Documentation: Twin Cities Project

Table of Contents

[Introduction 2](#_Toc129347903)

[Database 2](#_Toc129347904)

[Website overview 3](#_Toc129347905)

[API’s 4](#_Toc129347906)

[Files 5](#_Toc129347907)

[Validation 6](#_Toc129347908)

# 

# Introduction

Twin cities are two cities that have been paired together to encourage social and cultural links. In this project, four of us worked as a group to create a web application using a local database and API’s that displays information on one UK city and its twin. Cities can have more than one twin city but for this project we chose Liverpool and its twin New Orleans.

The tasks completed were as followed:

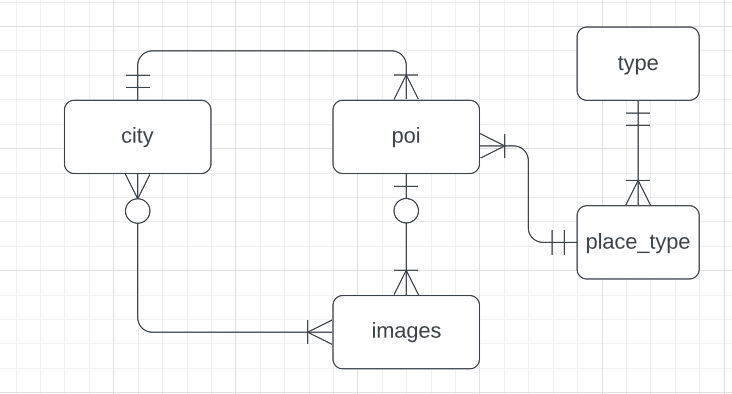
* Design and Create ER models that outline the entities and attributes and their relationship to each other.
* Implement a Database on a MySQL server
* Integrate two map API’s of the twin city locations with various points of interest that on a mouse Click lead to more detailed information pages of those locations.
* Design and use a configuration file that holds API keys
* Generate a RSS feed based on the data held in our MYSQL Database.

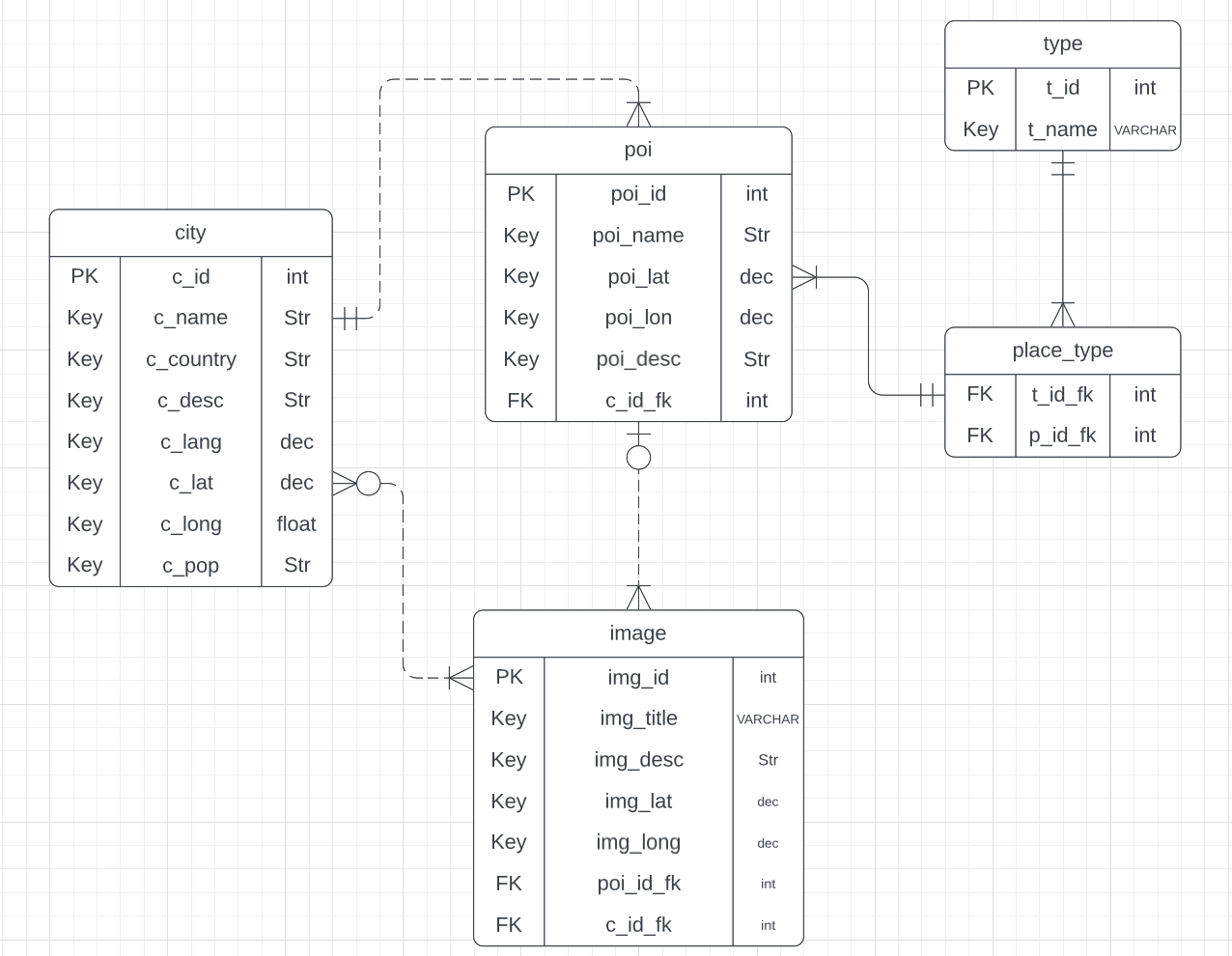
In addition to these group tasks, each team member selected an individual task that extended these requirements. The choices of our group were as follows:

* Document the Code and the System
* Make the website responsive using a CSS framework
* Integrate a Flickr widget.
* Integrate a twitter API

# Database

ERD Diagrams



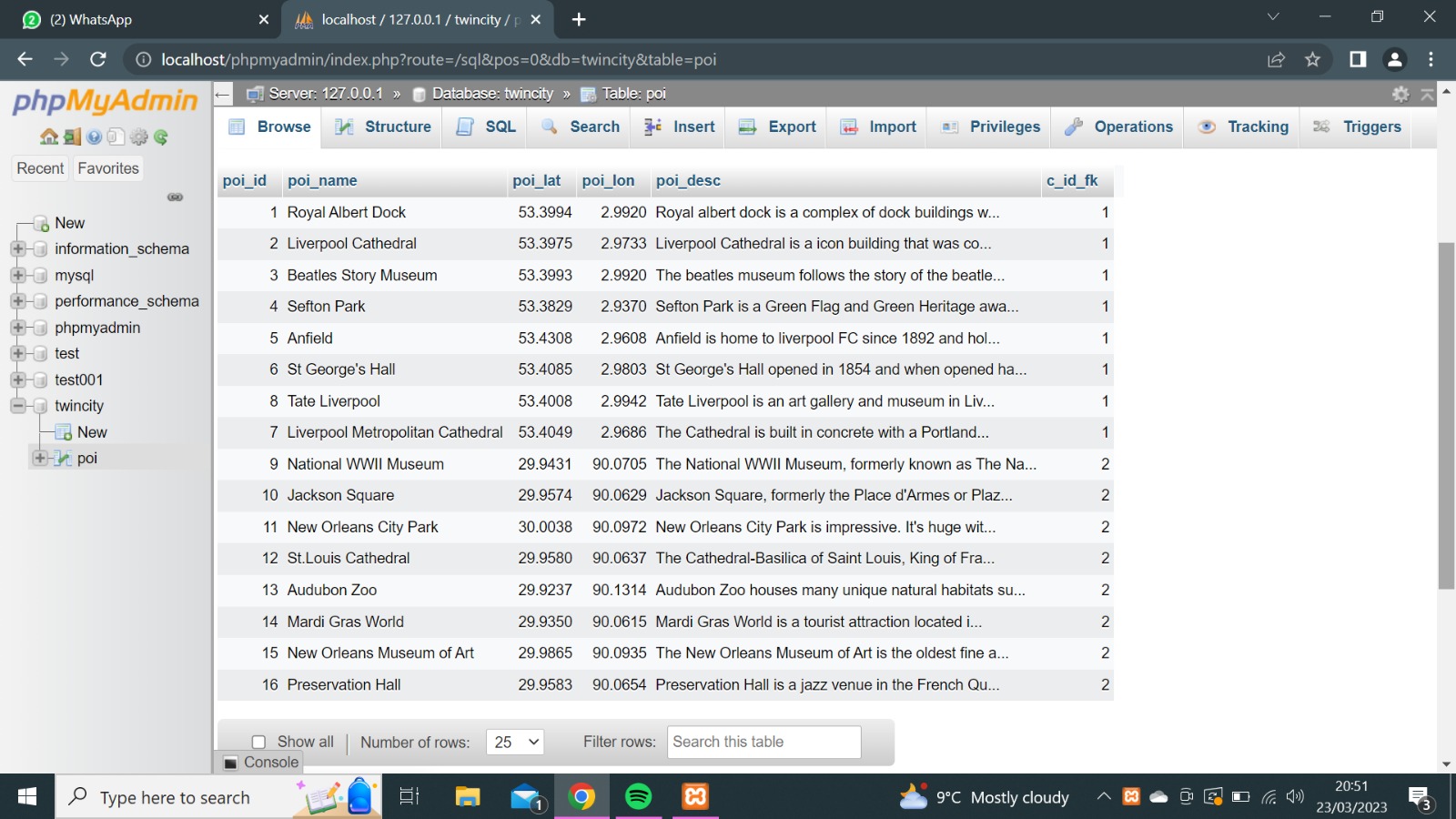


Attribute and Variable information:

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Meaning/Description** | **Data Type** |
| c\_id | Each City’s individual ID. This is the Primary Key for the city entity. | int |
| c\_name | The cities name | Str |
| c\_country | The country the city is in | Str |
| c\_desc | Description of the city | Str |
| c\_lang | The main language the city speaks | Str |
| c\_lat | The latitude of the city’s central point | Decimal |
| c\_long | The longitude of the city’s central point | Decimal |
| c\_pop | The population of the city | int |
| poi\_id | Point of interest ID, each different point of interest has its own key acting as the primary key for the POI entity. | int |
| poi\_name | Place of interest name | Str |
| poi\_lat | The latitude of the place of interest’s central point | Decimal |
| poi\_long | The longitude of the place of interest’s central point | Decimal |
| poi\_desc | Description of the place of interest | Str |
| img\_id | Each image’s individual ID, this is the Primary Key for the img entity | int |
| img\_title | The title of the image | Str |
| img\_lat | The latitude of the image | Decimal |
| img\_long | The longitude of the image | Decimal |
| img\_desc | A description of the image | Str |
| poi\_id\_fk | The point of interest foreign ID key, links entities together. | VARCHAR |
| t\_id\_fk | The types foreign key that links entities together. | VARCHAR |
| c\_id\_fk | The city foreign ID key, links entities together | VARCHAR |
| p\_id\_fk | The place foreign ID key, links entities together | VARCHAR |
| t\_id | The type of point of interest’s ID, acting as the primary key for the place\_type entity. | int |
| t\_name | The type of point of interest’s name. | Str |

MySQL

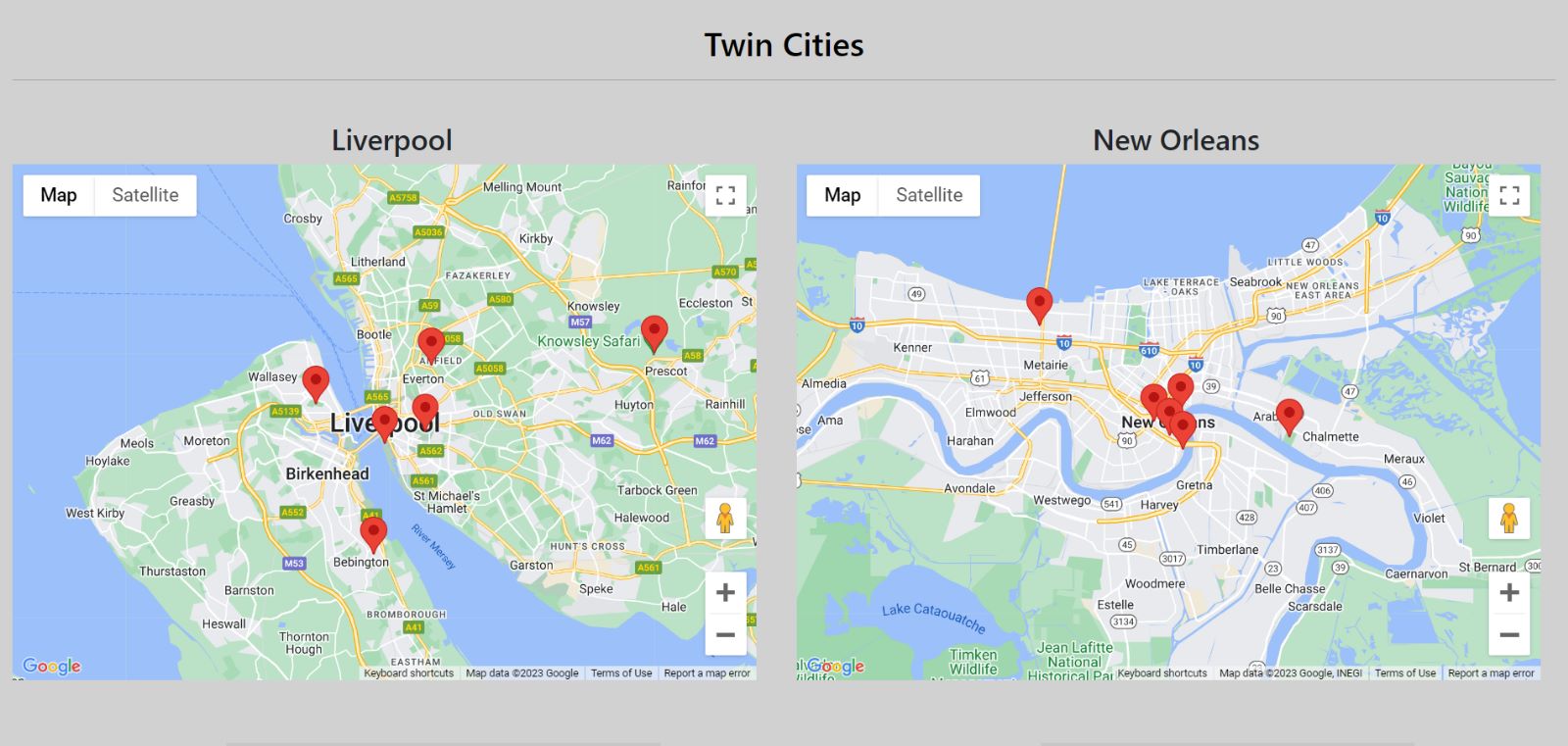
The database consists of one table holding all of the relevant information of the POI’s.



# Website overview

Design

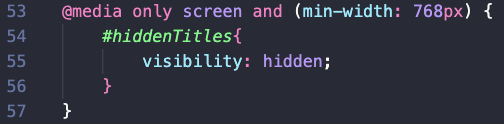
The website consists of one home page and 12 additional place of interest information pages.



The homepage shows the two twinned cities map API’s and the weather of each of these cities underneath their respective maps.

CSS

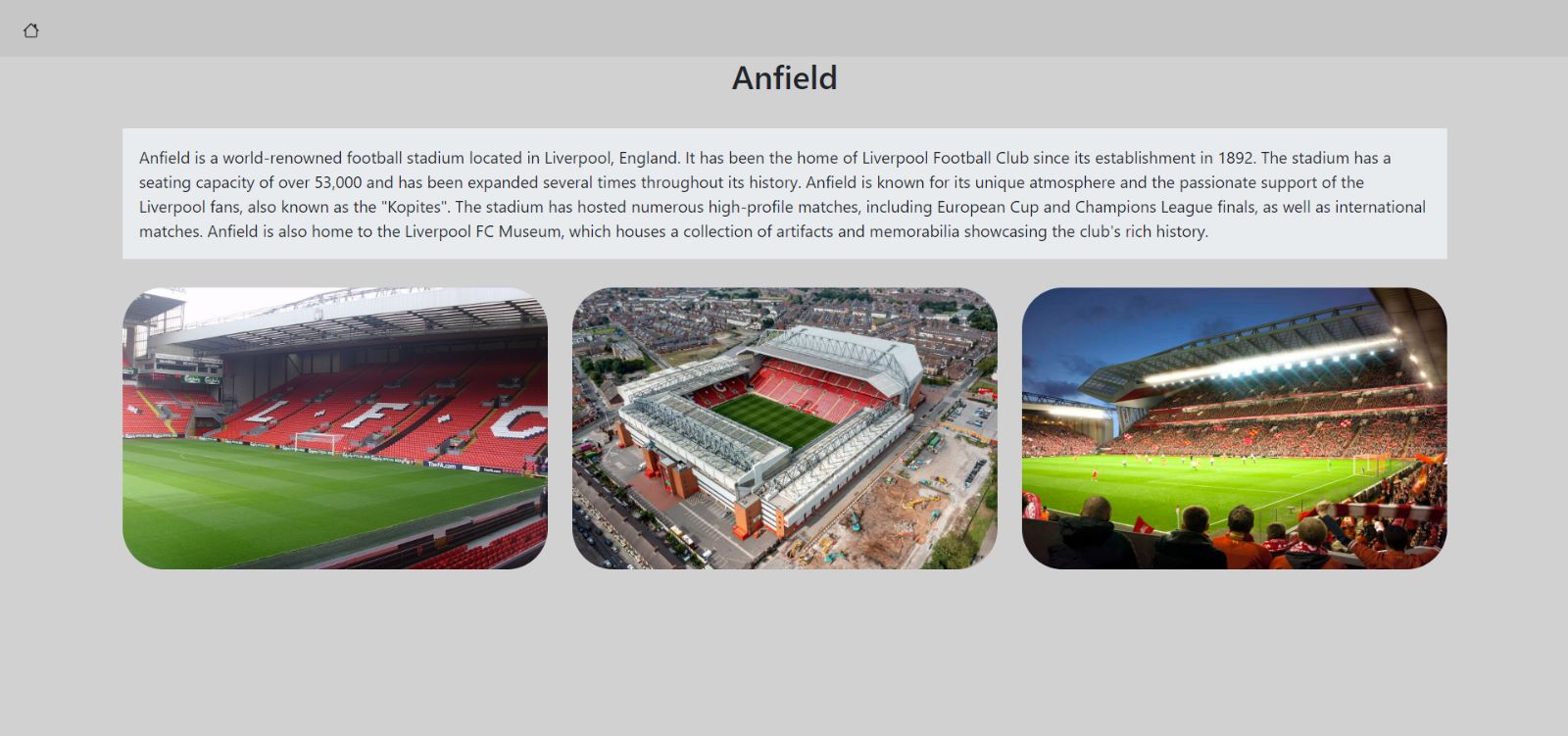
For responsiveness, if the window size is changed to anything above 768px then certain titles will be removed off the site to make the layout slicker. This is so people viewing the site using an iPad or phone will be able to see headers for the weather API’s so they can understand which weather API they are viewing. then the maps will no longer be side by side but are presented vertically with the weather.



Other CSS elements of the page are the styling elements such as background colour, fonts, spacing & margins and buttons, these are defined in an external style sheet which is linked to the index.php file in the header.



When a point of interest from the map is clicked on then the user will be directed to a new page that has a description of that point of interest as well as a collection of images of the site.



RSS feed

The RSS feed presents an updated feed of content from our MySql database.

# API’s

Four API keys were used for the group tasks of this application to configure the maps and the weather information for each city.

Map API

For the map for each city, we used Google API keys.

To get these we created an account using google cloud, created a project and then got two API keys. Available from: <https://cloud.google.com>

Weather API

For the weather API’s for each city we used the open weather map website. Available from: <https://openweathermap.org/api>

The weather API’s provide the following information: Current weather conditions, temperature, wind speed, humidity level, air pressure, the time of sunrise and the time of sunset.

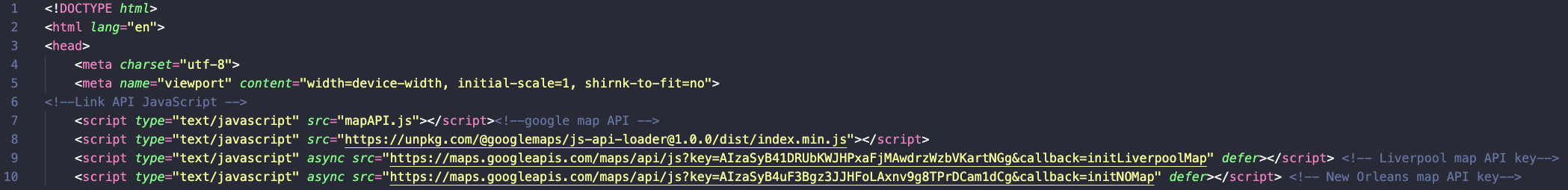
# Files

**README**

This file provides information about the configuration, requirements, troubleshooting, authors and licencing of the file.

**Index.php**

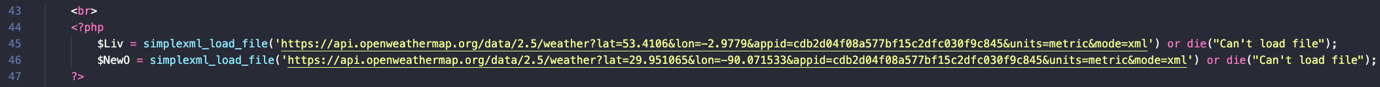
This is contains the main code for the site.



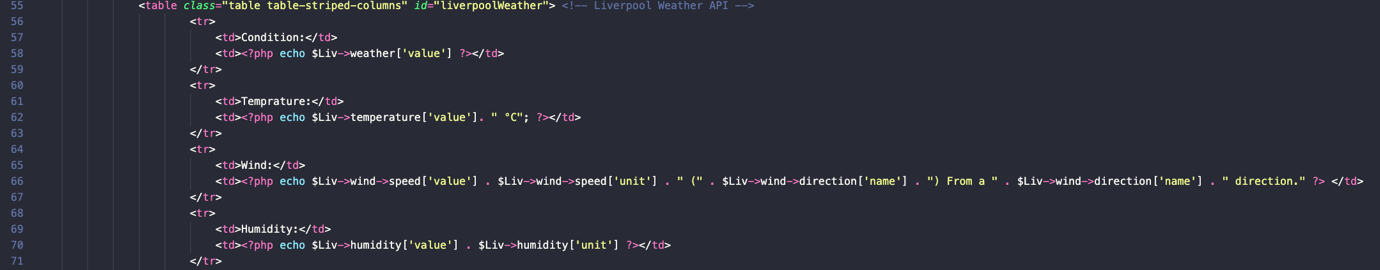
*This code links the API’s to the website using JavaScript.*

**

*\*Important\* This line of code enables the API’s to load together at the same time so that both of the Maps appear simultaneously on the page. Without this only one map at a time will display.*

**

*This code uses PHP to access the XML API files. It will show an error message if it cannot load the file.*

**

*To get the weather information the PHP echo method is used to print the data gathered from the API as demonstrated in the above code.*

**mapAPI.js**

This file configures the API’s using JavaScript. It also configures the points of interest into clickable markers on the map.

****

*The initialize method enables the API’s to load. The variables defined in this section of code configure the exact location that each map should display using latitude and longitude. “mapTypeID” selects which type of map is displayed from the google options of roadmap or satellite image.*

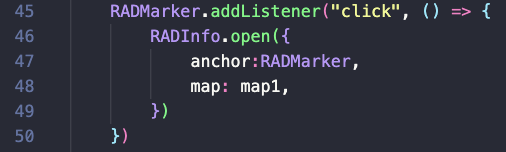
**

*This creates two variables to hold the information of the map API – they are called later during the implementation of the marker pins. (Explained below)*

**

*Point of interest marker pins- This code creates the marker pins and their locations.*

*The “map:map1” part assigns which map the point of interest belongs to. Map 1 refers to Liverpool, map 2 is New Orleans. This part also assigns information to the content variable such as the name and description of the point of interest. RADContent means “Royal Albert dock Content” each different point of interest has this variable but with different names holding different information but in the same layout.*

**

*This adds a listener event so that the user can see the information assigned in the RADContent variable when they click on the marker.*

**stylesheet.css**

This file creates the CSS styling for the site including font, sizes of photos and text areas, background design and colours etc.

**configuration.xml & config.php**

The configuration file’s purpose is to keep more sensitive data, such as the API keys and database information off of the main script files.

# Validation

We validated our website’s HTML using the W3C validator: <https://validator.w3.org>

We also validated our RSS feed using the W3C validator: <https://validator.w3.org/feed/>

The outcomes of this was a valid HTML site and RSS feed.