

Jeremy Keener
Amélie Noyelle
Josselin Peniguel

Web Dataming & Semantics report

Goal of the project : We had to create an application (html/CSS) that integrates tourist places and points of interest, including dynamic data.

I- Modeling the ontology on protégé

To be sure to set up the project properly, we created the ontology on protected. We used 3 distinct datasets :

The first dataset is a dataset that lists all the museums in France. We want to target our program on museums in Ile de France, but the dataset that concerns the whole France works too.

Museum dataset :

<https://data.culturecommunication.gouv.fr/explore/dataset/liste-et-localisation-des-musees-de-france/export/>

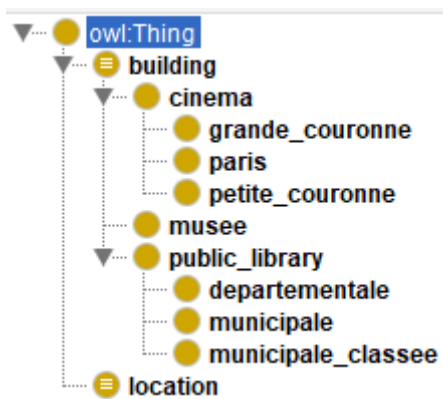
Cinemas dataset :

[https://data.iledefrance.fr/explore/dataset/les salles de cinemas en ile-de-france/export/](https://data.iledefrance.fr/explore/dataset/les_salles_de_cinemas_en_ile-de-france/export/)

Library Dataset :

<https://data.culturecommunication.gouv.fr/explore/dataset/adresses-des-bibliotheques-publiques/export/>

On protected, we used some attributes of the different datasets, and we added the class 'Location'.

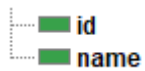


Then we added data properties :

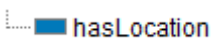
The data properties for the location class are :



The data properties for the cinema, musee and public_library are :



The only object property is this one :

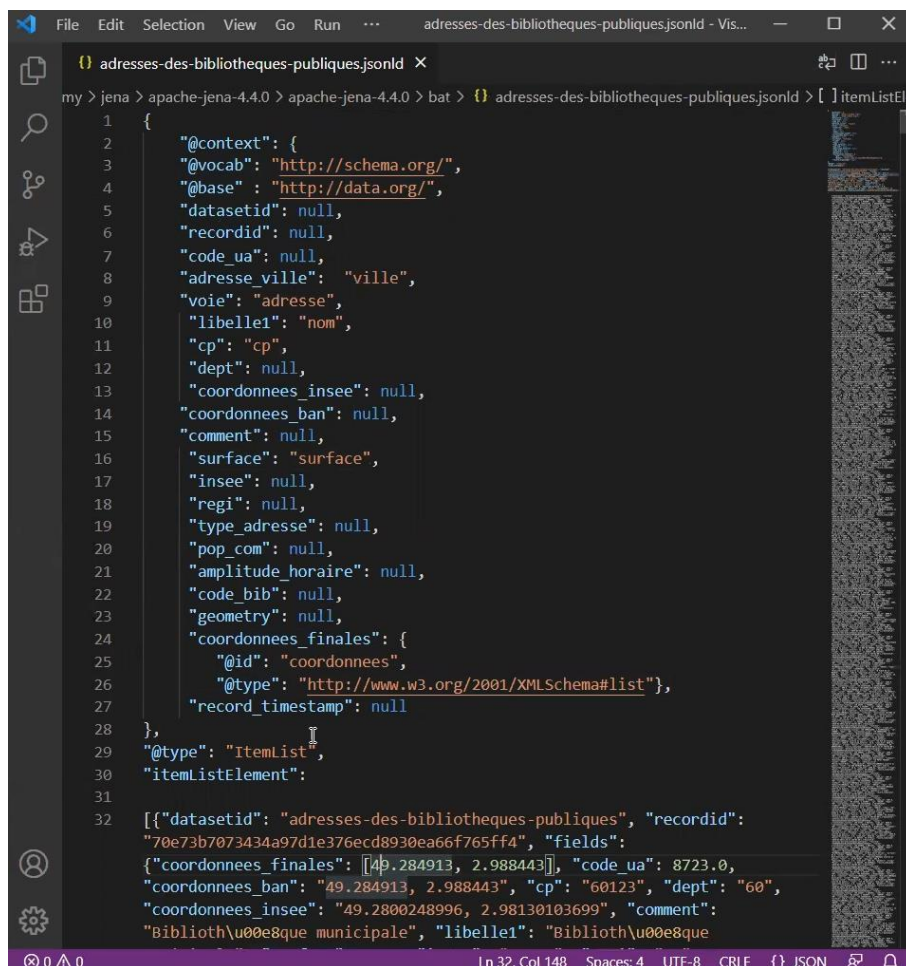


II- Creation of RDF files

Then, we plan to use Jena Fuseki to be able to implement our project. To be able to import our Json files into Jena Fuseki, we need to import files in RDF format. For that, we will download each dataset in Json format, then we will convert them manually to be sure that the conversion is well done.

Before converting the files to RDF format, we had to modify the files to JSON format as the variables present in the databases were not useful to us. As you can see on the slots, we assigned a value to each variable. And when this variable was not useful to us, we signed its value to "null". This allows us, once the conversion to RDF format is done, to have only the variables that we will use in our application, on the local host.

So we have done this task for the three datasets we have used. In this dictionary of variables, we can also observe that the final coordinate variable is a bit different compared to the others. Indeed, we have used an already existing anthology. This reuse of vocabulary allows us to directly import the geographical position in the coordinate format.

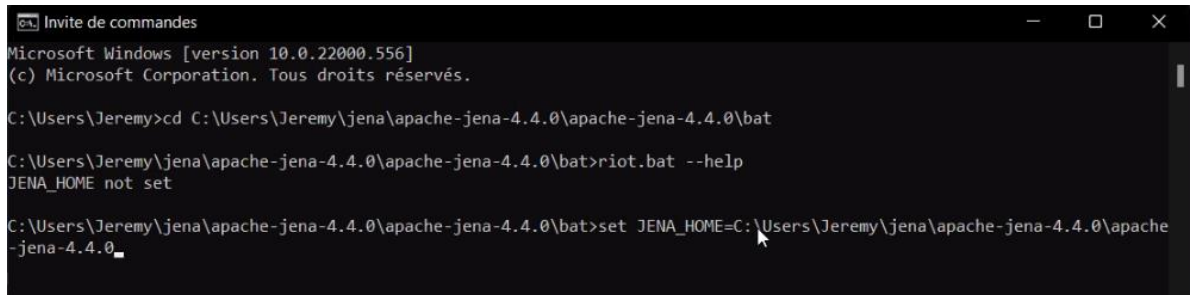


```
1 {
2   "@context": {
3     "@vocab": "http://schema.org/",
4     "@base": "http://data.org/",
5     "datasetid": null,
6     "recordid": null,
7     "code_ua": null,
8     "adresse_ville": "ville",
9     "voie": "adresse",
10    "libelle1": "nom",
11    "cp": "cp",
12    "dept": null,
13    "coordonnees_insee": null,
14    "coordonnees_ban": null,
15    "comment": null,
16    "surface": "surface",
17    "insee": null,
18    "regi": null,
19    "type_adresse": null,
20    "pop_com": null,
21    "amplitude_horaire": null,
22    "code_bib": null,
23    "geometry": null,
24    "coordonnees_finales": {
25      "@id": "coordonnees",
26      "@type": "http://www.w3.org/2001/XMLSchema#list"},
27    "record_timestamp": null
28  },
29  "@type": "ItemList",
30  "itemListElement":
31
32  [{"datasetid": "adresses-des-bibliotheques-publiques", "recordid":
33    "70e73b7073434a97d1e376ecd8930ea66f765ff4", "fields":
34    [{"coordonnees_finales": "[49.284913, 2.988443]", "code_ua": 8723.0,
35      "coordonnees_ban": "49.284913, 2.988443", "cp": "60123", "dept": "60",
36      "coordonnees_insee": "49.2800248996, 2.98130103699", "comment":
37        "Biblioth\u00e8que municipale", "libelle1": "Biblioth\u00e8que
```

Once the various modifications of the JSON files were made, we carried out the conversion manually thanks to the windows command prompt. We will explain below how we proceeded.

The first thing to do, which is a detail that goes up in the transformation of the file, is to change the file extension from .json to .jsonld. The file remains similar but this will make the conversion easier later.

Then, once in the windows command prompt, you have to go to the folder that contains the .jsonld files, then run the command `riot.bat --help`. On the image above, we can see that this command returns "JENA_HOME not set". This tells us that we have to launch Jena, before we can transform our files to RDF format.



```
Microsoft Windows [version 10.0.22000.556]
(c) Microsoft Corporation. Tous droits réservés.

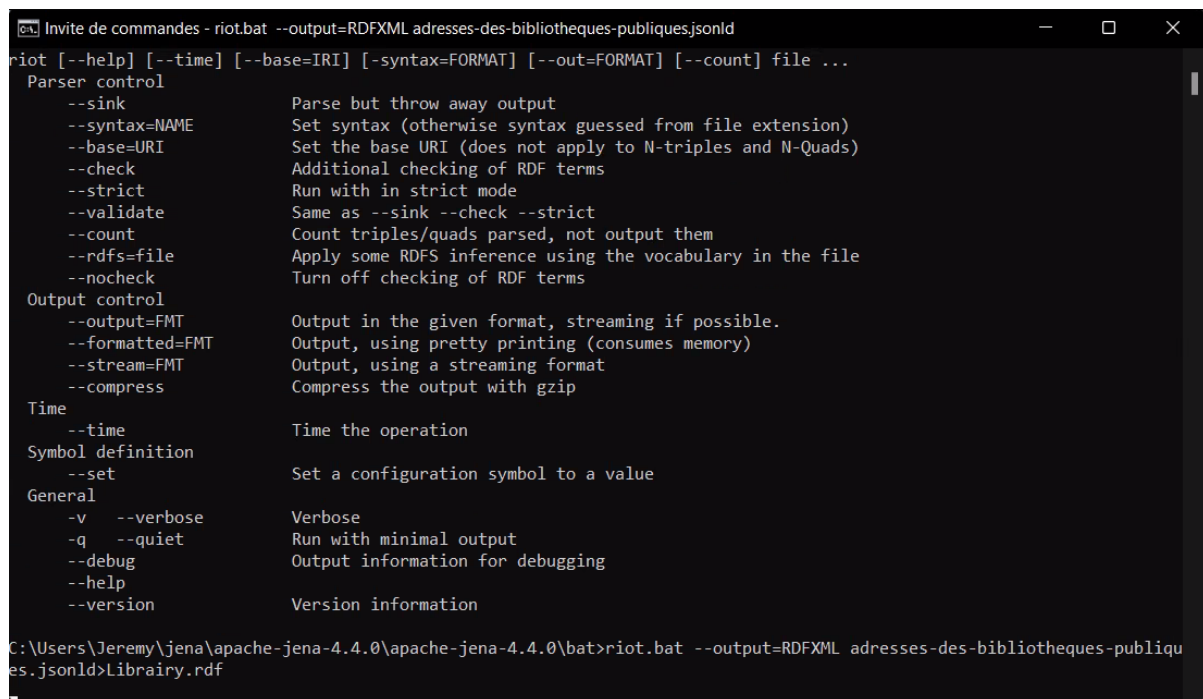
C:\Users\Jeremy>cd C:\Users\Jeremy\jena\apache-jena-4.4.0\apache-jena-4.4.0\bat

C:\Users\Jeremy\jena\apache-jena-4.4.0\apache-jena-4.4.0\bat>riot.bat --help
JENA_HOME not set

C:\Users\Jeremy\jena\apache-jena-4.4.0\apache-jena-4.4.0\bat>set JENA_HOME=C:\Users\Jeremy\jena\apache-jena-4.4.0\apache-jena-4.4.0_
```

So we launch Apache Jena, then, by re-executing the help command, we obtain all the possible commands with Apache Jena. In our particular case, the command that interests us is the command `output=RDFXML`, and that we can see at the end of the screenshot below. This command allows us to convert the file in .jsonld format and in a new file of the name of our choice, in RDF format.

We have therefore realized this task for the three databases they have leaked we then attacked to import these files in our triple store, Fuseki.



```
Invite de commandes - riot.bat --output=RDFXML adresses-des-bibliotheques-publiques.jsonld
riot [--help] [--time] [--base=IRI] [--syntax=FORMAT] [--out=FORMAT] [--count] file ...

Parser control
  --sink                Parse but throw away output
  --syntax=NAME         Set syntax (otherwise syntax guessed from file extension)
  --base=URI            Set the base URI (does not apply to N-triples and N-Quads)
  --check               Additional checking of RDF terms
  --strict              Run with in strict mode
  --validate            Same as --sink --check --strict
  --count               Count triples/quads parsed, not output them
  --rdfs=file           Apply some RDFS inference using the vocabulary in the file
  --nocheck             Turn off checking of RDF terms

Output control
  --output=FMT          Output in the given format, streaming if possible.
  --formatted=FMT       Output, using pretty printing (consumes memory)
  --stream=FMT          Output, using a streaming format
  --compress            Compress the output with gzip

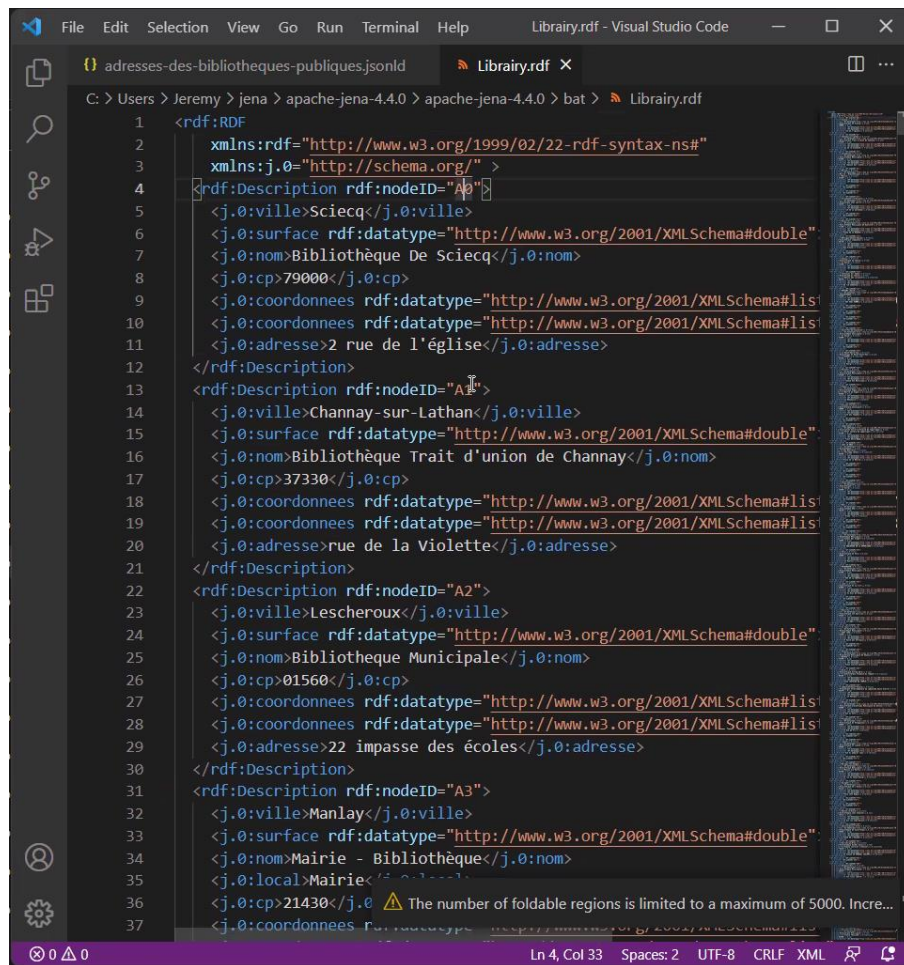
Time
  --time                Time the operation

Symbol definition
  --set                 Set a configuration symbol to a value

General
  -v --verbose          Verbose
  -q --quiet            Run with minimal output
  --debug              Output information for debugging
  --help               Version information

C:\Users\Jeremy\jena\apache-jena-4.4.0\apache-jena-4.4.0\bat>riot.bat --output=RDFXML adresses-des-bibliotheques-publiques.jsonld>Library.rdf
```

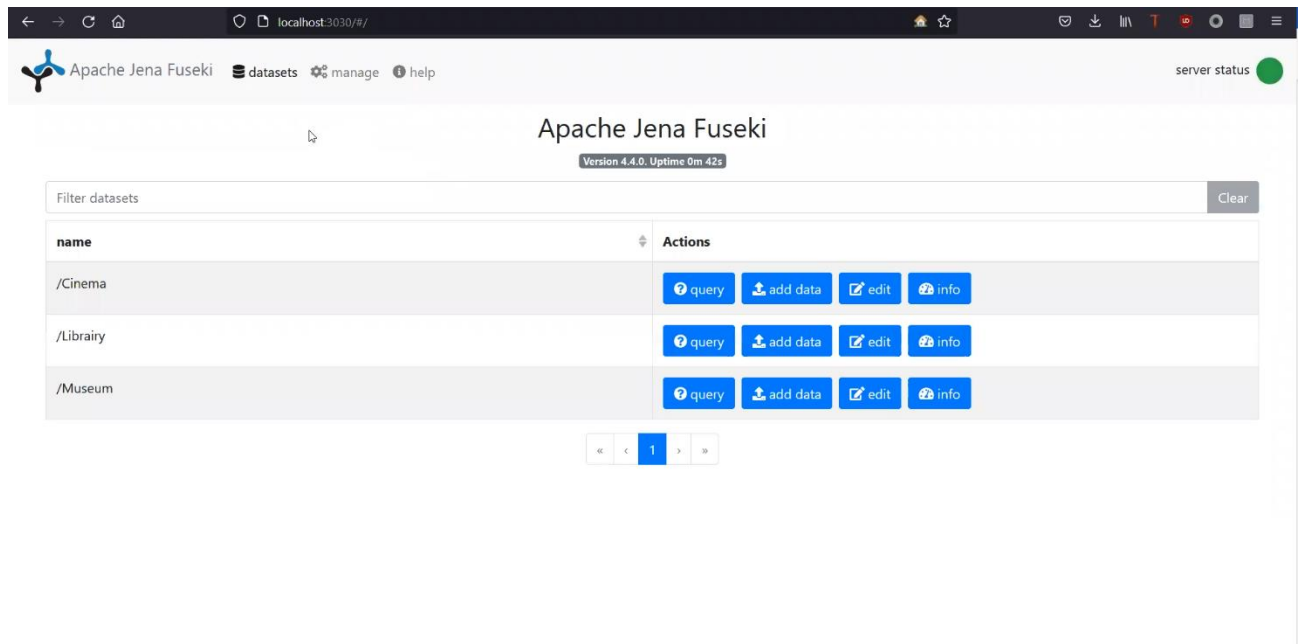
We obtain the files in RDF format:



```
1 <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3   xmlns:j.0="http://schema.org/" >
4   <rdf:Description rdf:nodeID="A0">
5     <j.0:ville>Sciecq</j.0:ville>
6     <j.0:surface rdf:datatype="http://www.w3.org/2001/XMLSchema#double">
7       <j.0:nom>Bibliothèque De Sciecq</j.0:nom>
8       <j.0:cp>79000</j.0:cp>
9       <j.0:coordonnees rdf:datatype="http://www.w3.org/2001/XMLSchema#list">
10        <j.0:coordonnees rdf:datatype="http://www.w3.org/2001/XMLSchema#list">
11          <j.0:adresse>2 rue de l'église</j.0:adresse>
12        </rdf:Description>
13      <rdf:Description rdf:nodeID="A1">
14        <j.0:ville>Channay-sur-Lathan</j.0:ville>
15        <j.0:surface rdf:datatype="http://www.w3.org/2001/XMLSchema#double">
16          <j.0:nom>Bibliothèque Trait d'union de Channay</j.0:nom>
17          <j.0:cp>37330</j.0:cp>
18          <j.0:coordonnees rdf:datatype="http://www.w3.org/2001/XMLSchema#list">
19            <j.0:coordonnees rdf:datatype="http://www.w3.org/2001/XMLSchema#list">
20              <j.0:adresse>rue de la Violette</j.0:adresse>
21            </rdf:Description>
22          <rdf:Description rdf:nodeID="A2">
23            <j.0:ville>Lescheroux</j.0:ville>
24            <j.0:surface rdf:datatype="http://www.w3.org/2001/XMLSchema#double">
25              <j.0:nom>Bibliothèque Municipale</j.0:nom>
26              <j.0:cp>01560</j.0:cp>
27              <j.0:coordonnees rdf:datatype="http://www.w3.org/2001/XMLSchema#list">
28                <j.0:coordonnees rdf:datatype="http://www.w3.org/2001/XMLSchema#list">
29                  <j.0:adresse>22 impasse des écoles</j.0:adresse>
30                </rdf:Description>
31              <rdf:Description rdf:nodeID="A3">
32                <j.0:ville>Manlay</j.0:ville>
33                <j.0:surface rdf:datatype="http://www.w3.org/2001/XMLSchema#double">
34                  <j.0:nom>Mairie - Bibliothèque</j.0:nom>
35                  <j.0:local>Mairie</j.0:local>
36                  <j.0:cp>21430</j.0:cp>
37                  <j.0:coordonnees rdf:datatype="http://www.w3.org/2001/XMLSchema#list">
```

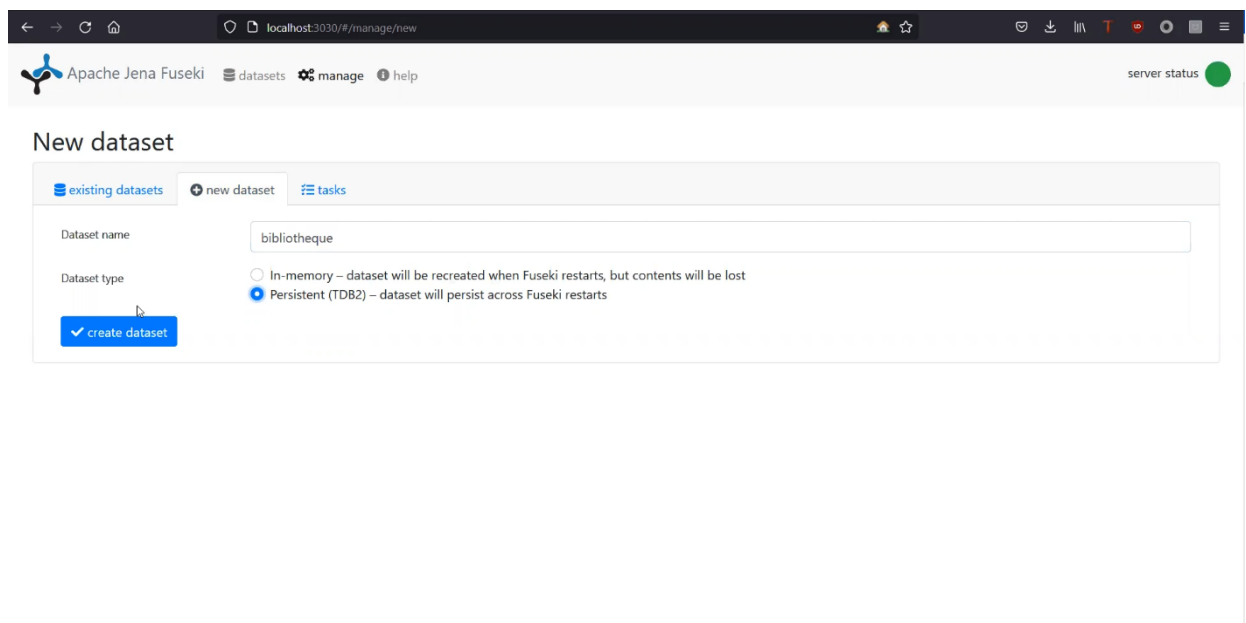
III- Use of the Triple Store

To use apache Fuseki, we downloaded the version available on the website, then, we launched the file fuseki-server.bat. Once we have run this file, we can go on our internet explorer and go on the page "http://localhost:3030/". We will then arrive on our Fuseki server :



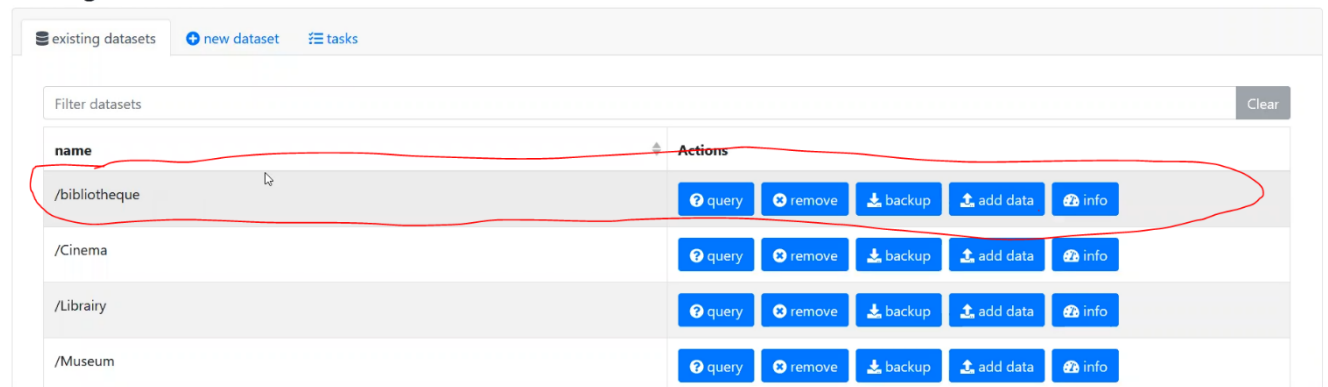
We observe that Canton opens this page, we arrive directly on the tab datasets, which represents all the datasets that have been brought. In this case, our three datasets were well imported.

Although we dataset is already imported, for the good understanding of the process, we will import a fourth fictitious dataset 'bibliotheque' :

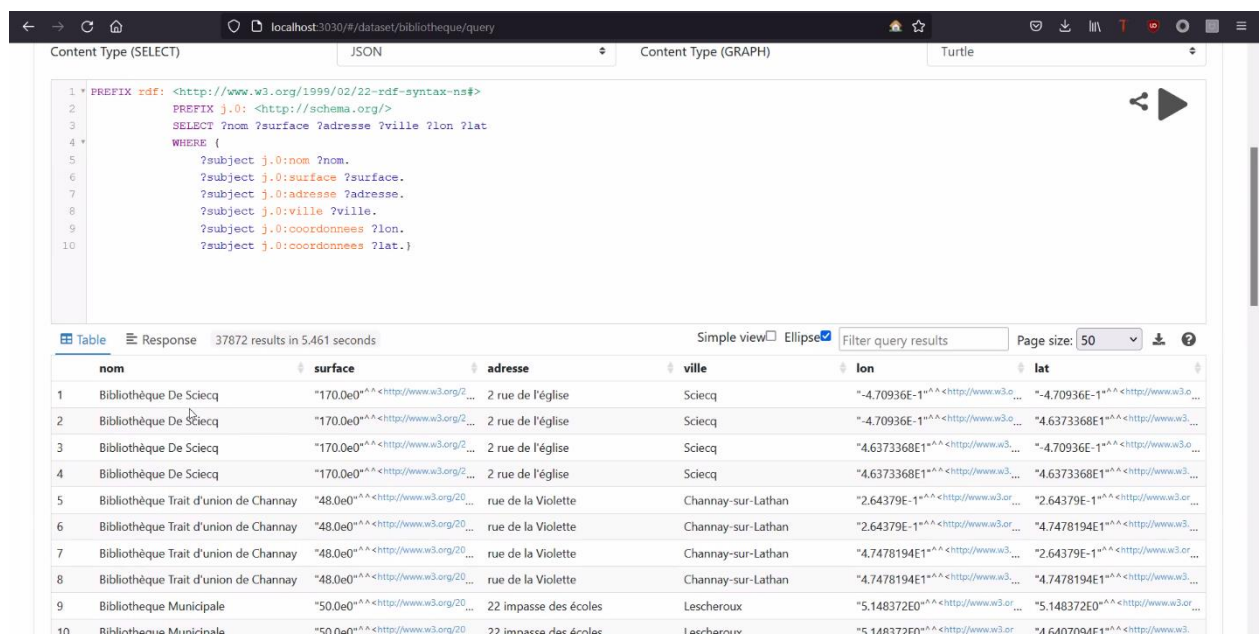


By clicking on the button one dataset, the dataset has been created :

Manage datasets



For the moment, our library dataset does not contain any data. This is where the conversion to RDF format takes all its importance. Indeed, it is much easier to import files in RDF format into Fuseki than files in json format. To import data, just click on the 'add data' button, then select the RDF file you want to import. Finally, to test that everything works, we can make a request within apache fuseki directly, to observe what it returns.

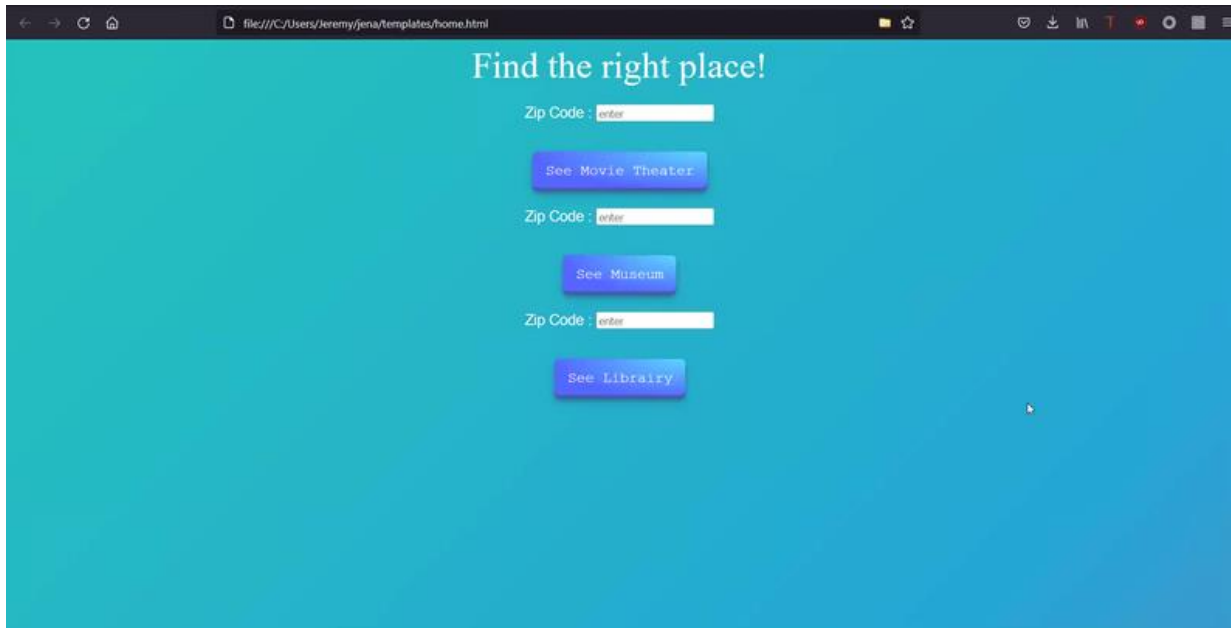


We see that the databases and the query are functional.

IV- Creation of the application (html/css) using python

The final step of this project is to create a web application, which allows us to visualize on a map the different points of interest (in our case cinemas, museums or libraries), using several python libraries and SPARQL queries.

In the same folder as the python file, we created a Template folder, which contains the html and css. We chose to code in html and css to make it look nicer. Within our python code, we have made functions, which perform queries on our apache fuseki database.



For the map display, we used the folium library and we created our application with Spark.

In our python file, we have 3 functions, which allow respectively the display of the coordinates (cinema, museum and libraries).

Then, we realized several functions which allow to realize queries with the various datasets which we used. These functions allow to put in relations the information that the user will inform on the web page (html), to recover them and then to make SPARQL requests in apache fuseki, using this information.

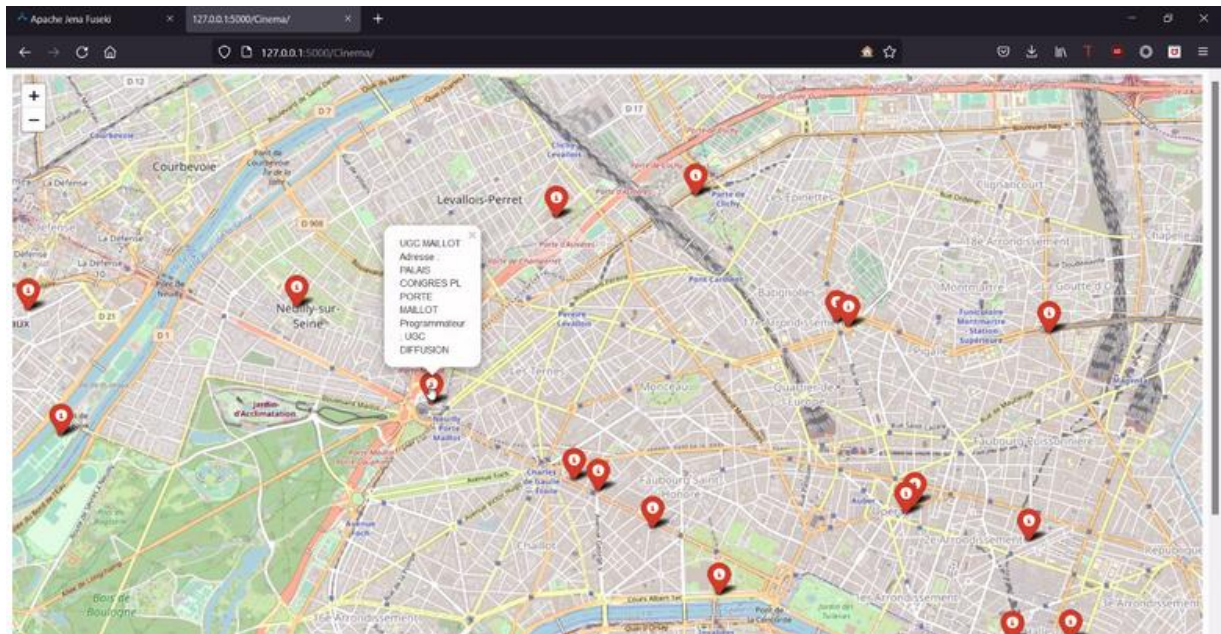
Our interface allows to make several types of requests. First of all, we can search for cinemas using a department number.

Then, for museums and libraries, it is possible to display them on the map according to the department or directly according to the postal code to get a more precise result. These queries are therefore dynamic since the user enters a specific postal code, which we then use in our Sparql queries.

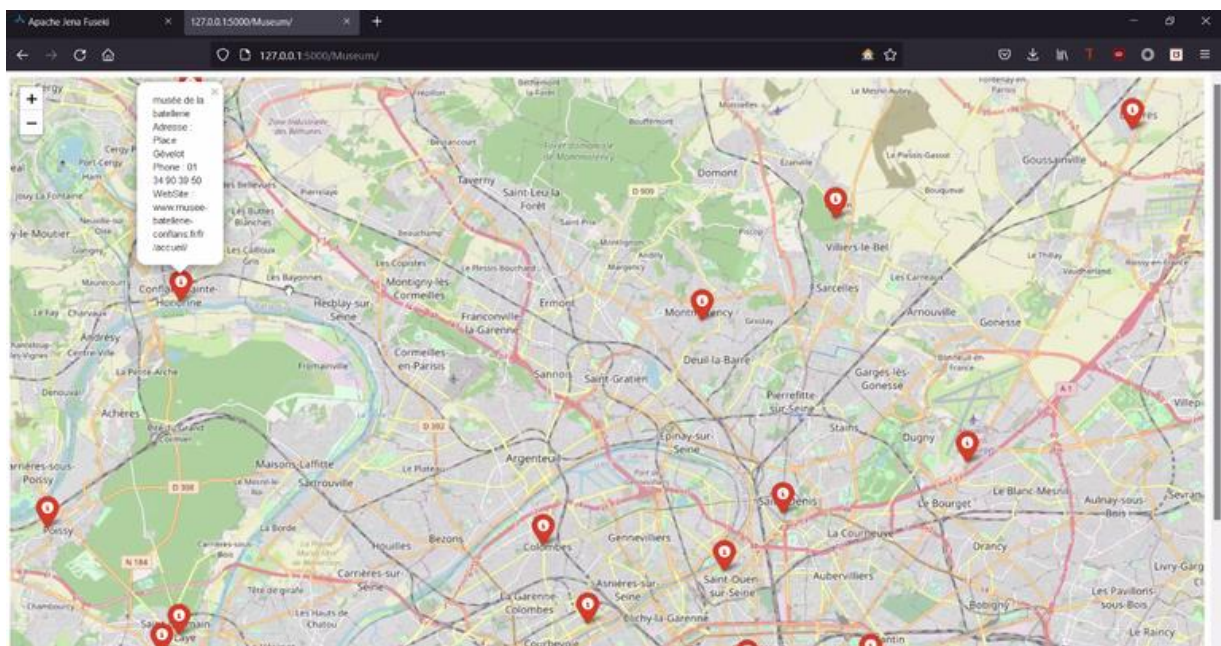
In order to let the user choose to search by department or by postal code, we used regexes in our python code to get the right results and let the user choose.

Here are some examples of how our project is used:

In this first image, we did a search for cinemas in Paris.



In this second example, we made the search of all the museums in ile de france.



Finally, in this 3rd example, we have made the search of all the libraries in all the ile de france.

