ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

DÉVELOPPEMENT D'UNE APPLICATION MOBILE CONCERNANT LA PHYSIOTHÉRAPIE DE L'ÉPAULE ET DU GENOU

RAPPORT DE PROJET PRÉSENTÉ
COMME EXIGENCE PARTIELLE À L'OBTENTION DE
LA MAITRISE EN GÉNIE DES TECHNOLOGIES DE L'INFORMATION

PAR
Quentin MURET

MONTRÉAL, MAI 2019

PRÉSENTATION DU JURY

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

Professeur Alain April, directeur de projet
Département du génie logiciel et des TI's à l'École de technologie supérieure

Professeur Abdelaoued Gherbi, président du jury
Département du génie logiciel et des TI's à l'École de technologie supérieure

REMERCIEMENTS

C'est une réelle chance pour moi d'avoir pu effectuer ce projet, qui m’a permis d’approfondir mes compétences en développement logiciel.

Je remercie Alain April, professeur au département de Génie logiciel de l’ÉTS et également superviseur du projet, pour m’avoir encadré et accompagné durant toutes les étapes du projet, avec patience et attention.

Je remercie également George Demirakos, client du projet, pour m’avoir proposé ce projet et accepté de me rencontrer lorsque j’en avais besoin.

Un grand merci également à Julie Vincent et Mathieu Crochet pour leurs précédents travaux qui m’ont permis d’avoir une base de travail.

DÉVELOPPEMENT D'UNE APPLICATION MOBILE CONCERNANT LA PHYSIOTHÉRAPIE DE L'ÉPAULE ET DU GENOU

QUENTIN MURET

RÉSUMÉ

Ce projet de recherche appliquée, à la maitrise en génie des technologies de l'information de l'École de Technologies, vise à redévelopper une application mobile nommée FixMyShoulder. Cette application mobile, fondée sur un livre de physiothérapie de l'épaule, a été développée pour le physiothérapeute George Demirakos.

L'application a pour but de permettre à George Demirakos de partager ses connaissances et d'offrir des instructions claires et imagées des différents traitements de l'épaule.

Un premier prototype de l'application a été développé en 2012 par Julie Vincent à l'aide de la plateforme technologique iOS. En 2013, une deuxième version a été développée, par Mathieu Crochet, pour y ajouter l'accès sur la plateforme Androïde. Par la suite des étudiants de maitrise ont développé une plateforme de synchronisation des mises à jour pour ces deux plateformes mobiles.

L'objectif de ce projet de recherche appliquée est de concevoir une version multiplateforme de l'application, permettant d'avoir un seul et même code source compatible à la fois avec la plateforme Androïde et iOS. De plus, le nouveau prototype devrait inclure le contenu d'un nouveau livre sur la physiothérapie du genou récemment publié.

Ce rapport présente les différences étapes concernant la sélection de la technologie, la conception et la réalisation du projet et est divisé principalement en trois chapitres. Le premier chapitre présente une revue de la littérature, qui consiste à effectuer une analyse synthèse des technologies actuelles en matière de développements d'applications mobiles multiplateformes. À la fin de ce premier chapitre, les technologies à la fois les plus adaptées pour le projet, mais aussi celles qui ont le plus d'avenir seront identifiées.

Le deuxième chapitre comporte la synthèse du travail de conception et de développement d'une troisième version du prototype logiciel de l'application mobile. Le troisième chapitre décrira l'expérimentation et les commentaires de l'utilisateur concernant l'application. Le rapport se terminera par une conclusion et les prochaines améliorations possibles.

Mots-clés: IONIC, CORDOVA, ANGULAR, ANDROÏDE, IOS, APPLICATIONS MOBILE

DEVELOPMENT OF A MOBILE APPLICATION FOR SHOULDER AND KNEE PHYSIOTHERAPY

QUENTIN MURET

ABSTRACT

This applied research project, part of the master degree in Information Technology at the École de Technologies Supérieure, aims to redevelop an existing mobile application named FixMyShoulder. This mobile application is based on a physiotherapy book aimed at the shoulder therapy, written by George Demirakos, a well-known Montréal physiotherapist.

The current application is intended to allow George Demirakos to share his knowledge and offer clear instructions, images and videos of various recommended treatments for the shoulder.

A first prototype of this application, for the iOS platform, was developed, in 2012, by Julie Vincent. In 2013, a second version was developed by Mathieu Crochet to add access to the application by Androïde devices. Next, a number of master students developed an application aimed at synchronizing changes on the two platforms.

The objective of this applied research project is to design a multi-platform version of the application, allowing one source code to operate and be compatible with both the Androïde and iOS platform. As well the new prototype should include the contents of a new book about knee physiotherapy recently published.

This report presents the different stages of the project technology selection, design and implementation and is divided into three main chapters. The first chapter presents the state of the art in current multi-platform mobile technologies. At the end of this first chapter, the technologies that are both the most suitable for the project and those with the most promising future will be identified.

The second chapter summarizes the design and development decisions for this third version of the mobile application's. Finally, the third chapter will describe the experiment and the user's comments about the application. The report end by a conclusion and future work.

Keyword: IONIC, CORDOVA, ANGULAR, ANDROID, IOS, MOBILE APPLICATIONS

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

ALGORITHMES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

CSS:	Cascading Style Sheets
GPL:	Licence publique générale
HTML:	Hypertext Markup Language
iOS:	iPhone Operating System
SDK:	Software Development Kit
SOM:	Système d'Opération Mobile, par exemple: iOS, Androïde
XML:	Extensible Markup Language
FAQ:	Foire aux questions

INTRODUCTION

Lorsque l'on souhaite développer une application mobile, il est aujourd'hui indispensable d'avoir une compatibilité pour les plateformes iOS et Androïde, dans le but d'une part de cibler un public plus large, et d'autre part de ne pas délaisser une partie de la population [1].

Développer des applications mobiles natives, qui requièrent une version pour Androïde et une version pour iOS, est plus complexe, augmente les coûts, l'effort et la maintenance [2]. Au cours des dernières années, un nouveau concept prend de plus en plus d'ampleurs: le développement d'applications mobiles multiplateformes [4]. Cette approche permet au développeur d'implémenter un seul et même code source et de l'utiliser à la fois sur Androïde, mais aussi sur iOS. Tel qu'introduit dans le résumé de ce rapport, le but de ce projet de recherche appliquée est de redévelopper le prototype d'application mobile native FixMyShoulder, en utilisant une technologie multiplateforme. Le prototype d'application résultant doit être compatible à la fois pour les téléphones intelligents et les tablettes.

FixMyShoulder s'inspire des contenus d'un livre de physiothérapie de l'épaule, écrit par le physiothérapeute George Demirakos, client du projet. La nouvelle version multiplateforme de l'application devra inclure les sections suivantes :

- Des vidéos (c.-à-d. les exercices proposés dans les livres);
- Une section programme, permettant à l'utilisateur de créer son propre programme d'exercices, incluant des rappels par notifications;
- Une section boutique, qui proposera à la vente les livres écrits par George Demirakos;
- Une section d'information (c.-à-d. nous contacter/Courriel/Adresse).

Ce rapport décrit les différentes étapes réalisées dans le cadre de ce projet de maîtrise, supervisé par le professeur Alain April et par le physiothérapeute George Demirakos. Il présente l'introduction, la revue de littérature, les travaux de conception et de réalisation de l'application, ainsi que l'analyse des résultats ainsi que des recommandations.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

1.1 Introduction

En dix ans, l'arrivée des téléphones intelligents, puis des tablettes et des montres intelligentes a changé notre façon de communiquer. Les plateformes utilisées par une même personne sont maintenant variées. Lors du développement d'une application logicielle, il est incontournable de prendre en compte ce nouveau contexte afin de concevoir une application accessible par toutes ces plateformes.

Actuellement, les deux systèmes d'opération prédominants, sur téléphones intelligents, sont Androïde et iOS. Selon la littérature [5] [14], il existe aujourd'hui cinq approches principales de développement d'une application mobile:

1) **Le développement d'application native** : Cette approche vise à développer l'application avec le langage natif du système d'exploitation ciblé, par exemple, une utilisant le langage de programmation Java ou Kotlin pour Androïde ou en langage Swift (ou Objectif C) pour iOS. Les applications développées en langage natif sont également appelées « applications endémiques » [14]. Cette approche à l'avantage d'être la plus optimisée pour chaque système d'exploitation, ce qui permet d'avoir des applications plus rapides avec des performances plus élevées. Cependant, l'inconvénient de cette approche est qu'elle nécessite deux développements simultanés (c.-à-d. deux codes sources différents) : un pour Androïde et un pour iOS, ce qui a pour conséquences d'engendrer des coûts de développement et de maintenance plus élevés ;
2) **Le développement d'une application Web (Web app)** : Cette approche vise à développer une application qui opère sur un navigateur Web. Il s'agit ici d'un site Web adapté pour mobile. L'inconvénient de cette approche est qu'elle ne nous permet

pas d'accéder aux fonctionnalités du téléphone, comme l'accès aux différents capteurs ou le stockage de données, ce qui était possible avec la première approche. De plus, Apple ne permet pas la publication de ce type d'application sur l'App Store [14] ;

3) **Le développement multiplateforme** : Cette troisième approche est la manière la plus récente pour développer une application mobile. Le but est de développer l'application mobile, à partir d'un seul code source, adapté à chacune des plateformes, rendant l'application à la fois compatible Androïde et iOS. Contrairement à l'approche précédente (c.-à-d. l'application Web), les applications multiplateformes permettent d'accéder aux fonctionnalités du téléphone. Ce type de développement a les avantages suivants [2] [3]:
    - Un seul et même code est implémenté pour les plateformes Androïde et iOS donc une seule équipe de développeur peut être utilisée pour travailler dessus;
    - Les coûts de maintenance de l'application diminuent considérablement;
    - Les entreprises n'ont plus besoin d'investir dans différents outils et technologies requis par chacune des plateformes;
    - Le développement d'applications est beaucoup plus rapide, ce qui permet au produit d'atteindre le marché plus tôt qu'avant;
    - Ce type de développement garantit que l'aspect global de l'application soit uniforme sur chacune des plateformes.

4) **Le développement d'applications interprétées** [14] : Les applications interprétées utilisent une manière de développement différente pour l'interface utilisateur et pour la logique. Elles utilisent des « widgets » natifs spécifiques à la plateforme pour interagir avec l'utilisateur, tandis que la logique est développée de manière indépendante du SDK natif ;
5) **Le développement d'applications générées** [14] : Les applications générées sont développées avec leurs langages natifs respectifs, à partir d'une base de code commun. Le code commun est alors traduit dans le langage natif de chaque plateforme. Cette solution n'est aujourd'hui pratiquement plus utilisée et les cadriciels l'utilisant ne sont désormais plus actifs.

Les applications Web et celles développées avec une technologie multiplateforme sont également appelées « applications pandémiques » [14].

Étant donné les avantages proposés par la troisième approche, décrite ci-haut, il serait préférable que le prototype d'application FixMyShoulder soit redéveloppé à l'aide d'une technologie multiplateforme récente. Il existe à ce jour deux types de développement multiplateforme [5] [8]:

1. Le développement multiplateforme natif. Ce type de développement va générer, à partir du code principal, des codes natifs pour chaque système d'opération mobile (SOM). Cette méthode est pratique, car elle permet de réaliser des applications multiplateformes qui offrent les mêmes performances qu'une application native. Elle est cependant limitée à ce que l'outil est capable de traduire en langage natif;
2. Le développement multiplateforme hybride. Il repose lui aussi sur un seul et même code source, mais à partir d'un langage plus proche du Web, qui va ensuite être interprété sur chacun des SOM. Ce mode de développement à l'avantage, grâce à son langage, d'être beaucoup moins limité que le développement multiplateforme natif. En contrepartie, les performances ne sont pas aussi optimales qu'avec une application native, puisque ces applications doivent être en quelque sorte interprétées par le moteur Web du système d'exploitation pour pouvoir être affichées. Ce moteur Web nécessite en l'occurrence beaucoup de ressources.

Le développement multiplateforme natif et hybride comporte donc chacun leurs avantages et inconvénients.

Le but de ce chapitre consiste à identifier, dans la littérature, les différentes technologies de développement mobile multiplateforme afin d'identifier celle qui est à la fois la plus adaptée pour le projet, mais aussi celle qui a le plus d'avenir selon les experts. En effet, il est très important d'étudier les publications concernant une technologie avant de considérer son utilisation, car une technologie qui perd en popularité ou n'est plus maintenue peut contenir de plus en plus de failles de sécurités, n'est pas mise à jour rapidement, perd sa compétitivité et devient de plus en plus difficile à maintenir.

Au début de cette recherche, il est important d'effectuer une étude des outils de développement multiplateforme mobile les plus populaires d'aujourd'hui. D'après quelques sites de comparaisons [6][7][8], les technologies, de ce domaine, les plus utilisés aujourd'hui sont les suivantes : Xamarin, Ionic, QT, React-Native et PhoneGap. Ce sont ces technologies qui ont été retenues pour cette étude comparative sommaire. Une source d'information comparative populaire, qui permet d'identifier rapidement les tendances d'utilisation d'une technologie, est Google Trend (trends.google.fr). Ce site d'information fournit la comparaison suivante, basée sur des fréquences de recherches Google sur les 12 derniers mois:

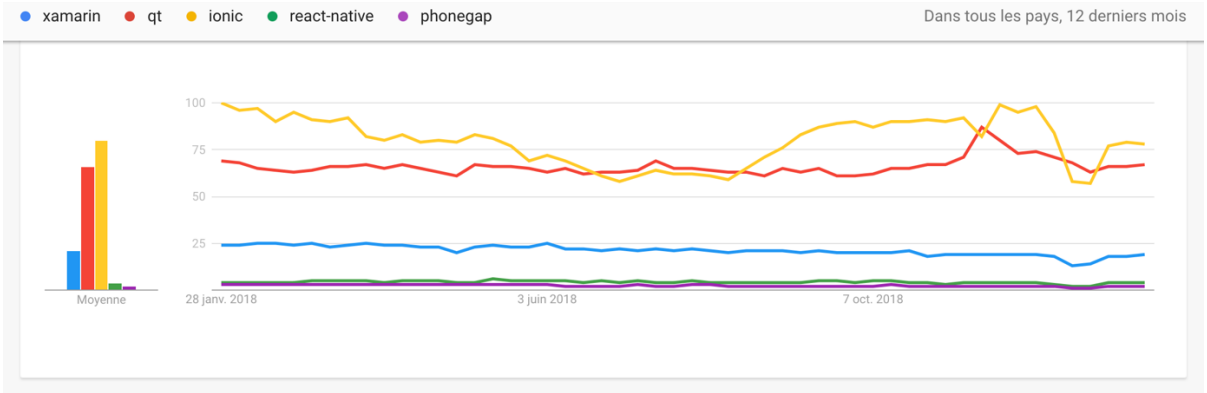

Figure 1.1 Comparaison des technologies actuelles

Selon Google Trend, en moyenne, Ionic atteint un score de 80, contre 67 pour QT, 21 pour Xamarin et moins de 5 pour React native et PhoneGap.

Le terme Ionic est donc clairement plus consulté par rapport à ses concurrents. Nous pouvons remarquer que les cadriciels React native et PhoneGap ne sont presque plus recherchés sur le Web. Ceci a une incidence sur la popularité de ces technologies. Le schéma ci-dessous affiche la répartition des recherches par région:

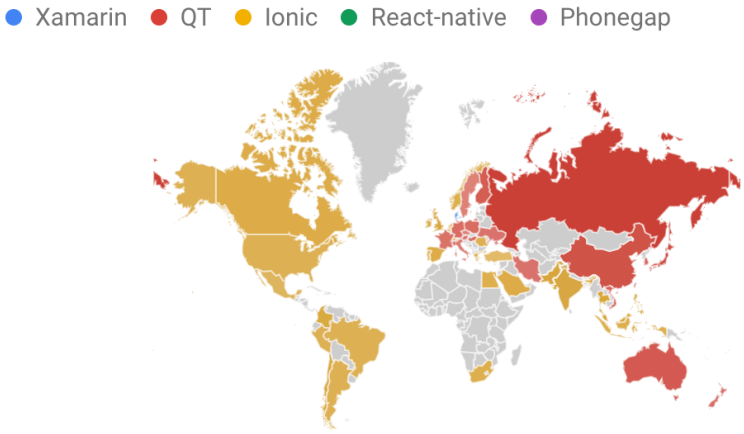

Figure 1.2 Comparaison de la répartition par région

On peut remarquer qu'Ionic est particulièrement recherché en Amérique, tandis que QT domine les statistiques de recherche en Asie et Australie. Xamarin, de son côté, est tout de même en tête des recherches au Danemark :

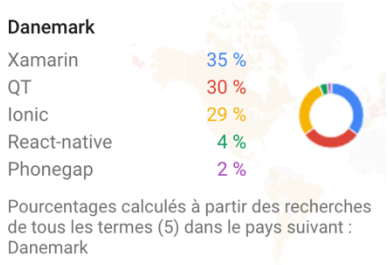

Figure 1.3 Statistique des recherches au Danemark

La suite de cette section propose une étude comparative de ces quatre technologies: Xamarin, Ionic, QT et React native.

Xamarin est un cadriciel C # alors React Native et Ionic sont basés sur les technologies du Web [10]. La différence principale entre Ionic et React est que React Native fonctionne similairement à Xamarin : l'interface graphique est développée en utilisant le langage du Web (c.-à-d. HTML, CSS et JavaScript), qui est ensuite converti en éléments natifs. Ce sont

donc des cadriciels multiplateformes natifs. Au contraire, Ionic n'utilise aucun « widget » natif, mais affiche directement une page Web, ce qui permet d'avoir une très bonne portabilité. Il s'agit ici d'un cadriciel multiplateforme hybride. Chaque technologie utilise ainsi une approche légèrement différente. Le tableau ci-dessous résume ces différences :

	Xamarin	**Ionic**	**React Native**	**QT**
Langage	C#	HTML / SCSS / TypeScript	HTML / CSS / JavaScript	C#
Éléments natifs	Oui	Non	Oui	Non

Tableau 1.1 Caractéristiques des cadriciels

1.2 Xamarin

Un atout majeur du cadriciel de développement d'applications mobiles Xamarin est qu'il utilise des « widgets » natifs pour afficher l'interface utilisateur [10]. Il propose deux possibilités dans la réalisation d'une interface utilisateur:

- La concevoir de la même façon que l'on procèderait pour un développement natif: de manière spécifique à chaque plateforme;
- En utilisant Xamarin.Forms, une boîte à outils complète d'éléments d'interface utilisateur.

L'utilisation de la manière native pour mettre en forme l'interface utilisateur est aussi rapide que de développer une application native iOS ou Androïde. Xamarin.Forms a comme avantage de permettre de réaliser une seule mise en page pour les deux plateformes, qui utilisera ensuite des widgets natifs.

1.3 Ionic

Ionic est un cadriciel de développement d'applications mobiles disponible librement qui se concentre sur l'interface utilisateur et l'interaction personne-machine. L'entreprise CruxLab [10] décrit bien ce cadriciel qu'ils utilisent fréquemment pour développer des sites mobiles : « Ionic offre la possibilité de créer un code qui fonctionnera à la fois sur les plateformes Androïde et iOS, en utilisant les technologies du Web (c.-à-d. HTML5, CSS3 et JavaScript). Pour cela, il utilise le cadriciel Cordova pour accéder aux fonctionnalités natives du téléphone comme les appels ou l'accès à la caméra, et AngularJS, un autre cadriciel permettant d'optimiser l'utilisation du code pour toute la partie Web de l'application. »
En se référant à l'article [15], Ionic permet un développement simple, rapide, des coûts de développement faible et un cycle de développement court.

Grâce aux avantages qu'apporte le développement multiplateforme, selon le site officiel d'Ionic (https://ionicframework.com/), à ce jour plus d'un million de développeurs utilisent Ionic, et par de grandes entreprises, comme IBM ou MasterCard.

Afin de faciliter le développement de l'interface utilisateur, Ionic intègre la gestion des évènements tactiles (appuis d'un bouton par exemple), ainsi qu'une fonction permettant de tester l'application dans le navigateur Web de l'ordinateur directement. Cela permet un gain de productivité par rapport au développement d'une application native, ou il faut systématiquement compiler le code et le mettre sur le téléphone (ou compilateur) pour le tester.
Enfin, Ionic intègre de nombreux éléments de désign comme des barres de navigations ou des messages d'alertes, utilisables très facilement en se référant à la documentation officielle d'Ionic. Selon l'article [15], le temps de développement d'une application Ionic est beaucoup plus court que celui des applications natives et le code est relativement facile à maintenir.

1.4 QT

Les informations présentées dans cette section proviennent de la documentation présentée sur site Web officiel de QT (https://www.qt.io/). QT est un cadriciel qui offre des composants d'interface graphique, de connexions réseau, d'accès aux données, de gestion des fils d'exécution ou encore d'analyse XML. La particularité de QT est qu'il permet aux applications qui n'utilisent que ses composants d'être compatible avec plusieurs systèmes d'exploitation avec une recompilation du code source. Il s'agit d'un outil qui permet à la fois le développement multiplateforme mobile et bureautique.

Il est disponible sur une large gamme de systèmes d'exploitation comme Unix, Mac OS, Windows, iOS et Androïde. Il existe deux façons de développer une application avec QT:

- En payant une licence commerciale propriétaire;
- En développant un programme libre qui utilise une Licence publique générale (GPL).

QT est utilisé dans de larges domaines tels que les systèmes embarqués, les interfaces utilisateur, l'internet des objets, l'automobile, l'automatisation, les décodeurs numériques, le médical ainsi que le développement mobile.

1.5 React Native

React Native est une bibliothèque JavaScript libre qui nous permet de créer des applications mobiles en utilisant uniquement les technologies du Web (par exemple : Ionic) [10]. L'avantage de React Native est qu'il permet d'utiliser des widgets natifs lorsque l'on a besoin d'optimiser quelques aspects de l'application. Ces éléments natifs sont utilisés par l'intermédiaire de JavaScript et React. Il est ainsi possible de développer une partie de l'application avec du code React, et une autre avec un langage natif. React offre néanmoins de moins bonnes performances que Xamarin ou Ionic.

Pendant le développement d'une application React Native, il est également possible de recharger notre application instantanément, sans devoir la recompiler systématiquement. À ce jour, de nombreuses applications connues ont été développées avec React Native comme Facebook, Skype ou Uber [11].

1.6 Comparaison des cadriciels de développement mobile

L'article [13] présente un cadriciel d'évaluation des outils de développement multiplateformes. Ce dernier présente des tests qui ont été effectués sur différents cadriciels et démontrent que leurs différences peuvent avoir un impact important, tant négatif que positif, sur le développement d'une application mobile.

L'article démontre qu'en général, les outils de développement multiplateformes présentent soit des problèmes de performances, soit un manque de fonctionnalités.
Autrement dit, plus un cadriciel est performant, plus il est limité en fonctionnalités, et plus un cadriciel possède de fonctionnalités, moins il est performant. Cette hypothèse a été démontrée également dans ce document. L'étude de Xamarin a démontré que ce cadriciel possède l'interface utilisateur la plus rapide, car elle permet d'utiliser directement les outils natifs d'iOS et Androïde [10]. Le processus de développement est par ailleurs similaire à celui de Xcode et Androïde Studio. Cependant, ce cadriciel est limité à ce que l'outil est capable de traduire en langage natif. Inversement, Ionic utilise le langage du Web, ce qui permet d'être moins limité que Xamarin, mais a pour conséquence d'avoir une perte de performance.

Cependant, comme énoncé dans l'article [13], IOS et Androïde ont tendance à améliorer leurs navigateurs Web, et a les rentre de plus en plus conformes aux normes. Cela permettra aux cadriciels de développements multiplateformes hybrides d'améliorer leurs performances dans les années à venir. Pour résumer, l'interface utilisateur la plus rapide est fournie par Xamarin, car elle permet d'utiliser directement les outils natifs d'iOS et Androïde [10]. Ce cadriciel est cependant limité à ce que l'outil est capable de traduire en langage natif. De

plus, le résultat de l'étude effectuée avec Google Trends, présentée ci-dessus, démontre que ce cadriciel mobile est nettement en nette perte de vitesse.

React Native offre un cadre vraiment intéressant et différent (c.-à-d. de par son fonctionnement) du développement sous Xamarin, QT ou Ionic. Il offre cependant de moins bonnes performances que ses deux concurrents et n'est pas très populaire, tel que démontré lors de l'investigation de popularité réalisée à l'aide de Google Trends.

QT est un cadriciel très intéressant, car il permet une bonne portabilité des applications qui n'utilisent que ses composants. Cependant, si QT est un cadriciel plutôt populaire pour le développement d'application bureautique, ce n'est pas le cas pour développement mobile. Le développement mobile avec QT n'est en effet pas très courant. Il est possible de le remarquer avec les statistiques Google Trends, présentées ci-dessous. En effet, QT, qui est utilisé dans de nombreux domaines, a obtenu un score de fréquence de recherche de 67, alors qu'Ionic, qui n'est utilisé que dans le développement mobile, a obtenu un score de 80.

Enfin, Ionic utilise le langage du Web, ce qui permet d'être moins limité que Xamarin, React Native et QT. Cela permet également d'avoir une très bonne portabilité, des rechargements très rapides, mais a pour conséquence d'avoir une perte de performance. Cependant, comme énoncé dans l'article [15], « avec la popularité des téléphones intelligents, la vitesse du processeur, la capacité de mémoire, la puissance de calcul et les problèmes de compatibilité des navigateurs des terminaux ne cessent de s'améliorer ». Grâce à ces améliorations, les téléphones d'aujourd'hui sont de plus en plus rapides et les pertes de performances des applications hybrides sont de moins en moins visibles par les utilisateurs. Il est à noter que toutes ces technologies offrent une utilisation gratuite.

1.7 Choix de la technologie pour le projet

Un compromis entre performance et processus de développement moderne mène à la décision de choisir la technologie Ionic pour ce projet de recherche appliquée. Ionic permet

d’obtenir une application avec une interface utilisateur presque similaire aux interfaces natives, en utilisant les technologies du Web. Cela permet aussi de bénéficier d’une très large documentation. Le site Web codepen.io offre accès à de nombreux exemples de codes sources libres d’éléments et des applications graphiques réutilisables.

D’autre part, en plus des spécificités techniques, il faut également prendre en compte la popularité du cadriciel. Tel que proposé par la firme de consultation Zirous, spécialisée en recommandation de technologies [12], « Un cadriciel actif et populaire est important, car il offre aux développeurs davantage d'outils, de communautés et de ressources à exploiter au cours du développement. Cela signifie également que des correctifs sortiront régulièrement pour résoudre les défauts et les failles de sécurité ».

Selon les résultats fournis par Google Trends, par rapport à ses concurrents, Ionic se classe bien en avance, sur le plan de la fréquence de recherche. QT obtient également un bon score, mais il ne faut pas oublier que ce cadriciel est utilisé dans sept domaines différents (donc dans six qui ne concernent pas le développement mobile). Du plus, une étude réalisée par g2crowd [9] et basée sur des avis utilisateur démontre que sur sept critères, Ionic est mieux noté que QT sur six d’entre eux (voir annexe 1 de ce rapport).

Pour toutes ces raisons, et en plus de ses performances et de son processus de développement moderne, Ionic caractérise sans doute la meilleure solution actuelle en matière de développement mobile multiplateforme. C’est pourquoi c’est cette technologie qui a été retenue pour la rénovation du projet FixMyShoulder.

CHAPITRE 2

DÉVELOPPEMENT DE L'APPLICATION

2.1 Introduction

Ce chapitre présente le contenu de mon travail original de recherche. Il décrit les différentes étapes d'avancement du redéveloppement de l'application FixMyShoulder.

FixMyShoulder est une application mobile qui se base sur un livre de physiothérapie des épaules, du même nom. Comme écrit précédemment dans ce rapport, une première version de l'application a été développée en 2013 par Mathieu Crochet sur la plateforme Androïde, et en 2012 sur la plateforme IOS par Julie Vincent. Cette application a pour but de permettre au physiothérapeute George Demirakos de partager ses connaissances et d'offrir une documentation claire et simplifiée sur les différents traitements de l'épaule possibles. L'objectif de ce projet est de développer une version multiplateforme de l'application, c'est-à-dire une seule application compatible avec les plateformes Androïde et IOS.

Le développement de l'application s'est déroulé selon les étapes suivantes: l'analyse de l'existant, l'élicitation des exigences et des contraintes, le développement d'une stratégie de travail, et tout le travail réalisé en matière de conception et d'implémentation.

2.2 Analyse de l'existant

Cette analyse visait à étudier le travail effectué par Mathieu et Julie sur le développement des précédentes versions de l'application. Le but était d'étudier le cahier des charges et les besoins du client à l'époque, ainsi que la technologie utilisée afin d'y déterminer les faiblesses de l'approche utilisée.

2.2.1 Cahier des charges et besoins du client

L'application contenait une barre de menu composée de quatre onglets :

- **Introduction** : C'est la page d'accueil de l'application. Elle contient une brève description du livre Fix My Shoulder, ainsi qu'une présentation du reste de l'application ;
- **Chapitres** : Cet onglet présente la liste des chapitres du livre, leurs sous-chapitres, ainsi que leurs contenus ;
- **Vidéos** : Cet onglet contient une illustration de certains exercices ;
- **Informations** : On retrouve dans cette section des informations concernant l'auteur du livre, les copyrights et un moyen de contacter l'auteur.

Le diagramme suivant représente une vue d'ensemble de l'application. Il a été réalisé par Julie Vincent :

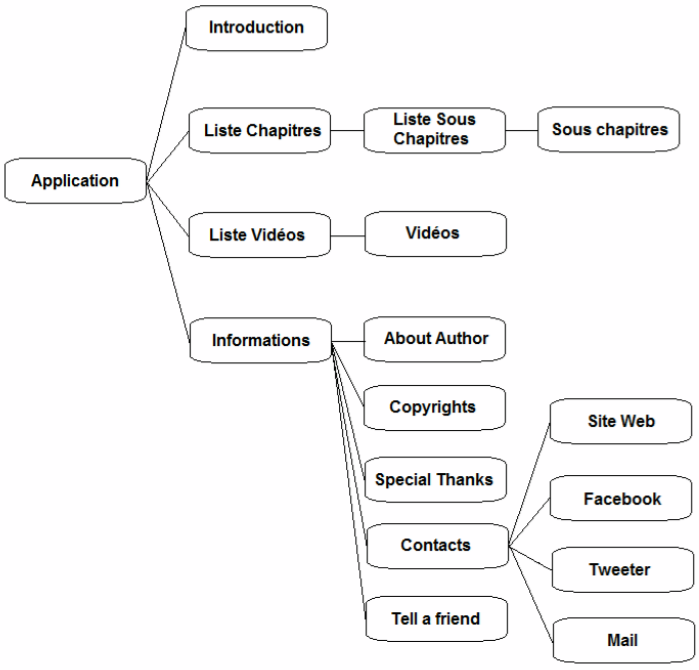

Figure 2.1 Vue d'ensemble de l'application originale

2.2.2 Technologie utilisée

Les premières versions originales de l'application, que ce soit sur Androïde ou IOS, ont été développées avec le langage natif du système d'exploitation ciblé. Ces deux langages étaient à l'époque le java pour Androïde, et l'objectif C pour IOS. Comme décrit dans le chapitre 1 de ce rapport, cette approche à l'avantage d'être la plus optimisée pour chaque système d'exploitation, car elle permet d'avoir de bonnes performances, mais nécessite deux développements parallèles (sois deux codes sources différents). Étant donné que le coût moyen de maintenance d'une application caractérise 80% de son coût total (développement + maintenance), nous pouvons facilement démontrer que cette approche n'est pas la plus économique sur le long terme.

2.3 Exigences et contraintes

L'élicitation des exigences et des contraintes pour la nouvelle version de l'application a été faite par George lui-même. Ce dernier a communiqué ses exigences lors d'une rencontre à ses bureaux, au tout début du projet. Il avait déjà réfléchi aux exigences qu'il souhaitait avoir pour cette nouvelle version de l'application. Il a revu complètement les fonctionnalités de l'application par rapport à la version originale.

2.3.1 Exigences fonctionnelles

Voici la liste des exigences fonctionnelles :

EF-1: Afficher des articles d'information. FixMyShoulder doit permettre d'afficher des articles d'informations sur les sujets suivants :

- Tendinitis;
- Acupuncture/ Yoga / Pilates / Nutrition;
- X-ray / MRI;
- Best tips in practice;
- Posture.

EF-2: Lire des vidéos d'exercices d'entraînements. FixMyShoulder doit permettre de lire des exercices d'entraînements sous format vidéo sur les sujets suivants :

1. strength exercises

- Lying ER (external rotation);
- Airplane;
- Alphabet;
- Isometric flex abduction (external rotation);
- Superman;
- Prone (external rotation);
- Overhead alphabet.

2. Range of motion

- Stick exercises:
  - Front;
  - Side;
  - Circles.
- Thoracic extension;
- Pendulum;
- Wall climb.

3. Gym routine

- Rowing;
- Face pulls;
- Cable (external rotation);
- Do's and don'ts.

EF-3: Permettre à l'utilisateur de créer un programme d'exercices. FixMyShoulder doit contenir une section *My program* qui contient les exercices que l'utilisateur aura inclus dans son programme. Cette section doit contenir un calendrier pour permettre de planifier ses exercices et d'ajouter des rappels d'exercices par notifications.

EF-4: Permettre au client d'exposer ses livres en vente. FixMyShoulder doit contenir une section permettant à George de présenter ses livres en vente.

EF-5: Permettre au client d'avoir un moyen de rémunération via l'application. FixMyShoulder doit proposer du contenu accessible à tous gratuitement, et proposer en parallèle du contenu accessible uniquement via l'abonnement à un compte Premium payant.

EF-6: Afficher des informations complémentaires sur l'application FixMyShoulder doit afficher des informations à propos de l'auteur, des copyrights et des différentes façons de contacter l'auteur ou de partager l'application.

2.3.2 Exigences non fonctionnelles

2.3.2.1 Convivialité

ENF-1. Le système doit permettre une compréhension facile de son utilisation ;
ENF-2. Le système doit être visuellement beau et attrayant ;
ENF-3. Le système doit être fluide.

2.3.2.2 Maintenabilité

ENF-4. Le système doit être réutilisable pour les versions à venir de l'application ;
ENF-5. Le système doit permettre de modifier facilement les données dynamiques de l'application (images, titres, sous-titres, contenus des articles, vidéos, articles en vente…).

2.3.2.3 Performance

ENF-6. L'application doit avoir un temps de chargement court et répondre aux besoins de l'utilisateur dans un délai respectable.

2.3.2.4 Portabilité

ENF-7. Le système doit être fonctionnel sur IOS et Androïde, et l'interface doit s'adapter à toutes les tailles d'écran.

2.3.2.5 Fiabilité

ENF-8. Le système doit s'exécuter sans fautes à 97% du temps.

2.3.3 Contraintes

C-1. L'application doit respecter les standards d'Apple et Google ;
C-2. L'application doit s'adapter parfaitement à tous les formats et tailles d'écran qu'il existe sur le marché ;
C-3. L'application doit utiliser une technologie de développement fiable et récente, qui sera maintenue au cours des prochaines années.

2.4 Méthodologie

2.4.1 Procédure de travail

Le développement de l'application s'est déroulé selon les étapes suivantes :

1. L'étude des technologies actuelles, qui a permis de faire une sélection de la technologie la plus adaptée pour le projet, en prenant en compte ses différentes contraintes ;
2. L'étude de l'existant, en étudiant les travaux réalisés par Julie Vincent et Mathieu Crochet ;
3. La conception de l'application ;
4. L'apprentissage des technologies utilisées ;
5. L'implémentation de l'application.

2.4.2 Conditions nécessaires au développement

Dans le cadre du développement d'une application, il est nécessaire d'avoir une machine capable de supporter tous les logiciels ou technologies requises par ce développement. Le développement de ce projet a nécessité l'utilisation des technologies suivantes :

- Ionic 4 ;
- Angular 6 ;
- Cordova 6 ;
- Un navigateur Web récent (pour tester l'application directement sur un navigateur)
- SDK d'Androïde (pour compiler l'application sur Androïde) ;
- Xcode (pour compiler l'application sur IOS).

L'intégralité des technologies présentées ci-dessus, à l'exception d'Xcode, sont compatible avec des machines 32 et 64 bits Windows (c.-à-d. Vista ou plus récent) et Mac OS (c.-à-d.

10.5.8 ou plus récent). Malheureusement, la compilation d'une application IOS nécessite d'avoir impérativement Xcode, compatible uniquement avec la dernière version de MacOS (à ce jour 10.14.4). Dans le cadre de ce projet, il est donc impératif d'avoir un ordinateur sous la dernière version de MacOS.

2.5 Conception

2.5.1 Conception de l'application

2.5.1.1 Structure de l'application

La première étape de conception du projet fut de définir une vue d'ensemble de l'application, à l'aide d'un diagramme représentant la structure de celle-ci, du point de vue de l'utilisateur (interface graphique). Le but n'était pas de concevoir la maquette de l'application, mais de la représenter graphiquement. Ionic mettant à disposition de nombreux éléments d'interface graphique, simple à utiliser, qui respecte les codes visuels et les styles d'IOS et Androïde (boutons, cases à cocher, listes, menus, barres de recherche, etc.), il était préférable de concevoir la maquette de l'application en l'implémentant directement, pendant la phase d'implémentation du projet.

Pour concevoir la vue d'ensemble du projet, je me suis référé aux exigences définies par George. Ces dernières sont totalement différentes de celles de la première version de l'application, réalisée par Julie et Mathieu. Il m'a donc fallu penser à une toute nouvelle navigation, intégrant les fonctionnalités requises par les exigences.

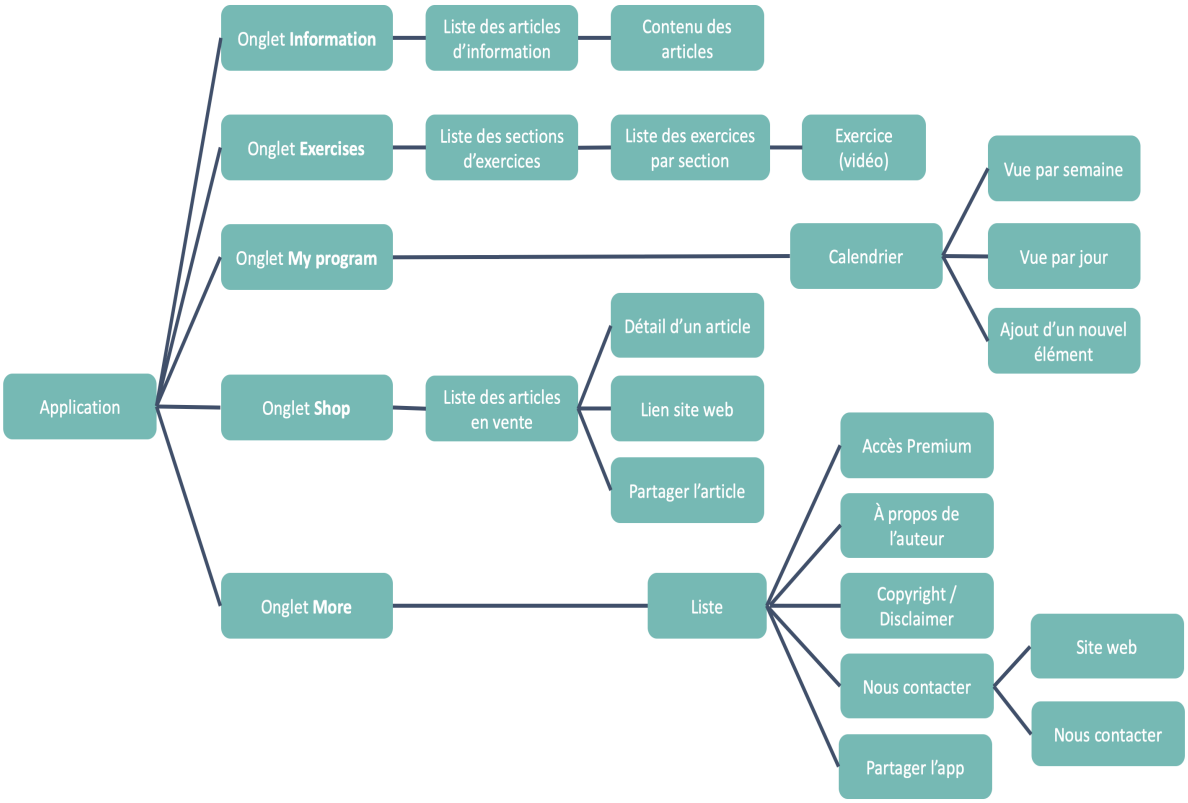

Figure 2.2 Vue d'ensemble de la nouvelle version de l'application

Cette nouvelle conception contient maintenant cinq onglets :

- **Information** : cet onglet intègre la liste des articles d'information de George, ainsi que leurs contenus ;
- **Exercises** : cet onglet intègre la liste des sections d'exercices, ainsi que, pour chacune d'entre elles, la liste des exercices correspondante (sous forme de vidéos) ;
- **My program** : cet onglet contient un calendrier permettant de planifier ses exercices (ajout et suppression). Le calendrier propose une vue par semaine et par jour ;
- **Shop** : cet onglet présente les livres de George, avec pour chacun, une description, un lien commercial pour l'acheter, ainsi que la possibilité de partager l'article par courriel ;
- **More** : cet onglet affiche une liste de cinq sections : Premium, À propos de l'auteur, Droit d'auteur/Avertissement, Nous contacter, Partager l'application. La page Premium permet d'activer l'accès aux contenus réservés uniquement aux membres Premium.

2.5.1.2 Couleur principale de l'application

Le choix des couleurs principales d'une application est une étape importante, car notre cerveau associe à chaque couleur une émotion (voir l'article [17]) :

Couleur	Émotion
Blanc	Calme, neutre
Vert	Croissance, santé
Bleu	Confiance, sûr, force
Rouge	Excitation, audace
Mauve	Créativité, imagination
Orange	Amical, joyeux, confiance
Jaune	Optimisme, clarté, chaleur

Tableau 2.1 Liens entre les couleurs et leurs émotions

Il est donc important de choisir une couleur qui sera liée à l'émotion véhiculée par le thème de l'application, en l'occurrence le médical. Elle est donc liée aux émotions santé, confiance, sûr, et force. En se référant au tableau ci-dessus, nous pouvons en déduire qu'une couleur qui correspondrait à l'application se situerait entre le bleu et le vert.

Il était également important d'avoir une couleur en corrélation avec celles de la couverture du livre Fix My Shoulder. L'application mobile étant liée au livre, il est important d'avoir les mêmes codes visuels, afin de faire comprendre à l'utilisateur que ces deux produits sont liés. Le site coolors.co permet, à partir d'une image, d'en trouver les couleurs les plus présentes. Ce site a été utilisé dans le but de choisir le plus possible une couleur qui est à la fois présente dans la couverture du livre du George, et qui correspond aux codes émotionnels présentés dans le tableau 1.1. :

Figure 2.3 Couleurs présente dans la couverture du livre Fix My Shoulder

Parmi les couleurs présentes dans la couverture, cette qui corrèle le plus avec l'émotion véhiculée par l'application est la troisième (#7DCEC8), car elle est constituée principalement de bleu et de vert. C'est celle dernière qui a été retenue comme couleur principale de l'application.

2.5.2 Technologies utilisées

2.5.2.1 Ionic

La technologie retenue pour ce projet est Ionic. Le choix de la version d'Ionic à utiliser pour le développement du projet a été une longue décision. En effet, nous sommes actuellement à la version 4 d'Ionic, sorti fin 2018, qui est toujours disponible en version bêta. La technologie n'est donc pas encore mature et comporte beaucoup de dysfonctionnement. En plus de cela, il n'existe aujourd'hui que très peu de tutoriels ou documentation sur le Web, concernant Ionic 4.

Malgré toutes ces contraintes, lorsque l'on développe une application, il est préférable d'utiliser une version récente de la technologie qu'on utilise, car à l'avenir elle sera

maintenue plus longtemps. C'est pour cette raison que Ionic 4 a été retenu pour le développement du projet.

Comme décrit précédemment dans la revue de littérature, Ionic est un cadriciel spécialisé dans le mobile, et particulièrement dans le UI et l'expérience usager. Ce cadre logiciel utilise le Cordova, un autre cadriciel qui permet d'accéder aux fonctionnalités natives du téléphone, comme le stockage de données, ou l'accès aux données de l'accéléromètre, et Angular 6, une plateforme d'application permettant entre autres de modulariser une application TypeScript (ce cadriciel est développé plus bas dans ce rapport).

2.5.2.2 Cordova

Cordova est un cadriciel disponible librement, supporté par Adobe, pour la création d'applications mobiles. Ce cadre logiciel fonctionne à la fois sur Androïde et IOS. Il utilise les mêmes technologies que le Web 2.0, c'est à dire HTML 5, CSS 3 et JavaScript. Les applications qui en résultent sont des applications hybrides, car tout l'affichage est réalisé dans une vue Web, c'est-à-dire que c'est le navigateur du téléphone qui s'occupe de l'affichage de l'application (la barre d'adresse et les menus du navigateur ne sont bien sûr pas visibles). Voici son fonctionnement :

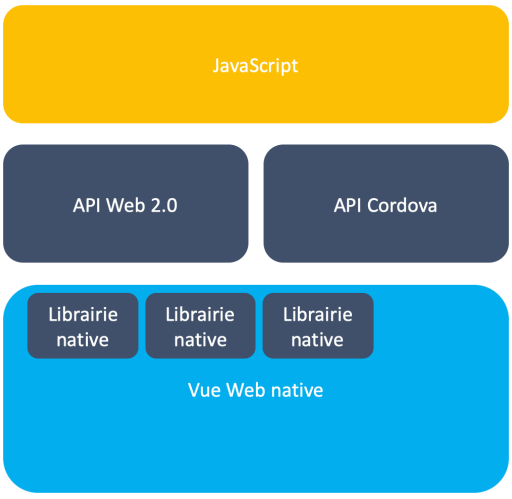

Figure 2.4 Fonctionnement de Cordova

Lorsqu'on réalise une application avec Cordova on utilise uniquement le cœur du cadriciel qui ne comprend pas l'accès aux fonctionnalités du téléphone, mais simplement la génération d'une application. Si l'on souhaite accéder à une native du téléphone, on importe un des « plug-ins » que Cordova met à disposition. Cela permet de n'utiliser que ce dont on a besoin et ainsi d'alléger l'application. Les « plug-ins » font la liaison entre la partie Web de l'application et l'API de l'OS du téléphone.

Pour utiliser les fonctionnalités natives du téléphone, on peut également utiliser les API standards du W3C (World Wide Web Consortium), la principale organisation internationale de normalisation pour le World Wide Web (abrégé WWW ou W3). Les API standards sont disponibles pour le développement Web classique.

L'API Cordova est donc un « plug-in » qui met à disposition des libraires natives. Une application Cordova qui n'utilise que des API standards, ou peux d'API Cordova peut ainsi facilement être transformé en application Web.

Cordova et le développement classique d'un site Web utilisent donc les mêmes langages. Cependant, Cordova comporte quelques caractéristiques qui lui est propres, comme le fait que toute la logique de l'application doit se faire côté client, que l'interface utilisateur doit être adaptée pour une utilisation sur mobile, ou qu'il faille considérer que l'accès à internet pourrait ne pas être disponible.

2.5.2.3 Angular

Angular est un cadriciel JavaScript libre initialement créé par Google. Il fait partie des cadres logiciels les plus utilisés de ces dernières années. Son but est de simplifier la syntaxe JavaScript et de combler les faiblesses de ce langage. Celui-ci est construit autour de plusieurs concepts améliorant l'utilisation du code qui devient alors un gain de temps pour un développeur.

Le concept principal d'Angular est son architecture MVC (modèle, vue, contrôleur). Ce modèle amène à une séparation visible lors du développement entre les données (Modèle), la représentation de ces données c'est-à-dire ce que voit l'utilisateur (Vue) et les différents traitements que nous pouvons effectuer sur ces données (Contrôleur). Voici un schéma simple pour mieux comprendre l'architecture MVC avec Angular :

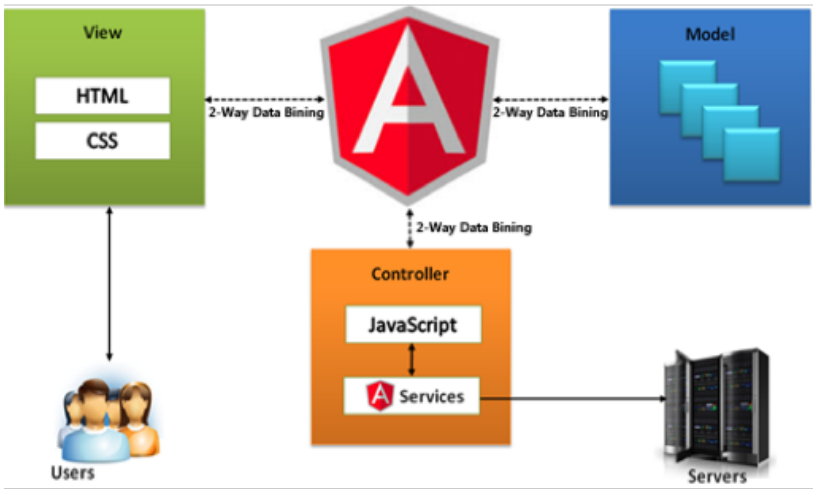

Figure 2.5 Fonctionnement d'Angular

Angular embarque aussi le concept de « data-binding » (liaison de données) offrant une meilleure communication entre le modèle et la vue. Ce concept nous permet d'afficher de manière dynamique le contenu d'une variable. Lorsque le contenu de la variable affichée est modifié, la vue affiche automatiquement sa nouvelle valeur, sans que l'on ait besoin de recharger la page. Le « data-binding » se charge de la synchronisation des données entre la vue et le modèle.

2.5.3 Outils utilisés

Pour le développement de l'application, j'ai utilisé Visual Studio Code. Visual Studio Code est un éditeur de code source développé par Microsoft pour Windows, Linux et Mac OS. Il inclut de la nombreuse fonctionnalité intéressante telle que :

- La prise en charge du débogage, permettant de détecter d'éventuelles erreurs dans le code ;
- L'intégration de Git, pour l'utilisation de logiciel de versions ;
- La coloration syntaxique, pour une meilleure lisibilité du code ;
- La complétion intelligente de code, pour gagner du temps.

Pour la compilation Androïde, j'ai utilisé le SDK d'Androïde, qui permet de développer des applications sous Androïde. Pour la compilation IOS, j'ai utilisé Xcode, le logiciel de développement intégré (IDE) d'Apple pour le développement d'application IOS. Ce logiciel offre la possibilité de tester une application sur des simulateurs iPhone et iPad, ou sur des appareils physique.

Pour la gestion des versions du projet, j'ai utilisé git (un logiciel de gestion de versions décentralisé), avec GitLab, un système de gestion de développement collaboratif de logiciel qui utilise la technologie git, permettant d'héberger le code source de l'application sur un serveur, et d'en gérer facilement ses versions.

Les patrons de conception principalement utilisés dans le projet sont le MVC (expliqué précédemment) et l'observateur, qui est utilisé pour transmettre les modifications apportées à un objet.

2.6 Implémentation

2.6.1 Implémentation Ionic

L'arborescence du projet est la suivante :

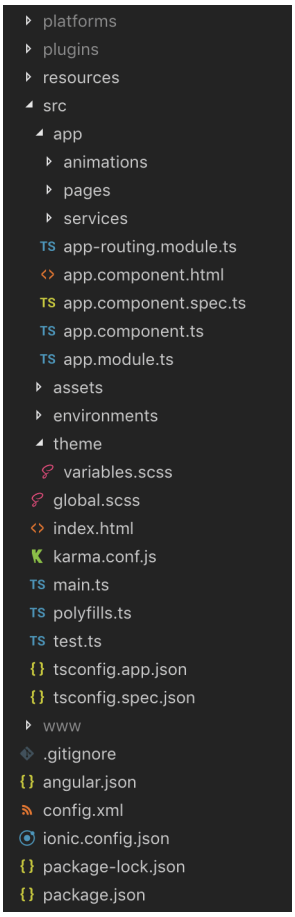

Figure 2.6 Arborescence du projet

L'arborescence d'un projet sous Ionic 4 comporte de nombreux dossiers ou fichiers donc voici une description pour les plus importants d'entre eux :

- Un dossier **plateforms** : qui contient les applications compilées pour IOS et Androïde ;
- Un dossier **plug-ins** : qui contient tous les « plug-ins » de Cordova ;
- Un dossier **ressources** : qui contient les ressources des applications IOS et Androïde, à savoir leurs icônes et écrans d'accueil ;
- Un dossier **src/app/animations** : qui contient toutes les animations personnalisées de l'application ;
- Un dossier **src/app/pages** : qui contient toutes les pages de l'application. Chaque page contient entre autres un fichier pour le modèle, pour la vue, et pour le contrôleur ;
- Un dossier **src/app/services** : qui contient les services, ou « providers » du projet. Un service est un fichier qui offre un service qui peut être utilisé par n'importe quel autre contrôleur du projet. Pour ce projet, j'ai créé le service **data.service.ts**, dont le

but est de fournir les données dynamiques de l'application (titres, contenus des articles, vidéos…) ;

- Un fichier **src/app/app-routing.module.ts** : qui inclut toutes les routes des pages de l'application. Pour ce projet, il contient une route vers la page « tab », ainsi qu'une route pour chaque sous-page des onglets de l'application (détail des articles d'information, liste des vidéos par sections…). La page « tab » gère tout ce qui est lié au menu du bas de l'application (qui contient les cinq onglets). Cette dernière contient elle-même les routes vers les cinq pages principales des cinq onglets (Information, Exercises, My program, Shop et More) ;
- Un fichier **src/app/app.module.ts** : qui contient principalement les « plug-ins » à utiliser dans l'application ;
- Un dossier **src/app/assets** : qui contient toutes les ressources externes utilisées dans l'application (images, bruitages des notifications, polices…) ;
- Un fichier **src/app/theme/variables.scss** : qui contient les couleurs principales de l'application ;
- Un fichier **src/app/global.scss** : qui peut contenir des variables globales qui peuvent être utilisées dans tout le projet ;
- Un fichier **src/app/config.xml** : qui contient des réglages et préférences concernant l'application, par plateforme ;
- Un fichier **src/app/packages.json** : qui contient la liste de tous les « plug-ins » installés.

Le développement d'un projet sous Ionic requiert l'utilisation d'un terminal pour effectuer des actions telles que l'ajout d'une page, l'installation d'un « plug-in » ou la compilation de l'application sur IOS. Il est très facile de comprendre le fonctionnement d'un « plug-in » Cordova, en se référant à la documentation officielle d'Ionic, qui propose des exemples d'utilisation.

2.6.2 Développement de l'application

L'application contient un menu incluant cinq onglets :

- **Informations**, incluant la sous-page **infos-details** ;
- **Exercises**, incluant la sous-page **exercises-details** ;
- **My program**, incluant les sous-page **add-event** et **events-details** ;
- **Shop**, incluant la sous-page **shop-details** ;
- **More**, incluant les sous-page **premium**, **about-author**, **contact-us** et **copyright**.

L'intégralité des données dynamiques présentées dans la première version du projet, comme les textes de remplissages, images, couleurs ou vidéos, concernant les exercices, articles d'information, livres dans l'onglet « Shop », et autres données dans l'onglet « More », ont été choisis de façon arbitraire, afin de montrer des exemples de ce que pourrait être l'application une fois finie. L'intégralité de ces données peut être modifiée dans le service **data.service.ts** (présenté ci-dessus) du projet.

L'icône du projet, ainsi que l'écran de changement (de l'anglais, le Splash screen) de l'application ont été réalisés à l'aide de Photoshop, à partir de la couverture du livre Fix My Shoulder :

Figure 2.7 Icône et écran de chargement de l'application

Afin de respecter les styles d'icônes présents sur IOS et Androïde, l'idée ici était de donner à l'icône de l'application une impression d'image modélisée en trois dimensions (c.-à-d. image de synthèse), et non une photographie, en appliquant plusieurs filtres, comme la réduction de grain, et l'uniformisation de la couleur d'arrière-plan. Que ce soit sur IOS ou Androïde, Apple et Google préconisent de ne pas utiliser de photographie pour les icônes d'application, mais de rechercher un seul élément qui capture l'essence de notre application, en l'exprimant sous une forme simple et unique. L'article [16], présent sur le site officiel d'Apple, décrit bien toutes ces règles à respecter. Une épaule est un élément simple qui représente bien la fonction de l'application.

2.6.2.1 Onglet Information

L'onglet Informations se présente comme suit :

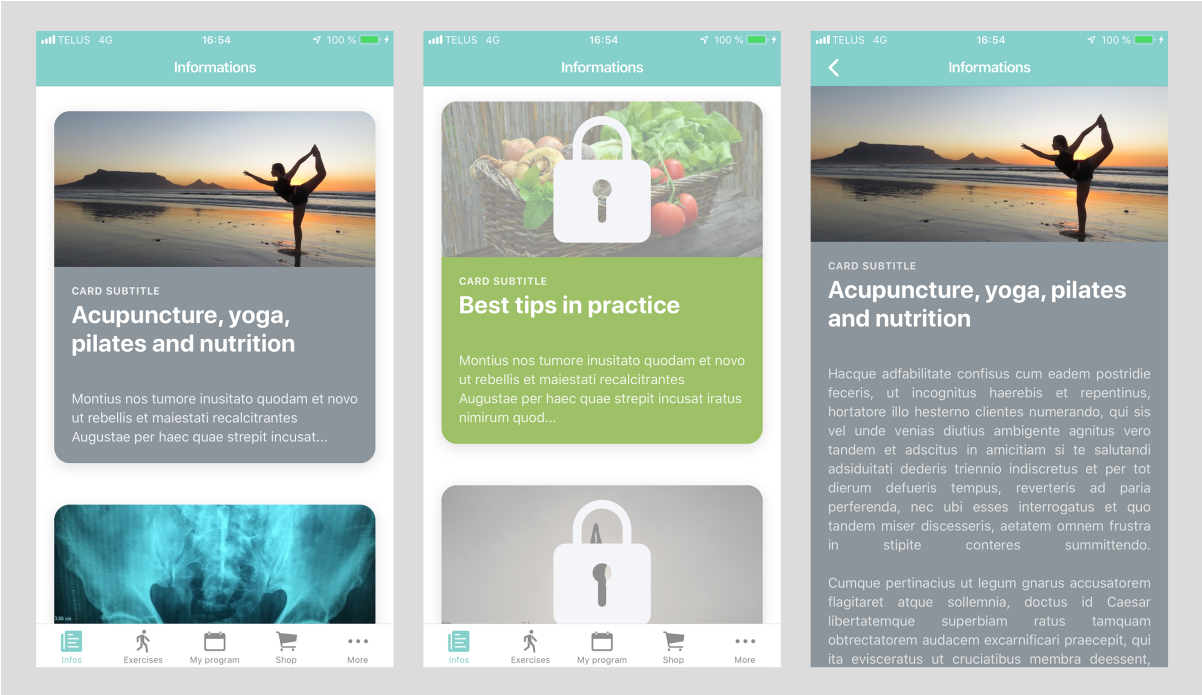

Figure 2.8 Onglet Information

Il contient une liste des articles d'informations. Dans cette liste, chaque article est représenté avec une image, une couleur qui lui est propre, ainsi qu'un titre, un sous-titre et une description courte. Cette présentation a été choisie afin de rendre chaque article unique et attrayant, dont le but est de donner envie à l'utilisateur de cliquer dessus. L'image est très importante, car elle permet d'attirer l'attention de l'utilisateur.

Lorsque l'on clique sur une carte, une nouvelle page s'ouvre et affiche son contenu. Les cartes réservées aux membres Premium sont verrouillées et affichent un cadenas pour

indiquer à l'utilisateur qu'il faut payer un abonnement Premium pour y accéder. Un appui dessus nous amène directement à la page de gestion du mode Premium.

Les données dynamiques sont fournies par le service **data.service.ts** aux contrôleurs des pages **infos** et **infos-details.** Voici un exemple d'objet fourni par le service :

```
infosData: any[] = [
    {
      id: "1",
      title: "Acupuncture, yoga, pilates and nutrition",
      subtitle: "Card Subtitle",
      preview: "Montius nos tumore inusitato quodam et novo ut rebellis et... ",
      content: "Hacque adfabilitate confisus cum eadem postridie feceris…",
      picture: "assets/pictures/yoga.jpg",
      whiteText: true,
      color: "#8C959C",
      lock: false
    }]
```

Algorithme 1 Modèle de données de l'onglet Information

Cet objet contient les données d'un article. Chaque article contient les données suivantes :

- Un « **id** », pour identifier de façon unique un article ;
- Un **titre** ;
- Un **sous-titre** ;
- Une **courte description** (preview) ;
- Le **contenu de l'article** (sous forme de texte) ;
- **L'image** de l'article ;
- Un **booléen** qui indique si les textes de l'article doivent être noir (booléen à faux) ou blanc (booléen à vrai) ;
- Une **couleur de fond** ;
- Un **booléen** qui indique si oui ou non, l'article est réservé aux membres Premium.

Les cartes des présentations des articles ont été réalisées avec le composant « **ion-card** » d'Ionic (https://ionicframework.com/docs/api/card). Voici leurs implémentations HTML:

```
      <ion-card *ngFor= "let item of data" [ngStyle]="{'background-color': item.color} (click)="openModal(item)">
       <img src="{{item.picture}}"/>
       <div *ngIf="item.lock && !isPremium" >
         <ion-icon name="lock" color="light" class="icon-lock"></ion-icon>
       </div>
       <ion-card-header>
```

```
        <ion-card-subtitle [ngClass]="{'light-color': item.whiteText}">{{item.subtitle}}</ion-card-subtitle>
        <ion-card-title [ngClass]="{'white-color': item.whiteText}">{{item.title}}</ion-card-title>
      </ion-card-header>
      <ion-card-content [ngClass]="{'light-color': item.whiteText}">
        {{item.preview}}
      </ion-card-content>
    </ion-card>
```

Algorithme 2 Implémentation HTML de la liste des articles

Le contrôleur de la page récupère une liste d'articles du service **data-service** et la stocke dans une variable nommée *data*. Dans le fichier HTML, l'opération `*ngFor= "let item of data"` de la balise `<ion-card>` permet d'afficher un élément `<ion-card>` pour chaque article contenu dans la variable *data*. Chaque article unique est représenté par une variable temporaire nommée *item*, qui contient toutes les données de l'article qui lui est référé.

L'opération `[ngStyle]="{'background-color': item.color}` permet d'appliquer la classe CSS *background-color* (qui applique un texte blanc) à la balise `<ion-card>` uniquement si la variable *color* de l'article en question (item.color), qui représente un booléen, est à *vrai*. Cette variable permet de mettre un texte blanc lorsque la couleur de fond de la carte est foncée. Autrement, si cette variable est à *faux*, la couleur du texte de l'article sera par défaut foncée (c.-à-d. utile lorsque la couleur de fond est claire).

L'opération `(click)="openModal(item)` permet d'indiquer que lorsque l'utilisateur clique sur une carte, il faut appeler la méthode `openModal()`, avec comme paramètre l'article correspondant à la carte cliquée. Cette méthode ouvre une page de détails de l'article passé en paramètre.
L'accès aux données de chaque article (c.-à-d. titre, couleur…) se fait selon sa syntaxe suivante : `"{{item.[donnée]}}"`, ou [donnée] représente le nom de la variable lié à la donnée que l'on veut récupérer (id, title, subtitle…).

La balise `<div>` contient l'opération suivante: `*ngIf="item.lock && !isPremium"`. Cette dernière permet d'afficher cette balise, qui contient une icône cadenas, uniquement si l'article est réservé aux membres Premium (item.lock) et si l'utilisateur n'a pas d'abonnement Premium (!isPremium).

Cette même logique est utilisée dans toutes les pages de l'application.

2.6.2.2 Onglet Exercises

L'onglet exercices se présente comme suit :

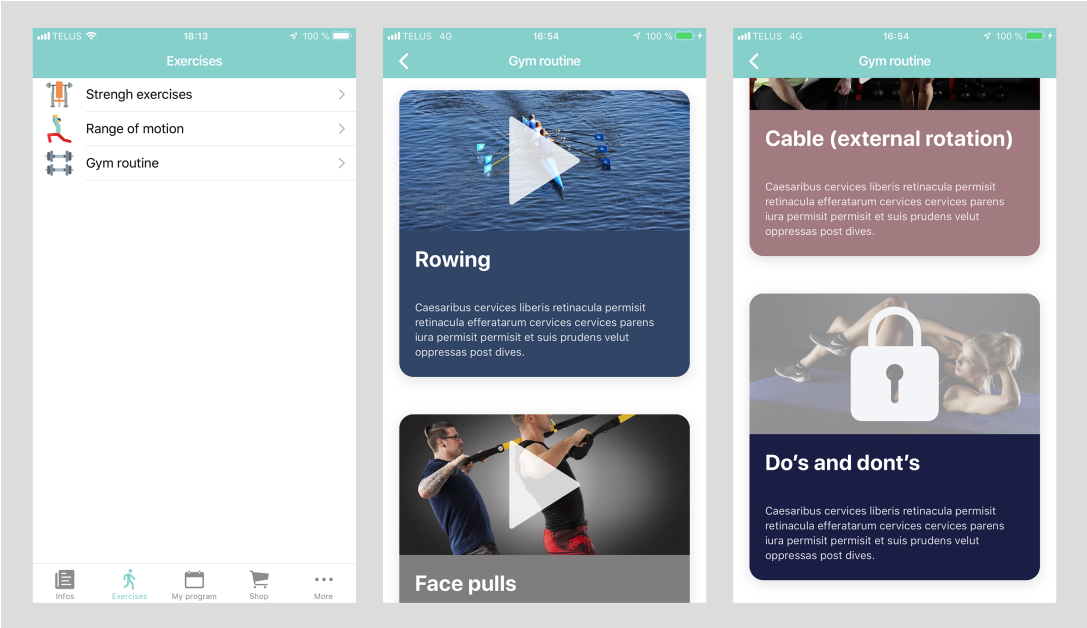

Figure 2.9 Onglet Exercises

Voici un exemple de lecture d'une vidéo :

Figure 2.10 Exercice

Cet onglet présente une liste des sections d'exercices. Lorsque l'on clique sur une section, nous avons accès à la liste des exercices de la section. Un appui sur un exercice lance la lecture de la vidéo de l'exercice. La liste des vidéos reprend le même style que celle des articles d'informations. La lecture des vidéos est faite directement à partir de YouTube, via le « plug-in » Cordova **Cordoba-plugin-youtube-video-player**. Son utilisateur est très simple et se fait comme suit :

```
this.youtubeVideoPlayer.openVideo(video.id);
```

Algorithme 3 Implémentation lecture vidéo YouTube

Pour lancer la lecture d'une vidéo, il suffit d'utiliser la méthode openVideo(), avec comme paramètres l'« id » de la vidéo YouTube que l'on souhaite lire.

Voici la structure de donnée de l'onglet exercise :

```
exercisesData: any[] = [
    {
      id: String,
      sectionTitle: String,
      icon: String,
      videos: [{
        title: String,
        description: String,
        id: String,
        picture: String,
```

```
    whiteText: Boolean,
    color: String,
    lock: Boolean
}]
```

Algorithme 4 Structure de donnée de l'onglet « exercise »

Chaque section contient un « id », un titre, une icône et une liste de vidéos. Chaque vidéo contient un titre, une description, un « id » (ID de vidéo YouTube), une image d'illustration, une couleur de texte (c.-à-d. blanc ou noire) et de fond ainsi qu'un booléen qui indique si oui ou non l'exercice est réservé aux membres Premium.

2.6.2.3 Onglet My program

L'onglet *My program* se présente comme suit :

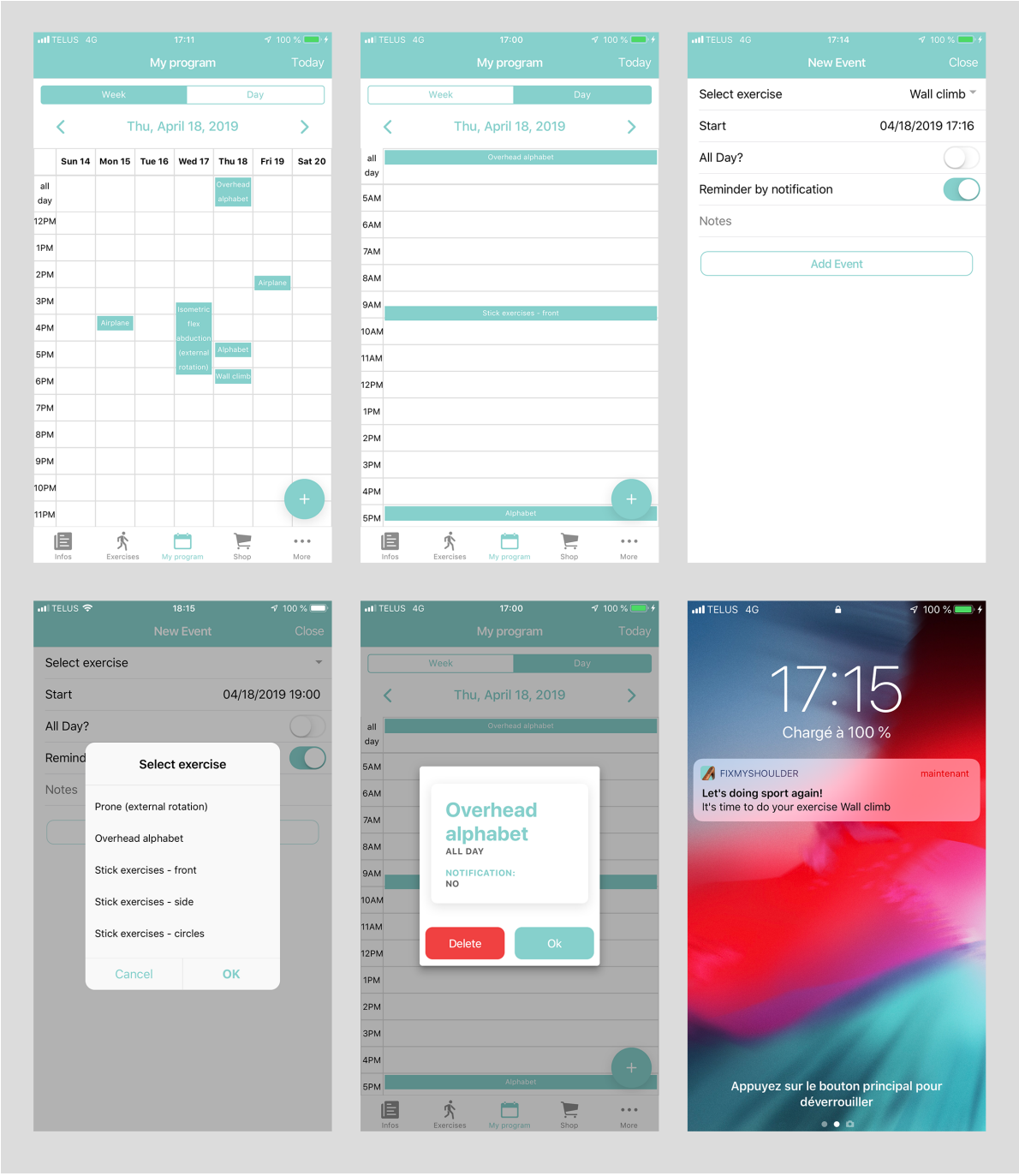

Figure 2.11 Onglet My program

Ce dernier est composé d'un calendrier permettant à l'utilisateur de créer son propre programme d'exercices. Le calendrier propose une vue par semaine, et par jour. Lorsque l'utilisateur ajoute un évènement, il doit renseigner les informations suivantes :

- L’exercice à effectuer. L’application lui propose une liste d’exercice. Cette liste contient tous les exercices de l’onglet *exercises*. ;
- La date et l’heure ;
- Indiquer si l’évènement durera toute la journée. Dans ce cas, ce dernier sera affiché en haut de la journée ;
- Indiquer si l’on veut un rappel par notification ou non ;
- Une note (facultative).

Sur la vue calendrier, pour avoir tous les détails concernant un évènement, il suffit de cliquer dessus. On a alors la possibilité de le supprimer. Le calendrier a été réalisé à l’aide du « plug-in » *ionic2-calendar*, les notifications à l’aide du « plug-in » *cordova-plugin-local-notification*, et le stockage des évènements sur le téléphone l’aide du « plug-in » *cordova-sqlite-storage.*

2.6.2.4 Onglet Shop

L’onglet Shop se présente comme suit :

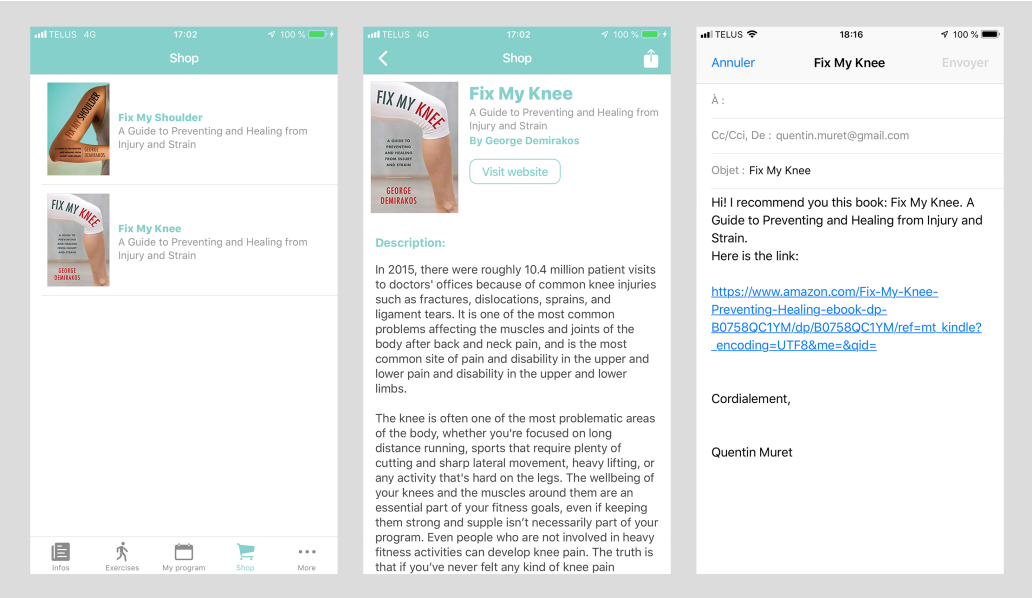

Figure 2.12 Onglet Shop

Cet onglet est composé d'une liste de livres écrits par George, et mise en vente sur le site Amazon. Lorsque l'on clique sur un article de la liste, une nouvelle page s'ouvre et en affiche plus d'informations. On a alors la possibilité de partager l'article par courriel ou d'ouvrir le lien Amazon sur un navigateur Web externe. Lorsque l'on partage l'article par courriel, le corps du courriel est généré automatiquement et comporte entre autres le lien vers l'article.

Le modèle de donnée de l'onglet Shop est le suivant :

```
shopData: any[] = [
    {
      id: String,
      title: String,
      subtitle: String,
      description: String,
      picture: String,
      author: String,
      link: String
    }]
```

Algorithme 5 Modèle de donnée de l'onglet Shop

Chaque article possède un « id », un titre, un sous-titre, une description, une image, un auteur et un lien de vente.

Le partage de l'article par email a été implémenté à l'aide du « plug-in » *cordova-plugin-email-composer* et l'ouverture du lien de vente dans un navigateur externe à l'application a été effectué grâce au « plug-in » *cordova-plugin-inappbrowser*.

2.6.2.5 Onglet More

L'onglet More est composé de cinq pages : *Premium*, *About Author*, *Copyright / Disclaimer*, *Contact us* et *Share the app*, dont voici la liste :

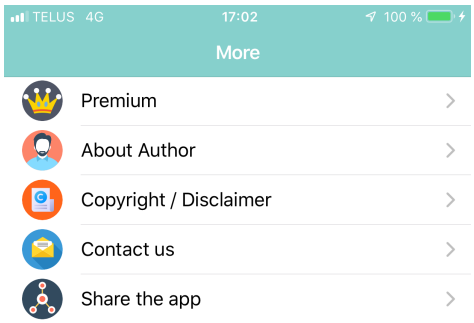

Figure 2.13 Onglet More

Voici le contenu de chacune des sous-pages :

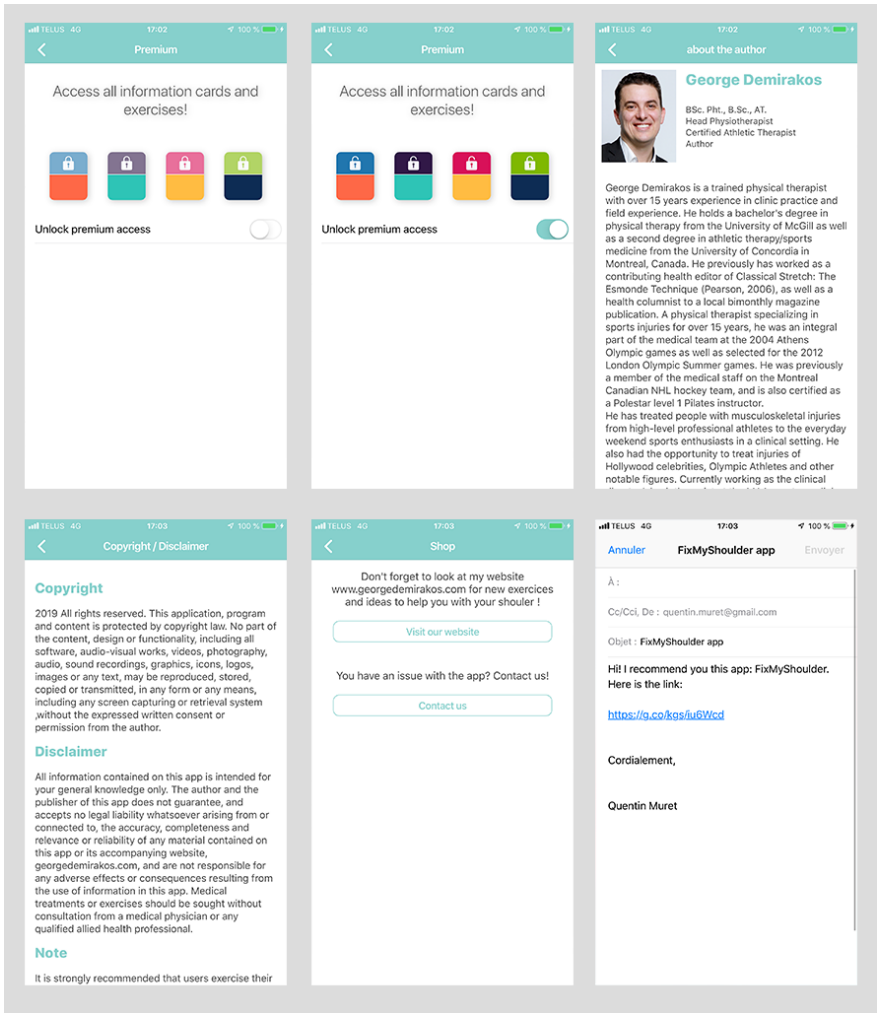

Figure 2.14 Sous-pages de l'onglet More

La page *Premium* permet d'activer le mode Premium de l'application, donnant accès aux articles et exercices réservés aux membres Premium (voir les deux premières images de la figure 2.14). L'implémentation de l'achat intégré (c.-à-d. le in app purchace) n'a pas été encore effectuée, car il nécessite la création d'un compte développeur Apple et Google. Cette étape sera donc à implémenter durant la phase de production de l'application. L'implémentation de cette fonction devra se faire via le « plug-in » *cordova-plugin-inapppurchase.* Pour l'instant, la page Premium inclut un simple interrupteur.

Les informations des pages *About Author*, *Copyright / Disclaimer*, *Contact us* ont été reprises de la version originale de Mathieu. La page *Share the app* utilise, tout comme l'onglet *Shop*, le « plug-in » *cordova-plugin-email-composer,* permettant ainsi de partager l'application par email.

Voici le modèle de donnée de l'onglet *More* :

```
moreData: any[] = [
    {
      id: String,
      sectionTitle: String, //Premium
      icon: String,
      link: String,
      text: String
    },
    {
      id: String,
      sectionTitle: String, //About author
      link: String,
      icon: String,
      title: String,
      subtitle: String,
      text: String,
      picture: String
    },
    {
      id: String,
      sectionTitle: String, //Copyright / Disclaimer
      icon: String,
      link: String,
      copyright: String,
      disclaimer: String,
      note: String
     },
    {
```

```
    id: String,
    sectionTitle: String, //Contact us
    icon: String,
    link: String,
    text1: String,
    text2: String,
    website: String
  },
  {
    id: String,
    sectionTitle: String, //Share the app
    icon: String,
  }
```

Algorithme 6 Modèle de donnée de l'onglet More

Ce dernier comporte un lot de donnée par page de l'onglet.

2.6.3 Difficultés rencontrées

Durant le développement de l'application, plusieurs problèmes ont été rencontrés, dont la plupart sont liés à la version 4 d'Ionic. Cette dernière, actuellement en version bêta, comporte encore beaucoup de dysfonctionnements et de problèmes de performances. Voici les principaux problèmes que j'ai rencontrés :

2.6.3.1 Bouton retour

La particularité de la version 4 d'Ionic est qu'elle comporte une table de routage (src/app/app-routing.module.ts), qui contient les routes des pages de l'application. Lorsque l'on utilise une barre d'onglet pour la navigation (c.-à-d. tabbar), cette dernière est considérée comme étant une page à part entière. La table de routage contient donc la route de ce menu (appelé page « tabs »). C'est cette page « tabs » qui contient les routes de ses propres onglets (c.-à-d. les pages Information, Exercises, My program, Shop et More). Les routes des sous-pages sont-elles bien stockées dans le fichier src/app/app-routing.module.ts. Le problème est le suivant : à partir d'une sous-page, lorsque l'on clique sur le bouton de retour, l'application nous redirige vers la page précédente affichée, sois la page « tabs ». En l'occurrence, l'onglet

affiché par défaut de cette page est l'onglet Information. Ainsi, à chaque retour en arrière dans l'application, nous nous retrouvons systématiquement à l'onglet Information, et non pas à l'onglet précédemment ouvert. Ce défaut d'Ionic 4 est en réalité très commun et ne peut être résolu de manière simple. La seule solution à ce jour est d'attendre une future mise à jour d'Ionic qui corrigera ce défaut.

Pour contourner ce problème, j'ai décidé d'afficher chaque sous-page non pas comme une page de navigation classique, mais comme une page « modal ». Un page « modal » est une page particulière, car elle se superpose à la navigation de l'application, sans modifier son état. Ainsi, lorsqu'on ferme ce type de page, nous retrouvons l'état original de la navigation.
En plus de résoudre ce problème, l'utilisation de page « modal » a permis d'améliorer les performances de l'application, car ces dernières utilisent moins de ressources qu'une page de navigation classique. Sur IOS, les pages « modal » n'ont cependant pas la même animation d'apparition qu'une page de navigation classique (mouvement de droite à gauche). J'ai donc réalisé une animation personnalisée pour l'apparition des pages « modal » quasiment similaire à celle utilisée pour les pages de navigation classique.

2.6.3.2 Problème de son des notifications

Le « plug-in » *cordova-plugin-local-notification* a été utilisé pour la génération des notifications d'évènements du calendrier. Ce « plug-in » permet de créer des notifications locales. Par défaut, avec ce « plug-in », les notifications sont muettes. Il est impossible d'avoir de son ou de vibrations à la réception d'une notification. Pour corriger ce problème, il est nécessaire d'installer le « plug-in » *Cordoba-plugin-avaudiosession,* permettant d'activer entre autres le son et les vibrations des alertes par notifications.

2.6.3.1 Couleur de la barre de statuts sur IOS

Sur IOS, il est possible de modifier le texte de la barre de statuts (c.-à-d. en noir ou blanc), qui donne entre autres des informations sur l'état de la batterie, ou les connexions cellulaires ou wifi. La documentation officielle nous propose d'installer un « plug-in » qui permet de faire ce changement. Cependant, bien que non précisé, ce dernier ne marche pas sur Ionic 4. En effet, sous Ionic 4, les paramètres de démarrage d'une application défini systématiquement la couleur de la barre de statuts sur noire. Pour définir cette dernière en blanc, il faut modifier les réglages de démarrage de l'application, dans la méthode *initializeApp()* du fichier *app.components.ts.*

2.6.3.2 Couleur du calendrier

Dans l'onglet *My program*, le calendrier a été réalisé grâce au « plug-in » *ionic2-calendar*. Ce dernier donne la possibilité d'afficher une vue par mois, par semaine ou par jour. Par défaut, les couleurs des cases sont composées de vert et de bleu. Selon la documentation officielle du « plug-in », s'il est possible de modifier ces couleurs dans les vues par semaine et par jour, ce n'est malheureusement pas le cas pour la vue par mois. Ces couleurs n'étant pas du tout en accord avec le thème de l'application, je n'ai eu d'autres choix que je désactiver la vue par mois du calendrier.

CHAPITRE 3

RÉSULTATS

3.1 Présentation des résultats

La nouvelle version de l'application respecte les nouveaux besoins définis par le client, Georges Demirakos, en début de projet. L'application contient les cinq onglets définis pendant la phase d'élicitation des exigences. Elle permet d'afficher des articles d'information. L'intégralité des articles définis par Georges ont été réalisées. Leurs contenus actuels sont composés de données factices, que Georges devra remplacer par son contenu le moment venu.

La nouvelle application permet de lire des vidéos d'exercices d'entraînements. Ces vidéos sont lues directement depuis l'API YouTube. Elle contient également un calendrier qui permet de planifier les exercices que l'utilisateur aura inclus dans son programme, et d'ajouter des rappels par notifications.

L'application contient une section qui permettra à Georges d'exposer ses livres en vente, et une autre pour les informations complémentaires à propos de l'auteur, des copyrights et différentes façons de contacter l'auteur ou de partager l'application.

Enfin, l'application inclut un moyen de rémunération. À ce jour, elle propose du contenu exclusif aux membres Premium. Pour être membre Premium, l'utilisateur doit effectuer un achat intégré à l'application. L'achat devra être effectué soit une seule fois pour un accès illimité, soit sous forme d'abonnement régulier. L'application ne contient à ce jour aucune publicité. Le modèle de rémunération est sujet à changer et devra être discuté avec Georges. La gestion des achats intégrés devra être implémentée durant la phase de production de l'application.

L'application est facile à utiliser, intuitive et attrayante. Un profond travail a été effectué dans le but de rendre l'application visuellement belle et de donner envie à l'utilisateur de rester le plus longtemps possible, et d'accéder aux contenus exclusifs aux membres Premium. Concernant la maintenabilité, l'application recense toutes ses données dans un seul fichier, facilitant la mise à jour des données concernant les articles d'informations, les vidéos ou les articles en vente. Le code source de l'application a été conçu pour être facile à comprendre, avec beaucoup de commentaires d'explication, dans le but de permettre aux futures équipes de maintenance de le comprendre plus facilement. Finalement, la nouvelle application répond au besoin de portabilité. Elle est fonctionnelle sur IOS et Androïde, et l'interface s'adapte à toutes les tailles d'écran.

3.2 Limites de l'application et évolutions possibles

Bien que cette nouvelle application répond aux exigences définies pendant la phase de conception, elle possède cependant certaines limites. La liste suivante présente des points d'amélioration possibles afin d'adresser ces limites dans le futur:

- **Ajout d'un serveur de données** : Avec l'application actuelle, il est possible de modifier les données dynamiques de l'application, mais uniquement à travers son code source. Chaque modification nécessite ainsi une mise à jour de l'application iOS et Androïde, ce qui prends beaucoup de temps. Il serait ainsi plus judicieux d'avoir un serveur distant sur lequel une modification des données serait immédiate, et ne nécessiterait pas de mise à jour de l'application. Le processus de migration vers un serveur serait très facile, car toutes les données de l'application sont gérées par un seul et même fichier ;
- **Possibilité d'ajouter des données aux articles** : À ce jour, chaque article d'information contient une image de présentation. Il n'est pas possible d'en ajouter une deuxième. Il serait intéressant de permettre à l'auteur d'ajouter si nécessaire plus d'illustrations, que ce soit des images, des vidéos ou des schémas d'explication. Cela pourrait rendre les articles plus vivants, captivants et faciles à comprendre ;
- **Ajout d'une foire aux questions pour éviter les questions redondantes** : L'application offre la possibilité de contacter l'auteur pour lui poser des questions.

Afin d'éviter toute redondance dans les questions, il serait intéressant d'ajouter une section « FAQ », dans l'onglet « More », qui inclurait toutes les questions fréquemment posées ;

- **Ajout d'une barre de recherche** : Il serait pratique d'avoir une barre de recherche dans l'application qui nous permette de trouver directement le contenu que l'on cherche, que ce soit des articles ou des vidéos. Pour implémenter cette fonctionnalité, il serait important d'ajouter à chaque contenu, une liste de 'tags' caractérisant la métadonnée principale de son contenu. Cela permettrait de trouver plus facilement ce que l'on cherche, sans devoir taper le titre exact du contenu que l'on cherche, dans la barre de recherche. Par exemple, pour un article d'une recette de gâteau au chocolat, les mots-clés correspondants pourraient être chocolat, cuisine, gâteau, dessert et gastronomie. Il suffirait de rentrer un de ces mots dans la barre de recherche pour faire apparaître l'article concernant la recette de gâteau au chocolat ;
- **Changer le calendrier** : le calendrier de l'application utilise le plug-in *ionic2-calendar*. Ce dernier comporte quelques désavantages, dont notamment l'impossibilité de changer les couleurs des évènements pendant un affichage par mois, qui comporte des couleurs qui ne vont pas avec le thème de l'application. C'est pour cette raison que la vue d'affichage par mois a été désactivée. Du plus, pendant un affichage par jour, lorsque que l'on change de jour en effectuant un « scroll » horizontal sur l'écran, la date du jour sélectionnée n'est pas mise à jour automatiquement. Il faut alors toucher le calendrier pour la mettre à jour, ce qui n'est pas intuitif et impacte l'expérience utilisateur de l'application ;
- **Lecture des vidéos depuis un serveur privé** : La lecture des vidéos se fait actuellement depuis YouTube. Il en est de même pour les vidéos réservées aux membres premiums. L'application stocke en réalité l'ID de la vidéo YouTube, et lance une lecture via l'API YouTube. Lorsqu'un utilisateur non Premium souhaite lire une vidéo réservée aux membres Premium, l'application ne lance pas la vidéo, mais le redirige vers la page de gestion du compte Premium. Cependant, un utilisateur Premium pourrait très bien récupérer les ID de toutes les vidéos Premium et les partager sur internet. Il serait alors possible pour n'importe qui de lire ces vidéos

directement depuis YouTube, sans payer d'abonnement Premium. Si la création d'un serveur de données (voir premier point ci-dessus) est réalisée, il serait très simple d'y stocker toutes les vidéos, et de ne plus utiliser YouTube ;

- **Modification des évènements** : Il serait intéressant de pouvoir modifier un évènement planifié, sans devoir le supprimer et le recréer ;
- **Répétition des évènements** : Il serait intéressant de pouvoir ajouter une répétition aux exercices planifiés (par exemple, tous les jours/semaines/mois) ;
- **Amélioration des performances sur iOS** : Actuellement, la version iOS de l'application est moins fluide que la version Androïde. Cela est dû à Ionic 4, actuellement en version bêta, qui offre des performances moins bonnes sur cette plateforme. Il n'y a pas d'autre choix que d'attendre une future mise à jour d'Ionic qui permette de corriger ce problème ;
- **Possibilité de mettre du contenu en favoris** : Il serait intéressant de pouvoir mettre du contenu en favoris, afin de permettre d'accéder plus rapidement aux contenus que nous utilisons régulièrement ;
- **Déplacer un évènement après un appui long** : Il serait pratique de pouvoir déplacer un exercice planifié (c.-à-d. à l'aide de la fonctionnalité d'appui sur l'objet pour une longue durée) comme sur les calendriers de Google et d'Apple.

CONCLUSION

Dans le cadre de ce projet, le but était de redévelopper l'application FixMyShoulder pour le physiothérapeute George Demirakos. Une première version de l'application a été développée en 2013 par Mathieu Crochet sur la plateforme Androïde, et en 2012 sur la plateforme IOS par Julie Vincent, à l'aide d'un langage natif propre à chaque système. L'application n'a cependant jamais été mise en production. Cette année, Georges a donné son accord pour un redéveloppement de l'application, visant cette fois de nouveaux besoins. L'objectif de cette nouvelle version de l'application était de proposer non plus les chapitres du livre Fix My Shoulder, comme c'était le cas de la première version, mais d'offrir une application complémentaire au livre, proposant des instructions claires et imagées des différents traitements de l'épaule, avec des exercices sous forme de vidéos, et en plus des articles d'information qui permettra à Georges de partager ses connaissances sur des domaines plus divers. La principale contrainte du projet fut d'utiliser une technologie multiplateforme, qui permet de réduire les coûts de maintenance.

La rénovation de l'application a été réalisée avec succès. Toutes les exigences fonctionnelles et non fonctionnelles ont été complétées. L'application est maintenant prête à être mise en production. Il y a toutefois un champ possible d'améliorations proposées au chapitre 3 de ce rapport, qui permettraient de corriger quelques limites de la nouvelle application.

Pendant ce projet, j'ai rencontré plusieurs difficultés. Bien que je connaissais déjà les versions 2 et 3 d'Ionic, il m'a fallu apprendre la version 4, qui comporte beaucoup de nouveautées par rapport aux versions précédentes. La version 4 est actuellement en version bêta, elle comporte donc à ce jour encore beaucoup de dysfonctionnements, et c'est la plus grosse difficulté que j'ai rencontrée durant ce projet. Développer une application sous Ionic 4 peut aujourd'hui être compliqué, car il faut souvent trouver des solutions pour contourner ses dysfonctionnements (voir la partie 2.6.3 – Difficultés rencontrées). De plus, il n'y a que très peu de documentation sur internet, car la version 4 d'Ionic est très récente.

Ce projet a été pour moi une expérience très enrichissante, car il m'a permis d'apprendre à développer une application du début jusqu'à la fin. Bien qu'ayant déjà quelques expériences en développement mobile, je n'avais jamais communiqué directement avec un client. Durant ce projet, je devais comprendre les besoins du client afin de définir un cahier de charges détaillé, dresser une liste d'exigences, concevoir et implémenter l'application et en faire la démonstration pour obtenir ses commentaires. L'expérience utilisateur, les performances et la stabilité de l'application faisaient aussi partie des exigences du projet. Je devais également rendre l'application conviviale et attractive. Ces différentes exigences m'ont permis de me familiariser avec les langages du Web. Ce projet m'a finalement permis de développer mon sens de l'organisation et ma créativité, et a confirmé mon intérêt particulier pour le développement mobile.

ANNEXE I

COMPARAISON ENTRE IONIC ET QT

Cette étude a été réalisée par g2crowd et présente la comparaison entre Ionic et Qt en se basant sur les données des avis utilisateur.

Source : https ://www.g2crowd.com/compare/ionic-vs-qt-creator

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Bozhenko, K. (2017). Why Having Both iOS And Androïde App Versions Is So Important. Repéré à https ://octodev.net/why-having-both-iOS-and-android-app-versions-is-so-important/

[2] Librojo, C. (2018). Six Advantages of Cross-Platform Mobile Application Development. Repéré à https ://alligatortek.com/blog/six-advantages-of-cross-platform-mobile-application-development/

[3] Goyal, A. (2017). 5 Advantages of Cross Platform Mobile App Development. Repéré à https ://www.digitaldoughnut.com/articles/2017/april/5-advantages-of-cross-platform-mob-app-development

[4] Rouse, M. (2018). Cross-platform mobile development. Repéré à https ://searchmobilecomputing.techtarget.com/definition/cross-platform-mobile-development

[5] Sonmez, J. (2016). What is Mobile Development? Repéré à https://simpleprogrammer.com/what-is-mobile-development/

[6] Idesis, S. (2018). Free Cross-Platform Mobile App Development Tools Compared - 2018. Repéré à https://www.outsystems.com/blog/free-cross-platform-mobile-app-development-tools-compared.html

[7] Clutch (2018). Looking for an App Developer to Build an App for You? Repéré à https://clutch.co/app-development/cross-platform

[8] ElHady, H. (s.d.). Your Guide to Cross-Platform Mobile App Development Tools. Repéré à https://instabug.com/blog/cross-platform-development/

[9] G2crowd (s.d.). Compare Ionic vs Qt Creator. Repéré à https://www.g2crowd.com/compare/ionic-vs-qt-creator

[10] Cruxlab, Inc (s.d.). Xamarin vs Ionic vs React Native: differences under the hood. Repéré à https://cruxlab.com/blog/reactnative-vs-xamarin/

[11] Warcholinski, M. (s.d.). 10 Famous Apps Built with React Native. Repéré à https://brainhub.eu/blog/react-native-apps/

[12] Zirous, G. (2018). Why Are Web Frameworks Important? Repéré à https://www.zirous.com/2018/03/13/why-are-Web-frameworks-important/

[13] Dhillon S. & Qusay H. M. (2015). An evaluation framework for cross-platform mobile application development tools. SOFTWARE: PRACTICE AND EXPERIENCE, 45 , 1331–1357. Repéré à https://onlinelibrary.wiley.com/doi/full/10.1002/spe.2286

[14] Nunkesser R. (2018). Beyond Web/Native/Hybrid: A New Taxonomy for Mobile App Development. MOBILESoft, 18. Repéré à https://ieeexplore.ieee.org/document/8543456

[15] Yang Y., Zhang Y., Xia P., Li B. & Ren Z. (2017). Mobile Terminal Development Plan of Cross-platform Mobile Application Service Platform based on Ionic and Cordova. International Conference on Industrial Informatics, 978(1), 5386-2434. Repéré à https://ieeexplore.ieee.org/document/8543442

[16] Apple (2019). Human Interface Guidelines - App Icon. Repéré à https://developer.apple.com/design/human-interface-guidelines/ios/icons-and-images/app-icon/

[17] CreaNino (2015-2019). Les couleurs et les émotions. Repéré à https://www.creanico.fr/prestations/les-couleurs-et-les-emotions/