Documentation Technique

WEBBQ-EUROPA V4.0

Points Technique

Écully, le 17 Juin 2020

Partie 1. Création de base de donnés

1.1 Création initiale de donnés

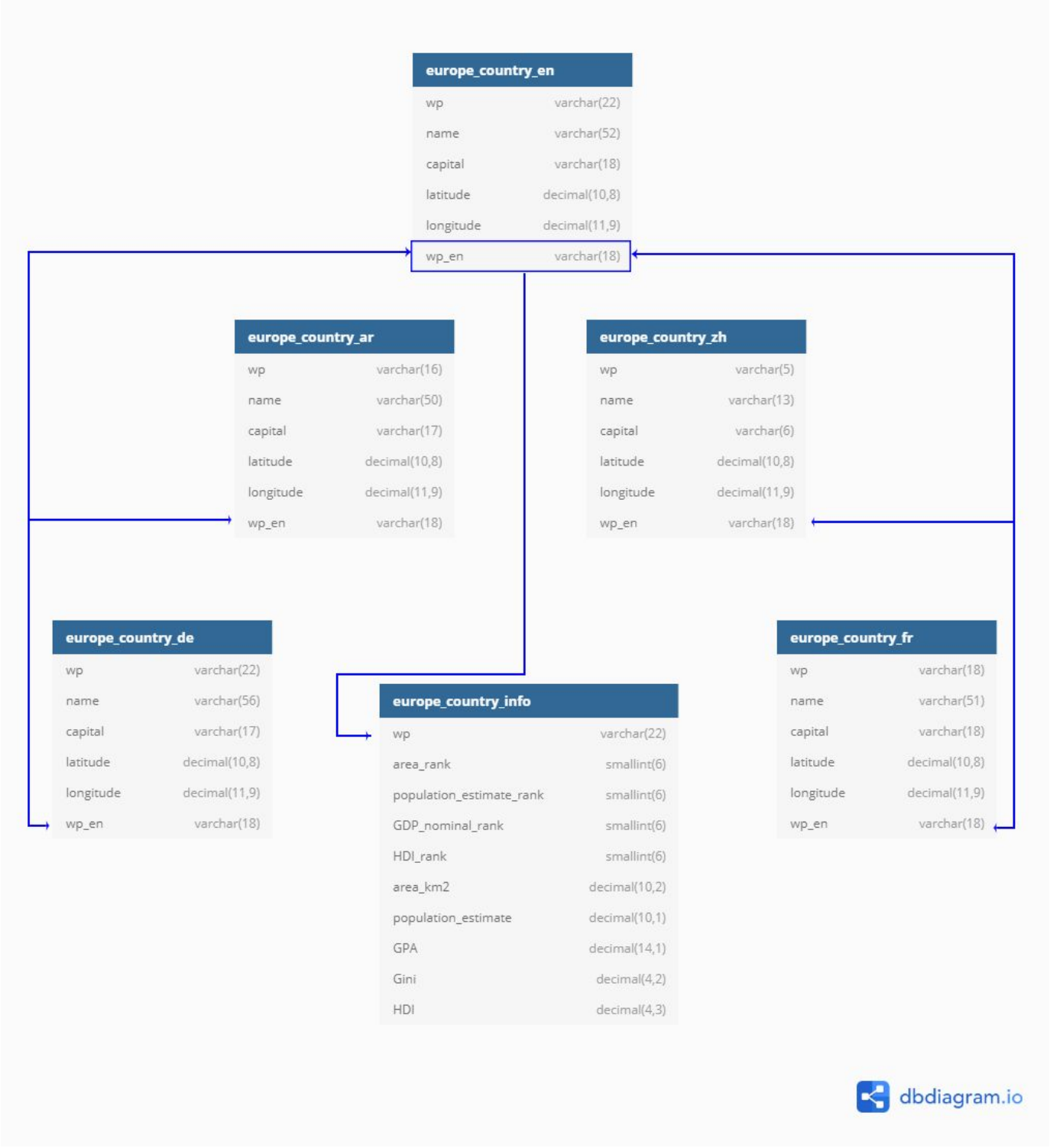


Figure 1.1. Schéma de base de donnés

Nous voulons bien créer une base de donnés avec les infos basiques(nom officiel, capital, latitude, longitude) ainsi que des infos supplémentaires(valeurs et classements mondiaux de surface, population, Gini, IDH et PIB) de chaque pays. En effet, vu que les infos supplémentaires sont les chiffres qui ne nous demandent pas de traduction, nous les mettons plutôt dans une nouvelle sous-base pour réduire les travaux superflus. Un autre aspects est que ces chiffres sont variables par rapport aux infos basiques, cette séparation nous permet donc des actualisations annuelles via Wikipédia au lieu de donnés fixes en format de .zip comme les infos basiques.

Le « Database_creator.py » et « dbinfo_creator.py » nous permet alors de créer initialement les donnés. Précisément, « Database_creator.py » enregistre les infos basiques à partir des donnés en format de .zip et les convertit vers une sous-base de donnés; « dbinfo_creator.py » cherche les infos supplémentaires plutôt via Wikipédia. Pour associer deux sous-base de donnés, nous ajoutons une colonne de « wp_en » dans ce premier, correspondante aux mots clés de ce dernier.

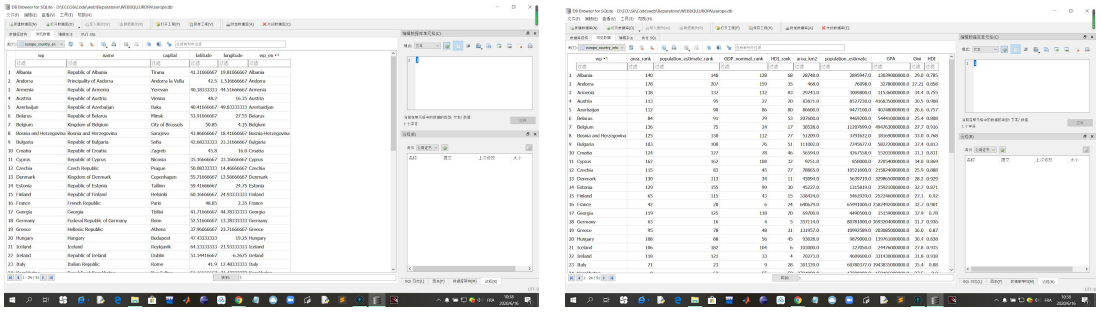


Figure 1.2. Exemple de base de donnés respectivement pour les infos basiques(Gauche) et supplémentaires (Droite)

1.2. Modification de donnés

En revanche, nous remarquons que le Wikipédia manque beaucoup de donnés sur certains pays comme Vatican. Il existe deux possibilité : soit cette pays ne donne pas leurs donnés aux associations mondiales, soit tout simplement Wikipédia n’enregistre pas ces donnés.

Pour le premier cas, nous trouvons de nouveau des listes récentes affichées par les associations et complétons ces cellules initialement vides. Pour le deuxième cas, au lieu d’utiliser des symbols comme NaN qui n’est pas connaissable par la base de donnés, nous préférons utiliser plutôt une valeur nonsense comme « -1 » pour distinguer ces cas. L’intérêt de cette notation est de les adapter aux contraintes réelles pour ces colonnes de chiffres.

1.3. Traduction de donnés

Nous nous intéressons aussi à traduire notre base de donnés basique aux langues diverses. Vu que la base de donné ne connaît que le « . », au lieu de convertir les chiffres aux habitudes écrites ici, nous préférons laisser ce traitement au javascript (voir Partie 4).

Partie 2. Échange avec les clients

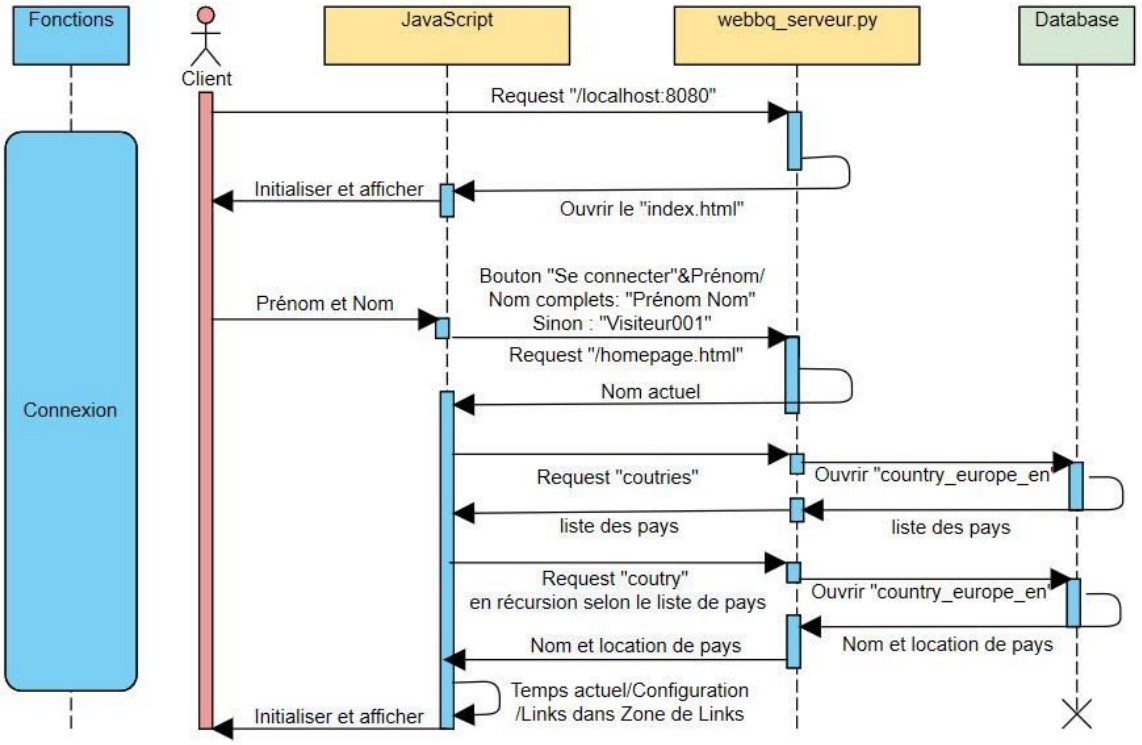


Figure 2.1. Diagramme UML d'une connexion

2.1. Façon de Connexion

Après avoir lancé le serveur et avoir typé « localhost:8080 » dans la barre de navigateur, nous accédons à la page de connection. Nous proposons deux stratégies de connexion : le visiteur peut choisir soit « Continuer sans connexion » sans aucun enregistrement comme un visiteur anonyme « Visiteur 001 », soit « Se connecter » en tapant toutes ses infos. Nous ne pouvons pas éviter les mauvaises entrées de clients. Nous précisions alors que dans le cas où les infos ne sont pas complètes, il sera traité aussi comme « Visiteur 001 ».

La transmission de ces valeurs, c-à-d, le nom et le prénom, nous les enregistrons dans une variable dans notre serveur et les envoyons à la nouvelle page par une requête GET.

Pour la changement de langue, c'est le même fonctionnement.

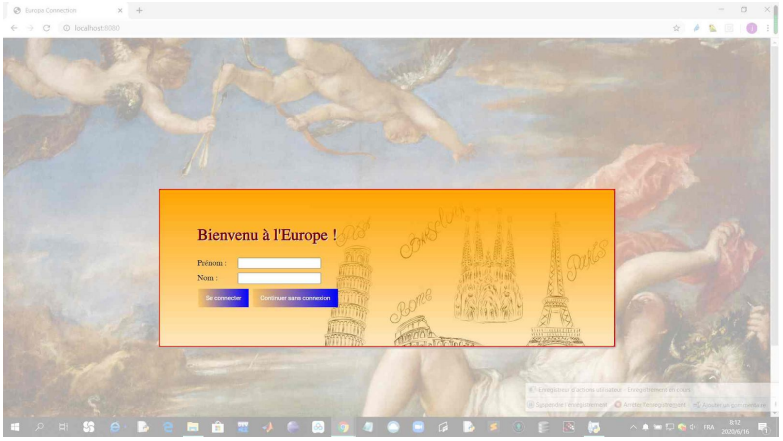


Figure 2.2. Page de connexion

2.2. Scène de passage

Pour éviter parfois l'actualisation lente à cause d'Internet de part de clientèles, nous ajoutons une scène animée pour rendre le passage plus acceptable.

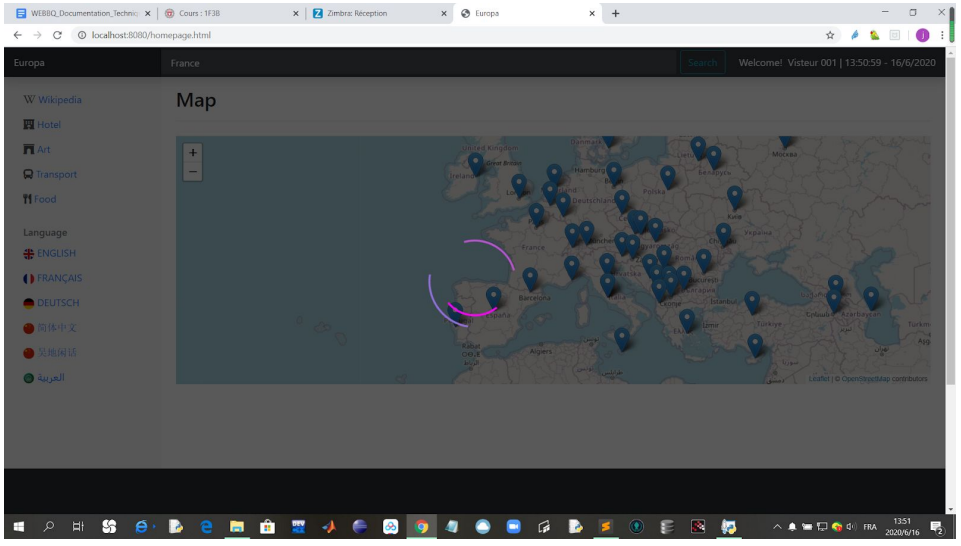


Figure 2.3. Page de passage

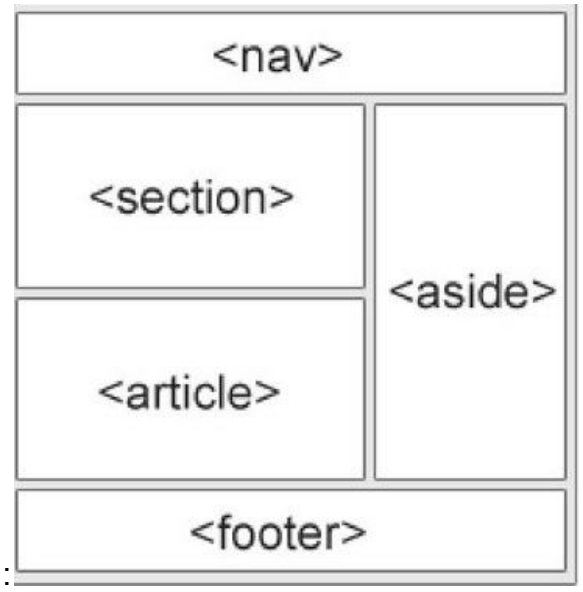
Partie 3. Choix d'architecture

Nous avons choisis une architecture client développée par HTML/CSS et Javascript qui est basé sur :

- Une page **index.html** qui apparaît lors de la première connexion au serveur, et qui est développé par HTML/CSS et un petit script JS pour pouvoir communiquer avec le serveur python.
- Une page **homepage.html**, c'est la page qui apparaît après avoir passer la page d'accueil, cette page contient du code HTML et du code Javascript (à la fin du code) et fait appel à **style.css** : un fichier css ou on a enregistré tout le code css nécessaire,
  leaflet.js/leaflet.css : les fichiers JS/CSS essentiels pour le fonctionnement de la map
  flags.js : Un code JS qui permet d'avoir un code du pays à partir de son nom (ce code est ensuite utilisé pour récupérer les drapeaux des pays).
  Code JS fontawesome : pour des icônes vectorielles et des logos dans la page.

- Pour rendre notre Projet Web plus compatible avec toutes les screens, nous avons utilisé la bibliothèque **Bootstrap**.

La page **homepage.html** est d'architecture **(nav/section/article/aside/footer)** représenté dans la figure :



:

Figure 3.1. architecture de la page

- **navbar : <nav></nav> :** contient
  - un titre <a>Europa
  - un input ou on peut faire la recherche <input> suivi d'un <datalist></datalist> pour fournir une fonctionnalité de "saisie semi-automatique" pour l'élément <input>
  - un bouton pour envoyer une requête GET au serveur

- un pour afficher le nom du visiteur et la date d'aujourd'hui.

- **aside : <nav id="sidebarMenu"></nav> :** contient
  - une première liste qui contient les liens (Wikipedia, Hotel ...)
  - une deuxième liste qui contient les langues (English, Français, Deutsch...)
- **section : ** : c'est l'élément ou la map est chargé.
- **article : :** qui contient :
  - une table <table></table> contenant une seul <tr></tr> (qui représente une seul ligne)
  - dans cet élément <tr> on retrouve un <td> (colonne) pour le drapeau, un <td> pour les informations du pays et un <td> pour le diagramme de l'adar du pays.

- **footer :** contient :
  - un contenant les nouvelles actuelles.
  - un deuxième contenant les membres du groupe.
  - un troisième ou on fait apparaître des informations de contact

Partie 4. Service fundamental

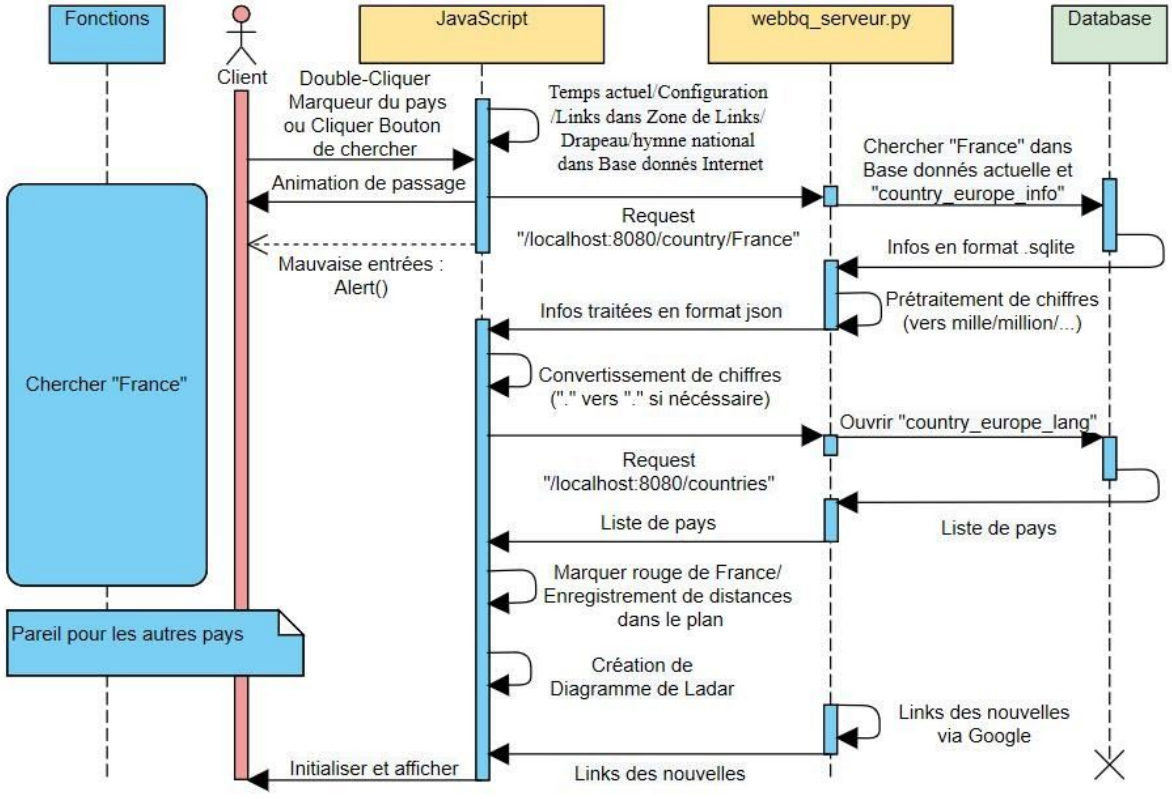


Figure 4.1. Diagramme UML d'une recherche

4.1. Zone de plan

La zone de plan, supporté par le javascript *leaflet.js*, affiche le plan mondial en marquant tous les pays européens. Les clients peuvent chercher les pays en double-cliquant le marqueur correspondant. Une fois que l'on entre dans le page de pays, le pays est marqué par un marqueur rouge et on peut trouver la distance entre une autre pays et ce pays en cliquant sur ce premier.

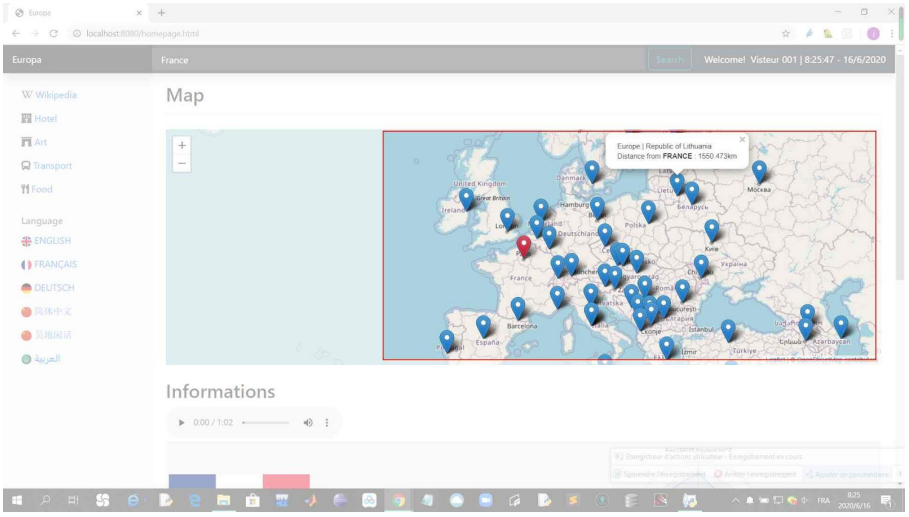


Figure 4.2. Zone de plan

4.2. Zone de recherche

La zone de recherche en haut permet aux clients de chercher le pays en typant le nom. Ici nous fournissons aussi une soutien de recherche vague avec le liste de pays en langue exigée actualisé lors d'initialisation de plan.

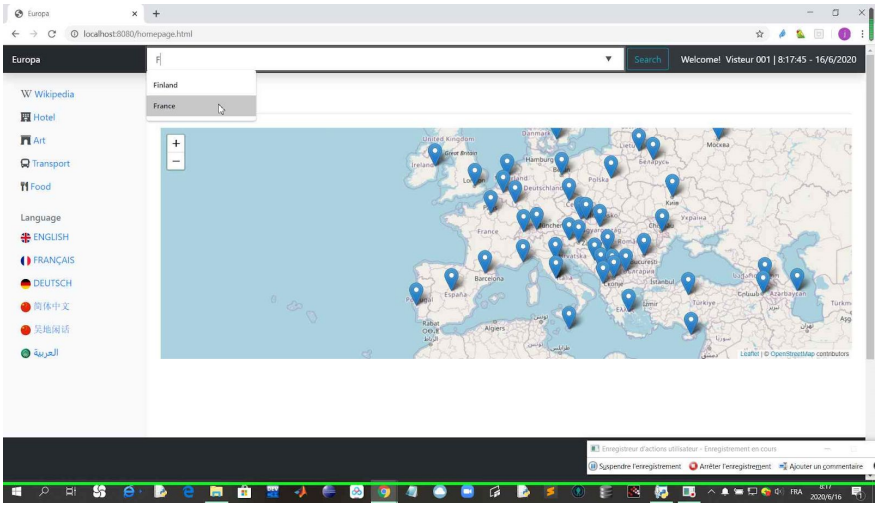


Figure 4.3. Zone de recherche

4.3. Zone de links

La zone de links permet de visiter des pages associés au pays que les clients sont en train de regarder, comme Wikipédia, Airbnb, Google Culture, Kiwi, Michelin Guide en langue choisie. Initialement, toutes les links est accrochés sur le mot clé « Europe ».

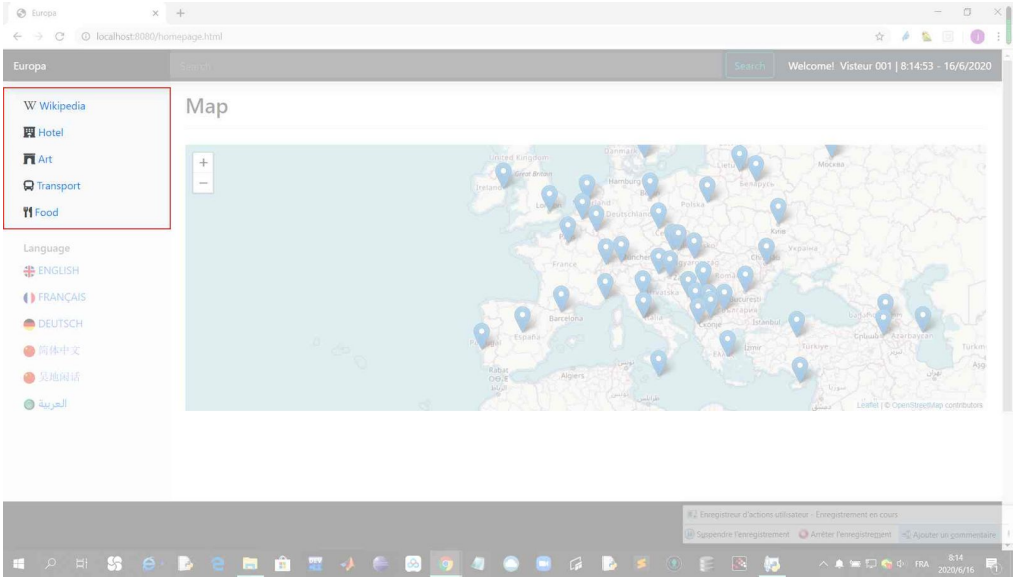


Figure 4.4. Zone de links

4.4. Zone d’informations

La zone d'informations inclut le hymne national et le drapeau de ce pays via 2 bases sur Internet respectivement *https://anthemworld.com/~anthemwo/themesongs* et *https://lipis.github.io/flag-icon-css/flags/4x3/gb.svg*, les infos basiques et ainsi un diagramme de L'adar qui permet aux clients de mieux connaître ce pays par les donnés supplémentaires.

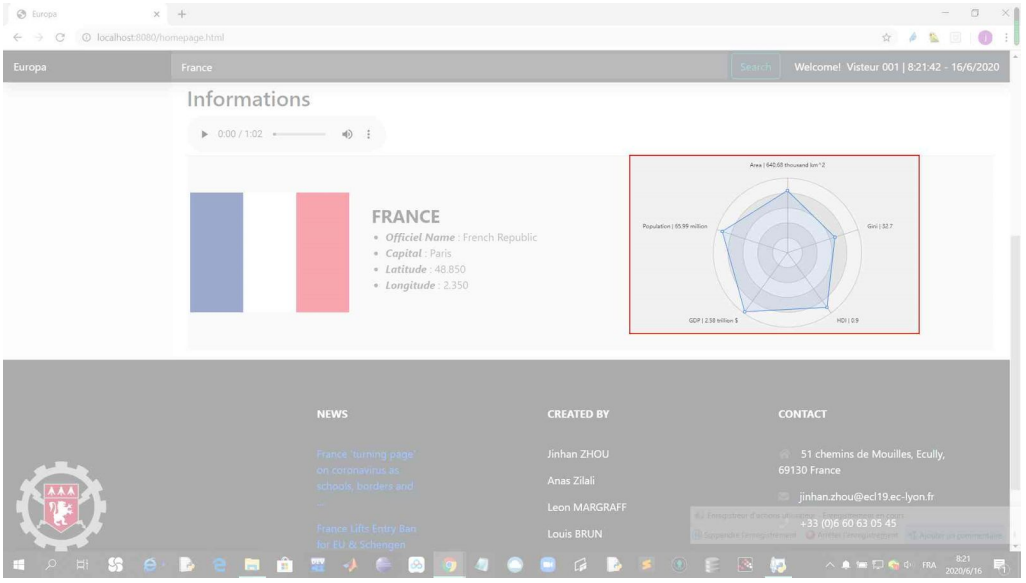


Figure 4.5. Zone d'informations

4.5. Sous-zone de nouvelles dans le footer

La sous-zone de nouvelles affiche initialement les infos de notre équipe(membres et façon de contraction). Initialement, il n'apparaît pas dans le footer. Une fois que les clients cherchent un pays, elle affichera ainsi les nouvelles actuelles de ce pays(via Google).

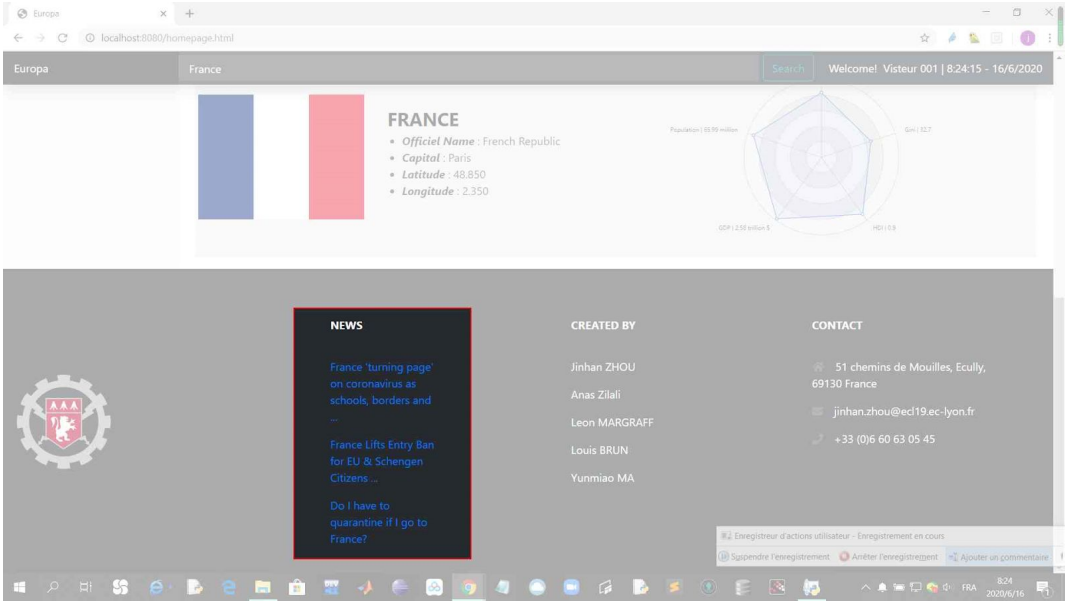


Figure 4.6. Sous-zone de nouvelles dans le footer

4.6. Affichage de temps

Pour l'horloge, nous obtenons le temps actuel lors d'initialisation et le refresher par une récursion « setTimeout(getTime,1000) ».

Partie 5. Respecte de la culture

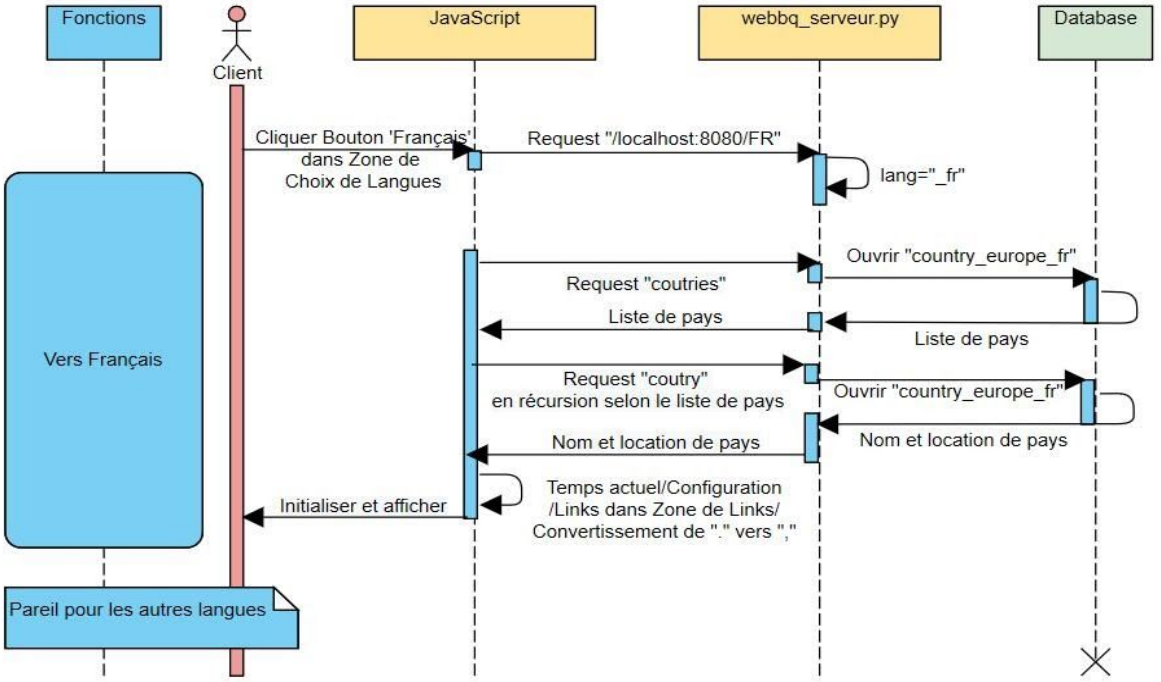


Figure 5.1. Diagramme UML d'un changement de langues

5.1. Langue diverses

À l'aide de base de donnés en langues diverses, nous changeons de sous-base de donnés lors de changement de langues. Aussi, nous utilisons les variables pour les textes sur notre pays pour les traduire immédiatement lors d'actualisation. Ainsi, les zones de links changes de langues en modifiant les affixes de links.

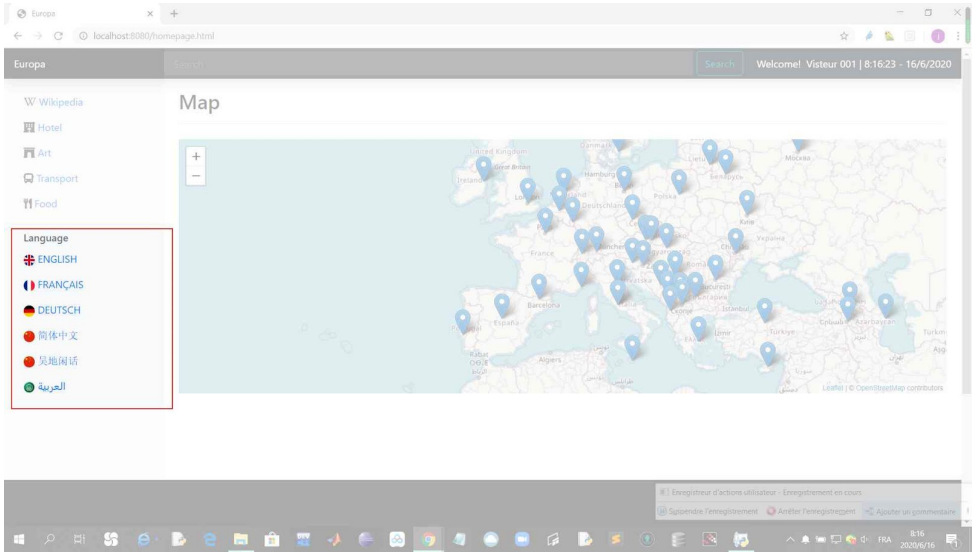


Figure 5.2. Zone de choix de langues

5.2. Page en français et en allemand

Nous remarquons que pour les français et les allemands, ils préfèrent d'utiliser « , » au lieu de « . ». Nous laissons le javascript à les convertir lors d'affichage.

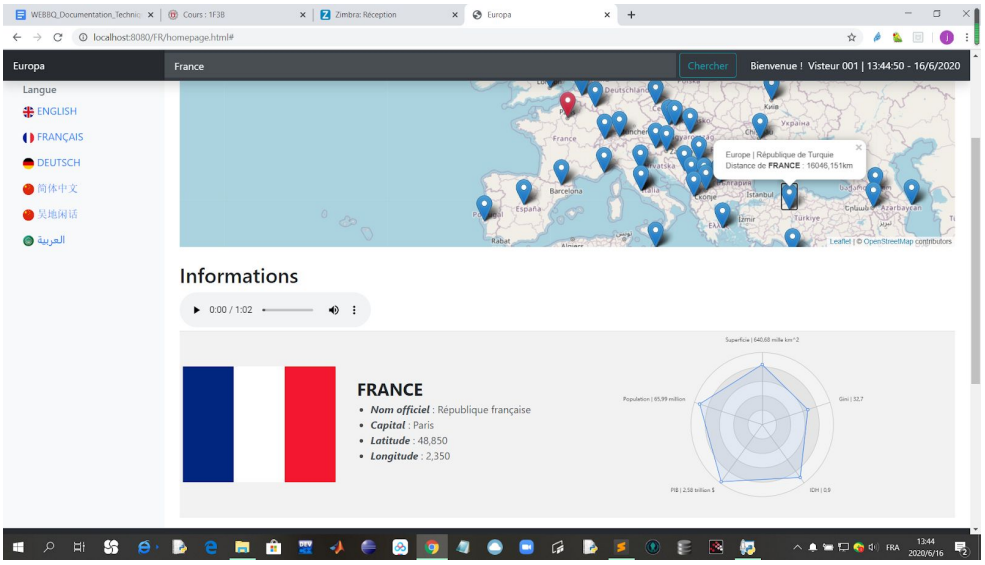


Figure 5.3. Page en français

5.3. Page en chinois et en dialecte Wu

Nous pré-traitons les chiffre en forme de "mille/million/billion/trillion" avant d'afficher. Nous remarquons que les chinois utilisent plutôt « mille(千)/dix mille(万)

/cent million(亿)/trillion(兆) » et alors nous ajoutons cette détaillés dans la fonction de pré-traitement dans le Python.

Ainsi, la langue Wu est une dialecte chinoise et alors il peut partager les donnés basiques avec celle de la langue chinoise. Pour les links, c'est pareil.

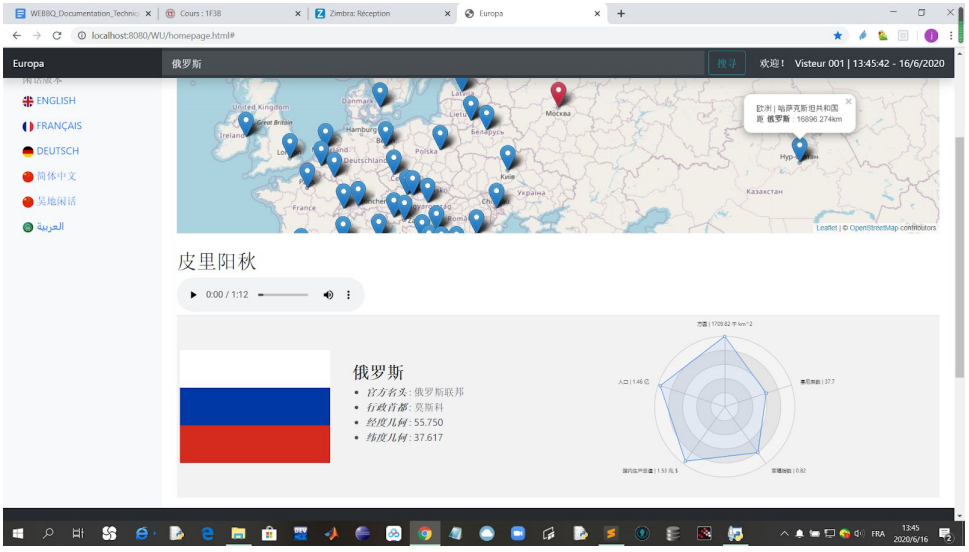


Figure 5.4. Page en Wu

5.4. Page en arabe

Nous remarquons que les arabe lisent la page de droite vers gauche et alors nous configurons le page « document.documentElement.style.direction="ltr/rtl" ».

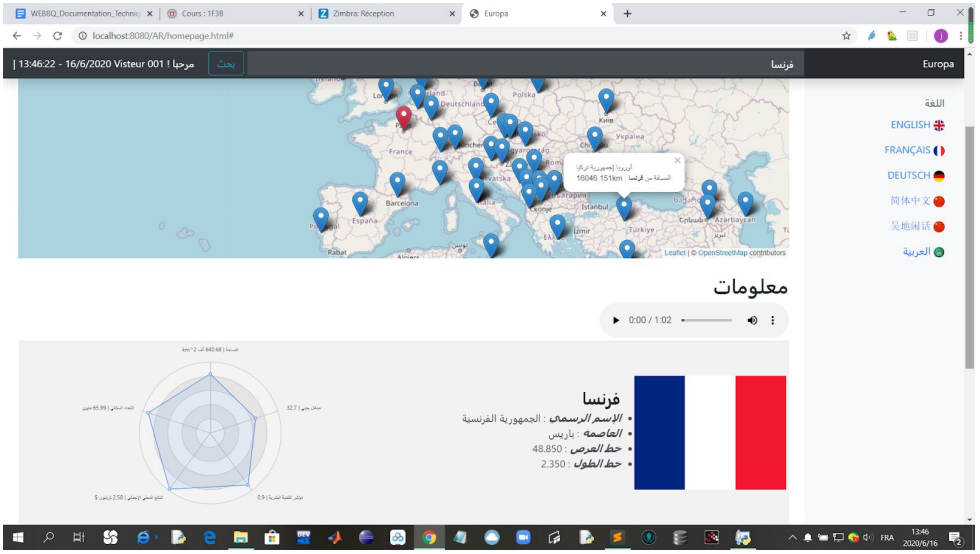


Figure 5.5. Page en arabe

Partie 6. Phénomènes possibles normals et leurs solutions

À cause de service forcément effectué par le navigateur, nous remarquons quelques phénomènes possibles. Ce n'est pas une erreur techniques de notre part et nous vous donnons aussi de solutions pour les résoudre.

Pour le passage de pages, surtout entre la page de connexion et la homepage et entre les pages en différente langues, il apparaît parfois des requests non effectué. C'est causé par le remplissement de donnés automatique de navigateur à partir de mémoire qui annule la request. Pour le résoudre, il vous suffit de faire une refresh et la request serait effectué de cette réinitialisation.

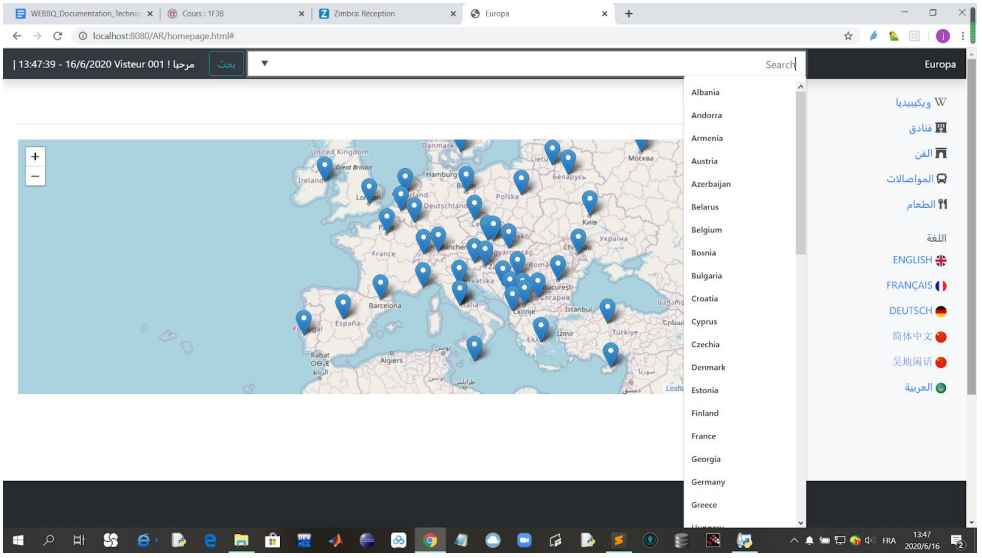


Figure 6.1. Phénomène 1 - Réinitialisation incomplète

Ainsi, les links de nouvelles parfois vous orientent à une page de 405, c'est à cause de la protection légal de certains pays et ne vous inquiétez pas alors à ce phénomène normal.

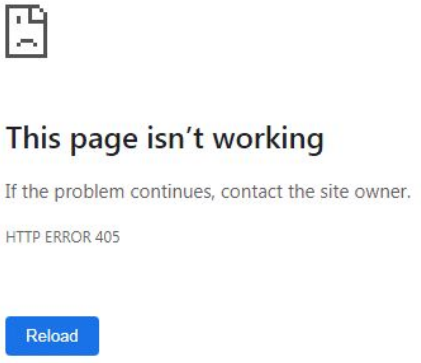


Figure 6.2. Phénomène 2 - Links de nouvelles vers 405

Il y a encore un détaille à préciser, à cause des locations suffisamment proche de Rome et Vatican, le marqueur de Vatican couvre une grande partie du marqueur d'Italie. Mais si vous zoomez le plan, vous pouvez aussi distinguer ces deux pays.

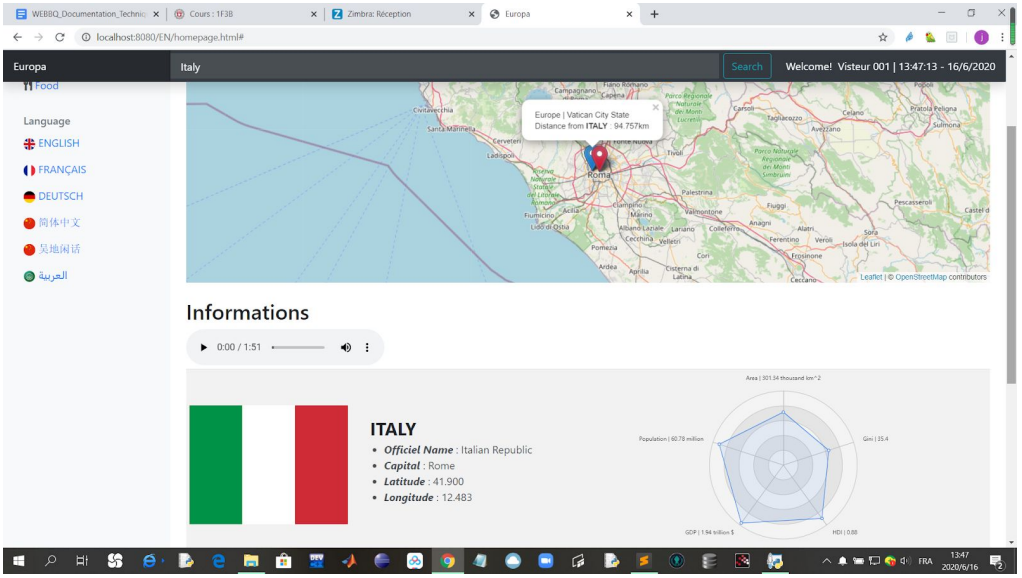


Figure 6.3. Phénomène 3 - Vatican et Italie

Partie 7. Remerciement

À la fin de cette documentation, nous remercions à tout le monde qui contribue à notre projet.

Nous remercions les professeurs de l'École Centrale de Lyon, qui nous ont offrir les cours de base ainsi que leurs conseils utiles. Nous remercions aussi les créateur des ressources secondairement-développable communes qui nous permet de réaliser notre conception plus efficacement.

Nous remercions chaque membre de notre équipe.

Nous remercions Jinhan ZHOU et Anas ZILALI, concepteur principaux de notre projet.

Nous remercions à Leon MARGRAFF, Louis Brun et Yunmiao MA, établisseur de base de donnés en langues diverses. Même que le mode que nous choisissons pour le département de base de donnés (c-à-d, échanger par Messenger en format .csv et commettre par le chef sur notre Github pour clarifier notre répertoire) n'affiche pas toute leur contribution sur le timeline de notre projet, nous vous prions vraiment de vous souvenir leurs efforts derrière alors lors d'utilisation.

Merci de votre écoute !

Jinhan ZHOU Anas ZILALI

Leon Alwin MARGRAFF

Louis BRUN Yunmiao MA

Équipe de WEBBQ-EUROPA

(Groupe C-B1a / Ecole Centrale de Lyon)