire donnée. Ce TP nous permettra de nous familiariser avec les graphes RDF et leur utilisation dans d'autres interfaces via des API. Premièrement, nous allons présenter les points importants de ce TP, puis dans une deuxième partie, nous allons voir l'implémentation de cette application en utilisant les ressources que nous avons à notre disposition. Finalement, nous décrirons les difficultés rencontrées, notamment avec l'api dbpedia et le filtrage de nos données osm.

Présentation des ressources

GPX

GPX(GPS eXchange Format) est un format de fichiers permettant l'échange de coordonnées GPS. Ce format permet de décrire une collection de points utilisables sous forme de point de cheminement (waypoint), trace (track) ou itinéraire (route). Le fichier GPX est en fait un fichier xml qui contient des informations importantes sur une trajectoire.

Elle se présente de la manière suivante.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gpx version="1.1" creator="Visorando" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.topografix.com/GPX/1/1"
    xsi:schemaLocation="http://www.topografix.com/GPX/1/1 http://www.topografix.com/GPX/1/1/gpx.xsd">
 <metadata>
  <name>Refuge Maison Vieille – Refuge Bertone</name>
  <link href="https://www.visorando.com/randonnee-refuge-maison-vieille-refuge-bertone/">
   <text>https://www.visorando.com/randonnee-refuge-maison-vieille-refuge-bertone/</text>
  </link>
 </metadata>

 <wpt lat="45.790956" lon="6.931023">
  <ele>1954</ele>
  <time>2019-10-23T13:46:57+02:00</time>
  <name>Refuge de Maison Vieille</name>
 </wpt>
    ...
 <trk>
  <name>Refuge Maison Vieille – Refuge Bertone</name>
  <trkseg>
   <trkpt lat="45.790956" lon="6.931023">
    <ele>1954</ele>
    <time>2019-10-23T13:46:57+02:00</time>
   </trkpt>
   <trkpt lat="45.790777" lon="6.932203">
    <ele>1950</ele>
    <time>2019-10-23T13:47:02+02:00</time>
   </trkpt>
```

Figure 1: Un fichier .gpx

Elle est composée des éléments suivants :

- <wpt>, qui représente un point de cheminement (ou les points importants) d'une trajectoire. Elle est précisée, dans notre cas, par *latitude, longitude, elevation , nom*
- <trk>, la trace qui contient dans l'ordre tous les points appelés *trkpt* décrivant une trajectoire.
- <trkpt> , une liste ordonnée de point de trace (<trkpt>) dont le format est identique à la balise <wpt> décrite précédemment

Dans ce TP, nous allons utiliser ces trois éléments pour décrire et interpréter une trajectoire. Les appareils GPS utilisent le format GPX comme le ou l'un des formats d'échange avec d'autres applications ou d'autres appareils GPS, en import ou en export. Le format GPX étant un format XML, il est possible de visualiser et de modifier le fichier avec un éditeur de texte. L'utilisation d'un éditeur syntaxique compatible XML (comme Notepad++ avec le plugin XML Tools) permet de s'assurer que le fichier GPX reste valide sur le plan de la syntaxe et réduit le risque de produire un fichier illisible par les applications/appareils.

Une interprétation visuelle de ce fichier GPX est:

Figure 2: Visualisation d'un fichier GPX

Les points rouges représentent les waypoint ($< wpt >$) et la ligne rouge représente la trace ($< trk >$) qui contient l'ensemble des points ($< trkpt >$) de cette trace.

Rappelons que le but de ce TP est de rechercher les points d'intéret importants sur une trajectoire. Il nous faut donc une interface qui retourne ces informations. On présente donc par la suite *OSM : Open Street Map*.

Open Street MAP

OpenStreetMap (OSM) est un projet collaboratif de cartographie en ligne qui vise à constituer une base de données géographiques libre du monde (permettant par exemple de créer des cartes sous licence libre), en utilisant le système GPS et d'autres données libres.

Elle possède une API (interface de programmation) qui permet de demander des informations sur un espace rectangulaire du globe de la manière suivante :

https://api.openstreetmap.org/api/0.6/map?bbox=**longitude-min,latitude-min,longitued-max,latitude-max**

bbox est en effet une zone rectangulaire avec laquelle OSM API va retourner un fichier (*.osm*) qui contient tous les éléments qui sont dans cette zone.

Le problème c'est que le fichier retourne quasiment tout ce qui se trouve dans cette zone (arbre, station de métro, maison, église, restaurant ...). Il est donc important de notre part de filtrer les informations et de constituer une base de données qui contient uniquement les informations qui nous intéressent.

Le fichier (*.osm*) est un fichier (*.xml*) qui se représentent de la manière suivante.

```
<node id="528023413" visible="true" version="2" changeset="5650345" timestamp="2010-08-31T23:00:54Z" user="Marc Mongenet" uid="74847" lat="46.1626891" lon="6.0999989"/>
<node id="528023414" visible="true" version="2" changeset="5650345" timestamp="2010-08-31T23:01:19Z" user="Marc Mongenet" uid="74847" lat="46.1623226" lon="6.1000896"/>
<node id="528023424" visible="true" version="2" changeset="5650345" timestamp="2010-08-31T23:00:54Z" user="Marc Mongenet" uid="74847" lat="46.1669122" lon="6.0968684"/>
<node id="603323360" visible="true" version="3" changeset="18187796" timestamp="2013-10-05T00:46:29Z" user="Marc Mongenet" uid="74847" lat="46.1654152" lon="6.0913984"/>
<node id="777915144" visible="true" version="4" changeset="67980884" timestamp="2019-03-10T07:54:08Z" user="Dharma-ae" uid="314664" lat="46.1664426" lon="6.1075906">
 <tag k="crossing_ref" v="zebra"/>
 <tag k="highway" v="crossing"/>
</node>
<node id="777915153" visible="true" version="6" changeset="84198910" timestamp="2020-04-27T12:17:44Z" user="Romain1228" uid="9551672" lat="46.1711218" lon="6.1080182">
 <tag k="bus" v="yes"/>
 <tag k="ele" v="400"/>
 <tag k="highway" v="bus_stop"/>
 <tag k="name" v="Les Bruyères"/>
 <tag k="operator" v="TPG"/>
 <tag k="public_transport" v="platform"/>
 <tag k="uic_name" v="Plan-les-Ouates, Les Bruyères"/>
 <tag k="uic_ref" v="8593061"/>
</node>
```

Figure 3: Exemple de fichier *.osm*

On voit ici le fichier de sortie de l'API OSM, qui décrit les entités géographique comme des *nodes* ou des *ways*. Il faut donc faire le tri sur ces données.

DBpedia

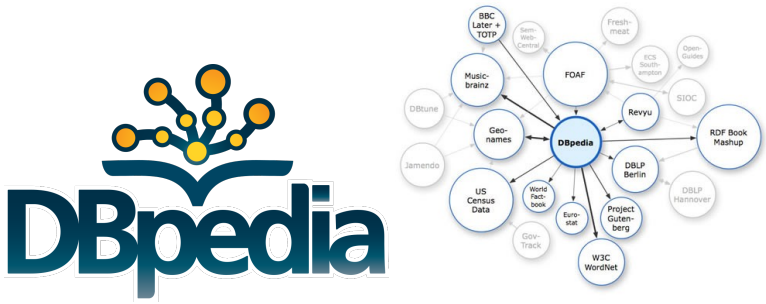


Figure 4: DBpedia

Nous allons maintenant présenter le rôle de DBpedia dans cette application. DBpedia est un projet universitaire et communautaire d'exploration et d'extraction automatique de données dérivées de Wikipédia. Son principe est de proposer une version structurée et normalisée au format du web sémantique des contenus de Wikipedia.

DBpedia adopte les normes du réseau linked open data et du Web sémantique. La ressource est donc livrée sous une forme de dépôt en format RDF regroupé au sein de documents dérivés de l'encyclopédie Wikipédia. Ainsi, pour chaque document encyclopédique, il existe une page de ressources contenant toutes les données sous formes de triplets RDF. Ces triplets peuvent représenter une information telle que la date de naissance d'une personne qui prendra la forme : (personne, date de naissance, date).
L'intérêt de cette interface dans notre application est de l'utiliser comme une base de données de connaissances, que l'on va interroger pour trouver des informations supplémentaires

Nous allons prochainement vous expliquer la structure de nos données collectées via OSM et GPX. Nous allons stocker nos données sous forme de graphe RDF, de la même manière que DBpedia afin qu'on puisse exécuter les requêtes SPARQL avec ces base de données.

Nous allons également expliquer comment les données ont étés triées efficacement pour être mises dans le graphe. Nous avons également fait un tri supplémentaire entre le graphe et la page HTML, afin d'accentuer la resemblance à un guide touristique

Graphe RDF

Graphe RDF (Resource Description Framework) est un modèle de graphe destiné à décrire formellement les ressources Web et leurs métadonnées, afin de permettre le traitement automatique de telles descriptions.

Notre graphe permet de combiner les trackpoints avec les nodes associés à chacune de ces trackpoints. Elle est liée via le predicat : $hasPointOfInterest$. Elle s'exprime de la manière suivante:

```
@prefix ns1: <http://osmInfo/> .
@prefix ns2: <http://trackpoint/> .

ns1:100 a ns1:node ;
    ns1:information "guidepost" ;
    ns1:latitude "45.7925201" ;
    ns1:longitude "6.9833412" .

ns2:1 a ns2:point ;
    ns2:latitude "45.790777" ;
    ns2:longitude "6.932203" .


ns2:1 a ns2:point ;
    ns2:HasPointOfInterest ns1:100,
        ns1:88 ;
    ns2:latitude "45.789519" ;
    ns2:longitude "6.962995" .
```

Figure 5: Un fichier turtle qui exprime une partie du graphe

On procède à l'explication de notre graphe.

On définit chaque *trackpoint* du fichier GPX avec l'url : $< http://trackpoint/ >$.
On définit chaque *node*, un point de OSM avec l'url : $< http://osmInfo/ >$.

- $ns1$: 100 est le 100ème node qui a les informations fournies par les tags du fichier *.osm*. suivis de la longitude et latitude du noeud correspondant
- $ns2 : 1$ est un point($ns1 : point$,trackpoint) qui contient les attributs suivants : longitude, latitude.
- (LIEN) $ns2 : 1$ est un point($ns2 : point$,trackpoint) qui a un point d'intérêt qui est le node $ns1 : 100$.

Le lien est défini par une relation de distance entre le trackpoint et le node. On calcule à l'aide d'une fonction, la distance entre chaque trackpoint et tout les nodes, si la distance est inférieur à une valeur d, on considère que ce node est un point d'intérêt pour ce trackpoint. La valeur d, est un paramètre défini par l'utilsateur dans l'application que nous allons expliquer dans la prochaine partie.

Nous allons maintenant montrer le graphe RDF de nos données, combinant les données GPX et le données de OSM.

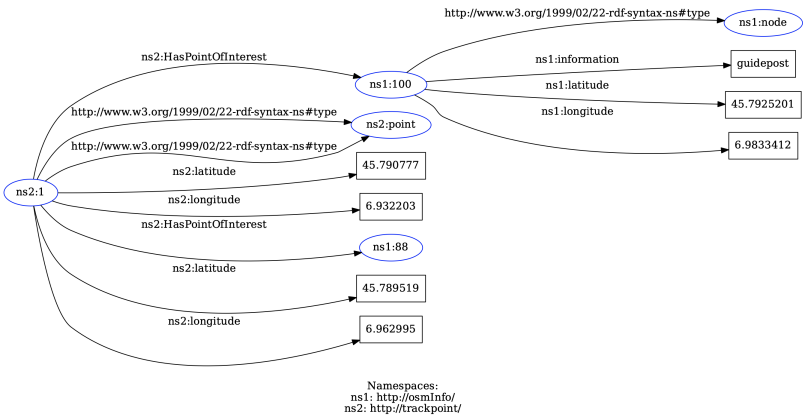


Figure 6: Graphe RDF

Ce graphe ne représente pas l'intégralité de notre graphe pour un fichier *.gpx* donné.

Requête SPARQL

Après avoir transformé nos données en graphe rdf, il nous était nécessaire de récupérer les points d'interêts de chacun de nos trackpoints pour pouvoir les afficher lors du lancement de notre application. Pour cela, il suffit d'exécuter une query SPARQL sur notre graphe. C'est également à ce moment que nous avons décidé de trier nos données *.osm* par leur tag "name" (nom) et "amenity" (catégorie) afin de récuperer les informations pertinentes pour l'utilisateur. Voici donc l'aspect de notre requête SPARQL, dans un premier temps, sur GrapheDB:

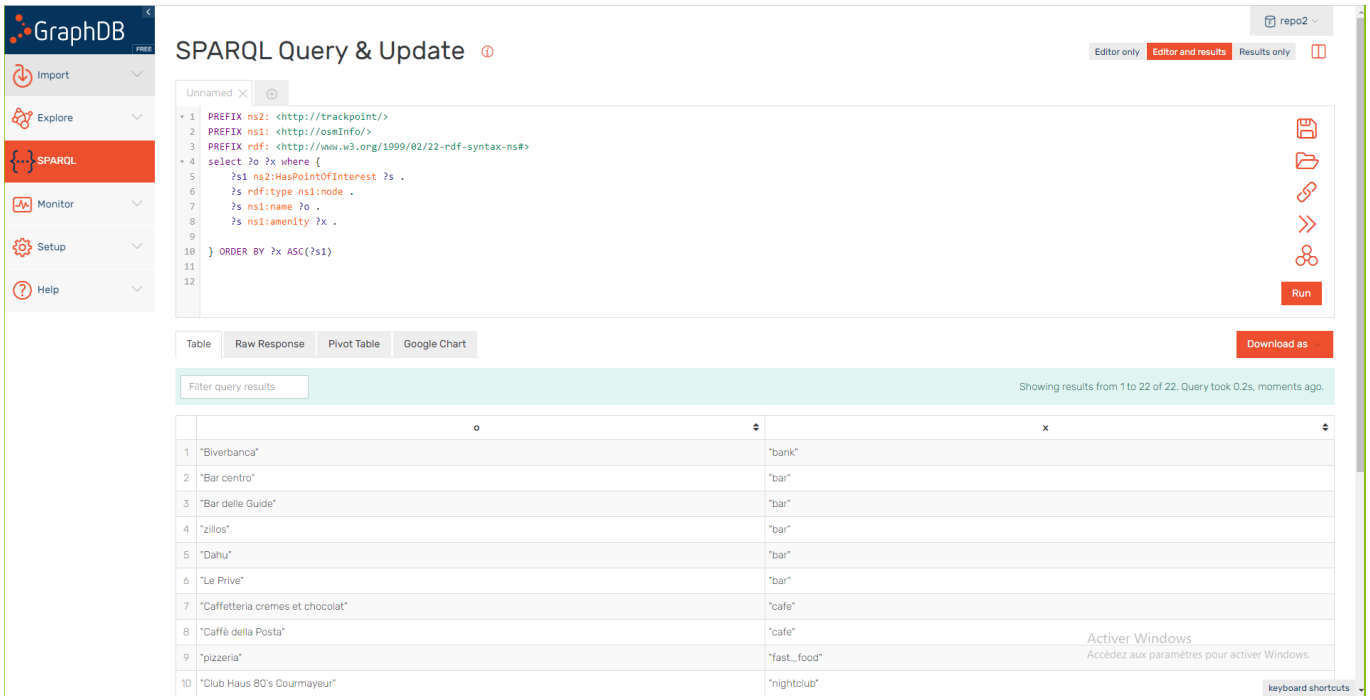


Figure 7: Exemple de requête SPARQL

Il nous a donc fallut faire la même chose dans notre code python à l'aide de la librairie rdflib. Le résultat de cette requête nous permet de définir un dictionnaire indexé par la catégorie "amenity" (Par example: Bar) qui contient une liste de tous les noms possèdants cette même catégorie. Ainsi, il devient très simple d'afficher sur une page html chacun des points d'interêts.

Application : UNIGE_MAP

Dans cette partie, nous allons vous présenter notre interface utilisateur. Elle est implémentée en Python (cf *app.py*) en utilisant la librairie PyQt5. Pour lancer l'application, il suffit d'exécuter la commande: *python app.py* dans l'emplacement du fichier *app.py*.

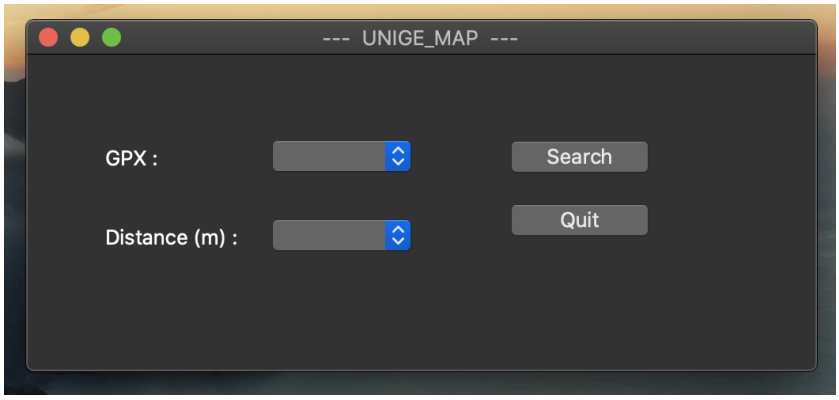


Figure 8: Interface utilisateur

Elle prend deux paramètres d'entrées et lance une page HTML contenant les informations sur une trajectoire donnée par le fichier *.gpx*. Les deux paramètres sont : un fichier GPX et une distance en mètres. Cette distance est utilisée pour identifier les points d'intérêts. (On explique cela dans la partie précédente).

- bouton **Search** : permet de lancer la page HTML
- bouton **Quit** : permet de quitter l'application

Résultats

Testons notre application avec un exemple.

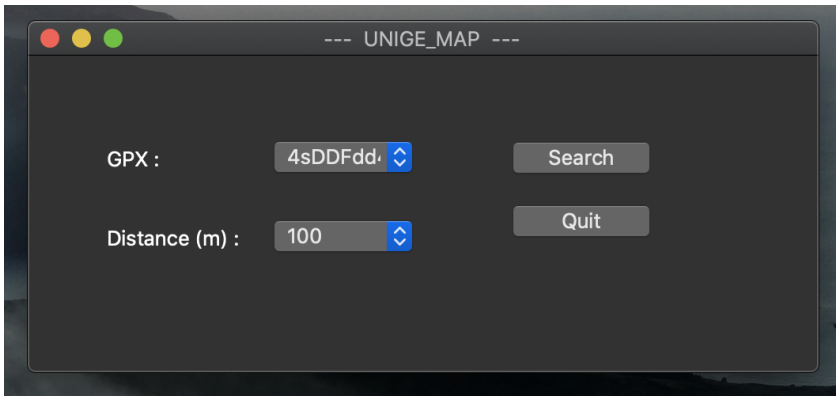


Figure 9: Test de l'application

Nous avons choisi un fichier GPX donné avec le projet (4sDDFdd4cjA.gpx) et une distance de 100 mètres.

La page HTML se présente de la manière suivante :

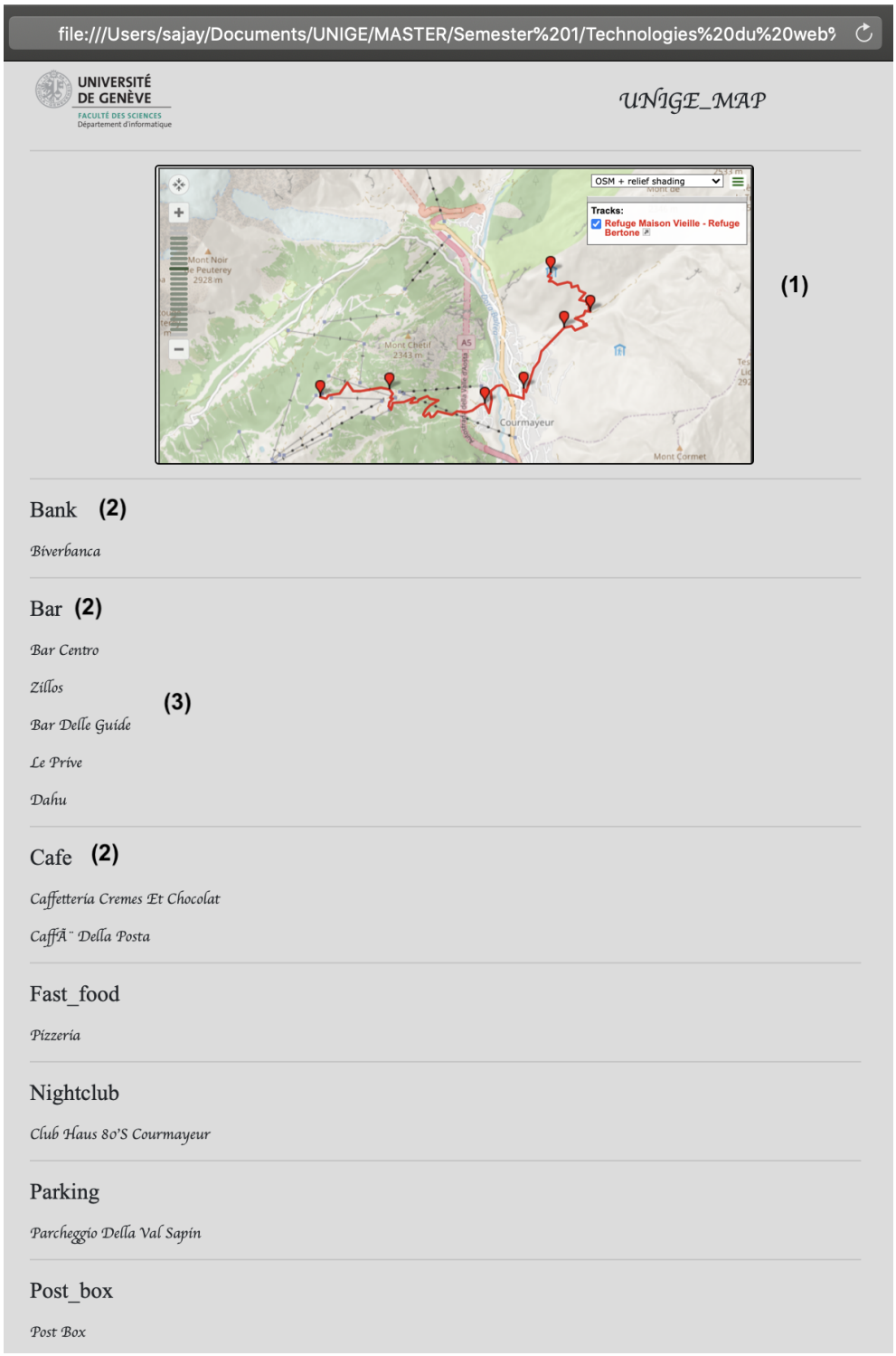


Figure 10: Page HTML, le résultat d'une recherche

On a le résultat de notre application qui est une page HTML comprennant plusieurs informations.

- (1) : Le fichier GPX transformé en carte
- (2) : Les catégories des points d'intérêts (restaurants, bar, parking, etc .)
- (3) : Le nom des points d'intérêts

Amélioration possibles

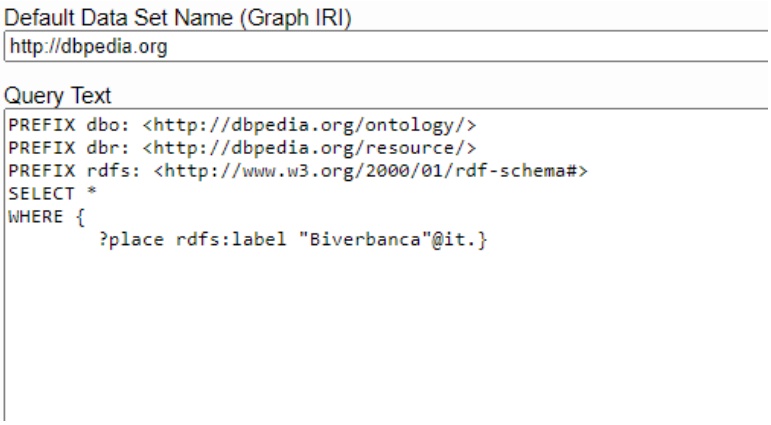
Graphe DBpedia

Malheureusement, nous n'avons pas pu effectuer des requêtes sparql pour obtenir des informations supplémentaires sur les lieux proposés par l'application et cela pour plusieurs raisons:

La première est notre manière de définir nos points d'intérêts provenant du fichier osm correspondant. Puisque nous avons jugé plus pertinent de filtrer nos noeuds par les tags "name" et "amenity", afin de prendre uniquement les éléments les plus essentiels pour un utilisateur qui compte marcher sur un parcours donné, cela pose problème lorsque nous souhaitons étoffer la description d'un point puisque ceux-ci ne possède pas forcément une référence dbpedia. Prenons par example un bar nommé "Dahu", il nous ai possible d'effectuer une recherche que par l'attribut name pour trouver des informations sur ce lieu en particulier, puisque la combinaison bar et nom ne mène à aucun résultat. D'un autre coté avec l'attribut name, la requête nous donne la description d'un animal sauvage, ce qui ne correspond pas aux données recherchées. En résumé, en filtrant nos données osm par certains tags, nous avons également restreint nos possibilités de recherche lors d'une requête sur dbpedia.

La deuxième provient de la langue de certains lieux qui ne sont pas toujours françaises. Lors d'une requête, il est nécessaire de spécifier la langue sur un label pour obtenir une description dans la même langue. Cela pose problème lorsque nous voulons automatiser les requêtes sur chacun de nos lieux, puisqu'une recherche sur un nom italien en spécifiant la langue française ne donnera aucun résultat. De plus, si nous décidions de donner la langue italienne en paramètre, la description se verra être dans cette même langue, ce qui ne convient pas aux résultats attendus.

Voici en image le problème décrit avec la banque nommé "Biverbanca" avec la requête suivante:



Et voici la description obtenu:

Malheureusement dans ce cas précis, en effectuant une recherche en français nous n'obtenons aucun résultat.

Une application générale

Une amélioration possible est d'implémenter cette application de manière générale. Dans notre cas, nous avons fait, par manque de temps, séparemment la partie OSM et stocké en local pour pouvoir faire des graphe RDF. Ce qui fait que cette application ne va pas fonctionner pour un cas générale. Une amélioration possible est d'inclure la partie communication avec le API de Open Street Map dans la fonction principale. De telle manière, notre code fonctionne pour tout fichier *.gpx*.

Ce que le TP nous apporte ...

Tout d'abord, ce TP nous a permis de travailler en binôme et de mettre en commun nos connaissances pour aboutir à un résultat. Il nous a également permis de nous familairiser avec les API de Open Street Map, GraphDB et DBpedia. Nous avons également pu apprendre beaucoup de choses sur l'interface utilisateur PyQt5 de python. Le plus important, il nous a permis de découvrir la structure sémantique du web, notamment avec les graphes RDF, RDFS. Nous avons également appris l'outil *rdflib* de python pour construire , traiter, et analyser les graphes RDF.