RAPPORT PROJET WEB ENSIIE FIPA

Site web d'aide au stationnement dans Paris

Geoffrey DELVAL, Rémi GUIJARRO-ESPINOSA, Adelino ARAUJO, François SAINTIER

04/03/2020
ENSIIE Projet Web – Rémi PARPAILLON

SOMMAIRE

INTRODUCTION

Dans le cadre du projet web, il nous fallait dans un premier temps déterminer un sujet correspondant aux contraintes fixées. Nous avons donc cherché à explorer des idées novatrices mais en même temps réalisables dans le temps imparti. Après de longs échanges et l'estimation des différentes propositions, nous avons choisi de répondre à une problématique de plus en plus présente:

Comment aider et améliorer l'expérience utilisateur des automobilistes parisiens dans leur quête continuelle de place de stationnement?

Après de plus amples recherches, nous avons découvert ce qui deviendrait notre première source de données principales: les données openData de la ville de Paris. C'est ainsi qu'est né notre projet de développer un site web qui permettrait de trouver une place de stationnement dans Paris.

APPROCHE MISE EN PLACE

La toute première étape une fois le sujet déterminé a été de définir le langage utilisé. Certes l'existence du squelette PHP proposé par notre encadrant était un argument de poids, mais nous aimons le défi, l'audace et le non-conformisme. Nous sommes donc partis sur une stack technique en javascript, qui nous paraît modulaire, évolutive et bien adaptée à notre projet.

Une fois le langage choisi, nous nous sommes lancés dans la mise en place des outils de gestion de projet et de collaboration. Pour les échanges, entraides, discussions et débats nous avons sélectionné Discord. Pour la gestion de versions des fichiers sources, nous avons utilisé Github. Et pour le suivi du projet, le suivi d'avancement, la répartition des tâches nous avons expérimenté les fonctions que propose Github.

Nous avons initié cet outil de gestion des tâches en catégorisant les grandes étapes, qu'elles soient indispensables ou facultatives, et en se fixant l'objectif d'avoir un livrable fonctionnel à l'issue de chacune d'entre elles, un peu à l'instar de la méthode scrum.

Voici les étapes définies:

1. Récupérer la liste des places et les afficher sur une carte
2. Gérer la géolocalisation
3. Créer un compte utilisateur
4. Ajouter des champs pour filtrer la recherche en fonction des critères disponibles
5. Gérer les stationnements avec la possibilité de réserver une place et d'informer les autres utilisateurs qu'une place va se libérer
6. Gérer un panel d'administration pour gérer les utilisateurs, leurs réservations ainsi que les places de stationnement
7. Ajouter un aspect social à l'application afin de créer une communauté et la possibilité de noter et récompenser les bons utilisateurs

PROBLÉMATIQUES RENCONTRÉES ET SOLUTIONS APPORTÉES

Plusieurs problèmes ont été rencontrés au cours de ce projet, leur temps de résolution a été plus ou moins long mais voici la liste des principaux:

1. La première problématique que nous avons rencontré a été la mise en place du squelette initial en javascript, que l'on puisse installer et lancer depuis le Makefile et tout ça dans un Docker. De nombreux utilitaires et sites proposent des boilerplates déjà tout prêts mais nous voulions avoir la main sur les différents packages installés. De plus, il a fallu générer ces squelettes pour la partie back et pour la partie front.

   Pour générer le squelette de la partie back, nous avons utilisé 'express-generator' qui crée les répertoires de base, le fichier package.json et une première page "Hello World". Pour générer le squelette de la partie front, nous avons utilisé 'vue-cli' qui crée également un projet basique, sans superflu.

2. La seconde problématique a été de choisir la méthode pour requêter l'API de la ville de Paris, sélectionner les données utiles pour le projet, optimiser le logiciel afin de limiter le nombre de requêtes à l'API potentiellement génératrices de latence réseau.

   Pour résoudre le problème de limitations de requêtes, nous avons mis en place le module apicache qui permet de stocker en cache le retour de chaque requête à l'API et de travailler avec durant un temps que nous avons défini arbitrairement à 5 minutes.

3. Le choix des packages npm à utiliser pour les différentes fonctions que nous voulions mettre en place a également été une problématique. Un panel énorme existe et il faut réussir à sélectionner le bon, qui corresponde à notre besoin et l'intégrer dans le projet.

   Pour cela il n'existe pas de solution unique, nous avons dû en tester plusieurs et ne garder que ceux qui répondaient à nos attentes.

4. Une des grosses problématiques a été l'affichage de la carte et d'y superposer les markers représentants chaque place de stationnement. Nous sommes partis initialement sur la solution proposée par leaflet mais avons finalement choisi de travailler avec MapBox qui propose des options supplémentaires et un design plus actuel.

5. Une des problématiques que nous avons rencontré mais que nous n'avons pas eu le temps de traiter est que l'API de la ville de Paris renvoie un nombre de places de parking limité à 10000 alors qu'elle en comporte plus de 30000 au total.

   Pour cela, nous pensions soit cumuler plusieurs requêtes pour récupérer l'intégralité des données, ce qui serait très lourd et peu optimisé, soit gérer différents layers en fonction du niveau de zoom et renvoyer une nouvelle requête avec un filtre de zone au moment du zoom.

6. En lien avec le problème précédent, nous avons également rencontré un problème de performances pour afficher les 10000 places de parking simultanément sur la carte. En effet, tel que nous avons fait l'architecture, le site ne permet pas d'afficher la totalité des places. Nous avons cherché une solution mais celle-ci s'est avérée trop longue à mettre en place par rapport au temps qu'il nous restait.

   L'utilisation de layers dans la carte permettrait notamment d'afficher des informations différentes en fonction du niveau de zoom (par exemple le nombre de places par quartier plutôt que l'ensemble des places constamment). L'utilisation du format d'encodage en GeoJSON permettrait en complément de simplifier la gestion de toutes les données récupérées afférentes aux places de parking.

7. Nous avons également manqué de temps pour pouvoir aller au bout de nos idées.

   Il a donc fallu faire des choix et revoir à la baisse certaines fonctionnalités. Ce fut instructif mais un peu frustrant de ne pas pouvoir aller au bout.

8. Enfin la dernière problématique a été que nous avons tous découvert réellement le véritable développement web autour de ce projet réaliste et ambitieux.

   Pour cela, seuls l'entraide, les cours particuliers, gettings started et tutos sur internet nous ont permis d'avoir au final une application fonctionnelle, même si toutes les fonctionnalités que l'on souhaitait à l'origine n'ont pas pu être intégrées.

POSSIBILITÉS D'ÉVOLUTION

Le timing du projet ne nous ayant pas permis de développer toutes les fonctionnalités, voici une liste des fonctionnalités envisageables:

- Améliorer les filtres de sélection pour permettre à l'utilisateur de trouver une place où il puisse recharger son véhicule, choisir une zone résidentielle s'il habite dans Paris,
- Finaliser les actions accessibles depuis le compte utilisateur et ajouter les données personnelles qu'il pourra y renseigner (véhicules favoris, adresses favorites, localisation de la place de stationnement utilisée),
- Ajouter une fonction 'Find My Car' pour enregistrer le lieu du stationnement et pouvoir y retourner facilement,
- Gérer la disponibilité des places,
- Se développer à d'autres villes en France, puis à terme à l'international,
- Elargir l'offre et proposer les places dans les parkings souterrains et parkings de particuliers,
- Développer une application mobile en parallèle du site web
- Ajouter la possibilité de payer son stationnement via l'application

RÉPARTITION DES RÔLES AU SEIN DE L'ÉQUIPE

Les rôles n'ont pas été strictement définis, chacun a eu le loisir de travailler sur la partie qui l'intéressait, qu'elle mêle back et front ou uniquement l'un des deux. Nous avons donc plutôt travaillé en terme de feature. Chacun d'entre nous a eu aussi le rôle de code reviewer pour valider les pull request des autres membres de l'équipe.

Collaborateur	Rôles
Geoffrey	- Compte utilisateur - Authentification - Inscription - Validation des données
Rémi	- Mise en place du squelette front - Gestion de la carte - Affichage des markers - Panel d'admin - Option d'affichage de la map
Adelino	- Requêtage de l'API depuis le back - Sauvegarde du résultat et renvoi au front via une API
François	- Mise en place du squelette back - Dockerisation du projet - Gestion de la carte - Autocompletion du champ de saisie de l'adresse

CONCLUSION

Tout d'abord, nous avons beaucoup apprécié ce projet web et la possibilité de choisir le sujet de notre choix. Son développement a été très instructif, notamment au travers des différentes problématiques rencontrées.

Le livrable et les différentes fonctionnalités sont opérationnels même si nous sommes un peu frustrés et que nous aurions voulu aller plus loin si nous avions eu un peu plus de temps alloué au projet.

STACK TECHNIQUE

Voici la stack technique que nous avons utilisé pour ce projet:

- Back:
    - express: serveur
    - jest: tests unitaires
    - newman: tests API
    - nodemon: démon pour le développement
    - apicache: stockage des données en cache
- Front:
    - Vue.js
    - Mountbank: simulation des réponses du back pour les tests
    - OpenStreetMap: fonds de cartes
    - Mapbox: Outil avancé de gestion de carte
    - vue-places: Saisie d'adresses auto complétées
    - Bootstrap: Boîte à outils de composants web
    - axios: Requêtage d'API
    - Cypress.io: test fonctionnel front
    - vuelidate: validation d'interface graphique