h e g

Haute école de gestion
Genève

Développement d'une application mobile d'échange de parking, quel Framework choisir ?

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Loïc Schupbach

Conseiller au travail de Bachelor :

Rolf Hauri, professeur HES

Haute École de Gestion de Genève, 1 septembre 2018

Haute École de Gestion de Genève (HEG-GE)

Filière IG

Hes·so GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre « Bachelor of Science HES-SO en Informatique de gestion ».

L'étudiant atteste que son travail a été vérifié par un logiciel de détection de plagiat.

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Thônex, le 01.09.2018

Loïc Schupbach

[SIGNATURE]

Remerciements

Si vous avez des remerciements à formuler, à l'entreprise ou à toute autre personne qui a pu vous aider dans la réalisation du travail.

Les remerciements sont rédigés dans le style « **Corps de texte** ».

Résumé

Il est toujours difficile, lorsque l'on a une idée d'application, de savoir quels outils utiliser pour concrétiser cette idée. De nos jours, il est devenu essentiel de fournir notre application sur les plus grandes plateformes existantes (c-.à-d. IOS et Android) mais il n'est pas concevable d'adapter plusieurs fois l'application, une pour chaque plateforme.

C'est là qu'apparait les Frameworks de développement mobile. Ils permettent de créer une seule application dans un langage commun unique (souvent Javascript / HTML) puis de transpiler ces applications dans un langage qui permet de la lancer sur chacun des système mobiles précédemment cités.

Ces Frameworks sont très pratiques mais nécessitent une certaine connaissance de leurs fonctionnement pour pouvoir faire ce que nous voulons. Malheureusement, vu le nombre de Frameworks existant, il est impossible de tous les maitriser et de les utiliser correctement. Il faut donc faire un choix et développer entièrement notre projet sur le Framework sélectionné.

En expliquant les avantages et désavantages des Frameworks les plus connus / utilisés, cela permettra de mieux comprendre leurs fonctionnements et de choisir le plus adéquat à chaque projet.

Table des matières

Liste des tableaux

Liste des figures

1. Introduction

2. Etude des Frameworks existants

Le marché actuel propose pleins de Frameworks différents permettant de créer des applications mobiles. La plupart sont « cross-platform », c'est-à-dire qu'ils offrent la possibilité de créer un seul projet qui peut ensuite être utilisé sur plusieurs systèmes d'exploitation mobiles (IOS et Android, le plus souvent).

Pour permettre ce système « cross-platform », ces Frameworks n'utilisent pas le langage natif du téléphone (Java pour Android et Objective-C pour IOS) car il serait alors nécessaire de créer un projet pour chaque langage de programmation et donc pour chaque système d'exploitation mobile. Les Frameworks utilisent un langage différent des langages natifs (souvent Javascript pour la partie logique et HTML pour la partie visuel). Évidemment, ces langages ne sont pas compris par les téléphone mobiles et doivent être exécuté dans un environnement spécifique : le « Runtime Environnement ». Ces environnement encapsulent les projets « cross-platform » dans une « boite » qui comprend le langage du Framework. Cette boite offrent des entrées et des sorties permettant ensuite au Framework de communiquer directement avec les composants du téléphone (RAM, CPU, GPU, etc.). L'environnement de runtime peut différer selon les Frameworks. La partie visuel, quand à elle est le plus souvent affichée dans une WebView. Une WebView est un fenêtre prenant toute la place de l'écran et qui comprend les langages Web (HTML / CSS). Cette dernière est donc affichée à l'écran et la partie visuel de notre application est par la suite « collée » sur cette WebView.

Chaque langage possède son propre environnement de runtime. La WebView quant à elle est propre à chaque système d'exploitation mobile. Chaque plateforme propose nativement une WebView différente plus ou moins performante.

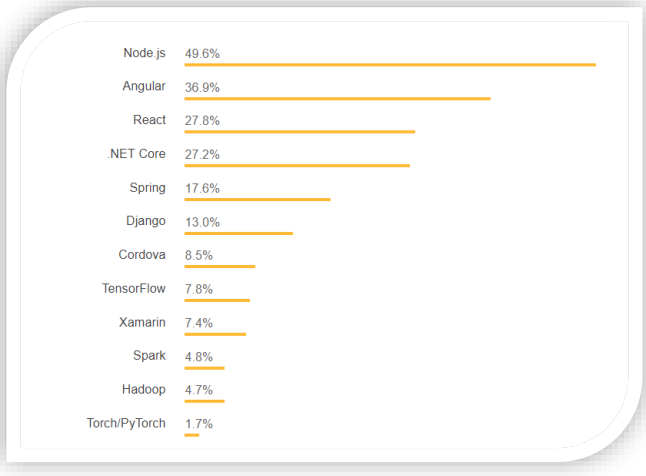
Il faut faire attention de bien vérifier si les fonctionnalités visuels que nous mettons en place sont compatibles avec la WebView native du système d'exploitation cible.

Comme expliqué précèdemment, il existe un grand nombre de Frameworks différents offrant la possibilité de créer une application « cross.platform ». Pour définir lequel est le plus adapté à la création d'une application d'échange de place de parking, je vais expliquer les avantages et les désavantages des 3 Frameworks les plus connus et utilisés.

Pour faire ce choix, je me suis basé sur un sondage réalisée par le site internet « Stack Overflow » (Stack Overflow, 2018).

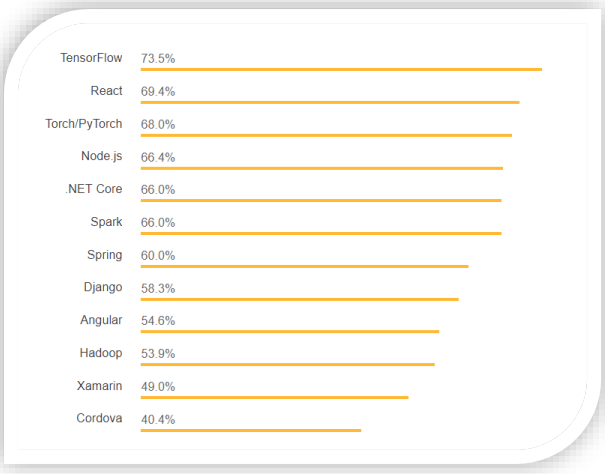
Ce sondage, réalisé pendant le mois de janvier 2018 a été répondu par plus de 67 000 développeurs jugés « qualifiés ». Les graphiques suivants représentent l'utilisation des Frameworks :

Figure 1 - Frameworks, Librairies et Outils les plus populaires en 2018



(Stack Overflow, 2018)

Figure 2 - Frameworks, Librairies et Outils les plus aimés en 2018



(Stack Overflow, 2018)

Ces deux graphiques représentent très bien la situation actuelle. Sur tous les Frameworks présents, 4 sont des Frameworks permettant le développement mobiles cross-platform triés par ordre de popularité :

- React (et sa version de développement mobile React Native)
- Angular (et sa version de développement mobile Ionic)
- Xamarin
- Cordova

Ces quatre Frameworks sont les plus populaire auprès de la communauté des développeurs en 2018. Je vais donc expliquer le fonctionnement de « React Native », « Ionic » et « Xamarin »

2.1 Ionic

Ionic est un Framework utilisant les technologies WEB (Javascript et HTML) permettant la création de site Web et d'application mobile cross-platform. Ce Framework est une surcouche du Framework AngularJs, Angular2 ou Angular4 (dépendant de la version d'Ionic) permettant le binding de donnée (explication détaillée à la section AngularJs).

Il offre des composants graphiques déjà créés qu'AngularJS ne propose pas. De plus, l'architecture des dossiers et des fichiers est beaucoup plus structurée. Un système de thème a aussi été ajouté, permettant de créer des règles de styles générales à tout le projet.

L'objectif de ce Framework est d'offrir un développement court, efficace et ne nécessitant pas de grande connaissances dans le domaine. Le développement WEB étant souvent la première chose que tout informaticien apprend, il est très facile de se lancer dans la création d'une application utilisant le Framework Ionic.

Ionic ne peut donc pas être utilisé seul. Il a besoin de fonctionner en corrélation avec AngularJs (pour le binding de données) ainsi qu'Apache Cordova (pour transformer notre projet en un package que le téléphone mobile peut comprendre).

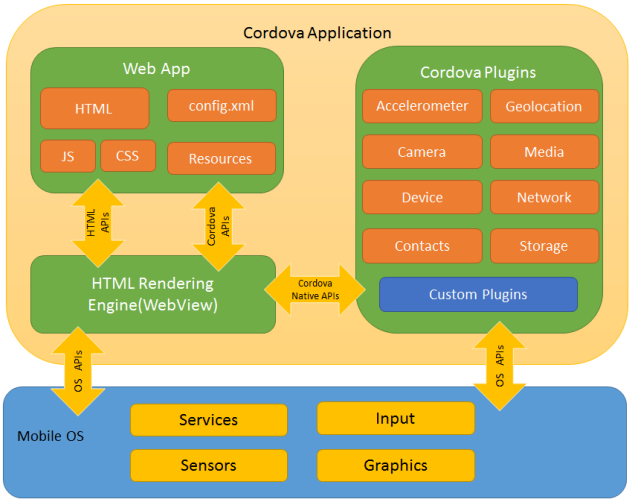
2.1.1 Architecture

Comme décrit précédemment, Ionic est une surcouche du Framework AngularJs qui, lui-même, utilise le Framework Apache Cordova pour la création de l'APK. Le fonctionnement de ces trois Frameworks vont être expliqués si après.

2.1.1.1 Apache Cordova

La création de l'APK et le test de notre application sur le téléphone mobile est une étape essentiel du développement. Il existe plusieurs façons de créer un APK. Apache Cordova propose d'encapsuler facilement notre application dans une package lisible par les systèmes d'exploitation mobiles et offre des accès aux composants natifs du téléphone (appareil photo, accéléromètre, etc.) via une collection de plugins (qui font office d'interface) intégrés à Apache Cordova.

Figure 3 - Architecture d'Apache Cordova



(Cordova, 2012)

Le schéma ci-dessus représente bien l'architecture de ce Framework.

Notre application, qui peut être créée avec n'importe quel Framework compatible, sera encapsulé dans la « boite » Web App. Cette dernière discute avec la WebView grâce à des systèmes d'Apis IN / OUT pour afficher le rendu visuel. De son côté, la WebView à un accès aux composants natifs du téléphone via les plugins d'Apache Cordova. Il est donc possible de récupérer des informations natives et de les traiter. Ces informations seront récupérées / calculées sur les composants natifs qui seront ensuite envoyées aux plugins d'Apache Cordova. Ceux-ci vont transpiler les données en quelque chose

de lisible par notre application et les envoyer à la WebView qui, elle-même, transmettra les informations à la partie logique de notre application.

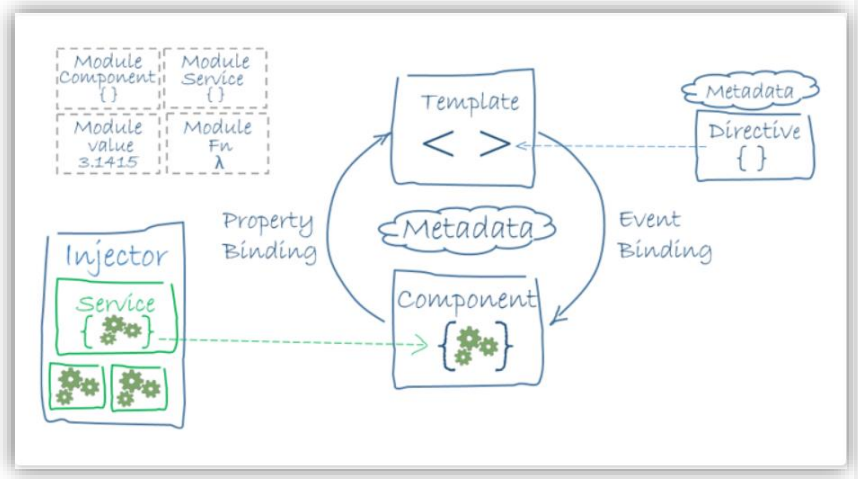
Apache Cordova utilise l'environnement de runtime natif JavaScript de chaque système d'exploitation pour fonctionner l'application et créer les apks.

2.1.1.2 AngularJs

AngularJs est un Framework offrant la possibilité de créer des pages WEB et des applications mobiles de manière simplifié grâce à des systèmes de déclaration de modèles, d'injection de dépendances et de binding de données.

AngularJs va relier de manière simplifié les données de la partie logique à l'affichage et inversement.

Figure 4 - Architecture d'AngularJs



(Angular, 2009a)

On peut voir sur ce schéma le fonctionnement d'AngularJs. Chaque objet logique de notre site Web / application (page, liste, bouton personnalisé, etc.) va être enregistrée en tant que « composant ». Un composant désigne la partie logique d'une page et est donc reliée à un modèle écrit en HTML / CSS. Cette liaison permet de faire du « Property Binding » et de « l'Event Binding ».

Le Property Binding permet de mettre à jour automatiquement les données de notre application. C'est-à-dire que quand une donnée est affichée sur le modèle et que la variable représentant cette donnée change dans le composant, l'affichage de cette

donnée va automatiquement être mis-à-jour dans le modèle. Ce système fonctionne aussi en sens inverse. Lorsque l'utilisateur va écrire du texte dans un champ de texte, il est possible d'enregistrer le contenu de ce champ en temps réel dans une variable du composant.

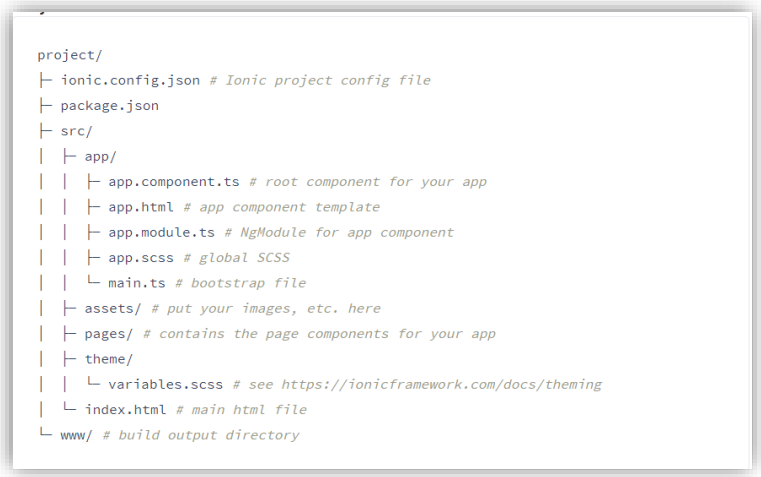
L'Event Binding quant à lui permet de déclencher une certaine méthode de la partie logique quand l'utilisateur déclenche une certaine action de la partie visuel de notre application.

AngularJs propose aussi un système d'injection de dépendance. Cela permet d'ajouter dans un composant un lien sur un autre composant ou un autre service (Accéléromètre, Network, etc.) et de pouvoir accéder aux données et fonctions de ce dernier.

2.1.1.3 Ionic

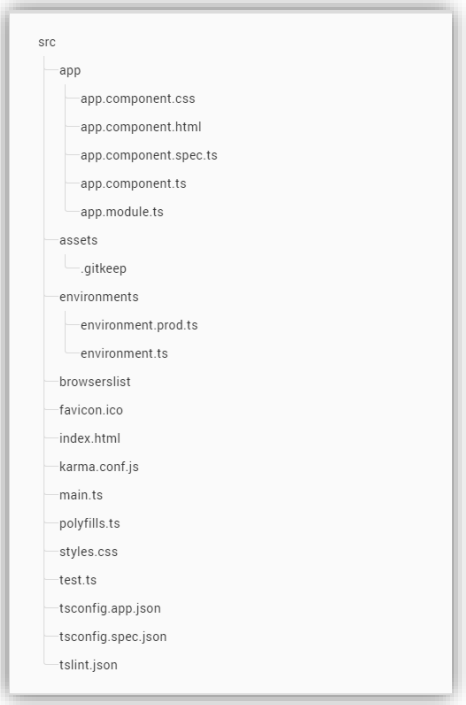
Ionic n'est qu'une surcouche à AngularJs. Il est là uniquement pour simplifier la structuration de notre projet et offrir une stylisation de l'application plus poussée qu'AngularJs.

Figure 5 - Structure de fichiers d'un projet Ionic



(Drifty, 2017)

Figure 6 - Structure de fichiers d'un projet AngularJS



(Angular, 2009b)

On peut constater que la structure de dossier d'une projet Ionic est clairement plus structurée que celle d'un projet AngularJS. Le dossier « src » ne contient que les éléments à modifier contrairement au dossier « src » d'AngularJs qui contient plusieurs fichiers de configuration. De plus, le dossier « src » d'Ionic contient des sous-dossier structurant le projet avec des pages, des assets et différents thèmes.

De plus, Ionic à inclue dans son Framework plusieurs composants graphiques responsive ainsi que cross-platform facilement utilisables.

Il est aussi possible d'utiliser le langage TypeScript à la place du langage JavaScript pour le développement avec le Framework Ionic. TypeScript est un langage basé sur le JavaScript (une sorte de surcouche) offrant le typage, les interfaces, les génériques, les namespaces ainsi que un contrôle des erreurs plus poussé.

2.1.2 Avantages

Ionic est un Framework récent permettant le développement simple et efficace d'application cross-platform. C'est dans sa structuration et son fonctionnement simplifié que réside sa plus grande force.

Le développement cross-platform devenant la norme dans la création d'application, cela en fait un atout majeur pour la Framework Ionic. C'est tout à fait compréhensible quand on sait que le développement cross-platform permet :

- Une réduction des coûts de développement et de maintenance
- Des connaissances nécessaires réduites
- La création de code réutilisable
- Une migration et maintenabilité simplifié

Commenté [SL1]: Peut être à mettre dans l'intro

Comme Ionic utilise le JavaScript ou TypeScript ainsi que l'HTML pour la partie visuel, cela permet à un grand nombre de développeurs d'utiliser le Framework. En effet, les langages WEB sont les premiers langages que tout développeur apprend car ils sont très simple de compréhension. Il est très facile de trouver de l'aide ou des tutoriels sur internet.

Un autre grand atout d'Ionic est sa portabilité. Il est effectivement possible de créer un site web responsive utilisant les composants donnés par Ionic et ensuite de transformer ce site web en une application en seulement quelques modifications.

Ionic offre en effet une portabilité tellement grande avec les composants qu'il offre qu'il est même possible de créer un seul code pour une application Android, une application IOS et un site web (pour autant qu'il n'y ait pas de PHP). Seul quelques pourcents du code doit être adapté aux plateformes pour permettre sont fonctionnement (cela concerne surtout les plugins qui ne fonctionnement pas de la même manière sur toutes les plateformes).

Un point important d'Ionic est son système de débugge. Comme l'application peut être lancée dans un navigateur, il est possible d'afficher des informations dans la console et de modifier l'interface utilisateur en temps réel.

2.1.3 Faiblesses

Actuellement, l'expérience utilisateur est au centre de toutes les préoccupations dans l'informatique. Que ce soit pour des clients lourds, riches, léger, des applications de bureau ou des applications mobiles, il est important que l'utilisateur ait un ressenti positif de l'application.

Un point qu'Ionic ne parvient pas toujours à combler. En effet, comme Ionic est en fait composé de 3 couches (Ionic ➔Angular ➔ Apache Cordova), les performances de l'application sont grandement diminuées. En cas d'utilisation gourmande de l'application, le nombre d'images par seconde va grandement chuter et l'utilisateur aura l'impression que l'application fonctionne au ralenti et qu'elle n'est pas fluide.

Un autre gros problème de ce Framework est la compatibilité entre les différents « sous-frameworks » qui le compose. Il est assez fréquent que des changements soient faits au niveau d'Apache Cordova (surtout au niveau des plugins) et que ces changements cassent le fonctionnement actuel de notre application. De plus, certains plugins ne sont pas fonctionnels et ne peuvent pas être utilisé lors du passage en production de l'application. (Vincent, 2016)

2.2 Xamarin

Xamarin est un Framework dédié au développement d'application mobile « cross-platform » (Android, IOS et pour ceux qui veulent, Windows Phone). Contrairement à la plupart des autres Frameworks de développement cross-platform, Xamarin est basé sur le Framework .NET et utilise donc le langage C# et pas JavaScript.

Contrairement aux pensées, Xamarin n'est pas un Framework interprété mais compilé nativement, ce qui lui donne la possibilité de créer des applications très performantes avec un design proche d'un design natif. (Altexsoft, 2018a)

Ce Framework offre la possibilité de créer un seul code source logique codé en C# et transférable sur plusieurs OS cibles. En ce qui concerne la partie visuel de l'application, il existe deux options :

- Xamarin.Android et Xamarin.IOS (ou encore Xamarin.Windows)
- Utilisation de Xamarin.Forms permettant de créer un seul design « cross-platform » pour Android, IOS et Winows Phone.

La première option est bien évidemment la plus performante. En effet, chaque OS cible aura son propre design créer en langage natif et sera donc très performant. De plus, créer un design en langage natif nous permet de rester le plus fidèle possible aux normes des interfaces utilisateurs proposées par chaque plateforme. Le désavantage réside dans l'obligation d'écrire plusieurs fois le même code pour chaque OS ciblé.

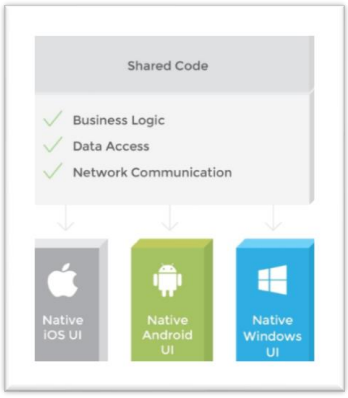
La deuxième option, Xamarin.Forms, facilite grandement le développement et permet d'avoir un design unifié entre les différentes plateformes mais pose un problème de taille. Les performances seront grandement diminuées pour certaines opérations et la création de design compliqué sera impossible. (Altexsoft, 2018b)

2.2.1 Architecture

Comme expliqué précédemment, Xamarin offre la possibilité de créer un seul code logique et une ou plusieurs interfaces utilisateurs.

L'image ci-contre représente la structure simplifiée d'une application Xamarin. Toute la partie logique et accès aux composants natifs du téléphone se trouve dans un bloc de codé partagé et unique pour toutes les plateformes. Le seul code non partagé est le design de l'interface, pour autant qu'il ne soit pas créé avec Xamarin.Forms.

Figure 7 - Structure simplifié d'une application Xamarin

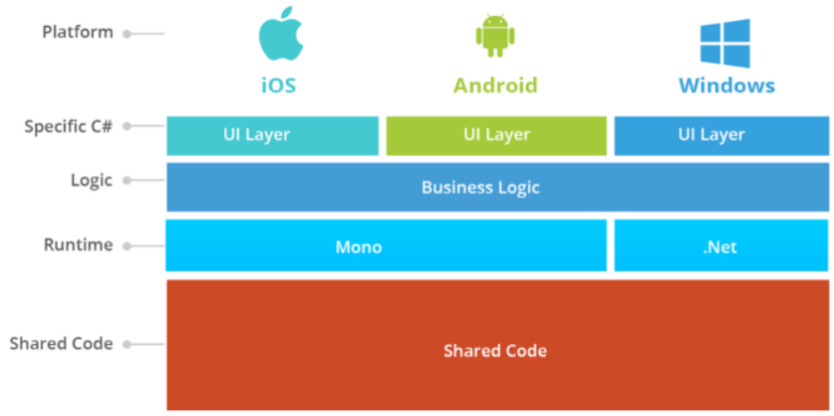


(Altexsoft, 2018a)

Tout comme pour Ionic, les systèmes d'exploitations mobiles ne comprennent pas le langage C# et doivent donc avoir un environnement de runtime spécifique. Étant donné que ces plateformes n'ont pas cette environnement de manière native, Xamarin a pris la décision de l'ajouter dans le code source de l'application ainsi. De cette manière, quel que soit le système d'exploitation mobile sur lequel est lancé l'application, elle pourra fonctionner.

Le schéma ci-dessous représente l'architecture complète d'une application Xamarin, incluant l'environnement de runtime, le code partagé et les interfaces utilisateurs.

Figure 8 - Architecture de Xamarin



(Nexgendesign, Non daté)

En créant des interfaces différenciées pour chaque plateforme cible, les composants de ces interfaces utilisateurs vont être des composants natifs. Le seul code non natif sera alors le code partagé en C# qui tourne dans sont propre environnement de runtime créer pour avoir des performances maximales. Si l'on décide d'utiliser Xamarin.Forms, l'interface utilisateur ne sera alors plus composée de composants natifs et perdra alors grandement en performance. Il sera néanmoins possible de partagé de 90% à 100% de code en utilisant cette option.

2.2.2 Avantages

Xamarin est un Framework proposant une approche totalement différentes des autres Frameworks de développement mobiles. Le fait qu'il utilise le langage C#, un langage fortement typé, y ajoute une couche de protection contre les éventuels bugs générés par un code approximatif.

Comme expliqué précédemment, les performances d'une application créée grâce à Xamarin est très proche des performances d'une application native. Cela en fait un atout majeur pour ce Framework. En effet, il est très important d'avoir une application rapide et fluide car les utilisateurs aiment de moins en moins attendre qu'une application charge.

La possibilité qu'offre Xamarin de créer l'interface utilisateur de 2 manières différentes est aussi un atout pour le Framework. Si l'application développé n'utilise pas beaucoup de ressources, il est possible de créer une seule interface utilisateur partagée sur

chaque plateforme. Mais si l'application est une application lourde, Xamarin donne aussi la possibilité de créer des interfaces utilisateurs natives pour garder des performances proches du natif. Il est aussi possible d'utiliser Xamarin.Forms pour créer des prototypes de notre application est de les montrer aux clients. Le client donne sont avis et lorsque l'interface sera définitive, elle peut être réécrite pour chaque OS cible et donc augmenter les performances de l'application.

2.2.3 Faiblesses

L'utilisation de Xamarin oblige le développeur a se former à deux nouveaux langages : le C# et le langage qu'utilise Xamarin pour créer ces interfaces Xamarin.Forms, le XAML.

Commenté [SL2]: Expliquer d'où vient le xaml

La communauté utilisant le Framework Xamarin est aussi bien plus faible que celle utilisant les Frameworks utilisant les technologies WEB. Il sera donc plus difficile de trouver des réponses à nos questions lorsque nous aurons un problème.

Comme expliqué précédemment, l'utilisation de Xamarin.Forms réduit grandement les performances d'une application créée avec Xamarin. Il n'est donc pas possible de créer des interfaces utilisateurs compliquées ou contenant des graphismes lourds.

La taille des applications est aussi une inconvénient de Xamarin par rapport à ces concurrents. Comme une application Xamarin n'est pas comprise par le système d'opération du téléphone mobile, l'application emporte avec elle d'autres outils pour la faire fonctionner (runtime, assemblage de libraires de classes de base, etc.)

Figure 9 - Taille d'une application "Hello World" avec Xamarin

(Altexsoft, 2018a)

Xamarin ayant été acheté par Microsoft, ce dernier a rendu l'utilisation de cet outil gratuit mais l'a aussi intégré à son IDE : Visual Studio.

Il est donc nécessaire d'obtenir Visual Studio pour créer une application Xamarin. Heureusement, une version gratuite de Visual Studio existe mais elle n'offre pas toutes les fonctionnalités qui nous permettent de travailler correctement sur notre projet.

Pour profiter pleinement de Visual Studio et de Xamarin, il est essentiel d'acquérir une licence Visual Studio qui coute extrêmement cher (plus de 1000 dollars)

Commenté [SL3]: Ne pas donner mon avis sur le prix de la licence

2.3 React Native

React Native est un Framework de développement mobile basé sur le Framework ReactJs pour créer et composer des interfaces riches à partir de composants déclaratifs codés en JavaScript.

React Native se distingue des autres Frameworks de développement mobile utilisant les technologies WEB car il ne crée par une interface tournant dans une WebView (comme Ionic) mais en proposant une système semblable à Xamarin qui permet la création d'interfaces utilisateurs natives. React Native va, en quelque sorte, encapsuler les composants natifs dans des composants compréhensibles pour le Framework. (React Native, 2018)

Ce système impose à React Native de créer les interfaces dans un langage différent du HTML. C'est pour cela que ce Framework utilise un langage différent du HTML pour la création de ces interfaces utilisateurs. Ce langage ressemble fortement à l'HTML mais ne représente pas les mêmes objets. Chaque balise est une encapsulation d'une élément natif du téléphone. Ce système a été choisi car React par du principe que la logique du rendu est fortement couplée à la logique d'affichage des interfaces utilisateurs : comment les évènements sont traités, comment l'état de l'application change au fil du temps et comment les données sont préparées pour l'affichage. (React, 2018)

2.3.1 Architecture

React Native à un fonctionnement très spécifique à lui-même. En effet, un Framework basique tel que Ionic va avoir ses propres composants qui seront connectés via des APIs aux composants natifs du téléphone. Une multitude de couches se superposent alors pour offrir l'accès aux fonctionnalités du téléphone.

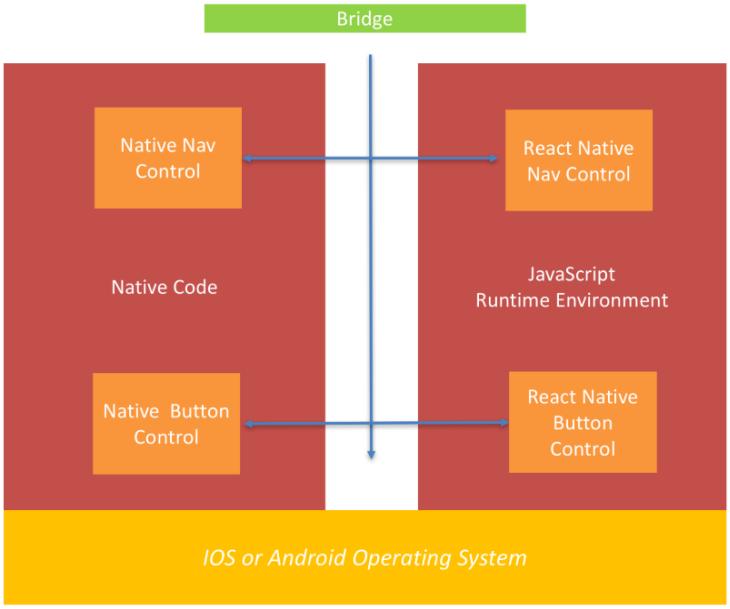
React Native a été construit d'une toute autre manière. Chaque composant est directement branché au composant natif dont il fait la référence. Aucune couche supplémentaire n'est nécessaire pour communiquer entre les composants natifs et ceux offerts par React Native.

Il est néanmoins important de comprendre que l'on n'interagit pas immédiatement avec les composants natifs du téléphone. React Native reste un Framework utilisant le

langage JavaScript et ne peut donc pas directement être lancé sur un téléphone Android ou IOS.

C'est pour cela que React Native utilise un environnement de runtime spéciale à Javascript.

Figure 10 - Architecture de React Native

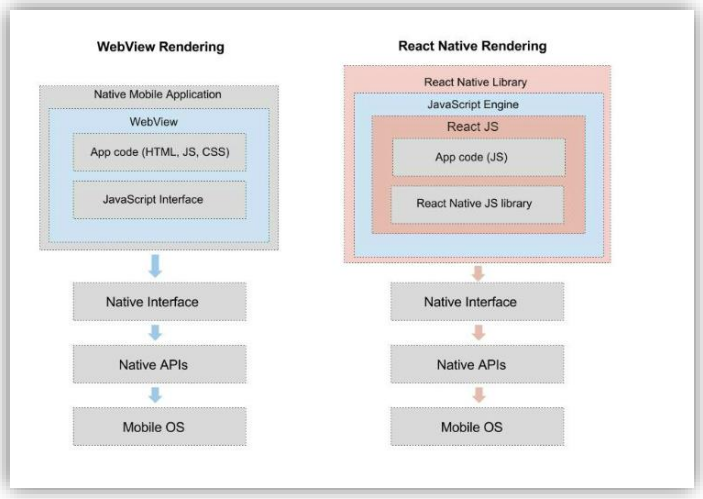


(Heard, 2017)

L'application est alors lancée dans cet environnement, ce qui permet d'accéder au langage JavaScript et de le faire tourner. Des ponts sont ensuite créés entre les composants offerts par React Native et les composants natifs du téléphone. Seuls les composants sont donc connectés au téléphone. Le reste de l'application communique ensuite uniquement via les composants React Native qui sont eux même connectés aux composants natifs du téléphone.

L'interface créée via le Framework React Native n'est donc pas une interface « HTML/CSS ». L'interface est semi-native et est lancée directement sur la couche native du téléphone. React Native n'a pas besoin d'une WebView pour afficher l'application. La disparition de la couche WebView ainsi que de la couche d'accès aux composants augmente grandement les performances d'une application React Native.

Figure 11 - Différence WebView et React Native



(Altexsoft, 2018)

On peut voir sur cette image que React Native ne dispose pas de WebView. L'interface utilisateur, qui est dans la WebView pour une application JavaScript / HTML, est ici directement dans la boite « Native interface ».

2.3.2 Avantages

Le point important à retenir du Framework React Native est la manière de lier le Framework aux composants Natifs du téléphone. Cela enduit des performances accrues, très proches d'une application native tout en gardant un développement cross-plateforme (un seul langage pour chaque plateforme). Evidemment, chaque système d'exploitation a ces spécificités et il sera nécessaire d'adapter légèrement son code pour chaque plateforme ciblée. De 90% à 100% du code créer pourra être partagé entre toutes les plateformes.

Il est possible également de créer ses propres composants natifs (en Objective-C ou Java) et de les implémenter directement dans le Framework React Native. Comme le Framework utilise des ponts pour communiquer, il est très simple de rajouter ces propres composants.

Évidemment, toute la partie logique de l'application étant codée en JavaScript, aucun apprentissage devra être fait pour apprendre ce langage qui est très connu.

Un point important de React Native est son système de débugg. Il est possible de lancer son application directement sur son téléphone est de pouvoir débugger en tant réel les problèmes depuis son ordinateur. Ce système est très semblable à celui d'Ionic.

2.3.3 Faiblesses

Comme React Native est basé sur le Framework ReactJS, une connaissance de ce dernier est conseillée.

Comme React Native créer une encapsulation des composants natifs du téléphone, il ne propose qu'une petite collection de composants. Pour créer des éléments compliqués, il faudra faire un empilement de ces composants qui peut vite se révéler compliqué.

Commenté [SL4]: Qu'est-ce que ca signifie

Le plus gros désavantage réside dans l'ajout de composants externes / tierces à notre application. La communauté à créer pleins de composants et modules augmentant les possibilités de création de React Native. Malheureusement, ces composants ne sont pas toujours mis à jour et souvent, certains composants ne vont pas pouvoir être implémenté à notre projet.

Commenté [SL5]: Source !

2.4 Comparaison des Frameworks

Chaque Framework comporte des avantages et des désavantages. Une application créée via le Framework Ionic pourra être très personnalisée mais sera vite limitée au niveau des performances. Xamarin est un juste milieu avec des bonnes performances, un accès aux composants natifs simplifiés mais une création d'interface limitée et l'obligation de connaitre le langage C#. React Native propose un mélange des 2 autres Frameworks avec des performances proches des performances natives, un développement simplifié offrant le cross-plateforme mais une personnalisation moyenne.

Les tableaux ci-dessous représentent les avantages et désavantages de chaque Framework au niveau des performances, du développement et des aspects techniques.

2.4.1 Comparaison

Les quelques graphiques et tableaux ci-dessous résument les points importants :

Tableau 1 – Résumé simple des Frameworks

	Ionic	Xamarin	React Native
Langages	JavaScript / TypeScript et HTML	C# et XAML	JavaScript et composants natifs
Accès aux composants natifs direct	Non	Oui	Oui
Idées générales	Utilise les technologies Web pour une portabilité maximale	Reste le plus proche possible du natif	Approche fonctionnelle : l'interface est une fonction de l'état

(Cruxlab, 2017)

Tableau 2 - Résumé des performances des Frameworks

	Plateformes	**Ionic**	**Xamarin**	**React Native**
Compilation du code	IOS	Interpréteur JIT avec plugins	AOT	Interpréteur
	Android	JIT	JIT	JIT
Support du 64-bit	IOS	Oui	Oui	Surement (Apple force le 64 bits)
	Android	Aucune informations	Oui	Non (prévu en aout 2019)
UI performance	IOS et Android	HTML	Composants natifs	Composants natifs

(Cruxlab, 2017)

Question performance, Xamarin remporte haut la main la première place. Etant donné qu'il emporte son propre environnement de runtime et que l'interface utilisateur peut être écrite en code natif, il est bien plus rapide. En 2ème place, React Native. Bien que le 64 bits ne soit pas supporté, il reste plus performant car il utilise les composants natifs du système d'exploitation. Le fonctionnement d'Ionic le place bon dernier au niveau des performances.

Commenté [SL6]: Mettre date de quand ca va être supporté (juin 2019 il me semble) => deadline google

Tableau 3 - Résumé des aspects de développement des Frameworks

	Plateformes	**Outils officiels**	**Ionic**	**Xamarin**	**React Native**
Lancement manuel		Oui	Oui	Oui	Oui
Lancement automatique		Non	Oui	Non	Oui
Permutation du code à chaud	IOS	Non	Non	Non	Oui
	Android	Oui	Non	Non	Oui
Permutation du code à froid	IOS	Non	Oui	Non	Oui
	Android	Oui	Oui	Oui	Oui
Développement dans le navigateur		Non	Oui (aspect différent !)	Non	Non
Mise-à-jour instantanées		Non	Oui	Non	Oui
Débugging en temps réel		Log et break point	Console et modification de l'UI en temps réel	Log et break point	Console, modification de l'UI en temps réel et affichage des state / props

(Cruxlab, 2017)

Au niveau des aides de développement, React Native passe premier et de loin. En effet, Xamarin ne change en rien du développement natif car il n'ajoute aucune aide au développement. Au niveau d'Ionic, seulement la permutation du code à froid et la possibilité de visualiser son application dans un navigateur ont été ajoutés. Il faut faire néanmoins attention à cette deuxième option car l'application n'aura pas exactement le même design que sur le téléphone et les fonctionnalités natives n'y sont pas prisent en compte (géolocalisation, accéléromètre, appareil photo, etc.). React Native, de son coté, ajoute énormément d'aide au développement qui facilite grandement la création de son application.

Tableau 4 - Résumé des aspects techniques des Frameworks

		Ionic	Xamarin	React Native
Capacités de liaison de code	Utiliser les classes Java, Objective-C et @objc Swift à partir d'un Framework multi-plateforme	Nécessité d'avoir un adaptateur	Toutes les libraires binaires	Nécessité d'avoir un adaptateur
	Utilisation de classes multi-plateformes à partir d'un code traditionnel	Non	Compliqué mais faisable	Oui
Liaison a des bibliothèques standards	Set de fonctionnalités supportés	Partiel	Complet	Partiel
	Type de liaison	Couche d'abstraction	One-to-one	Couche d'abstraction
UI	Set complet de composants disponible	Non	Oui	Non
	Design ressemblant au design natif	Oui	Partiel	Partiel
Exécution du code en tâche de fond	IOS	Uniquement prévention de sommeil du téléphone	Semblable à du code natif	Non
	Android	Uniquement prévention de sommeil du téléphone	Semblable à du code natif	Headless JS

(Cruxlab, 2017)

Au niveau des aspects techniques, les Frameworks sont assez semblable, Xamarin est légèrement au-dessus car il supporte mieux les composants natifs et l'ajout de libraire tierces. En ce qui concerne React Native et Ionic, la différence n'est pas notable.

2.4.2 Résumé

Tableau 5 - Résumé des Frameworks

	Ionic	**Xamarin**	**React Native**
Performances	Mauvaises / Moyennes	Bonnes	Bonnes
Look natif	Oui	Partiel	Partiel
Outils de développement modernes	Partiel	Non	Oui
Support des fonctionnalités natives	Partiel	Oui	Partiel
Connaissances nécessaires	Faibles	Hautes	Moyennes

Inspiré de (Cruxlab, 2017)

On peut voir ici que chaque Framework à ses avantages et ses inconvénient. Xamarin est plus destiné à des développeurs confirmé qui connaissance bien le langage C# ainsi que pour des projets nécessitant des performances extrêmement accrues. React Native est un bon milieu car il propose de bonnes performances ainsi que des aides au développement poussées tout en ne nécessitant pas trop de connaissances supplémentaires aux technologies WEB. Ionic, quand à lui, est clairement le Framework le plus simple a prendre en main mais aussi le moins performant.

3. Etude de l'application

L'application vise à régler le problème du manque de place de parking dans la ville de Genève. Elle permettra à toute personne disposant d'une place de parking privée de la mettre en location et toute personne possédant l'application pourra alors louer cette place pour un certain temps.

3.1 Fonctionnalités de l'application

L'application offrira les fonctionnalités suivantes :

- Enregistrement au sein de l'application
- Suppression de son compte
- Modification de son compte
- Connexion à son compte
- Créer une place de parking
- Modifier une place de parking
- Supprimer une place de parking
- Ajouter un horaire à une place de parking
- Supprimer un horaire
- Modifier un horaire
- Visualiser toutes les places de parking actuellement disponibles dans un cercle de recherche sur une carte
- Louer une place de parking pour un certain horaire
- Supprime une location
- Voir toutes ses places et réservations

3.2 Besoins techniques de l'application

Cette application vise a être utilisée à tout temps. Il est donc essentiel qu'elle soit rapide, fluide et simple d'utilisation.

Après s'être connecté, une carte s'affichera sur l'écran de l'utilisateur. Il pourra alors positionner une cercle de recherche et visualiser toutes les places disponibles dans ce cercles. L'utilisateur pourra alors comparer les différentes places et en réserver une pour un certain horaire.

L'application sera disponible sur le Play Store d'Android puis par la suite sur le l'App Store d'IOS.

Comme l'intérêt de l'application réside dans la réservation rapide et instinctive d'une place de parking, il faut que les performances de l'application soient le plus proche des performances d'une application native.

4. Choix du Framework

Les trois Frameworks que je vous ai présentés sont très intéressants. Chacun a ses avantages et désavantages.

Après avoir analysé les fonctionnalités et besoins de l'application, j'ai estimé que React Native était le Framework le plus adapté. Il offre des performances accrues ainsi que des accès aux fonctionnalités natives complètes. De plus, beaucoup de libraires peuvent y être ajoutés pour offrir une expérience utilisateurs améliorée.

4.1 Analyse détaillé du choix

Pour sélectionner quel Framework est le plus approprié dans mon cas, j'ai utilisé une analyse multicritère basée sur certains critères définis grâce à une matrice de préférence.

Voici les différents critères sélectionnés :

1) Connaissance du langage associé
2) Accès aux fonctionnalités natives du téléphone
3) Framework bien documenté
4) Performances
5) Maintenabilité

Tableau 6 - Matrice de préférence des critères de choix du Framework

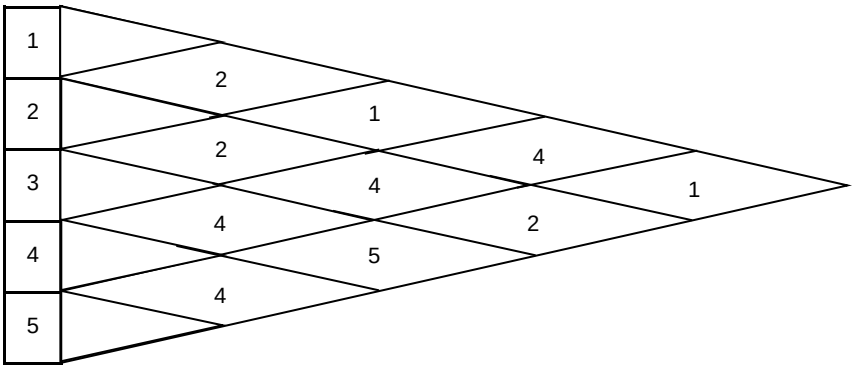


Tableau 7 - Pondération des critères de choix du Framework

Numéro	Critère	Pondération
1	Connaissance du langage associé	2
2	Accès aux fonctionnalités natives du téléphone	3
3	Framework bien documenté	0
4	Performances	4
5	Maintenabilité	1

Évidemment, comme mon travail de Bachelor est un mix entre une explication des Frameworks et la création d'une application, il est essentiel de choisir un Framework proposant des performances importantes ainsi qu'un accès aux fonctionnalités natives facilité. Néanmoins, ne disposant pas d'un temps de travail illimité, il est tout aussi important que le langage associé au Framework soit un langage que je connaisse. Je ne dispose pas du temps pour apprendre les spécificités du Framework que je choisis et aussi un nouveau langage.

J'ai, pour un premier temps, décidé de créer mon application sur la plateforme Android pour une question de simplicité et de couts. Mais il est important de savoir que j'envisage, à terme, de fournir mon application sur Android et IOS. C'est pour cela que tous les Frameworks proposés sont cross-platform et donc que ce critère ne rentre pas en compte dans le choix du Framework

Tableau 8 - Analyse multicritère du choix du Framework

Critères		Ionic		Xamarin		React Native	
	Pondération	Points	Val. Pond	Points	Val. Pond.	Points	Val. Pond.
Connaissance du langage associé	2	10	20	3	6	8	16
Accès aux fonctionnalités natives du téléphone	3	8	24	10	30	8	24
Framework bien documenté	0	Critère supprimé					
Performances	4	5	20	9	36	9	36
Maintenabilité	1	8	8	5	5	8	8
Total		72		79		84	

4.2 Concordance avec l'application

React Native est le Framework répondant le mieux aux fonctionnalités et besoins de mon application. L'analyse multicritère le prouve. Il est précédé par Xamarin qui est seulement à 5 points de React Native. En effet, ce Framework propose de bonnes performances ainsi qu'un accès aux fonctionnalités natives complet (qui sont les critères les plus importants) mais son langage ne concorde pas avec mes connaissances. Il perd donc beaucoup de points à ce niveau-là. Effectivement, Xamarin embarque son propre environnement de runtime qui doit s'adapter à chaque mise à jour de la plateforme ciblée. Un point négatif pour Xamarin mais positif pour React Native et Ionic qui utilise un environnement de runtime JavaScript intégré à l'appareil et donc mit à jour en même temps que l'OS.

5. Implémentation de l'application

5.1 Use-case

Figure 12 - Diagramme de use-case de l'application

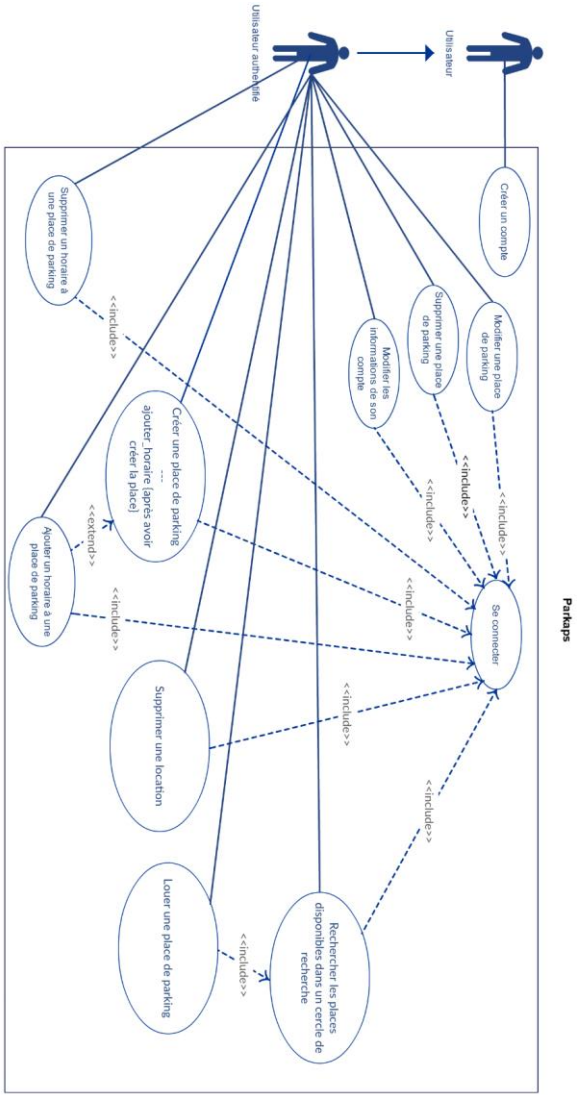


Tableau 9 - Use-case "Créer un compte"

Nom	Créer un compte
Acteurs	Utilisateur
Déclencheur	L'utilisateur clique sur le bouton « S'enregistrer »
Pré-conditions	L'utilisateur s'est rendu sur la page d'enregistrement
Flux principale	1. L'utilisateur clique sur le bouton « S'enregistrer » 2. Les informations d'enregistrement sont envoyé à l'API 3. L'API définit si les informations reçues sont correctes 4. L'API enregistre l'utilisateur et lui envoie un email de validation 5. L'utilisateur se rend sur le lien affiché dans l'email de validation 6. L'API valide le compte de l'utilisateur
Flux secondaire	3.1 Les informations sont correctes 3.1.1 Retour au flux N° 4 3.2 Les informations ne sont pas correctes 3.2.1 L'API retourne un message d'erreur 3.2.2 Les champs de texte faux sont surlignées en rouge et un message d'erreur est affiché 3.2.3 Retour au flux N° 1

Tableau 10 - Use-case "Se connecter"

Nom	Se connecter
Acteurs	Utilisateur
Déclencheur	L'utilisateur clique sur le bouton « Se connecter »
Pré-conditions	L'utilisateur s'est rendu sur la page de connexion et a créer un compte (use-case « Créer un compte »
Flux principale	1. L'utilisateur clique sur le bouton « Se connecter » 2. Les informations de connexion sont envoyées à l'API 3. L'API définit si les informations reçues sont correctes 4. L'API créé un token et le retourne 5. Le token est enregistré localement sur le téléphone 6. La page « Carte » s'affiche

Flux secondaire	3.1 Les informations reçues sont correctes 3.1.1 Retour au flux N° 4 3.2 L'utilisateur n'a pas validé son compte 3.2.1 L'API retourne un message d'erreur 3.2.2 Le message d'erreur est affiché 3.2.3 Retour au flux N° 1 3.3 Les informations de connexion ne sont pas correctes 3.3.1 L'API retourne un message d'erreur 3.3.2 Le message d'erreur est affiché 3.3.3 Retour au flux N° 1

Tableau 11 - Use-case "Créer une place de parking"

Nom	**Créer une place de parking**
Acteurs	Utilisateur authentifié
Déclencheur	L'utilisateur clique sur le bouton «Enregistrer la place de parking »
Pré-conditions	L'utilisateur s'est rendu sur la page de création d'une place de parking et est authentifié (use-case « Se connecter »
Flux principale	1. L'utilisateur clique sur le bouton « Enregistrer la place de parking » 2. Les informations sont envoyées à l'API 3. L'API définit si les informations reçues sont correctes 4. L'API créé la place de parking
Flux secondaire	3.1 Les informations reçues sont correctes 3.1.1 Retour au flux N° 4 3.2 Les informations ne sont pas correctes 3.2.1 L'API retourne un message d'erreur 3.2.2 Le message d'erreur est affiché 3.2.3 Retour au flux N° 1

Tableau 12 - Use-case "Ajouter un horaire à une place de parking"

Nom	**Ajouter un horaire à une place de parking**
Acteurs	Utilisateur authentifié
Déclencheur	L'utilisateur clique sur le bouton «Ajouter un horaire »
Pré-conditions	L'utilisateur s'est rendu sur la page ajout d'un horaire et est authentifié (use-case « Se connecter »
Flux principale	1. L'utilisateur clique sur le bouton « Enregistrer la place de parking » 2. L'utilisateur sélectionne une date de début, une date de fin 3. L'API définit si les informations reçues sont correctes 4. L'API créé l'horaire
Flux secondaire	3.1 Les informations reçues sont correctes 3.1.1 Retour au flux N° 4 3.2 Les informations ne sont pas correctes 3.2.1 L'API retourne un message d'erreur 3.2.2 Le message d'erreur est affiché 3.2.3 Retour au flux N° 1

Tableau 13 - Use-case "Supprimer un horaire d'une place de parking"

Nom	**Supprimer un horaire d'une place de parking**
Acteurs	Utilisateur authentifié
Déclencheur	L'utilisateur clique sur le bouton «Supprimer l'horaire »
Pré-conditions	L'utilisateur s'est rendu sur la page ajout d'un horaire et est authentifié (use-case « Se connecter »
Flux principale	1. L'utilisateur clique sur le bouton « Supprimer l'horaire » 2. Une confirmation est demandée 3. L'API supprime l'horaire sélectionné
Flux secondaire	2.1 L'utilisateur s'est trompé 2.1.1 Retour au Flux N° 1 2.2 L'utilisateur veut supprimer la place 2.2.1 Retour au flux N° 3

Tableau 14 - Use-case "Rechercher les places disponibles dans un cercle de recherche"

Nom	**Rechercher les places disponibles dans un cercle de recherche**
Acteurs	Utilisateur authentifié
Déclencheur	L'utilisateur clique sur le bouton «Recherche»
Pré-conditions	L'utilisateur s'est rendu sur la carte, à définit un cercle de recherche et est authentifié (use-case « Se connecter »
Flux principale	1. L'utilisateur clique sur le bouton « Rechercher» 2. L'API définit si les informations reçues sont correctes 3. Les places de parking sont affichées sur la carte à leurs positions
Flux secondaire	2.1 Les informations reçues sont correctes 2.1.1 Retour au flux N° 3 2.2 Les informations reçues ne sont pas correctes 2.2.1 L'API retourne un message d'erreur 2.2.2 Le message d'erreur est affiché 2.2.3 Retour au flux N° 1

Tableau 15 - Use-case "Louer une place de parking"

Nom	**Louer une place de parking**
Acteurs	Utilisateur authentifié
Déclencheur	L'utilisateur clique sur le bouton «Louer»
Pré-conditions	L'utilisateur a sélectionné une place sur la carte et est authentifié (use-case « Se connecter »)
Flux principale	1. L'utilisateur clique sur le bouton « Louer» 2. L'utilisateur sélectionne une date de début et une date de fin 3. L'API définit si les informations reçues sont correctes 4. L'API enregistre la location
Flux secondaire	3.1 Les informations reçues sont correctes 3.1.1 Retour au flux N° 4 3.2 Les informations reçues ne sont pas correctes 3.2.1 L'API retourne un message d'erreur 3.2.2 Le message d'erreur est affiché 3.2.3 Retour au flux N° 1

Tableau 16 - Use-case "Supprimer une location"

Nom	**Supprimer une location**
Acteurs	Utilisateur authentifié
Déclencheur	L'utilisateur clique sur le bouton «Supprimer la location»
Pré-conditions	L'utilisateur a sélectionné une location et est authentifié (use-case « Se connecter »)
Flux principale	1. L'utilisateur clique sur le bouton « Supprimer la location» 2. Une confirmation est demandée 3. L'API supprime la location
Flux secondaire	2.1 L'utilisateur s'est trompé 2.1.1 Retour au Flux N° 1 2.2 L'utilisateur veut supprimer la location 2.2.1 Retour au flux N° 3

Tableau 17- Use-case "Supprimer une place de parking"

Nom	**Supprimer une place de parking**
Acteurs	Utilisateur authentifié
Déclencheur	L'utilisateur clique sur le bouton «Supprimer la place»
Pré-conditions	L'utilisateur a sélectionné une place de parking et est authentifié (use-case « Se connecter »)
Flux principale	1. L'utilisateur clique sur le bouton « Supprimer la place» 2. Une confirmation est demandée 3. L'API supprime la place
Flux secondaire	2.1 L'utilisateur s'est trompé 2.1.1 Retour au Flux N° 1 2.2 L'utilisateur veut supprimer la place 2.2.1 Retour au flux N° 3

Tableau 18 - Use-case "Modifier une place de parking"

Nom	Modifier une place de parking
Acteurs	Utilisateur authentifié
Déclencheur	L'utilisateur clique sur le bouton «Modifier la place»
Pré-conditions	L'utilisateur a sélectionné une place de parking et est authentifié (use-case « Se connecter »)
Flux principale	1. L'utilisateur clique sur le bouton « Modifier la place» 2. Une confirmation est demandée 3. L'API définit si les informations reçues sont correctes 4. L'API modifie la place avec les nouvelles données
Flux secondaire	2.1 L'utilisateur veut modifier la place 2.1.1 Retour au flux N° 4 2.2 L'utilisateur c'est trompé 2.2.1 Retour aux flux N° 1 3.1 Les informations sont correctes 3.1.1 Retour au flux N° 3 3.2 Les informations ne sont pas correctes 3.2.1 L'API retourne un message d'erreur 3.2.2 Le message d'erreur est affiché 3.2.1 Retour au flux N° 1

Tableau 19 - Use-case "Modifier les informations de son compte »

Nom	Modifier les informations de son compte
Acteurs	Utilisateur authentifié
Déclencheur	L'utilisateur clique sur le bouton «Modifier»
Pré-conditions	L'utilisateur est son profil et est authentifié (use-case « Se connecter »)
Flux principale	1. L'utilisateur clique sur le bouton « Modifier» 2. Une confirmation est demandée 3. L'API définit si les informations reçues sont correctes 4. L'API modifie les informations de l'utilisateur
Flux secondaire	2.1 L'utilisateur veut modifier la place 2.1.1 Retour au flux N° 4 2.2 L'utilisateur c'est trompé

2.2.1 Retour aux flux N° 1

3.1 Les informations sont correctes

3.1.1 Retour au flux N° 3

3.2 Les informations ne sont pas correctes

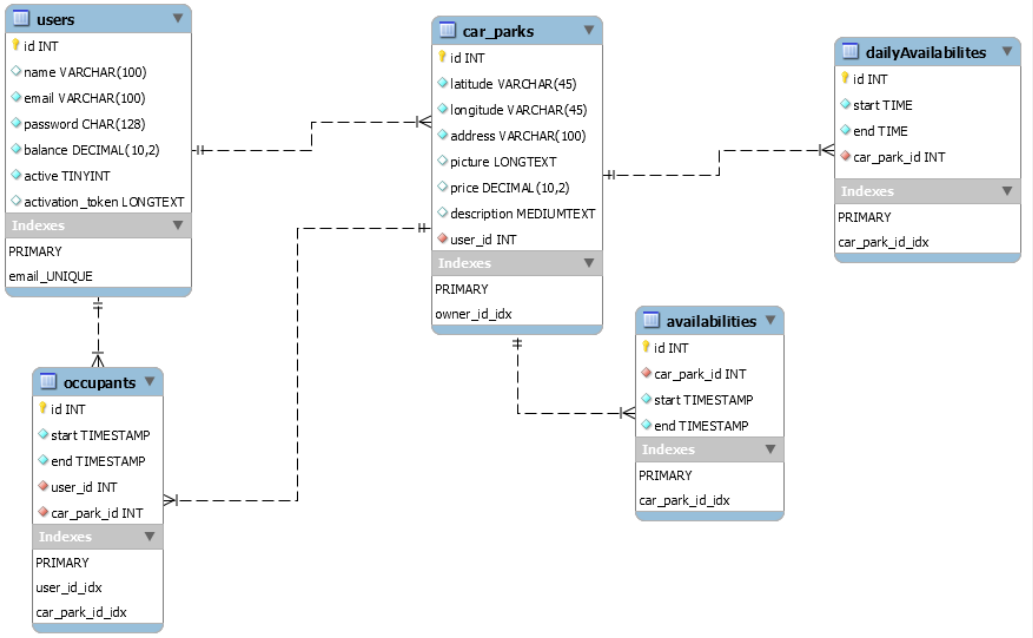
3.2.1 L'API retourne un message d'erreur

3.2.2 Le message d'erreur est affiché

3.2.1 Retour au flux N° 1

5.2 Modèle de données

Figure 13 - Modèle de données de l'application



Ma base de donnée est composée de 5 tables. Une stockant les utilisateurs, une pour les places de parking, une pour les locations et finalement deux pour les horaires des places de parking.

Les horaires sont enregistrés dans deux tables différentes pour une question de simplicité. Effectivement, un horaire peut être journalier ou bien il avoir une certaine durée. Cela permet de séparer les 2 systèmes d'horaires et d'avoir une base de donnée plus propre. De plus, en cas de recherche pour une durée plus grande que 1

jour, il ne sera même pas nécessaire de rechercher dans la table des horaires journaliers.

Au niveau du front-end, cela facilitera également l'affichage car il sera alors possible de séparer les horaires journaliers des horaires normaux de manière bien plus simplifié.

5.3 Choix du système back-end

Le back-end de mon application doit pouvoir gérer un enregistrement et une connexion sécurisée ainsi que le stockage des données dans une base de données. Pour répondre à ces attentes, j'ai décidé de créer une API REST récupérant et retournant des objets au format JSON.

Je dispose d'un hébergement suffisamment gros pour y mettre mon API. Elle ne doit cependant pas nécessiter JAVA pour fonctionner car mon hébergement ne le supporte pas. De plus, elle ne doit pas être créée avec PHP 5.6 ou moins.

En étudiant les différentes solutions, j'ai décidé de créer mon back-end en utilisant le Framework Laravel. Ce dernier est un Framework PHP très complet proposant un système de façade pour accéder directement aux puissantes classes offertes par le Framework.

De plus, Laravel est très simple à apprendre et très facilement maintenable.

5.3.1 Les routes de l'API

Commenté [SL7]: A changer dans laravel !

Toutes les routes de l'api commencent par http://parkaps.chalet-schupbach.ch/api ! Les paramètres des requêtes et leurs réponses sont passés au format JSON.

Pour le middleware auth:api, il faut ajouté un header « Authorization ».

Tableau 20 - Liste des routes de l'API

Numéro	Verbe	URL	Middleware	Paramètres	Résultat	Description

1	POST	/users	-	{
				name
				email
				password
				password_confirmation
				}
					201	Enregistrement d'un utilisateur

2	PUT	/users/activate/{token}	-	-	Page HTML de réussite	Activation du compte d'une utilisateur

3	POST	/login	-	{
				email
				password
				remember_me
				}
					200	Connexion d'un utilisateur, retourne un token bearer

4	GET	/logout	auth:api	-	200	Déconnexion de l'utilisateur connecté, revoke le token
5	GET	/user	auth:api	-	200	Récupération des informations relatives à l'utilisateur connecté

6	PUT	/user	auth:api	{
				name
				email
				password
				password_confirmation
				}
					201	Modification de l'utilisateur connecté

Numéro	Verbe	URL	Middleware	Paramètres	Résultat	Description
8	DELETE	/user	auth:api	-	200	Suppression de l'utilisateur connecté. Suppression des informations y relatives en cascades

9	POST	/carparks	auth:api	{
				latitude
				longitude
				address
				price
				picture
				description
				}
					201	Création d'une place de parking. Le détenteur est l'utilisateur connecté

10	PUT	/carparks/{id}	auth:api	{
				latitude
				longitude
				address
				price
				picture
				description
				}
					201	Modification d'une place de parking. Le détenteur est l'utilisateur connecté.
						Vérification que le détenteur de la place de parking soit le même que celui appelant l'API

11	GET	/carparks/{id}	auth:api	-	200	Récupération d'une place de parking
12	DELETE	/carparks/{id}	auth:api	-	200	Suppression d'une place de parking Vérification que le détenteur de la place de parking soit le même que celui appelant l'API

Numéro	Verbe	URL	Middleware	Paramètres	Résultat	Description

13	POST	/availabilites	auth:api	{
				start
				end
				daily
				car_park_id
				}
					201	Création d'horaire pour une place de parking
						
						Vérification que le détenteur de la place de parking de l'horaire soit le même que celui appelant l'API

14	PUT	/availabilites/{id}	auth:api	{
				start
				end
				daily
				}
					201	Modification d'un horaire pour une place de parking
						
						Vérification que le détenteur de la place de parking de l'horaire soit le même que celui appelant l'API

15	DELETE	/availabilites/{id}	auth:api	{
				daily
				}
					200	Suppression d'un horaire
						
						Vérification que le détenteur de la place de parking de l'horaire soit le même que celui appelant l'API

16	GET	/carparks/{id}/availabilites	auth:api	-	200	Récupération des horaires d'une place de parking

17	POST	/occupants	auth:api	{
				start
				end
				car_park_id
				}
					201	Création d'une location

Commenté [SL8]: A garder ?, plutôt supprimer, a voir !

Numéro	Verbe	URL	Middleware	Paramètres	Résultat	Description
18	DELETE	/occupants/{id}	auth:api	-	200	Suppression d'une location Vérification que le détenteur de la location soit le même que celui appelant l'API
19	GET	/user/occupants	auth:api	-	200	Récupération des locations de l'utilisateur connecté
20	GET	/token	auth :api	-	200	Vérification du fonctionnement du token

5.3.2 Fonctionnement de l'API

Laravel est un Framework de développement WEB entièrement constitué en PHP. Il se base sur composer (https://getcomposer.org/) pour gérer ses dépendances. Laravel offre de multiples fonctionnalités, tel que le routage de requêtes, le mapping d'objet relationnels, l'authentification, la migration de base de données, la gestion des exceptions, les tests unitaires et l'affichage de page HTML (dans le format propriétaire Blade).

Ce Framework est très complet et permet la création d'un site WEB comme d'une API complexe. Il est séparé en 2 parties : la partie WEB et la partie API. Il est donc possible de créer un site WEB et une API tournant sur le même serveur et utilisant la même base de données.

Le Framework est très bien créé. Il est séparé en plusieurs sous-dossiers ayant chacun un objectif bien défini (un pour la gestion de la base de données, un pour la gestion des vues, un pour la configuration du Framework, etc.). Il est très facile de se familiariser avec ce Framework et de le prendre en main.

Pour la gestion de l'accès à la base de données, Laravel a créé un système de modèle éloquent, qui est directement connecté à la bonne table dans la base de données. Il n'est donc pas nécessaire de créer nous-mêmes nos requêtes SQL. Il suffit de créer un objet « modèle » pour qu'il s'enregistre automatiquement dans la base de données.

De même lorsque l'on va vouloir récupérer des données dans la base de données, il est possible de faire des recherches directement via l'objet correspondant. Les données seront alors retournées sous la forme d'un tableau d'objet « modèle ».

En ce qui concerne l'authentification, Laravel propose un système appelé « Passport ». Ce package permet de faire de l'authentification OAuth2 de manière simplifié et très rapide.

Pour sécuriser une route, il suffit de la placer dans le middleware « auth:api ». Celui-ci vérifiera si le token fournit dans la requête correspond à un utilisateur et si ce n'est pas le cas, répondra avec une erreur 401, Unauthorized. Pour fournir le token dans la requête, il suffit de rajouter le header Authorization avec le format suivant :

Bearer mon_token

En premier lieu, le type de token (Bearer pour passport) puis un espace et finalement le token en lui-même.

Bibliographie

ALTEXSOFT, 2018a. Pros and Cons of Xamarin vs Native Mobile Development. In : [en ligne]. 8 mai 2018. [Consulté le 17 août 2018]. Disponible à l'adresse : https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/.

ALTEXSOFT, 2018b. Xamarin vs React Native vs Ionic: Cross-platform Mobile Frameworks Comparison. In : [en ligne]. 19 février 2018. [Consulté le 13 juin 2018]. Disponible à l'adresse : https://www.altexsoft.com/blog/engineering/xamarin-vs-react-native-vs-ionic-cross-platform-mobile-frameworks-comparison/.

ANGULAR, 2009a. Angular - Architecture overview. In : [en ligne]. 2009. [Consulté le 18 juin 2018]. Disponible à l'adresse : https://angular.io/guide/architecture.

ANGULAR, 2009b. Angular. In : [en ligne]. 2009. [Consulté le 18 juin 2018]. Disponible à l'adresse : https://angular.io/.

REACT, 2018. Introducing JSX – React. In : [en ligne]. 2018. [Consulté le 19 août 2018]. Disponible à l'adresse : https://reactjs.org/docs/introducing-jsx.html.

ALTEXSOFT, 2018. Pros and Cons of ReactJS and React Native. In : [en ligne]. 10 mai 2018. [Consulté le 17 août 2018]. Disponible à l'adresse : https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native/.

CORDOVA, Apache, 2012. Architectural overview of Cordova platform - Apache Cordova. In : [en ligne]. 2012. [Consulté le 15 juin 2018]. Disponible à l'adresse : https://cordova.apache.org/docs/en/latest/guide/overview/index.html.

CRUXLAB, 2017. Xamarin vs Ionic vs React Native: differences under the hood. In : [en ligne]. 2017. [Consulté le 8 juin 2018]. Disponible à l'adresse : https://cruxlab.com/blog/reactnative-vs-xamarin/.

DRIFTY, 2017. Ionic Framework. In : *Ionic Framework* [en ligne]. 2017. [Consulté le 18 juin 2018]. Disponible à l'adresse : https://ionicframework.com/docs/cli/projects.html.

HEARD, Pete, 2017. React Native Architecture : Explained! In : *React Native Architecture : Explained!* [en ligne]. 13 juin 2017. [Consulté le 25 juin 2018]. Disponible à l'adresse : https://www.logicroom.co/react-native-architecture-explained/.

NEXGENDESIGN, Non daté. 7 Reasons Xamarin Can Be a Trouble | NexGenDesign. In : [en ligne]. Non daté. [Consulté le 19 juin 2018]. Disponible à l'adresse : http://www.nexgendesign.com/xamarin-troubles.

REACT NATIVE, 2018. React Native · A framework for building native apps using React. In : [en ligne]. 2018. [Consulté le 25 juin 2018]. Disponible à l'adresse : https://facebook.github.io/react-native/index.html.

STACK OVERFLOW, 2018. Stack Overflow Developer Survey 2018. In : *Stack Overflow* [en ligne]. 31 janvier 2018. [Consulté le 18 juin 2018]. Disponible à l'adresse : https://insights.stackoverflow.com/survey/2018/.

VINCENT, 2016. Cordova : Applications mobiles hybrides. In : *VinceOPS* [en ligne]. 25 février 2016. [Consulté le 18 juin 2018]. Disponible à l'adresse : https://vincent-g.fr/2016/02/25/ionic-apache-cordova-developpement-mobile-hybride/.