Compte rendu de TP :

Implémentation libre d'une application mobile hybride

M2 Miage - Module Développement d'applications mobiles

BLAIECH Emna
HOMBERT Sarah

UNIVERSITÉ DE NANTES

Table des matières

1. Introduction

Ce projet s'inscrit dans le cadre du module de développement d'application mobile en M2 MIAGE ISI. L'objectif est l'implémentation d'une application mobile hybride, basée sur les technologies VueJS et Capacitor. VueJS est un framework de développement web Single Page Application qui utilise principalement les langages HTML et Javascript.
Capacitor est une solution qui permet de créer des applications Android, IOS ou progressive depuis une application Web en Javascript.
Notre projet consiste donc en la création d'une application web responsive, qui sera ensuite la base de l'application Android créée à l'aide de Capacitor.

2. L'application

Nous avons développé l'application Parkin. Cette dernière permet de retrouver les parkings à proximité de la ville de Nantes et de consulter les informations d'accessibilité les concernant.

2.1. Fonctionnalités

L'application est divisée en plusieurs composants :

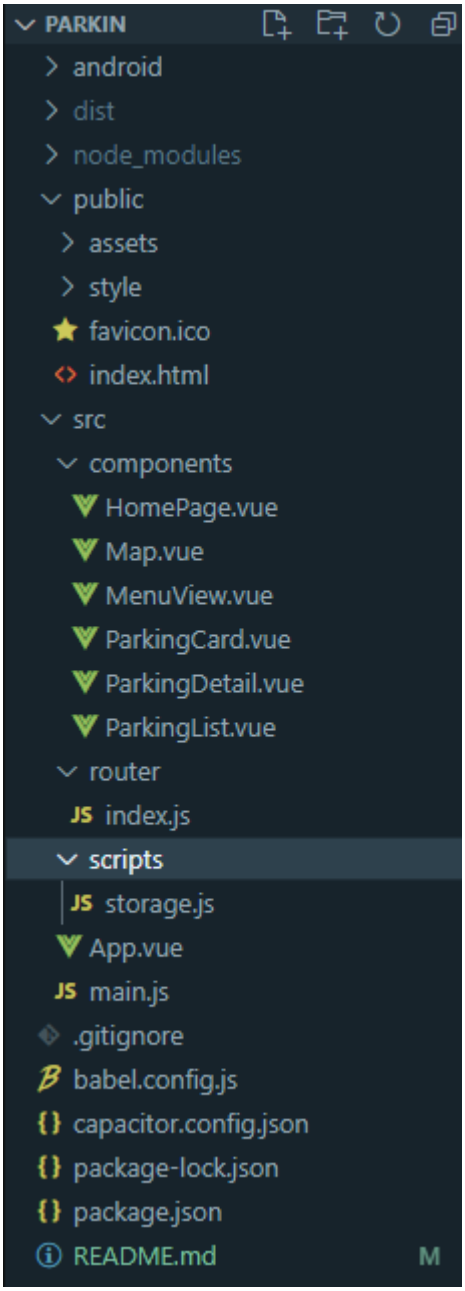
- un menu permettant d'accéder aux différentes pages (accessible depuis toutes)
- 4 pages différentes :

La page **Home** présente les parkings à proximité . La page **Parkings** présente tous les parkings du jeu de données et un bouton détails menant vers la page **Parking details** où les informations relatives au parking sélectionné remontent. La page **Carte** propose un affichage des parkings sur une carte selon la localisation de l'appareil via un plugin capacitor.

2.2. Utilisation des APIs

Nous avons choisi d'exploiter l'API REST de Nantes Métropole permettant de récupérer les données des parkings publics de Nantes disponible via ce lien. Un deuxième jeu de données de Nantes Métropole propose les disponibilités en temps réel de certains des parkings proposés sur le premier jeu (mis à jour toutes les 5 minutes).

3. Choix d'implémentation



Le code source de l'application est consultable à la page suivante : https://github.com/SHombert/Parkin

3.1. Architecture globale

L'application est composée de différentes parties :

- les composants Vues, qui comprennent le code métier de l'application
  - `HomePage.vue` : il s'agit de la page d'accueil Home
  - `MenuView.vue` : composant de menu accessible depuis toutes les pages
  - `Map.vue` : la page Carte accessible depuis le menu
  - `ParkingList.vue` : la page de liste des parkings accessible depuis le menu, affichés dans des ParkingCard
  - `ParkingCard.vue` : composant permettant d'afficher un aperçu d'un parking
  - `ParkingDetail.vue` : page d'informations détaillées d'un parking, accessible depuis un ParkingCard ou depuis un marqueur sur la carte
- le fichier index.js qui permet de configurer les routes de l'application
- le dossier Scripts qui contient les scripts personnalisés, notamment celui pour la gestion du local Storage (voir plugins capacitor)

3.2. Routes

L'application possède quatres routes définies dans le fichier router/index.js :

- / : route par défaut de l'application, mène directement à la page d'accueil Home
- /parkings : route vers le composant ParkingList
- /map : route vers le composant Map
- /parking-detail/:id : route vers un composant ParkingDetail en passant en paramètre la props "id" pour identifier le parking. Cette route n'est pas accessible depuis le menu, seulement depuis les composants ParkingList et Map.

3.3. Style

L'application a un design responsif. Les classes de style CSS utilisées sont celles de la librairie Bootstrap 5. La plupart sont responsives par défaut, nous n'avons donc que très peu de CSS personnalisé.
Nous n'avons pas utilisé le plugin bootstrapvue spécifique à VueJS car celui-ci n'est pas compatible avec la version 3 de VueJS que nous utilisons.

3.4. Appels REST

Pour effectuer les appels REST nous avons importé la librairie Axios. Celle-ci permet d'utiliser et afficher les données de l'API en se basant sur les promesses. Nous avons repris cette librairie après l'avoir découverte en cours et pour les multiples avantages qu'elle offre.
Avant d'interroger l'API, il fallait consulter la structure des données qu'elle renvoie afin de savoir le détail du contenu affiché. La structure de l'API Parking de Nantes Métropoles est présentée en annexe (Annexe A : structure de données des parkings de Nantes).

3.5. Maps

Afin d'intégrer une carte dans notre application pour le composant Map, nous avons utilisé la librairie Leaflet. Notre choix s'est porté sur cette solution car il s'agit d'une librairie open Source. Pour le fond de carte, nous avons choisi OpenStreetMap : "OpenStreetMap® est un ensemble de *données ouvertes*, disponibles sous la licence libre Open Data Commons Open Database License (ODbL) accordée par la Fondation OpenStreetMap (OSMF)."
Il existe également une librairie leaflet-vue pour des composants Vue. Nous avons utilisé cette dernière. Toutefois, si celle-ci est désormais compatible avec VueJS 3, il reste de nombreuses issues en cours de résolution car la migration est encore récente. Nous avons rencontré plusieurs problèmes assez chronophage à cause de ceci. Après coup, il aurait pu être plus simple d'utiliser la librairie leaflet directement en JS.

3.6. Plugins Capacitor

Nous avons implémenté deux plugins capacitor :

- API de géolocalisation : fournit des méthodes permettant d'obtenir et suivre la position de l'appareil en temps réel à l'aide du GPS intégré de celui-ci. Elle offre aussi des informations d'altitude, de cap et de vitesse si disponibles.
  Dans l'application, ce plugin est mis en place au niveau de la page Carte afin de renvoyer la position de l'utilisateur connecté à l'application et lui permettre de visualiser les parkings à proximité.
- API de stockage : fournit une base de clé-valeur pour sauvegarder des données.

Dans l'application, ce plugin est mis en place pour stocker en cache le résultat de la requête REST qui récupère les données et pour éviter de lancer la requête plus d'une fois en naviguant entre les différentes pages.
Nous avons choisi de sauvegarder les parkings avec le recordid en tant que clé et l'objet complet retourné par l'appel REST en valeur.

4. Evolution de l'application

Nous avons identifié plusieurs pistes d'améliorations pour l'application, présentées ci-dessous à défaut de temps pour les mettre en place.

4.1. Design

La page Home pourrait être revue avec notamment la mise en place d'un Carousel présentant l'application, puis deux autres slides pour présenter les pages liste et map, avec des liens pour s'y rendre.
La page Liste a un défaut d'alignement au niveau des composants Card de bootstrap.

4.2. Géolocalisation dynamique

La géolocalisation mise en place jusqu'ici n'est pas dynamique. Pour l'instant elle propose uniquement de localiser l'appareil au lancement du composant, mais elle ne se met pas à jour. Il faudrait ajuster le composant avec l'API Capacitor pour que ce soit le cas.

4.3. Places disponibles

Cela consisterait en la mise en place d'un deuxième jeu de données permettant d'accéder aux informations sur les places disponibles de certains parkings. Ces informations devraient compléter celles déjà récupérées et être visibles depuis la popup du marqueur sur la carte, ainsi que dans les composants ParkingCard et ParkingDetail.

4.4. Recherche sur la liste des parkings

Il s'agit d'ajouter une barre de recherche dans le composant ParkingList, pour faciliter l'accès aux parkings souhaités. La recherche doit toutefois être pertinente, et pourrait se baser sur d'autres critères que le nom du parking : accès PMR, vélo, chargement électrique, adresse…
Les parkings résultant de la recherche s'afficheraient dans le composant.

5. Conclusion

Ce projet nous a permis de mettre en œuvre tout ce que nous avons appris pendant le module de développement d'applications mobiles. Étant toutes les deux des novices dans ce domaine, cela nous a permis de grandement monter en compétences. Nous avons rencontré des difficultés principalement dues au fait que certaines librairies utilisées ne sont pas encore compatibles avec la version 3 de vue. Le deuxième point important où nous étions moins à l'aise était dans la mise en œuvre des plugins capacitor. Nous avons trouvé peu d'exemples sur internet, et la documentation à elle-seule pour des développeurs peu expérimentés peut s'avérer légère.

Nous avons toutefois réussi à respecter toutes les contraintes du sujet demandé pour avoir une application résultante fonctionnelle et un rendu graphique travaillé.

Sitographie

Capacitor Plugins - Documentation Plugins Capacitor

Introduction · Bootstrap v5.0 - Documentation Bootstrap V5

vue-leaflet/vue-leaflet: vue-leaflet compatible with vue3 - Repository Git Vue Leaflet

Leaflet - a JavaScript library for interactive maps - Documentation Leaflet library

https://data.nantesmetropole.fr/explore/dataset/244400404_parkings-publics-nantes-disponibilites/api/?disjunctive.grp_nom&disjunctive.grp_statut - API Disponibilités Parkings publics de Nantes, Data Nantes métropole

https://data.nantesmetropole.fr/explore/dataset/244400404_parkings-publics-nantes/api/?disjunctive.libcategorie&disjunctive.libtype&disjunctive.acces_pmr&disjunctive.service_velo&disjunctive.stationnement_velo&disjunctive.stationnement_velo_securise&disjunctive.moyen_paiement - API Parkings de Nantes, Data Nantes Métropoles

Annexes

A. Annexe A : structure de données des parkings de nantes