GIS & Spatial Web

Coursework Report

Author: Martiño Rivera Dourado (18111777)

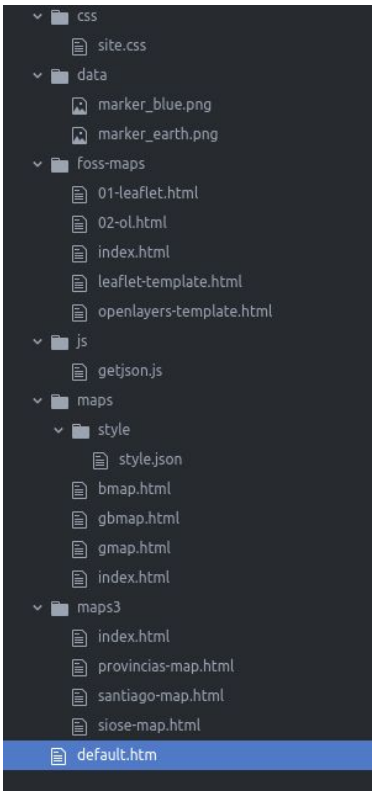
Index

1. Introduction

Geographic Information Systems are the core of the web mapping technologies. Nowadays, the web has enabled an user interface easily customizable and adaptable to almost any OS. That made the web technologies to join the GIS in order to create user interfaces with mapping tools, to be able to show spatial data in an interactive way.

This coursework consists in three parts or deliverables. In the first deliverable, Google and Bing mapping technologies are used through the use of their APIs. In the second deliverable, FOSS JS technologies as Leaflet and OpenLayers are used. The last deliverable aims to use GeoServer with self-sourced data stored in a PostGIS database and displayed using OpenLayers JS Framework.

I want to highlight that all the code included in this coursework has been **self-developed**. Although it is based in the study of code of some examples, it has been developed based in the documentation of the APIs and JS frameworks used. No copy-paste was used, aiming a better understanding and greater learning of web and GIS development. Besides, the main idea was to make the code clean, easy to read and optimized.



Finally, this document has additional information specified as “Technical details”, having the objective to clarify the features added to the maps. In the deliverables II and III, a brief description is included. All those extra contents should be treated as additional for proving the work done in this coursework.

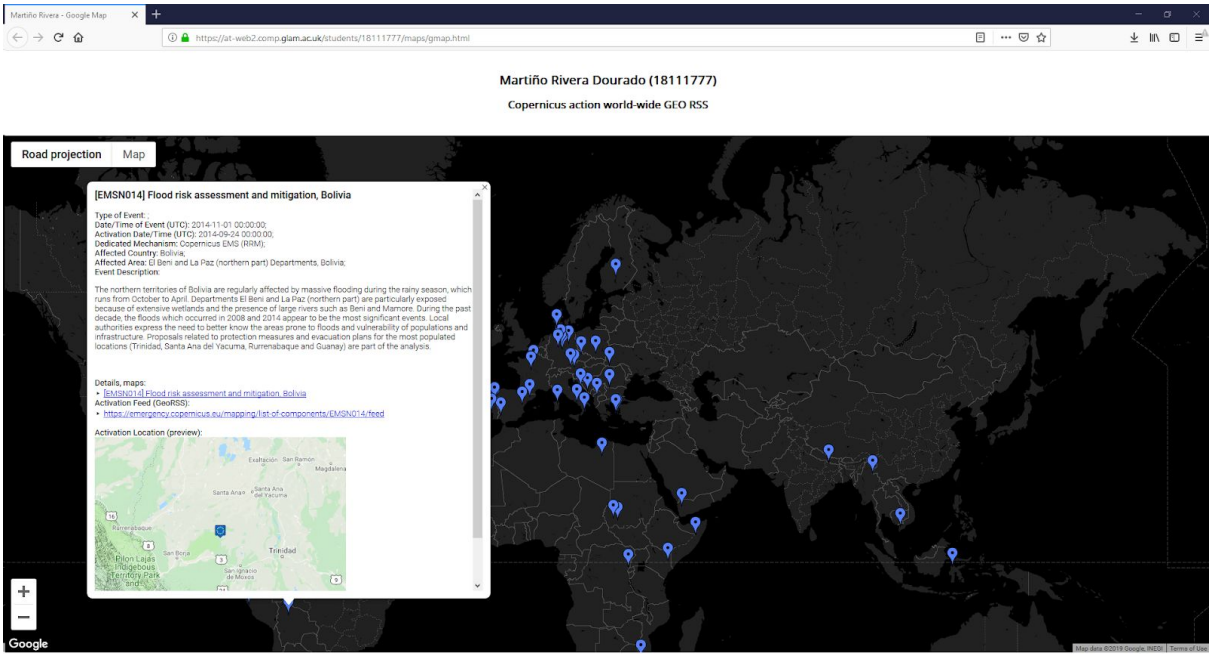
All the web pages developed can be found up and running in the following link: https://at-web2.comp.glam.ac.uk/students/18111777/ .

The file structure can be seen in the image at the left. The code has been attached in a ZIP file.

2. Deliverable I

The first part of this coursework involves the use of Google and Bing privative APIs. For this purpose, I tried to add useful data and functionality to the maps. Those are available here: https://at-web2.comp.glam.ac.uk/students/18111777/maps/index.html

2.1 Copernicus action world-wide GEO RSS



2.1.1 Brief Description

"*Copernicus is the European Union's Earth Observation Programme, looking at our planet and its environment for the ultimate benefit of all European citizens.*"[1]

In 2015, *Copernicus* has started a new project: the **Copernicus Emergency Management Service (EMS)**, that *"uses satellite imagery and other geospatial data to provide free of charge mapping service in cases of natural disasters, human-made emergency situations and humanitarian crises throughout the world."*[2]

This map requests one of the GeoRSS feed that EMS has, making it interactive for the user by pop-ups. It also provides a map style that highlights the main roads available to access those points of Emergency.

[1] https://www.copernicus.eu/en/about-copernicus
[2] https://emergency.copernicus.eu/mapping/ems/service-overview

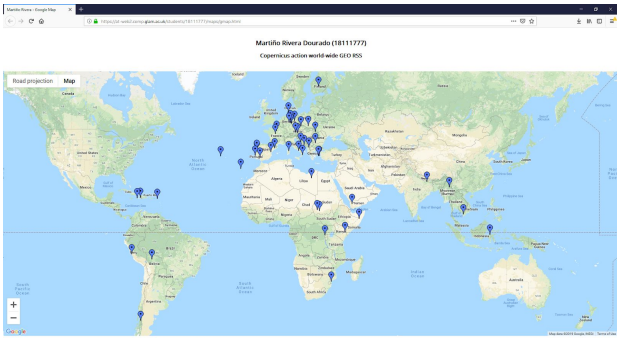
2.1.2 Technical details

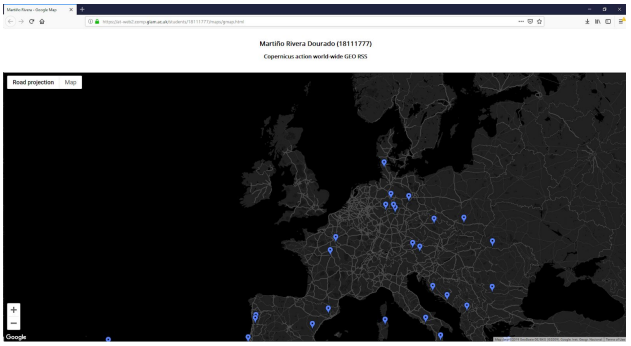
For the purpose of this map, it is needed to load a JSON file located in `maps/style` folder, where I have placed the style created with a Google tool[3]. For this, I have implemented an AJAX function in a separate file `js/getjson.js`. The code is included in the Appendix A. This file is going to be used in some of the next maps, making it reusable.

On the other hand, I have changed the default behaviour of the scroll function to prevent the movement[4] of the web page itself. Besides, I changed the location of the zooming buttons[5], and I added an extra layer.

Finally, I have created an effect of delay, that changes the style and zoom of the map:

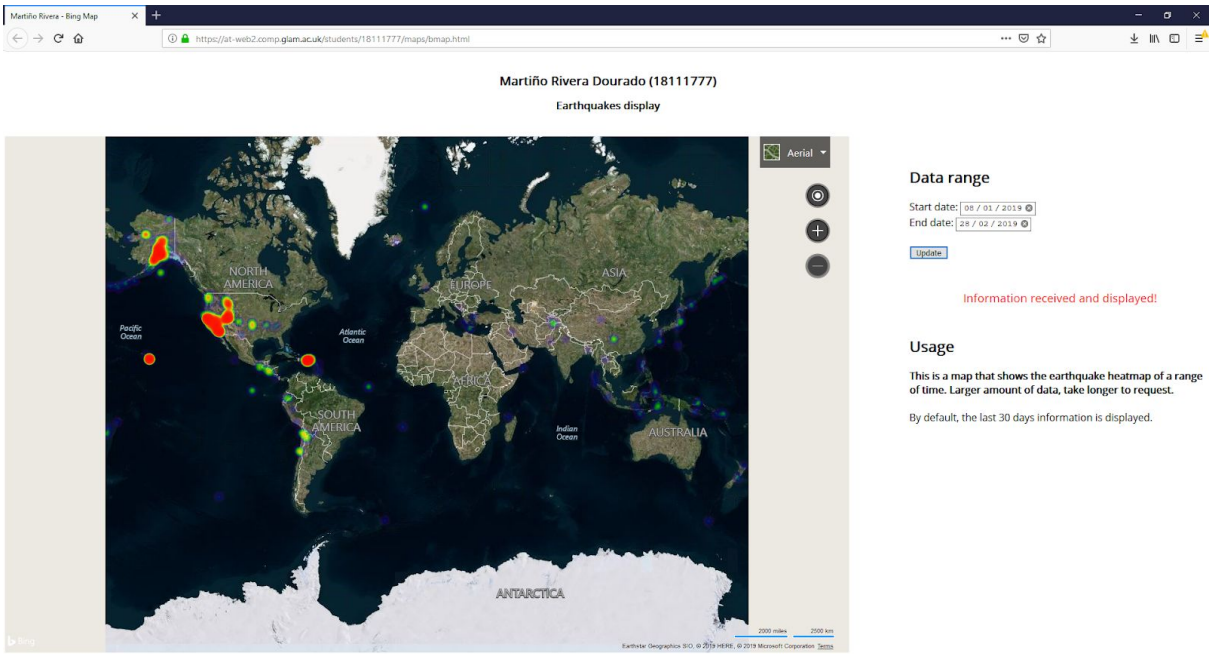
```
// Adjust center and zoom after loading the data
       setTimeout(function(){
              map.setCenter({lat:51.503376, lng:-0.119538});
              map.setZoom(5);
              map.setMapTypeId('road_project');
       }, 2000);
```





[3] https://developers.google.com/maps/documentation/javascript/styling
[4] https://developers.google.com/maps/documentation/javascript/interaction
[5] https://developers.google.com/maps/documentation/javascript/controls

2.2 Earthquakes display



2.2.1 Brief description

USGS or *United States Geographical Survey* [6] is "the sole science agency for the Department of the Interior" of the United States. Among other fields, this agency analyzes the earthquake data on the whole Earth.

The goal of this map is to make use of the implementation of the *FDSN Event Web Service Specification*[7] this agency has, deployed as an API[8] to search and request the earthquake data using several parameters. Once we have the data, we will create a heatmap that will show us the Earth zones within one period of time where the majority of earthquakes occured.

2.2.2 Technical details

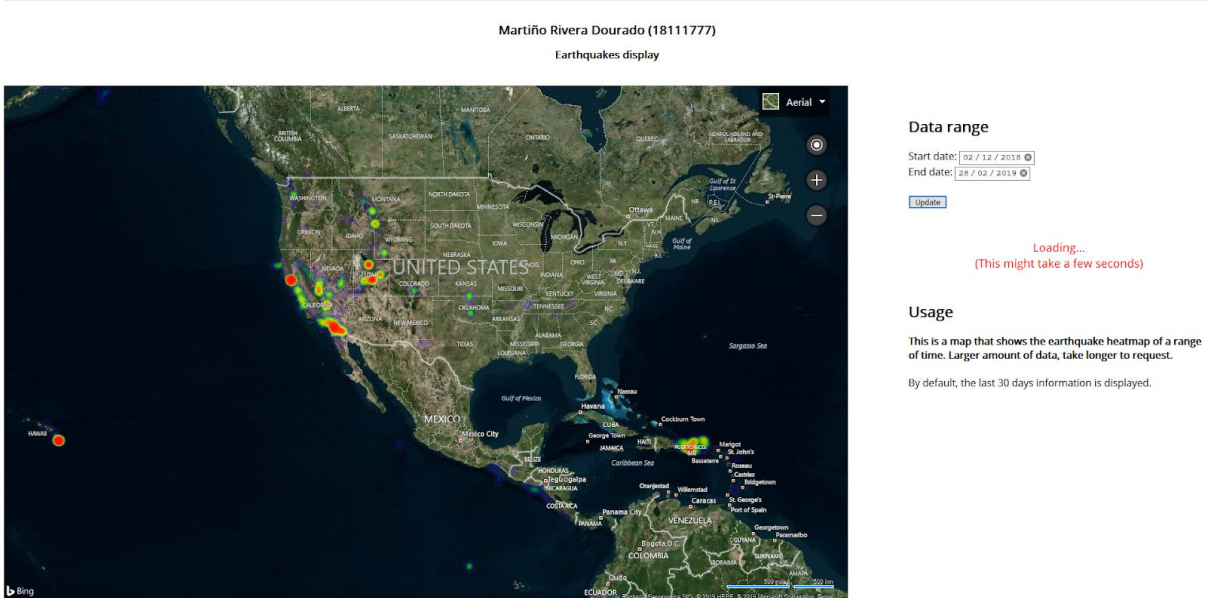
For this purpose, I have created a HTML5 form to introduce two dates: starting and ending date of the time period we want the search. Then, using those parameters, I make use of the http API to request the GeoJSON data. This request is done asynchronously using the AJAX function located in `js/getjson.js`. [9] This way, we can navigate through the web without any problem.

[6] https://www.usgs.gov/about/about-us/who-we-are
[7] http://www.fdsn.org/webservices/FDSN-WS-Specifications-1.0.pdf
[8] https://earthquake.usgs.gov/fdsnws/event/1/
[9] Appendix A

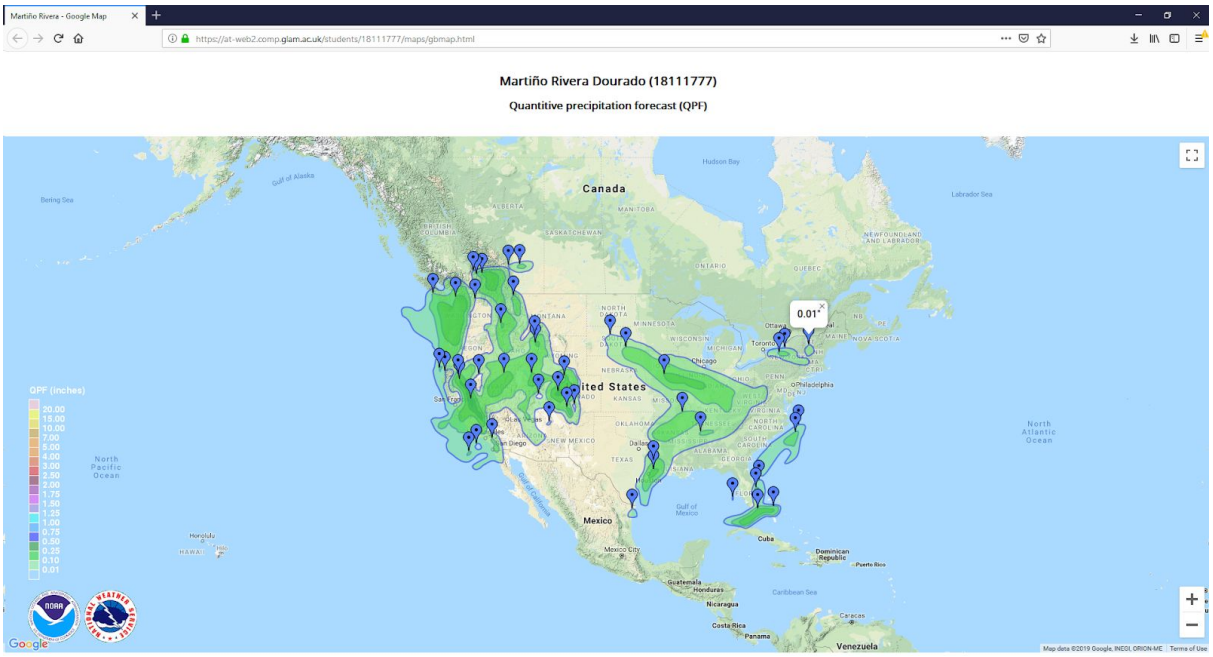
In addition, I have managed to make possible to make several searches by deleting the previous heatmap and loading a new one. For the website to be more interactive, some html content has been made adaptable to those changes. This feature is implemented as additional functions defined:

```
function display_loading(){
            var element = document.getElementById("loading");
            element.innerHTML = "Loading... <br> (This might take a few
seconds)";
}

function display_complete(){
            var element = document.getElementById("loading");
            element.innerHTML = "Information received and displayed!";
 }
```

More comments can be found in the original code.



2.3 Quantitative precipitation forecast (QPF)



2.3.1 Brief description

The Weather Prediction Centre[10] mission's stated in their webpage is being *"A leader in the collaborative weather forecast process by delivering responsive, accurate, and reliable national forecasts and analyses. "*

One of the functions it offers is the "Quantitative Precipitation Forecast" : *"The QPF desk prepares and issues forecasts of accumulating (quantitative) precipitation, heavy rain, heavy snow, and highlights areas with the potential for flash flooding."*

This map loads the latest release of the KML file containing this forecast and loads it on top of a Google Map.

2.3.2 Technical details

For the purpose of this map, a TERRAIN style has been loaded. Besides, map elements have been explicitly specified, as `mapTypeControl` or `gestureHandling`.

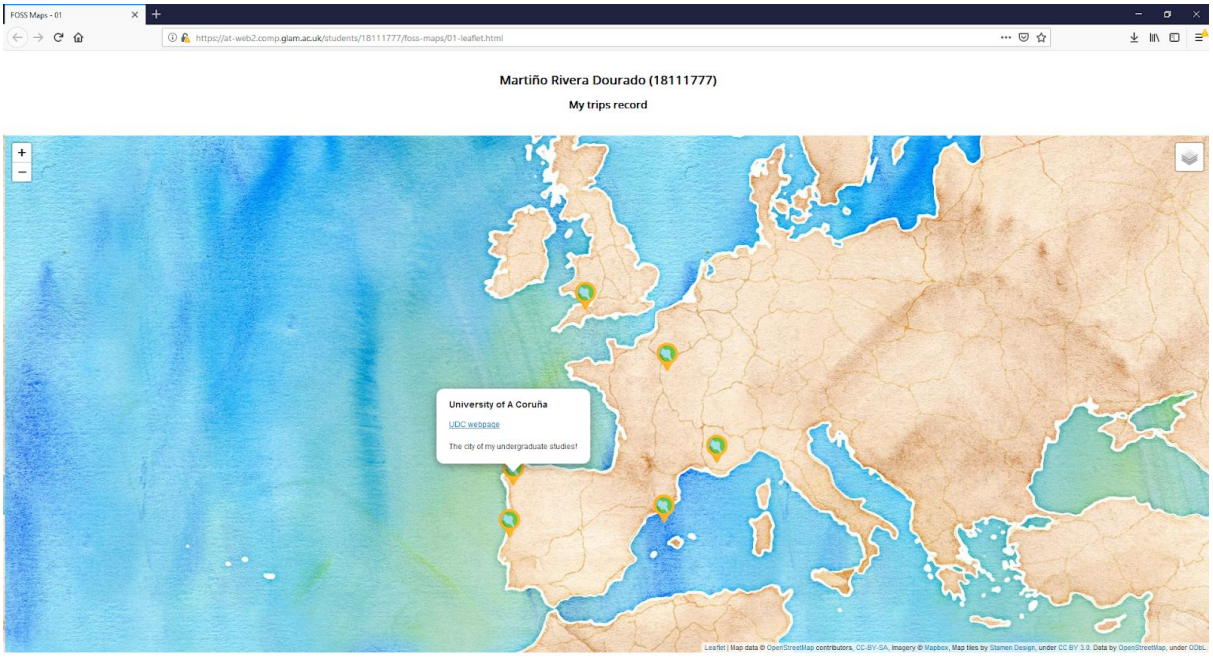
[10] https://www.wpc.ncep.noaa.gov/

```
// Set map options
              var mapOptions = {
                     center: new google.maps.LatLng(47.595712, -122.327695),
                     zoom: 8,
                     mapTypeId: google.maps.MapTypeId.TERRAIN,

                     // Map elements typing
                     disableDefaultUI: true,
                     zoomControl: true,
                      fullscreenControl: true,
                     mapTypeControl: false,
                     zoomControlOptions: {
                            position: google.maps.ControlPosition.RIGHT_BOTTOM
                     },
                     gestureHandling: 'greedy', // disables the ctrl+scroll
zooming
              };
```

3. Deliverable II

The second part of the coursework is based in the use of FOSS technologies. Namely, Leaflet[11] and Openlayers[12]. For this, I have used both technologies, each in one map. However, I emphasized more in OpenLayers, as shown later. The website can be accessed in this link: https://at-web2.comp.glam.ac.uk/students/18111777/foss-maps/index.html

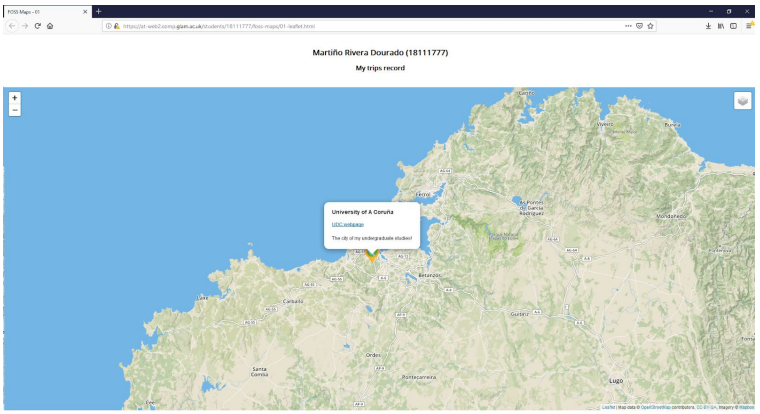
3.1 Leaflet: My trips record



3.1.1 Brief description

The idea behind this map is to show some of my international trips. It shows markers of the places I have been with some description in the form of a popup. It also supports two tile layers, giving the user the opportunity to change to a street map to view more details about the places where the markers are located.

[11] https://leafletjs.com/
[12] https://openlayers.org/



3.1.2 Technical details

For this map, two tile servers have been used: Mapbox[13] Streets and Stamen[14] Watercolor. Besides, a control to enable changing between both maps and personalized icons were added.

Additionally, I have designed a JS class to wrap in an object the marker and the popup. That way, it is easier to create a marker with a popup:

```
// Personalized class of a marker with a description
class PopMarker {
          constructor({location, description}){
             this.description = description;
             this.popup = L.popup({closeButton: false});
             this.marker = L.marker(location, {icon: earth_icon});
          };

       add(map) {
          this.popup.setContent(this.description);
          this.marker.bindPopup(this.popup);
          this.marker.addTo(map);
       }
};
```

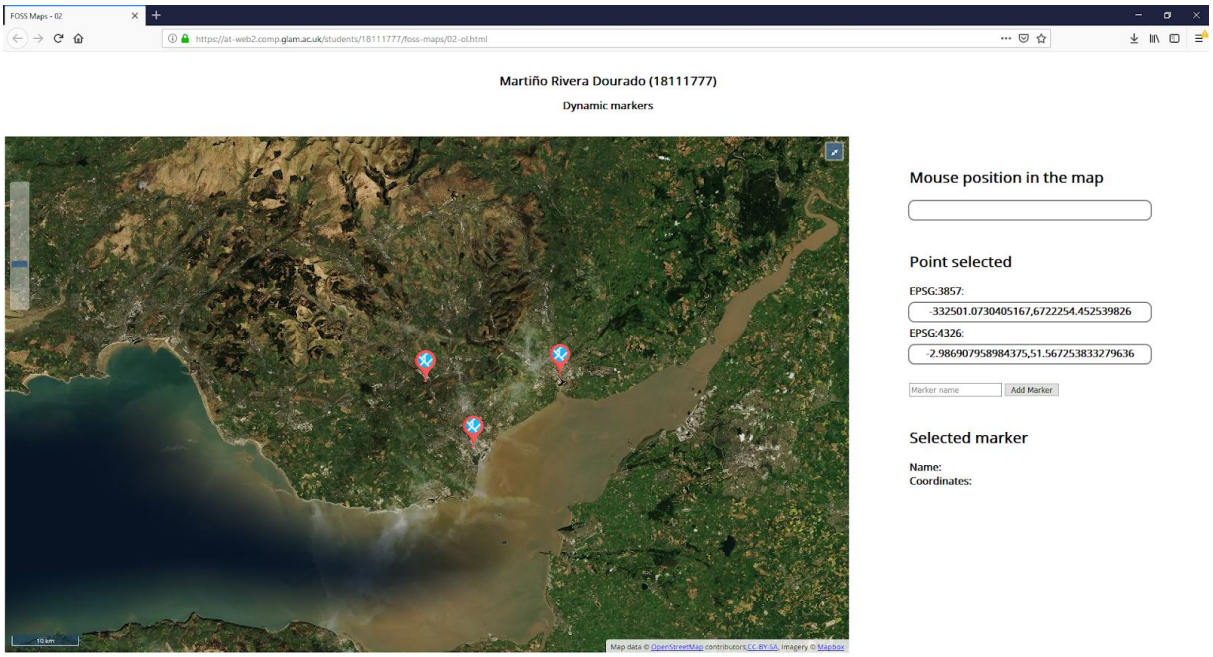
```
new PopMarker({
        location: [51.590030, -3.329507],
        description:
          '<h3> University of South Wales </h3>' +
          '<a href="https://www.southwales.ac.uk/" target = "_blank">USW
webpage</a>' +
          '<p>I have studied my third year of Computer Science here as an Erasmus
student<p>'
      }),
```

[13] https://www.mapbox.com/
[14] http://stamen.com

This way, I have initialized several `PopMarker` objects in a list, and them add them with the built-in function of the objects by simply looping the list:

```
markers.forEach(
        function(item){
            item.add(mymap);
        }
);
```

3.2 OpenLayers: Dynamic markers



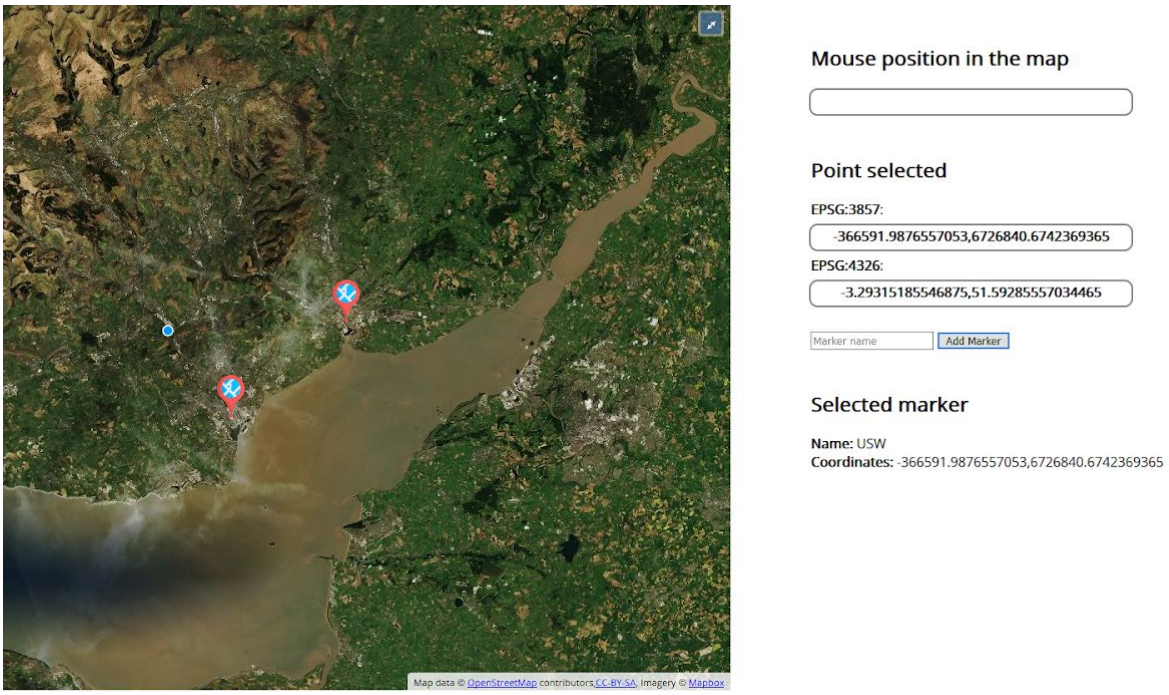
3.2.1 Brief description

The idea of this map is to add markers dynamically. That is, given an user input as a click on a pixel of the map, set a marker with a name or id assigned. Additionally, it shows the coordinates of the mouse pointer on the map. Once it is clicked, coordinates are selected. Also, the *EPSG: 4326* projection of the coordinates is shown.

In order to add the marker, once a point is selected, a name has to be specified. Otherwise, when clicking '*Add Marker*', an alert is shown.

Finally, markers added can be selected, as shown in the next screenshot. Selecting a marker will display the name and the coordinates assigned. The layout used is explained later in this document[15].

[15] 6. Web page layout



3.2.2 Technical details

First, I specified the map controls. Namely, the *scale line*, the *full screen* button and the *zoom slider.* Additionally, layout has been adapted to be able to display the mouse position off the map.

```
controls: [
       new ol.control.Attribution({
              collapsible: false
       }),
       new ol.control.FullScreen({
              keys: true
       }),
       new ol.control.MousePosition({
              className: 'mouse-position',
              target: 'mouse-position'
       }),
       new ol.control.ScaleLine({
              minWidth: 100
       }),
       new ol.control.ZoomSlider(),
 ],
```

```
<div class="desk">
       <div class="leftContainer" id="map"></div>
          <div class="rightContainer">
             <div class="vertical-box">
                <h2>Mouse position in the map</h2>
                <div id="mouse-position" class="coordinate-display"></div>
             </div>
...
```

In order to create the markers dynamically, another layer has been created, with a personalized icon:

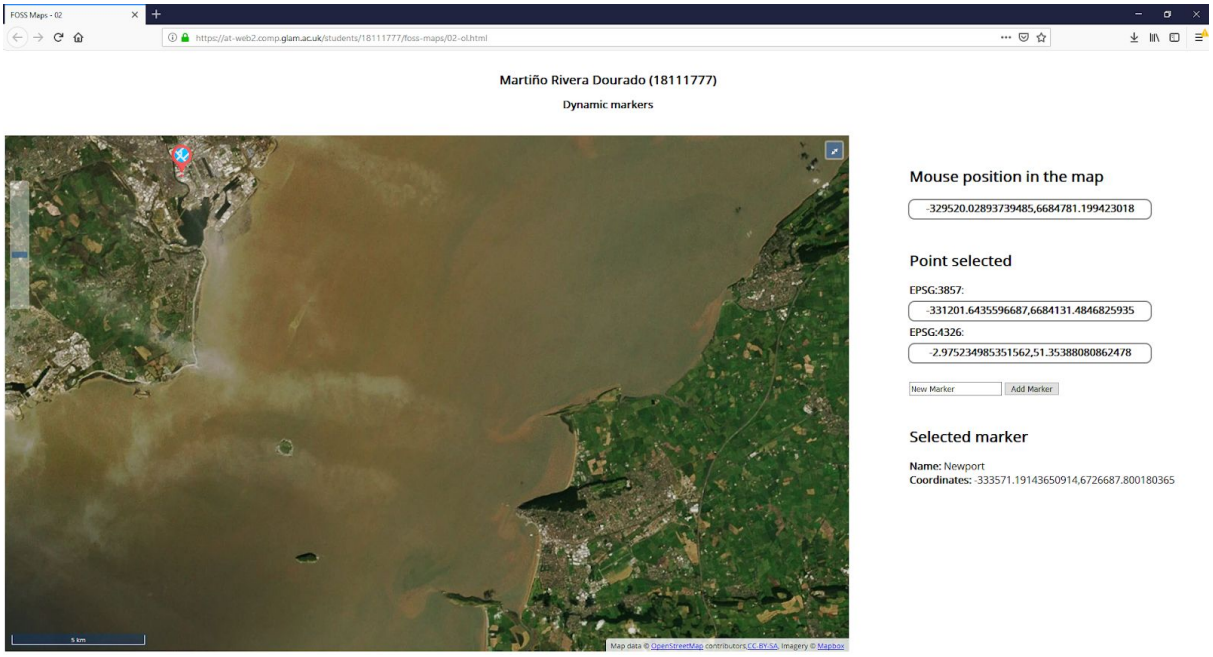
```
// Markers layer
       var markers_source = new ol.source.Vector();
       var anon_counter = 0; // Counter of not named markers added

       map.addLayer(new ol.layer.Vector({
          source: markers_source,
          style: new ol.style.Style({
             // Personalized icon
             // Icon made by https://www.freepik.com/ from www.flaticon.com
             image: new ol.style.Icon({
                 scale: 0.1,
                 anchor: [0.5, 1],
                 anchorXUnits: 'fraction',
                 anchorYUnits: 'fraction',
                 src: '../data/marker_blue.png'
             })
          }),
       }));
```

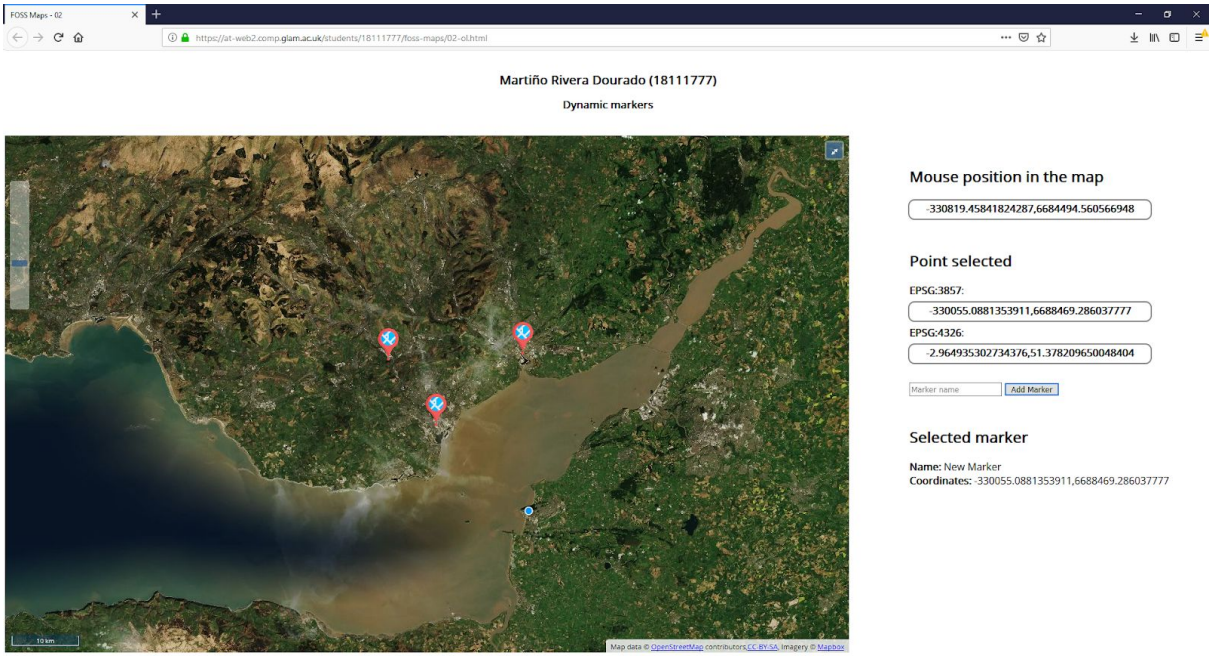
As it can be noticed, the icon parameter had to be modified in order to place the tail of the image actually on the point selected. Otherwise, the center of the image would be placed.

Adding the marker, adds a feature to this layer created:

```
function addMarker() {
   var id = document.getElementById('marker_id').value;
   var coordinates
=document.getElementById('selected_point').innerHTML.split(',');
   var marker_feature = new ol.Feature(new ol.geom.Point(coordinates));
   document.getElementById('marker_id').value = "";    // Reset input
   if (id.length == 0){
      alert('Insert a name for the marker');
      return;
   }
   marker_feature.setId(id);
   markers_source.addFeature(marker_feature);
};
```

The process of creating a marker can be seen in the next screenshots:





Finally, for selecting a feature (i.e. a marker), an event handler, an event listener and an OpenLayer *Iteraction*[16] have to be enabled:

```
// Dynamic point selection
  // Event handler
  function changetoclicked (evt) {
       document.getElementById('selected_point').innerHTML = evt.coordinate;
       document.getElementById('selected_point_4326').innerHTML =
             ol.proj.transform(evt.coordinate, 'EPSG:3857', 'EPSG:4326');
  };

  // Event listener (interaction)
  map.on('click', changetoclicked);

  // Feature selection and information display
  var mouse = new ol.interaction.Select({
     condition: ol.events.singleclick
  });

  map.addInteraction(mouse);
  mouse.on('select', function(evt){
     if (evt.selected.length == 0) return;
     document.getElementById('selected-name').innerHTML =
evt.selected[0].getId();
     document.getElementById('selected-coordinates').innerHTML=
evt.mapBrowserEvent.coordinate;
  });
```

[16] https://openlayers.org/en/latest/apidoc/module-ol_interaction.html

4. Deliverable III

In the last part of this coursework a GeoServer and a PostGIS database have been deployed. As it has been privately deployed, that is, in a private address, it is needed to access those maps from inside the university network domain.

In this case, I have used OpenLayers JS framework to display the maps, requesting the WMS of the GeoServer. All the maps are available in the following index page: https://at-web2.comp.glam.ac.uk/students/18111777/maps3/index.html

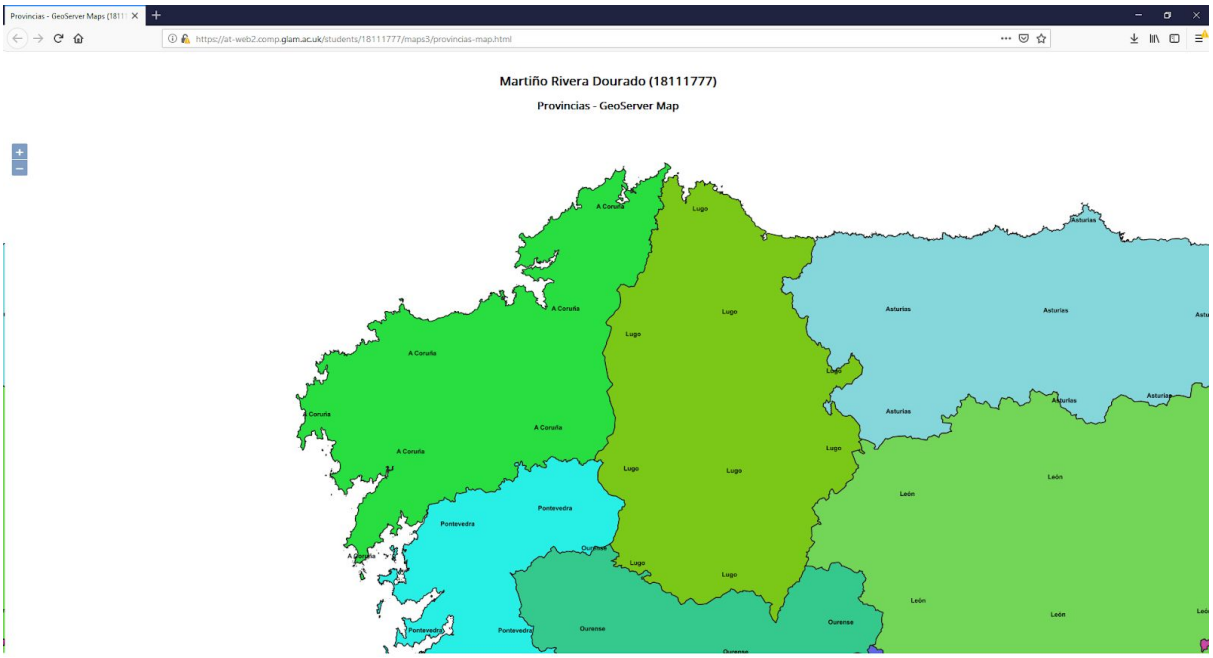
4.1 Provincias map

4.1.1 Brief description

A *"provincia"* is an administrative distribution of Spain. One Autonomic Community can be divided in more than one *"provincia"*. This map aims to show the different provinces in Spain, displaying their name.

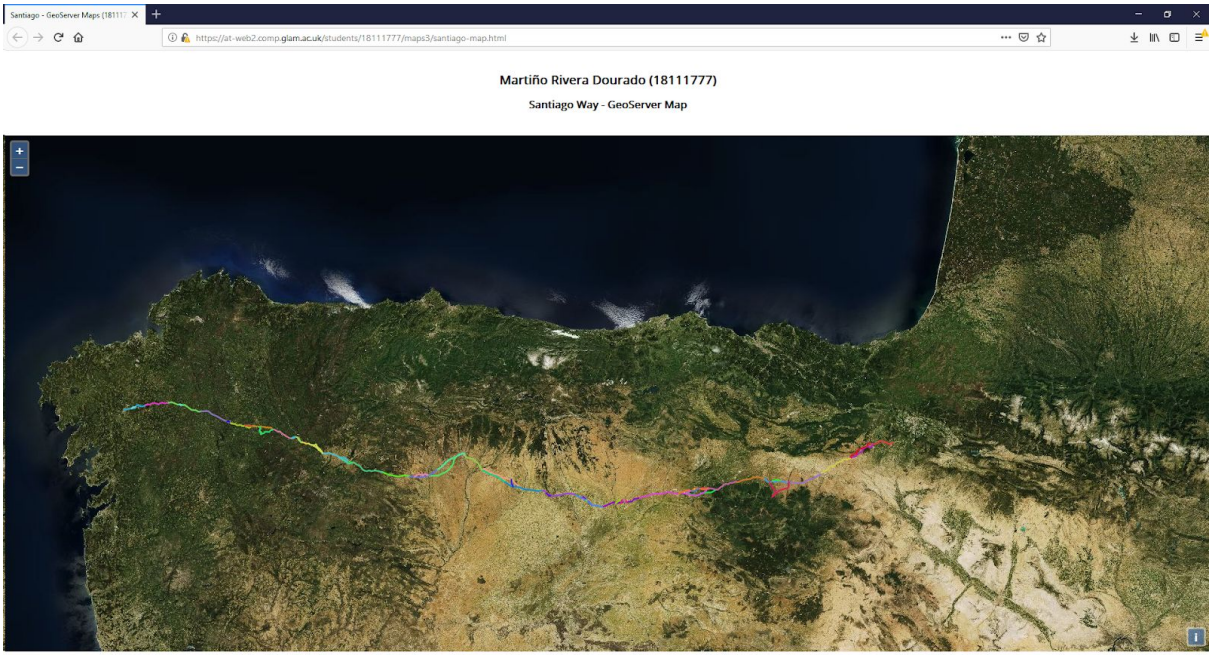
4.1.2 Data preparation and styling

This data has been obtained from the *National Geoinformation Institute of Spain*[17]. The copyright information details are included both in the code and in the GeoServer layer. The data preparation has been done in *QGIS*. Besides, I have created a SLD file from *QGIS*, adding the labels with the correct codification (UTF-8) and different colours for each *"provincia"*. This information has been stored in the *PostGIS* database, where the GeoServer layer was getting the information from. A style has also been created from the SLD file.



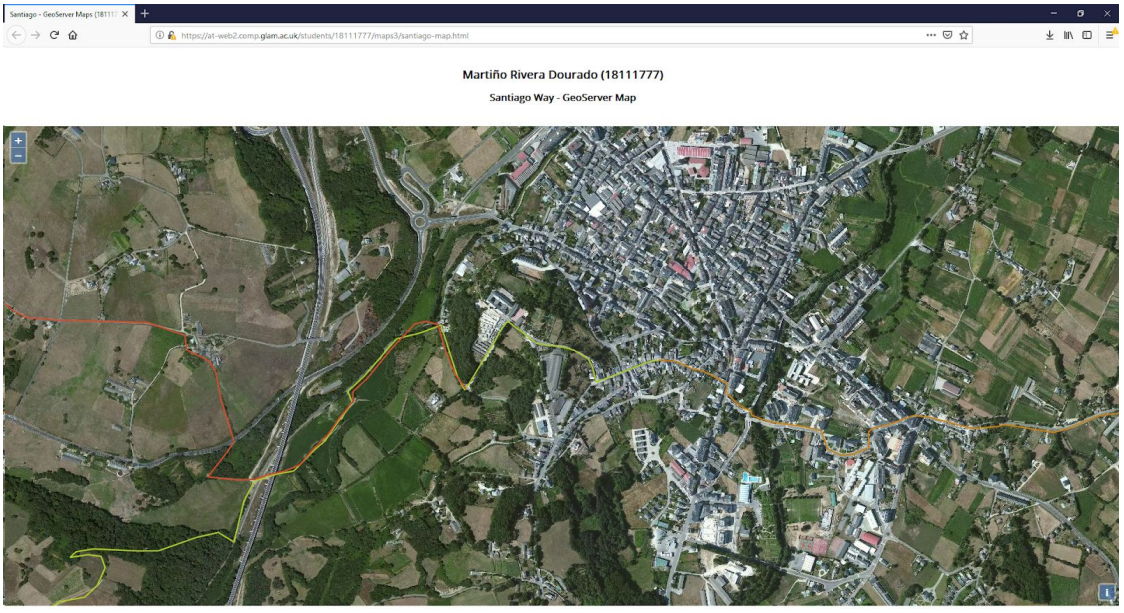
[17] http://www.ign.es/web/ign/portal

4.2 Santiago French Way map



4.2.1 Brief description

Camino de Santiago[18] is world-wide famous. It is both touristic and religious destination for pilgrims. This map shows all the stages of the *French Way*[19]. Each stage is shown in a different colour, on a satellite tile layer from *Mapbox.*



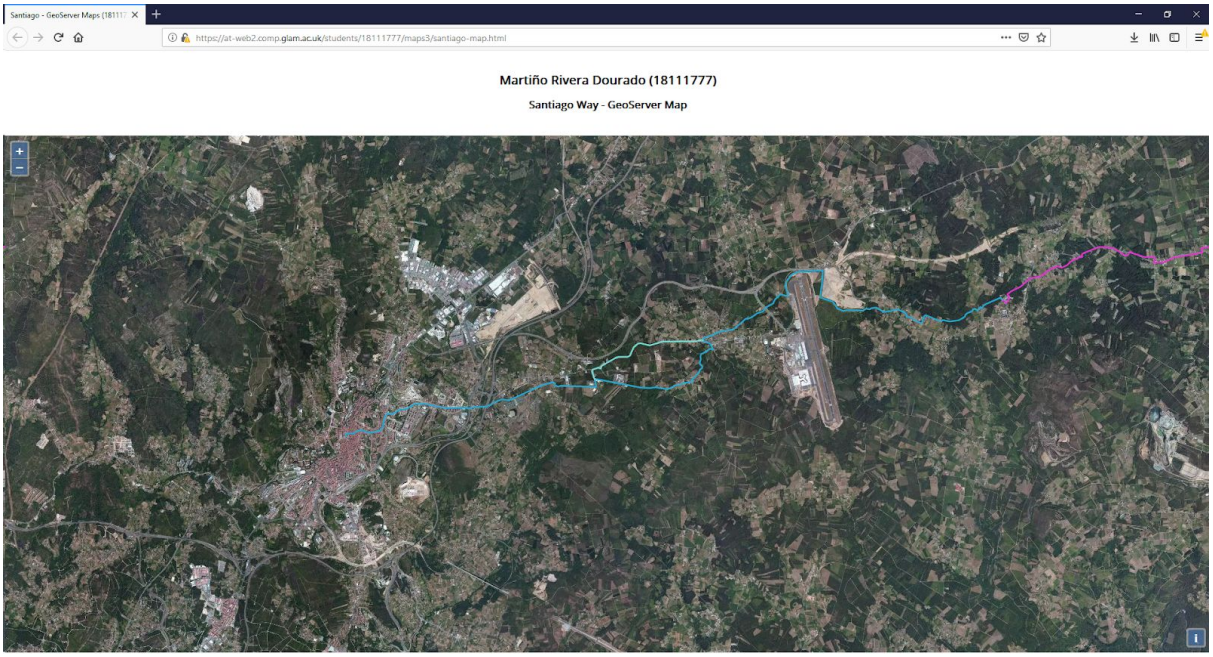
[18] https://en.wikipedia.org/wiki/Camino_de_Santiago
[19] https://en.wikipedia.org/wiki/French_Way

4.2.2 Data preparation and styling

As in the previous map, *QGIS* was used to upload the data and create an SLD file to create a style in *GeoServer*. However, the data source[20] were several KML files, that had to be merged to upload to a single table.

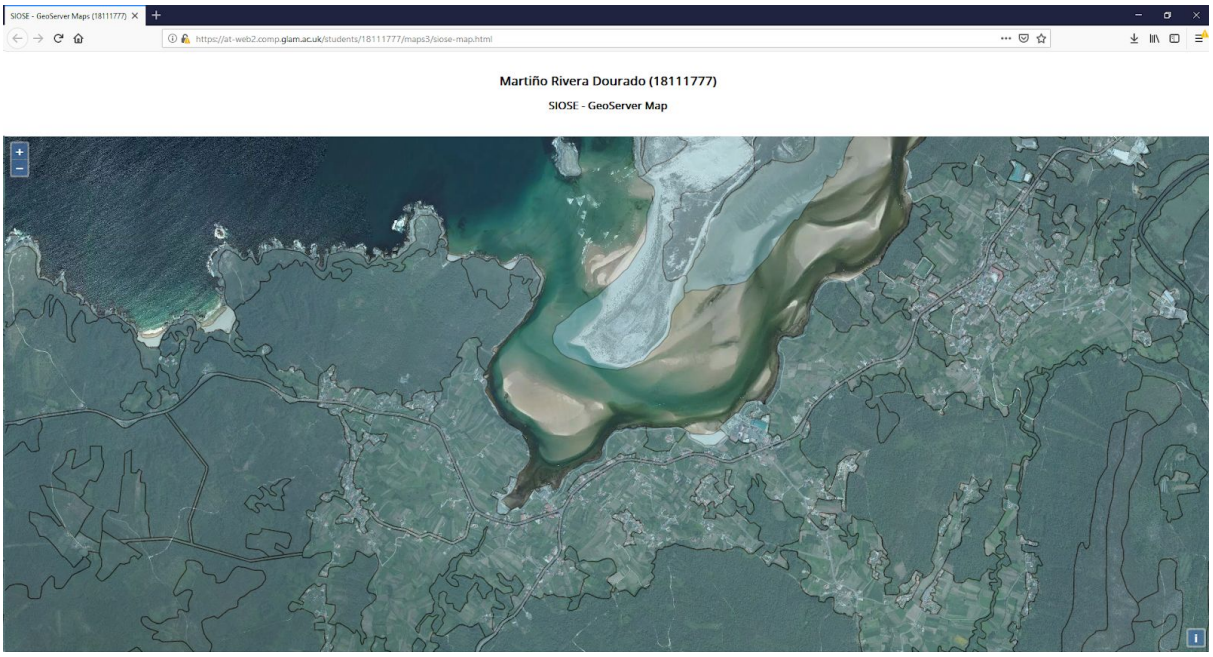
For this, all the layers were loaded to *QGIS*, merged and exported as a Shapefile. Then, this file was loaded and uploaded.

Again, copyright is shown in the map and it is attached to the layer in the *GeoServer*.



[20] https://www.caminosantiago.org/cpperegrino/comun/inicio.asp & http://www.ign.es/web/ign/portal

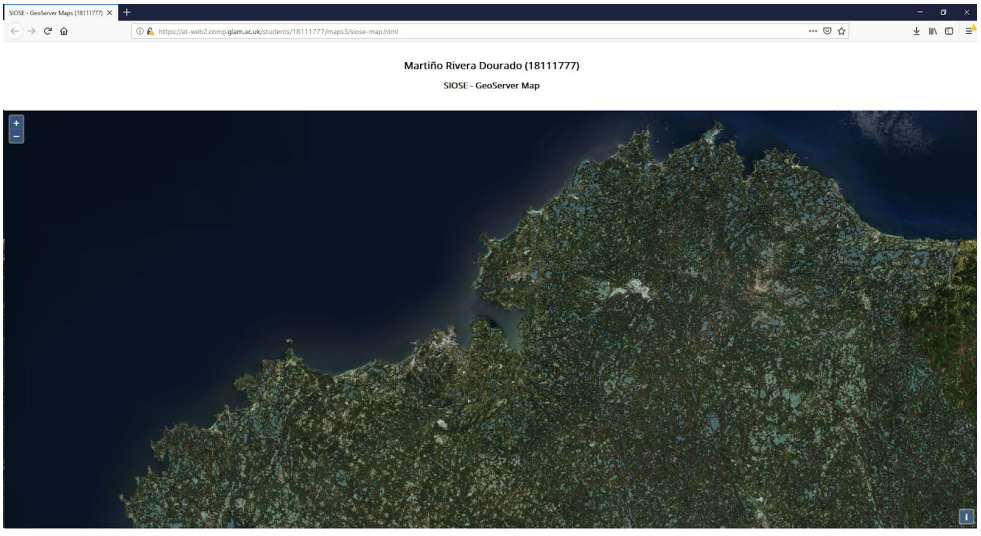
4.3 SIOSE map



4.3.1 Brief description

SIOSE (*Sistema de Información de Ocupación del Suelo de España*) [21], or *Spanish Land Occupancy Information System*, is the main source of information about the Spanish land. This is produced by each *Autonomous Community*[22] of Spain. This map shows the *SIOSE* correspondent to *Galicia* by loading it on top of a satellite layer and by modifying the opacity.

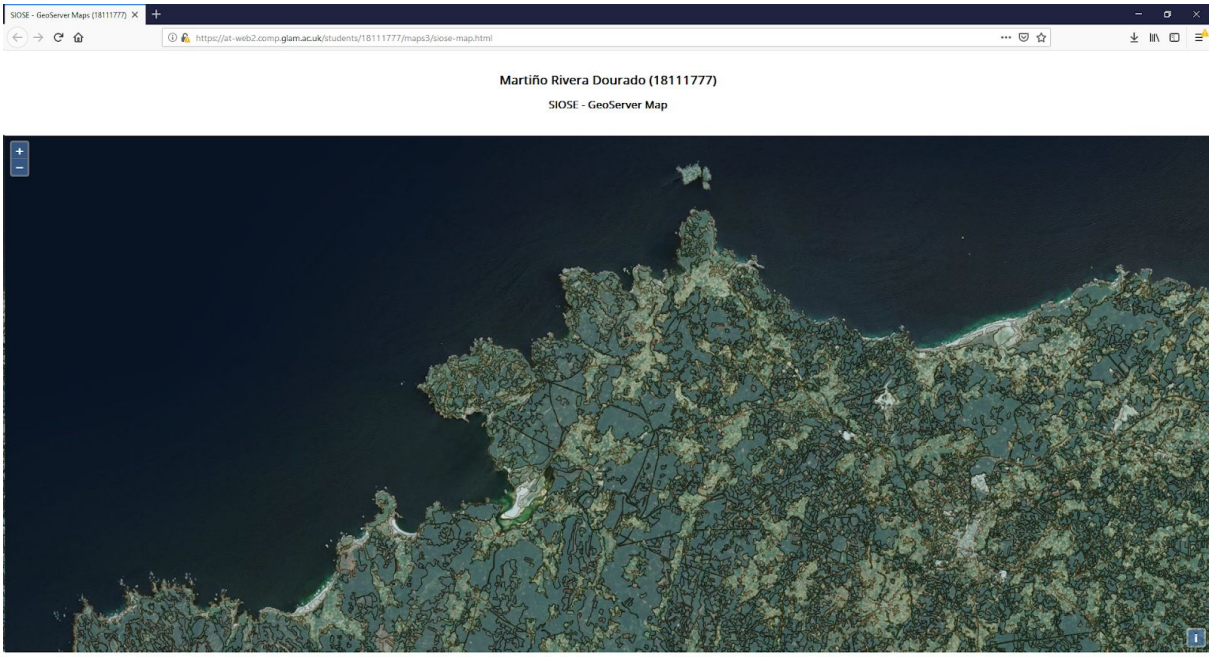
As it is a huge amount of information, the *GeoServer* can experience some delays processing and delivering the layer.

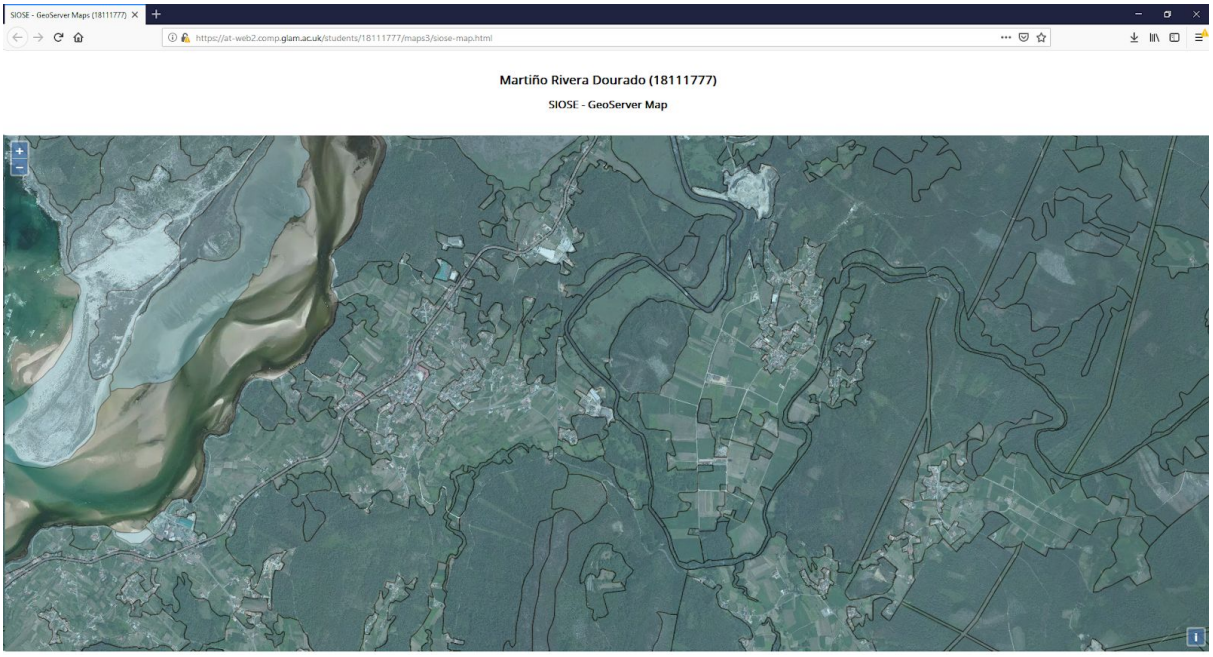


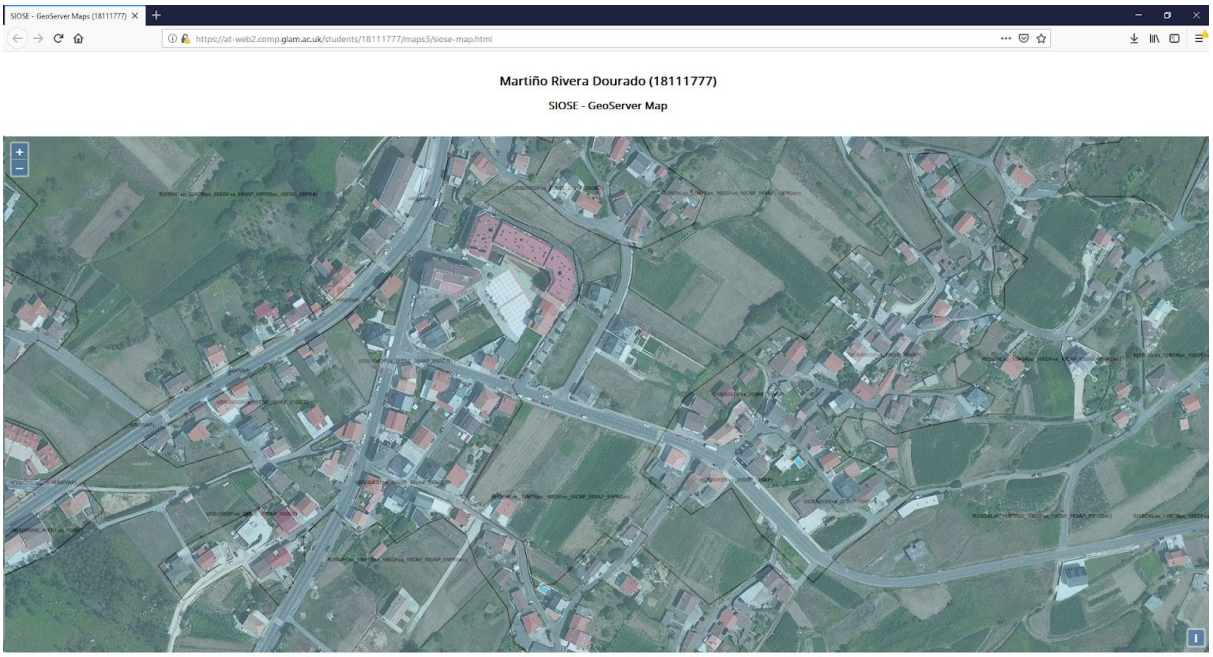
[21] http://www.siose.es/
[22] https://en.wikipedia.org/wiki/Autonomous_communities_of_Spain

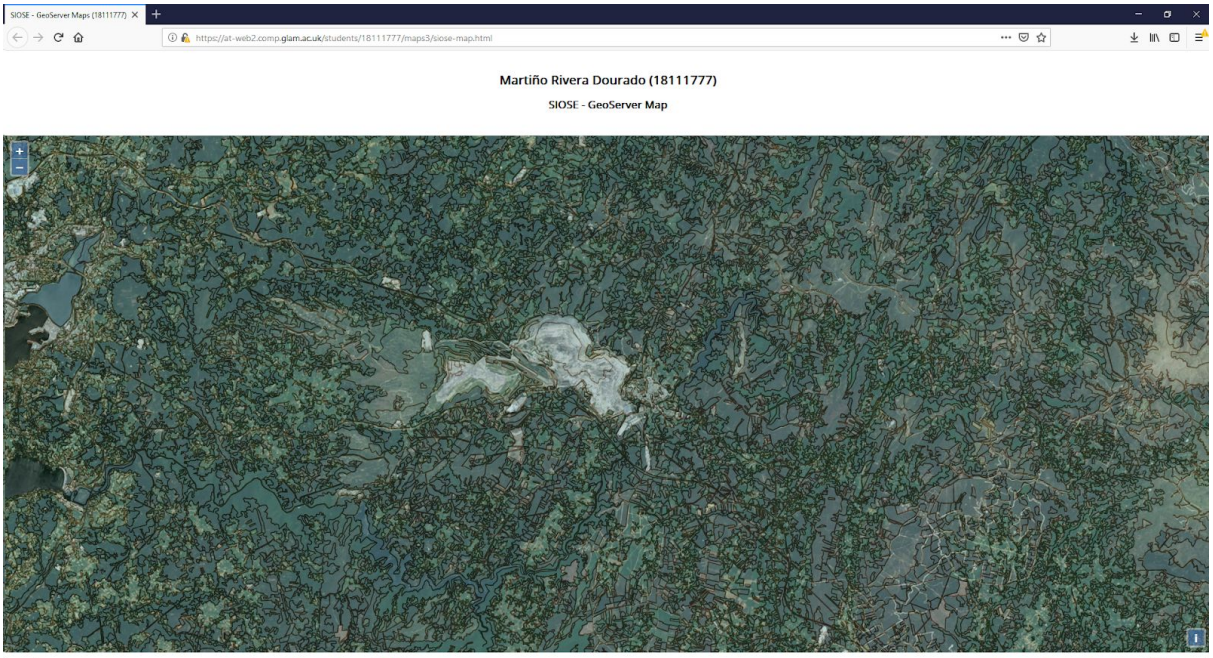
4.3.2 Data preparation and styling

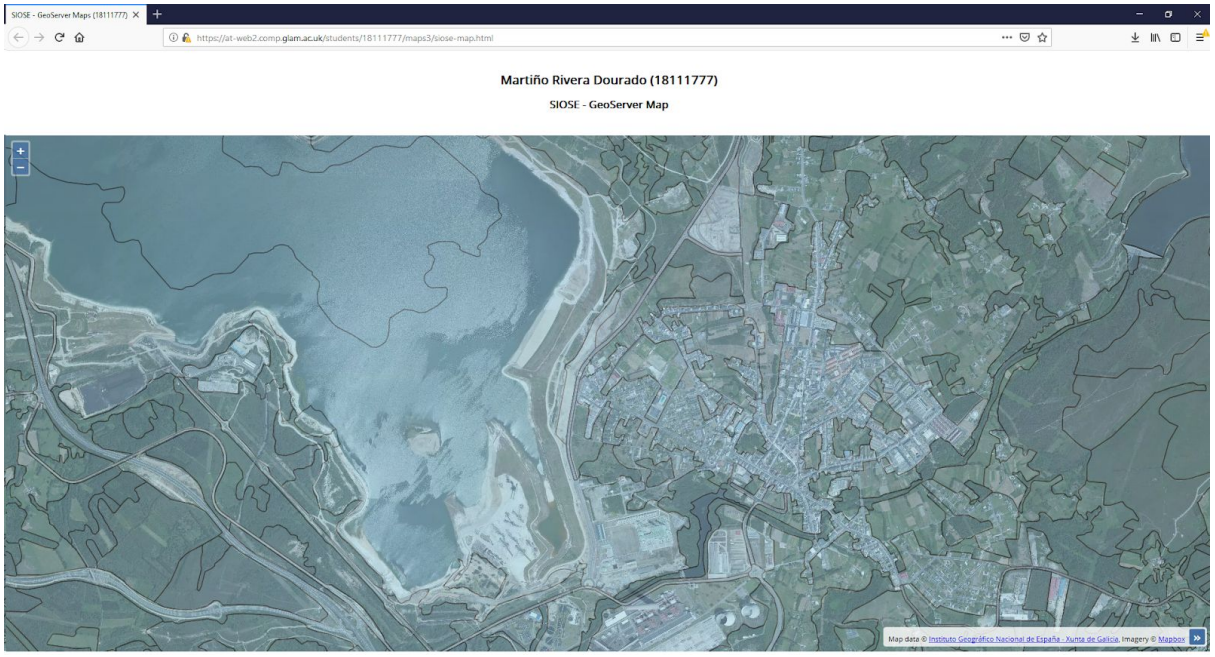
As with the previous maps, the data was loaded in *QGIS* and in *PostGIS*, creating the Store and the Layer in *GeoServer*. An SLD file was created, this time using the styling based on rules. That is, if the user zooms in, the layer shows the *SIOSE* code of each land area, as it can be seen in the following screenshots.

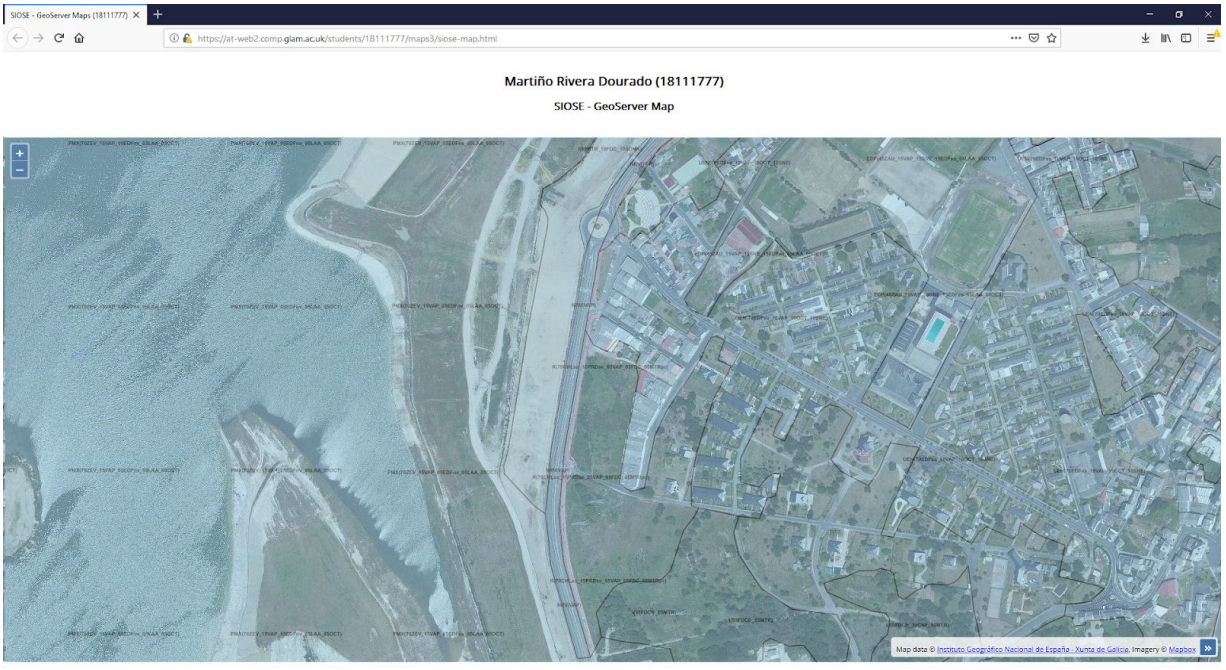





SIOSE - GeoServer Maps (18111777)
https://at-web2.comp.glam.ac.uk/students/18111777/maps3/siose-map.html
Martiño Rivera Dourado (18111777)
SIOSE - GeoServer Map


SIOSE - GeoServer Maps (18111777)
https://at-web2.comp.glam.ac.uk/students/18111777/maps3/siose-map.html
Martiño Rivera Dourado (18111777)
SIOSE - GeoServer Map


SIOSE - GeoServer Maps (18111777)
https://at-web2.comp.glam.ac.uk/students/18111777/maps3/siose-map.html
Martiño Rivera Dourado (18111777)
SIOSE - GeoServer Map
Map data © Instituto Geográfico Nacional de España - Xunta de Galicia, Imagery © Mapbox

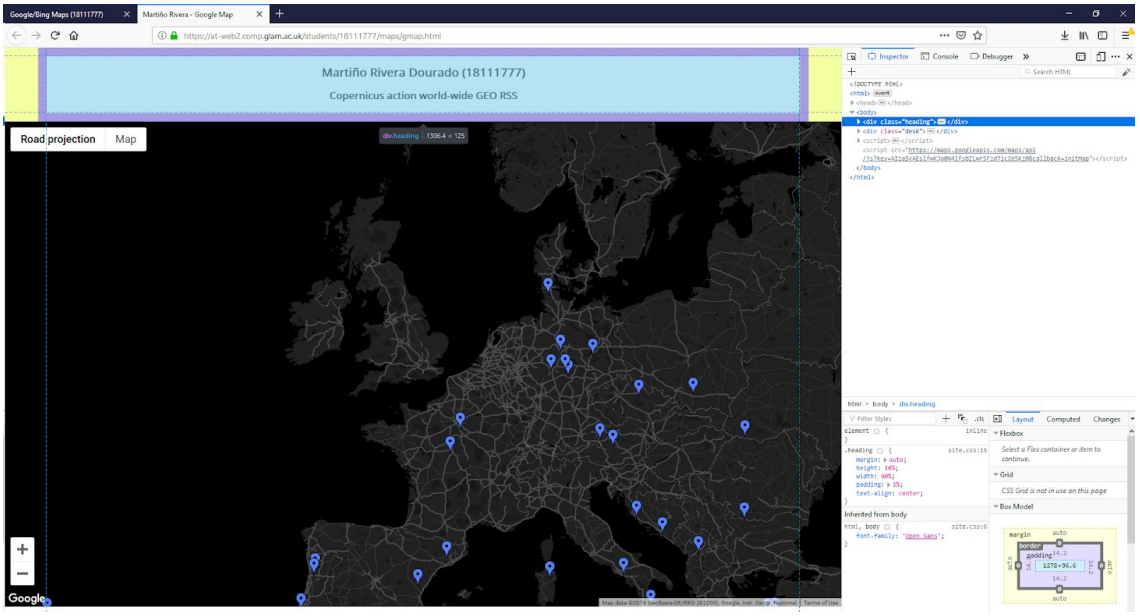

SIOSE - GeoServer Maps (18111777)
https://at-web2.comp.glam.ac.uk/students/18111777/maps3/siose-map.html
Martiño Rivera Dourado (18111777)
SIOSE - GeoServer Map
Map data © Instituto Geográfico Nacional de España - Xunta de Galicia, Imagery © Mapbox

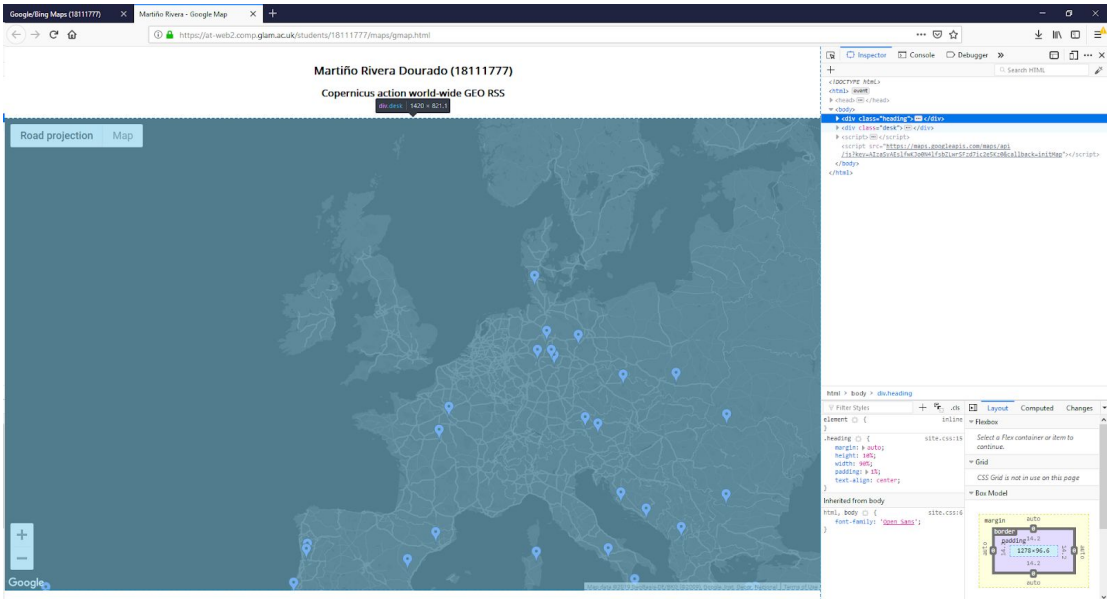
6. Web page layout

The design of the web page and all the maps aimed to be uniform. The styling details can be found in the css file `css/site.css`[23].

This design is adaptable. That is, if the window size is modified, the elements will adapt their size. That is applied to both maps and other elements. Additionally, the layout is designed to prevent the page to be bigger than the window, preventing to have a scroll that would ruin the user experience with the maps.

The layout is divided in the class `heading` and `desk`:

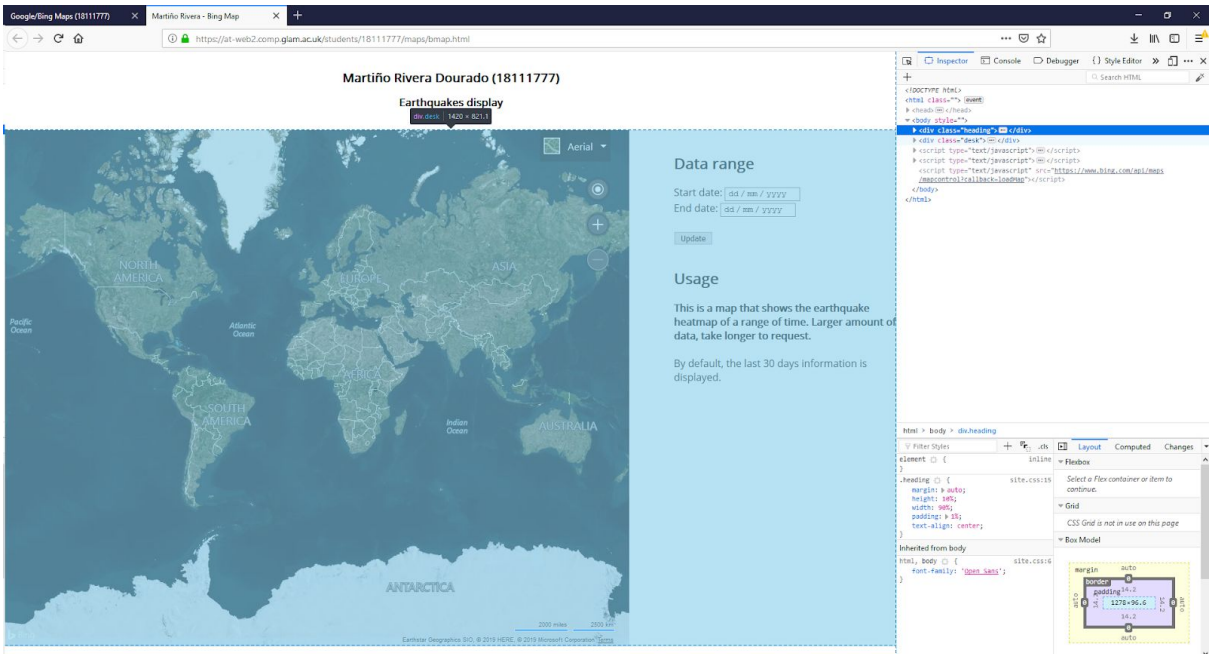


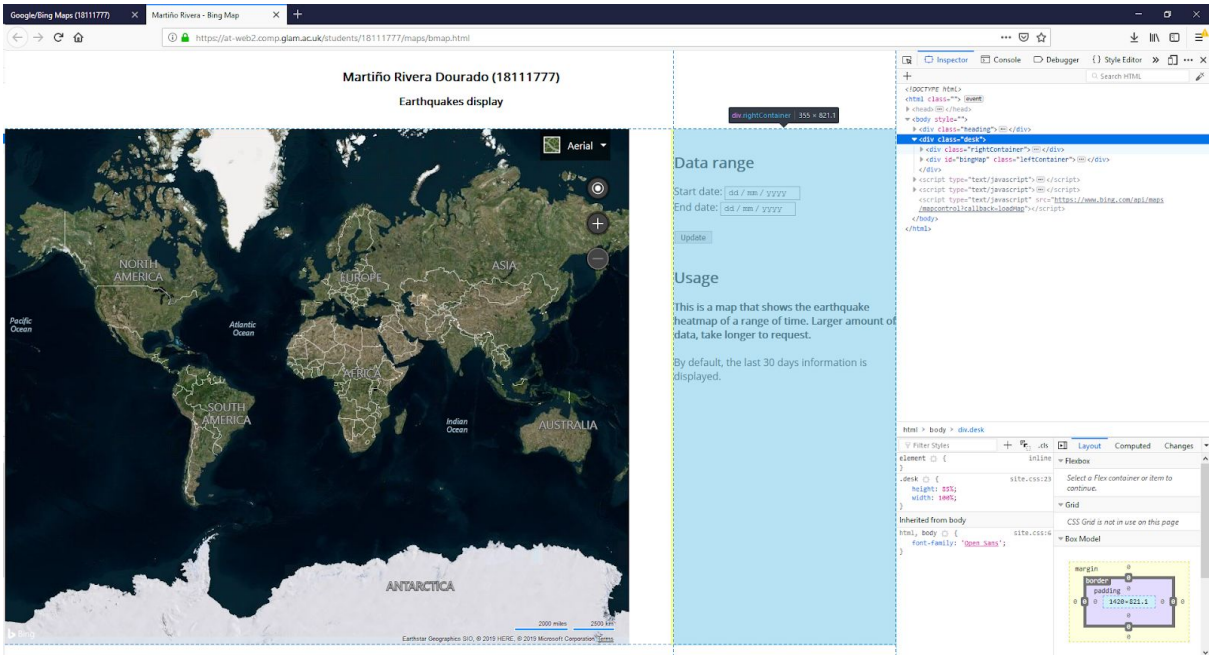


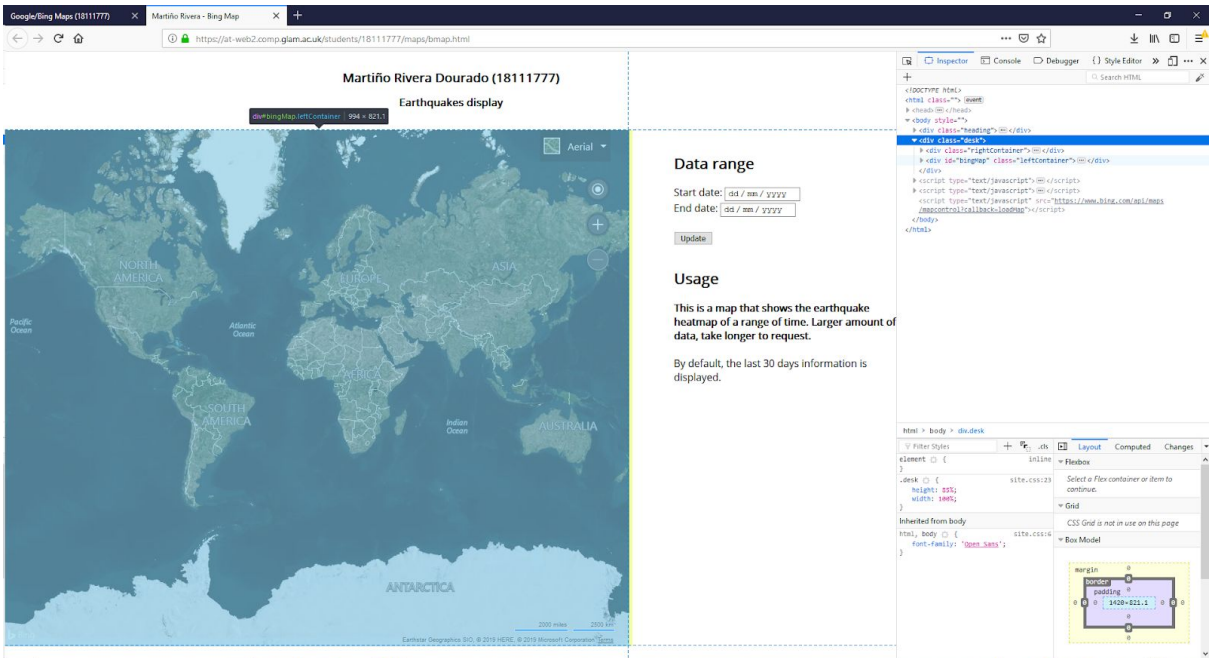
[23] Appendix A

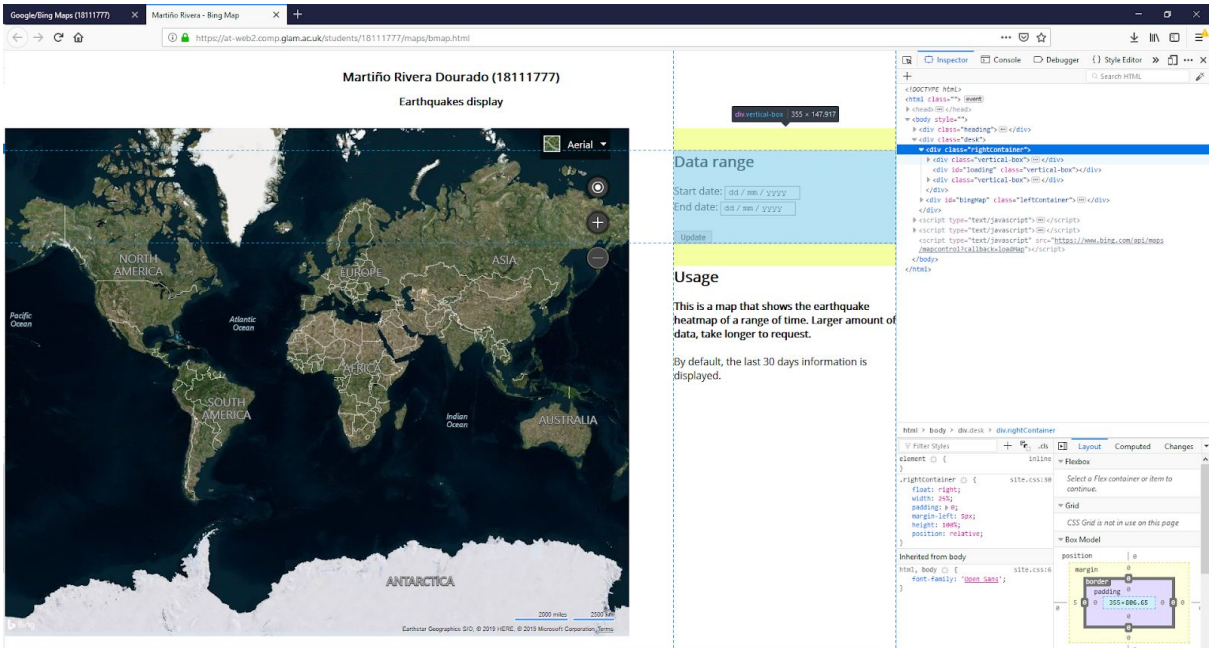
Usually, the pages where one map is placed, all the `desk` box is filled with the map itself. However, when additional html content is needed, the desk element is divided in `leftContainer` and `rightContainer`.

The `rightContainer` is also dividied in `vertical-box` elements:






Martiño Rivera Dourado (18111777)
Earthquakes display
Aerial
NORTH AMERICA
EUROPE
ASIA
AFRICA
SOUTH AMERICA
AUSTRALIA
ANTARCTICA
Pacific Ocean
Atlantic Ocean
Indian Ocean
Data range
Start date: dd / mm / yyyy
End date: dd / mm / yyyy
Update
Usage
This is a map that shows the earthquake heatmap of a range of time. Larger amount of data, take longer to request.
By default, the last 30 days information is displayed.


Martiño Rivera Dourado (18111777)
Earthquakes display
Aerial
NORTH AMERICA
EUROPE
ASIA
AFRICA
SOUTH AMERICA
AUSTRALIA
ANTARCTICA
Pacific Ocean
Atlantic Ocean
Indian Ocean
Data range
Start date: dd / mm / yyyy
End date: dd / mm / yyyy
Update
Usage
This is a map that shows the earthquake heatmap of a range of time. Larger amount of data, take longer to request.
By default, the last 30 days information is displayed.

7. Summary - reflective evaluation

Through the development of those web-based maps I have learned an extensive amount of technologies and I have developed some very useful skills.

Considering that the web is the interface of those *GIS* tools, I have gained experience in *JavaScript* language. I am now familiar with *DOM* elements and how to modify them. Besides, the effort to adapt the web pages to the window size has driven me to use *CSS* and design patterns. I have also used some *HTML5* elements as the forms. This has helped me to build web technologies skills based in *HTML* + *CSS* + *JS*.

In the other hand, using *QGIS* together with *PostGIS* and *GeoServer* has given me backend skills, as well as familiarity with *GIS* data processing. Besides, I have used KML, SHP, GeoJSON file formats, being able to load them in web-based maps.

Additionally, the use of *JS* Frameworks as *OpenLayers* or *JS API* as the *Google Maps API* has given me the knowledge about integrative programming and the use of *FOSS* and closed-source technologies.

Finally, the approach I have assumed in this coursework has improved my skills further. Self-learning based in code examples and documentation has helped me to better understand the technologies we were using in class. This, together with the lectures, made me understand the main idea behind those tools and technologies related to *GIS*.

Appendix A

js/getjson.js

```
// JS aditional functions

// AJAX function to load a JSON
function fetchJSONFile(path, callback) {
  var httpRequest = new XMLHttpRequest();
  httpRequest.overrideMimeType("appplication/json");
  httpRequest.onreadystatechange = function() {
       if ((httpRequest.readyState === 4) && (httpRequest.status === 200)) {
         var data = JSON.parse(httpRequest.responseText);
         if (callback) callback(data);
       }
  };
  httpRequest.open('GET', path);
  httpRequest.send();
  if ((httpRequest.readyState === 4) && (httpRequest.status === 200)) return
data;
};
```

css/site.css

```
/*This is the style file*/

@import url('https://fonts.googleapis.com/css?family=Open+Sans');

/* Basic */
html, body {
    font-family: 'Open Sans';
    width: 100%;
    height: 100%;
    margin: 0;
    padding: 0;
}

/* Layout: heading and desk */
.heading{
    margin: auto;
    height: 10%;
    width: 90%;
    padding: 1%;
    text-align: center;
}

.desk{
    height: 85%;
    width: 100%;
}

/* Layout: containers */

.rightContainer {
    float: right;
    width: 25%;
    padding: 0;
    margin-left: 5px;
    height: 100%;
    position: relative;
}

.leftContainer {
    float: left;
    width: 70%;
    padding: 0;
    margin-right: 5px;
    height: 100%;
    position: relative;
}

.vertical-box{
        margin-top: 10%;
        margin-bottom: 10%;
}

/* Font size heading */
.heading > h1{font-size:20px}
.heading > h2{font-size:17px}


/* Listing layout */
.inline-block {
        text-align: center;
        margin-top: 3%;
```

```
        margin-bottom: 3%;
        padding: 5px 5px 5px 5px;
}

.inline-block div {
        border-width: 3px;
        border-color: grey;
        margin-left: 5%;
        margin-right: 5%;
          display: inline-block;
}

/* MAPS layout */
#googleMap {
    height: 100%;
}

#bingMap {
    height: 100%;
}
/* Any map */
#map {
    height: 100%;
}

/* Animations */
#loading {
    animation: blinker 3s linear infinite;
    color: red;
    top: 30%;
    text-align: center;
    width: 100%;
    font-size: 1.15em;
}

@keyframes blinker {
  50% {
       opacity: 0;
  }
}

/* Other elements */
.coordinate-display {
       right: 8px;
        height: 25px;
        width: 380px;
        position: relative;
        padding: 2px;
        margin: 6px;
        border: 2px solid grey;
        border-radius: 10px;
        font-weight: bold;
        text-align: center;
}

#selected-name, #selected-coordinates {
        display: inline-block;
}
```