# MAP DISPLAY

## Index



## 1. Overview

This file explains how the data obtained is represented in a map, and how this map is shown to each client who requests it.

The aim is to have a webpage where the pollution map is shown. Map will ask for data obtained and will represent it. Then, the webpage containing the map will be hosted in a server and distributed to clients.

## 2. Map creation

Our willing is to create a map without relying on third parties as much as possible. Therefore, we will use our own programming code to create the map. As our intention is to develop a webpage (made in HTML, see section 4) the idea is to include the map within this code via JavaScript.

Different methods are known to create maps with JavaScript. Our decision has been to use the Mapbox GL library. Mapbox GL is a library which comes from Leaflet, the map creation library by excellence. The reason why we have chosen Mapbox instead of Leaflet is because the first one offers a prettier user interface and some design aspects which cannot be performed with Leaflet and which could be visually attractive for users, such showing 3D buildings once the map is very zoomed in.

From there, Mapbox GL uses different styles of map to represent data as shown in images 1 and 2. We want to create a coloured map based on the use of these styles (by mixing or by using a single one).

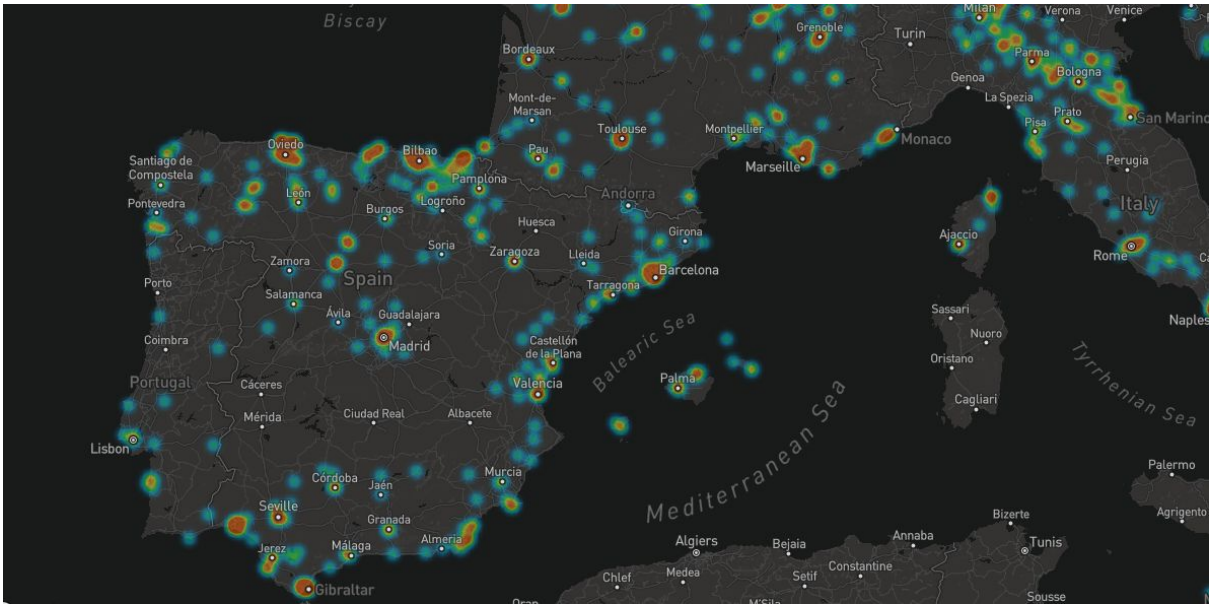


**Figure 1 : HeatMap layer made with Mapbox GL**

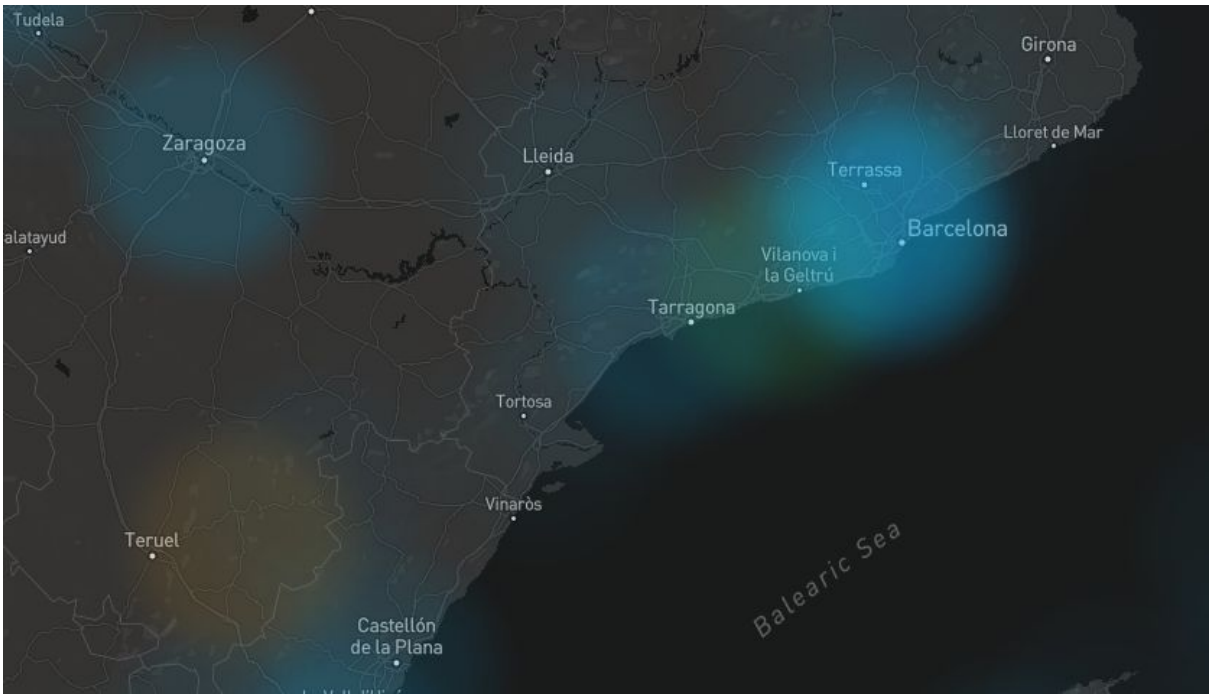


**Figure 2 : CirclesMap layer made with Mapbox GL**

Mapbox also offers the possibilities of layer controls (in client part) or zoom getting and setting, amongst other functionalities permitted to client.

## 3. Data retrieved and moved to map

Data will be retrieved from a MySQL database using PHP (see in section 4). In this database, there will be stored 3 tables, corresponding to 3 different layers we want to represent depending on the zoom at which user is.

The most external layer will have the data obtained from satellites. The other 2 ones will contain interpolated data.

As said, this data will be stored in a database. The retrieve of this data will be in format (longitude, latitude, pollutant value, interpolated/taken time), in order to show old data when needed.

The reason why we are using this format is because Mapbox needs a JSON with at least two parameters (coordinates in latitude and longitude, value or weight to represent - also known as GeoJSON) to perform a well-built map. These GeoJSON are made in the JavaScript code itself once data is retrieved, and passed as data to represent to the created map in Mapbox.

Layer division will be controlled setting a maximum and minimum zooms between a layer and the next/previous one: data taken from first table will only be visualized between zoom a and b. Same for the second and third table.

The algorithm for displaying will be simple: there will be a similar traffic light gradient color code which indicates the concentration of pollution in AQI scale. For every coordinate passed as data there will be that value for each pollutant plus a general one created in the interpolation process. Then, depending on that value one or other color will be associated to that coordinate in the map.

Data will be queried again in a 30 minutes period, when interpolation storage in database is finished.

## 4. Webpage and client

As explained lines above, the Map created with JavaScript will be shown in a webpage.

This webpage will be created in an HTML code for simplicity reasons: it is the best extension HTTP protocol can support when using the Internet connection.

We are not going to deep into the styling or structuring of this webpage, but only indicate the final file will have a PHP extension as the calling to database has to be performed in the server part.

```php
$sql = "SELECT latitud, longitud, pes FROM valors";
$result = mysqli_query($conn, $sql);

$data = array();
while ($enr = mysqli_fetch_assoc($result)) {
    $a = array($enr['latitud'], $enr['longitud'], $enr['pes']);
    array_push($data, $a);
}


?>

<nav id="menu"></nav>
<div id="map"></div>

<script>
    var dades = <?php echo json_encode($data); ?>;
```

**Figure 3 : Example of request to DB and data storing. Query is done using SQL code in PHP, but data is saved in a JavaScript variable (dades).**

This HTML (with PHP) will be served in the form that when a user does a request to the server using a browser, the webpage with the included map will be shown in the device screen.

## 5. Graphs and statistical data

Taking advantage of the JavaScript code, another kind of data display will be charts or graphs which represent statistical information. Specific requests will be done to the database between certain periods of time in order to know some values like the average, minimum or maximum concentration of pollution/pollutant in closed areas previously determined.

These values will be then represented using graphs that will show all this data, not specific ones for client. In other words, graph won't be created depending on the client request, but the client must search using filters the kind of data that requires between all available graphs.

Our idea is to use the D3 JavaScript library. As it can work with JSON formats and our data is stored in a GeoJSON form when using Mapbox, this data taken from the database can be both used for the map and the graphs creation.

Then, adding some kind of user interface, there will be the possibility of show graphs per day, month or year; per pollutant or in general; per zone…

## 6. Links and references

MySQL documentation: https://www.mysql.com/

Mapbox GL API documentation: https://docs.mapbox.com/mapbox-gl-js/api/