WebMapping

Carte interactive du chemin principale de Saint-Jacques-de-Compostelle

https://www.routard.com/

Master 2 Géomatique

Nathan Folmer
Nicolas Ferrer
2022/2023

SOMMAIRE

1. Introduction **2**

2. Sujet **2**

3. Base de données **3**

3.a) Récupération et traitement de la donnée 3

3.b) Intégration dans une base de données 4

3.c) Difficultés rencontrées 5

4. Serveur **6**

4.a) Serveur Apache 6

4.b) Requêtage avec le langage PHP 6

4.c) Difficultés rencontrées 7

5. Client **7**

5.a) Structure de la page HTML 8

Librairies 8

Couche de style CSS 8

Création de menu déroulant et de bouton 8

5.b) Interactivité via le langage Javascript 9

Eléments de base de la carte leaflet 9

Overlays 9

Récupérations des points d'intérêt via le script php 10

Technologies secondaires 11

6. Pistes d'évolution **11**

7. Bibliographie **12**

7.a) Références 12

7.b) Données utilisées 12

1. Introduction

Le webmapping est une technologie qui permet aux utilisateurs de visualiser des données géographiques dans un navigateur Web et d'accéder facilement des informations plus ou moins complexes à l'aide d'un navigateur Web. Grâce à la librairie Leaflet de Javascript, nous réaliserons une carte interactive mettant en avant des informations localisables dans l'espace. Les données utilisées dans ce projet seront disposées dans une base de données locale et appelées via un serveur pour être ensuite affichées dans un client. À travers ce rapport, nous présenterons l'ensemble de cette infrastructure et des outils mis en place dans cette application web cartographique.

2. Sujet

Le chemin de Saint-Jacques-de-Compostelle est l'un des principaux chemins de pèlerinage médiéval, qui relie le Puy-en-Velay, en France, à Santiago de Compostela, en Espagne. Ce chemin est aussi connu sous le nom de « chemin de Saint-Jacques » ou de « chemin de Compostelle » et est considéré comme l'un des pèlerinages les plus importants et les plus anciens de l'Occident chrétien. Le chemin part du Puy-en-Velay et passe par le Massif Central et le sud-ouest de la France avant de traverser les Pyrénées pour entrer en Espagne. Les randonneurs qui suivent ce pèlerinage rencontrent de magnifiques points de vue, paysages, monuments et églises le long du chemin. Ce parcours initiatique donne l'opportunité aux pèlerins la rencontre avec l'inconnu et la découverte des cultures locales au travers des nombreux territoires traversés.

Pour notre application, nous avons choisi de traiter la partie du chemin français. Notre zone s'étend de Puy-en-Velay jusqu'à la frontière espagnole, près de la ville de Saint-Jean-Pied-de-Port. Par cette application, nous voulons offrir aux pèlerins un ensemble d'informations de premier ordre qui se voit essentiel lors d'un tel périple. Ainsi, sous forme d'une carte, nous souhaitons représenter les endroits où peuvent loger les randonneurs, les zones à visiter absolument, les petits conseils et lieux stratégiques pour se reposer sur le chemin, les bâtiments où il est possible de faire tamponner la crédencial (Passeport du pèlerin) et enfin les lieux pour s'alimenter.

Notre projet prend la forme d'une infrastructure 3 Tiers. Une base de données qui regroupent dans 5 classes l'ensemble des informations utiles aux pèlerins. Un serveur qui interroge cette base via le PHP. Un client qui construit et affiche les informations dans une carte interactive au sein d'une page web. Cette dernière correspond à l'interface des utilisateurs.

3. Base de données

3.a) Récupération et traitement de la donnée

Avant de commencer, nous avons dû définir les données nécessaires à la création de notre application.

Nous avons tout d'abord tracé le chemin de Compostelle dans QGIS en nous aidant du Scan 25, qui est une base de données géographiques contenant des données sur les villes, les routes, les lacs, les cours d'eau et les frontières, récupérées avec le flux WMS.

Ensuite, nous avons récupéré les couches de départements et de communes fournies par la BD TOPO de l'IGN. Nous en avons extrait les zones d'intérêt, c'est-à-dire, les communes traversées par le chemin de Compostelle.

Après, nous avons défini les données que nous voulions intégrer dans notre application. Nous avons sélectionné 11 classes regroupées en 5 thèmes :

- S'alimenter : Restaurants et boulangeries.
- Pour dormir : hôtels et campings.
- Utile sur le chemin : Fontaines et pique-niques.
- A visiter : Musées, point de vue et bâtiments remarquables.
- Tampon de la credencial : Mairies et offices de tourisme.

Ces données ont été récupérées sur OSM (*Figure 1*).

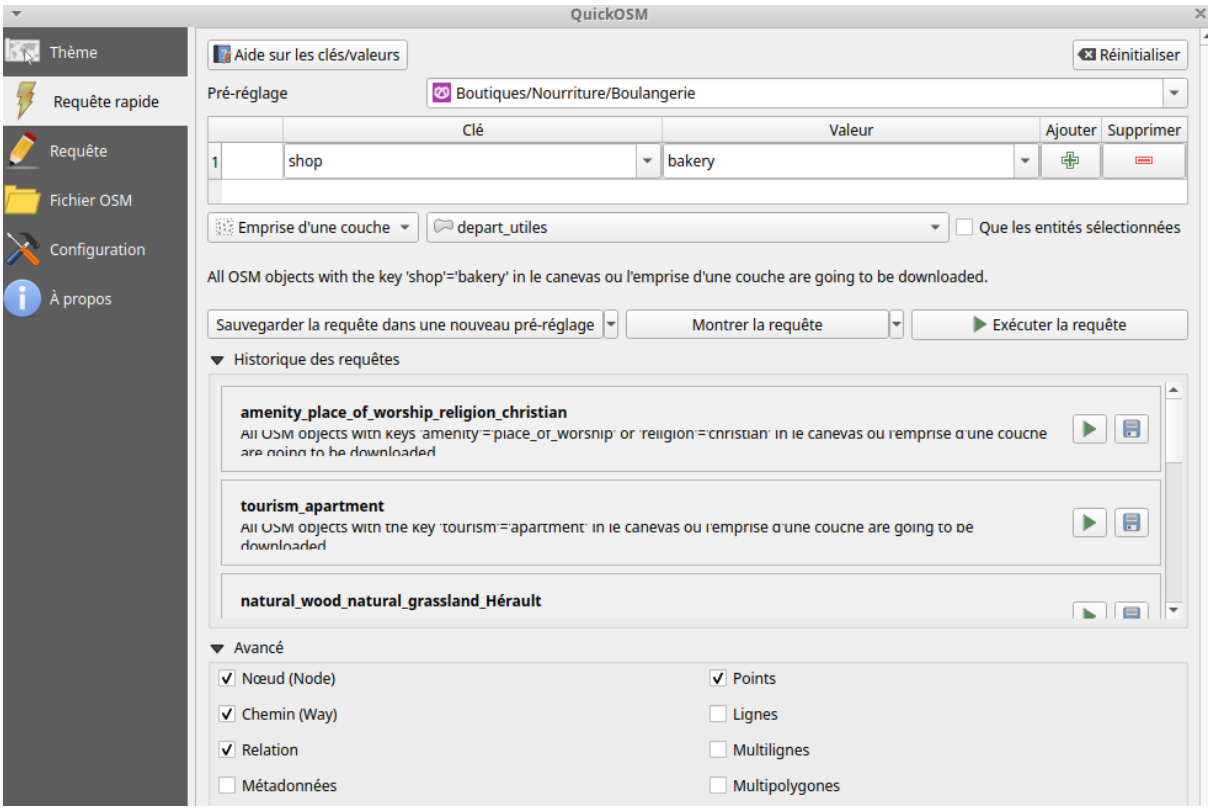


Figure 1: extraction de la donnée via QuickOSM

Les données étant très volumineuses, nous avons choisi de représenter seulement celles se trouvant dans un rayon de quatre kilomètres autour du chemin. Nous avons choisi quatre kilomètres car les utilisateurs seront principalement à pied et doivent avoir accès à des informations à une distance acceptable.
Les données ont été nettoyées car les couches possèdent une multitude de champs souvent vide. Nous avons décidé, pour chaque couche, d'utiliser le champ "name" et de créer un champ "classe" qui distingue le type de point d'intérêt au sein de chaque classe. Par exemple, pour la table "alimenter", dans le champ "classe", nous avons le choix entre "boulangerie" ou "restaurant" (*Figure 2*).

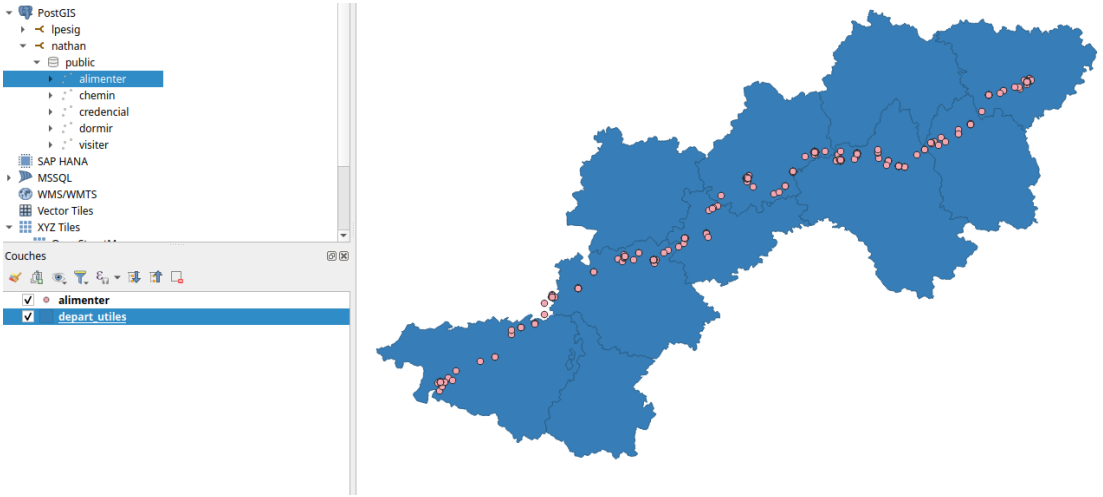


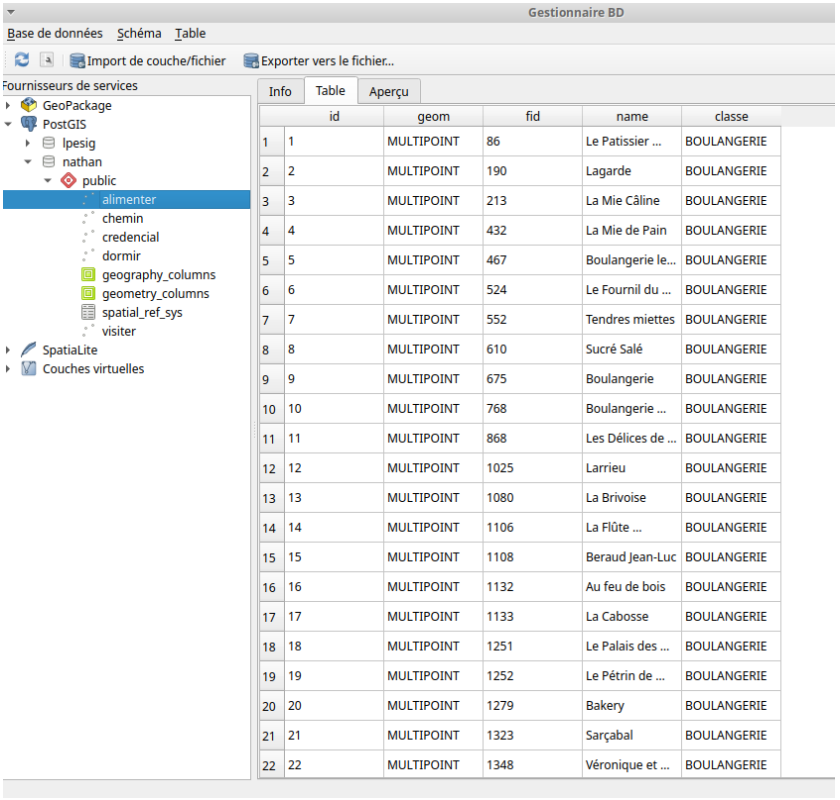
Figure 2 : Affichage des points d'alimentation le long du chemin

La dernière étape avant l'intégration des données dans une base de données fut leurs re-projections en WGS84 (nécessaire pour leurs affichages) et leurs conversions en geojson.

3.b) Intégration dans une base de données

PostgreSQL est un système de gestion de base de données relationnelles open source. Il est utilisé pour stocker et gérer des données géographiques, notamment des données JSON, et peut-être intégrées à des applications Webmapping.

Via l'application Pgadmin, nous avons créé une base de données PostgreSQL. Pour pouvoir l'utiliser sous qgis, nous avons ajouté l'extension postgis via la requête :
"create extensions PostGIS"
Dans Qgis, nous avons ensuite créé une nouvelle connexion à PostgreSQL. Une fois ce travail réalisé, nous avons importé les données au format geojson précédemment extraites d'OSM directement dans la base qui s'appelle "nathan".



	id	geom	fid	name	classe
1	1	MULTIPOINT	86	Le Patissier ...	BOULANGERIE
2	2	MULTIPOINT	190	Lagarde	BOULANGERIE
3	3	MULTIPOINT	213	La Mie Câline	BOULANGERIE
4	4	MULTIPOINT	432	La Mie de Pain	BOULANGERIE
5	5	MULTIPOINT	467	Boulangerie le...	BOULANGERIE
6	6	MULTIPOINT	524	Le Fournil du ...	BOULANGERIE
7	7	MULTIPOINT	552	Tendres miettes	BOULANGERIE
8	8	MULTIPOINT	610	Sucré Salé	BOULANGERIE
9	9	MULTIPOINT	675	Boulangerie	BOULANGERIE
10	10	MULTIPOINT	768	Boulangerie ...	BOULANGERIE
11	11	MULTIPOINT	868	Les Délices de ...	BOULANGERIE
12	12	MULTIPOINT	1025	Larrieu	BOULANGERIE
13	13	MULTIPOINT	1080	La Brivoise	BOULANGERIE
14	14	MULTIPOINT	1106	La Flûte ...	BOULANGERIE
15	15	MULTIPOINT	1108	Beraud Jean-Luc	BOULANGERIE
16	16	MULTIPOINT	1132	Au feu de bois	BOULANGERIE
17	17	MULTIPOINT	1133	La Cabosse	BOULANGERIE
18	18	MULTIPOINT	1251	Le Palais des ...	BOULANGERIE
19	19	MULTIPOINT	1252	Le Pétrin de ...	BOULANGERIE
20	20	MULTIPOINT	1279	Bakery	BOULANGERIE
21	21	MULTIPOINT	1323	Sarçabal	BOULANGERIE
22	22	MULTIPOINT	1348	Véronique et ...	BOULANGERIE

Figure 3 : Intégration des données dans PostGIS

La *figure 3* ci-dessus représente un exemple pour la classe “alimenter”. On peut voir que toutes les "features" ont un index spatial et son de type point. De plus, pour chacun d’eux, on a le nom et la classe.

3.c) Difficultés rencontrées

La première difficulté rencontrée lors de cette première partie du travail a été la numérisation du chemin de St-Jacques car nous n’avons pas trouvé au format shapefile, le sentier complet accessible gratuitement. Ainsi, ce dernier a été vectorisé manuellement à un par d’un scan 25 de l’IGN.
Le second problème a été dans un premier temps, de nommer les tables et les champs en majuscules. En effet, dans l’écriture du script PHP, les majuscules empêchent de faire des requêtes SQL opérationnelles. Nous avons dû renommer nos tables et les noms des champs en minuscule pour que le script fonctionne.
Le troisième problème était le SCR des couches GEOJSON. La carte Leaflet, ayant comme système de projection EPSG4326, n’arrivait pas a afficher les données OSM projetées en EPSG2154. Il a été donc nécessaire de re-projeter toutes nos données pour qu’il y ait un affichage correct dans la carte interactive.
Enfin, les données libres et gratuites d’OSM ne comprennent que partiellement les informations sur les gîtes étapes et églises du chemin de St-Jacques. Ces éléments ne seront donc pas implémentés.

4. Serveur

4.a) Serveur Apache

Apache est un serveur web qui gère les requêtes HTTP envoyées par les utilisateurs via un client. Pour communiquer avec la base de données PostgreSQL, nous avons téléchargé l'extension "**mod_dbd_pgsql**" qui permet de se connecter à notre base. L'idée est de pouvoir utiliser des commandes SQL pour accéder et manipuler les données en utilisant une API de programmation telle que PHP.
D'autre part, le serveur Apache permet aussi de stocker des fichiers composant notre site web :

- Le fichier javascript.
- Le fichier css.
- Document html.

4.b) Requêtage avec le langage PHP

Via le langage PHP, nous nous connectons à notre base de données. Il est nécessaire d'informer le nom de la base, le nom de l'utilisateur, le mot de passe.
Ensuite dans l'exemple ci-dessous (*Figure 4*), nous réalisons une requête pour extraire de la table "alimenter", le nom, la géométrie et la classe de chaque objet. Nous récupérons aussi le résultat de la requête si le résultat est null dans le champs "classe". Il faut préciser que nous récupérons la géométrie de chaque objet que nous convertissons en geoJson.

```php
<?php
    $config_dbname = 'nathan'; // nom de la base de données
    $config_user = 'nathan'; // utilisateur ayant accès à la base
    $config_password = 'nathan'; // mot de passe de l'utilisateur
    $pg_conn = pg_connect("host=localhost dbname=$config_dbname user=$config_user password=$config_password");
    pg_set_client_encoding($pg_conn,"UNICODE"); // déclaration du jeu de caractèresen unicode pour la gestion correcte des accents

    if (isset($_GET["classe"]) && $_GET["classe"] != "tout") {
        $sql = "SELECT ST_AsGeoJSON(geom) as geojson, name, classe  FROM alimenter WHERE classe = '" . $_GET["classe"] . "'";
    }
    else {
        $sql = "SELECT ST_AsGeoJSON(geom) as geojson, classe, name FROM alimenter WHERE classe=null" ;
    }
    $result = pg_query($pg_conn,$sql); //result est le résultat de ma requête

    $feature = array(); // on créer le tableau vide
    while ($row = pg_fetch_assoc($result)){
        $res['name'] = $row['name'];
        $res['classe'] = $row['classe'];
        $geom = $row['geojson']; // chargement de la colonne géométrique en GeoJSON

        $feature[] = '{"type": "Feature", "geometry": ' . $geom . ', "properties": '. json_encode($res) . '}'; // création de l'objet GeoJSON contenant la gé
    }
    echo '{"type": "FeatureCollection", "features": [' . implode(', ',$feature) . ']}'; // liste de tous les objets GeoJSON provenants de la base
    //echo json_encode($feature);


?>
```

Figure 4 : Requête appelant la base de données

Dans la page javascript de notre projet, nous appelons le résultat de cette requête grâce à $.getJSON et l'ajouter à notre carte leaflet. Tout l'intérêt du code consiste à pouvoir afficher séparément les markers en fonction de leur classe. Le choix se fait en fonction d'un menu déroulant que nous présenterons dans la partie suivante. Dans l'idée, en fonction du choix de la valeur du menu déroulant (soit boulangerie, soit restaurant, soit rien), nous récupérons les objets grâce à $get.JSON pour les objets qui ont une classe égal à "value". Nous affichons ensuite le résultat sur la carte (*Figure 5*).

```
var alimentation;


function afficheAlimentation(filter) {
    value = document.getElementById("alimentation").value;
    if (typeof alimentation !== 'undefined') {
        console.log(value);
        alimentation.removeFrom(map);
    }
    alimentation = null;
    if (value =="Restaurants") {
        classe = "RESTAURANT"
    }
    else if (value =="Boulangeries"){
        classe = "BOULANGERIE"
    }
    else
        {classe="null"}

    $.getJSON("PHP/alimenter.php?classe="+classe, function( data ) {
        alimentation = L.geoJSON(data, {
            onEachFeature: nameElement
        })
        alimentation.addTo(map);
    });
}
```

Figure 5 : Fonction qui appelle la table “alimentation”

4.c) Difficultés rencontrées

Il nous a été assez compliqué de réaliser un affichage de ces objets en fonction de leurs classes. Nous avons perdu beaucoup de temps pour des erreurs mineures comme le fait de mettre 1 seul “=” au lieu de deux. De plus, les markers ne changeaient pas lorsque nous modifions la valeur de “value” dans le menu déroulant. Ainsi nous avons donc dû réaliser une suppression des markers avant d’en afficher d’autres ayant une autre classe.

5. Client

Le html est un format de données pour encoder et structurer tous les contenus du page html. Lorsque l’on regarde une page d’un site web, on reçoit des documents html et css envoyés par le serveur. C’est le navigateur web qui interprète l’information html et css pour créer et afficher la page. L’information interactive des pages internets et créés par le document javascript du serveur. Le code javascript peut modifier l’information visualisable sur la page internet.

- Le html : Détermine ce que l’on va voir, le contenu en somme.
- Le css : Correspond au style de la page internet.
- Le javascript : Le côté interactif de la page.

5.a) Structure de la page HTML

Librairies

La librairie Leaflet est une bibliothèque JavaScript qui permet de créer et d'interagir avec des cartes interactives. Elle fournit des fonctionnalités avancées telles que des couches, des marqueurs, des lignes et des polygones, ainsi que des options de zoom et de contrôle de la carte sous forme de script.
Grâce à ça, nous avons pu accéder à des scripts (via des liens url) afin d'ajouter des éléments à notre carte comme la mini-map ou un bouton de localisation (*Figure 6*).

```
<script src="https://unpkg.com/leaflet@1.3.4/dist/leaflet.js"></script>
<script src="https://unpkg.com/leaflet-control-geocoder/dist/Control.Geocoder.js"></script>
<script type="text/javascript" src="ComCompostelle.js"></script>
<script type="text/javascript" src="CheminCompostelle.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet-minimap/3.6.1/Control.MiniMap.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/leaflet.locatecontrol@0.79.0/dist/L.Control.Locate.min.js" charset="utf-8"></script>
```

Figure 6 : Script leaflet

Couche de style CSS

Le CSS est un langage de style qui permet de modifier l'apparence des éléments d'une page Web. Il peut être utilisé pour modifier l'apparence des éléments de la carte Leaflet. Nous avons premièrement appelé des feuilles de style afin de personnaliser le logo d'outil de notre application. Secondement, au sein même de notre page html, nous avons créé des styles pour le titre de l'application, pour modifier l'emplacement de la carte et des outils ainsi que les couleurs de fonds. (*Figure 7*).

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.3.4/dist/leaflet.css" />
<link rel="stylesheet" href="https://unpkg.com/leaflet-control-geocoder/dist/Control.Geocoder.css" />
<link
    rel="stylesheet"
    href="https://cdnjs.cloudflare.com/ajax/libs/leaflet-minimap/3.6.1/Control.MiniMap.css"
/>
<link rel="stylesheet" href="style.css" />

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/leaflet.locatecontrol@0.79.0/dist/L.Control.Locate.min.css" />
```

Figure 7 : Style leaflet

Création de menu déroulant et de bouton

Dans le corps de la page html, nous avons construit des menus déroulant permettant de sélectionner les données regroupées dans les cinq tables de notre base de données. Ainsi, cinq menus déroulants ont été mis en place dans lesquels il est possible de choisir la valeur du champ "classe". Ainsi pour la catégorie "Où dormir ?", nous pouvons sélectionner la valeur "hotel" qui à travers une fonction **afficheDormir()** dans le javascript, réalise une requête SQL sur la table "dormir" et affiche la géométrie des objets. Si la valeur dans le menu déroulant est hotel, alors les objets représentés sont ceux pour lesquels la valeur du champ "classe" est "HOTEL". C'est donc une interaction constante qui se forme entre la page html, la page javascript et la base de données.

5.b) Interactivité via le langage Javascript

Afin d'apporter plus de clarté, le script javascript n'est pas directement dans le body de la page HTML. Nous avons créé un lien dans cette dernière qui appelle une page dans laquelle nous écrivons le code JS.

Eléments de base de la carte leaflet

- Flèche du nord avec **L.Control.Watermark** : Méthode qui permet d'ajouter un filigrane, image transparente superposée à la carte Leaflet.
- Échelle avec **L.control.scale** : Méthode permettant d'ajouter une échelle à une carte Leaflet. L'échelle est affichée sous la forme d'un curseur qui permet aux utilisateurs de connaître la distance entre deux points sur la carte.
- Zoom avec **L.control.zoom** : Méthode qui permet d'ajuster le niveau de zoom d'une carte Leaflet. Il peut être utilisé pour les zooms avant et arrière, et permet aux utilisateurs de voir plus de détails sur la carte, ou de voir une vue plus large.
- Mini-map avec **L.Control.MiniMap** : Méthode qui permet d'ajouter une carte miniaturisée à une carte Leaflet. Il s'agit d'une carte réduite qui affiche une vue plus petite de la carte principale, ce qui peut être utile pour orienter les utilisateurs.
- Base Maps : Couches servant de fond pour une carte Leaflet. Elles fournissent des informations géographiques de base, telles que les limites des pays, les rivières et les villes.

Overlays

Les Overlays sont des couches qui peuvent être ajoutées à une carte Leaflet pour fournir des informations supplémentaires. Ils peuvent être des couches vectorielles ou des couches de données raster et sont sélectionnables depuis la carte.
Les ajouts de données au format Geojson avec **L.geoJSON,** qui est une méthode qui permet de créer des couches vectorielles à partir de données géographiques au format GeoJSON pouvant être ajoutée à une carte Leaflet pour afficher des informations géographiques telles que des marqueurs, des lignes et des polygones. Nous avons ajouté à notre carte (*Figure 8*) :

- Chemin
- Communes traversées par le chemin: avec **bindTooltip** qui permet d'ajouter des infobulles affichant les noms des communes lorsque l'on passe le curseur dessus.
- Villes étapes avec un marqueur (**L.markers**) spécifique (**L.icon**) et une fenêtre pop-up (**BindPopup**) lorsque l'on clique dessus. Le pop-up affiche un titre, une image, et une description avec un lien internet.

Ces overlays sont sélectionnables et dé-sélectionnables dans le menu de la carte.
Par ailleurs, il est a précisé que le chemin et les communes sont des données geojson qui ne sont pas intégrées à la base de données. Ceux-ci se situent directement dans le répertoire où il est présent le script html, css, javascript.

Figure 8 : Overlays

Récupérations des points d'intérêt via le script php

Afin de fournir les informations nécessaires au marcheur, nous avons regroupé sous forme de marqueurs, les points d'intérêt regroupé en 5 classes :

- Pour dormir : Hôtels/Campings.
- Pour s'alimenter : Restaurants/Boulangeries.
- à visiter: Musées/ Châteaux/ Points de vue.
- Pour le tampon de la crédencial (passeport du pèlerin) : Mairies/Offices du tourisme.
- Utile sur le chemin : Fontaines/picnics

Figure 9 : Menu déroulant

Ces marqueurs ne sont donc pas sélectionnables via un contrôleur des couches directement présentes dans la carte leaflet, mais plutôt via un menu déroulant extérieur à la carte comme il a été expliqué précédemment.

Technologies secondaires

- Localisation GPS avec **L.control.locate** : Méthode qui permet d'ajouter un contrôle à une carte Leaflet pour localiser l'utilisateur (marcheur). Lorsqu'il est activé, le contrôle affiche l'emplacement de l'utilisateur sur la carte, et permet à l'utilisateur de naviguer sur la carte en fonction de son emplacement.
- Recentrage au centre de la carte avec un bouton retour.
- Coordonnée latlng qui s’affiche avec un clic sur la carte.

6. Pistes d’évolution

Suite à ce premier travail en webmapping, nous souhaiterions améliorer les fonctionnalités et les technologies utilisées durant ce projet.
Premièrement, il serait intéressant de pouvoir sélectionner dans un buffer, les points d’intérêts qui l’intersectent. Cette zone tampon prend forme à partir d’un point cliqué sur la carte. La taille du tampon est renseignée via un “numeric input”.
Deuxièmement, nous pourrions, à partir de la localisation de l'utilisateur et un point cliqué sur la carte, visualiser la distance en km à parcourir à vol d’oiseau.
Enfin concernant les points d’intérêts, nous pourrions modifier l’image des marqueurs en fonction de ce qu’il représente et réaliser un regroupement en cluster pour améliorer la clarté de la carte Leaflet.

7. Bibliographie

7.a) Références

https://geobgu.xyz/web-mapping/spatial-queries-1.html
https://blog.hubspot.fr/website/inserer-image-html
https://geoservices.ign.fr/
Support TD/Cours Webmapping

7.b) Données utilisées

https://www.openstreetmap.org/#map=6/46.449/2.210
https://geoservices.ign.fr/

Visualisation du projet Webmapping: