Institut für raumbezogene
Informations- und Messtechnik
Fachhochschule Mainz

4th Year engineering internship

Internship subject: Developer on GeoTimeWFS project

Name and surname of the student: MAZUREK Pierre

Date of internship: 26/04/2021 to 23/07/2021

Promotion: 4A IE QUANTUM

PROJECT :

Name of the company: i3Mainz - Hochschule Mainz - University of Applied Sciences

Address: Lucy-Hillebrand-Straße 2, 55128 MAINZ ALLEMAGNE

Name and surname of the training supervisor: Claire PRUDHOMME

First and last name of the teacher tutor: Jean-Marie BILBAULT

Table of contents

Table of figures

Acknowledgements

First of all, I want to thanks several persons for their help and involvement in my internship.

Claire PRUDHOMME, my internship tutor who gave her precious time as a young mother to help me on technical points, who integrated me well within I3MAINZ and who knew how to trust me in the realization of the tasks I was asked.

Jean-Jacques PONCIANO, I3Mainz member who replaced Claire during her maternity leave. He always helped me when I had technical problems and knew, although involved in other projects related to the laboratory, to remain available for Christopher and me.

Christopher BLARD, student at ESIREM and internship partner, with whom I was able to perfect my teamwork and create a work strategy leading to the completion of the requested tasks.

Jean-Marie BILBAULT, my tutor at ESIREM.

Falk WÜRRIEHAUSEN, and all the other members of BKG for their trust, their listening and their sympathy.

And all the members of I3Mainz that I could see during all the meetings that we could make with them, for the efforts that they produced to know better Christopher and me, and to have spoken English so that we could understand them.

Introduction

During my internship, my missions were to get acquainted with the project in progress and to participate in its advancement in order to respect a delivery date of the final prototype planned for August.

This project is realized by the laboratory I3Mainz, based in Germany to answer an order of the company BKG. It allows essentially to manage geospatial data. It also works on search and queries on ontologies which will be explained later in this report.

I was put in complete autonomy on the project, in pair of work with Christopher BLARD another student of ESIREM.

At first, this report will detail all the aspects of the project, by presenting the actors of this one as well as a clear explanation of the various functionalities wished by the customer. It will also highlight all the techniques used to build the project.

In a second step, I will present my role in this project. I will start by introducing my work rhythm, then the different concepts and notions that I had to learn to be able to correctly intervene on the project. Then, I will detail the different tasks I performed on the project. I will also talk about what this internship brought me in relation to my professional ambitions.

I – Company presentations

I.1 – Introduction

The GeoTimeWFS project was ordered by the company BKG in order to have a prototype with several interfaces to manage different tools related to metadata and geospatial data.

The development of the application was started about a year and a half ago, being the title of an internship of a student of ESIREM last year, Boris BORDEAUX who worked on the implementation of the first features of the application.

Christopher and I are involved in the finalization of this project, which must be delivered to the client at the end of August. We have to realize mainly tasks related to the formatting of the application. We also need to add some functionalities concerning the data processing itself.

I.2 – Presentation of the main actors of the project

I.2.1 – I3 Mainz

The I3Mainz Laboratory (logo is in *figure 1*), which is an institute for spatial information and measurement technology, is directly affiliated to the University of Applied Sciences in Mainz. This institution is engaged in research and development tasks as well as technology transfer, mainly in the spatial fields of 3D measurement technology, image analysis, geo-data infrastructures, optical technologies, positioning and navigation, semantic modeling, spatial humanities and visualization.

The institute was founded in February 1998 and is organized by several professors of the University of Applied Sciences Mainz, who also led the respective projects. The vast majority of these projects are interdisciplinary and are carried out in close cooperation with public and private partners.

The two persons within I3Mainz dealing with the GeoTimeWFS application are Claire Prudhomme who is the main person in charge of the project, owner of the GitHub code of the application. Jean-Jacques Ponciano also intervenes on the project where he manages mainly everything that concerns the semantic web.

Figure 1 : I3Mainz logo

I.2.1 – BKG

The Bundesamt für Kartographie und Geodäsie, or simply BKG, whose logo is in *figure 2*, is the main service provider of the federal government for all matters concerning topographic base data, geodetic reference systems and cartography.

BKG manages all its federal reference data collections and exchanges according to the Federal Georeferencing Data Act (BGeoRG). It is a specialized authority within the Federal Ministry of the Interior, Construction and Housing with specialized departments for geo-information and geodesy.

Its main tasks are to ensure the correct application of a uniform coordinate system for the whole of Germany. It is also responsible for providing current official geodata of Germany via internet services. It aims to make it possible for every citizen to search, find and use the spatial data of the German federal, state and local governments.

Bundesamt für
Kartographie und Geodäsie

Figure 2 : BKG logo

II – The GeoTimeWFS project

II.1 – Context

First of all, it is important to know that BKG is a general company that includes several branches specialized in specific fields. Each branch follows its own development and carries out its own projects that may or may not be linked to each other. The branch of BKG that manages metadata and other geospatial data has called upon I3 Mainz in order to obtain a prototype working with the semantic web.

II.2 – Concept used in the project

II.2.1 – Semantic Web

Today, the semantic web is taking an increasingly important place in data processing (Cousin, 2008) since it simplifies the way data is expressed. Indeed, it is based on a descriptive language called RDF (Resource Definition Framework) which was designed by the web standardization organization, the W3C [1]. This format describes the data according to a system of triples which functions in the following way (*figure 3*) :

Subject	Object	Predicate

Figure 3 : RDF format

This makes it possible to link these data between several applications in order to better organize them and to be able to use them more easily. Each RDF triplet has a URI identifier.

The interest of the semantic web is to bring a definition to the data in order to be able to use and manipulate them from any computer. All these triples are stored and linked together in what is called an ontology. The project includes several ontologies.

II.2.2 – Model-view-controller architecture

The project uses a particular software architecture that is composed of 3 distinct entities that each have their role to play in the operation of the application [2] like shown in *figure 4* :

Model	**View**	**Controller**
Contains the data used by the application for the display	Graphical interface that manages the data display, from which the methods are called	Set of methods usable by the application, called by the view and acts on the model

Figure 4 : MVC caracteristics

The interest of this architecture is that it facilitates the implementation of user interfaces, which are interfaces in which the user can act on the data of the application.

For the project, the views were developed in HTML, including the use of .CSS and JavaScript files. The controllers and models are coded in Java.

II.2.3 – Spring BOOT

The project was created with a framework called Spring Boot (*figure 5* represents the logo). It allows to create and organize the structure of a Java application and is Open Source. Within this application it allows to facilitate the development of the different entities and to manage the tests.

To go into a little more detail, we say that Spring is a lightweight container [3]. Indeed, it allows to create objects and to put them in relation with each other thanks to a configuration file. This file describes the objects to be implemented in the application and specifies their dependency relationships. Each model, view or controller is located in a specific folder, recognized by Spring.

Figure 5 : Spring Boot logo

II.3 – Details of the project

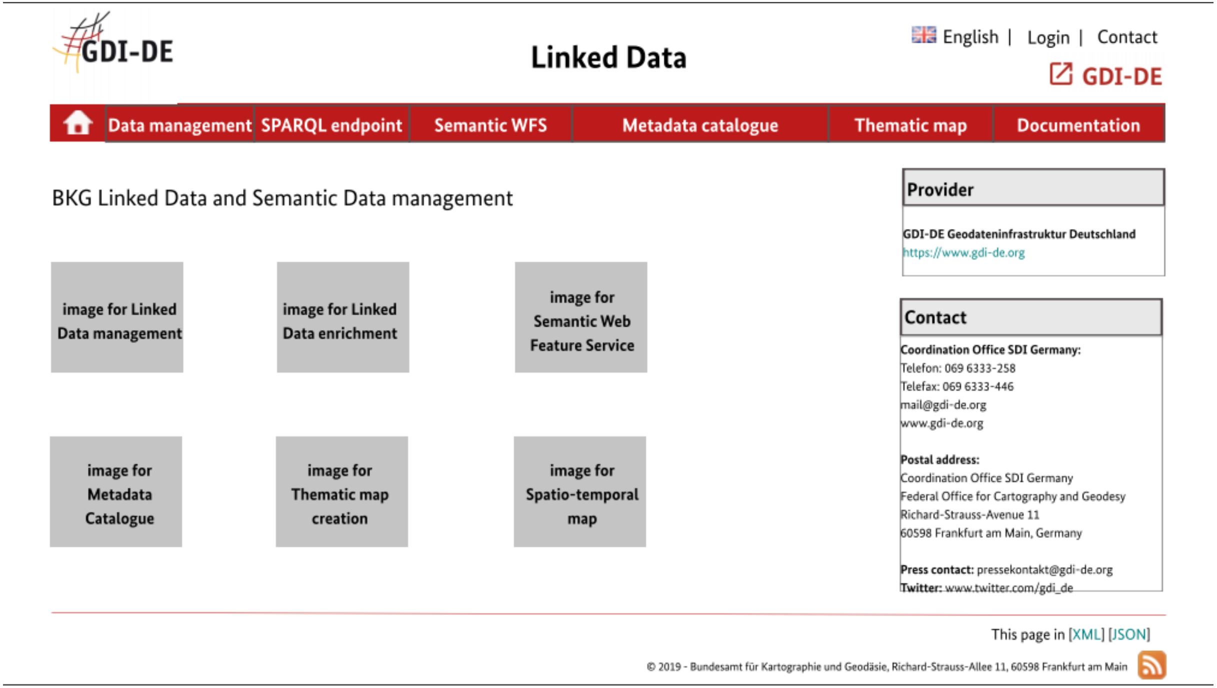
II.3.1 – Global presentation

This project aims at enriching spatial data with linked data [4] (W3C structured data). This will allow that metadata, which are data giving information about data, are no longer collected via a record but by RDF triplet in a knowledge base.

The other main objective is to implement a data suggestion system to enrich this knowledge base in order to make the management of geospatial data easier. The application will also dynamically generate maps from this data enrichment.

II.3.2 – Desired application

The desired application was clearly defined by the actors of this project. It consists of a multitude of views, each corresponding to a specific functionality of the application. These functionalities are divided in the form of a navigation bar tab. Each tab also has sub-menus. Here is the interface of the home page of the application, here *figure 6*.


GDI-DE

Linked Data

English | Login | Contact

GDI-DE

Data management | SPARQL endpoint | Semantic WFS | Metadata catalogue | Thematic map | Documentation

BKG Linked Data and Semantic Data management

image for Linked Data management

image for Linked Data enrichment

image for Semantic Web Feature Service

image for Metadata Catalogue

image for Thematic map creation

image for Spatio-temporal map

Provider

GDI-DE Geodateninfrastruktur Deutschland
https://www.gdi-de.org

Contact

Coordination Office SDI Germany:
Telefon: 069 6333-258
Telefax: 069 6333-446
mail@gdi-de.org
www.gdi-de.org

Postal address:
Coordination Office SDI Germany
Federal Office for Cartography and Geodesy
Richard-Strauss-Avenue 11
60598 Frankfurt am Main, Germany

Press contact: pressekontakt@gdi-de.org
Twitter: www.twitter.com/gdi_de

This page in [XML] [JSON]

© 2019 - Bundesamt für Kartographie und Geodäsie, Richard-Strauss-Allee 11, 60598 Frankfurt am Main

Figure 6 : Desired interface of the application homepage

Let's detail each tab, to better understand the shape of the application:

The **Home** menu presents the home page of the application, with images for each menu that will be chosen downstream.

The **Data Management** menu concerns everything related to the processing of data used by the application. In particular, it offers a sub-menu for importing shapefile files, which are vector data files used to characterize a geographical entity. It has a sub-menu that also offers other import tools already created by I3Mainz. There are also two views of this menu corresponding to the verification of the imported data and the other to the enrichment of this data via remotely obtainable data, a functionality that I will discuss in more detail later in this document. Finally, it contains two sub-menus corresponding to the import and verification of mapping or XSD schema types. These schema types help define the structure and type of content of an XML document.

The **SPARQL endpoint** menu will allow it to perform SPARQL queries on the application's ontology [5]. SPARQL allows to retrieve data stored in the form of triples, on a local or remote repository, we will come back to this later.

The project also includes a **Semantic WFS** menu which is divided into its home page, a configuration page and an ontology browser. This corresponds to another project developed upstream that works with semantic web.

Then there is a **Metadata Catalogue** menu which presents three views. The first one is the Semantic OGC API Records home page which presents documentation about the application. There is also a view that allows you to import metadata and a view that allows you to link that metadata to data in order to match it.

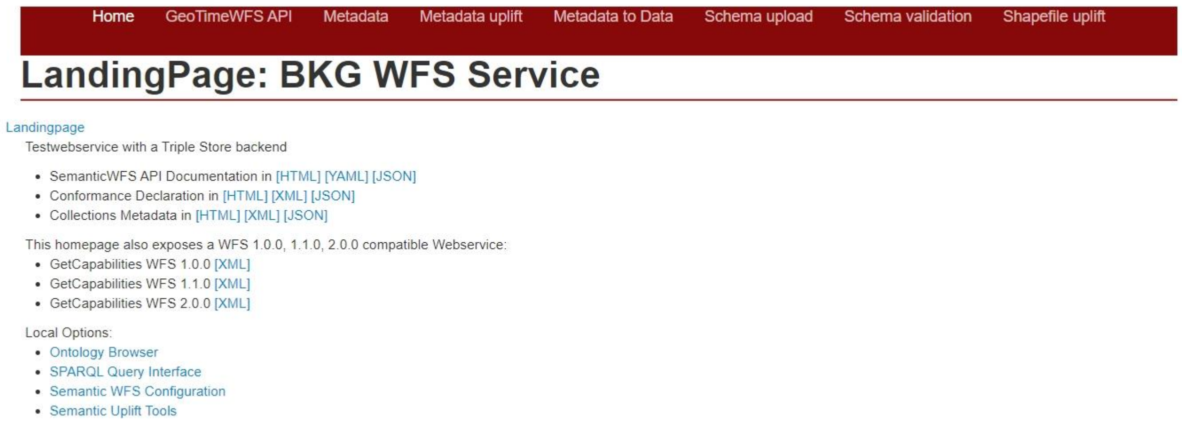
It also has a **Thematic map** menu which includes a submenu for creating a map in which we have different options on what we want to display on this map. Then there is a sub-menu that represents an example map with all the spatial data used to create it. There is also a last view that allows us to find all the maps created and saved with the application.

The last menu **Documentation** provides an important documentation on all the functionalities of the application, on the project report and on the documentation of the ontology of the application.

III – Missions and tasks

III.1 – Introduction

As said before, Christopher's and my internship started in the final phase of the project. Thus, the majority of the views, methods, and features have already been implemented. *Figure 7* shows the interface of the application at the beginning of our work.



LandingPage: BKG WFS Service

Landingpage
Testwebservice with a Triple Store backend

- SemanticWFS API Documentation in [HTML] [YAML] [JSON]
- Conformance Declaration in [HTML] [XML] [JSON]
- Collections Metadata in [HTML] [XML] [JSON]

This homepage also exposes a WFS 1.0.0, 1.1.0, 2.0.0 compatible Webservice:
- GetCapabilities WFS 1.0.0 [XML]
- GetCapabilities WFS 1.1.0 [XML]
- GetCapabilities WFS 2.0.0 [XML]

Local Options:
- Ontology Browser
- SPARQL Query Interface
- Semantic WFS Configuration
- Semantic Uplift Tools

Figure 7 : Current interface of the application

At this stage of the project, there were still some important tasks to be done. First of all, the application must be formatted so that it looks like the prototype presented beforehand. There are also two features to add to the project: **enrichment** and **thematic map creation**, which correspond to two views of the **Data Management** menu and the **Thematic map** menu. Christopher focused on the creation of the thematic maps while I took care of the enrichment, which will be detail in the next section. The formatting of the application has been shared by working each one of us on some aspects of style.

III.2 – Presentation of the working condition

III.2.1 – Meetings and work rhythm

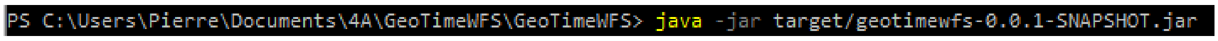
During this internship, it was necessary to assist to some meetings. Indeed, meetings were every 3 weeks, on Monday at 2 pm with the client of the project, BKG. The company was represented by Falk WÜRRIEHAUSEN. One of our internship supervisors, Claire PRUDHOMME and Jean-Jacques PONCIANO also attended these meetings. The goal was to present the work done and to keep the client informed of the progress of the project.

There were also regular meetings with our tutors in order to ask for their help with possible technical problems.

Finally, there was also the possibility to participate in stand-up meetings with all I3Mainz members, where everyone talks about the project he is working on.

III.2.2 – Operation and execution of the application

The files composing the application are available via the GitHub repository of the application: https://github.com/Cprudhomme/GeoTimeWFS. This project was created with Spring Boot and its models and controllers were written in Java. To have the possibility to work directly on the project, it is necessary to clone this repository directly into a local folder with Eclipse, which is a Java development environment. A new branch must be created to apply the local modification of the project and publish it in the git repository. To launch the application, the project must be run in a compiler (here Eclipse) where it builds a **.jar** (java executable file, generated by Spring). This file has to be run in Windows PowerShell using the following command (*figure 8*):



```
PS C:\Users\Pierre\Documents\4A\GeoTimeWFS\GeoTimeWFS> java -jar target/geotimewfs-0.0.1-SNAPSHOT.jar
```

Figure 8 : Running the .jar in PowerShell

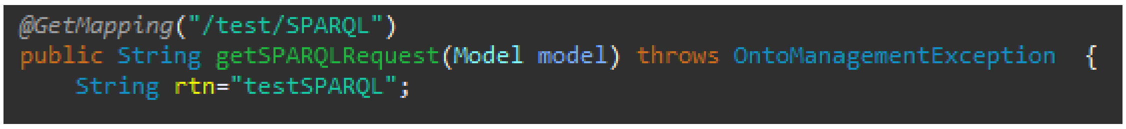
The project is then available at http://localhost:8080/. As soon as a modification is made, the whole process must be redone to see the modification on the displaying.

III.3 – Enrichment

III.3.1 – Creation of the view and the controller

In order to be able to work on the enrichment functionality, it is necessary to create a new controller and a new view to add the work done to the project.

It is important to know that Spring makes the methods present in the controllers work via @GetMapping queries, *figure 9* shows the query I created to perform SPARQL queries.



```
@GetMapping("/test/SPARQL")
public String getSPARQLRequest(Model model) throws OntoManagementException  {
    String rtn="testSPARQL";
```

Figure 9 : getMapping in controller

It permits to link the view to the controller like explained in *figure 10*. It is possible to give a variable of the controller to the view via the *model.addAttribute()* method.

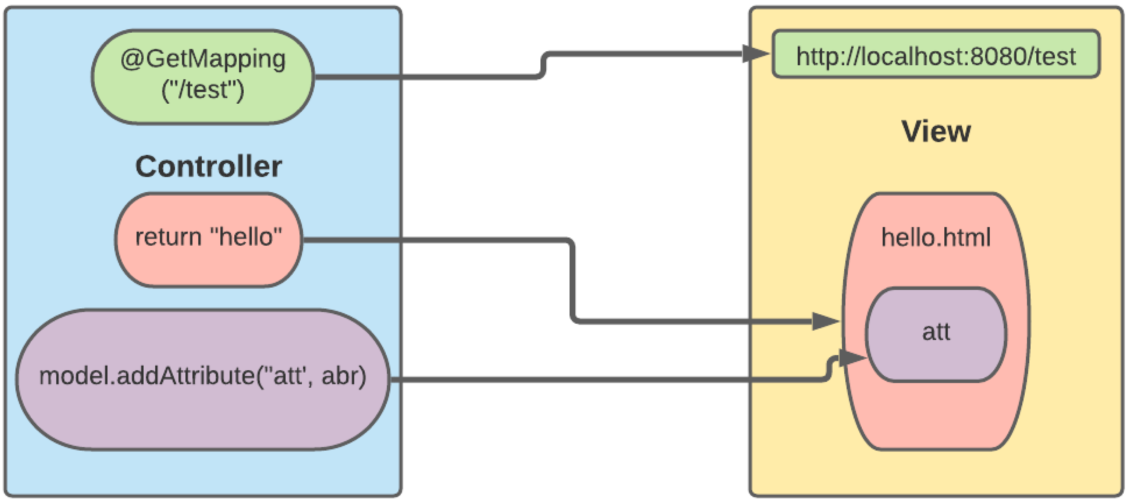
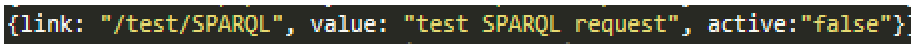


Figure 10 : link between view and controller with Spring Boot

The query created here will call the *getSPARQLRequest* method corresponding when the url http://localhost:8080/test/SPARQL/ is reached. The String *rtn* returned by this query must match the name of the view, here *testSPARQL.html*.

Once this is done, it is necessary to link this controller to *testSPARQL.html* and include this view in the application menu to make it accessible. To do this, *Thymeleaf*, which is an engine template is used in the project to create a navigation bar containing the views and their corresponding terminology. The navigation bar is called in all the views and it allows to access each of them.

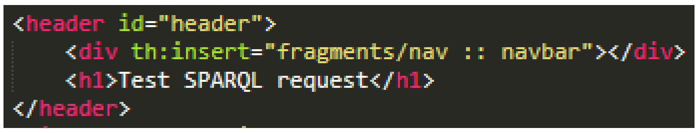
It is created from the *nav.html* file. It is in this file that the following line is added, shown in *figure 11* :



```
{link: "/test/SPARQL", value: "test SPARQL request", active:"false"}
```

Figure 11 : added the testSPARQL view to the navigation bar

After indicating the terminology of the desired .html link as well as the title to be displayed in the navigation bar then it is added to the list of other views and terminologies. In the *testSPARQL.html* view, the following lines shown in *figure 12* allowing to display the navigation bar in the header.



```
<header id="header">
    <div th:insert="fragments/nav :: navbar"></div>
    <h1>Test SPARQL request</h1>
</header>
```

Figure 12 : display of the navigation bar

As shown in *figure 13*, the view has been added to the navigation bar and it is now accessible.

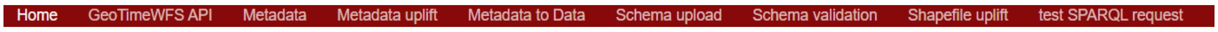


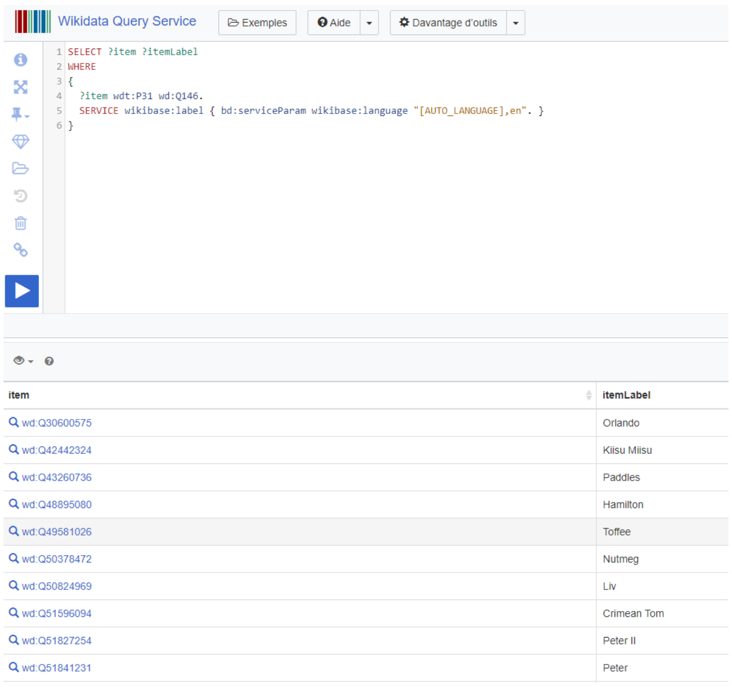
Figure 13 : modified navigation bar

III.3.2 – Remote SPARQL query

III.3.2.1 – Query execution

Enrichment is the act of refining the application's ontology data with data obtained from remote sites. To progress in the task, it is necessary to study how to make a remote SPARQL query to complete the enrichment. This is the first step of the work.

The site https://www.*Wikidata*.org/ which lists all the data of Wikipedia in the form of RDF triples [6]. This site has a SPARQL query service that allows to make queries directly on the data available on *Wikidata*. This service is available at https://query.*Wikidata*.org/. It works as follows in f:



```
SELECT ?item ?itemLabel
WHERE
{
  ?item wdt:P31 wd:Q146.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

item	itemLabel
wd:Q30600575	Orlando
wd:Q42442324	Kiisu Miisu
wd:Q43260736	Paddles
wd:Q48895080	Hamilton
wd:Q49581026	Toffee
wd:Q50378472	Nutmeg
wd:Q50824969	Liv
wd:Q51596094	Crimean Tom
wd:Q51827254	Peter II
wd:Q51841231	Peter

Figure 14 : example of results of a query

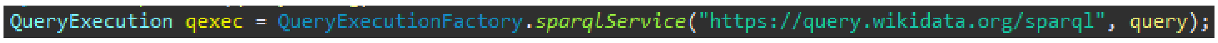
Figure 14 shows the results that can be obtained by searching, for example, for all the names of cats available on the site. The query is about to get the instance of the object and the name of the object. Then the property *wdt:P31* which correspond to the nature of an element, is called. It permits to return the instance of the obtained object and the property *wd:Q146* which corresponds to the domestic cats. The results are returned in the form of an array with two columns *item* and *itemLabel.*

After understanding the principle, the point is now about how to make these requests directly from the controller. To do this, it is necessary to use the *org.apache.jena.query* library which allows to make remote queries. This library needs to write the name of the query in *String* format.

It is important to note the importance of prefixes. Indeed, *Wikidata* works a lot with abbreviations of web links used to make a query.

For example, the prefix "wd" is the abbreviation of "http://www.*Wikidata*.org/entity/" which is the http path that leads to the *Wikidata* entities. There are other prefixes that all serve a similar purpose. The final query is the *String* resulting from the assembly of the prefixes and the query name.

The query is created using the *QueryFactory.create(String)* method and then an executable of this query is built thanks to the line of code shown in *figure 15*.



```
QueryExecution qexec = QueryExecutionFactory.sparqlService("https://query.wikidata.org/sparql", query);
```

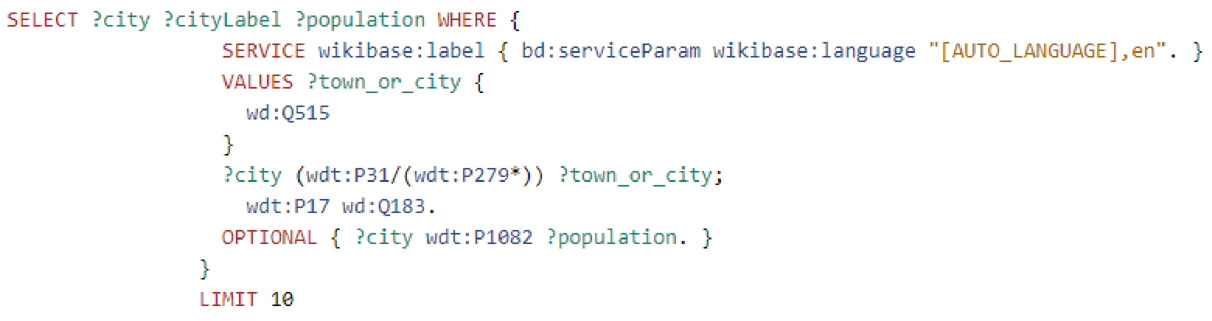
Figure 15 : link the query to Wikidata

The executable of the created query is done with an endpoint corresponding to the *Wikidata* query service and contains the text of the query we wrote earlier.

The *execSelect()* method permits to execute the query and get the results in the form of *ResultSet*.

III.3.2.2 – Title of the request made

As part of the remote SPARLQ request test, this query shown in *figure 16* has been chosen:



```
SELECT ?city ?cityLabel ?population WHERE {
                 SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
                 VALUES ?town_or_city {
                   wd:Q515
                 }
                 ?city (wdt:P31/(wdt:P279*)) ?town_or_city;
                   wdt:P17 wd:Q183.
                 OPTIONAL { ?city wdt:P1082 ?population. }
               }
               LIMIT 10
```

Figure 16 : request used

Elements of the request that are looked for are the instance of the city object, the name of the city and its population. The individual *wd:Q515* corresponds to a city object in *Wikidata.* The field *((wdt:P31/(wdt:P279*))* tells that the query gives instances of the city names. The property *wdt:P17* indicates that these cities are looked within a country object and the individual *wd:Q183* specify that this country is Germany. The property *wdt:P1082* corresponds to the population of the city objects found. This line is placed in the OPTIONAL field because some cities may not have population data attached to them on *Wikidata*. A limit of possible results to 10 is also specified in order not to take too much time when running the query. Note that the prefixes are already understood by *Wikidata*, so it is not necessary to put them back to write the query.

III.3.2.3 – Formatting of the query

Now the goal is to be able to display correctly the results obtained in the view associated. The best solution is to put the data obtained in *ResultSet* in the form of an *ArrayList* that can be displayed in the view.

The formatting start with the creation of an array with three entries, corresponding to each triplet obtained in the results (*city*, *cityLabel* and *population*). Each *QuerySolution* composing the *ResultSet* is browsed to apply the method on its. An array of the size of the number of columns is then created corresponding on the obtained results for each *QuerySolution*. A *RDFNode* is extracted from each element of the *QuerySolution* corresponding to one of the 3 fields contained in a solution. The *RDFNode*, is looked to know if it is a type *Literal* or *Resource*. The method transforms these *RDFNode* into *String* and store them in the 3 entries array according to its index. Once the 3-entry array is complete, it is added to the *ArrayList* and the same operation is repeated for each *QuerySolution*.

After the end of the loop, the *ArrayList* is finished and contains all the elements of the solution. It is now possible to link it with the view. To do this, the line shown in *figure 17* are used.

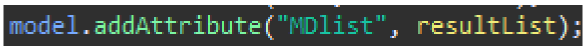

```
model.addAttribute("MDlist", resultList);
```

Figure 17 : add attribute to model

A new data attribute for the model is created. Here it stores the *ArrayList* named *resultList* in the attribute named *MDlist* that will be used in the view.

The view can now use the *ArrayList* under the name *MDlist* (*figure 18*).

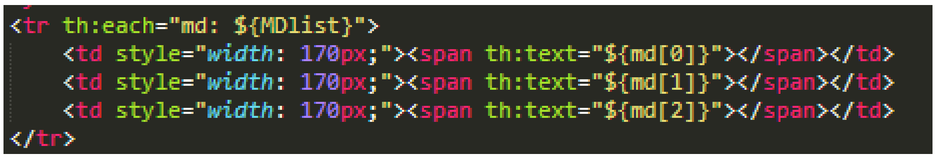
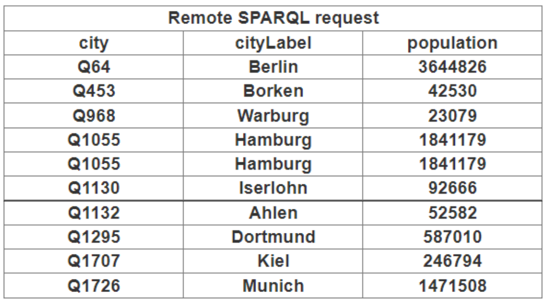

```
<tr th:each="md: ${MDlist}">
    <td style="width: 170px;"><span th:text="${md[0]}"></span></td>
    <td style="width: 170px;"><span th:text="${md[1]}"></span></td>
    <td style="width: 170px;"><span th:text="${md[2]}"></span></td>
</tr>
```

Figure 18 : MDlist call in view

These lines of code can be translated by the fact that for each element *md* contained in the *MDlist*, in this case the triple entry arrays corresponding to each solution, it displays for column 1, the content of the first element of each element *md*, similarly for column 2 and 3.

The result is displayed in a <tbody> which corresponds to the body of an .html table. The view puts in the header of this table, the names of the columns and the title "Remote SPARQL request".

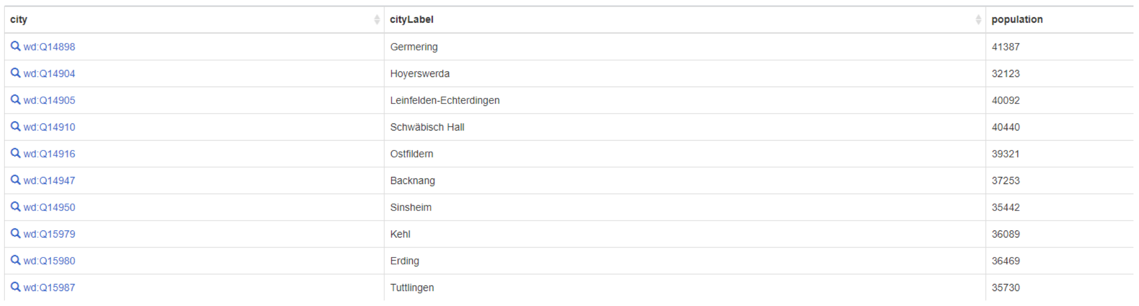
Some style elements are applied using a .css file linked to the project which is linked via the *head.html* file. The result on *testSPARQL.html* is shown in *figure 19*.



Remote SPARQL request		
city	cityLabel	population
Q64	Berlin	3644826
Q453	Borken	42530
Q968	Warburg	23079
Q1055	Hamburg	1841179
Q1055	Hamburg	1841179
Q1130	Iserlohn	92666
Q1132	Ahlen	52582
Q1295	Dortmund	587010
Q1707	Kiel	246794
Q1726	Munich	1471508

Figure 19 : result of the request printed in the view

The results that match the results of the same query directly on the *Wikidata* query service (*figure 20*).



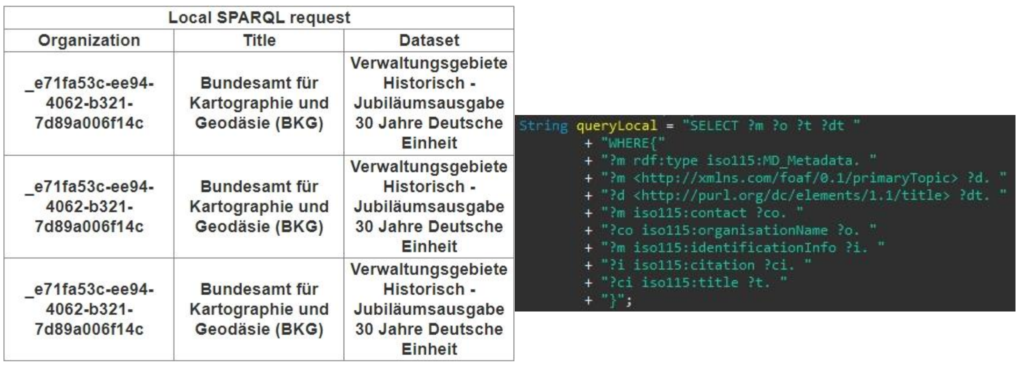
city	cityLabel	population
wd:Q14898	Germering	41387
wd:Q14904	Hoyerswerda	32123
wd:Q14905	Leinfelden-Echterdingen	40092
wd:Q14910	Schwäbisch Hall	40440
wd:Q14916	Ostfildern	39321
wd:Q14947	Backnang	37253
wd:Q14950	Sinsheim	35442
wd:Q15979	Kehl	36089
wd:Q15980	Erding	36469
wd:Q15987	Tuttlingen	35730

Figure 20 : result of the request in Wikidata service

The fact that they are not the same cities is normal because not any particular ranking was added, so they are 10 cities taken completely at random and which change at each execution of the query.

III.3.3 – Local SPARQL request

On the same principle as the remote queries was tested the making of a SPARQL query about local ontology of the application and the displaying of it. *Figure 21* shows the title of the query and the result obtained on the view following the same formatting principles.



Local SPARQL request		
Organization	Title	Dataset
_e71fa53c-ee94-4062-b321-7d89a006f14c	Bundesamt für Kartographie und Geodäsie (BKG)	Verwaltungsgebiete Historisch - Jubiläumsausgabe 30 Jahre Deutsche Einheit
_e71fa53c-ee94-4062-b321-7d89a006f14c	Bundesamt für Kartographie und Geodäsie (BKG)	Verwaltungsgebiete Historisch - Jubiläumsausgabe 30 Jahre Deutsche Einheit
_e71fa53c-ee94-4062-b321-7d89a006f14c	Bundesamt für Kartographie und Geodäsie (BKG)	Verwaltungsgebiete Historisch - Jubiläumsausgabe 30 Jahre Deutsche Einheit

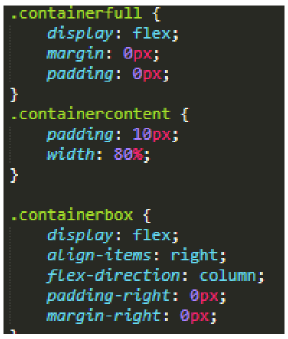
```
String queryLocal = "SELECT ?m ?o ?t ?dt "
        + "WHERE{"
        + "?m rdf:type iso115:MD_Metadata. "
        + "?m <http://xmlns.com/foaf/0.1/primaryTopic> ?d. "
        + "?d <http://purl.org/dc/elements/1.1/title> ?dt. "
        + "?m iso115:contact ?co. "
        + "?co iso115:organisationName ?o. "
        + "?m iso115:identificationInfo ?i. "
        + "?i iso115:citation ?ci. "
        + "?ci iso115:title ?t. "
        + "}";
```

Figure 21 : Local SPARQL request

III.4 – Style tasks

III.4.1 – “Provider” and “Contact” boxes

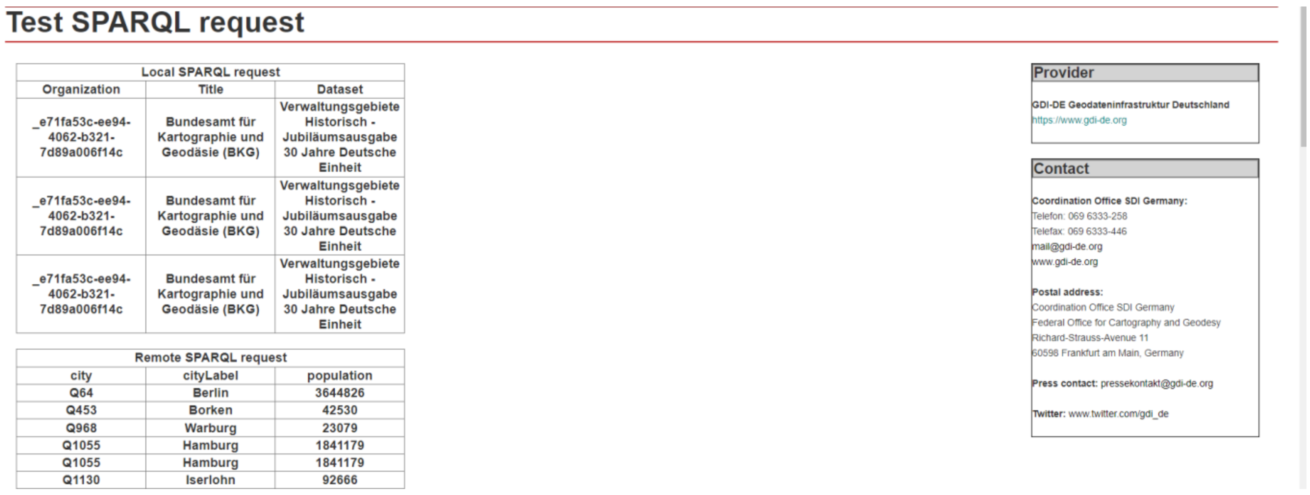
Styling work was also done to get closer to the desired prototype. In particular, the two frames containing the contact information of the company have been made, and the one giving the website of GDI-DE Geodateninfrastruktur Deutschland. *Figure 22* shows the part of the .css file that allows to put the two frames on the right. The class *containerfull* represents the whole screen between the header and the footer. This container contains two sub-containers. The class *containercontent* takes 80% of the total width and is left aligned. This corresponds to the content of the application's features while the remaining 20% on the right is the class *containerbox* and represents the two frames.



```
.containerfull {
    display: flex;
    margin: 0px;
    padding: 0px;
}
.containercontent {
    padding: 10px;
    width: 80%;
}

.containerbox {
    display: flex;
    align-items: right;
    flex-direction: column;
    padding-right: 0px;
    margin-right: 0px;
```

Figure 22 : .css container class

Figure 23 shows the rendering of the two frames compared to those desired by the prototype (*Figure 5*).



Test SPARQL request

Local SPARQL request		
Organization	Title	Dataset
_e71fa53c-ee94-4062-b321-7d89a006f14c	Bundesamt für Kartographie und Geodäsie (BKG)	Verwaltungsgebiete Historisch - Jubiläumsausgabe 30 Jahre Deutsche Einheit
_e71fa53c-ee94-4062-b321-7d89a006f14c	Bundesamt für Kartographie und Geodäsie (BKG)	Verwaltungsgebiete Historisch - Jubiläumsausgabe 30 Jahre Deutsche Einheit
_e71fa53c-ee94-4062-b321-7d89a006f14c	Bundesamt für Kartographie und Geodäsie (BKG)	Verwaltungsgebiete Historisch - Jubiläumsausgabe 30 Jahre Deutsche Einheit

Remote SPARQL request		
city	cityLabel	population
Q64	Berlin	3644826
Q453	Borken	42530
Q968	Warburg	23079
Q1055	Hamburg	1841179
Q1055	Hamburg	1841179
Q1130	Iserlohn	92666

Provider

GDI-DE Geodateninfrastruktur Deutschland
https://www.gdi-de.org

Contact

Coordination Office SDI Germany:
Telefon: 069 6333-258
Telefax: 069 6333-446
mail@gdi-de.org
www.gdi-de.org

Postal address:
Coordination Office SDI Germany
Federal Office for Cartography and Geodesy
Richard-Strauss-Avenue 11
60598 Frankfurt am Main, Germany

Press contact: pressekontakt@gdi-de.org

Twitter: www.twitter.com/gdi_de

Figure 23 : box Provider & Contact

The view uses .css attributes and plays on the alignment of the elements on the right with the *float: right* property and on the fact to choose the size of the font but also its bolding. There is also *href* tags to refer to all the other links to contact the company.

III.4.2 – Footer of the application

The footer of the application has been also realized (*figure 24*), and it is present in all the views by aligning the text elements and by recovering an inline image to display it at the bottom right of the text. The footer was made by doing a .css file playing with the colors and the font size.

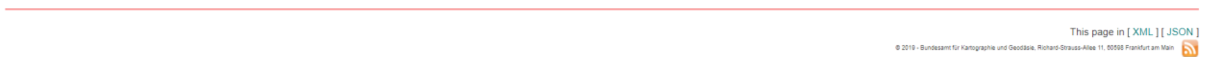

This page in [XML] [JSON]
© 2019 - Bundesamt für Kartographie und Geodäsie, Richard-Strauss-Allee 11, 60598 Frankfurt am Main

Figure 24 : footer of the application

III.4. – Navigation bar

To match the view of the application prototype, we had to integrate a new navigation bar. The challenge was to display the submenus below the menus containing them. This display had to be dynamic.

To do this, menus have been assimilated to <button> type elements in the *nav.html* view. It was then indicated that a <button> representing a menu containing sub-menus will display these sub-menus when the mouse cursor hovers over it. A <button> representing a menu without submenus will lead to the url containing the view representing it when clicking on this <button>. The elements representing the sub-menus are <a> elements that lead to the url of the sub-menu view.

Figures 25 show respectively a button representing a menu containing submenus and a button representing a menu without submenus.

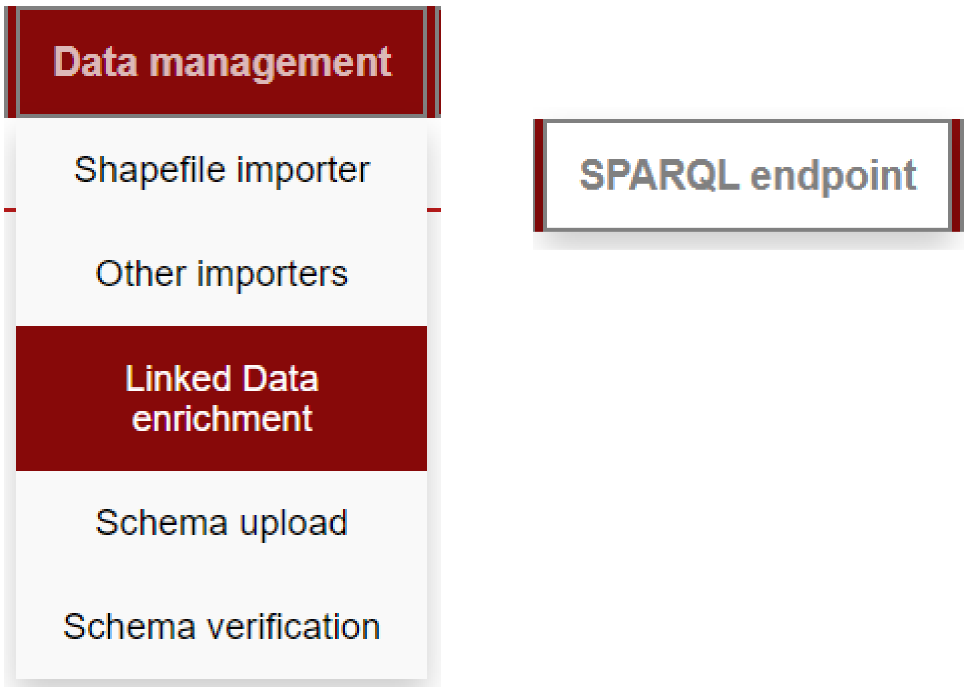


Figure 25 : menu with submenus/menu without submenus

III.4.4 – Header

The last element to realize is the header. It must correspond to the view of the prototype provided upstream. This header is composed of several independent elements.

The first image with the GDI-DE logo is clickable and refers to the url of the view corresponding to the home page. The second element is the text "Linked Data" which is the title of the application and is centered in the header. At the top right, there are the fields "English | Login | Contact" including "Contact" which leads to a view that will be presented later in this report. *Figure 26* shows the view of this header.

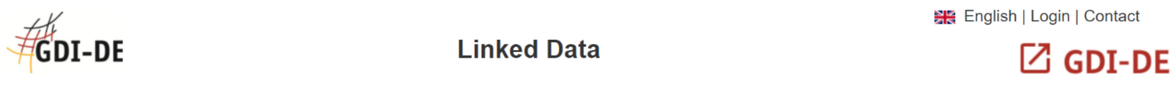


Figure 26 : header of the application

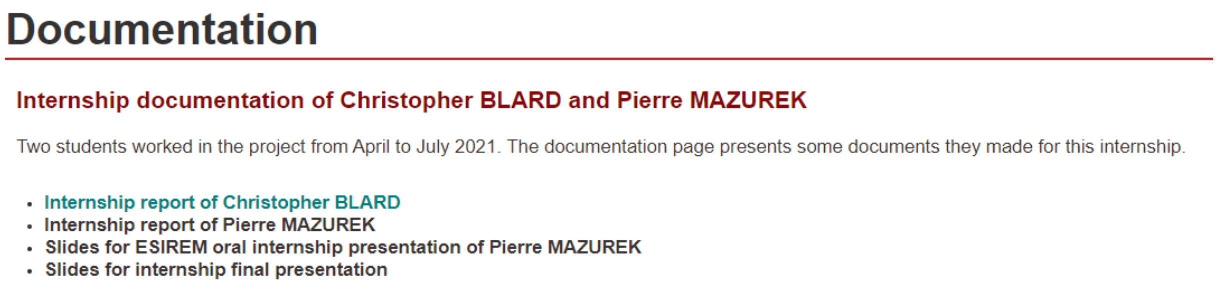
III.4.5 – Permanent fields

All the elements that have been presented correspond to what are called permanent fields. This means that they are coded on all views and present the structure of the application beyond the content of the view. They are fixed and do not vary. They are used to format the application to look as similar as possible to the desired prototype view. This was a big part of the styling work provided.

III.5 – Added menus and functionalities

III.5.1 – The documentation menu

The documentation menu allows you to display all the documentation related to the project. It has been made to list the available documents and to be able to open them in a new window by clicking on them. To do this, it was added in the view corresponding to this menu, <a> elements referring to the local link of these files. A "documents" folder has been created in the static folder and all the documents to be displayed have been added there. These files are accessible via the following url "https://localhost:8080/documents/file_name". This link has been added in the <a> elements with the file names corresponding to the names displayed on the view. *Figure 27* shows the display of this menu.



Documentation

Internship documentation of Christopher BLARD and Pierre MAZUREK

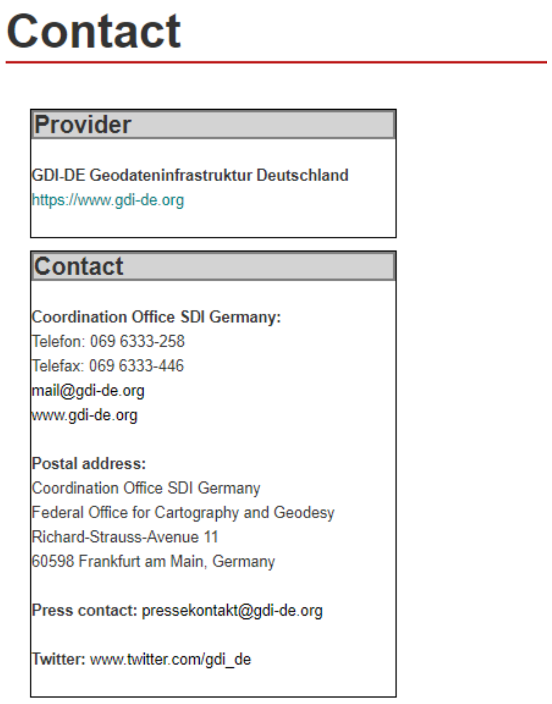
Two students worked in the project from April to July 2021. The documentation page presents some documents they made for this internship.

- Internship report of Christopher BLARD
- Internship report of Pierre MAZUREK
- Slides for ESIREM oral internship presentation of Pierre MAZUREK
- Slides for internship final presentation

Figure 27 : Documentation menu

III.5.2 – The contact tab

The contact tab is called when clicking on the "Contact" field at the top right of the header. This view displays the Provider and Contact boxes directly in the page. This view has been added to the application (shown in *figure 28*).



Contact

Provider

GDI-DE Geodateninfrastruktur Deutschland
https://www.gdi-de.org

Contact

Coordination Office SDI Germany:
Telefon: 069 6333-258
Telefax: 069 6333-446
mail@gdi-de.org
www.gdi-de.org

Postal address:
Coordination Office SDI Germany
Federal Office for Cartography and Geodesy
Richard-Strauss-Avenue 11
60598 Frankfurt am Main, Germany

Press contact: pressekontakt@gdi-de.org

Twitter: www.twitter.com/gdi_de

Figure 28 : Contact tab

III.5.3 – Linked Data enrichment submenu

The Linked Data enrichment menu was realized in direct collaboration with Christopher BLARD. Indeed, this functionality is based on the work he has done to display geographical points on a map and on the work done on the retrieval of SPARQL queries results from a distance. The view that has been realized allows to realize several operations and works thanks to different elements.

III.5.3.1 – "Class to enrich" field

The "class to enrich" field allows you to display an example SPARQL query in an .html <textarea> element. This is a <select> element that allows you to display a list and select an option from that list. Each option is represented by an <option> element and is contained within that list.

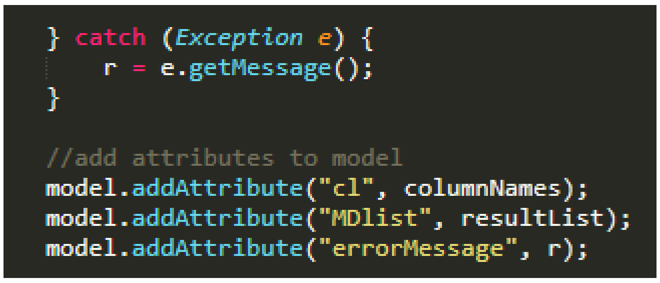
A JavaScript function present in the <script> field of the .html file presenting the Linked Data enrichment menu view allows the selected option to be associated with a text that is the title of the corresponding SPARQL query. This text is then displayed in the <textarea>. It is possible to modify the displayed text in order to customize the chosen query.

III.5.3.2 – "items number" field

Following the same principle as the JavaScript function explained earlier, an element of type <slider> was created. It allows you to slide a slider on a bar. Each location of this slider on the bar represents a value. This value is then retrieved by a JavaScript function that will add it to the <textarea>. This number represents the limit of the desired number of results. If the cursor is placed on the value 150, then the field "LIMIT 150" is added to the query. So, running the query will return 150 results.

III.5.3.3 – "potential error" field

In order to obtain the query results, the same method as explained in part II.3.3 was used. An element has been added to this method in order to get an error message if the query is wrong. The lines of code shown in *figure 29* have been added. The error message is retrieved and added to the model so that it can be displayed in the associated view.



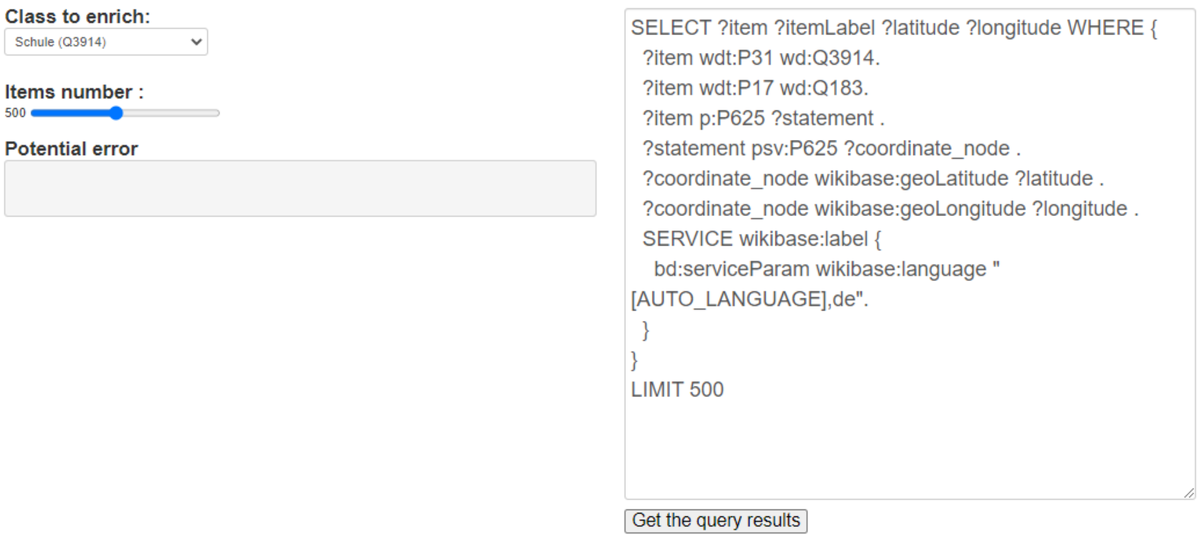
```
} catch (Exception e) {
    r = e.getMessage();
}

//add attributes to model
model.addAttribute("cl", columnNames);
model.addAttribute("MDlist", resultList);
model.addAttribute("errorMessage", r);
```

Figure 29 : add of the error message in the method

The potential error is displayed in the view in an element of type <pre> that provides a frame to display text in.

II.5.3.4 – Execution of the query and displaying

The method to retrieve the query results from the heading contained in the <textarea> is called when clicking the "get the query results" button. The entire parameter interface is shown in *figure 30* where "school" is chosen and 500 for the item number limit.


Class to enrich:
Schule (Q3914)

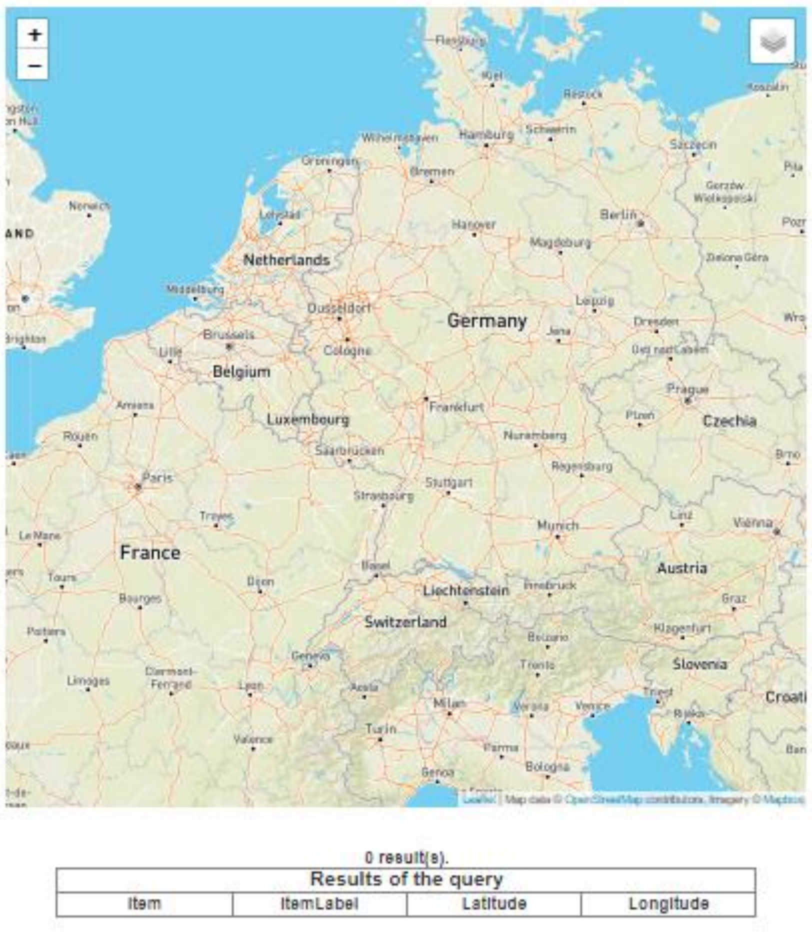
Items number :
500

Potential error

```
SELECT ?item ?itemLabel ?latitude ?longitude WHERE {
  ?item wdt:P31 wd:Q3914.
  ?item wdt:P17 wd:Q183.
  ?item p:P625 ?statement .
  ?statement psv:P625 ?coordinate_node .
  ?coordinate_node wikibase:geoLatitude ?latitude .
  ?coordinate_node wikibase:geoLongitude ?longitude .
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "
[AUTO_LANGUAGE],de".
  }
}
LIMIT 500
```

Figure 30 : interface for setting the parameters of the request

Christopher's job was to retrieve the query results and display them on a map. The display of the results in a table was also put back below the map as shown in *figure 31*.



0 result(s).

Results of the query			
Item	ItemLabel	Latitude	Longitude

Figure 31 : display of the query in a map and in a table

Conclusion

At the end of my internship, we managed to make remote queries from *Wikidata* and display them properly. We created a kind of SPARQL query tutorial to learn the language through the different examples of queries available and the possible error messages returned by the method. The application interface corresponds to the desired prototype view, with all the permanent elements added to the project. The next step is to link directly a local SPARL query with a remote SPARQL query in order to be able to link all these data together. Thus, the ontology of the application would be enriched by all the remote queries, either on *Wikidata* or on other sites.

This internship gave me a very good vision of what programming in a company can be. I was able to learn very important new notions, such as the structuring of a project by Spring Boot, the semantic web in general or the use of new query languages like SPARQL. I also learned about the model-view-controller architecture and its use, which is simple and makes it easier to understand the code. With the use of Java, .html and .css programming languages, I was able to improve my skills in these areas.

I was also able to see what telecommuting could mean in a company. It forced me to be well organized, diligent and motivated, but above all not to hesitate to work in a team and to communicate with my colleagues.

I3Mainz is a German laboratory, so I was able to practice my English during the different meetings I had with them but also with the customer. I learned what a PERT diagram was, which I made myself and which allowed us to better organize ourselves in terms of deadlines for our tasks.

This internship was a real learning experience for me and gives me a glimpse of the possible professional opportunities after finishing my studies at ESIREM.

Bibliographie

[1] Cousin, C. (2008). *Tout sur le web 2.0.*

[2] Stephen Walther, *ASP.NET MVC Framework Unleashed*, Sams Publishing – 2009

[3] SEAN GRAHAM, DICK'S SPORTING GOODS, https://spring.io/why-spring

[4] Bizer, Heath, Berners-Lee, « Linked Data - The Story So Far », 2009

[5] Elena Simperl, Reusing ontologies on the Semantic Web: A feasibility study, October 2009

[6] https://www.*Wikidata*.org/wiki/*Wikidata*:SPARQL_query_service/queries/examples