Équipe :

COBAT Guillaume
BUCHE Sylvain
GODET Louis-Xavier
BOIVENT Pierre
PEDRON Matisse

PROJET DE SYNTHESE

VEILLE TECHNOLOGIQUE

OppRoadInfo

Client : M. Le Sommer
Tuteur : M. Kerbellec

Sommaire

Introduction

La veille technologique nous permet de sélectionner de la meilleure façon possible, les technologies à utiliser pour réaliser ce projet. Un bon choix technologique est primordial pour simplifier le développement, car il ne pourra pas être changé au cours du projet. La technologie choisie devra donc répondre à nos attentes, mais aussi aux attentes du client.

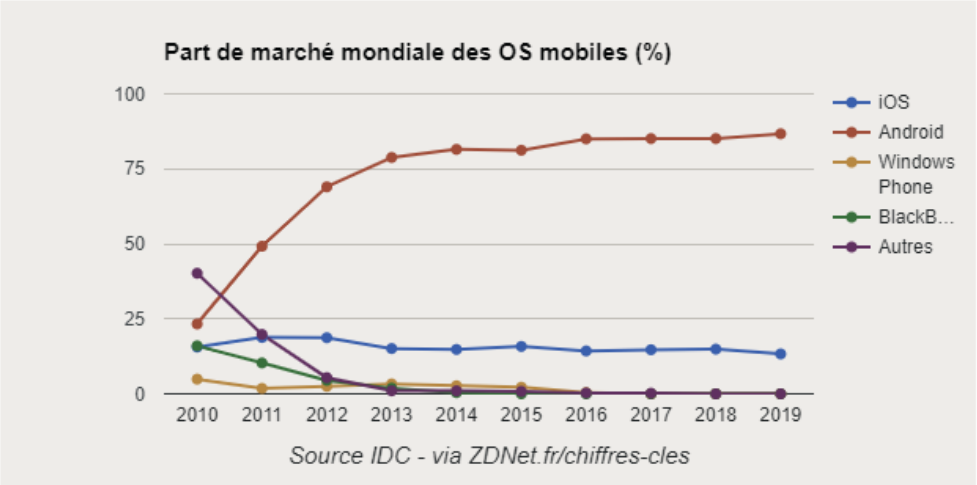
Nous avons orienté nos recherches sur 3 technologies, React-Native, NativeScript et Android Native. Il existe de nombreuses autres technologies telles que Flutter, Apache Cordova, Ionic… Nous n'avons pas eu le temps de toutes les tester, c'est pourquoi nous nous sommes concentrés sur les technologies qui nous ont été conseillées par notre tuteur et le client.

Dans un premier temps, nous avons étudié pour chaque technologie leurs niveaux de documentation. Ensuite nous avons essayé de nous les approprier et d'implémenter différentes cartes.

Historique des systèmes d'exploitation mobile

Notre application est destinée à être utilisée sur un système d'exploitation mobile. Il existe aujourd'hui plusieurs systèmes d'exploitation pour smartphones, mais il y en a deux qui dominent le marché.

En effet, les systèmes iOS et Android sont de loin les deux systèmes d'exploitation les plus utilisés. Le système iOS est développé par Apple pour les iPhones, et les iPads. Sous iOS, l'interface est très intuitive et il est facile de classer les applications. Le système Android est développé par Google, et est utilisé par de grandes marques telles que Samsung ou encore Xiaomi. La personnalisation du système Android est plus large que celle proposée par le système iOS. Il est déployé sur la très grande majorité des smartphones dans le monde (comme le montre le graphe ci-dessous).



Bien qu'au début des années 2010, la répartition en termes de pourcentage entre les différents systèmes d'exploitation était plus repartie, celui qui est le plus présent de nos jours sur le marché est très largement Android. C'est pourquoi il reste cohérent d'opter pour une application disponible sur ce seul système d'exploitation.

React-Native

React Native est un framework créé par Facebook en 2015. Il est basé sur React, une librairie Javascript. Ce framework est complètement open source, puissant et assez stable pour livrer des applications sur les stores d'Apple et de Google. React Native est donc cross-platform, c'est-à-dire qu'il y a besoin de coder une seule application pour qu'elle soit compatible sous IOS et Android. Il fonctionne avec les composants mobiles natifs et est entièrement gratuit.

Afin de tester React Native, nous avons suivi le cours d'OpenClassroom pour en apprendre les bases. Nous avons d'ailleurs utilisé la version 4.13.0 de React Native.

La prise en main de React Native s'est relativement bien faite, en effet nous avons trouvé que React était assez intuitif. Nous avons aussi vu qu'il y avait la possibilité de tester l'exécution de notre code en live, grâce à l'application Expo téléchargée sur nos smartphones. L'application fonctionne très bien, il suffit de scanner un QR code qui se crée à l'ouverture du serveur pour afficher le résultat de notre code.

Le framework React Native est basé sur le langage web JavaScript. Le langage JavaScript est par défaut asynchrone, ce qui permet une exécution plus rapide qu'un langage synchrone (ex: Java). Il se trouve que les projets React-Native peuvent prendre deux formes. La première est la forme CRNA (Create-React-Native-App). Plus simpliste, elle est destinée à du développement d'applications rapides, car il y a peu de configurations. De nombreuses plateformes permettent de créer des CRNA, notamment Expo comme précédemment décrite. La deuxième forme propose quant à elle les mêmes fonctionnalités que la première, mais est plus complexe, car elle permet d'intégrer des fonctionnalités natives.

Lorsque nous avons tenté d'implémenter une carte Mapbox ou Leaflet, les choses se sont compliquées. Tout d'abord, Leaflet ne semble pas être implémentable sur React-Native. Après plusieurs tentatives peu concluantes, nous avons décidé de ne pas continuer avec Leaflet. Ensuite, nous avons essayé d'implémenter Mapbox qui utilise du code natif, et qui n'est pas supporté par les applications CRNA.

Nous avons finalement réussi à implémenter une carte Mapbox après avoir suivi un tutoriel. Cette application fonctionne uniquement sur Android car l'application IOS nécessite une machine Mac pour la fabriquer, que nous ne possédons pas. L'un des défauts de devoir utiliser Mapbox est qu'il utilise du code natif et Expo ne le supporte pas. Cela signifie que pour avoir une preview de l'application, il faut configurer un environnement Android. Nous avons réussi à le faire, mais cela a été complexe.

Voici quelques résultats d'implémentations de Mapbox avec React-Native que nous avons pu expérimenter.

Puis, nous avons essayé de chercher une autre librairie de carte, et nous sommes tombés sur MapView. Nous l'avons trouvé très facile à intégrer à l'application. Cependant, l'inconvénient est que cette carte provient de Google, et n'offre pas un service entièrement gratuit.

Nos tests d'implémentation seront disponibles en .apk sur le GitLab.

Source du tuto : https://openclassrooms.com/fr/courses/4902061-developpez-une-application-mobile-react-native/4902068-decouvrez-le-developpement-mobile-actuel

NativeScript

NativeScript est un framework open source gratuit pour le développement mobile, développé par l'entreprise Telerik depuis 2015. On peut l'utiliser de plusieurs manières :

- En l'associant avec le framework Angular.
- En l'associant avec le framework Vue.js.
- En utilisant du javascript.
- En utilisant du typescript.

Il permet le développement d'applications cross-platform sur iOS et Android, et donne un accès direct aux fonctionnalités natives des différents appareils mobiles. NativeScript étant basé sur le JavaScript, l'exécution est asynchrone, ce qui permet une rapidité accrue. C'est un framework extensible qui peut être complété avec de nombreux plugins et librairies disponibles sur la marketplace.

NativeScript peut s'utiliser soit avec un éditeur de texte en ligne possédant des fonctionnalités de drag and drop, soit de façon locale avec un terminal et un IDE. Dans les 2 cas, le framework utilise une fonctionnalité de "preview" à l'aide d'un QR Code et d'une application mobile pour afficher directement le résultat de l'exécution de l'application.

NativeScript dispose d'une communauté grandissante, mais moins importante que les autres frameworks. Par exemple, il est assez compliqué de trouver des ressources (tutoriels, vidéos) en français sur un sujet précis. Même si l'on est capable de suivre des tutoriels en anglais, un tutoriel en français est tout de suite plus simple à suivre lorsque l'on découvre un nouvel outil.

La version de NativeScript que nous avons utilisée est la version 7.0.11.

Pour nos tests nous avons essayé de suivre 3 tutos différents qui se sont malheureusement tous révélés infructueux :

- https://dev.to/cendekia/develop-a-map-application-using-nativescript-and-map-box-10kk
- https://blog.angelengineering.com/nativescript-maps/
- https://nativescript.org/blog/include-feature-rich-maps-in-a-nativescript-vue-app-with-mapbox/

Nous avons dès la création du projet rencontré un problème : lors de l'exécution de l'application en mode "preview", une exception se produisait, alors même que le code était un "Hello World" créé par défaut. Nous avons ensuite essayé de réaliser le tutoriel avec {N}+Vue.js et avons rencontré le même problème (le tuto demandait {N}+Angular). Nous avons réussi à faire fonctionner la preview avec {N}+javascript (ce qui ne nous a pas servi car l'architecture du projet était trop différente de celle du tutoriel).

A noter : l'application "Native Playground" dans sa version iOS, n'est pas fonctionnelle. Elle se ferme peu après l'ouverture sans même laisser la possibilité de scanner un quelconque QR Code.

Nous avons ensuite essayé les 2ème et 3ème tutoriels. Nous avons cette fois-ci réussi à créer les projets correctement, mais l'implémentation de la carte n'a pas abouti.

Android Native

Au cours de nos recherches, nous avons également envisagé de développer l'application mobile en Android Native. L'Android Native est la technologie de base proposée par Google pour développer des applications Android. Les applications natives offrent de meilleures performances que celles développées en cross-platform. Cependant, elle a comme inconvénient de ne pouvoir être utilisée que pour les smartphones Android, elle n'est pas multiplateforme.

Développer en Android Native nous laisse plusieurs choix de langage :

- Java
- Kotlin
- C/C++

Nous ne sommes pas très familiers avec le C et le C++, cependant, le Java est un langage que l'on maîtrise bien. Quant à Kotlin, c'est le langage qui est officiellement recommandé par Google depuis quelques années, pour développer des applications Android. La partie "vue" du développement Android Natif se fait quant à elle en XML. Java étant le langage que nous maîtrisons le mieux, ce sera notre choix privilégié si nous choisissons de développer en Android Native.

Pour développer sur Android, il est conseillé d'utiliser certains IDE. Eclipse, qui est un IDE très populaire, en fait partie. Android Studio est un deuxième choix possible. AVD (Android virtual device) est inclus dans les deux IDEs et permet d'émuler des smartphones avec différentes versions d'Android. Cela a comme avantage de tester directement l'application dans l'IDE et qui plus est, contrôler la rétrocompatibilité entre les différentes versions d'Android.

Android Studio est un IDE basé sur IntelliJ proposé par JetBrain. Pour la plupart d'entre nous, nous utilisons déjà les logiciels de la suite JetBrain, nous ne serions alors pas dépaysés par l'utilisation d'Android Studio.

Afin d'avoir un avis objectif sur cette technologie de développement, nous avons essayé d'implémenter une simple carte avec Android Studio. Ce test nous permet d'appréhender le développement en Android Natif et sa difficulté, en fonction des connaissances que nous disposons déjà.

Pour ce test nous avons décidé d'implémenter une carte à l'aide de la librairie OSMdroid (version 6.1.0), qui permet d'utiliser la carte créée par le projet collaboratif Openstreetmap.

Le test s'est avéré concluant, car en une heure nous avons atteint notre objectif (Voir le fichier Android_Native_+_OSM.apk).

Ce test nous a aussi permis d'appréhender la librairie d'Android Native et sa documentation (https://developer.android.com/docs). Nous avons constaté que sa documentation était très riche, de plus l'utilisation du langage Java, plutôt qu'un langage inconnu, nous a grandement facilité la tâche.

Tableaux récapitulatifs

	React Native	**Android**	**NativeScript**
Editeur	Facebook	Google	Telerik
Langages	React	Java Kotlin C/C++	Javascript TypeScript
Combinaison avec d'autres frameworks	Non	Non	Angular Vue.js
Cross-platform	Oui	Non	Oui
Version actuelle	4.13.0	API 30	7.0.11
Grande popularité (nombreux tutoriels, vidéos, forums).	Oui	Oui	Non
Résultats des tests effectués	Difficultés d'implémentations des cartes	Réussite de l'implémentation d'une carte	Beaucoup de problèmes

	Mapbox	**Leaflet**	**Google Maps**	**Open Street Map**
Editeur	MapBox inc.	Cloud-Made	Google	Communauté OpenStreetMap
Version de l'API / librairie	1.12.0	1.7.1	3.42	6.1
Coût	Gratuit jusqu'à 25 000 utilisateurs par mois	Gratuit	Facturation en fonction de l'utilisation	Gratuit
Applications notables	Snapchat Strava Le Monde	OSM	Google Map, Waze...	Foursquare, Qwant, FlightGear Flight Simulator
Libre	Non	Oui	Non	Oui

Conclusion

Cette veille technologique nous a permis de nous projeter plus facilement dans le projet. Après avoir effectué de nombreux tests avec différentes technologies, nous pouvons déjà dire que nous n'utiliserons pas NativeScript qui n'est pas adapté à notre projet. Ensuite, React-Native est très intéressant, car il propose le cross-platform, mais il est très compliqué d'implémenter des cartes (hors MapView). Enfin Android Native limite les possibilités en restreignant l'application à un seul système d'exploitation, mais il est très facile de le prendre en main, et le système d'exploitation Android est le plus répandu sur le marché. De plus, nous avons réussi très rapidement à implémenter une carte en Android Native.