IUT Paris Descartes 2018-2019

Programmation Web Client Riche

Projet Logement sociaux

Maud GELLEE Groupe 201

Arsène LAPOSTOLET Groupe 202

Introduction

Introduction

Pour ce projet nous devions créer une application Web contenant une carte interactive, à la manière de google map. Pour ce faire nous devions utiliser le JavaScript et plus précisément la bibliothèque JQuery, ainsi que des Services distants, nous permettant de récupérer les informations géographiques telles que la longitude et la latitude.

Présentation de l'application

Fonctionnalités

Nous avons décidé de créer un site de sensibilisation et d'information sur le thème des logements sociaux. La carte est donc interactive et sert plusieurs fonctionnalités de l'application :

1) Indiquer les communes d'île de France n'ayant pas une proportion assez importante de logements sociaux par rapport au nombre d'habitant selon la loi solidarité et renouvellement urbains du 13 décembre 2000.

2) L'utilisateur peut aussi sélectionner n'importe quelle ville d'île de France sur la carte, ou en indiquant son nom dans la barre de recherche, afin d'obtenir certaines informations, comme le nombre d'habitants, ou le nombre de logements sociaux.

3) Après avoir sélectionné une ville l'utilisateur peut aussi avoir le temps de trajet ainsi que l'itinéraire entre le point sélectionné et la station Chatelet, afin d'avoir une idée de l'isolement de la ville.

En dessous de la carte, on peut voir une liste d'articles de lois concernant les logements sociaux en France ainsi qu'un forum permettant de déposer des propositions à propos de la gestion des logements sociaux, ou toute autres choses pouvant aider la communauté. Pour déposer un commentaire une personne doit être inscrite, et donc connectée à l'application, pour vérifier cela nous utilisons la base de données MySQL.

Provenance des données

Nous avons récupéré toutes les données des logements sociaux sur le site data.gouv.fr et en avons importées dans le projet dans le répertoire donné. Pour tout ce qui est données les API adresse.data.gouv.fr et navitia.io du STIF.

Bibliothèques et Outils

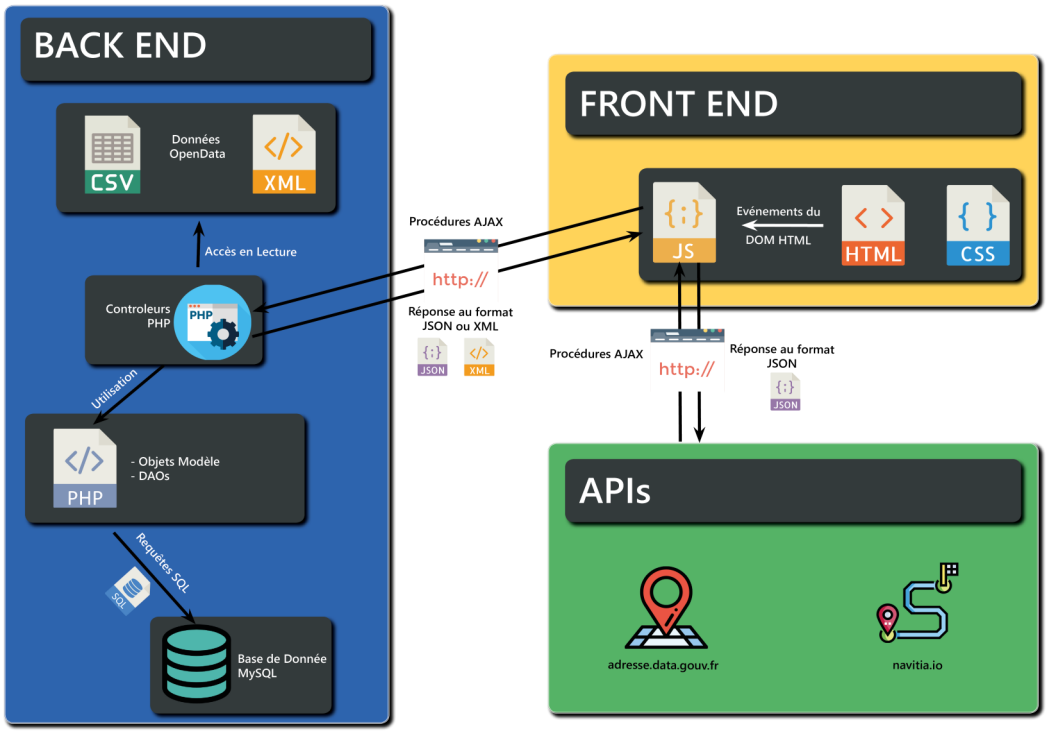
Nous avons utilisé le framework CSS Material Design Lite pour le style de l'interface.
Pour la carte nous avons utilisé Leaflet et jQuery pour le code client.

Architecture

Nous avons choisi une architecture micro-services, afin d'utiliser au maximum les procédures AJAX et d'avoir une Single Page Application n'imposant aucun temps de chargement à l'utilisateur.

Le code JavaScript est séparé en plusieurs fichiers et regroupés dans le dossier scripts afin de faciliter la lecture et la compréhension. Nous avons donc 5 fichiers javascript qui s'occupent de l'affichage et de la gestion de la carte, de l'affichage des lois, de la connexion et de l'inscription d'un utilisateur, de la responsivité du site et du scroll fluide de la page en utilisant le menu.

Nous avons fait de même avec les fichiers PHP, regroupés sous le répertoire ajax. Chaque fichier sert à récupérer des données stockées dans des documents du répertoire "donnees" et de les renvoyer au document javascript grâce à des *echo()* au format JSON, ou bien à manipuler les données stockées dans MySQL. Nous avons des fichiers contrôleurs qui font appels à des objets du modèle ainsi qu'à leurs DAO pour interagir avec la base de données MySQL.



Architecture de l'application

Services

La carte (carte.js)

Nous avons utilisé le bibliothèque Leaflet afin d'importer et de manipuler la carte le fond de carte que nous avons choisi est celui d'OpenStreetMap. Dès le chargement de la carte nous utilisons une procédure AJAX qui appelle une fonction PHP (deficitSRU.php) que nous avons développée permettant d'indiquer sur la carte les villes n'ayant pas le bon nombre de logements sociaux selon la loi. Ce code traite des données du fichier sru.csv contenant la liste des villes ne respectant pas la proportion de logement sociaux. Une fois toutes les villes récupérées nous comparons ces villes avec toutes les villes d'île de France, et affichons ensuite ces villes au format geoJSON, qui seront traitées dans le code javascript.

De la même manière (avec des procédures AJAX), si l'utilisateur clique sur un endroit de la carte, des informations à propos de cette ville seront affichées (nom de la ville, nombre d'habitants, nombre de logements sociaux). Pour ce faire nous avons utilisé l'API adresse.data.gouv.fr, qui nous permet en passant la longitude et la latitude du point de récupérer le nom de la ville, ainsi que les autres informations au format JSON, que nous avons juste à traiter et à afficher. Si un utilisateur préfère entrer le nom de la ville dans le champs prévu à cet effet, alors la recherche provoquera un clic sur la carte grâce à l'instruction fireEvent(), ce qui évite ainsi toute redondance de code.

Une fois une ville sélectionnée, l'utilisateur peut demander à voir l'itinéraire le plus rapide de la ville à Chatelet. Pour ce faire nous avons créé une fonction se déclenchant sur l'appui du bouton Itinéraire du centre. Afin de récupérer l'itinéraire nous utilisons l'API navitia.io, nous exécutons donc une procédure AJAX en lui passant les coordonnées de la ville cliquée et les coordonnées de Chatelet *(nous devons aussi fournir une clef d'API qui nous permet ainsi d'utiliser les services de l'API)*. La requête nous donne alors des informations aux formats JSON, dont le chemin, le temps de trajet et les différents transports à utiliser. Nous traitons ensuite ces informations et les convertissons au format geoJSON afin de les afficher sur la carte.

Enfin la fonction compteur(object) permet, lors de l'affichage d'un nombre, de ne pas le faire apparaitre d'un coup mais de faire monter le compteur progressivement de 0 au nombre souhaité de façon animée, pour donner plus de dynamisme à l'UI.

Les lois (LivreLois.js)

Dans ce fichier nous appelons une procédure AJAX qui exécute le code PHP du fichier lois.php qui nous permet simplement de récupérer des lois au formats xml. Nous traitons ensuite chacune des entrées dans le code JavaScript grâce à la fonction jQuery *find()* qui nous permet de récupérer des balises dans le DOM XML de la même façon que dans le DOM HTML. Nous créons ensuite du code HTML qui affiche le titre de la loi et son contenu si on le déroule, et en l'ajoutant dans notre page principale.

Connexion et inscription (connexion.js)

Lorsqu'un utilisateur cherche à se connecter ou à s'inscrire, une fenêtre modale apparait avec le formulaire correspondant, ce qui permet ainsi de ne pas changer de page, pour cela nous avons utilisé la méthode *showModal()* de Material Design Lite afin d'afficher une fenêtre modale.

Si un utilisateur cherche à se connecter, le fichier connectUtilisateur.php est appelé et vérifie si l'utilisateur existe bien dans la base de données en appelant UtilisateurDAO.php (ce fichier sert d'intermédiaire entre notre application et MySQL). Si l'utilisateur est connecté avec succès on fait disparaitre du menu les boutons connexions et inscription et rend disponible l'ajout de propositions.

De la même manière si un utilisateur cherche à s'inscrire, on vérifie que l'adresse mail fournie n'existe pas dans la base et si ce n'est pas le cas on inscrit l'utilisateur.

Dépôt de commentaires (propositions.js)

La dernière fonctionnalité que nous avons implémentée est le dépôt de commentaire, un utilisateur connecté peut, s'il le souhaite, déposer des propositions en rapport avec les logements sociaux (conseils sur les infrastructures, les démarches administrative, …). Pour cela il doit cliquer sur un bouton qui fera apparaitre dans une fenêtre modale, un formulaire de dépôt de commentaire. Une fois ce formulaire rempli, le contenu sera transmis à un service PHP (addProposition.php) qui ajoutera la proposition à la base de donnée MySQL.

Si l'utilisateur est connecté il pourra voir automatiquement les propositions déjà mises en ligne par les autres utilisateurs. Pour charger toutes les propositions nous faisons appel au service getAllPropositions.php par l'intermédiaire d'une procédure AJAX. Le fichier PHP nous rend, au format JSON, toutes les propositions que nous avons stockées dans la base, et la procédure ajoute les cartes de chaque proposition dans le code HTML.

Pour plus de clarté, nous avons réuni au sein de PorpositionsDAO.php tous le code manipulant la base de données et créé une classe proposition dans le fichier Proposition.php. Cette classe nous permet de manipuler plus simplement les données d'une proposition (auteur, titre et contenu).

Conclusion

Difficultés rencontrées

Nous n'avons pas rencontré de difficultés insurmontables pendant ce projet. Il fut parfois compliqué d'analyser et de traiter les données JSON et XML que nous renvoyais les différentes API (problème d'encodage en UTF-8, informations présentes pour certaines villes mais pas pour d'autres …) et de ne pas trop se mélanger les pinceaux entre toutes les informations fournies. Cependant en séparant le code comme nous l'avons fait (plusieurs fichiers PHP ayant chacun une tache spécifique), nous avons réussi à bien nous organiser.

Nous avons eu un petit problème lors de l'implémentation de la connexion et de l'inscription, mais ce problème était vraisemblablement lié au serveur que nous utilisions avec la base de données qu'au code lui-mêm. Nous avons donc effectué nos développements sur un serveur web local au serveur de base de donnée. Ces modifications n'ont pas influencé directement le code.

Bilan

En nous demandant de créer une carte interactive, nous avons pu découvrir et expérimenter de nombreuses nouvelles techniques. Cela nous a permis de devenir plus à l'aise avec le JavaScript mais aussi de manipuler à nouveau le PHP et d'apprendre à utiliser ces deux langages ensembles, notamment avec l'utilisation des procédure AJAX et bien évidemment avec l'utilisation des différentes APIs. Nous avons aussi eu l'occasion de manipuler à nouveau les données aux formats JSON et plus précisément la norme geoJSON.

Aussi l'architecture Simple Page Application à micro services, permise par les procédures AJAX permet de créer des application web plus modernes et ergonomiques puisqu'elles n'imposent pas de chargements à l'utilisateur.

Nous avons pu facilement travailler en groupe avec la séparation des services en plusieurs fichiers PHP et javaScript et en utilisant l'outil de programmation collaboratif Git ainsi que le client graphique Fork. Nous pouvions ainsi travailler chacun sur nos branche et Arsène s'occupait de fusionner toutes nos modifications une fois qu'elles étaient fonctionnelles.

Ce projet très complet nous a ainsi permis de travailler facilement, et sur un unique sujet toutes ces technologies. Il nous fut facile, avec l'aide de notre professeur, de trouver à chaque fois de nouveaux services à implémenter pour en apprendre toujours plus et ainsi être fières de notre application.

Améliorations possibles :

Une évolution de notre application serait de l'étendre à d'autres régions que l'ile de France. Nous avions déjà essayé de charger sur une zone plus étendue, mais les temps de chargements étaient beaucoup trop longs à cause de la quantité de données. Une chose qu'il serait intéressant de faire serait de demander, avant de charger la carte, quelle région de France intéresse le visiteur. Ainsi notre application fonctionnerait pour toutes les régions tout en limitant les temps de chargements. Il nous est cependant impossible de le faire actuellement car nous n'avons pas été en mesure de trouver les données nécessaires (nombre de logement sociaux …) pour toutes les régions de France. Mais si un jour de telles données deviennent disponibles il pourrait être intéressant de faire évoluer notre application.