RAPPORT

Master DSC2 CSPS2 – 2021/2022

Semantic Web

Encadré par : Pr. Antoine ZIMMERMANN

Réalisé par :

- Ali Hani ABID
- Adnane M'barki
- Arjun PULLIYASERI

Index

I. Introduction

Our web application is an application that integrates data from multiple sources, including dynamic data. The main objective is to be able to define a knowledge model that describes the types of entities stored in the Fuseki triplestore. Indeed, the display of the information collected on a web page with structured data and by querying our Triplestore The establishment of the RDF database is based on different RDF and non RDF sources. From a search engine, Our application makes it possible to browse the data related to the rooms of the building and the information collected by the sensors about the temperature, the date and time as well as the location of the sensors in the building when the temperature is different from of the outside temperature abstained thanks to the weather data and from this difference a measurement classification is established (normal, interesting, alarming).

II. Choice and study of technologies

Before proceeding to develop the application, we had first to take a brief study at the existing technologies. After research, we opted for:

-Java/Spring: The back end framework is known for the large documentation offered by Jena, concerning the creation of the model (Subject, predicate, object), furthermore the framework itself is lightweight, loose coupling…., However Spring is known for its complexity which is considered as an inconvenient for this framework, but since our application is simple (don't contain a lot of functionalities), we will not have this problem.

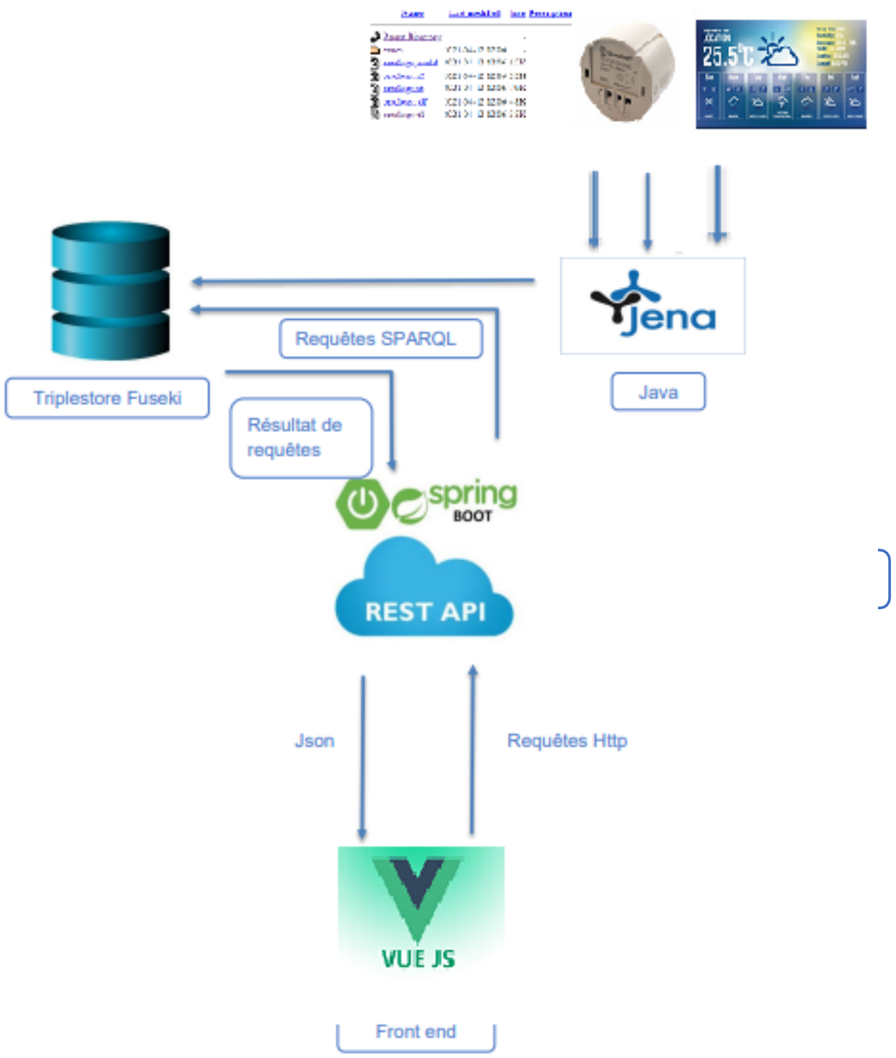
-Vue Js : The front end framework is known for Its simplicity, and use of Virtual Dom (functionality that permits to not refresh the page each time there is a client side update),we used this one for filtering the rooms that we will see in the final section of this report, finally it is very flexible.
-Triple store Fuseki : As indicated by its name ,it's a database which store a rdf data, and can be retrieved using a query language, for our case we used Sparql

-Triple store Fuseki : As indicated by its name ,it's a database which store a rdf data, and can be retrieved using a query language, for our case we used Sparql

-Web Service: we used Rest API ,to expose our data as Json,if you launch, the back end ,you could see the Json data of sensors at localhost:8080/api/sensors, and this data is retrieved in front end using Vue Js .

III. Application architecture

The figure below represents the general architecture of our semantic web application :



Here, we attempted to power the installation of many technologies in order to build cutting-edge web applications. The Apache Jena framework and technologies like as spring-boot and Vue.js manage the application's heterogeneous resourced input effectively.

The basic purpose is to mix numerous things from multiple sources in meaningful form. We obtain our weather information from 'meteociel.fr.' The data is translated from HTML to a CSV file containing the available parameters, such as temperature and humidity. Sensor information from the EMSE database, as well as EMSE's 'Geographical' knowledge graphs, are then colocated to the triple database, which is 'Apache Jena-fuseki,' with a vital link to Apache Jena's property allocations.

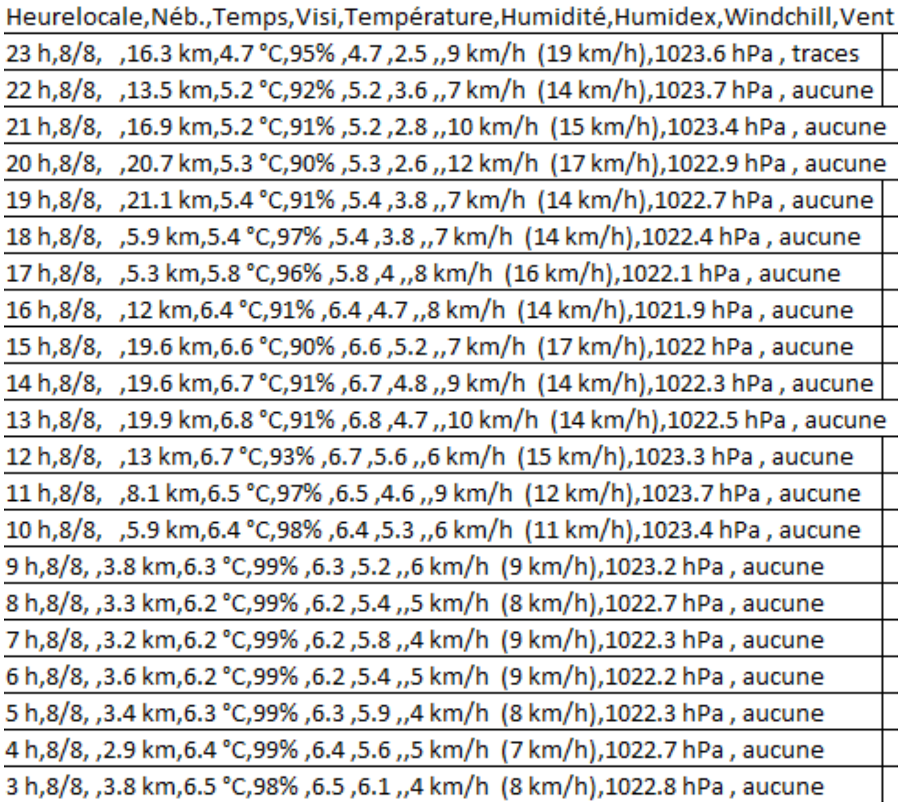
The Fuseki triple storage is armed with the knowledge connection between space data, sensor data, and metrological characteristics. The SPARQL query is generally transmitted down using our Backend engine - Spring-Boot throughout the project's middle phase. The frontend app received all of the RDF data in JSON format. To select and drop JSON data to the frontend, we've integrated REST Api. In order to execute in Frontend, the lightweight framework 'vue.js' has been engaged and fueled.

IV. Data processing

IV.1 Source and description of data

- Plateforme Territoire gives a describtion of the building of Mines Saint-Étienne at Espace Fauriel, its rooms, spaces, storeys,…,etc in RDF files, our choice is to take all the turtle files for this describtion.

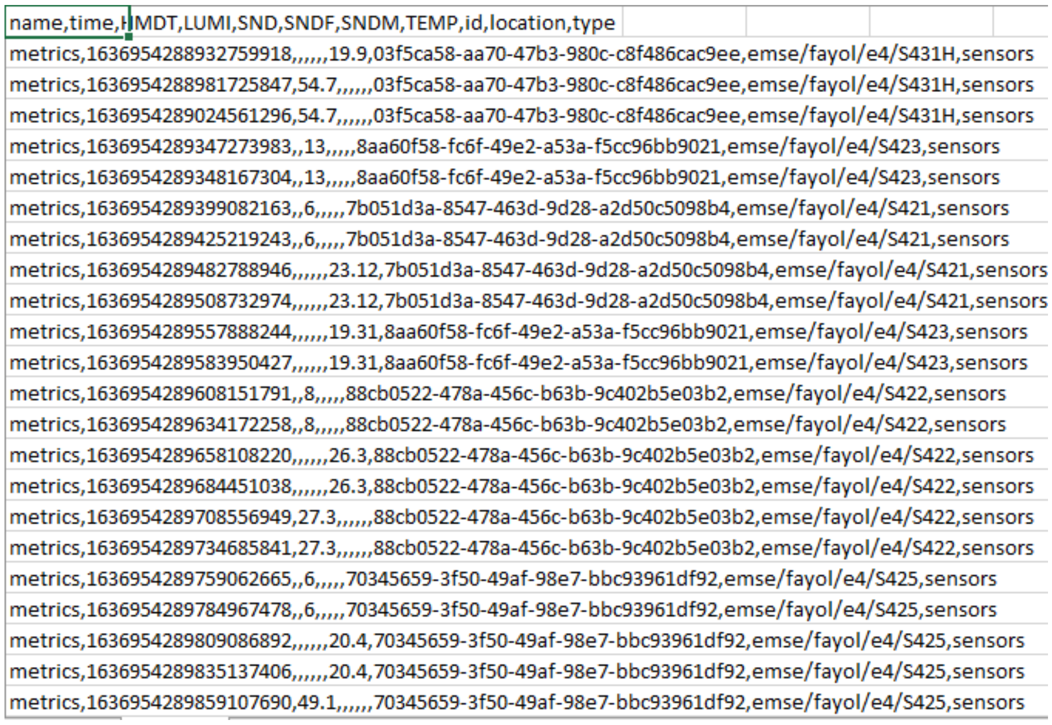
- Weather data for Saint-Étienne Meteociel.fr. The data is only available from an HTML table so we convert it to CSV format as we can see in the figure below:



```
Heurelocale,Néb.,Temps,Visi,Température,Humidité,Humidex,Windchill,Vent
23 h,8/8,  ,16.3 km,4.7 °C,95% ,4.7 ,2.5 ,,9 km/h  (19 km/h),1023.6 hPa , traces
22 h,8/8,  ,13.5 km,5.2 °C,92% ,5.2 ,3.6 ,,7 km/h  (14 km/h),1023.7 hPa , aucune
21 h,8/8,  ,16.9 km,5.2 °C,91% ,5.2 ,2.8 ,,10 km/h  (15 km/h),1023.4 hPa , aucune
20 h,8/8,  ,20.7 km,5.3 °C,90% ,5.3 ,2.6 ,,12 km/h  (17 km/h),1022.9 hPa , aucune
19 h,8/8,  ,21.1 km,5.4 °C,91% ,5.4 ,3.8 ,,7 km/h  (14 km/h),1022.7 hPa , aucune
18 h,8/8,  ,5.9 km,5.4 °C,97% ,5.4 ,3.8 ,,7 km/h  (14 km/h),1022.4 hPa , aucune
17 h,8/8,  ,5.3 km,5.8 °C,96% ,5.8 ,4 ,,8 km/h  (16 km/h),1022.1 hPa , aucune
16 h,8/8,  ,12 km,6.4 °C,91% ,6.4 ,4.7 ,,8 km/h  (14 km/h),1021.9 hPa , aucune
15 h,8/8,  ,19.6 km,6.6 °C,90% ,6.6 ,5.2 ,,7 km/h  (17 km/h),1022 hPa , aucune
14 h,8/8,  ,19.6 km,6.7 °C,91% ,6.7 ,4.8 ,,9 km/h  (14 km/h),1022.3 hPa , aucune
13 h,8/8,  ,19.9 km,6.8 °C,91% ,6.8 ,4.7 ,,10 km/h  (14 km/h),1022.5 hPa , aucune
12 h,8/8,  ,13 km,6.7 °C,93% ,6.7 ,5.6 ,,6 km/h  (15 km/h),1023.3 hPa , aucune
11 h,8/8,  ,8.1 km,6.5 °C,97% ,6.5 ,4.6 ,,9 km/h  (12 km/h),1023.7 hPa , aucune
10 h,8/8,  ,5.9 km,6.4 °C,98% ,6.4 ,5.3 ,,6 km/h  (11 km/h),1023.4 hPa , aucune
9 h,8/8, ,3.8 km,6.3 °C,99% ,6.3 ,5.2 ,,6 km/h  (9 km/h),1023.2 hPa , aucune
8 h,8/8, ,3.3 km,6.2 °C,99% ,6.2 ,5.4 ,,5 km/h  (8 km/h),1022.7 hPa , aucune
7 h,8/8, ,3.2 km,6.2 °C,99% ,6.2 ,5.8 ,,4 km/h  (9 km/h),1022.3 hPa , aucune
6 h,8/8, ,3.6 km,6.2 °C,99% ,6.2 ,5.4 ,,5 km/h  (9 km/h),1022.2 hPa , aucune
5 h,8/8, ,3.4 km,6.3 °C,99% ,6.3 ,5.9 ,,4 km/h  (8 km/h),1022.3 hPa , aucune
4 h,8/8, ,2.9 km,6.4 °C,99% ,6.4 ,5.6 ,,5 km/h  (7 km/h),1022.7 hPa , aucune
3 h,8/8, ,3.8 km,6.5 °C,98% ,6.5 ,6.1 ,,4 km/h  (8 km/h),1022.8 hPa , aucune
```

Figure 1 : weather data converted to CSV format

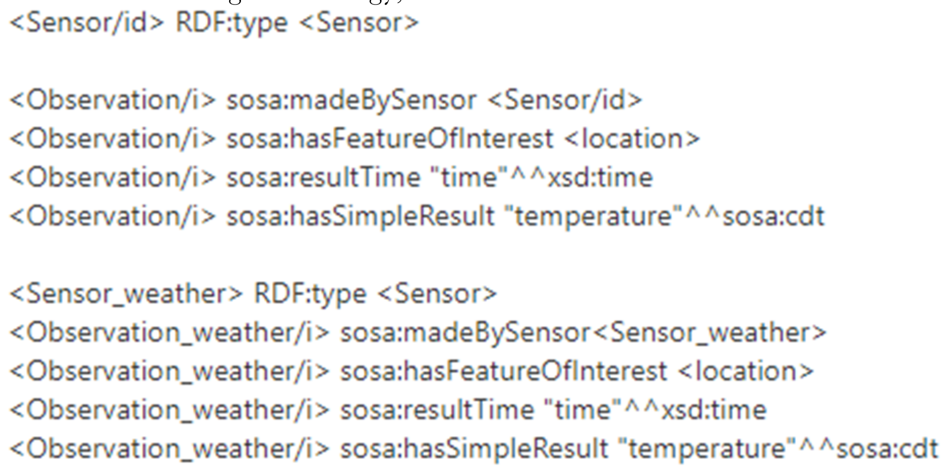
- Sensor data collected at the building of Espace Fauriel for one day in CSV format.



```
name,time,HMDT,LUMI,SND,SNDF,SNDM,TEMP,id,location,type
metrics,1636954288932759918,,,,,,19.9,03f5ca58-aa70-47b3-980c-c8f486cac9ee,emse/fayol/e4/S431H,sensors
metrics,1636954288981725847,54.7,,,,,,03f5ca58-aa70-47b3-980c-c8f486cac9ee,emse/fayol/e4/S431H,sensors
metrics,1636954289024561296,54.7,,,,,,03f5ca58-aa70-47b3-980c-c8f486cac9ee,emse/fayol/e4/S431H,sensors
metrics,1636954289347273983,,13,,,,,8aa60f58-fc6f-49e2-a53a-f5cc96bb9021,emse/fayol/e4/S423,sensors
metrics,1636954289348167304,,13,,,,,8aa60f58-fc6f-49e2-a53a-f5cc96bb9021,emse/fayol/e4/S423,sensors
metrics,1636954289399082163,,6,,,,,7b051d3a-8547-463d-9d28-a2d50c5098b4,emse/fayol/e4/S421,sensors
metrics,1636954289425219243,,6,,,,,7b051d3a-8547-463d-9d28-a2d50c5098b4,emse/fayol/e4/S421,sensors
metrics,1636954289482788946,,,,,,23.12,7b051d3a-8547-463d-9d28-a2d50c5098b4,emse/fayol/e4/S421,sensors
metrics,1636954289508732974,,,,,,23.12,7b051d3a-8547-463d-9d28-a2d50c5098b4,emse/fayol/e4/S421,sensors
metrics,1636954289557888244,,,,,,19.31,8aa60f58-fc6f-49e2-a53a-f5cc96bb9021,emse/fayol/e4/S423,sensors
metrics,1636954289583950427,,,,,,19.31,8aa60f58-fc6f-49e2-a53a-f5cc96bb9021,emse/fayol/e4/S423,sensors
metrics,1636954289608151791,,8,,,,,88cb0522-478a-456c-b63b-9c402b5e03b2,emse/fayol/e4/S422,sensors
metrics,1636954289634172258,,8,,,,,88cb0522-478a-456c-b63b-9c402b5e03b2,emse/fayol/e4/S422,sensors
metrics,1636954289658108220,,,,,,26.3,88cb0522-478a-456c-b63b-9c402b5e03b2,emse/fayol/e4/S422,sensors
metrics,1636954289684451038,,,,,,26.3,88cb0522-478a-456c-b63b-9c402b5e03b2,emse/fayol/e4/S422,sensors
metrics,1636954289708556949,27.3,,,,,,88cb0522-478a-456c-b63b-9c402b5e03b2,emse/fayol/e4/S422,sensors
metrics,1636954289734685841,27.3,,,,,,88cb0522-478a-456c-b63b-9c402b5e03b2,emse/fayol/e4/S422,sensors
metrics,1636954289759062665,,6,,,,,70345659-3f50-49af-98e7-bbc93961df92,emse/fayol/e4/S425,sensors
metrics,1636954289784967478,,6,,,,,70345659-3f50-49af-98e7-bbc93961df92,emse/fayol/e4/S425,sensors
metrics,1636954289809086892,,,,,,20.4,70345659-3f50-49af-98e7-bbc93961df92,emse/fayol/e4/S425,sensors
metrics,1636954289835137406,,,,,,20.4,70345659-3f50-49af-98e7-bbc93961df92,emse/fayol/e4/S425,sensors
metrics,1636954289859107690,49.1,,,,,,70345659-3f50-49af-98e7-bbc93961df92,emse/fayol/e4/S425,sensors
```

IV.2 Defining Sensor Ontology

Before converting the csv files to a turtle format we should first define the sensor ontology, to do this, we used the semantic Sensor Network https://www.w3.org/TR/vocab-ssn/ described by w3c.
After understanding the ontology, we came out with this result:

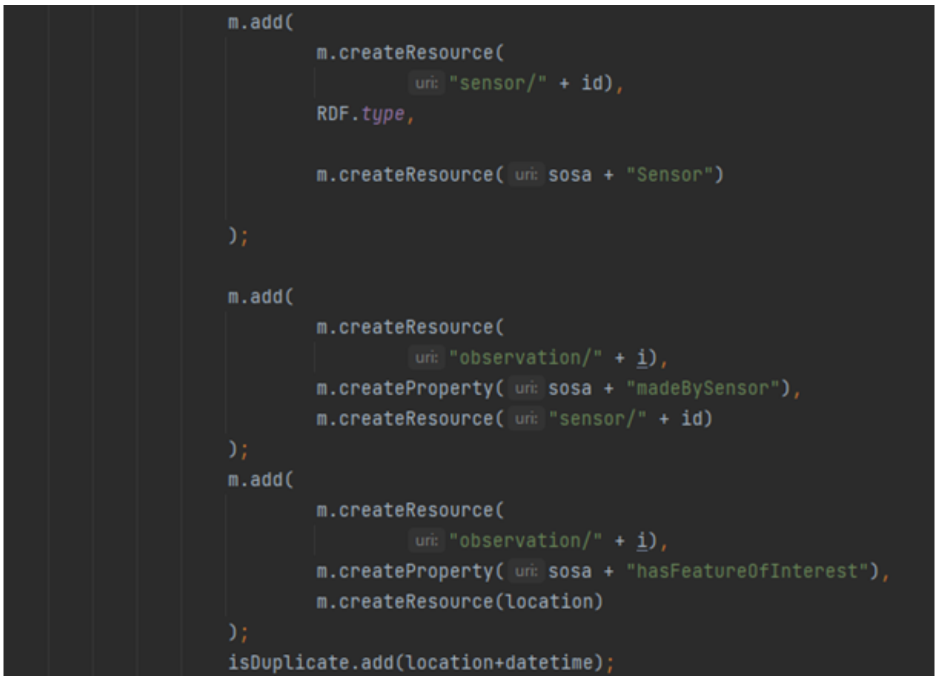


```
<Sensor/id> RDF:type <Sensor>

<Observation/i> sosa:madeBySensor <Sensor/id>
<Observation/i> sosa:hasFeatureOfInterest <location>
<Observation/i> sosa:resultTime "time"^^xsd:time
<Observation/i> sosa:hasSimpleResult "temperature"^^sosa:cdt

<Sensor_weather> RDF:type <Sensor>
<Observation_weather/i> sosa:madeBySensor<Sensor_weather>
<Observation_weather/i> sosa:hasFeatureOfInterest <location>
<Observation_weather/i> sosa:resultTime "time"^^xsd:time
<Observation_weather/i> sosa:hasSimpleResult "temperature"^^sosa:cdt
```

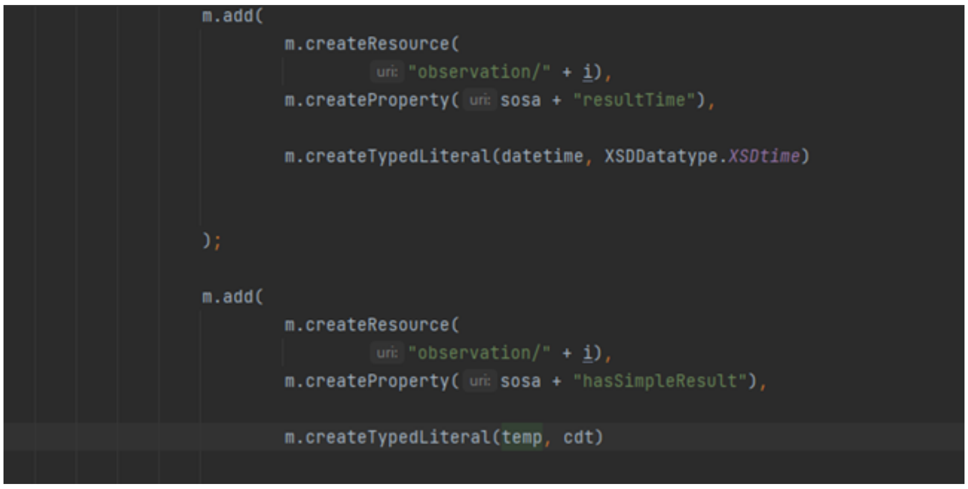
In fact, each room in the building has a sensor with an id, this sensor has an observation each ms with a value in temperature, and humidity… meanwhile we have one sensor outside, with many observations of the weather temperature, humidity, with a value of these temperature each hour.

IV.3 Converting from Csv to Ttl using java

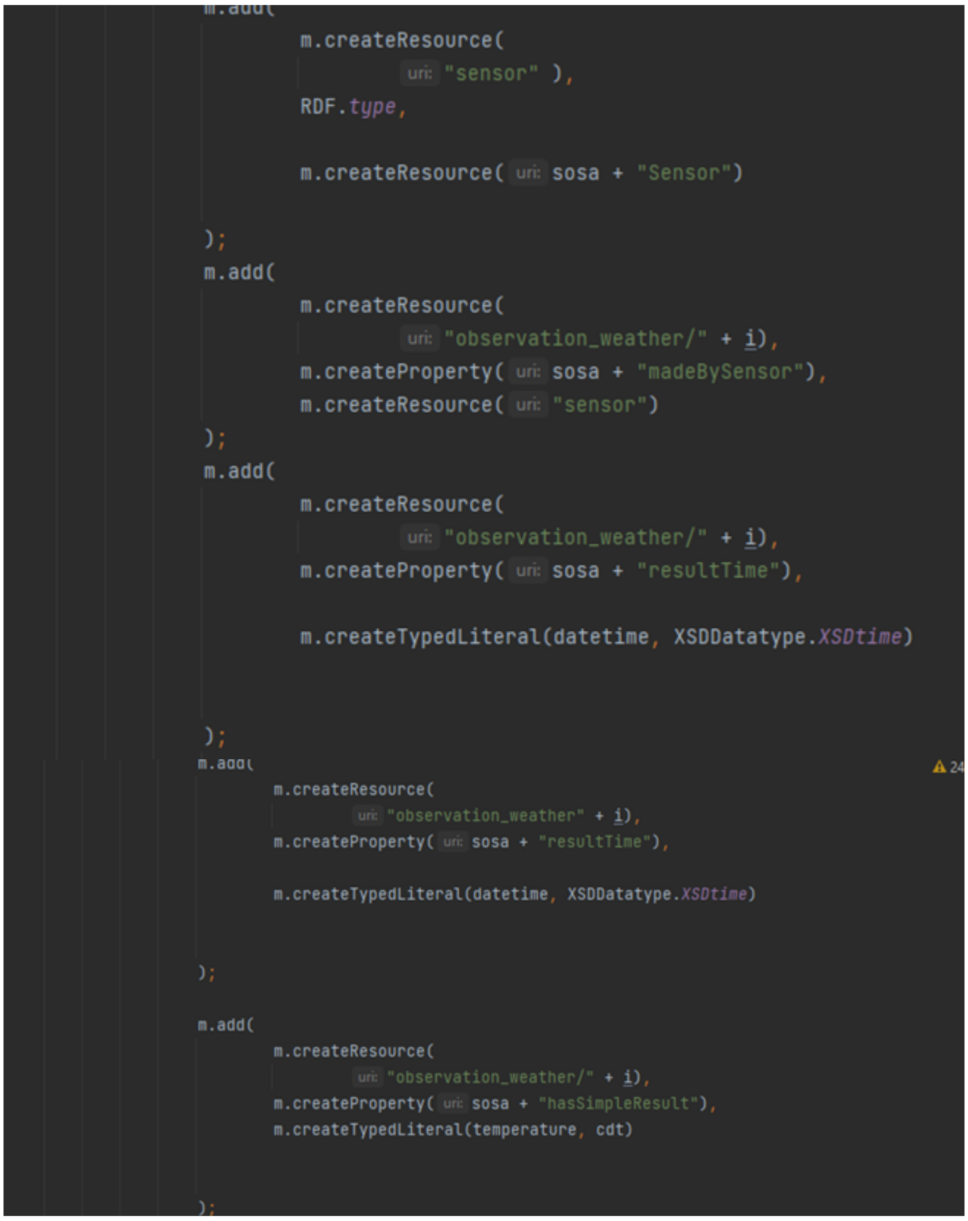
To be able to describe this ontology in java, we used the Apache Jena dependence in maven. As consequence, we can create a model with a subject, predicate and object. The code below describes this ontology in java.



```
m.add(
        m.createResource(
                uri: "sensor/" + id),
        RDF.type,

        m.createResource( uri: sosa + "Sensor")

);

m.add(
        m.createResource(
                uri: "observation/" + i),
        m.createProperty( uri: sosa + "madeBySensor"),
        m.createResource( uri: "sensor/" + id)
);
m.add(
        m.createResource(
                uri: "observation/" + i),
        m.createProperty( uri: sosa + "hasFeatureOfInterest"),
        m.createResource(location)
);
isDuplicate.add(location+datetime);
```



```
m.add(
        m.createResource(
                uri: "observation/" + i),
        m.createProperty( uri: sosa + "resultTime"),

        m.createTypedLiteral(datetime, XSDDatatype.XSDtime)


);

m.add(
        m.createResource(
                uri: "observation/" + i),
        m.createProperty( uri: sosa + "hasSimpleResult"),

        m.createTypedLiteral(temp, cdt)
```



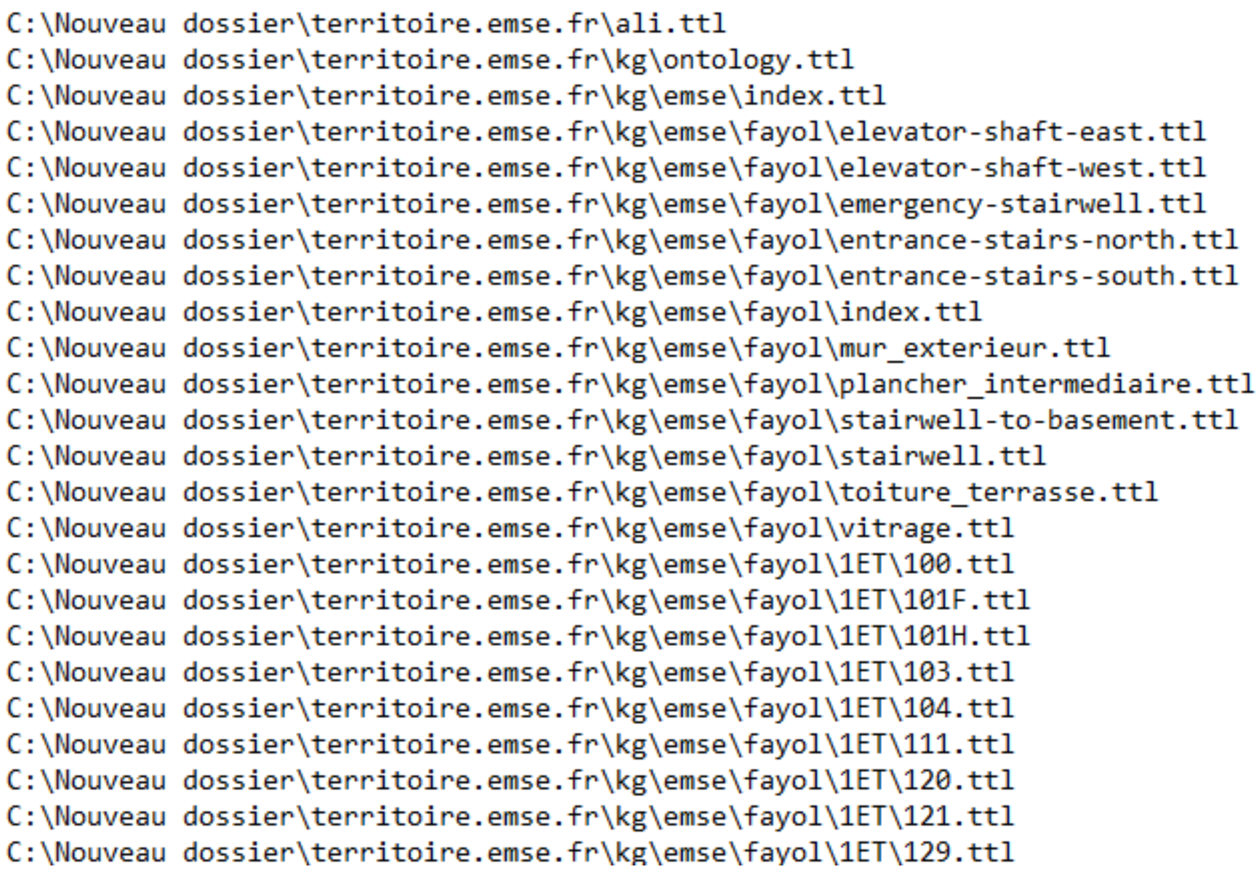
```
m.add(
        m.createResource(
                uri: "sensor" ),
        RDF.type,

        m.createResource( uri: sosa + "Sensor")

);
m.add(
        m.createResource(
                uri: "observation_weather/" + i),
        m.createProperty( uri: sosa + "madeBySensor"),
        m.createResource( uri: "sensor")
);
m.add(
        m.createResource(
                uri: "observation_weather/" + i),
        m.createProperty( uri: sosa + "resultTime"),

        m.createTypedLiteral(datetime, XSDDatatype.XSDtime)


);
m.add(
        m.createResource(
                uri: "observation_weather" + i),
        m.createProperty( uri: sosa + "resultTime"),

        m.createTypedLiteral(datetime, XSDDatatype.XSDtime)


);

m.add(
        m.createResource(
                uri: "observation_weather/" + i),
        m.createProperty( uri: sosa + "hasSimpleResult"),
        m.createTypedLiteral(temperature, cdt)


);
```

There are around 1 millions line in the sensor csv file ,so we tried to minimize these rows, by taking the average value per hour ,since the values were taken each ms,we also had to eliminate rows where value of temperature was empty.To combine the csv files concerning the sensor data of the building and weather data, we used a sparql query.

To efficiently import the RDF data that describe the building and store it in the triplestore we use the comande :

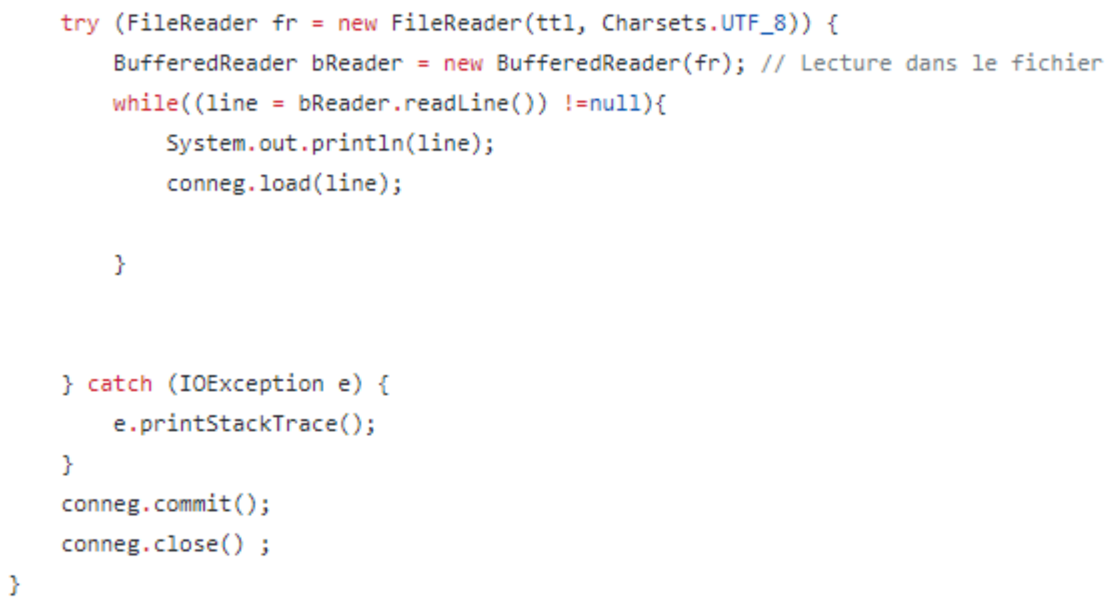
```
wget -r --no-parent --reject "index.html*" https://territoire.emse.fr/kg/
```

To dowloand all the data files recursively in the repesetory of our project, and the with the command Find we store all the links to the turtles files that we want to import in our triplstore in a file as we can see bellow :



```
C:\Nouveau dossier\territoire.emse.fr\ali.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\ontology.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\index.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\elevator-shaft-east.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\elevator-shaft-west.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\emergency-stairwell.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\entrance-stairs-north.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\entrance-stairs-south.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\index.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\mur_exterieur.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\plancher_intermediaire.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\stairwell-to-basement.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\stairwell.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\toiture_terrasse.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\vitrage.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\1ET\100.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\1ET\101F.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\1ET\101H.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\1ET\103.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\1ET\104.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\1ET\111.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\1ET\120.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\1ET\121.ttl
C:\Nouveau dossier\territoire.emse.fr\kg\emse\fayol\1ET\129.ttl
```

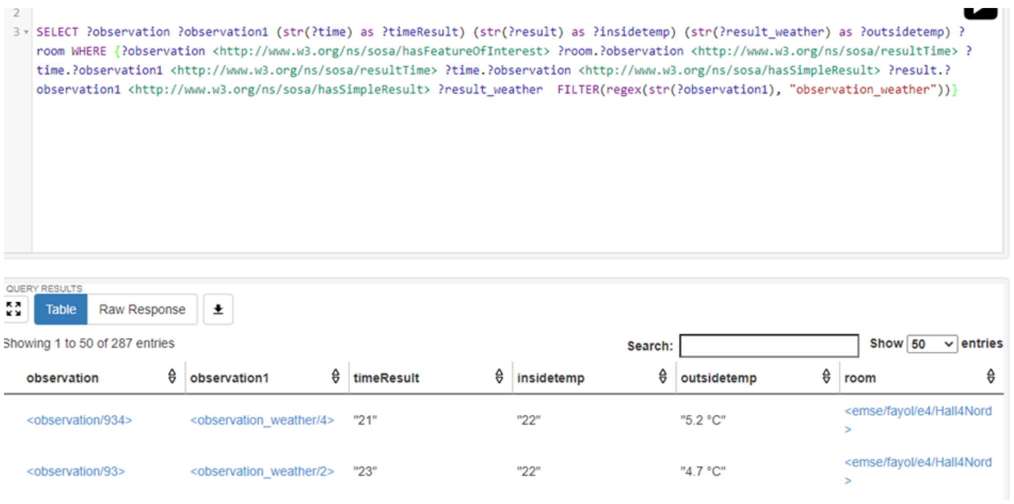
Then we establish a connection to the Fuseki serveur and start importing the turtles filesas we can see in the code bellow:



```
        try (FileReader fr = new FileReader(ttl, Charsets.UTF_8)) {
            BufferedReader bReader = new BufferedReader(fr); // Lecture dans le fichier
            while((line = bReader.readLine()) !=null){
                System.out.println(line);
                conneg.load(line);


            }



        } catch (IOException e) {
            e.printStackTrace();
        }
        conneg.commit();
        conneg.close() ;
    }
```

IV.4 SPARQL Queries

The Sparql query below, allow to link between the csv file of the building rooms, and outside based on time, and shows the outside and inside temperature of each room.



```
SELECT ?observation ?observation1 (str(?time) as ?timeResult) (str(?result) as ?insidetemp) (str(?result_weather) as ?outsidetemp) ?
room WHERE {?observation <http://www.w3.org/ns/sosa/hasFeatureOfInterest> ?room.?observation <http://www.w3.org/ns/sosa/resultTime> ?
time.?observation1 <http://www.w3.org/ns/sosa/resultTime> ?time.?observation <http://www.w3.org/ns/sosa/hasSimpleResult> ?result.?
observation1 <http://www.w3.org/ns/sosa/hasSimpleResult> ?result_weather  FILTER(regex(str(?observation1), "observation_weather"))}
```

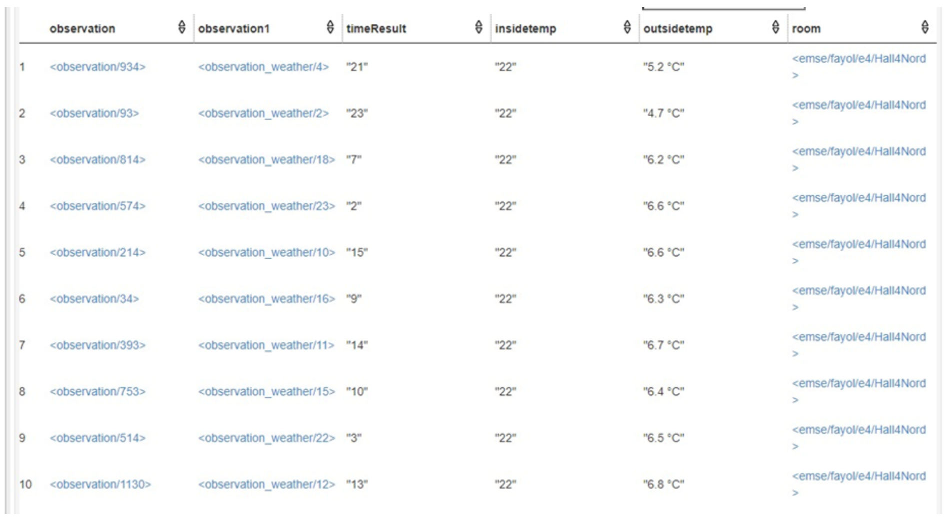
QUERY RESULTS

Showing 1 to 50 of 287 entries

Search: Show 50 entries

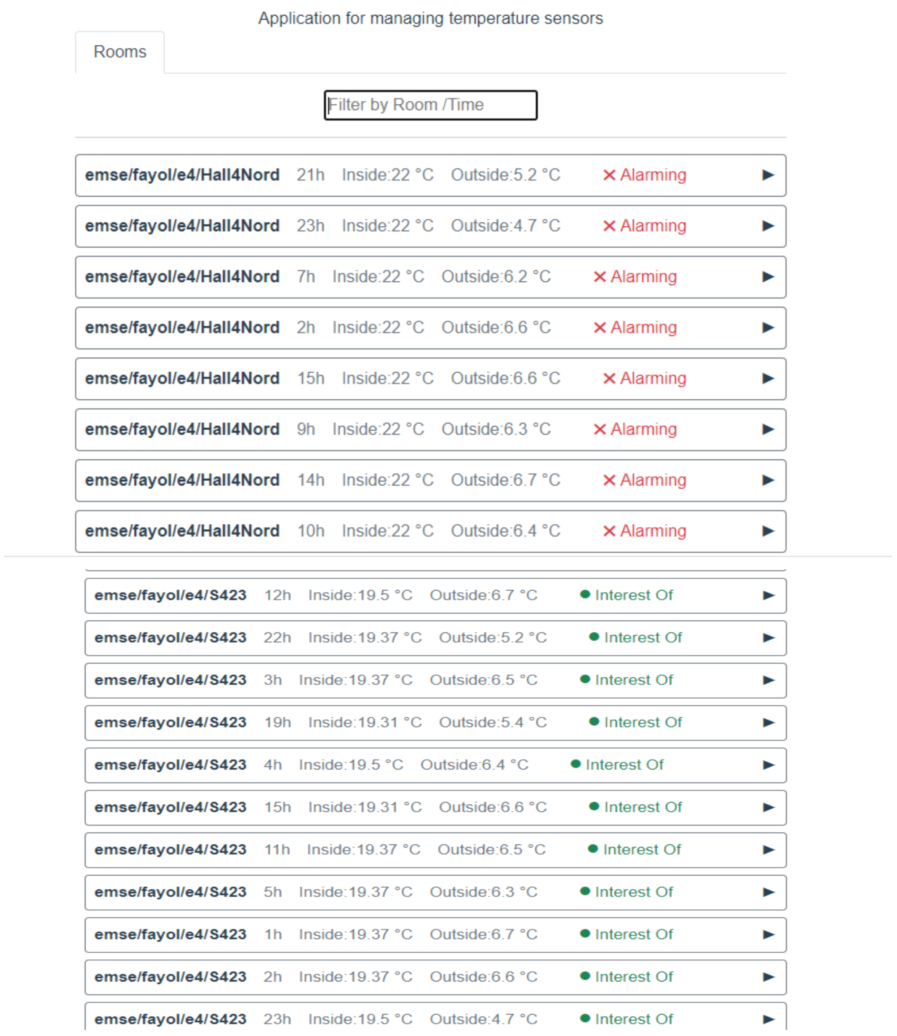
observation	observation1	timeResult	insidetemp	outsidetemp	room
<observation/934>	<observation_weather/4>	"21"	"22"	"5.2 °C"	<emse/fayol/e4/Hall4Nord >
<observation/93>	<observation_weather/2>	"23"	"22"	"4.7 °C"	<emse/fayol/e4/Hall4Nord >

As result we have:



	observation	observation1	timeResult	insidetemp	outsidetemp	room
1	<observation/934>	<observation_weather/4>	"21"	"22"	"5.2 °C"	<emse/fayol/e4/Hall4Nord >
2	<observation/93>	<observation_weather/2>	"23"	"22"	"4.7 °C"	<emse/fayol/e4/Hall4Nord >
3	<observation/814>	<observation_weather/18>	"7"	"22"	"6.2 °C"	<emse/fayol/e4/Hall4Nord >
4	<observation/574>	<observation_weather/23>	"2"	"22"	"6.6 °C"	<emse/fayol/e4/Hall4Nord >
5	<observation/214>	<observation_weather/10>	"15"	"22"	"6.6 °C"	<emse/fayol/e4/Hall4Nord >
6	<observation/34>	<observation_weather/16>	"9"	"22"	"6.3 °C"	<emse/fayol/e4/Hall4Nord >
7	<observation/393>	<observation_weather/11>	"14"	"22"	"6.7 °C"	<emse/fayol/e4/Hall4Nord >
8	<observation/753>	<observation_weather/15>	"10"	"22"	"6.4 °C"	<emse/fayol/e4/Hall4Nord >
9	<observation/514>	<observation_weather/22>	"3"	"22"	"6.5 °C"	<emse/fayol/e4/Hall4Nord >
10	<observation/1130>	<observation_weather/12>	"13"	"22"	"6.8 °C"	<emse/fayol/e4/Hall4Nord >

VI. HMI(Human machine Interface) model


Application for managing temperature sensors

Rooms

Filter by Room /Time

emse/fayol/e4/Hall4Nord	21h	Inside:22 °C	Outside:5.2 °C	✕ Alarming	►
emse/fayol/e4/Hall4Nord	23h	Inside:22 °C	Outside:4.7 °C	✕ Alarming	►
emse/fayol/e4/Hall4Nord	7h	Inside:22 °C	Outside:6.2 °C	✕ Alarming	►
emse/fayol/e4/Hall4Nord	2h	Inside:22 °C	Outside:6.6 °C	✕ Alarming	►
emse/fayol/e4/Hall4Nord	15h	Inside:22 °C	Outside:6.6 °C	✕ Alarming	►
emse/fayol/e4/Hall4Nord	9h	Inside:22 °C	Outside:6.3 °C	✕ Alarming	►
emse/fayol/e4/Hall4Nord	14h	Inside:22 °C	Outside:6.7 °C	✕ Alarming	►
emse/fayol/e4/Hall4Nord	10h	Inside:22 °C	Outside:6.4 °C	✕ Alarming	►
emse/fayol/e4/S423	12h	Inside:19.5 °C	Outside:6.7 °C	● Interest Of	►
emse/fayol/e4/S423	22h	Inside:19.37 °C	Outside:5.2 °C	● Interest Of	►
emse/fayol/e4/S423	3h	Inside:19.37 °C	Outside:6.5 °C	● Interest Of	►
emse/fayol/e4/S423	19h	Inside:19.31 °C	Outside:5.4 °C	● Interest Of	►
emse/fayol/e4/S423	4h	Inside:19.5 °C	Outside:6.4 °C	● Interest Of	►
emse/fayol/e4/S423	15h	Inside:19.31 °C	Outside:6.6 °C	● Interest Of	►
emse/fayol/e4/S423	11h	Inside:19.37 °C	Outside:6.5 °C	● Interest Of	►
emse/fayol/e4/S423	5h	Inside:19.37 °C	Outside:6.3 °C	● Interest Of	►
emse/fayol/e4/S423	1h	Inside:19.37 °C	Outside:6.7 °C	● Interest Of	►
emse/fayol/e4/S423	2h	Inside:19.37 °C	Outside:6.6 °C	● Interest Of	►
emse/fayol/e4/S423	23h	Inside:19.5 °C	Outside:4.7 °C	● Interest Of	►

As we can see above, this represents our main interface which display the rooms of the building , the inside temperature and outside temperature, at specific hour, then based on the difference between the inside and outside temperature:

- If the difference > 15: It shows Alarming
- If the difference <15 and >12: It shows Interesting Of
- If the difference <12: It shows Normal

The user can also filters by Name and time.

Here is the link to a demo video demonstrates how the application works:
https://www.youtube.com/watch?v=so_P1g99Gmk

VII. Conclusion

The project is an application that integrates geospatial data from multiple sources, including dynamic data. The ontology used during this project defines a knowledge model describing the categories and properties of the data stored in the triplestore. The information and structured data displayed on the web pages comes from the Fuseki triplestore using Sparql queries processed and manipulated through Spring-boot.

This project allowed us to have a very applied experience in semantic web, especially with regard to project management, given that we took our project from the blank sheet to the final rendering. This work helped us to develop our personal knowledge and introduced us to new tools.

Finally, we are convinced that carrying out this work is a very fruitful experience for our practical and academic training. The theoretical knowledge that we have acquired throughout our training has been perfectly consolidated. At the same time, this project constitutes a very good human and social experience through remote meetings.

VIII. References

https://www.emse.fr/~zimmermann/Teaching/SemWeb
/https://www.w3.org/TR/turtle/
https://jena.apache.org/