# Table des matières

# Liste des tableaux



# Liste des figures

# 1. Introduction

L'informatique est un des rares domaines ou l'autodidactisme est possible et facile. De nombreux tutoriels et de grandes communautés sont disponibles sur internet pour nous permettre d'apprendre facilement et rapidement les bases du domaine. Il semble donc logique que des aides au développement mobile, qui est de nos jours une part essentiel de l'informatique, voient le jour et se popularisent.

Il a toujours existé une façon efficace de créer une application, le natif. C'est-à-dire, créer l'application dans le langage du téléphone. Malheureusement, la plupart des applications doivent être disponibles sur Android comme sur IOS. Dans ce cas-là, le travail doit être fait à double. Il faut créer une application utilisant le langage d'Android puis recréer exactement la même application mais avec le langage d'IOS.

Cela permet d'avoir des applications fiables mais augmente considérablement les coûts et la durée du développement.

Pour remédier à ce problème, plusieurs sociétés ont créer des Frameworks de développement mobile dits « cross-platform ». Ces Frameworks permettent le développement d'une seule application dans un langage spécifique qui peut ensuite être lancée sur Android comme sur IOS. Le développement cross-platform offre :

- Une réduction massive des coûts de développement et de maintenance
- Des connaissances nécessaires réduites
- La création de codes réutilisables
- Une migration et maintenabilité simplifiées

Le but de mon Bachelor est de présenter et expliquer les Frameworks de développement mobile cross-platform les plus utilisés actuellement et d'en prendre un en main pour soutenir mes explications.

# 2. Etude des Frameworks existants

Le marché actuel propose un certain nombre 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 le 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. Ils utilisent alors un langage différent des langages natifs (souvent Javascript pour la partie logique et HTML pour la partie visuel). Ces langages ne sont pas compris par les téléphones mobiles et doivent être exécutés dans un environnement spécifique : le « Runtime Environnement ». Ces environnements encapsulent les projets « cross-platform » dans une « boîte » qui comprend le langage du Framework. Cette boîte offre 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 visuelle, quant à elle, est le plus souvent affichée dans une WebView. Cette dernière est une fenêtre prenant toute la place de l'écran et qui comprend les langages Web (HTML / CSS). Cette dernière est affichée à l'écran et la partie visuelle de l'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, laquelle est plus ou moins performante.

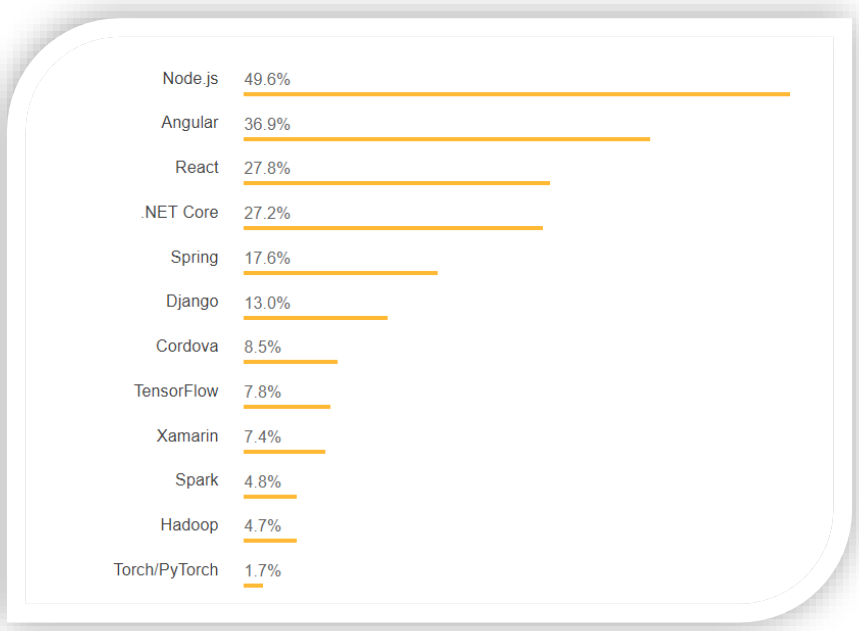
Il faut faire attention à bien vérifier si les fonctionnalités visuelles misent en place sont compatibles avec la WebView native du système d'exploitation cible.

Comme expliqué précédemment, 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 places de parking, je vais expliquer les avantages et les désavantages des trois Frameworks les plus connus et utilisés.

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

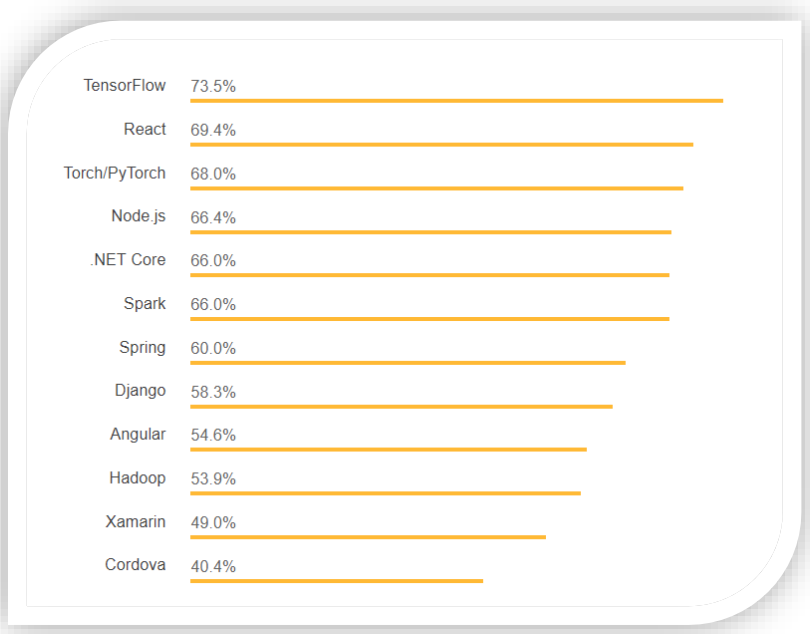
Ce sondage, réalisé pendant le mois de janvier 2018 , a reçu des réponses de plus de 67 000 développeurs jugés « qualifiés ». Les deux graphiques suivants représentent l'utilisation des Frameworks :

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 graphiques représentent très bien la situation actuelle. Sur tous les Frameworks présents, quatre sont des Frameworks permettant le développement mobile cross-platform, ci-dessous triés par ordre de popularité :

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

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

## 2.1 Ionic

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

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 grandes 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 Angular pour le binding de données ainsi qu'Apache Cordova pour transformer le projet en un package que le téléphone mobile peut comprendre, appelé APK.

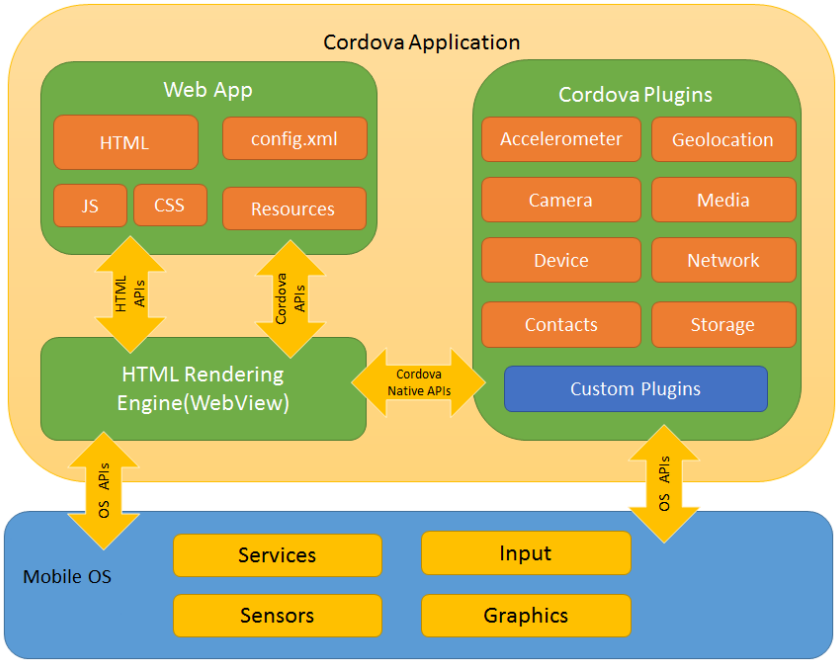
### 2.1.1 Architecture

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

#### 2.1.1.1 Apache Cordova

La création de l'APK et le test de l'application sur le téléphone mobile sont une étape essentielle du développement. Il existe plusieurs façons de créer un APK. Apache Cordova propose d'encapsuler facilement l'application dans un 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 l'architecture de ce Framework.

L'application, qui peut être créée avec n'importe quel Framework compatible, sera encapsulée 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, lequels seront ensuite envoyés aux plugins d'Apache Cordova. Ces derniers vont alors transpiler les

données en quelque chose de lisible par l'application et les envoyer à la WebView qui, elle-même, transmettra les informations à la partie logique de l'application.

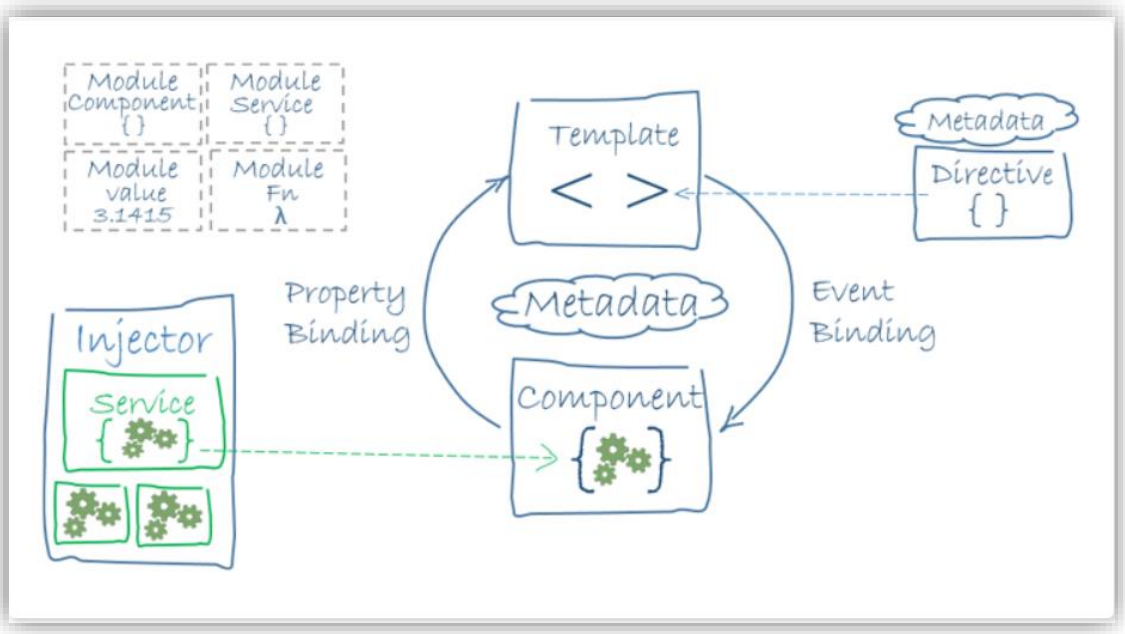
Apache Cordova utilise l'environnement de runtime natif JavaScript de chaque système d'exploitation pour faire fonctionner l'application et créer les APKs.

#### 2.1.1.2 Angular

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

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

Figure 4 - Architecture d'Angular




(Angular, 2009a)

On peut voir sur ce schéma le fonctionnement d'Angular. Chaque objet logique du site Web ou de l'application mobile (page, liste, bouton personnalisé, etc.) va être enregistré en tant que « composant ». Un composant désigne la partie logique de l'objet et est donc relié à un modèle écrit en HTML / CSS (l'interface affichée à l'utilisateur). Cette liaison offre la possibilité 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 qu'au moment où une donnée est affichée sur le modèle et

que la valeur de la variable représentant cette donnée change dans le composant, l'affichage de cette donnée va automatiquement être mise à jour dans le modèle. Ce système fonctionne aussi en sens inverse. Lorsque l'utilisateur va écrire du texte dans un champ, il est possible d'enregistrer son contenu, en temps réel, dans une variable du composant.

L'Event Binding permet, quant à lui, le déclenchement automatique d'une méthode du composant lorsque l'utilisateur va faire une action sur le template (appuyer sur un bouton par exemple).

Angular 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.) afin d'accéder aux données et méthodes de ce dernier.

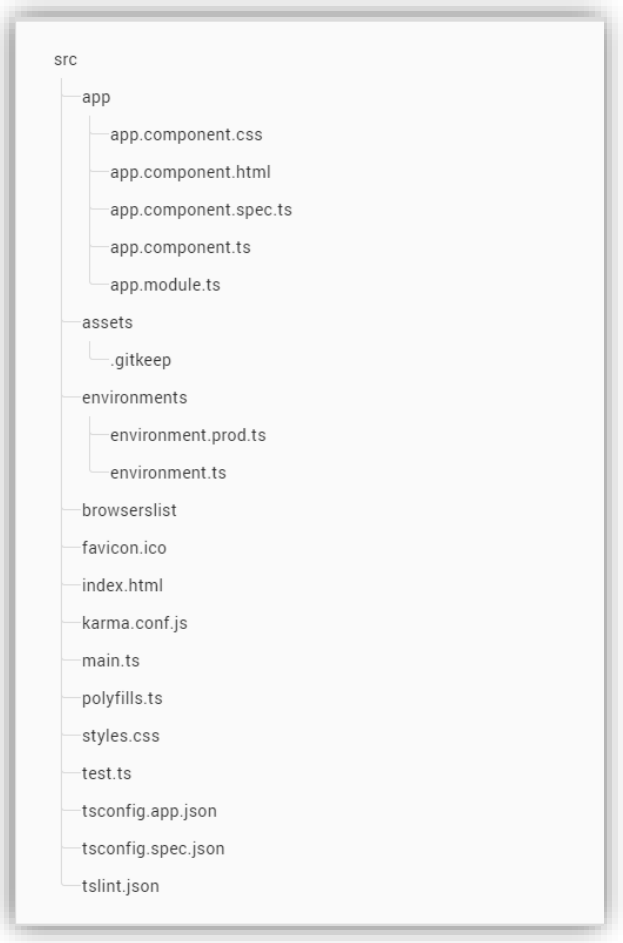
#### 2.1.1.3 Ionic

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

Figure 5 - Structure de fichiers d'un projet Ionic


(Drifty, 2017)

Figure 6 - Structure de fichiers d'un projet Angular




(Angular, 2009b)

La structure de dossier d'un 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'Angular qui contient plusieurs fichiers de configuration. De plus, le dossier « src » d'Ionic contient des sous-dossiers structurant le projet avec des pages, des assets et différents thèmes.

Ionic inclu dans son Framework plusieurs composants graphiques responsifs ainsi que cross-platform qui sont 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 qu'un contrôle des erreurs plus poussé.

### 2.1.2 Avantages

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

Comme Ionic utilise le JavaScript ou TypeScript ainsi que l'HTML pour la partie visuelle, cela permet à un grand nombre de développeurs d'utiliser ce Framework. En effet, les langages WEB sont les premiers langages que tout développeur apprend car ils sont très simples à comprendre. Il est très facile de trouver de l'aide ainsi que 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 le site web en une application par quelques modifications seulement.

Ionic a, en effet, une portabilité tellement grande avec les composants offerts qu'il est même possible de créer un seul code pour une application Android, IOS et un site web (pour autant qu'il n'y ait pas de PHP). Seuls quelques pourcents du code doivent être adaptés aux plateformes pour permettre son fonctionnement (cela concerne surtout les plugins qui ne fonctionnent 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, légers, 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 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 désavantage de ce Framework est la compatibilité entre les différents « sous-frameworks » qui le compose. Il est assez fréquent que des modifications soient faites au niveau d'Apache Cordova (surtout au niveau des plugins) et que celles-ci cassent le fonctionnement actuel de notre application. De plus, certains plugins ne sont pas fonctionnels et ne peuvent pas être utilisés lors du passage en production de l'application. (Vincent, 2016)

## 2.2 Xamarin

Xamarin est un Framework dédié au développement d'applications mobiles « 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 le langage C# et non JavaScript.

Contrairement aux idées reçues, 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, 2018b)

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 visuelle de l'application, il existe deux options :

- Xamarin.Android et Xamarin.IOS permettant de créer une interface native pour chaque OS cible
- Xamarin.Forms permettant de créer un seul design « cross-platform » pour Android, IOS et Windows Phone.

La première option est bien évidemment la plus performante, car chaque OS cible aura son propre design créé en langage natif. 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.

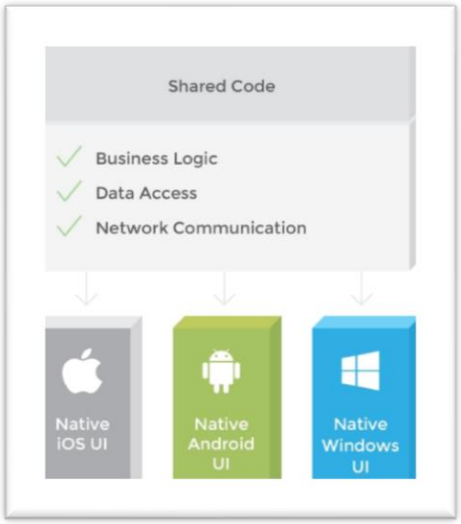
La deuxième option, Xamarin.Forms, facilite grandement le développement et permet d'avoir un design unifié entre les différentes plateformes mais pose un problème de taille. Les performances sont grandement diminuées pour certaines opérations et la création de designs compliqués sera impossible. (Altexsoft,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

### 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 les accès aux composants natifs du téléphone se trouvent dans un bloc de code 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 simplifiée d'une application Xamarin



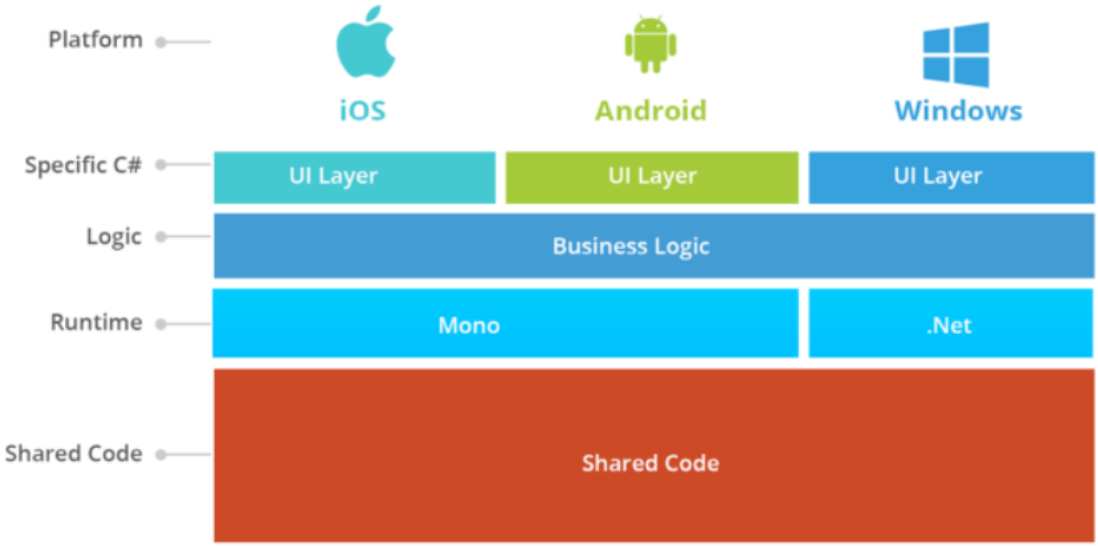

(Altexsoft, 2018b)

Les interfaces sont créées dans le langage XAML qui est une extension du langage XML. XAML permet l'intégration aux balises XML de commandes associées à des propriétés logiques. Ce système va fonctionner exactement de la même manière qu'Angular avec le binding de données. (Xaml.fr, 2012)

Tout comme pour Ionic, les systèmes d'exploitation mobiles ne comprennent pas le langage C# et doivent donc avoir un environnement de runtime spécifique. Étant donné que ces plateformes n'ont pas cet environnement de manière native, Xamarin a pris la décision de l'ajouter dans le code source de l'application. De cette manière, quel que soit le système d'exploitation mobile sur lequel est lancée 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 utilisateur différenciées pour chaque plateforme cible, ses composants vont être natifs. Le seul code non natif sera alors celui partagé en C# qui tourne dans son propre environnement de runtime créé 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 partager entre 90% et 100% du code en utilisant cette option.

### 2.2.2 Avantages

Xamarin est un Framework proposant une approche totalement différente des autres Frameworks de développement mobiles. L'utilisation du langage C#, 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 ce celles 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 le chargement d'une application.

La possibilité qu'offre Xamarin de créer l'interface utilisateur de deux manières différentes est aussi un atout pour le Framework. Si l'application développée n'utilise pas beaucoup de ressources, il est possible de créer une seule interface utilisateur partagée sur chaque plateforme. Même si l'application est une application lourde, Xamarin donne aussi la possibilité de créer des interfaces utilisateur natives pour garder des performances proches du natif. Il est aussi possible d'utiliser Xamarin.Forms pour créer des prototypes de l'application et de les montrer aux clients. Le client donne son avis et lorsque l'interface sera définitive, elle peut être réécrite pour chaque OS cible et ainsi augmenter les performances de l'application.

### 2.2.3 Faiblesses

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

La communauté utilisant le Framework Xamarin est aussi plus faible que celle utilisant les Frameworks utilisant les technologies WEB. Il sera donc plus difficile de trouver des réponses aux questions lorsque qu'il y aura 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 de l'application est aussi un inconvénient de Xamarin par rapport à ses concurrents. Comme elle n'est pas comprise par le système d'opération du téléphone mobile, elle emporte avec elle d'autres outils pour la faire fonctionner (runtime, assemblage de libraires de classes de base, etc.) qui l'alourdissent grandement. Ci-dessous, la taille d'une application affichant un simple « Hello World » à l'écran. L'affichage et la partie logique ne pèsent que 0.037% du poids total de l'application.

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


(Altexsoft, 2018b)

Xamarin a été acheté par Microsoft en février 2016, permettant son utilisation gratuitement. Il a également intégré le Framework à son IDE (environnement de développement) Visual Studio.

Il est conseillé d'obtenir Visual Studio pour créer une application Xamarin. Une version gratuite de Visual Studio existe mais elle n'offre pas toutes les fonctionnalités qui permettent de travailler correctement sur le projet.

Pour profiter pleinement de Visual Studio et de Xamarin, il est essentiel d'acquérir une licence Visual Studio.

## 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 il propose un 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, le JSX. Ce langage ressemble à l'HTML (avec son système de balise) mais ne représente pas les mêmes objets. Chaque balise est une encapsulation d'un élément natif du téléphone. Ce système a été choisi car React part du principe que la logique du rendu est fortement couplée à la logique d'affichage des interfaces utilisateur : 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

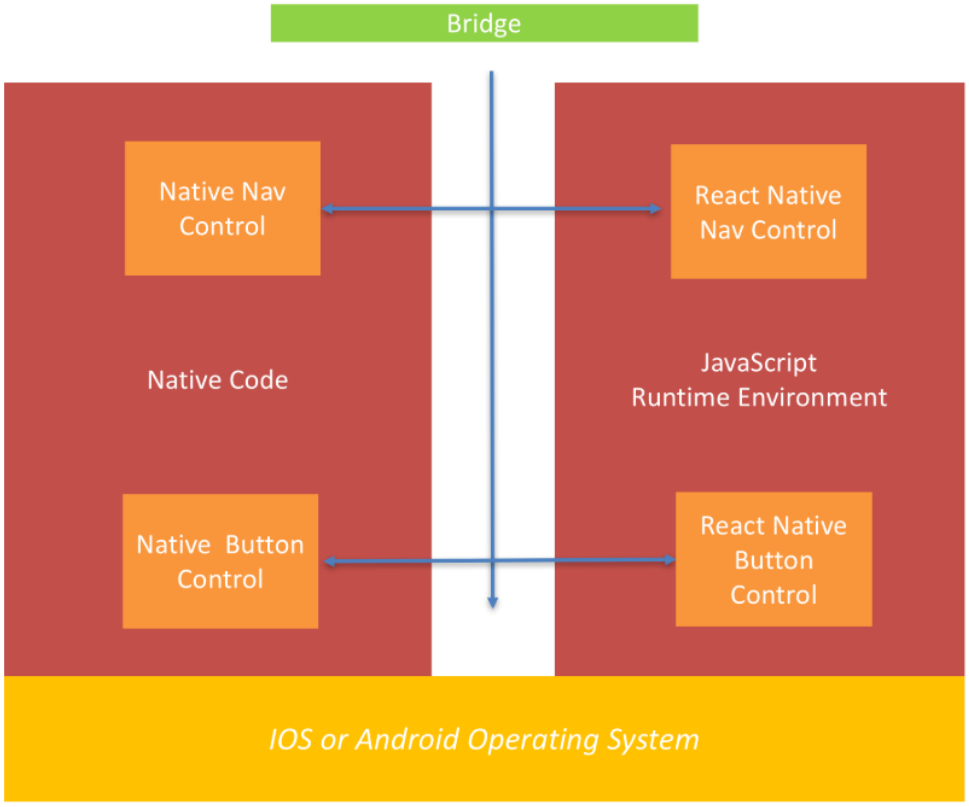
React Native a un fonctionnement très spécifique propre à lui-même. En effet, un Framework basique tel que Ionic possède ses propres composants qui seront connectés via des APIs aux composants natifs du téléphone. Une multitude de couches se superposent pour offrir l'accès aux fonctionnalités du téléphone.

React Native a été construit d'une toute autre manière. Chaque composant est directement branché au composant natif dont il fait 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 lié à Javascript.

Figure 10 - Architecture de React Native

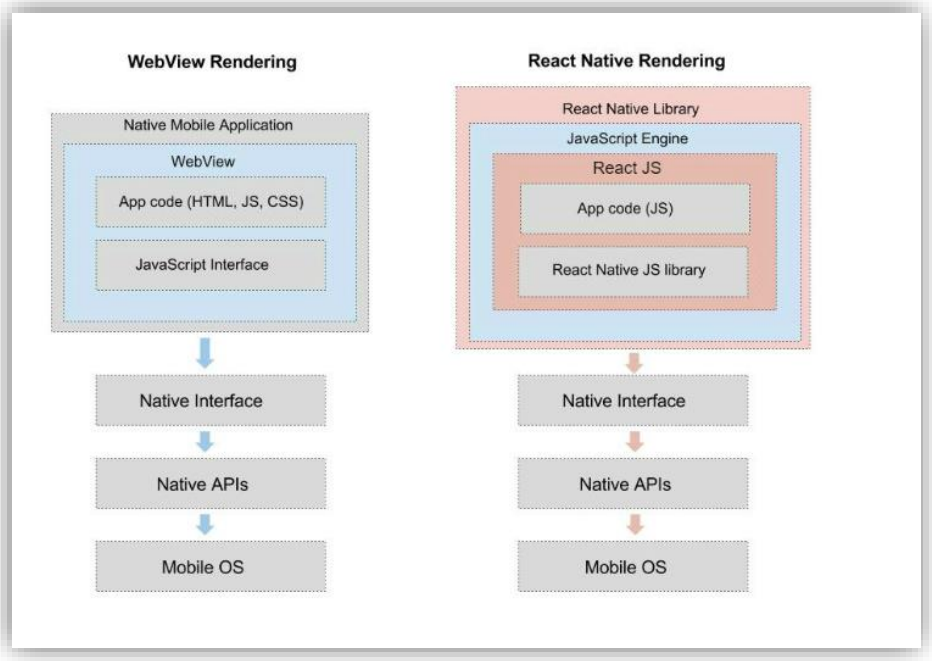



(Heard, 2017)

L'application est alors lancée dans cet environnement, ce qui permet d'accéder au langage JavaScript et de faire fonctionner l'application. Des ponts sont ensuite créés entre les composants offerts par React Native et les composants natifs du téléphone. Le reste de l'application communique ensuite uniquement via les composants React Native qui sont eux-mêmes 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 augmentent grandement les performances d'une application React Native.

Figure 11 - Différence WebView et React Native




(Altexsoft, 2018a)

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 dont il lie le Framework aux composants Natifs du téléphone. Cela permet 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 ses spécificités et il est nécessaire d'adapter légèrement son code pour chaque plateforme ciblée. De 90% à 100% du code crée 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 ses propres composants.

Toute la partie logique de l'application étant codée en JavaScript, aucun apprentissage spécifique ne doit normalement être fait pour apprendre ce langage très connu.

Un point important de React Native est son système de débogage. Il est possible de lancer son application directement sur son téléphone et de pouvoir débugger en temps 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.

React Native crée une encapsulation des composants natifs du téléphone et ne propose donc qu'une petite collection de composants. Pour créer des interfaces compliquées, il faudra empiler ces composants basiques les un dans les autres, chose qui peut vite se révéler complexe.

Le plus gros désavantage réside dans l'ajout de composants externes / tierces à l'application. La communauté a conçu des composants et modules augmentant les possibilités de création de React Native. Malheureusement, ces composants ne sont pas toujours mis à jour et souvent, ils ne vont pas pouvoir être implémentés au projet. (Altexsoft, 2018a)

## 2.4 Comparaison des Frameworks

Chaque Framework comporte des avantages et des inconvénients. Une application créée via le Framework Ionic pourra être personnalisée mais sera vite limitée au niveau des performances. Xamarin se trouve être un juste milieu permettant d'obtenir des bonnes performances et un accès aux composants natifs simplifiés. Toutefois, Xamarin limite la création d'interface et oblige à connaitre le langage C#. React Native propose un mélange des deux autres Frameworks avec des performances proches du natif et un développement simplifié cross-platform 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 cinq 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 août 2019) |
| UI performance | *IOS et Android* | HTML | Composants natifs | Composants natifs |

(Cruxlab, 2017)

Question performance, Xamarin remporte haut la main la première place. En raison de son propre environnement de runtime et de son interface utilisateur qui peut être écrite en code natif, il est bien plus rapide. En deuxième place se trouve React Native. Bien que le 64 bits ne soit pas encore supporté, il reste plus performant car il utilise les composants natifs du système d'exploitation. Le fonctionnement d'Ionic place ce Framework dernier au niveau des performances.

Il est cependant important de noter que Google va exiger dès août 2019 que toutes les applications sur le Play Store puissent supporter des libraires 64 bits en plus des 32 bits. Les applications déjà existantes devront alors être mises à jour et les Frameworks, tels que React Native, devront alors ajouter le 64 bits à leur fonctionnement.

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 au développement, React Native passe premier et de loin. En effet, Xamarin ne change en rien du développement natif. Au niveau d'Ionic, seules la permutation du code à froid et la possibilité de visualiser l'application dans un navigateur ont été ajoutées. Il faut être néanmoins attentif à cette deuxième option car l'application n'aura pas exactement le même design que sur le téléphone et les

fonctionnalités natives ne sont pas prises en compte (géolocalisation, accéléromètre, appareil photo, etc.). React Native, de son coté, ajoute énormément d'aides au développement qui facilite grandement la création de l'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 à des bibliothèques standards | *Set de fonctionnalités supportées* | Partiel | Complet | Partiel |
| | *Type de liaison* | Couche d'abstraction | One-to-one | Couche d'abstraction |
| UI | *Set complet de composants disponibles* | 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 semblables, Xamarin est légèrement au-dessus car il supporte mieux les composants natifs et l'ajout de libraires 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 comprendre ici que chaque Framework à ses avantages et ses inconvénients :

- Xamarin est destiné à des développeurs confirmés connaissant bien le langage C#. Il est également le Framework idéal pour des projets nécessitant des performances accrues mais sans interface utilisateur compliqué.
- React Native est un bon compris proposant de bonnes performances ainsi que des aides au développement poussées tout en ne nécessitant pas trop de connaissances supplémentaires aux technologies WEB.
- Ionic est le Framework le plus simple à prendre en main avec des possibilités de personnalisation énormes. Il est malheureusement aussi le moins performant des trois.

# 3. Etude de l'application

L'application Parkaps offre une solution pour régler le problème du manque de place de parking. Elle permet à toute personne disposant d'une place de parking privée de la mettre en location et à un utilisateur de la louer pour un laps de temps défini.

## 3.1 Fonctionnalités de l'application

L'application offre les fonctionnalités suivantes :

- S'enregistrer au sein de l'application
- Se connecter à son compte
- Supprimer son compte
- Modifier 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
- 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
- Supprimer une location
- Visualiser ses places et réservations

## 3.2 Besoins techniques de l'application

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

Après connexion de l'utilisateur, une carte s'affichera sur son écran. Il pourra alors positionner une cercle de recherche et visualiser toutes les places disponibles à cet endroit. L'utilisateur pourra alors comparer les différentes places et en réserver une en précisant ses horaires d'utilisation.

L'application sera disponible sur le Play Store d'Android et, par la suite, sur 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 de celles d'une application native.

# 4. Choix du Framework

Les trois Frameworks qui ont été présentés précédemment sont très intéressants. Chacun a ses avantages et ses inconvénients.

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

## 4.1 Analyse détaillée du choix

Pour sélectionner le Framework le plus approprié pour mon application, j'ai utilisé une analyse multicritères 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

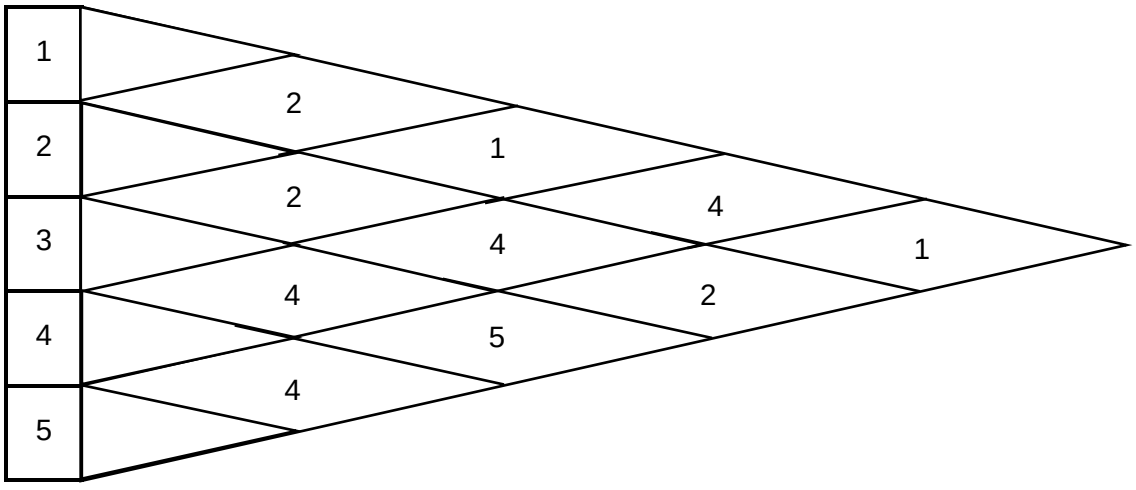


Tableau 7 - Pondération des critères de choix du Framework

| Numéro | Critère | Pondération |
|---|---|---|
| **1** | Connaissance du langage associé | 2 |
| **2** | Accès aux fonctionnalités natives du téléphone | 3 |
| **3** | Framework bien documenté | 0<br>(donc critère supprimé) |
| **4** | Performances | 4 |
| **5** | Maintenabilité | 1 |

Comme ce travail de Bachelor a deux volets, l'un pour expliquer les Frameworks et l'autre pour la création d'une application, il est essentiel de choisir un Framework proposant des performances adéquates à mon projet ainsi qu'un accès aux fonctionnalités natives facilité. Néanmoins, disposant d'un temps de travail limité, il est tout aussi important que le langage associé au Framework soit un langage que je pratique et connaisse. (Je ne dispose pas du temps nécessaire pour apprendre les spécificités du Framework choisis et m'adapter à un nouveau langage)

Dans un premier temps, j'ai décidé de créer mon application sur la plateforme Android pour une question de simplicité et de coûts. Toutefois, j'envisage, à terme, de fournir mon application également sur IOS. Raison pour laquelle tous les Frameworks proposés sont cross-platform.

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 multicritères le prouve. Il est précédé par Xamarin qui est situé à 5 points seulement 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). Toutefois, son langage ne concorde pas avec mes connaissances et perd dès lors des points. 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 utilisent un environnement de runtime JavaScript intégré à l'appareil et donc mis 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

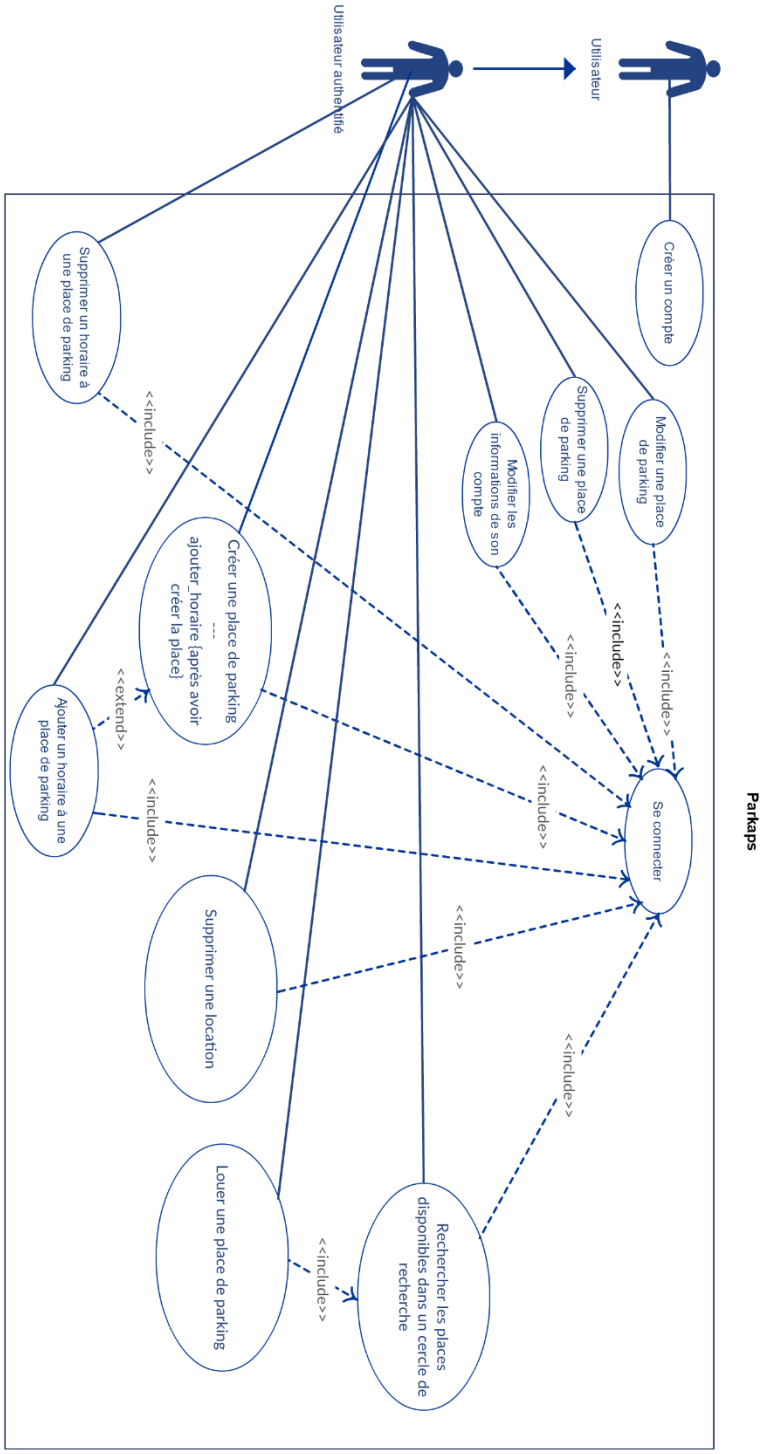


Tableau 9 - Use-case "Créer un compte"

| Nom | Créer un compte |
|---|---|
| **Acteurs** | Utilisateur |
| **Déclencheur** | L'utilisateur clique sur le bouton « S'enregistrer » |
| **Préconditions** | L'utilisateur s'est rendu sur la page d'enregistrement |
| **Flux principale** | 1. L'utilisateur clique sur le bouton « S'enregistrer »<br>2. Les informations d'enregistrement sont envoyées à l'API<br>3. L'API définit si les informations reçues sont correctes<br>4. L'API enregistre l'utilisateur et lui envoie un email de validation<br>5. L'utilisateur se rend sur le lien affiché dans l'email de validation<br>6. L'API valide le compte de l'utilisateur |
| **Flux secondaire** | 3.1 Les informations sont correctes<br>3.1.1 Retour au flux N° 4<br>3.2 Les informations ne sont pas correctes<br>3.2.1 L'API retourne un message d'erreur<br>3.2.2 Les champs de texte faux sont surlignés en rouge et un message d'erreur est affiché<br>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 » |
| **Préconditions** | L'utilisateur s'est rendu sur la page de connexion et crée un compte (use-case « Créer un compte » |
| **Flux principale** | 1. L'utilisateur clique sur le bouton « Se connecter »<br>2. Les informations de connexion sont envoyées à l'API<br>3. L'API définit si les informations reçues sont correctes<br>4. L'API créé un token et le retourne<br>5. Le token est enregistré localement sur le téléphone<br>6. La page « Carte » s'affiche |

| | |
|---|---|
| **Flux secondaire** | 3.1 Les informations reçues sont correctes<br>3.1.1 Retour au flux N° 4<br>3.2 L'utilisateur n'a pas validé son compte<br>3.2.1 L'API retourne un message d'erreur<br>3.2.2 Le message d'erreur est affiché<br>3.2.3 Retour au flux N° 1<br>3.3 Les informations de connexion ne sont pas correctes<br>3.3.1 L'API retourne un message d'erreur<br>3.3.2 Le message d'erreur est affiché<br>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 » |
| **Préconditions** | 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 »<br>2. Les informations sont envoyées à l'API<br>3. L'API définit si les informations reçues sont correctes<br>4. L'API créé la place de parking |
| **Flux secondaire** | 3.1 Les informations reçues sont correctes<br>3.1.1 Retour au flux N° 4<br>3.2 Les informations ne sont pas correctes<br>3.2.1 L'API retourne un message d'erreur<br>3.2.2 Le message d'erreur est affiché<br>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 » |
| **Préconditions** | 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 »<br>2. L'utilisateur sélectionne une date de début, une date de fin<br>3. L'API définit si les informations reçues sont correctes<br>4. L'API créé l'horaire |
| **Flux secondaire** | 3.1 Les informations reçues sont correctes<br>3.1.1 Retour au flux N° 4<br>3.2 Les informations ne sont pas correctes<br>3.2.1 L'API retourne un message d'erreur<br>3.2.2 Le message d'erreur est affiché<br>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 » |
| **Préconditions** | 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 »<br>2. Une confirmation est demandée<br>3. L'API supprime l'horaire sélectionné |
| **Flux secondaire** | 2.1 L'utilisateur s'est trompé<br>2.1.1 Retour au Flux N° 1<br>2.2 L'utilisateur veut supprimer la place<br>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» |
| **Préconditions** | 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 »<br>2. L'API définit si les informations reçues sont correctes<br>3. Les places de parking sont affichées sur la carte à leurs positions |
| **Flux secondaire** | 2.1 Les informations reçues sont correctes<br>2.1.1 Retour au flux N° 3<br>2.2 Les informations reçues ne sont pas correctes<br>2.2.1 L'API retourne un message d'erreur<br>2.2.2 Le message d'erreur est affiché<br>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 » |
| **Préconditions** | 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 »<br>2. L'utilisateur sélectionne une date de début et une date de fin<br>3. L'API définit si les informations reçues sont correctes<br>4. L'API enregistre la location |
| **Flux secondaire** | 3.1 Les informations reçues sont correctes<br>3.1.1 Retour au flux N° 4<br>3.2 Les informations reçues ne sont pas correctes<br>3.2.1 L'API retourne un message d'erreur<br>3.2.2 Le message d'erreur est affiché<br>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 » |
| **Préconditions** | 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 »<br>2. Une confirmation est demandée<br>3. L'API supprime la location |
| **Flux secondaire** | 2.1 L'utilisateur s'est trompé<br>2.1.1 Retour au Flux N° 1<br>2.2 L'utilisateur veut supprimer la location<br>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 » |
| **Préconditions** | 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 »<br>2. Une confirmation est demandée<br>3. L'API supprime la place |
| **Flux secondaire** | 2.1 L'utilisateur s'est trompé<br>2.1.1 Retour au Flux N° 1<br>2.2 L'utilisateur veut supprimer la place<br>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 » |
| Préconditions | 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 »<br>2. Une confirmation est demandée<br>3. L'API définit si les informations reçues sont correctes<br>4. L'API modifie la place avec les nouvelles données |
| Flux secondaire | 2.1 L'utilisateur veut modifier la place<br>2.1.1 Retour au flux N° 4<br>2.2 L'utilisateur s'est trompé<br>2.2.1 Retour aux flux N° 1<br>3.1 Les informations sont correctes<br>3.1.1 Retour au flux N° 3<br>3.2 Les informations ne sont pas correctes<br>3.2.1 L'API retourne un message d'erreur<br>3.2.2 Le message d'erreur est affiché<br>3.2.1 Retour au flux N° 1 |

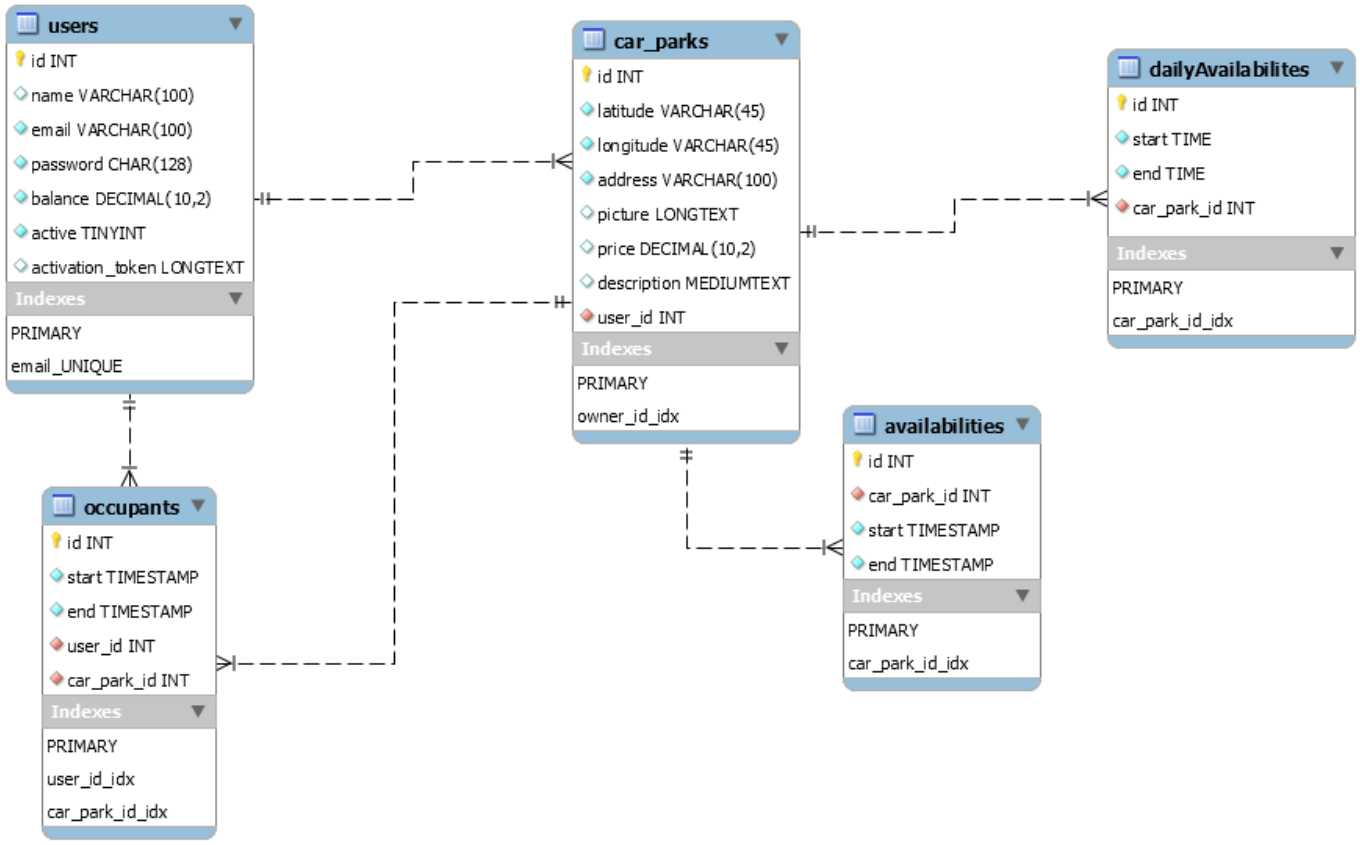
Tableau 19 - Use-case "Modifier les informations de son compte »

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

| | |
|---|---|
| | 2.2.1 Retour aux flux N° 1<br>3.1 Les informations sont correctes<br>3.1.1 Retour au flux N° 3<br>3.2 Les informations ne sont pas correctes<br>3.2.1 L'API retourne un message d'erreur<br>3.2.2 Le message d'erreur est affiché<br>3.2.1 Retour au flux N° 1 |

## 5.2 Modèle de données

Figure 13 - Modèle de données de l'application




Ma base de données est composée de cinq tables :

- Stockage des utilisateurs
- Stockage des places de parking
- Stockage des horaires
- Stockage des horaires journaliers
- Stockage des locations

Les horaires sont enregistrés dans deux tables différentes pour une question de simplicité. Effectivement, un horaire peut être journalier ou défini dans la durée. Cela permet de séparer les 2 systèmes d'horaires et d'avoir une base de données 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 laissant la possibilité de séparer les horaires journaliers de ceux non journaliers de manière bien plus simple.

## 5.3 Choix du système back-end

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

Je dispose d'un hébergement suffisamment grand 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 car celui-ci n'a plus de support actif depuis janvier 2017 et n'aura plus de support de sécurité dès décembre 2018. (PHP, Non daté)

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.

Laravel est très simple d'utilisation, de compréhension et de maintenance.

### 5.3.1 Les routes de l'API

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

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

Tableau 20 - Liste des routes de l'API

| Numéro | Verbe | URL | Middleware | Paramètres | Résultat | Description |
|---|---|---|---|---|---|---|
| **1** | POST | /users | - | {<br>name<br>email<br>password<br>password_confirmation<br>} | 201 | Enregistrement d'un utilisateur |
| **2** | GET | /users/activate/{token} | - | - | Page HTML de réussite | Activation du compte d'un utilisateur |
| **3** | POST | /login | - | {<br>email<br>password<br>remember_me<br>} | 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 | {<br>name<br>} | 201 | Modification de l'utilisateur connecté |

| Numéro | Verbe | URL | Middleware | Paramètres | Résultat | Description |
|---|---|---|---|---|---|---|
| **7** | DELETE | /user | auth:api | - | 200 | Suppression de l'utilisateur connecté. Suppression des informations relatives en cascades |
| **8** | POST | /carparks | auth:api | {<br>latitude<br>longitude<br>address<br>price<br>picture<br>description<br>} | 201 | Création d'une place de parking. Le détenteur est l'utilisateur connecté |
| **9** | PUT | /carparks/{id} | auth:api | {<br>latitude<br>longitude<br>address<br>price<br>picture<br>description<br>} | 201 | Modification d'une place de parking. Le détenteur est l'utilisateur connecté.<br>Vérification que le détenteur de la place de parking soit le même que celui appelant l'API |
| **10** | GET | /carparks/{id} | auth:api | - | 200 | Récupération d'une place de parking |
| **11** | DELETE | /carparks/{id} | auth:api | - | 200 | Suppression d'une place de parking<br>Vérification que le détenteur de la place de parking soit le même que celui appelant l'API |

| Numéro | Verbe | URL | Middleware | Paramètres | Résultat | Description |
|---|---|---|---|---|---|---|
| **12** | POST | /availabilites | auth:api | {<br>start<br>end<br>daily<br>car_park_id<br>} | 201 | Création d'horaire pour une place de parking<br><br>Vérification que le détenteur de la place de parking de l'horaire soit le même que celui appelant l'API |
| **13** | POST | /availabilites/{id} | auth:api | {<br>daily<br>} | 200 | Suppression d'un horaire<br><br>Vérification que le détenteur de la place de parking de l'horaire soit le même que celui appelant l'API |
| **14** | GET | /carparks/{id}/availabilites | auth:api | - | 200 | Récupération des horaires d'une place de parking |
| **15** | POST | /occupants | auth:api | {<br>start<br>end<br>car_park_id<br>} | 201 | Création d'une location |
| **16** | DELETE | /occupants/{id} | auth:api | - | 200 | Suppression d'une location<br><br>Vérification que le détenteur de la location soit le même que celui appelant l'API |

| Numéro | Verbe | URL | Middleware | Paramètres | Résultat | Description |
|---|---|---|---|---|---|---|
| **17** | GET | /user/occupants | auth:api | - | 200 | Récupération des locations de l'utilisateur connecté |
| **18** | GET | /token | auth :api | - | 200 | Vérification du fonctionnement du token |
| **19** | GET | /user/carparks | auth :api | - | 200 | Récupération des places de parking de l'utilisateur connecté |
| **20** | POST | /carparks/search | auth :api | {<br>latitude<br>longitude<br>radius<br>} | 200 | Récupération des places de parking disponibles dans un certain rayon |
| **21** | GET | /carparks/{id}/occupants | auth :api | - | 200 | Récupération des réservations d'une place de parking |

### 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, telles que le routage de requêtes, le mapping d'objets 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 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 conçu puisqu'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 mis en place 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 soi-même les requêtes SQL. Il suffit de générer un objet « modèle » pour qu'il s'enregistre automatiquement dans la base de données. De même lorsque l'on veut 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 simplifiée et très rapide.

Pour sécuriser une route, il suffit de la placer dans le middleware « auth:api ». Lors de la connexion à l'API, Laravel créer un « access token » et l'enregistre dans la base de donnée. Ce token est ensuite fourni à l'utilisateur. Lorsque l'utilisateur essaye ensuite d'accéder à une route qui se trouve dans le middleware « auth:api », il devra ajouter dans sa requête le header « Authorization » et y mettre dedans le token recu précédemment. Le middleware 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 mettre son application Laravel en déploiement, il suffit de copier tous les dossiers et fichiers sur son serveur et de faire pointer son « sous-domaine » sur le dossier « public » de Laravel.

# 6. Développement de l'application

## 6.1 Apprentissage du Framework

L'apprentissage de React Native est spécial. Il est très facile de se lancer dans la création d'une application car le Framework est bien documenté quand il s'agit de faire quelque chose de très simple. Par contre, lorsque il faut créer des fonctionnalités avancées, la documentation de React Native ne suffit plus. Il est nécessaire de rechercher par nous-même des informations auprès de la communauté.

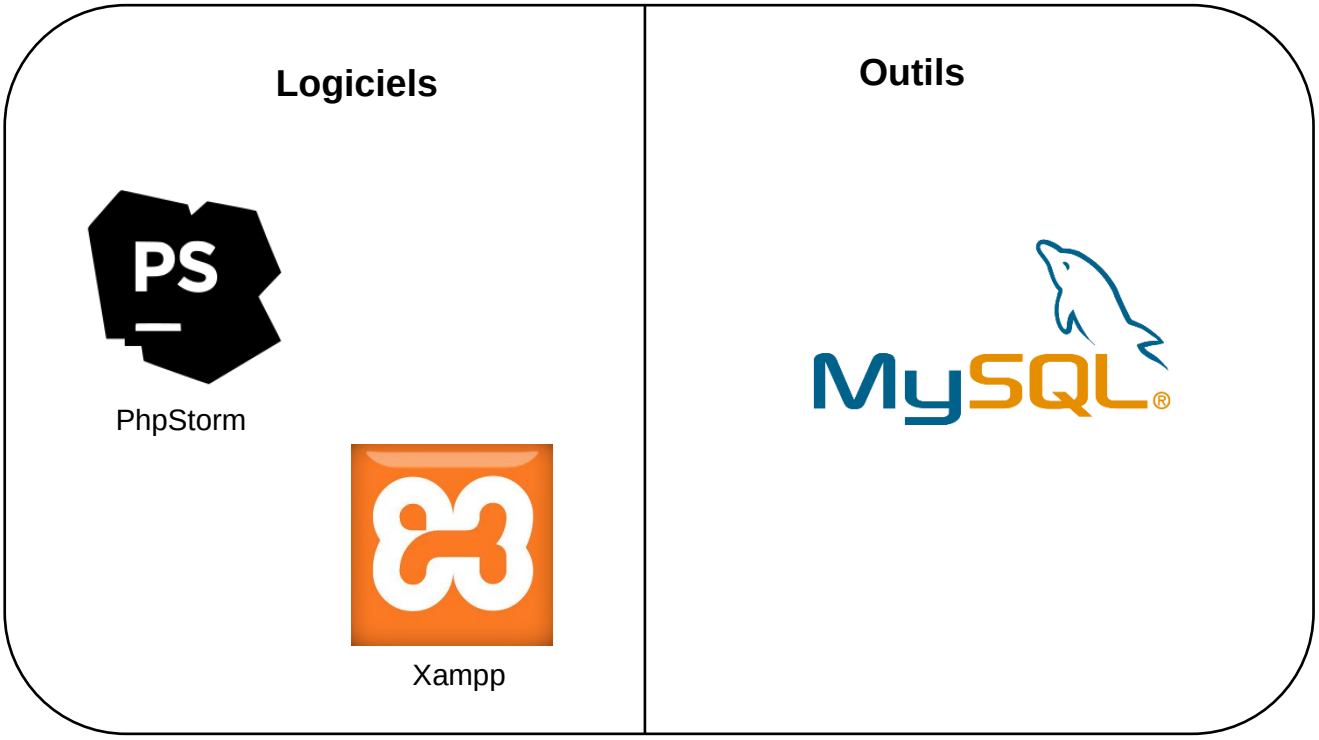
React Native utilise un système de state et de props pour la gestion de ses composants. Les props sont les données reçues du composant parent tandis que le state enregistre les données internes au composant. Lorsque une valeur dans le state change, React Native relance le rendu du composant pour réactualiser les données affichées. Il est assez difficile au début de savoir si une valeur doit être enregistrée dans le state ou bien simplement gardée dans les props. Au fur est a mesure de l'apprentissage de React Native, il va falloir repasser sur son code car on va se rendre compte que la manière dont l'on a créé un composant n'est pas forcement la plus efficace et la plus performante.

Un point important dans la création d'une application React Native est la clarté du code. Un composant comporte toute la logique ainsi que l'interface dans un seul et même fichier. Il est alors facile de se perdre dans son code et de commencer à créer un code « spaghetti ». Il est alors important de se donner, dès le début, des normes de codages pour garder un code propre et facilement lisible / maintenable.

React native utilise un système de props et de state pour la gestion des données mais ne permet pas un échange de données de façon simple entre les différentes pages de l'application. Il est alors nécessaire d'installer un système de gestion de state globale tel que « mobx » ou « redux » pour transférer les données de manière global et efficace. Ces derniers sont plus ou moins simple d'utilisation. Je conseille fortement de débuter avec mobx qui est plus simple de compréhension.

De manière général, React Native permet simplement et rapidement de créer une petite application. Par contre, lorsque l'on va vouloir créer quelque chose de plus conséquent, comme l'application parkaps, il va falloir bien faire attention à normaliser son code et ne pas partir tête baissée dans le code.

## 6.2 Environnement de développement




Pour créer toute l'application ainsi que le back-end Laravel, je n'ai eu besoin que de deux logiciels et un outil, qui est intégré à Xampp.

Xampp permet le lancement d'un serveur php ainsi que l'accès et la création de base de données mysql. Il intègre phpmyadming, qui est une interface graphique simple et efficace de gestion de base de données. Pour créer l'application et le back-end Laravel, j'ai utilisé PhpStorm qui offre plusieurs outils d'aide au développement (auto-complétion, recherche de fichiers, etc.). PhpStorm est le seul logiciel payant que j'ai utilisé mais il est tout-a-fait possible de coder tout le projet sans débourser un seul centime. Utiliser un IDE tel que Atom ou Visual Studio Code qui sont gratuit garantissent le même résultat.

# 7. Rapport de test

| Numéro | Description | Résultat attendu | Résultat obtenu |
|---|---|---|---|
| **1** | Lancement de l'application depuis un téléphone Android en cliquant sur son icone | L'application se lance et affiche la page d'accueil | L'application se lance et affiche la page d'accueil |
| **2** | Création d'un compte utilisateur avec des erreurs :<br>{<br>nom : null,<br>email : 'cecinestpasunemail',<br>password : '123'<br>password_confirmation : '1234'<br>} | Des messages d'erreurs s'affichent pour le nom, l'email, et pour les mots de passes non semblables | Les trois messages d'erreurs s'affichent |
| **3** | Création d'un compte utilisateur sans erreurs | Le compte est créé et un email est envoyé pour le valider | Le compte est créé dans la base de donnée et l'email a été reçu |
| **4** | Validation du compte avec l'email reçu | En cliquant sur le bouton « Confirmer le compte », redirection vers le site avec un message de réussite | Compte validé avec affiche du message de réussite |
| **5** | Connexion avec des erreurs :<br>{<br>email : 'emailinexistant'<br>password : '123'<br>} | Un message d'erreur s'affiche pour l'email | Un message d'erreur s'affiche pour l'email |
| **6** | Connexion sans erreurs | L'utilisateur est redirigé vers la page « map » | L'utilisateur est redirigé vers la page « map » |
| **7** | Recherche de places de parking sans résultats | Un message est affiché pour dire qu'il n'y a pas de places de parking dans ce rayon | Un message est affiché pour dire qu'il n'y a pas de places de parking dans ce rayon |

| Numéro | Description | Résultat attendu | Résultat obtenu |
|---|---|---|---|
| **8** | Recherche de places de parking avec résultats | Chaque place s'affiche avec un marqueur sur la carte et l'écran zoom sur les marqueurs | Chaque place s'affiche avec un marqueur sur la carte et l'écran zoom sur les marqueurs |
| **9** | Voir les infos rapides d'une place en cliquant sur un marqueur | Un bloc en bas de l'écran s'affiche avec les informations rapides de la place | Un bloc en bas de l'écran s'affiche avec les informations rapides de la place |
| **10** | Afficher la page de réservations d'une place en cliquant sur le bouton « réserver » | La page s'affiche avec les disponibilités et horaires de la place | La page s'affiche avec les disponibilités et horaires de la place |
| **11** | Cliquer sur le bouton « réserver » de la page de réservation affiche des sélecteurs de dates et d'heures | Une page modale s'affiche avec les horaires disponibles de la place | Une page modale s'affiche avec les horaires disponibles de la place |
| **12** | Cliquer sur le bouton « enregistrer » de la page modale de sélection d'horaire de réservation enregistre notre réservation et met à jour la dates disponibles | La réservation est faite et les horaires disponibles sont mis-à-jour | La réservation est faite et les horaires disponibles sont mis-à-jour |
| **13** | Lors de l'accès à la page de profil, nos places de parking et réservations sont affichées | Les places et les réservations sont affichées | Les places et les réservations sont affichées |
| **14** | Lorsque le bouton « corbeille » d'une réservation est appuyé, cette dernière est supprimée et la page est mise-à-jour | La réservation est supprimée et la page mise-à-jour | La réservation est supprimée et la page mise-à-jour |
| **15** | Lorsque le bouton « ajouter une place » est cliqué la page d'ajout d'une place s'affiche | La page d'ajout d'une place est affichée | La page d'ajout d'une place est affichée |
| **16** | Le bouton « enregistrer » de la page d'ajout d'une place est appuyé avec des erreurs | Des messages d'erreurs s'affichent | Des messages d'erreurs s'affichent |

| Numéro | Description | Résultat attendu | Résultat obtenu |
|---|---|---|---|
| **17** | Le bouton « enregistrer » de la page d'ajout d'une place est appuyé sans erreurs | La place de parking est enregistrée et on est redirigé vers la page de création d'horaires | La place de parking est enregistrée et on est redirigé vers la page de création d'horaires |
| **18** | Le bouton « ajouter un horaire » de la page horaires est appuyé | Une page modale s'affiche pour sélectionner les dates du nouvel horaire | Un page modale s'affiche pour sélectionner les dates du nouvel horaire |
| **19** | Le bouton « enregistré » de la page modale d'ajout d'horaire est appuyé | L'horaire est ajouté et la page mise-à-jour | L'horaire est ajouté et la page mise-à-jour |
| **20** | Sur la page de profile, le bouton « modification » (la clé anglaise) est appuyé | La page de modification d'une place est affichée avec les informations pré remplies | La page de modification d'une place est affichée avec les informations pré remplies |
| **21** | Le bouton « modifier » est appuyé sur la page de modification d'une place | La place est mise-à-jour avec les nouvelles informations | La place est mise-à-jour avec les nouvelles informations |
| **22** | Le bouton « modifier les horaires » est appuyé sur la page de modification d'une place | On est redirigé vers la page des horaires avec les horaires déjà existant affichés | On est redirigé vers la page des horaires avec les horaires déjà existant affichés |
| **23** | Le bouton « supprimer » est appuyé sur la page de modification d'une place | La place est toutes les données relatives à cette place sont supprimées | La place est toutes les données relatives à cette place sont supprimées |
| **24** | L'icône « corbeille » sur la page horaire est appuyé | L'horaire sélectionné est supprimé | L'horaire sélectionné est supprimé |
| **25** | L'icône « paramètres » de la page de profil est appuyé | La page paramètre s'affiche | La page paramètre s'affiche |

| Numéro | Description | Résultat attendu | Résultat obtenu |
|---|---|---|---|
| **26** | Le bouton « se déconnecté » de la page de paramètre est appuyé | L'utilisateur est déconnecté et redirigé vers la page de connexion | L'utilisateur est déconnecté et redirigé vers la page de connexion |
| **27** | Le bouton « Supprimer son compte » est appuyé sur la page de paramètres | Un message de confirmation est affiché et si l'utilisateur clique sur oui, son compte est supprimé | Un message de confirmation est affiché et si l'utilisateur clique sur oui, son compte est supprimé |
| **28** | Le bouton « modifier le nom d'utilisateur » de la page paramètres est appuyé | Une page modale est affichée pour écrire son nouveau nom d'utilisateur | Une page modale est affichée pour écrire son nouveau nom d'utilisateur |
| **29** | Le bouton « enregistrer » de la page modale de modification du nom d'utilisateur est appuyé | Le nom d'utilisateur est modifié et la page mise-à-jour | Le nom d'utilisateur est modifié et la page mise-à-jour |

L'application est disponible sur le Play Store sous le nom « Parkaps ». De plus, la totalité de mon Bachelor, soit le mémoire, l'application et le serveur Laravel, sont disponibles sur le répertoire GitHub suivant : https://github.com/schupbachl/Bachelor-HEG-2018

# 8. Conclusion

Les Frameworks de développement mobiles cross-platform sont, actuellement, la solution la plus efficace pour créer et déployer une application sur le Play Store et l'App Store. Ils permettent un développement uniforme entre les différentes plateformes cibles. Chaque Framework a ses avantages et ses inconvénients et il est important de les comparer et les étudier pour choisir celui qui convient le mieux à son projet. De manière générale, un Framework proposant des performances accrues aura une possibilité de customisation de ses pages plus faible et inversement.

Pour mon Bachelor, j'ai expliqué trois Frameworks : Ionic, Xamarin et React Native. Après avoir comparé les avantages et inconvénients de chacun d'eux, il m'a semblé évident que React Native était le Framework le plus approprié à la création d'une application d'échange de places de parking. Des performances proches du natif, tout en proposant une customisation suffisante et un langage de programmation connu et simple. Au fur et à mesure que je créais l'application, je me suis rendu compte que ce Framework comblaits totalement mes attentes. Il y a néanmoins un aspect négatif du Framework dont je ne m'attendais pas à rencontrer. Un manque cruel de mise-à-jour du Framework. En effet, React Native utilise le moteur de production Gradle pour créer et compiler les projets en APKS. React Native utilise toujours la version 2 de Gradle alors que ce dernier a déjà sorti des versions supérieures à 4. Ceci limite grandement la possibilité d'ajout des libraires tierces au projet car elle n'utilisent pas forcement la même version de Gradle que nous. De la part d'une entreprise aussi grande, on peut se poser des questions quant à la pérennité de son Framework à l'avenir.

Il est donc essentiel de penser à l'importance et la taille du projet avant de commencer. L'utilisation de Xamarin pour les gros projets est fortement recommandé car, étant donné que c'est un Framework propriétaire embarquant son propre environnement de runtime et de compilation, le problème de concordance de version ne se posera pas.

Ionic, quant à lui, sera parfait pour des petits projets solitaires.

Je pense que ce travail explique les points importants à prendre en considération pour choisir le Framework le plus adapté à son projet. Ma démonstration du Framework React Native avec l'application « Parkaps » le prouve bien.

# Bibliographie

ALTEXSOFT, 2018a. 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/.

ALTEXSOFT, 2018b. 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, 2018c. 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/.

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.

PHP, Non daté. PHP: Supported Versions. In : [en ligne]. Non daté. [Consulté le 24 août 2018]. Disponible à l'adresse : http://php.net/supported-versions.php.

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.

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/.

XAML.FR, 2012. Description de XAML. In : [en ligne]. 2012. [Consulté le 24 août 2018]. Disponible à l'adresse : https://www.xaml.fr/xaml-description.html.