POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

INGENIEUR EN SCIENCES INFORMATIQUES

RAPPORT DU PROJET WEB

Etudiant : Soulaiman ZABOURDINE
(SI5- Parcours AL)

Responsables : Peter Sanders, Hugo Mallet

Table des matières

1. Identifiant GitHub

Vous pouvez accéder au projet via ce lien ainsi qu'à mon espace GitHub (ZabourdineSoulaiman) via ce lien.

2. Tâches effectuées

Dans le cadre de ce projet, je me suis plus focalisé sur la partie back-end que front-end. Cependant, j'ai tout de même participé et implémenté plusieurs fonctionnalités de bout en bout dans le côté front-end de l'application.

Fonctionnalités Back-End implémentées :

- Téléchargement du fichier .zip via https://donnees.roulez-eco.fr/opendata. (1h)
- Extraction du fichier afin de récupérer le fichier xml. (2h)
- Conversion du fichier XML en données JSON. (Très long, j'ai dû essayer plusieurs librairies)
- Insertion du JSON dans la base de données. (1h)
- Implémentation d'un entry point GET permettant de récupérer les stations à proximités d'une coordonnées fournies. (2h)

Fonctionnalités Front-End implémentées

- Implémentation d'une barre de recherche permettant de définir la position actuelle de l'utilisateur. (2h)
- Implémentation d'une fonctionnalité d'itinéraire permettant à l'application d'interroger l'API OpenRouteService. (3h)
- Modification de la carte afin d'afficher un itinéraire. (Très long)
- Implémentation de la modification des prix de carburants (partie front). (1h)

Barre de recherche

L'application permet à l'utilisateur de visualiser un trajet d'un point A vers une station essence répertoriée par l'application. Pour cela, l'utilisateur peut utiliser une barre de recherche et y indiquer sa position actuelle grâce à une adresse. Afin d'aider l'utilisateur à retrouver une adresse, je voulais que la barre de recherche puisse proposer des suggestions d'adresse en fonction des informations entrée dans cette dernière.

Je me suis donc appuyée sur une API qui permet de récupérer des adresses : *api-adresse.data.gouv.fr*. L'API retourne un JSON contenant une liste de suggestion d'adresses, il suffisait alors de faire les implémentations nécessaires afin d'utiliser de l'autocomplétions.

Cette fonctionnalité a été implémentée dans le fichier 'addressLocator.js' et 'header.jsx'.

Affichage d'un itinéraire

Comme évoqué plus tôt, l'application permet à ses utilisateurs de construire un itinéraire d'un point A à une station d'essence. Pour cela, je me suis appuyé sur l'API *OpenRouteService*, qui permet de tracer l'itinéraire entre deux points en choisissant un moyen de locomotion (vélo/voiture etc.). Afin de récupérer la position de départ, je me suis appuyé sur la barre de recherche qui permet de trouver une adresse, mais aussi de trouver les coordonnées GPS d'une adresse.

L'API retourne donc un JSON contenant un trajet complet avec des informations supplémentaires tels que la durée du trajet. Concernant le trajet retourné dans le JSON, il s'agit d'une liste de coordonnées GPS. Lorsqu'on relie chaque point de la liste de JPS, on obtient un itinéraire de bout en bout.

Cette fonctionnalité a été implémentée dans le fichier 'ItineraryCalculator.js'.

Modification des prix

Nous voulions permettre à nos utilisateurs connectés de modifier/corriger le prix du carburant pour une station donnée lorsqu'une rectification est nécessaire. Pour cela, il est nécessaire d'envoyer une requête au backend où les informations liées aux stations sont stockées dans une base de données Mongo. Cette fonctionnalité nécessitait alors des implémentations dans le frontend ainsi que dans le backend.

Je me suis chargée de la partie front de la fonctionnalité, en m'insérant sur les pop-ups des stations essences. Dans un premier temps, il était nécessaire de faire la distinction entre les utilisateurs connectés ainsi que les utilisateurs non connectés. En fonction de cette information, le prix sera modifiable, et des boutons seront utilisables pour envoyer une requête bers le serveur backend. La requête contiendra alors l'ID de la station concernée, le nom du carburant à modifier, ainsi que la nouvelle valeur à affecter.

3. Organisation du groupe (GitHub)

Afin de nous départager les tâches équitablement, nous avions créées des issues sur le GitHub du projet. Nous pouvions alors y retrouver toutes les tâches qui restait à faire, ainsi que savoir ce que d'y ajouter des nouvelles tâches lorsqu'il était nécessaire.

4. Difficultés rencontrées

En ce qui concerne la partie front-end, j'ai été bloqué pendant très longtemps sur la fonctionnalité itinéraire, notamment pour ajouter des balises <Polyline> dans mon <MapContainer> afin de dessiner des lignes. Finalement, nous avions réussi à débloquer la situation en travaillant en binôme sur cette partie.

5. Commentaires sur le code

Code élégant / optimal : drawItinerary(endXString, endYString)
Cette fonction permet de de dessiner l'itinéraire d'un point A vers un point B. Pour cela, il est possible de s'appuyer sur une adresse fournie par l'utilisateur, ou bien sur la position actuelle de l'utilisateur, récupérée par le navigateur. Cette fonctionnalités effectue deux appels externes vers deux API distinctes, une pour récupérer une coordonnées GPS (ligne askOneAdress()), ainsi qu'une pour récupérer un trajet de bout en en bout (ligne getItinerary()).

Code Méritant une optimisation : loadFromList()
Cette fonction permet de modifier la barre de recherche d'adresses, afin de suggérer à l'utilisateur, des adresses en 'auto-complete'. Pour cela, la fonction s'appuie sur une API externe '*api-adresse.data.gouv.fr'.* Cette méthode est appelée via l'évènement 'onKeyUp', mais la recharge de la liste peut prendre du temps. Cette méthode peut être améliorer.