INF tc3 : Projet d'application web

B1a - Groupe D - Amérique du Nord

Documentation technique

Enseignant : MULLER Daniel

Élèves :
BRUNEAU Marin
DENISART Bastien
HAZIZA Lucie
PORTE Tom

I. Présentation générale de l'application

Cette application web a pour but de faire découvrir à l'utilisateur la région de l'Amérique du Nord, le tout au travers d'une carte du monde qui donne accès à des fiches d'informations. Il est donc possible d'interagir avec des POI (Point Of Interest) pour ouvrir ces fiches (contenant également le drapeau du pays), ou encore de cliquer sur le nom du continent pour accéder directement à sa page wikipédia.

L'interface a avant tout été pensée pour être agréable à l'oeil (les couleurs de l'arrière plan rappellent par exemple celles des Etats Unis et de la France), mais également simple d'utilisation et intuitive.

II. Schéma de la base de données

La base de données créée regroupe différentes informations sur les pays. Certaines sont nécessaires à leur identification (nom, capitale, drapeau, etc.) et d'autres sont des informations sur le pays qui ne sont qu'utiles ou complémentaires aux informations principales (population ou nom du dirigeant). Le nom de cette table est *countries* et possède une clé primaire (*wp*), même si on n'effectue pas ici de requête complexe dans le sens où on aurait besoin de croiser des tables. Toutefois, elle sera employée par le serveur pour sélectionner un pays dans cette base de données.

La base de données à été crée à partir des fichiers .json fournis et d'un programme python qui permet d'interpréter l'allure de ces données pour les formaliser dans la base de données. On trouvera ce programme dans le dossier du serveur.

Certains cas particuliers ont été distingués car les données n'étaient pas agencée de la même manière (notamment pour les Etats-Unis, qui ont des unités différentes pour les surfaces par exemple).

III. Fonctionnement du serveur

La principale fonction utilisée dans le serveur est la fonction *DoGET*, qui permet de répondre aux requêtes du client. Cette fonction est constituée de deux blocs.

Le premier permet d'initialiser les paramètres utilisés dans la suite de la fonction. C'est principalement l'analyse de la requête reçue du client, et plus précisément de la première ligne de son entête, qui permet de récupérer ces paramètres. En effet, le client peut envoyer deux types de requêtes via l'appel GET:

- GET / location
- GET / description/idnum

Il faut donc que le serveur puisse effectuer le routage. Pour ce faire, le serveur examine la chaîne de caractère de la requête et récupère sous forme de liste les éléments de cette chaîne. En particulier, le type de demande faite par le client (localisation d'un pays sur la carte ou description détaillée de ce pays) et le numéro de pays concerné (en cas de demande de description) sont stockés dans la variable **path_info** respectivement au rang 0 et 1.

Le deuxième bloc consiste à traiter correctement la demande, ie à effectuer le routage, récupérer les informations dans la base de données (bdd) et à envoyer la réponse au client sous format JSON. Le routage est effectué grâce à la variable **path_info**, suivant la valeur de son premier élément.

Dans le cas d'une demande de localisation, on initialise une variable **data** comme une liste, amenée à contenir des dictionnaires. Chaque dictionnaire sera associé à un pays de la bdd et comportera les 4 clés suivantes : *id* (identifiant du pays dans la bdd), *lat* et *long* (respectivement la latitude et la longitude renseignées dans la base de données) et *name*, qui est le nom du pays en question. En pratique, le serveur récupère l'intégralité de la base de données (par la fonction *db_get_coutries*) qu'il enregistre sous la forme de dictionnaires. Puis, pour chaque pays, le serveur ne sélectionne dans ces dictionnaires que les couples clé-valeur nécessaires. **data** est donc à ce stade une liste de 23 dictionnaires (puisque 23 pays ont été renseignés dans la bdd) contenant chacun les 4 clés précisées plus tôt. Il ne reste plus alors qu'à utiliser la fonction *send_json* pour encoder cette liste au format json et l'envoyer au client.

Dans le cas d'une demande de description, on initialise la variable **data** qui va contenir la base de données. Ensuite on parcourt **data** pour trouver l'identifiant du pays que l'on souhaite décrire. Une fois celui ci trouvé, (on rappelle que ce pays se présentera sous la forme d'un dictionnaire) on parcourt les clés de ce dictionnaire (et on les enregistre dans une variable '**donnee**') pour récupérer les informations qu'elles contiennent. Ces informations sont par ailleurs enregistrée dans la variable **donnee**). Ces clés sont : l'identifiant, le nom du pays, sa capitale, ses coordonnés (longitude et latitude), son drapeau, le nom du leader, la surface et la densité de population. Une fois que toutes les informations sont stockés dans la variable 'donnée', on les envoies sous format Json au client.

On retrouve aussi dans ce code toute la structure d'un serveur classique comme par exemple les méthodes pour analyser et traiter les requêtes HEAD et POST. La fonction db_get_countries déjà évoquée plus tôt permet de se connecter à la base de données et, grâce à une requête SQL, récupère les informations des pays initialement stockées dans la base de données.

IV. Description de la logique du client

Le rôle du client est d'assurer l'interaction entre l'utilisateur et le serveur. Ainsi, le client va réceptionner les demandes de l'utilisateur (clique de la souris sur des POI principalement), effectuer une requête au serveur (GET localisation ou GET description d'un pays, voir section III), réceptionner la réponse du serveur et la transmettre à l'utilisateur.

Le client proposé ici prend la forme d'un fichier html associé à une feuille de style CSS et un document leaflet. Ce dernier document permet d'afficher une carte glissante sur l'application web. La feuille de style CSS permet quant à elle de personnaliser l'affichage et les propriétés graphiques de la page.

Le fichier html est le coeur du client et ses premières commandes consistent à faire le lien avec les deux documents cités précédemment. Ce fichier html (appelé "le client" dans la suite) se décompose, lui aussi, en deux blocs distincts : un bloc assurant l'affichage et l'organisation de la page web et un bloc de script permettant une interaction dynamique avec le serveur.

- Organisation de la page : il s'agit ici d'un code HTML 'pure'.

Au chargement de la page, il est nécessaire que tous les POI soient automatiquement placés sur la carte. Pour cette raison, la fonction *load_data* est appelée au début du fichier, afin d'obtenir par le serveur les coordonnées des POI. Dans un second temps, le client organise la page : un titre en haut, une zone A pour afficher la carte, une zone B pour afficher la description d'un pays choisi sur la carte et un bas de page ("footer") pour afficher du texte complémentaire.

La zone A fait appel à la variable **map** pour gérer la carte et la zone B aux textes HTML **description** et **flag** générés grâce à une réponse du serveur: tous seront définis dans le second bloc du fichier html client. Ainsi, cette première partie du fichier HTML donne l'architecture globale de la page; le script permet de remplir cette structure avec du contenu dynamique.

- Script : ce bloc permet d'éditer des fonctions et d'effectuer des requêtes avec le serveur.

Afin que le bloc précédent soit fonctionnel, trois points doivent être développés : la variable **map**, la fonction *load_data* pour récupérer la position des POI sur la carte et les placer, et la fonction permettant de récupérer la description et le drapeau d'un pays.

- La variable map est créée au début du script et permet de positionner la vue de la carte sur un point donné, ainsi que de régler le niveau du zoom.

- Fonction *load_data* : cette fonction génère une requête de type GET localisation. La réponse renvoyée par le serveur au format JSON (voir section III), va être analysée par le client afin de manipuler une liste, dont chaque élément correspond à un pays. Finalement, pour chaque pays, le client va ajouter un marqueur à la position du pays récupérée par le client (les images de marqueur sont disponibles dans le dossier "Images"), lier un Popup à ce marqueur et associer une fonction à un clic sur le POI ainsi créé (cette fonction est *OnMarkerClick*). Cette association est la base de l'interactivité de l'application et c'est elle qui permettra de choisir un pays sur la carte

et d'afficher les informations qui lui correspondent. Enfin, un identifiant **idnum** est associé au pays, correspondant à la clé *wp* de la bdd.

A ce stade, la carte glissante est pleinement fonctionnelle. Elle apparaît correctement dans la zone A et dès le chargement de la page, des marqueurs se positionnent dessus à l'emplacement de la capitale de chaque pays. Par ailleurs, il est possible de cliquer sur ces POI et d'afficher un Popup sur la carte. La dernière étape consiste à remplir la zone B contenant les informations à propos d'un pays choisi sur la carte. C'est la fonction *OnMarkerClick* qui va rendre cela possible.

- Le fonctionnement de la fonction *OnMarkerClick* est le même que *load_data* : une requête est faite auprès du serveur, dont la réponse au format JSON sera analysée et adaptée pour manipuler une liste. La requête envoyée par le client est du type 'GET description' et la réponse du serveur adaptée par le client permet d'avoir accès à toutes les caractéristiques du pays sélectionné. A partir de ces caractéristiques, la fonction va générer deux chaînes de caractères au format HTML : **description** et **flag**. Ces deux chaînes sont intégrées au premier bloc du fichier HTML, de sorte à ne former finalement qu'une seule commande HTML, dépendant du POI sur lequel le clic a été fait et permettant d'afficher dans la zone B la fiche du pays et son drapeau. Par ailleurs, le client ne reçoit que le nom du drapeau, qui a été préalablement enregistré dans la bdd. Le rôle de **flag** est donc d'aller récupérer l'image correspondant dans le bon dossier et de l'afficher comme image dans la zone B.

  Remarque : il est crucial que la fonction *OnMarkerClick* sache quel pays a été sélectionné afin d'effectuer la bonne requête auprès du serveur. Ceci est rendu possible grâce à **idnum**, puisque la requête envoyée par la fonction *OnMarkerClick* comporte après le mot-clé description l'identifiant du pays. Grâce à cela, le serveur peut sélectionner dans la bdd le bon pays (puisque **idnum** est la clé primaire de la base).

En résumé, le fichier html du client est composé d'un premier bloc de texte, permettant de créer l'architecture générale de la page, et d'un bloc de script, comportant principalement deux fonctions. Ces deux fonctions suivent le même schéma : envoie d'une requête XMLHTTP au serveur, réception de la réponse au format JSON et traitement de cette réponse. La première fonction est appelée dès le chargement de la page et permet d'afficher les POI sur la carte glissante ainsi que d'associer une seconde fonction à un événement 'clic' sur ce POI. Cette seconde fonction (qui est donc appelée après un clic sur un POI) récupère les informations concernant le pays associé au POI et permet leur affichage dans une zone dédiée, définie dans le premier bloc.

V. Installation de l'application

L'installation de cette application se fait en plusieurs étapes.
Dans un premier temps, il s'agit de récupérer l'intégralité du répertoire github.
Dans un second temps, il faut lancer le serveur en exécutant le fichier *serveur.py*. Attention à bien l'exécuter dans une nouvelle console afin que le port utilisé (8080) soit bien libre.
Une fois le serveur lancé, on peut ouvrir dans la fenêtre de navigateur l'application web correspondant au client (fichier HTML) nommé *client_carte*. Le lien suivant peut être utilisé :

http://localhost:8080/client_carte.html

Il faudra toujours s'assurer que le fichier client est bien stocké avec : la feuille CSS *style*, les documents *leaflet*, le dossier *flags* (contenant les images de drapeau) et le dossier *images* (contenant les images de marqueurs). De plus, il faudra que le serveur (fichier *serveur.py*) et la base de données (*pays.sqlite*) restent dans le même dossier.