ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

RAPPORT DE PROJET PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA MAÎTRISE EN GÉNIE
CONCENTRATION, GÉNIE LOGICIEL

PAR
Zakaria Bellal BELLAHOUEL

MIGRATION DE L'APPLICATION DE SNOOBE VERS IOS ET PLATEFORME DE MISE À JOUR COMMUNE CMS

MONTRÉAL, LE 15 MARS 2016

PRÉSENTATION DU JURY

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

Professeur Alain April, directeur de projet
Département de Génie Logiciel et TI à l'École de technologie supérieure

Professeur Abdelaoued Gherbi, jury
Département de Génie Logiciel et TI à l'École de technologie supérieure

REMERCIEMENTS

La réalisation de ce projet a été possible grâce à la participation de plusieurs personnes à qui je voudrais adresser mes remerciements.

Je tiens à remercie en premier mon directeur de projet, M. Alain April, professeur au département Génie logiciel de l'École de technologie Supérieur, d'avoir accepté de diriger ce projet et aussi pour son excellent encadrement et son agréable suivi tout au long de la réalisation de ce projet.

Un grand merci à mes parents, Mahdia et Aissa, pour tous ce qu'ils ont sacrifié pour me permettre d'atteindre ces sommités. Malgré la grande distance qui nous sépare, mais ils n'ont jamais cessé de m'appeler pour me communiquer leurs encouragements de réussite dans ma Maîtrise.

Je tiens aussi à remercie M. Thierry Marechal, Propriétaire de Snoobe, pour le temps accordé et toutes les connaissances transmises qui m'ont beaucoup aidé pour comprendre le système existant et mener à bien ce projet.

Je remercie également M. Abelhakim Aissat, pour tout son support technique et moral dans les moments les plus difficiles dans la réalisation de ce projet.

Finalement, Je remercie tous mes amis et mes collègues qui m'ont encouragé et soutenus tout au long de mon projet de maîtrise.

MIGRATION DE L'APPLICATION DE SNOOBE VERS IOS ET PLATEFORME DE MISE À JOUR COMMUNE CMS

Zakaria BELLAHOUEL

RÉSUMÉ

Ce rapport de recherche appliquée de 15 crédits présente les travaux de recherche menés dans le cadre de la réalisation d'une application mobile multiplateforme dédiée aux deux plateformes dominantes du marché actuel, soit: Androïde et iOS. De plus, le cadriciel de développement mobile multiplateforme Xamarin a été sélectionné pour une expérimentation de réalisation d'un prototype d'application.

Ce rapport débute par la présentation des activités de réingénierie effectuée sur l'application existante, de l'entreprise Snoobe, visant à extraire et à récupérer toutes les informations et les connaissances nécessaires afin d'effectuer l'expérimentation d'un développement multiplateforme. Dans cette étape, le modèle théorique prix comme référence est montré, ainsi que les sources d'information explorées et les résultats souhaités suite à l'application de ce processus.

La deuxième phase du projet de recherche comprend les étapes de réalisation d'un prototype d'application mobile qui a suivi en premier les étapes d'élicitation des exigences et leur validation avec le client, ensuite l'établissement de l'architecture à l'aide du cadriciel Xamarin, l'identification des « APIs » Xamarin nécessaires et qui permettent d'effectuer une conception détaillée. À la fin de cette deuxième étape, un ensemble de tests et la présentation du prototype ont eu lieu. Le dernier chapitre de ce rapport présente l'analyse synthétique de cette recherche. Elle présente une synthèse des efforts déployés pour chaque étape du projet, une synthèse des difficultés rencontrées et des solutions mises en œuvre et finalement une discussion des travaux futurs nécessaires pour compléter ce projet.

Mots-clés : développement mobile multiplateforme; Androïde; iOS; Xamarin; services web; Mono; .NET; XAML.

MIGRATION OF THE APPLICATION SNOOBE TO IOS AND COMMON UPDATE PLATEFORM CMS

Zakaria BELLAHOUEL

ABSTRACT

The following applied research report describes the work done to migrate an existing Android mobile application to a cross-platform mobile application (using Xamarin). The objective of Snoobe is to deploy its services on both Android and iOS. This report is divided into two main parts: the reverse engineering of the existing Android application and the development of a new cross-platform mobile prototype.

The first part of the report presents the reverse engineering activities on the existing application. This includes the front-end Android mobile application as well as the back-end server side application. The Architexa reverse engineering plugin was used in order to extract some graphical presentation of the existing software.

The second part of the report presents the steps taken to redevelop the mobile application on a cross-platform framework. It begins by specifying and validating the user requirements, next it compares the existing requirements, implemented in the Android prototype, with the capabilities of the new cross-platform framework. Then, a new architecture is proposed as well as adjustments forced by iOS restrictions and the use of the Xamarin APIs. Finally the new software architecture and new user interfaces are presented as well as the tests of the resulting application prototype.

Keywords: Mobile Cross-platform development; Xamarin; Android; iOS; Mono; .Net; XAML; Web services.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

M-Marketing: Mobile Markéting.
SMS: Short Message Service.
MMS: MultiMedia Message Service.
GPS: Global Positioning System
Google Trends: Google Tendances des recherches
EDI: Environnement de développement intégré
API: Application Programing Interface
Webinars: Web Séminaire
AOT: Ahead Of Time
JIT: Just In Time
JSON: JavaScript Object Notation
JEE: Java Entreprise Edition
SGBD: Gestionnaire de Base de données
HTML: Hypertext Markup Language
CSS: Cascading Style Sheets
SDK: Software Development Kit
2D: 2 Dimensions
JDK: Java Développement Kit
SDK : Software development Kit
PCL : Portable Class Library
MVVM: Model View View-Model

INTRODUCTION

Les téléphones portables, de plus en plus puissants et intelligents, représentent actuellement la nouvelle tendance des technologies mobiles. Ils permettent, à leurs utilisateurs, d'avoir plusieurs services sous forme d'applications mobiles. Ces dernières peuvent servir d'outils de travail (c.-à-d. bureau mobile, messagerie, monitoring application et même appel de taxi) et aussi d'outils de lecture, de divertissement, d'études et même de promotions. Les applications proposées par les différents fournisseurs sont dédiées généralement à trois plateformes populaires: 1) Androïde; et 2) iOS qui, tous deux, dominent le marché (voir la figure 1) et avec une popularité grandissante le Windows Phone [18]. La figure suivante montre une étude faite par le site web «StatCounter» sur l'utilisation des plateformes mobiles dans le marché nord-américain durant les deux dernières années, le graphe généré montre que la popularité des deux plateformes citées précédemment se distingue de loin des autres plateformes, avec une croissance pour Windows Phone qui est classée en troisième position.

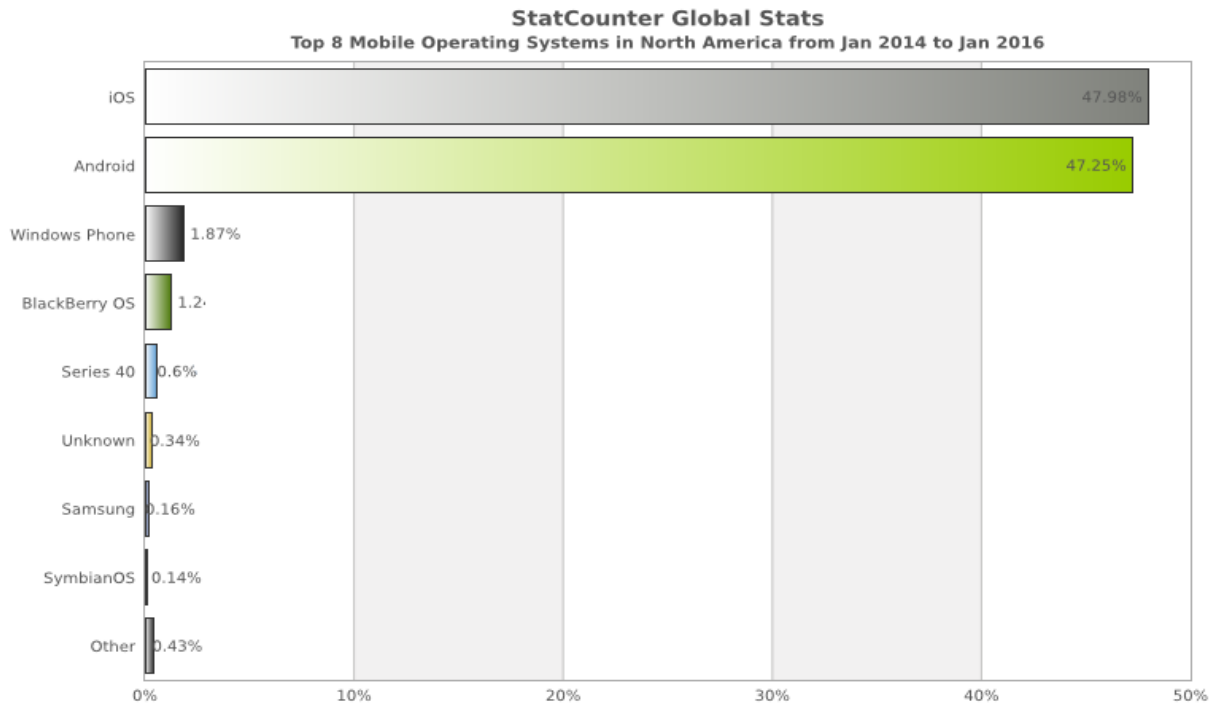


Figure 1 - Graphe de comparaison entre les plateformes mobiles les plus populaires (adapté de [17])

Ce projet de recherche appliquée, de 15 crédits, est réalisé dans le cadre d'une maîtrise en génie logiciel à l'École de Technologie Supérieure (ÉTS) de Montréal. Il implique un entrepreneur, Thierry Maréchal, le président d'une entreprise en démarrage nommée Snoobe. Ce projet représente la poursuite de la réalisation d'un prototype logiciel, effectué par Snoobe, opérant sur la plateforme Androïde. L'objectif de ce projet de recherche est d'investiguer l'utilisation d'un cadriciel multiplateforme pour assurer que l'application mobile actuelle de Snoobe puisse fonctionner sur les deux plateformes mobiles les plus populaires actuellement, soit : Androïde et iOS.

Un autre objectif de ce projet de recherche appliqué vise à ce que l'architecture de l'application résultante permette qu'elle partage le maximum de code entre ces deux plateformes. La fonctionnalité principale, du prototype actuel de Snoobe, permet qu'un ensemble d'informations saisies par l'utilisateur et recueillies automatiquement (c.-à-d. par l'application mobile Androïde) permettent de comparer le forfait téléphonique actuel utilisé à d'autres forfaits disponibles et, conséquemment, de proposer à l'utilisateur les options pour obtenir des économies pour son cas particulier d'utilisation.

Le premier chapitre de ce rapport est consacré à l'étude de la littérature de ce domaine d'affaire ainsi qu'au prototype logiciel existant de Snoobe. Cette revue de la littérature vise l'initiation à ce domaine d'affaires et à l'établissement d'une base de référence qui servira tout au long du processus de développement du prototype expérimental multiplateforme du deuxième prototype logiciel de Snoobe.

Le deuxième chapitre décrit les décisions technologiques et travaux de rétro-ingénierie du prototype Androïde existant, ainsi que la description de méthodes, approches et outils utilisés lors de cette étape de la recherche.

Le troisième chapitre décrit toutes les étapes du développement ainsi que les décisions concernant les technologies utilisées et surtout le choix d'un cadriciel multiplateforme pour applications mobiles. Il débute par la présentation d'une synthèse de l'étude des exigences, la

conception d’une architecture et de sa conception détaillée, la réalisation et les tests du prototype logiciel multiplateforme.

Le dernier chapitre est consacré à présenter et discuter des résultats obtenus, des limites du projet, des commentaires de Snoobe concernant cette expérimentation ainsi que les nouvelles fonctionnalités requises et avenues à explorer dans un projet futur.

CHAPITRE 1
REVUE DE LA LITTÉRATURE

1.1 Introduction

La première partie de ce chapitre est dédiée à la description du domaine d'application de Snoobe, le M-Markéting. Par la suite, le concept du service offert par Snoobe est présenté ainsi que l'objectif visé initialement par son application mobile. Ensuite, un bref historique de la réingénierie est présenté et plus particulièrement les aspects de la rétro-ingénierie logicielle. Un rappel de ses origines est fait, ainsi que ses besoins et ses objectifs. Une partie du chapitre est consacrée aux modèles théoriques de rétro-ingénierie proposés dans la littérature, leurs avantages et leurs inconvénients. Ensuite, une description du prototype logiciel existant de Snoobe est présentée, cette présentation porte sur son concept et les objectifs visés initialement par l'application mobile. La fin de ce chapitre introduit l'approche de développement mobile multiplateforme, proposée pour la réalisation du nouveau prototype, les avantages et les inconvénients de cette approche ainsi que son potentiel pour cette expérience. Le cadriciel Xamarin, utilisé dans l'étude de cas, est introduit, et ses avantages par rapport aux autres cadriciels disponibles sont décrits.

Ce chapitre vise à ce que le lecteur ait une idée claire du domaine d'application de ce projet de recherche ainsi que des processus utilisés pour le réusinage du prototype existant de Snoobe vers un logiciel multiplateforme mobile. Ainsi, l'identification d'un cadriciel multiplateforme, bien adapté au cas de figure de Snoobe, est une étape préalable très importante pour ce projet.

1.2 M-Markéting

1.2.1 Définition

Le domaine du markéting a été impacté, comme la majorité d'autres domaines, par l'avènement et l'expansion des technologies mobiles modernes. Le rapprochement de ces deux domaines (c.-à-d. le markéting et le mobile) a donné la naissance au terme : Mobile Markéting (M-Markéting). Le markéting mobile est défini, dans la littérature : « le processus qui permet la livraison de l'ensemble des messages de markéting, interactif et préautorisé par les clients, par le biais des services de communications basées sur les technologies mobiles » [1].

1.2.2 Historique

Le M-Markéting dépend, tout au long de sa courte histoire, de l'évolution des technologies et des services mobiles. Cette évolution est passée par un certain nombre de générations telles que chacune est caractérisée par le moyen de communication et le type d'information transmise aux clients potentiels. Ces générations de produits et services mobiles ainsi que leurs caractéristiques propres sont décrites dans les paragraphes qui suivent.

La première génération du M-Markéting est apparue avec la première génération des téléphones mobiles apparue en 1983. Elle était basée sur la communication de la voix afin de transmettre les messages de markéting. Cette première génération était une révolution pour le domaine et apportait beaucoup d'avantages par rapport aux méthodes de télémarkéting existantes : telles coûts réduits, l'héritage de tous les avantages du télémarkéting basé sur la téléphonie fixe, la mobilité de cette technologie ce qui permet de rejoindre les clients plus facilement. En autre cet avancement a permis d'avoir plus de précision sur le ciblage des clients potentiels, car le téléphone mobile est relié à une personne et non pas à un foyer [1].

La deuxième génération du M-Marketing est celle des messages mobiles et est apparue dans les années 1990s. Cette nouvelle génération de services utilise les avancements technologiques apportés par la première génération des téléphones mobiles tels que l'amélioration de la disponibilité des clients et la précision des appels téléphoniques et la possibilité d'automatisation des appels. Ainsi, l'utilisateur est devenu capable de faire des transmissions non seulement par voix, mais aussi à l'aide de Short Message Service (SMS) et de Multimedia Messaging Service (MMS). Cette nouvelle technologie a permis aux entreprises de markéting d'avoir un coût de communication encore plus bas que la génération précédente, mais aussi d'avoir la possibilité de faire une automatisation facile de l'envoi des messages à sa clientèle potentielle [1]. De plus, cette nouvelle génération de technologie mobile a permis à ce que les informations de markéting soient encore plus exactes vu que ces dernières sont communiquées par écrit et non seulement à l'aide d'un message vocal. Les messages vocaux sont parfois ambigus à cause de facteurs tels que la différence d'accent, la mauvaise qualité de la communication ou de l'enregistrement du message [2]. Un autre avantage vient du fait que la durée de vie des messages textuels est plus longue. Il a été observé que le message texte reste stocké dans le téléphone, ce qui donne à l'utilisateur l'opportunité de le relire occasionnellement [3].

La troisième génération de technologies mobile introduit l'accès généralisé à internet. Cette nouvelle technologie permet, via internet, d'avoir internet accessible et adapté au mobile. Il y a eu, au début, certains inconvénients, par exemple : les coûts élevés d'avoir internet sur un mobile (c.-à-d. par rapport à internet fixe à la maison), ainsi que la nécessité d'adapter le format des pages markéting web existantes sur internet pages à la taille des interfaces utilisateurs mobiles et aux performances qui varient d'un navigateur internet à un autre s'exécutant sur un appareil doté des capacités de traitement et de mémoire limitées [1]. Malgré ces difficultés initiales, cette nouvelle génération d'appareils devient vite populaire et permet ainsi d'avoir encore plus de moyens markétings pour atteindre les consommateurs [1].

Avec l'amélioration du débit d'internet sur mobile et l'intégration de services de télévision, la naissance à la 4^{e} génération de services mobiles est apparue en 2009 et a permis de bénéficier de tous les avantages du Markéting télévisé existant. Cette nouvelle génération permet, d'une part, de déployer des technologies récentes incluant de courts messages markétings télévisés, ainsi que des animations et effets spéciaux qui influencent la perception (c.-à-d. un élément psychologique) des consommateurs et les attirent encore plus, d'autre part la façon dynamique avec laquelle fonctionne donne aux fournisseurs la chance de mettre à jour leurs contenus en temps réels, ce qui valorise plus la valeur des messages communiqués du fait qu'ils sont plus à jour et plus exacts [1].

L'émergence des téléphones intelligents avec leurs grandes capacités de traitement et de stockage d'information, ainsi que leurs infrastructures logicielles qui permettent le déploiement simple et rapide des applications mobiles, a permis de créer une nouvelle tendance pour le M-Markéting. En effet, de nos jours, des applications mobiles dédiées qui intègrent plusieurs services (c.-à-d. télévision, messages de notifications et souscriptions) sont attractives et interactives. Cette tendance a permis de créer un nouvel espace pour des applications de markéting mobile qu'elles soient génériques servant plusieurs magasins et fournisseurs à la fois comme le cas des applications : BONIAL [4] et PRIXING [5], d'autres propriétaires et sont destinées à certaines marques spécifiques à l'exemple de l'application de CANADIAN TIRE ou celle du grand magasin BEST BUY.

1.2.3 Caractéristiques du m-marketing

Le M-Markéting possède plusieurs caractéristiques qui le rendent différent des autres types de markéting. Les principales caractéristiques de ce domaine sont les suivantes :

- La première caractéristique est que le M-Markéting est basé sur la localisation. En effet, les services du markéting mobile se basent sur trois niveaux de localisation pour fixer leurs stratégies : niveau national, niveau de la ville et même au niveau du quartier [1]. Avec l'arrivé des appareils mobiles dotés du service Global Positioning

System (GPS), cela a permis d'une part aux clients de localiser rapidement les produits et les services qui font partie de leurs centres d'intérêt dans les alentours et même si ces derniers se trouvent dans une région qui leur est étrange, d'autre part cela donne la chance aux fournisseurs d'être plus précis dans le choix des types des messages du markéting, du temps de l'envoi ainsi que le contenu de ces messages.

- La deuxième caractéristique est que le M-Markéting se base sur la personnalisation, cela vient du fait que l'appareil mobile peut servir comment identificateur de son propriétaire et pour les fournisseurs cela donne plus de précision sur le profile visé, et donc plus de précision sur les messages envoyés [1]. Le seul inconvénient de cette caractéristique est que pour obtenir certaines informations, le fournisseur doit obtenir, au préalable, l'autorisation du client pour obtenir des informations concernant sa localisation, informations bancaires comme le numéro de la carte de crédit et d'autres informations.
- Le domaine du markéting classique se base généralement sur la conception et la livraison de messages markéting. Avec l'arrivée du M-Markéting, l'utilisateur peut recevoir une promotion immédiate d'une offre et exprimer ses préférences par rapport à un produit ou un service recherché. De plus, il peut même donner l'autorisation aux fournisseurs de collecter certaines informations, à partir de son appareil mobile, pour que ses messages markéting reçus soient personnalisés [1]. Ceci représente l'interactivité qui est la troisième caractéristique du M-Markéting.

1.3 La rétro-ingénierie

Dans cette partie du rapport, le processus utilisé pour faire l'étude du prototype logiciel existant de Snoobe est introduit. Il s'agit du processus de la rétro-ingénierie. Pour bien comprendre ce processus, un ensemble de définitions sont présentées.

1.3.1 Introduction

La rétro-ingénierie peut être perçue comme une autre façon d'appliquer la science et l'ingénierie. La rétro-ingénierie converge avec l'ingénierie dans certains points à l'exemple que les deux approches sont méthodiques et que les deux ont des objectifs à atteindre, mais elles diffèrent dans les entrées, les objectifs et les résultats à atteindre. Contrairement à l'ingénierie qui permet de passer d'un ensemble d'idées abstraites à une réalisation des produits, la rétro-ingénierie quant à elle permet l'étude d'un produit déjà existant dans le but de lui faire de la maintenance ou lui développer de nouvelles capacités [6].

Historiquement, les premières utilisations de la rétro-ingénierie ont été popularisées par le grand constructeur d'automobile américain General Motors Corporation (GMC) qui voulait compétitionner les produits d'un autre géant de l'automobile américain Ford. Un autre exemple, la rétro-ingénierie, a joué un grand rôle dans l'industrie militaire afin de permettre la maintenance d'équipements qui possédait des pièces manquantes, ceci même après la fin du support du constructeur initial.

Les avantages tirés de la rétro-ingénierie lui ont permis de devenir de plus en plus répandue dans tous les domaines de l'industrie, surtout qu'elle vise de réduire les coûts de la recherche et développement [6].

L'une des forces de la rétro-ingénierie est qu'elle peut être appliquée non seulement à un système complet, mais aussi à des sous-parties du système, et ce dans le but d'améliorer la productivité [6]. Suite à l'introduction de la notion de réingénierie en général, l'application de ce processus au domaine du logiciel sera présentée dans la section suivante.

1.3.2 La rétro-ingénierie logicielle

L'activité de rétro-ingénierie logicielle est présentée dans cette section. Les besoins qui sont à l'origine de ce domaine sont abordés en premier, suivis des modèles, tâches et activités. Finalement les outils sont abordés ainsi qu'une synthèse des difficultés et des bénéfices anticipés par l'utilisation de la rétro-ingénierie dans le domaine du logiciel.

1.3.2.1 Besoins de la rétro-ingénierie logicielle

L'activité de la rétro-ingénierie peut englober plusieurs aspects du produit logiciel. Une telle activité peut être couteuse en termes de temps, de ressources humaines et financières. Pour cette raison, les besoins et les objectifs doivent être définis avant de commencer l'activité, à fin d'éviter toute déviation qui pourrait couter cher par la suite. Cette section est consacrée à la présentation de l'ensemble des questions qui peuvent servir comme guide dans la phase d'étude de l'activité de rétro-ingénierie [6].

Le premier ensemble de questions concerne la présence du système dans le fonctionnement de l'entreprise tel que:

- La nécessité du logiciel au fonctionnement de l'entreprise;
- Le degré d'interfaçage du système en question avec les autres systèmes opérationnels de l'entreprise;
- Le degré d'utilisation du système par les différents opérateurs dans l'entreprise.

Le deuxième ensemble de questions porte sur l'activité de la rétro-ingénierie elle-même et le degré de maturité de l'entreprise à une telle activité, ces questions peuvent être :

- Est-ce que les employés impliqués sont conscients des techniques et méthodes de la rétro-ingénierie?
- On demande si l'équipe de développement (Analyse, concepteur, programmeur) qui a monté le système est disponible, et si les méthodes de développements appliquées sont toujours suivies;
- On cherche le nombre de langage de programmation et technologies utilisées dans la réalisation du logiciel. Aussi, si le code source est disponible et est bien structuré.

1.3.2.2 Modèles de la rétro-ingénierie logicielle

Dans le domaine de la rétro-ingénierie, plusieurs modèles ont été proposés comme abstraction du processus de la rétro-ingénierie afin de guider les employés dans leurs travaux. Cette section présente le modèle initial qui a été proposé par Byrne en 1992 [7]. Le modèle proposé par Telea [8] est plus détaillé par rapport à celui de Burne. Il est illustré par la suite.

1.3.2.2.1 Modèle fondamental de Byrne

Un premier modèle théorique s'inspire du processus de cycle de vie de développement logiciel et se décompose sur quatre niveaux d'abstractions : 1) le niveau conceptuel est celui qui se trouve au sommet de la pyramide et dans lequel les fonctionnalités du logiciel sont définies sans détails; 2) le niveau suivant est dédié à la spécification des exigences et qui permet de définir en détail les fonctionnalités du logiciel; 3) Au troisième niveau, les technologies et l'architecture du logiciel sont déterminées, ainsi que la conception détaillée et les algorithmes à utiliser. Finalement, en bas de la pyramide, se trouve le niveau le plus détaillé, le niveau 4, qui est celui de l'implémentation. Il s'agit, à ce dernier niveau, de transformer toutes les informations et les décisions prises dans les niveaux précédents en lignes de code.

Le processus de la rétro-ingénierie représenté par ce modèle est donc l'inverse du processus

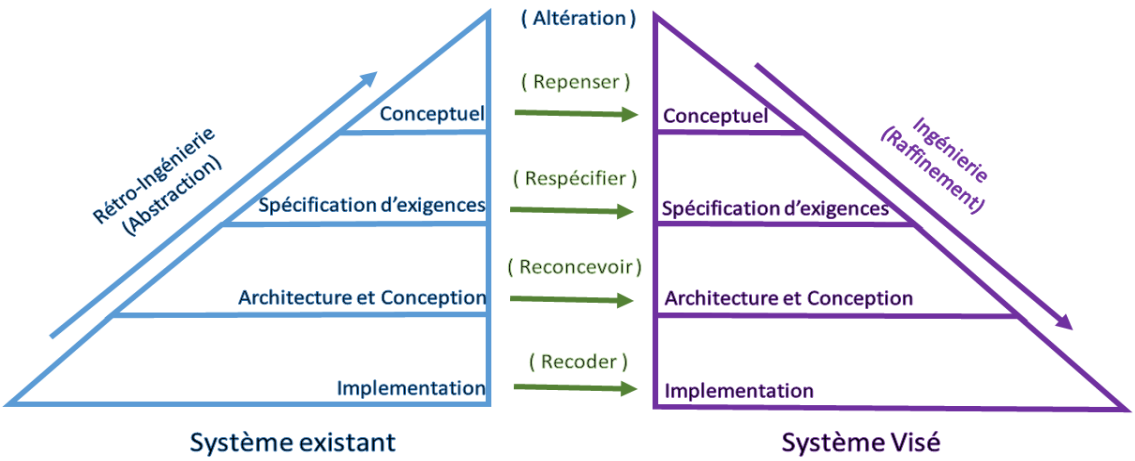


Figure 2 - Modèle général de la rétro-ingénierie logicielle (Byrne, 1992)

de cycle de vie de production d'un logiciel sur lequel se base l'ingénierie logicielle. Il s'agit donc d'un processus d'abstraction qui permet de passer d'un niveau plus détaillé à un autre niveau plus abstrait. La figure 2 décrit ce modèle en détail. Tel qu'il est décrit à dans la figure précédente, le passage de n'importe quel niveau d'abstraction du système existant (le système sujet de la rétro-ingénierie) vers le même niveau dans le système destination. Ce passage est garanti par un ensemble d'actions appelées «Type de changement » [7], ce dernier peut être l'un des cas suivants :

- **Recodage :** Ce type de changement est appliqué au niveau implémentation et vise à changer ses caractéristiques. À l'exemple changement des noms de variables, des méthodes ou autres pour que le code soit plus compréhensible, aussi la restructuration du code, l'ajout des commentaires ou l'adapte du code à des standards de programmation font aussi partie de ce type de modification.
- **Reconcevoir :** Il permet d'intervenir au niveau de la conception et changer quelques-unes de ces caractéristique**s,** il s'agit d'une modification au niveau de l'architecture globale du logiciel, l'introduction ou la découverte de certains patrons de conception, aussi l'amélioration des algorithmes intégrés dans la solution ou même l'amélioration du modèle de données et/ou bases de données.
- **Respécification :** Cette amélioration peut avoir effet au niveau de spécification des exigences. La modification des exigences peut référer à l'ajout, la suppression ou là l'amélioration des exigences déjà existante.
- **Repenser :** Ce type de changement influence sur le niveau le plus abstrait dans la pyramide des niveaux d'abstraction. Un changement à ce niveau signifie la modification des idées sur lesquels la construction du logiciel s'est basée, cela implique que plusieurs parties existantes du logiciel seront modifiées ou refaites complètement.

Les types de changement mentionnés sont tous destinés pour améliorer les caractéristiques d'un niveau d'abstraction bien précis, mais il faut noter que la modification d'un niveau d'abstraction de la pyramide entraine forcément des modifications dans les niveaux suivants.

Certaines stratégies peuvent être établies dès le départ afin de maitriser la propagation d'une modification aux autres niveaux. Byrne a cité, dans son article [7] trois stratégies possibles sont illustrées dans ce qui suit :

- **Stratégie de réécriture :** Cette stratégie est la plus simple. Elle permet qu'une amélioration touche juste un seul niveau d'abstraction sans impacter les autres, à l'exemple des ré-exprimer les exigences sans que ces dernières influencent sur les fonctionnalités déjà réalisées du logiciel ou la modification des noms de variables.
- **Stratégie de retravailler :** Cette stratégie se base sur la séquence de flux suivante : abstraction, amélioration puis raffinement. À partir d'un niveau d'abstraction les modifications sont identifiées, puis un traçage est fait jusqu'à le niveau d'abstraction pour identifier la présentation du système correspondante, l'amélioration est appliquée dans le même niveau dans le système visé et le processus de raffinement est utilisé pour faire propager des modifications sur les présentations du système des niveaux inférieurs.
- **Stratégie de remplacement :** Juste les deux processus d'abstraction et de raffinement sont utilisés dans cette stratégie. La modification du système se fait en commençant par l'abstraction de la partie du système concernée en construisant sa présentation. Par la suite, le processus de raffinement est appliqué sur la présentation obtenue pour réaliser les différentes présentations des niveaux inférieurs du nouveau système visé.

Ce modèle offre donc une bonne vue d'ensemble du cadre général de la rétro-ingénierie. Il se base sur des termes familiers du domaine de l'ingénierie logicielle de manière à faciliter l'assimilation des concepts par les spécialistes qui sont déjà familiers avec le processus de cycle de vie de développement d'un logiciel. Ainsi, la séparation des stratégies possibles permet d'offrir la possibilité qu'une modification touche juste les parties concernées dans le système visé et celui résultant d'une rétro-ingénierie. Malgré ces avantages, ce modèle est d'un très haut niveau conceptuel (c.-à-d. il adresse ce qu'il faut faire), mais en aucun moment il précise comment faire ce travail ou sur quel artéfact devrait être utilisé pour réaliser les différents processus mentionnés.

1.3.2.2.2 Modèle proposé par Télea

Le modèle proposé par Télea, dans son ouvrage (Reverse engineering – recent advances and applications *[8]*) offre plus de détails sur l'activité de la rétro-ingénierie logicielle. Ce second modèle est adapté plus précisément à la rétro-ingénierie des logiciels embarqués. La vue d'ensemble offerte par le modèle permet de séparer les concepts. Cette approche vise à séparer logiquement les différentes parties du processus de rétro-ingénierie. Tel que présenté à la figure 3, le modèle est décomposé en trois parties principales. La première partie (à gauche de la figure 3) est celle décrivant les artéfacts disponibles et sur lesquels l'activité de la rétro-ingénierie sera pratiquée. La deuxième partie (c.-à-d. au centre de la figure 3) présente les processus qui vont être opérés sur les artéfacts. Finalement, la troisième partie du

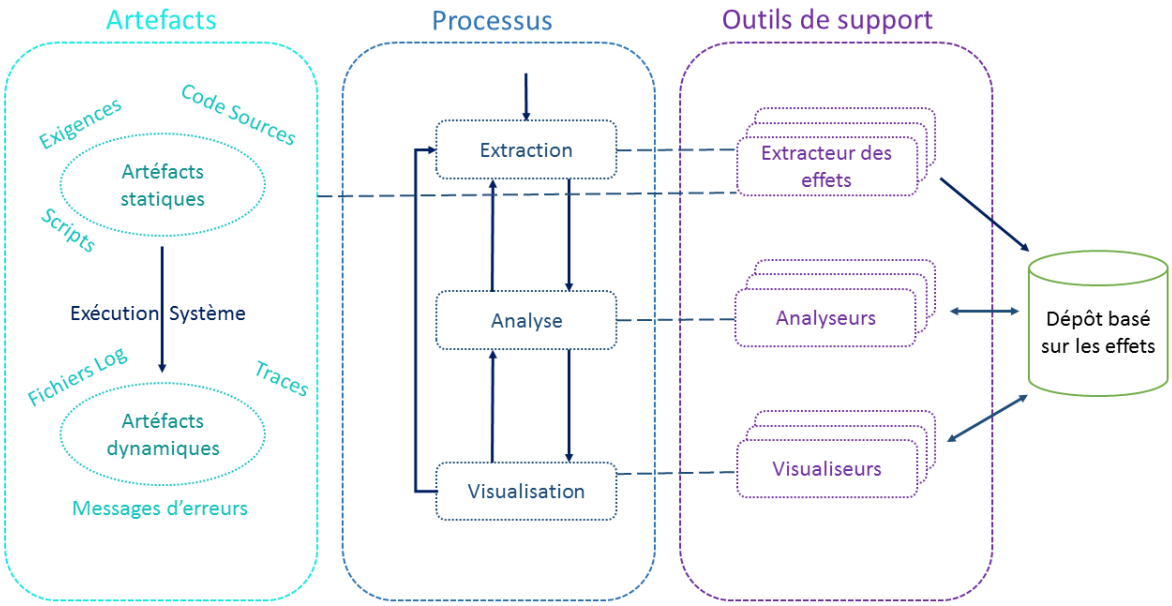


Figure 3 - Vue globale du processus de la rétro-ingénierie (Telea, 2012)

modèle représente l'ensemble des outils qui supportent les processus de rétro-ingénierie. Voici des explications concernant ce modèle:

La première partie du modèle représente l'ensemble des artéfacts qui se séparent eux-mêmes en deux sous-ensembles. Le premier contient tout ce qui fait une représentation statique du

logiciel. Ceci signifie les artéfacts qui sont présents sans la nécessité d'exécuter le programme et cela inclut : les documentations existantes, les scripts de configuration, le code source ... Le deuxième sous-ensemble contient les artéfacts résultants une fois que le logiciel est exécuté, ce qui représente toutes les traces d'exécution du logiciel à l'exemple des fichiers « logs », des messages d'erreurs, son interface, les fonctionnalités disponibles à l'exécution …

La deuxième partie du modèle présente les processus qui permettent de manipuler les artéfacts afin d'en extraire le maximum de connaissances sur le système existant. Il y a trois processus dans ce modèle: 1) l'extraction; 2) l'analyse; et 3) la visualisation. Le premier processus vise à extraire tous les d'indices et les informations qui seront analysées ensuite par le deuxième processus afin d'extraire et de visualiser les informations, les indices (c.-à-d. langages de programmation, cadriciel, les versions …) et les structures voulues. Ces processus sont itératifs, ce qui veut dire que chaque fois qu'il est nécessaire d'extraire plus d'information, alors plus d'itérations sont requises. La décision d'itérer est laissée aux responsables. C'est à eux que revient la décision d'établir le nombre d'itérations nécessaires selon les objectifs d'extraction de connaissances établies dès le départ.

Des outils ont été développés et mis à la disposition des ingénieurs logiciels afin de les assister dans leurs travaux de rétro-ingénierie. Ces outils représentent la troisième partie du modèle et sont classés par catégorie selon le type du processus qu'il leurs sont dédiés. Cet ensemble inclut les extracteurs de faits à l'exemple d'extracteur du code source. Elle inclut aussi les analyseurs qui permettent d'analyser les artéfacts disponibles afin d'identifier les informations utiles et nécessaires, la troisième catégorie des outils est le visualisateur qui permet de présenter les résultats obtenus par les deux autres processus. En général, les trois types d'outils sont intégrés dans un seul environnement de développement et c'est un peu difficile de distinguer entre les trois pour les novices.

Toutes les informations extraites par les processus de ce modèle sont stockées dans une base de connaissance. Elles seront utilisées par la suite afin de concevoir et réaliser le système visé.

Le modèle proposé par Télea est un modèle simple à comprendre par les spécialistes en logiciel. Cela provient du fait que sa structure est simple et que le vocabulaire qu'il utilise est un vocabulaire très familier avec le domaine du génie logiciel. Le plus grand ajout par rapport au premier modèle présenté (c.-à-d. celui de Byrne) est l'aspect détaillé et dynamique du modèle qui vise à représenter les activités permettant d'extraire beaucoup d'informations utiles sur le comportement du système. Malgré tous ces avantages, ce modèle partage le même inconvénient que le premier modèle qui est le manque d'instructions à propos du comment procéder pratiquement pour y arriver. Ainsi, le modèle proposé s'arrête au niveau d'extraction et de stockage des connaissances et il ne montre en aucun cas la correspondance entre le système existant et le nouveau système à réaliser.

1.3.2.3 Tâches de rétro-ingénierie logicielle

Les tâches de la rétro-ingénierie logicielle sont classées dans quatre grandes catégories: 1) trouver les liens entre l'application et le domaine d'application; 2) trouver les liens entre les niveaux abstraits et les niveaux concrets du programme; 3) redécouvrir la structure du programme; et 4) procéder à l'identification des manques entre la syntaxe et la sémantique du programme [6].

Dans la première catégorie, le besoin vient du fait que les programmes informatiques représentent généralement des solutions à des problèmes et non les problèmes eux-mêmes. Dans cette catégorie, le travail fait devrait se concentrer sur la découverte des problèmes pour lesquels répond le programme, d'où la création du lien entre les problèmes du domaine du logiciel et les solutions apportées par ce dernier [6].

Comme le processus de développement logiciel vise à passer d'un haut niveau d'abstraction vers un niveau concret présenté par une conception détaillée et une implémentation, les sous-tâches incluses dans cette catégorie permettent de faire le parcours inverse dans le but de

découvrir les autres niveaux abstractions pour le code sources et/ou la conception détaillée existante.
La troisième catégorie comprend l'ensemble des tâches visant à découvrir les objectifs de la création du programme, ainsi que sa structure générale, de très haut niveau, ce qui représente le découpage du programme en gros composants, accompagnés de la compréhension des fonctionnalités et l'utilité de chacun.

La dernière catégorie, dans cet ensemble, contient les tâches qui permettent de remettre les liens entre les aspects syntaxiques et sémantiques du programme. Souvent, la syntaxe est disponible (c.-à-d. étant donné que le code source dans la majorité des cas est sauvegardé ou obtenu avec des outils spécifiques), mais c'est la sémantique qui représente l'aspect volatile du fait qu'elle est reliée à la présence de la documentation bien entretenue à jour et/ou les personnes qui ont contribué à la construction du système. Les sous-tâches de cette catégorie ont comme objectif de découvrir les aspects sémantiques perdus et de créer le lien entre ces derniers et la syntaxe du programme.

1.3.2.4 Activités de rétro-ingénierie logicielle

L'activité de rétro-ingénierie reflète l'objectif précis qui a incité l'ingénieur logiciel à utiliser une approche ou une autre. La section suivante est présente un ensemble d'approches et de techniques de rétro-ingénierie populaires :

- **Compréhensions du programme :** La compréhension de programme peut être une bonne technique de rétro-ingénierie. En effet, dans la majorité des situations dans le domaine professionnel ou académique, l'ingénieur se retrouve face à des programmes inconnus et qui n'ont aucune documentation disponible. Comprendre ce programme, ses fonctionnalités ainsi que son fonctionnement devient incontournable et la lecture du code source devient une activité de compréhension importante;
- **Découvrir la conception :** le processus de la rétro-ingénierie permet aussi d'approfondir les analyses sur le programme et ainsi découvrir son architecture

d'ensemble ainsi que sa conception détaillée. Cette activité a servi, au début, à titre d'étape préliminaire pour les autres techniques.

- **Comprendre l'architecture de haut niveau :** comprendre et documenter la structure globale d'un programme peut représenter une bonne motivation pour que l'ingénieur logiciel procède à la rétro-ingénierie de ce dernier.
- **Identification des composants spécifique :** Une activité de la rétro-ingénierie peut être menée à fin de découvrir certains éléments essentiels d'un programme, à l'exemple des éléments réutilisables, les éléments qui représentent le maillon faible du programme et qui nécessite une restructuration ou même découvrir les règles d'affaire de l'application en question.

1.3.2.5 Outils de rétro-ingénierie logicielle

Au même titre que pour l'ingénierie logicielle, les outils augmentent la productivité et permettent d'assister l'ingénieur logiciel afin d'obtenir des résultats rapidement. C'est aussi le cas pour la rétro-ingénierie. En effet, beaucoup d'outils peuvent assister les ingénieurs logiciels durant la rétro-ingénierie. Ils ont été développés et sont mis à disposition afin d'accélérer le processus. Ces outils sont classés selon leurs responsabilités dans quatre grandes catégories qui sont les suivantes :

Catégorie 1 : Les outils qui servent à l'analyse du code source et de retourner des informations utiles sur ce dernier à l'exemple de la complexité du programme et la référence module/responsabilité [6];

Catégorie 2 : Cette deuxième catégorie rassemble les outils servant à la conversion du code source d'un langage vers un autre [6], cette conversion est utile pour rendre le code source plus compréhensible en le conversant vers un langage mieux maitrisé ou un langage dont les outils d'analyse sont disponibles et abordables.

Catégorie 3 : Cette catégorie inclut les outils servant à l'amélioration du code source. Ces derniers comprennent génération des opérations de restructuration du code, de son débogage ainsi que sa normalisation. Ces outils servent non seulement à améliorer la qualité du code, mais aussi de montrer l'utilité de la rétro-ingénierie [6];

Catégorie 4 : Cette dernière catégorie regroupe les outils qui permettent de faire la rétro-ingénierie pour le code source et/ou une source de donnée comme les bases de données. En effet, ils permettent de remontrer à un niveau d'abstraction plus haut et de donner une présentation du code/donnée sous forme plus compréhensible et même la découverte des liens entre les modules extraits [6].

Tous les outils cités précédemment peuvent être utilisés ensemble ou séparément. Tout dépend des besoins et des objectifs visés par l'activité de rétro-ingénierie. La section suivante décrit les activités qui peuvent faire sujet de la rétro-ingénierie.

1.3.2.6 Difficultés et bénéfices anticipés de la rétro-ingénierie logicielle

La rétro-ingénierie, comme toute autre pratique du domaine du génie logiciel, apporte des bénéfices lors de son application et, en contrepartie, possède ses défis et ses difficultés. Les défis les plus importants sont présentés à la section suivante :

1.3.2.6.1 Difficultés de la rétro-ingénierie logicielle

Lors d'un projet de rétro-ingénierie, les difficultés suivantes peuvent être rencontrées [1] :

- Le code analysé est souvent très volumineux et mal structuré. Son analyse nécessite beaucoup de compétence et l'utilisation de techniques spéciales, et souvent des outils d'assistances afin d'aider à l'analyse et en tirer les résultats désirés rapidement;
- La documentation est, dans la majorité des cas, non existante. S'il y en a, elle n'est souvent pas à jour ou a été mal rédigée. Donc cette situation demande des efforts supplémentaires afin de comprendre le logiciel;
- Chaque produit logiciel représente en lui-même un cas d'étude différent des autres. D'où l'absence d'une méthode universelle bien définie qui unifie la façon de procéder à la rétro-ingénierie de tous les logiciels.
- La rétro-ingénierie s'applique sur des produits développés avec différentes technologies et langages de programmation. Le personnel disponible pour effectuer la rétro-ingénierie manque souvent d'expertise pour accomplir cette tâche rapidement.

1.3.2.6.2 Bénéfices de la rétro-ingénierie logicielle

Il y a des bénéfices obtenus lors de l'utilisation de techniques de rétro-ingénierie d'un logiciel existant. Ces bénéfices touchent à la fois l'entreprise et le spécialiste qui effectue l'activité. Ces bénéfices sont les suivantes [06] :

- Elle garantit la formation continue des employées et les motivant toujours à découvrir des nouvelles technologies;
- Elle permet à l'entreprise de rester toujours compétitive, cela vient du fait que l'entreprise qui est capable de comprendre les produits de ces concurrents a toujours l'avantage de prendre de l'avance par rapport à eux;
- Contrôler continuellement la qualité de ces propres produits;
- Elle permet de réduire le cout de maintenance des produits logiciels comme elle vise à avoir plus de compréhension et de documentation de l'existant;
- Elle encourage à avoir toujours des produits de qualité. En effet, en faisant des retours en arrière permet de vérifier les parties développées, le critiquer et proposer ce qui est mieux.

La rétro-ingénierie, dans son ensemble, a été présentée dans cette section, la majorité de cette partie a été consacrée à la description détaillée de la rétro-ingénierie appliquée au domaine du logiciel. Les besoins, les activités ainsi qu'un ensemble de difficultés et de bénéfices y ont aussi été présentés. Finalement, une grande partie de ce chapitre a été réservé pour l'explication des modèles théoriques qui vont être utilisés, d'une manière pratique, dans l'expérimentation faite dans le cadre de ce projet.

1.4 Le prototype Snoobe

Cette section est dédiée à la présentation de l'application Androïde existante de Snoobe. Premièrement, le concept de l'application est introduit, suivi de la présentation de plateforme sur laquelle cette première application a été développée ainsi que les raisons de ce premier choix. Finalement, les fonctionnalités sont présentées et décrites de manière à présenter une vue générale de ses fonctionnalités.

1.4.1 Le concept Snoobe

Le concept Snoobe porte sur l'idée d'offrir aux utilisateurs de la téléphonie mobile, la possibilité d'être assistait à fin de déterminer leur position par rapport à forfait utilisé. L'application Snoobe permet de comparait l'utilisation du forfait mobile, en termes de Minutes de communication, de SMS et de données, avec les autres forfaits proposés dans la même région et qui sont enregistrés dans une base de données centralisée. Le résultat retourné par l'application est un ensemble de forfaits qui correspondent au profil de l'utilisateur, et qui vont lui permettre d'avoir une idée claire sur le rapport de consommation/prix de son plan mobile actuelle par rapport aux autres disponibles dans le marché de la téléphonie.

1.4.2 La plateforme mobile de Snoobe

La plateforme mobile visée initialement par l'application Snoobe est Androïde. Cette plateforme a été choisie comme première cible vu qu'elle offre l'accès à un ensemble d'interfaces de programmation, ces interfaces permettent la collecte des données de l'utilisateur après son autorisation. Ces informations sont envoyées au serveur, qui est hébergé dans le nuage, sous forme d'une requête JavaScript Object Notation (JSON) de type GET. Le serveur analyse ces informations, prépare une liste des forfaits et la retourne au client mobile sous format d'un fichier JSON. L'application interprète ces résultats et les affiche à l'utilisateur pour qu'il puisse décider si son forfait actuel est le meilleur choix possible.
La partie serveur a été développée en utilisant la technologie Java Entreprise Edition (JEE). Cette technologie permet de faire le développement Web qu'il soit à des fins de présentation sous forme de pages web ou pour offrir des services Web aux autres applications.

Du côté stockage de donnée, le gestionnaire de base de données (SGBD) MySQL a été choisi pour la base de données de Snoobe du côté serveur.

1.4.3 Les fonctionnalités de Snoobe

Les fonctionnalités de l'application mobile Snoobe sont décrites dans cette section. La figure 4 présente l'interface utilisateur principale de l'application mobile actuelle de Snoobe.

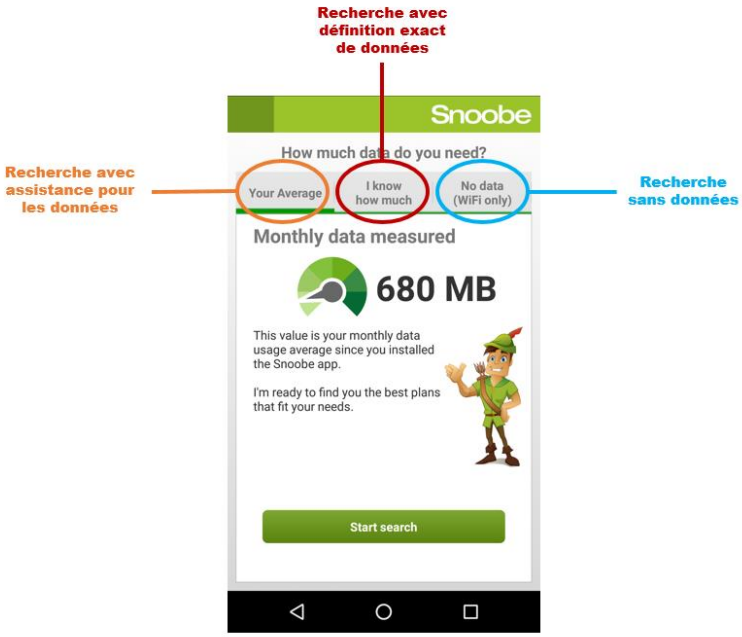


Figure 4 - Interface principale de l'application mobile Snoobe

Toutes les fonctionnalités offertes à l'utilisateur utilisent la même procédure (c.-à-d. composante logicielle) qui sert à extraire les données reliées à l'utilisation du forfait actuel (c.-à-d. minutes entrantes, minutes sortantes, SMS). Il y a trois fonctionnalités principales qui sont accessibles aux clients de Snoobe actuellement. Ces trois fonctionnalités sont présentées dans la section qui suit.

1.4.3.1 Recherche avec assistance pour les données

Dans ce cas de figure, l'application de Snoobe ne connait pas l'utilisation actuelle réelle, en terme de données mobiles, du client. Il est donc nécessaire de lui demander des informations afin de lui faire une proposition de forfait. Cette première fonctionnalité permet à l'utilisateur

de définir son besoin en terme de quantité de données, ce besoin est déterminé après que l'utilisateur ait répondu à une série de questions qui lui permettent de saisir son utilisation pour la majorité des applications mobiles qui consomment les données, à l'exemple de la navigation avec le Système de Positionnement Global (GPS), le taux horaire de visionnement des vidéos en ligne, le temps d'utilisation des médias sociaux et de visionnage des autres sites web. Suite à ce questionnaire, une estimation de l'utilisation de données est faite par l'application Snoobe automatiquement et la recherche de forfaits alternatifs est lancée en se basant sur cette estimation.

1.4.3.2 Recherche avec quantité de données connue

Cette fonctionnalité est semblable à la première, sauf que dans ce cas-ci, l'utilisateur connait son besoin de données et il le choisit parmi une liste de choix offert (par exemple : 500 Mb, 1 Gb, ou plus ...). En se basant sur cette information et les données d'utilisation du forfait, la recherche des forfaits alternatifs est lancée.

1.4.3.3 Recherche sans données

Cette fonctionnalité permet à l'utilisateur d'effectuer une recherche des forfaits qui n'incluent aucune formule pour les données, la recherche est effectuée en se basant uniquement sur l'utilisation du forfait actuel.

1.5 Le développement mobile multiplateforme et Xamarin

Les plateformes de développement pour les appareils mobiles deviennent de plus en plus nombreuses. Chaque plateforme de développement représente, en elle-même, un ensemble de technologies comportant : un ou plusieurs langages de programmation dédiés, des « APIs », des fonctionnalités spécifiques et même un environnement de développement qui inclut un «market place» pour l'achat et la vente d'items et d'applications propres à sa technologie.

Avec tous ces choix, le développement d'une application mobile, couvrant la majorité des plateformes populaires, devient une tâche difficile qui nécessite souvent beaucoup d'effort pour reproduire une même application sur plusieurs plateformes différentes. C'est exactement le problème actuel de Snoobe. Doit-il développer plusieurs applications différentes sur chaque technologie mobile ou bien une seule qui pourrait fonctionner sur plus d'une technologie? Cette problématique est l'une des raisons principales qui a donné naissance à l'approche de développement mobile multiplateforme [10].

En effet, l'approche de développement mobile multiplateforme se base sur l'approche de développer une seule application, qui fonctionnera sur plusieurs plateformes. Cela conduit à diminuer le coût de développement des applications et aussi le coût de maintenance et de la reprise [10].

Conséquemment, plusieurs cadriciels ont été proposés à fin de mettre en application cette approche. Chaque cadriciel repose sur une technologie de développement logiciel déjà existante et utilisée dans la production des logiciels industriels. Chaque cadriciel propose une façon de faire qui implique qu'une certaine partie (ou pourcentage) de code source puisse être partagée entre les différentes plateformes sans être modifiée. Donc, plus le pourcentage de code réutilisable est grand et plus grand est le bénéfice.

Cette prochaine section est consacrée à la synthèse des cadriciels multiplateformes mobiles les plus populaires et les plus utilisés actuellement. La description du cadriciel et de la technologie sous-jacente sont présentées en premier lieu, suivies des avantages et des inconvénients de la proposition.

1.6 Apache Cordova (Phonegap)

Apache Cordova, anciennement nommé Phonegap, est l'un des cadriciels de développement multiplateforme les plus populaires actuellement. Il a été développé initialement par la société Nitobi, lancé à San Francisco en 2008, avant d'être acheté par Adobe et finalement

remis sous License libre Apache sous une formule d'incubation afin d'améliorer sa documentation et son support [11] grâce aux contributions publiques des volontaires. Phonegap est considéré comme le seul cadriciel de développement mobile qui est à la fois, libre et fonctionnes sur sept plateformes mobiles populaires (c.-à-d. iOS, Androïde, BlackBerry OS 6.0, BlackBerry 10, Windows Phone, Ubuntu et Firefox OS) [12].

Le fonctionnement du cadriciel libre Phonegap se base sur le composant « vu web » des plateformes natives. Ce composant se retrouve dans les navigateurs des téléphones mobiles et est capable d'encapsuler le Web comme une application native et ainsi établir une communication entre elle et les « APIs » natives de la plateforme. Ce mécanisme permet à une application mobile d'être exécutée premièrement sur le composant « vue web » qui permet, à son tour, d'interroger les « APIs » natives de la plateforme.

Le cadriciel Apache Cordova utilise donc une technologie Web pour bâtir des applications mobiles multiplateformes. Il se base sur les trois technologies principales utilisées dans les développements Web typiques : il s'agit du trio HyperText Markup Language (HTML), Cascading Style Sheets (CSS) et du JavaScript. Les vues de l'application sont faites avec l'aide de HTML et CSS, tandis que la fonctionnalité est assurée par le JavaScript. Ce dernier assure aussi les appels aux « APIs » natives de la plateforme mobile.

1.6.1.1 Avantages d'Apache Cordova

Le premier avantage de ce cadriciel vient du fait qu'il permet de faire le développement mobile multiplateforme avec une couverture d'un grand nombre de plateformes mobiles et une couverture complète de leurs « APIs ». L'autre avantage est qu'il permet aux développeurs web d'avoir un accès rapide au domaine du développement mobile sans à avoir fourni beaucoup d'efforts pour son apprentissage. De plus, il permet à l'application finale développée d'être à la fois disponible sur le Web et sur le mobile. Bien sûr il faudra tenir compte du type de fonctionnalité, par exemple : si ça ne nécessite pas beaucoup de fonctionnalité spécifique aux téléphones mobiles à l'exemple d'accéléromètre. Un avantage

intéressant de Phonegap, et qui le rend une solution vedette par rapport aux autres, est qu'il est non seulement gratuit, mais aussi disponible en logiciel libre, ce qui encourage la majorité des entreprises à l'utiliser : solution assez complète et aucuns frais n'est exigé pour commencer à l'utiliser.

1.6.1.2 Inconvénient d'apache Cordova

Le premier inconvénient de Phonegap vient du fait qu'il se base sur la technologie Web. Ceci nécessite que les interfaces des applications soient les mêmes pour toutes les plateformes, ce qui n'est pas souhaitable dans tous les cas. Pour certaines applications, les producteurs d'applications mobiles préfèrent que leurs applications suivent une conception graphique native de la plateforme visée, car les utilisateurs y sont habitués. Aussi, l'amélioration de Phonegap est fortement reliée au développement de la technologie web. Finalement, Phonegap est critiqué pour sa faible performance vu qu'il ne s'exécute pas directement sur la plateforme, mais à travers la composante « vue web », ce qui augmente le temps d'exécution et diminue la performance d'exécution.

1.6.2 Appcelerator (Titanium)

Titanium est un cadriciel, disponible sous licence libre, développée par l'entreprise Appcelerator. Il permet de faire le développement d'applications mobile multiplateforme avec une couverture de toutes les « APIs » des trois plateformes mobiles les plus populaires. Son approche est similaire à celle de Phonegap, basée sur la technologie Web, mais avec des mécanismes de compilation d'exécutions différentes que ceux de Phonegap. La technologie de Appcelerator repose sur la technologie JavaScript, mais en ajoutant des spécificités par un ensemble d' « APIs » propres à ce cadriciel. Ces « APIs » permettent, non seulement d'accéder aux fonctionnalités natives de la plateforme, mais aussi elles permettent de réaliser les interfaces de l'application mobile. L'exécution des applications mobiles développées avec Appcelerator est réalisée sur une composante propriétaire, il s'agit d'une machine virtuelle JavaScript. Finalement, Appcelerator offre également son propre IDE qui est adapté à sa technologie et à son style de développement.

1.6.2.1 Avantages d'Appcelerator

L'avantage principal d'Appcelerator est qu'il permet de faire le développement multiplateforme avec sa propre technologie. Aussi, il couvre la totalité des fonctionnalités natives de la plateforme cible. Appcelerator est aussi disponible sous une licence libre de droits, ce qui lui permet de bénéficier de tous les avantages des produits sous licence libre. Cela facilite aussi son acquisition et son adaptation d'une manière simple et rapide. En plus de ces avantages, son IDE propriétaire proposé permet le développement d'une manière encore plus simple et plus rapide avec une adaptation totale avec sa technologie.

1.6.2.2 Inconvénient d'Appcelerator

L'inconvénient majeur d'Appcelerator est qu'il couvre seulement trois plateformes mobiles: iOS, Androïde et BlackBerry. De plus, l'utilisation d'une machine virtuelle JavaScript, sur laquelle s'exécutent les applications développées, influence négativement sur la performance des applications mobiles générées.

1.6.3 Sencha Touch

Sencha Touch est un cadriciel JavaScript utilisé pour le développement des applications Web multiplateformes. Il se base principalement sur les normes HTML et CSS, réalisés dans le cadre de développement Web, destinés aux appareils mobiles. Sencha Touch se caractérise par son ensemble de composants graphique riche, qui lui permet de couvrir tous les composants graphiques des plateformes mobiles natives [13]. Grâce au système de vecteurs graphiques, Sencha Touch offre la possibilité d'avoir des applications qui s'adaptent avec la taille d'écran de tous les appareils mobiles, ainsi que toutes les résolutions. Il permet aussi de supporter instantanément les orientations portrait et paysage grâce au mécanisme des dispositions adaptatives [13].

Ce cadriciel offre un ensemble d'« API » riche qui permet l'exploitation des données à partir de la majorité des sources de données (c.-à-d. base de données, service Web et bien d'autres). Il propose aussi un « package » riche qui permet de visualiser les données récupérées dans de multitudes formes graphiques, qu'elles soient dans des diagrammes sous forme de barres, de ligne ou même des diagrammes circulaires. Pour accéder aux « APIs » natives des appareils mobiles à l'exemple de la caméra, GPS, Accéléromètre ou autre, Sencha Touch s'appuie sur le cadriciel Phonegap présenté dans une section passée. En effet, Sencha Touch a la possibilité d'être intégré complètement avec Phonegap pour pouvoir produire des applications qui ont la capacité d'accéder à toutes les « APIs » natives [14].

1.6.3.1 Avantages de Sencha Touch

Le premier avantage de Sencha Touch est qu'il facilite la réalisation des interfaces graphiques utilisateurs grâce à un ensemble riche de composants graphiques disponibles. Cet ensemble inclut des listes, des carrousels, des formes, des menus, et des barres d'outils adaptés aux appareils mobiles. Sencha Touch possède aussi un ensemble d'« APIs » HTML5 qui permet l'accès à certaines d'« APIs » native de l'appareil, ce qui améliore la performance ainsi que le temps de chargement de l'application [14].

1.6.3.2 Inconvénient de Sencha Touch

L'inconvénient majeur de Sencha Touch est qu'il se base sur d'autres cadriciels pour accéder les « APIs » natives des appareils mobiles. Aussi, il ne couvre actuellement que quatre principales plateformes mobiles : iOS, Android, Windows Phone et BlackBerry.

1.6.4 Corona SDK

La plateforme Corona SDK est une plateforme qui a été réalisée initialement pour supporter le développement mobile multiplateforme des jeux vidéo et des livres électroniques. Sa version récente offre beaucoup d'améliorations qui permettent de supporter les applications mobiles dédiées au commerce électronique [15]. Corona SDK se base un langage de «

Scripting » nommé Lua (le nom Lua provient du portugais et signifie la lune). L'interpréteur de ce langage a été développé en langage C et le but de son développement est de donner aux applications existantes la possibilité d'être étendues. Lua a été largement utilisée dans le développement de jeux vidéo. Corona SDK est considérée comme l'une des meilleures plateformes de développement de jeux vidéo à 2 dimensions (2D), il offre aussi l'avantage de faire des appels à n'importe quelle librairie native des langages C, C++, Objectif-C et même JAVA [16].

Ce cadriciel couvre présentement les plateformes mobiles iOS, Androïde et Windows phone. Il permet également de faire le développement de livres électroniques pour la plateforme Kindle et le développement des applications pour OSX de Apple. Corona SDK est offert dans une version gratuite qui peut être utilisée pour le développement et la publication des applications mobiles de type iOS, Androïdes, Windows Phone et Kindle. Ce cadriciel offre aussi une version payante (c.-à-d. version professionnelle) qui contient des composants additionnels qui permettent l'ajout de fonctionnalités riches comme les micros-transactions ou encore de bénéficier des dernières mises à jour de versions [15].

1.6.4.1 Avantages de Corona SDK

L'avantage principal de Corona SDK est le support pour le développement mobile multiplateforme des jeux vidéo. Corona SDK est aussi avantageux pour son support d'un grand nombre de normes de l'industrie logicielle : OpenGL, 3D OpenAL, et d'autres comme Facebook, JSON, SQLite. Aussi, Corona SDK vient avec un simulateur qui permet le visionnage des changements effectués en temps réel. Finalement, la réalisation d'applications à l'aide du langage de « Scripting » permet une facilité d'apprentissage et de l'application des notions du langage Lua.

1.6.4.2 Avantages de Corona SDK

Le premier inconvénient commun entre la majorité des cadriciels est qu'il se limite dans sa couverture juste aux plateformes les plus populaires : iOS, Androïde. Corona SDK est

concentrée sur le développement des jeux vidéo, quand il s'agit de développement des applications mobiles autres que les jeux vidéo, Corona SDK n'est pas le meilleur cadriciel de développement mobile multiplateforme à choisir. Comme aucun IDE spécifique à ce cadriciel n'a été réalisé, le développeur se trouve face aux choix des « IDEs » tierces parties qui supportent Lua. Ce choix ne favorise en aucun cas les opérations de débogage et de maintenance des applications mobiles développées avec Corona SDK. Finalement, il ne supporte pas tous les accès aux « APIs » natifs principaux des plateformes mobiles, à l'exemple d'accès à la caméra pour la plateforme Windows mobile.

1.7 Conclusion du chapitre 1

Une vue d'ensemble des sujets pertinents à cette recherche appliquée a été faite dans ce premier chapitre, cette revue a englobé tous les domaines et toutes les notions reliées à ce projet de recherche appliquée de 15 crédits. Premièrement, la notion de M-Markéting, qui est le domaine d'application de l'application Snoobe, a été présentée. Par la suite l'approche théorique appliquée pour cette étude de cas, afin d'extraire les connaissances nécessaires à partir du logiciel existant de Snoobe a été présenté. Ensuite, le concept du service de l'application Snoobe a été présenté de manière à introduire le lecteur au domaine d'application. À la fin de ce chapitre, le développement multiplateforme mobile ainsi que les technologies disponibles, de ce domaine, ont été présentés accompagnés de leurs avantages et inconvénients. Finalement, un cadriciel a été choisi et sera utilisé pour l'étude de cas de ce projet.

Pour ce projet, le cadriciel Xamarin qui a été choisi. Il sera décrit en détail dans le chapitre suivant. Premièrement, une vue d'ensemble de sa technologie sera présentée. Par la suite ses avantages qui ont fait qu'il a été sélectionné sont décrits, suivis de ses inconvénients. Les justifications de ce choix technologique sont finalement débattues.

CHAPITRE 2
CHOIX TECHNOLOGIQUE ET L'APPLICATION DE RÉTRO-INGÉNIERIE

2.1 Introduction

Ce deuxième chapitre présente les réalisations faites pendant cette recherche appliquée. En effet, la sélection du cadriciel de développement multiplateforme utilisée sera justifiée. Par la suite, une description des activités de rétro-ingénierie est décrite. Une partie de ce chapitre révèle les difficultés rencontrées et qui ont influencé les décisions prises tout au long de la conception du prototype multiplateforme. À la fin de ce chapitre, le lecteur aura une bonne compréhension des enjeux de conversion du prototype existant de Snoobe vers un logiciel multiplateforme mobile. Ainsi, il suivra le parcours du travail de rétro-ingénierie du prototype existant et les travaux de modernisation effectués.

2.2 Choix technologique

Dans cette première section, la technologie de développement mobile multiplateforme est présentée. Une description du cadriciel et la justification de ce choix sont faites, par la suite, les avantages de ce choix sont présentés.

2.2.1 Le cadriciel Xamarin

L'appellation Xamarin ne signifie plus qu'un simple cadriciel de développement mobile multiplateforme, mais plutôt une solution quasi complète qui permet la réalisation, le test, le support ainsi que la surveillance des applications mobiles intéressantes. Avec Xamarin, le développement d'application mobile se base sur un seul langage de programmation qui est le C# en s'appuyant sur la technologie .NET, et tout en gardant la possibilité de faire des implémentations spécifiques pour chaque plateforme, avec le Java pour Android et l'Objectif-C ou Swift pour iOS. L'utilisation de la technologie Xamarin permet de créer des applications mobiles d'une façon plus simple, plus rapide et surtout plus agréable.

2.2.2 Critères de sélection de Xamarin

La sélection de la technologie Xamarin n'a pas été le fruit du hasard, mais plutôt suite à une étude de l'ensemble des technologies concurrentes du domaine. Cette étude porte principalement sur les qualités et les avantages que peut offrir chaque cadriciel et\ou technologie par rapport aux autres. Dans cette section, un passage sur l'ensemble des critères qui ont favorisé Xamarin par rapport aux autres est fait. Les critères utilisés sont les suivants :

1. **Xamarin répond-il au besoin ?**

Le premier critère sur lequel la sélection de Xamarin a été faite est la faisabilité. En d'autres termes, est-ce que Xamarin, comme technologie répond aux besoins de réalisation de l'application de Snoobe? La réponse à cette question a été évidemment favorable. Xamarin permet de viser les trois plateformes mobiles les plus populaires et qui ont été exigées par le client, il s'agit d'iOS et Android, et Windows Phone pour une version future. Aussi, il permet de réaliser tout ce qui peut se faire, en mode natif, pour ces trois plateformes. Il supporte même toutes leurs nouvelles versions à leur sortie. Donc, le nouveau prototype multiplateforme de Snoobe peut être réalisé avec la technologie Xamarin.

2. **Popularité de Xamarin**

La popularité d'une solution peut refléter la maturité de cette dernière, la facilité de son adaptation par le grand public, son succès et beaucoup d'autres critères. Pour ce faire, l'outil Google Tendances des recherches (c.-à-d. Google Trends) a été utilisé pour évaluer la popularité relative de Xamarin avec les autres cadriciels concurrents: Apache Cordova anciennement appelé Phonegap, Appcelerator Titanium, Sencha Touch et Coroca SDK. Le graphe présenté à la figure suivante (voir figure 5) a été obtenu comme résultat :

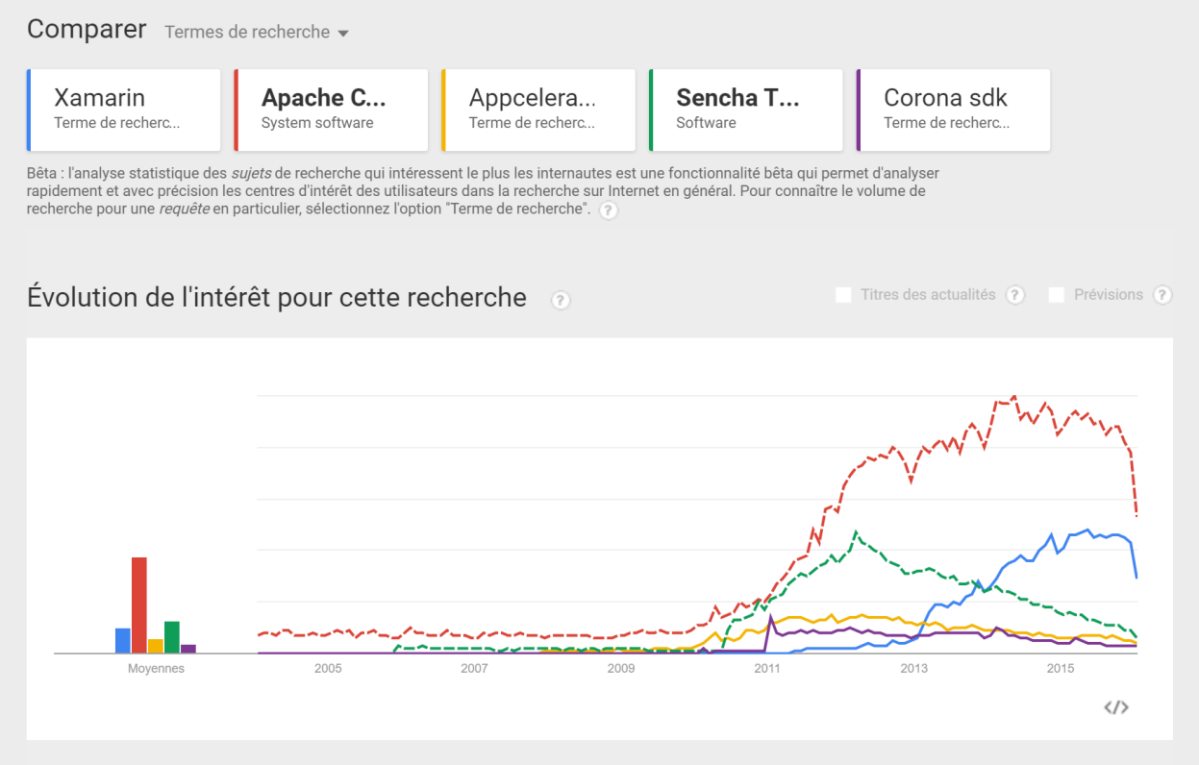


Figure 5 - Graphe comparatif de la popularité des cadriciels multiplateformes mobiles

À la lecture de ce graphe, la popularité de Xamarin et Apache Cordova se distingue de toutes les autres technologies, et ce avec un grand écart. Mais, il est clair que l'évolution de Xamarin est la plus remarquable depuis 2013 par rapport aux autres, tel qu'il est en évolution continue depuis sa première apparition dans le début de l'année 2011. Cette popularité reflète qu'il y a de plus en plus de développeurs et de compagnies qui portent intérêt à cette technologie, en d'autres termes, reflète son succès. La popularité d'une technologie est un facteur très important dans la décision de l'utiliser. Plus une technologie est populaire et plus elle survivra longtemps.

3. Richesse de la plateforme Xamarin

Tel qu'il a été mentionné précédemment, Xamarin n'est pas seulement un cadriciel de développement mobile multiplateforme, mais aussi une solution de développement riche et

complète. Cette solution est composée de quatre composants principaux tels que présentés à la figure 6 :

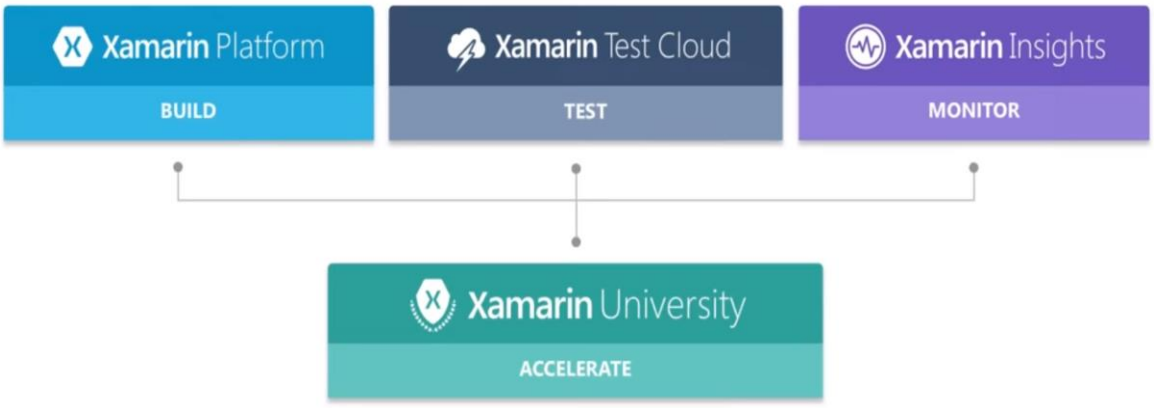


Figure 6 - Les quatre composantes de la technologie Xamarin [9]

Tel que décrit à la figure 6, Xamarin est composé de quatre composantes principales. Chacune couvre une partie du projet de développement mobile à l'aide de Xamarin. La description de l'utilité de chacune des quatre composantes suit :

- **Xamarin Plateforme :** Cette composante englobe l'Environnement de Développement intégré (EDI) qui est propre à Xamarin et qui offre tous les outils permettant d'assister le développeur lors du développement de l'application mobile, et ce rapidement et efficacement. Elle contient aussi toutes les interfaces de programmation d'application nécessaires (API) qui permettent la réalisation de l'application mobile avec le code partagé ou l'accès aux « APIs » natifs spécifiques aux plateformes. Aussi, cette plateforme permet le débogage et la compilation des applications, qu'elles soient sur des simulateurs ou sur des appareils réels. L'EDI Visual Studio peut être aussi utilisé pour la réalisation des applications comment remplaçant de Xamarin Studio.

- **Xamarin Test « Cloud » :** L'un des grands problèmes des applications mobiles est que le développeur doit prévoir la compatibilité de son application avec la majorité des téléphones mobiles qui existent sur le marché. Ce problème se manifeste beaucoup plus pour les appareils supportant les systèmes Android et Windows Phone vu que ces derniers sont utilisés par plusieurs fabricants. D'où la nécessite pour que l'application soit compatible graphiquement et aussi par rapport à la performance de l'appareil et ses composants matériels. Xamarin répond à ce besoin par une plateforme de test hébergée sur leurs serveurs dans le nuage et qui sont accessibles via internet, cette plateforme permet d'exécuter l'ensemble des tests de l'interface graphique à partir de milliers d'appareils simulés dans la plateforme, prévoir les échecs et les corriger avant que l'application soit publiée et exploitée par l'utilisateur final.

- **Xamarin Insights :** Une fois que l'application est en production, il est nécessaire d'effectuer un suivi de son exécution pour détecter les failles qui n'ont pas été persuadées dans les phases de développement et les corriger. Beaucoup d'équipes de développement utilisent des systèmes de surveillance externe pour répondre à ce besoin, à l'exemple de Google Analytics, Amazon Mobile Analytics ou même Fabric. Xamarin offre sa propre solution de surveillance qui permet de garantir un suivi en temps réel grâce au système d'alerte qui envoie des rapports directement suite à l'apparition d'une défaillance sur l'une des instances de l'application mobile. Aussi, il permet un regroupement intelligent des défaillances selon la similarité des utilisateurs et des appareils, ou encore la possibilité de consulter l'état de l'appareil avant et après la défaillance pour détecter les évènements clés à la résolution du bogue.

- **Xamarin University :** Xamarin offre son propre système de formation, ce système vise à former des utilisateurs certifiés à utiliser cette technologie ou même dans le but de la mise à niveau des développeurs existants. Cette formation est garantie grâce à

un ensemble de formules proposées. Ces dernières peuvent être des classes en ligne avec des cours enseignés par des professionnels, des cours sont aussi disponibles pour des lectures et des examens hors ligne. Xamarin met aussi à la disposition des professionnels pour une consultation directe dans le but d'aider les développeurs dans l'architecture, la conception ou encore même les détails d'implémentation de leurs applications actuelles.

4. Support et documentation

Xamarin dispose d'un riche ensemble d'outils servant à supporter les utilisateurs dans leur utilisation de cette plateforme. En plus de Xamarin université décrite dans le paragraphe précédent, Xamarin offre un guide complet d'utilisation de ces « APIs » qui sont disponibles et accessibles en ligne sur leur site web. Cette description est faite d'une manière simple et pédagogique pour permettre, même aux débutants, d'accélérer leur processus d'apprentissage. Aussi, ils offrent plusieurs exemples d'applications avec leurs codes sources pour aider les utilisateurs à comprendre et surtout assimiler les manières et les bonnes pratiques de réalisation des applications en utilisant cette technologie. De plus, un ensemble de forums et de blogues sont mis à la disposition des développeurs Xamarin. Ces espaces sont animés par des professionnels qui assistent les utilisateurs pour répondre à leurs questions via les forums, et écrire continuellement des articles et les publier dans les blogues, ces articles servent comme guide pour tous les outils et toutes les nouveautés de la plateforme. Pour ajouter de l'interactivité à leur système de support. Xamarin offre des séances d'apprentissage via des web séminaires (c.-à-d. des Webinaires) pour présenter des nouveautés ou encore faire connaitre de nouveaux composants. Les utilisateurs ont aussi la possibilité, grâce à ces Webinaires, de poser des questions et obtenir de l'aide pour pallier de leurs difficultés d'utilisation de la plateforme.

Tous ces outils et ces avantages de support ajoutent une grande valeur pour que Xamarin soit choisi comme technologie de développement pour n'importe quel développeur voulant opter pour un cadriciel de développement mobile multiplateforme.

5. Avantages apportés à l'application

En plus de tous les avantages cités précédemment, Xamarin offre aussi un ensemble de spécificités à l'application mobile qui le rend différent de toutes les autres technologies. Ces avantages portent sur les trois principales dimensions suivantes de l'application mobile : Son interface graphique, les accès aux « APIs » et les performances l'application.

Du côté de l'interface graphique, contrairement aux autres cadriciels qui se basent principalement sur les technologies web (c.-à-d. HTML, CSS et JavaScript), Xamarin permet à partir d'un seul code partagé, écrit en C# avec Xamarin.Forms, d'avoir des interfaces natives pour chaque plateforme. Cette approche donne l'impression que l'application a été développée nativement. La figure suivante schématise la façon avec laquelle le code de l'interface graphique est partagé :

Figure 7 - Code de l'interface graphique native partagée avec Xamarin.Forms [9]

Concernant les « APIs », Xamarin garde toujours la possibilité d'accéder aux « APIs » natives de l'une des trois plateformes. Ceci est fait dans le but de garder toujours la possibilité d'implémenter des spécificités qui nécessite l'accès à des « APIs » propres à une plateforme par rapport aux autres.

En ce qui concerne la performance de l'application mobile, Xamarin offre des mécanismes de compilation spécifiques pour chaque plateforme qui permettent d'avoir des performances équivalentes à celle d'une application développée en native. En effet, Xamarin se base sur la

compilation anticipée (AOT) pour générer des applications prêtes à s'exécuter des iOS et sur la compilation à la volée (JIT) pour le cas des applications Android. La figure suivante montre le processus de la compilation pour les deux plateformes mobiles iOS et Androïde.

Figure 8 - Processus de compilation AOT et JIT de Xamarin [9]

2.2.3 Cadriciel Grails

Le choix technologique existant du côté serveur est basé sur la technologie Java (le J2EE) et utilise le cadriciel Grails (en ce moment à la version 2.24). Cette technologie est conservée et sera utilisée vu qu'elle répond à toutes les exigences de réalisation de la nouvelle application mobile. En effet, l'application actuelle, du côté serveur, est bien structurée et offre des services web qui seront réutilisés tout en offrant la possibilité d'en ajouter d'autres, au besoin, d'une manière simple et rapide.

2.3 La rétro-ingénierie du système Snoobe

Cette partie du rapport est consacrée à la présentation de la rétro-ingénierie du cas d'étude Snoobe. Le modèle proposé est décrit en premier, et par la suite l'application de techniques, les résultats, les technologies et l'architecture découverte y sont présentés. La dernière partie de cette section traite de l'analyse du code source qui a permis d'identifier les parties intéressantes et utiles pour le développement du prochain prototype logiciel de Snoobe.

2.3.1 Les buts à atteindre

Avant de commencer le travail de rétro-ingénierie, un ensemble de buts et d'objectifs ont été fixés. En effet, vu la nature du projet, qui est la migration d'une application mobile vers une application mobile multiplateforme, les objectifs de rétro-ingénierie suivants ont été établis:

- Inventorier toutes les composantes du prototype actuel;
- Identifier les technologies impliquées dans chaque partie du prototype, ainsi que les versions utilisées;
- Parvenir à compiler et faire fonctionner le prototype existant, en mode local, dans le but de permettre l'analyse comportementale des fonctions du prototype actuel;
- Identifier les parties existantes réutilisables dans le nouveau prototype;
- Découvrir l'architecture et la conception :
    - Du côté client mobile, pour identifier le maximum de fonctionnalités de l'application actuelle.
- Préparer un texte synthèse de cette expérimentation et utiliser les informations découvertes.

2.3.2 Modèle proposé

En tenant compte de la spécificité du projet en cours qui consiste à faire la rétro-ingénierie d'un système existant comme une étape préliminaire, pour produire par la suite un nouveau système qui porte sur les mêmes concepts, mais en utilisant toute une nouvelle technologie. Les modèles présentés dans la littérature n'étaient pas applicables séparément à ce cas d'étude, alors une combinaison de ces modèles sera utilisée pour la rétro-ingénierie du prototype logiciel actuel de Snoobe.

Les deux modèles de rétro-ingénierie pris en considération sont ceux présentés dans le chapitre 1, il s'agit du modèle fondamental proposé par Byrne [7] et celui proposé par Télea [8] dans son livre sur la rétro-ingénierie. Le modèle de Byrne offre la vue globale qui permet de faire la rétro-ingénierie d'un système déjà existant, il montre également l'action à

entreprendre pour atteindre l'autre niveau du système visé. Ce modèle aidera à définir les parties importantes de l'application qui seront utilisées ainsi que les parties correspondantes dans le système visé. Étant donné que le code source de l'application existante ne sera pas réutilisé (vu que c'est un nouveau cadriciel, avec sa propre technologie et son propre langage de programmation qui sera utilisé), alors le but principal est de découvrir les fonctionnalités et les exigences du prototype existant. Le modèle de Byrne décrit le cadre global du processus, mais qu'il n'offre pas de détails sur comment procéder, alors le modèle de Telea est introduit dans cette section pour combler ce vide et fournir en le fusionnant avec celui de Byrne un modèle utilisable dans ce projet. La figure 9 illustre une proposition novatrice de modèle de rétro-ingénierie qui est la fusion des deux précédents. Tel que présenté à cette figure, ce modèle est adapté aux cas ou la seule source d'information disponible du système existant est le code source, ainsi qu'un ensemble fichiers de configuration (c.-à-d. le niveau implémentation). Une étape préliminaire dans ce nouveau modèle est requise : la collecte de tout type de données sur le code et la configuration du système existant. Suite à cette étape, un ensemble de données situées au niveau de l'implémentation, dans le logiciel existant, représentent aussi l'ensemble de départ du processus.

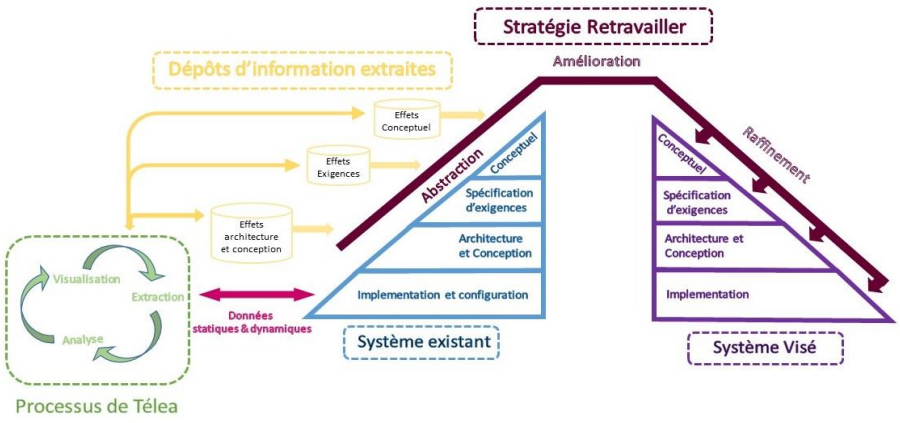


Figure 9 – Modèle de rétro-ingénierie proposé pour la rétro-ingénierie et la migration du prototype de Snoobe

Les processus de rétro-ingénierie de Télea sont appliqués au départ, d'une manière itérative, afin d'extraire des informations sur les autres niveaux d'abstraction du système existant (c.-à-

d. l'architecture et la conception et ses spécifications). Les informations extraites sont alors classées dans une base de connaissance qui sera utilisée dans le processus de réingénierie suivant les étapes proposées par Byrne [7]. La stratégie de retravailler les contenus a été appliquée aussi dans ce nouveau modèle. Elle permet de remonter en abstraction à partir d'un niveau d'abstraction du système existant, d'appliquer des améliorations et de propager ces améliorations dans les différents niveaux d'abstraction du prototype logiciel visé grâce au processus de raffinement.

2.3.3 Les étapes de la rétro-ingénierie du système Snoobe

Les travaux menés pour la découverte, l'analyse ainsi que l'extraction des connaissances du prototype actuel de Snoobe sont décrits dans cette section. Ces travaux sont présentés à travers un ensemble d'étapes de rétro-ingénierie suivantes :

2.3.3.1 Étape 1 : Collecte des informations

L'entreprise en démarrage Snoobe ne possède pas d'endroit physique où sont localisés un ensemble d'ordinateurs utilisés pour le développement de leurs logiciels. Le premier prototype a été réalisé par des collaborateurs qui travaillent tous à distance chacun sur son ordinateur personnel. Cette situation a rendu la tâche de collecte des données encore plus difficile. La première étape de collecte d'information a été faite lors de réunions successives avec le président, Thierry Maréchal. Le but de ces réunions était d'obtenir le maximum d'information sur la documentation et le dépôt de code source existant. Suite à ces réunions, l'ensemble des informations suivantes a été identifié :

- La documentation existante sur un compte de l'entreprise sur la technologie FlexPark de Atlassian;
- La structure du prototype actuel :
    - Le prototype mobile publié sur Google Play;
    - La partie serveur est hébergée sur le Nuage d'Amazon;
    - La base de données est hébergée aussi sur le Nuage d'Amazon.

- Le code source pour l'une des dernières versions du prototype;
- Le contact de Stephane Rainville, l'un des collaborateurs de la partie serveur;
- Les accès aux comptes d'Amazon et de FlexPark.

Une réunion avec le développeur principal, Stéphane Rainville, a eu lieu et par la suite quelques rencontres sur Skype pour obtenir des connaissances sur la partie serveur. Finalement, une version du code source a été récupérée ainsi que des connaissances sur la technologie utilisée dans le développement, ces dernières sont les suivantes :

- IDE : Intellij IDEA Ultimate version;
- Le cadriciel Grails.

À la fin de cette première étape, l'information recueillie permet d'analyser, plus en profondeur, le prototype existant.

2.3.3.2 Étape 2 : Faire fonctionner le prototype existant en mode local

Suite à une première collecte de données et de code source, l'étape suivante consiste à compiler le code source du prototype existant, en mode local, dans le but de s'assurer que les composants récupérés sont fonctionnels. Pour ce faire, une analyse des dossiers de projet a été faite afin de découvrir l'IDE utilisé ainsi que le cadriciel sur lequel le prototype a été développé. La partie serveur a été analysée en premier, une première inspection a permis de distinguer deux dossiers de tout le reste. Il s'agit de : «.idea » et «grails-app». Le dossier «.idea» indique le l'IDE Intellij Idea a été utilisé dans le développement tandis que le deuxième «grails-app» indique que le cadriciel Web Grails a aussi été utilisé dans le développement. Un autre fichier découvert, le « snoobeserver-grailsPlugins.iml » a été analysé afin d'extraire la version utilisée. Ces informations vont été localisées dans les lignes de code suivantes :

- «<orderEntry type="jdk" jdkName="1.7" jdkType="JavaSDK" />»
- «<orderEntry type="library" name="grails-2.2.4" level="application" />»

Ces deux dernières lignes de code indiquent que la version de JAVA utilisée est la version 7 et celle de Grails est la version 2.2.4. Un script de sauvegarde de la base de données a été aussi retrouvé : «snoobeserver\src\sql\». Ce script indique que le SGBD MySQL a été utilisé avec la version 5.5.16. La compagnie Jetbrains a été contactée et une licence académique a été obtenue afin d'utiliser l'IDE Intellij Idea Ultimate.

Après l'obtention des technologies, des bonnes versions et de tous les composants de la partie serveur du prototype existant, la mise en place du serveur peut être effectuée. L'ordinateur utilisé pour cette reconstruction est un portable DELL XPS P31F doté d'un processeur quad-core i7 2.30ghz, d'une mémoire de 16 gigabits et d'un disque dur de 1 téraoctet. Le système d'exploitation Windows 10 version éducation est aussi utilisé.

Le Java Développement Kit (JDK) version 7 et Grails Version 2.2.4 ont été téléchargés, installés et configurés (c.-à-d. la configuration des variables d'environnement). Par la suite, le MySQL version 5.6 a été téléchargé et mis en place. Le nom d'utilisateur et le mot de passe d'accès à la base de données ont été mis à jour dans le fichier de configuration : «grails-app\conf\ DataSource.groovy».

Finalement, le projet a été importé et compilé après l'ajout du «plugin» Grails pour Intellij Idea, lors de la compilation l'IDE a automatiquement détecté tous les autres «plugins» décrits dans le fichier de configuration et les a téléchargés. Lors de l'exécution, l'erreur suivante est apparue :

« *FAILURE: Build failed with an exception.*

** What went wrong:*

Task 'grails-run-app' not found in root project 'snoobeserver'.

** Try:*

Run gradle tasks to get a list of available tasks. Run with --stacktrace option to

get the stack trace. Run with --info or --debug option to get more log output.

BUILD FAILED

Total time: 42.763 secs
Task 'grails-run-app' not found in root project 'snoobeserver'.
12:29:27 AM: External task execution finished 'grails-run-app'. »

Plusieurs essais ont été menés afin d'essayer de résoudre cette erreur, mais aucun n'était en mesure de la corriger. L'étape suivante a été d'essayer l'exécution directe de la commande «grails run-app» dans le dossier source du projet. L'exécution a été faite avec succès et l'application serveur déroulée sur l'ordinateur portant l'adresse IP : 192.168.1.70 sous le port 8080.

La deuxième partie de cette étape consiste à exécuter l'application mobile pour qu'elle communique avec l'application serveur lancée sur l'hôte 192.168.1.70. Il est donc nécessaire ici d'effectuer des recherches dans le dossier de l'application pour obtenir des indices sur l'IDE et le cadriciel utilisés. L'environnement de développement découvert est celui d'Androïde studio. Ce prototype logiciel avait été réalisé avec le moteur de production Gradle. En premier lieu, la dernière version d'Android Studio a été téléchargée et installée. Par la suite, le moteur Gradle, version 2.4, a aussi été téléchargé et une variable d'environnement indiquant le dossier source a été créée.

Pour faire exécuter une application Androïde, le kit de développement logiciel (SDK) approprié a aussi été requis, téléchargé et installé. Une fois que cet environnement de développement a été préparé, le projet du prototype actuel Snoobe a été importé et deux modifications ont été requises avant la compilation du logiciel :

- La première est de modifier la localisation d'Androïde SDK dans l'un des fichiers du projet. Il s'agit du fichier «*snoobeandroid\local.properties*», le paramètre «sdk.dir» a été pour qu'il pointe vers dossier contenant le SDK installé «*C\:\\Users\\Billy\\AppData\\ Local \\ Android \\sdk*»;
- La deuxième étape consistait à chercher et trouver l'attribut contenant l'adresse du serveur afin de la modifier pour que l'application pointe vers le nouveau serveur qui s'exécuter sur l'hôte 192.168.1.70. Cet attribut a été trouvé dans la class

«*snoobeandroid\SnoobeAndroid\src\main\java\com\snoobemobile\remote\messages\JSONfields.java*». Il s'agit de l'attribut « *BASE_ADDRESS* » dont sa nouvelle valeur est devenue «*http://192.168.1.70:8080/snoobeserver/*».

Suite à ces ajustements, le prototype logiciel existant a été compilé avec succès et exécuté sur un téléphone mobile LG Nexus 5 sur Androïde.

Avec l'exécution de l'application mobile, tout le prototype logiciel existant (c.-à-d. logiciel client, logiciel serveur et base de données) a fonctionné correctement et toutes les parties ont pu communiquer entre elles afin de simuler l'exécution du système, mais en mode local.

2.3.3.3 Étape 3 : Comprendre les fonctionnalités du système

Une fois les données sont collectées et le logiciel est exécuté, l'étape suivante est d'investiguer l'application mobile afin de découvrir toutes les fonctionnalités qu'elle offre. Ceci est fait en citant l'ensemble des cas d'utilisation offerts aux utilisateurs mobiles. Ces cas sont démontrés dans le diagramme de cas d'utilisation suivant :

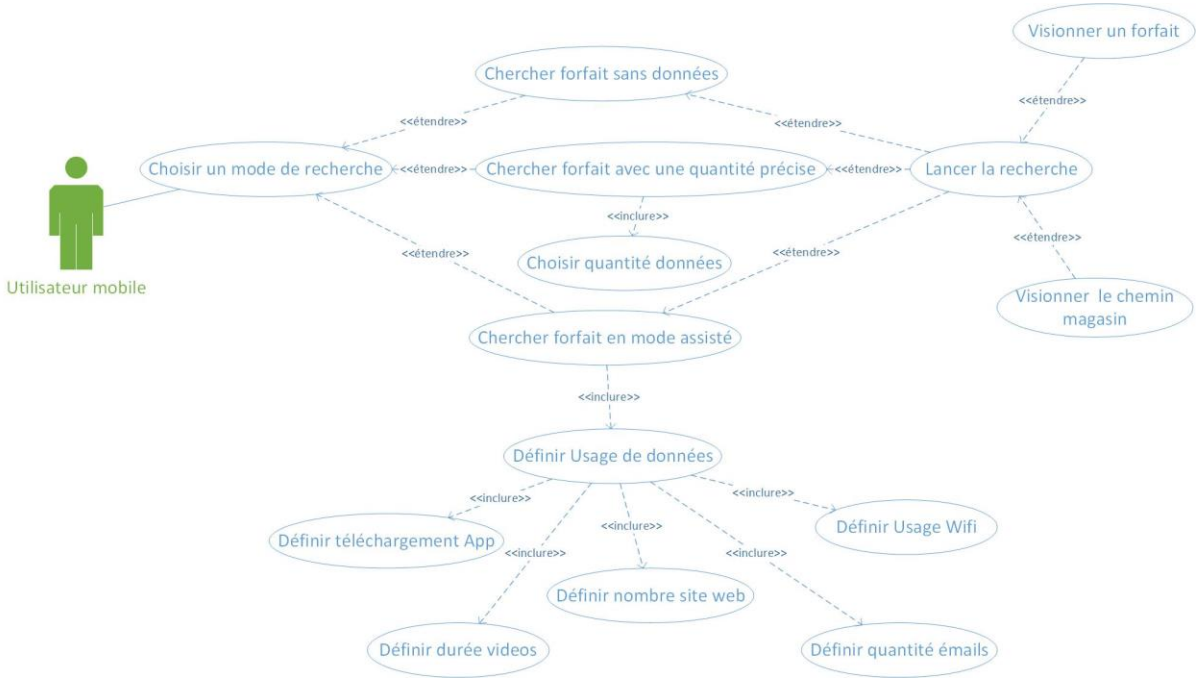


Figure 10 - Diagramme de cas d'utilisation de l'application mobile Snoobe

Tel que présenté par le diagramme ci-haut, la première option qui s'affiche à l'utilisateur, lors du lancement de l'application, est le choix du type de la recherche voulue. Il s'agit des trois cas suivants :

- Recherche sans données;
- Recherche avec une quantité connue de données;
- Recherche assistée, elle inclut un ensemble de questions pour aider l'utilisateur pour avoir une estimation de son utilisation de données mobiles.

Suite à la saisie de la définition de la quantité des données désirées, l'application offre à l'utilisateur la possibilité de lancer sa recherche. Une fois que le résultat de la recherche est obtenu, l'utilisateur peut choisir de consulter plus de détails sur l'un des forfaits affichés en exploitant la fonctionnalité de visionnage du forfait. L'adresse du magasin, qui offre ce forfait, est affichée parmi les détails. L'utilisateur a également la possibilité de visionner le trajet vers ce magasin en utilisant son application GPS.

Les fonctionnalités décrites, dans cette partie du rapport, ont été obtenues en analysant visuellement l'exécution de l'application qui concerne l'acteur «utilisateur mobile». D'autres fonctionnalités ont aussi été découvertes, par la suite, lors de l'analyse du code source.

Du côté du serveur, l'analyse détaillée va se concentrer sur les services web offerts à l'application mobile et non les fonctionnalités offertes à l'utilisateur final concernant l'administration du système.

2.3.3.4 Étape 4 : Analyse des données

Dans cette partie du rapport, une analyse du code source et de la base de données est décrite. Comme les deux applications :1) mobile et 2) serveur, ont été développées avec l'aide du langage Java, l'utilisation des outils de rétro-ingénierie a été prévue (vu que plusieurs outils dédiés à ce langage ont été identifiés dans la revue littéraire). En effet, durant ce projet plusieurs essais d'utilisation d'outils de rétro-ingénierie ont été effectués sans qu'aucun n'ait offert de résultats très intéressants. Dans le but de guider les futures recherches, dans ce domaine, une réflexion rapide, concernant ces outils, est présentée dans ce qui suit.

Le premier outil de rétro-ingénierie essayé, Architexa, est un outil puissant qui offre la possibilité de générer plusieurs diagrammes. L'inconvénient d'Architexa est qu'il est offert seulement en tant que «plugin» pour l'IDE Eclipse. Conséquemment, il est seulement applicable pour les projets qui ont été développés à l'aide de cette technologie. Dans le cas du prototype de Snoobe, les deux projets Androïde et Grails ont été importés dans Eclipse, afin d'expérimenter avec cet outil, mais les diagrammes générés n'étaient pas lisibles, ni utilisables.

Le deuxième outil de rétro-ingénierie essayé a été PlanUML. PlanUML est un «plugin» Gradle et permet de générer des diagrammes de classe et d'autres types de diagrammes. Le problème avec cet outil est que son utilisation doit être prévue au début du projet afin d'inclure des scripts spécifiques (c.-à-d. qui sont ensuite utilisé par cet outil) dans le code

source original. Ceci n'étant pas présent dans le prototype de Snoobe, il n'a pas pu être utilisé.

Suite à l'essai de ces deux outils, il a été conclu que l'analyse du code source pourrait être faite manuellement. La prochaine section présente des vues organisationnelles sous forme d'organisation modulaire pour l'application mobile et serveur. Par la suite, la rétro-ingénierie faite sur la base de données est présentée.

2.3.3.4.1 Vue modulaire de Snoobe mobile

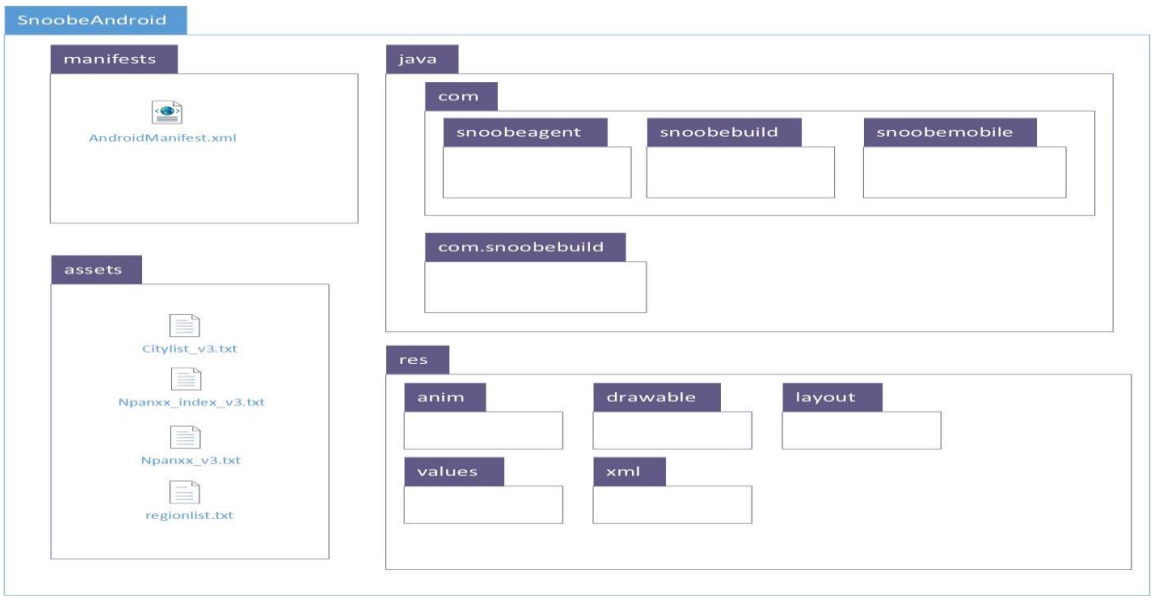


Figure 11 - Diagramme de package de l'application Snoobe

La figure 11 présente une représentation graphique du prototype actuel de Snoobe. Il se compose de quatre principaux «packages». Le premier porte le nom «Manifests» et contient le fichier principal de la configuration de l'application Androïde. Le deuxième «package» porte le nom de «res» et contient toutes les ressources nécessaires pour l'application : il s'agit des interfaces graphiques classées dans le sous-dossier «layout», les animations dans «anim», les icônes dans «drawable», la définition des chaines de caractères et des images dans «values» et les fichiers de structuration des données localisées dans «xml». Le troisième

«package» se nomme «assets» et inclut des fichiers textes contenant un ensemble de données nécessaires au fonctionnement de l'application, par exemple : la liste des villes canadiennes et américaines ainsi que leurs coordonnées géographiques. Le quatrième et dernier «package java» représente la partie la plus importante de cet ensemble et contient tout le code source du premier prototype logiciel de Snoobe. Ce code est contenu dans le sous-ensemble «com» et partagé entre les trois sous-package suivants :

- Snoobeagent : Contiens le code source de l'agent Snoobe;
- Snoobebuild : Contiens la classe de configuration de la compilation de l'application mobile Snoobe;
- Snoobemobile : Ce «package» englobe la grande partie du code de l'application, il sera étudié dans la section suivante afin d'approfondir la compréhension de l'application Snoobe, de ses fonctionnalités ainsi que de pouvoir extraire son architecture.

2.3.3.4.1.1 Vue modulaire du package Snoobe mobile

Une étude du paquage «Snoobemobile» a été faite et le résultat obtenu est décrit à la figure 12. Les couleurs attribuées aux «packages» reflètent leurs responsabilités dans le logiciel. La couleur bleue a été attribuée aux «packages» contenant des classes dont la fonctionnalité principale est de contrôler le fonctionnement de l'application. Les «packages» en couleur mauve contiennent les classes qui représentent les modèles de données utilisés par l'application, ces informations sont retirées à partir des fichiers de données qui sont empaquetées avec l'application elle-même ou récupérées des «APIs» d'un tiers parti (c.-à-d. à l'exemple des informations de la géolocalisation ou d'utilisation du téléphone). Les «packages» en couleur verte sont responsables sur la collecte des données à partir des «APIs» externes, qu'elles soient propres à la plateforme mobile ou disponible via internet. La dernière catégorie des «packages» rassemble ceux qui sont utilitaires.

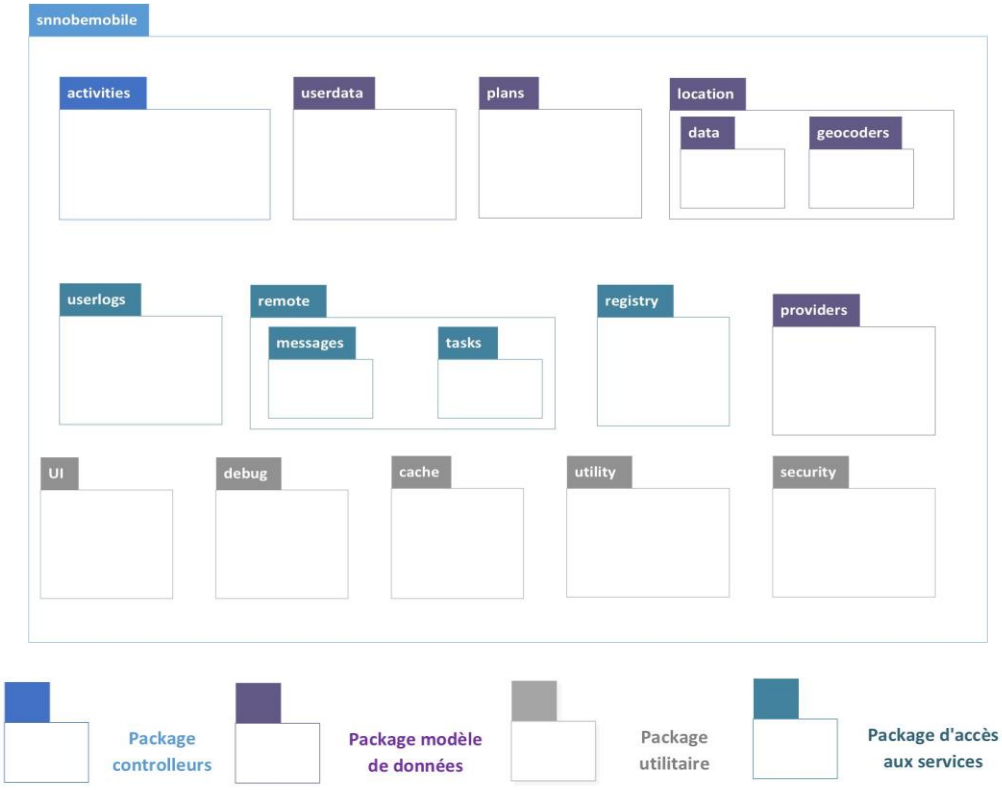


Figure 12 - Digramme du «package» pour le module snoobemobile

L'ensemble des «packages» présentés à la figure 12, ainsi que leurs responsabilités sont les suivants :

- Activities : ce dossier contient toutes les activités de l'application, elles représentent tous les états possibles de l'application et contrôle au même temps l'application à chaque état;
- Userdata : Toutes les classes représentant les données d'utilisation du forfait, cela inclut le nom des SMS, le nombre et la durée des appels entrant et sortant ainsi que les données mobiles;
- Plans : Ce «package» englobe les classes qui représentent le modèle logique d'un forfait mobile;
- Location : toutes les classes responsables sur la représentation des fichiers de données contenant les noms des villes (Américaines et Canadiennes) et leurs localisations,

aussi les classes responsables de la localisation de l'utilisateur actuel sont incluses dans ce «package»;

- Userlogs : Ce «package» contient les classes qui s'assurent les fonctionnalités de récupération des données d'utilisation du forfait actuel de l'utilisateur;
- Remote : la responsabilité principale des classes de ce dossier est de représenter tous les messages qui peuvent être envoyés par l'application au serveur ainsi que les tâches qui effectuent l'envoi. Une classe « SnoobeServer.java » permet d'accéder aux «APIs» du serveur;
- Registry : Lors du lancement de l'application ou même durant son exécution, un ensemble d'informations devraient être sauvegardées et vérifiées continuellement pour guider le comportement de l'application. L'ensemble des classes, qui assurent la cohérence des informations et leurs validités, est inclus dans ce «package»;
- Providers : Ce «package» contient une seule classe qui représente l'ensemble de fournisseurs mobiles;
- UI : Un ensemble d'actions supplémentaires sur l'interface graphique sont requises dans l'application, à l'exemple de faire apparaitre les détails d'un élément. Les classes assurant ce genre de fonctionnalités sont incluses dans ce dossier;
- Cache : Ce «package» contient une seule classe « planCaches,java » qui permet de faire la gestion des forfaits sauvegardés dans la cache du téléphone mobile;
- Utility : Un ensemble de classes servant comment classes utilitaires ont été développées et rassembler dans ce «package», ces classes offrent des services comme le calcul de la distance entre deux points, la détection du fournisseur téléphonique actuel ou même retourner les informations de l'utilisateur (numéro de téléphone et l'émail).
- Security : Ce «package» inclut les classes responsables des opérations de sécurité telle que le cryptage des messages à envoyer;
- Debug : Ce «package» a été développer dans le but de support les opérations de débogue de l'application, il contient un ensemble de classe qui capte les bogues s'ils apparaissent et de les communiquer au serveur;

2.3.3.4.1.2 Analyse du «package activities»

Suite à la collecte d'information et l'obtention de la compréhension globale du fonctionnement de tous les «packages», une étape d'investigation détaillée a été faite sur le «package activities» afin de déterminer le comportement général du prototype logiciel. Ce «package» a été choisi, car il représente le point central de toute l'application et une idée claire peut être obtenue suite à son analyse. Afin de s'assurer, de l'enchainement des activités ainsi que de leur présence dans la version actuelle, une analyse dynamique, de plusieurs cas d'utilisation, a été requise et a été faite à l'aide d'un téléphone mobile :

- Avec l'activation du réseau mobile et le service de localisation GPS;
- Sans l'activation du réseau mobile et sans l'activation du GPS;
- Avec l'activation du réseau mobile et sans l'activation du GPS;
- Sans l'activation du réseau mobile et avec l'activation du GPS.

Suite à l'analyse de ces cas d'exécution, certaines activités ont été identifiées comme : non incluses dans la version actuelle, et d'autres, utilisées dans une version de test de l'application Snoobe (c.-à-d. non destinée aux utilisateurs finaux). La figure 13 résume l'ensemble des activités ainsi que leurs enchainements. Le prototype actuel Snoobe se lance toujours avec l'activité «SplahActiviy» qui présente une interface utilisateur d'accueil, d'une durée de quelques secondes, visant à permettre l'exécution, en arrière-plan, d'un ensemble de tests de configuration.

L'utilisateur doit approuver un ensemble de conditions, présentées par l'activité «PrivacyActivity». Par la suite l'activité «DataPlanActivity» est exécutée. Elle représente l'activité principale ou la majorité des traitements fonctionnels du prototype actuel sont faits. Les activités «BetterPlanActivity» et «NoPlanActivity» sont utilisées pour présenter les résultats de la recherche des forfaits. Pour montrer les détails d'un forfait choisi, l'activité «MarchantActivity» est utilisée. D'autres activités utilitaires sont présentent comme «MessageActivity» qui est utilisée pour afficher les messages d'erreurs ou plus de détails, et

les deux activités «FacebookActivity» et «FacebookReminder Activity» pour interfacer avec Facebook et permettre de laisser des commentaires.

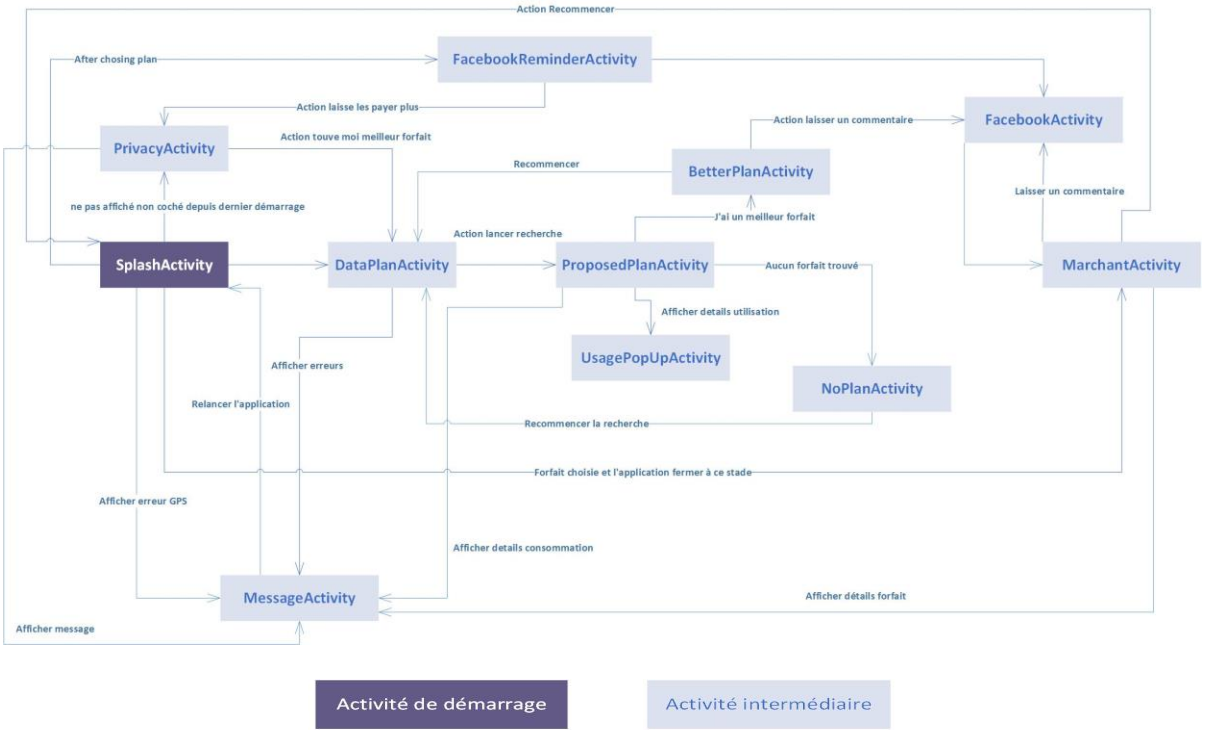


Figure 13 - Enchainement des activités de l'application Snoobe

2.3.3.4.1.3 Architecture de l'application Snoobe mobile

Une étude approfondie a été menée dans le but de découvrir la structure architecturale du prototype actuel de Snoobe. Cette étude va permettre d'avoir une compréhension de l'ensemble du logiciel existant et permettre la réalisation, plus rapide, d'une nouvelle application qui doit inclure les composants actuels qui doivent être migrés.

Une première étape consiste à décrire le système actuel à l'aide de modules de haut niveau. Le résultat obtenu est un ensemble de cinq modules représentés à la figure 13. Le premier module est celui de la présentation. Il inclut toutes les vues ainsi qu'un ensemble de fichiers de mise en forme (c.-à-d. cela inclut la majorité des fichiers présents dans le dossier «values»

du projet. Un deuxième module représente le point central du contrôle de l'application, il englobe toutes les classes de type activité de l'application. Le module modèle est aussi représenté et contient un ensemble de classes qui incluent des concepts clés de l'application : par exemple : les SMS, les appels et même les données mobiles. Les activités du module contrôle accèdent aux différentes sources données via un ensemble de tâches qu'elles lancent durant leur exécution. Ces tâches jouent le rôle des services et sont incluses dans le module service. Les données manipulées par l'application proviennent de deux sources : 1) un ensemble de fichiers de données et de configurations qui viennent avec l'application et sont enregistrées sur le téléphone de l'utilisateur une fois que l'application est installée, et 2) la d'un ensemble d'«APIs» du serveur Snoobe accessible via internet.

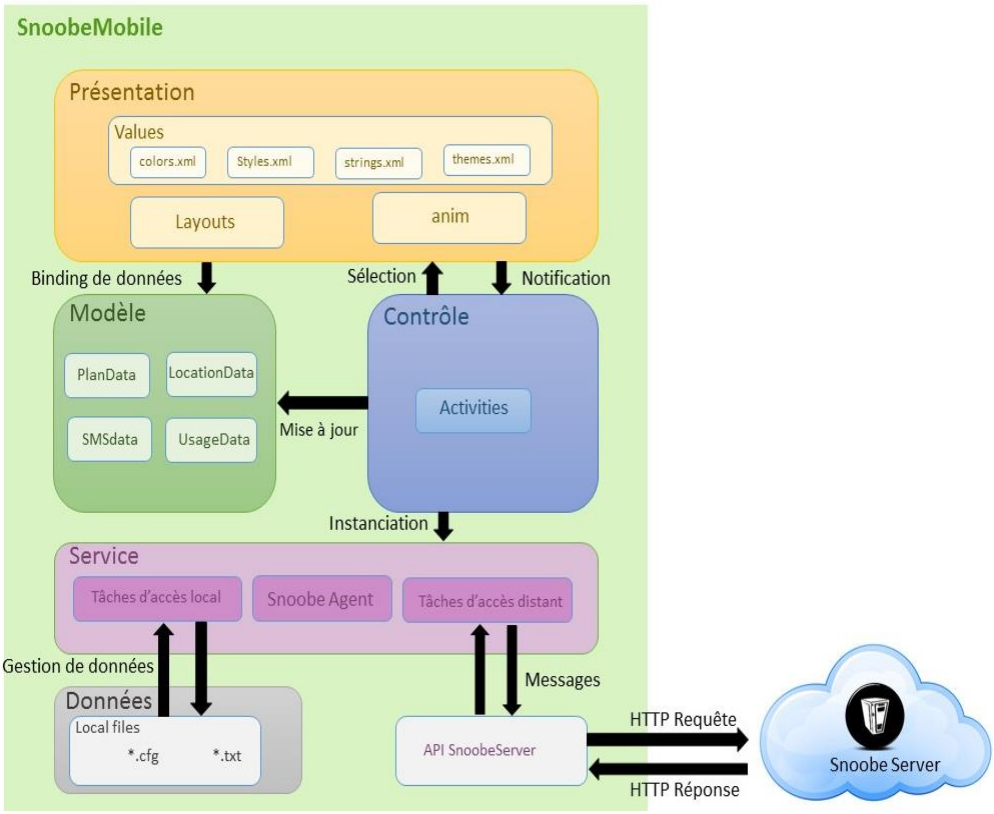


Figure 14 - Architecture de l'application Snoobe Mobile

Durant cette première phase de la rétro-ingénierie détaillée de la partie mobile de Snoobe, un ensemble de tests ont été exécutés (voir section 2.3.2) afin d'obtenir une compréhension du

comportement de prototype actuel de Snoobe. Un ensemble d'interfaces utilisateurs ont été obtenues suite à ces tests. Ils sont présentés à l'Annexe 1 de ce rapport.

2.3.3.4.2 Diagramme de classe de la base de données Snoobe

Afin d'extraire encore plus de connaissances, sur la partie serveur et son fonctionnement, une étude de la base de données a été menée. Le but principal, visé par cette étude, est d'extraire un diagramme modélisant la structure des données afin d'en faciliter la compréhension. Pour ce faire, l'outil «MySQL WorkBench» proposé par MySQL a été utilisé. Sa version 6.3.5 a été téléchargée et installée. Ensuite, cet outil a été connecté à la base de données du serveur Snoobe installée en local et l'action « Base de données -> rétro-ingénierie » a été lancée.

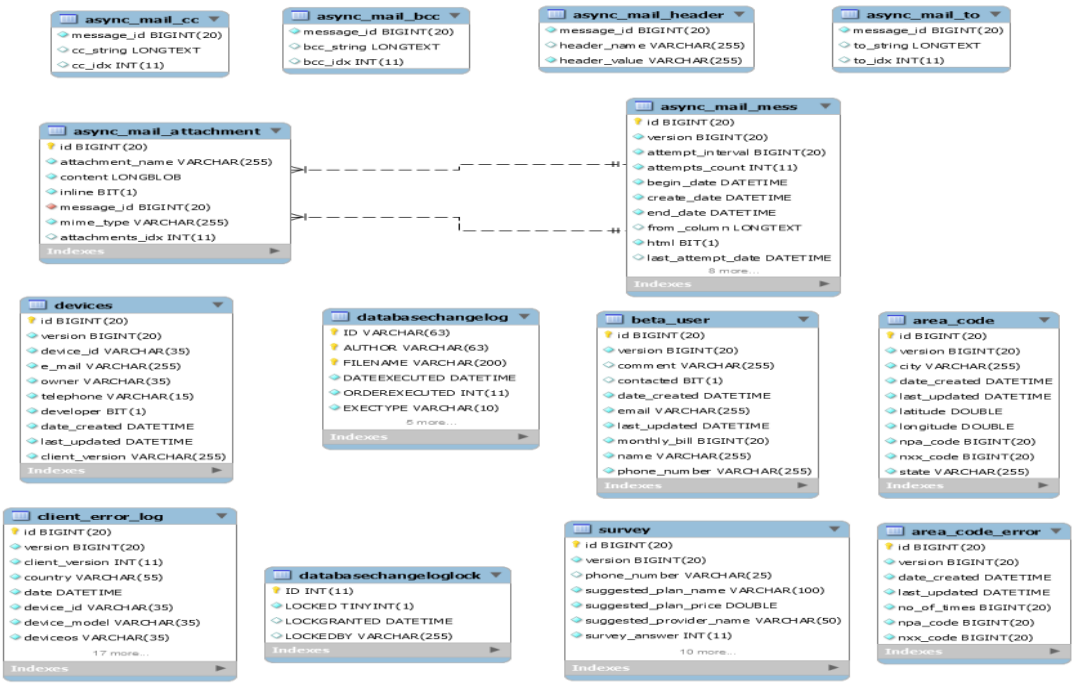


Figure 15 - Partie 1 du diagramme de la base de données de Snoobe Server

Le diagramme de la base de données a été généré automatiquement, un ensemble d'action de raffinement et d'organisation ont été appliquées sur le diagramme pour qu'il soit plus lisible et compréhensible. Les résultats obtenus sont présentés dans la figure 15 et la figure 16.

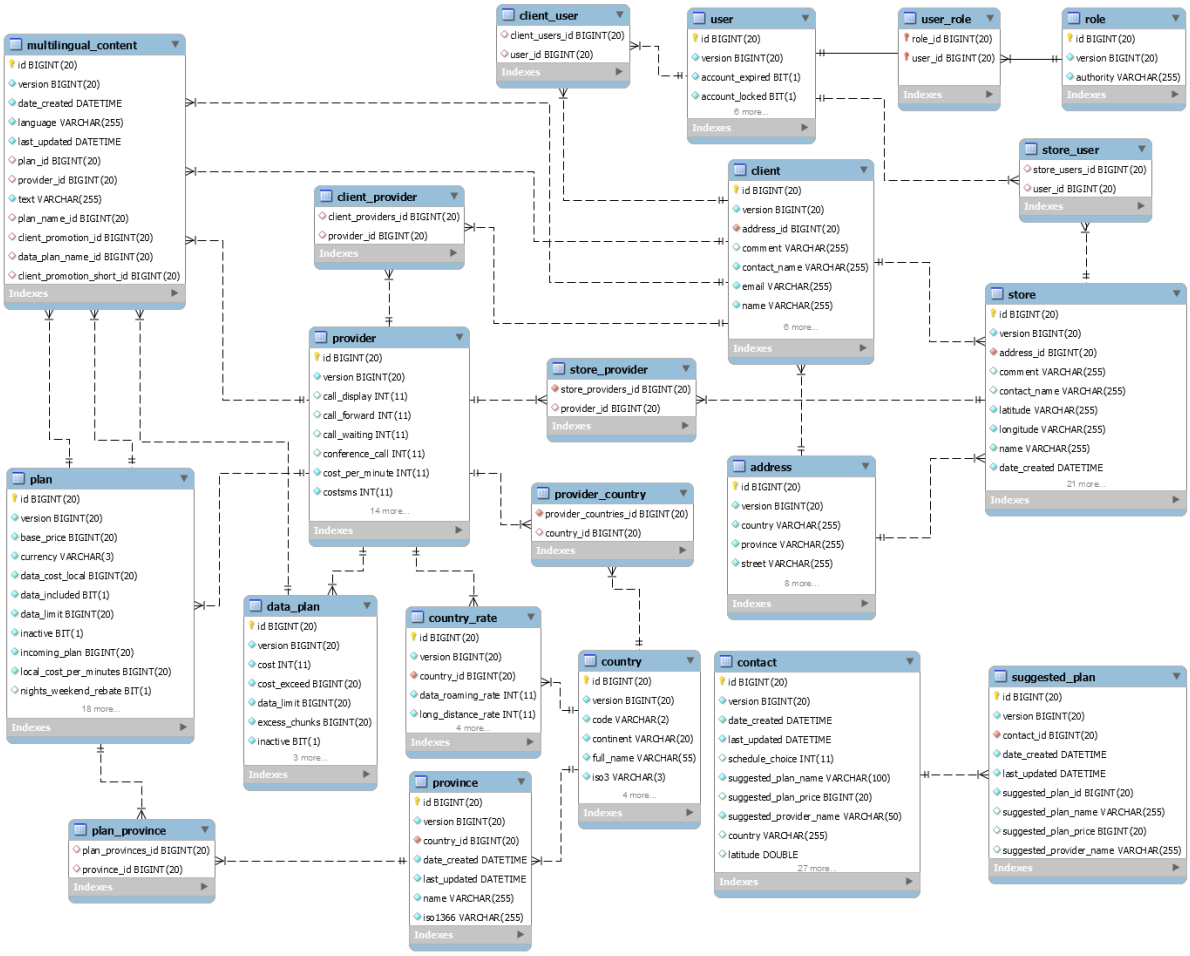


Figure 16 - Partie du diagramme de la base de données Snoobe Server

Tel que présenté à la figure 16, les entités de données de l'application se distinguent clairement à l'exemple de : «Client, Plan, Provider, DataPlan, Device …». Aussi les relations découvertes par l'outil décrivent la navigation entre elles pour mieux comprendre la logique entre les données.

Du côté Serveur, l'analyse a été faite dans le but de localiser et comprendre le «package» et les classes responsables des services web exposés pour l'application mobile.

2.4 Conclusion du chapitre 2

Ce deuxième chapitre a été consacré à la description des étapes préliminaires de réalisation du nouveau prototype Snoobe. La première section a été consacrée à la description et la justification du choix de la technologie. Par la suite, la revue du prototype existant a été faite dans le but d'extraire le maximum de connaissances utiles pour la réalisation du nouveau prototype multiplateforme. Le chapitre suivant sera dédié à la description des étapes de réalisation du nouveau prototype multiplateforme à l'aide du cadriciel Xamarin.

CHAPITRE 3
DÉVELOPPEMENT DE L'APPLICATION MOBILE MULTIPLATEFORME

3.1 Introduction

Ce chapitre décrit les étapes suivies lors du développement du nouveau prototype multiplateforme de Snoobe. La première section consiste à planifier la faisabilité de la réalisation des fonctionnalités qui ont été extraites du premier prototype. Ensuite, les modifications et améliorations apportées sont présentées, suivies de la nouvelle conception de l'application multiplateforme. Par la suite, les exigences fonctionnelles et non fonctionnelles de l'application sont énoncées. En se basant sur cet ensemble d'exigences, l'architecture mise en place est présentée suivie par la conception du prototype logiciel. La dernière partie de ce chapitre présente une synthèse des aspects les plus importants de l'implémentation réalisée du nouveau prototype ainsi que les tests effectués.

3.2 Études des fonctionnalités

Cette section est consacrée à l'étude des fonctionnalités reliées au concept de Snoobe. La différence entre les plateformes mobiles visées représente un grand enjeu de ce projet de migration. Cette différence réside dans l'ensemble des APIs publics de la plateforme qui permet d'obtenir des informations d'utilisation du forfait mobile (c.-à-d. nombre d'appels, la durée des appels, le nombre de SMS), aussi d'autres informations, par exemple le numéro de téléphone et l'adresse email utilisée pour l'enregistrement dans le service Snoobe. Cet inconvénient empêche l'implémentation des mécanismes existants, du prototype existant, qui permettent l'obtention de ce type d'informations. De ce fait, le nouveau prototype nécessitera plus d'implication, du côté utilisateur, pour qu'il paramètre lui-même son choix par rapport à sa prévision d'utilisation (c.-à-d. pour refléter lui-même son utilisation actuelle plutôt que de l'obtenir automatiquement).

3.3 Améliorations apportées

3.3.1 Interface graphique

Une amélioration de l'interface graphique principale, consiste à introduire le patron de conception graphique « maître-détails ». Ce parton repose sur deux parties : la première partie « maître » est représentée par un menu déroulant qui est présenté quand l'utilisateur actionne l'icône située dans le coin haut-gauche de l'interface utilisateur. Ce menu contient des titres qui représentent les fonctionnalités principales de l'application. La partie qui se trouve à gauche, c'est-à-dire « détails » présente les détails des titres du menu principal, représentés par un ensemble de pages.

3.3.2 Nouvelles fonctionnalités

Le prototype actuel, n'offre pas la possibilité aux utilisateurs de marquer des forfaits qui leurs semblent intéressants et qu'ils veulent garder (c.-à-d. dans une liste de favoris pour une consultation ultérieure). En effet, cette fonctionnalité a été jugée intéressante pour assister les utilisateurs dans leur processus du choix, car les utilisateurs préfèrent toujours confirmer le choix avant de prendre une décision finale. Cette fonctionnalité sera ajoutée dans le nouveau prototype multiplateforme.

Du côté markéting, malgré que l'application mobile Snoobe fait partie du type d'application M-Markéting, la seule action markéting actuelle se fait au moment de présenter le résultat de la recherche. Une nouvelle fonctionnalité markéting sera ajoutée, au prototype multiplateforme, afin de permettre à l'utilisateur de visionner des forfaits mobiles promotionnels.

Une amélioration a aussi été conçue et ajoutée à une des fonctionnalités existantes de l'application. Il s'agit d'un ajout au mécanisme de la recherche des forfaits mobiles. Le prototype existant ne permet pas à l'utilisateur de faire une recherche tout en choisissant un paramètre autre que les données du forfait. Le nouveau prototype permettra à l'utilisateur de

choisir d'autres paramètres du forfait (c.-à-d. minutes, SMS, inclusion d'autres options du forfait : afficheur, boite vocale, doubles appels et appel en conférence) qui ne correspond pas forcément à son utilisation actuelle. Avec cet ajout, l'utilisateur pourra exploiter d'autres forfaits et non seulement ceux qui correspondent à son utilisation.

3.4 Exigences

Cette partie du rapport est consacrée à décrire les exigences fonctionnelles et non fonctionnelles du nouveau prototype multiplateforme.

3.4.1 Exigences fonctionnelles

Les exigences fonctionnelles de l'application mobile Snoobe sont :

- Le nouveau prototype Snoobe doit permettre à l'utilisateur de faire une recherche du forfait tout en choisissant ses paramètres. Il doit aussi pouvoir assister l'utilisateur à définir son besoin de données mobiles avec un ensemble de questions;
- Le nouveau prototype Snoobe doit permettre à l'utilisateur de faire une recherche du forfait tout en choisissant ses paramètres. De plus, avec une sélection précise de la quantité de données mobiles voulues;
- Le nouveau prototype Snoobe doit permettre à l'utilisateur de faire une recherche du forfait sans données mobiles;
- Le nouveau prototype Snoobe doit permettre à l'utilisateur de marquer des forfaits et pouvoir les consulter plus tard sans effectuer de nouvelles recherches.
- Le nouveau prototype Snoobe doit permettre à l'utilisateur de visionner les forfaits qui sont en promotions dans sa région.

3.4.2 Exigences non fonctionnelles

Dans cette partie, les exigences qui ont une relation avec le comportement et les caractéristiques non fonctionnelles de l'application sont présentées :

- **Maintenabilité :** Cette exigence représente l'un des axes principaux de ce projet de migration, le nouveau prototype Snoobe doit avoir un maximum de code partagé afin de faciliter sa maintenabilité;
- **Convivialité :** La facilité d'utilisation de l'application ainsi que l'expérience utilisateur, sont deux exigences qui ont été prises en considération lors de la conception graphique de nouveau prototype. L'introduction du patron graphique Maître-détails représente l'action entreprise durant le projet pour augmenter la satisfaction de cette exigence;
- **Performance :** La performance représente une exigence importante que l'application mobile doit satisfaire. Un manque de performance peut nuire l'utilisateur et affecter la réputation de l'application mobile. Lors du choix du cadriciel de développement multiplateforme mobile, le critère de la performance a été pris en considération et le choix de Xamarin a été fait en se basant sur sa qualité qui offre des performances natives aux applications développées;
- **Compatibilité :** L'application produite doit non seulement être compatible avec les plateformes mobiles Androïde et iOS, mais aussi avec les différentes versions de ces deux plateformes ainsi que les différentes tailles d'écrans des appareils de ces plateformes;

3.5 Architecture et conception

3.5.1 Architecture

Cette section du rapport présente l'architecture et la conception originale du nouveau prototype multiplateforme de Snoobe. L'architecture conçue ici correspond à l'architecture d'une application mobile classique développée à l'aide du cadriciel Xamarin et inclue seulement les deux premières plateformes ciblées : iOS et Android. Il serait possible d'en ajouter d'autres au besoin. L'architecture de l'application se compose de deux parties principales : 1) présentation incluant la couche application; et 2) la partie du code partagé incluant l'interface utilisateur avec Xamarin.

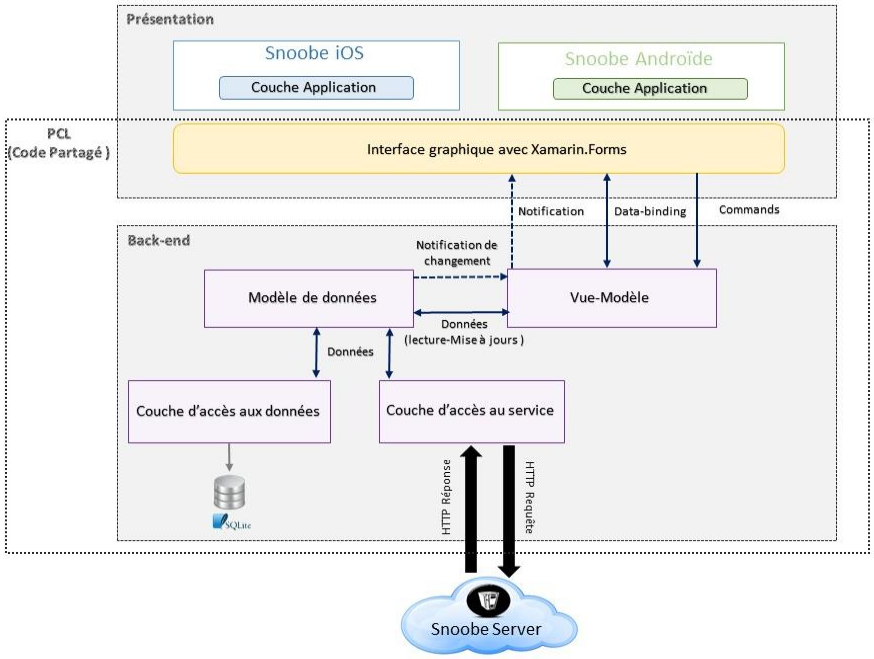


Figure 17 - L'architecture du nouveau prototype mobile Snoobe

La figure 17 décrit chaque application (c.-à-d. application destinée à la plateforme Androïde ou iOS) qui regroupe l'ensemble de ses spécificités dans la couche application. Le composant Xamarin.Forms, du cadriciel Xamarin, a été choisi pour la réalisation des interfaces graphiques de l'application multiplateforme de Snoobe. Ces interfaces représentent, en elles-mêmes, la composante vue du patron architectural MVVM sur lequel repose l'architecture de l'application. Tous les relieurs de données qui représentent l'état des données dans le modèle sont rassemblés dans la composante modèle vue.

Les données sont situées, soit dans le serveur Snoobe hébergé sur le nuage, ou dans une base de données locale. Les classes qui offrent des APIs d'accès à ces données ont été regroupées dans les deux couches : Couche d'accès aux données et couche d'Accès au service.

3.5.2 Conception

La conception de la nouvelle application multiplateforme est présentée dans cette section à l'aide d'une description détaillée des fonctionnalités et de l'enchaînement de ses interfaces.

Le démarrage de l'application se fait à l'aide d'une interface utilisateur d'accueil qui prend quelques secondes afin de permettre la création de la base de données locale, ses tables et l'insertion des informations de configurations lors de la première exécution. L'état suivant de l'application est de présenter l'interface utilisateur principale de la fonctionnalité « Recherche assistée ». L'utilisateur aura le choix d'utiliser la fonctionnalité de la recherche assistée afin de définir son utilisation du forfait et les données du forfait en répondant à un ensemble de questions, ou alternativement, l'utilisateur peut actionner le bouton du menu principal pour afficher les restes des fonctionnalités offertes par l'application multiplateforme.

La première catégorie de fonctionnalités est celle de la recherche des forfaits mobiles. Ces fonctionnalités partagent toutes la même interface utilisateur qui sert à définir l'utilisation désirée (c.-à-d. minutes d'appels, nombre SMS, l'inclusion d'autre option). La différence principale se situe au niveau de la définition de la quantité des données désirées : la recherche assistée permet définir ce niveau à l'aide d'une suite d'interfaces utilisateurs qui incluent des questions qui permettent d'établir cet estimé. Tandis que la recherche de type « je connais mes données » permet à l'utilisateur de choisir une quantité de données précise parmi une liste proposée. Le dernier type de recherche permet de lancer la recherche sans aucune définition de données. Dans ces trois types de recherche, une fois que toutes les données de recherche sont capturées et que le bouton « lancer recherche » est actionné, une interface utilisateur permettant l'attente est affichée et présente le résumé des données choisies et une icône de progression pour indiquer que la recherche est en cours. Une fois que les résultats sont récupérés à partir du serveur, ils sont démontrés dans une nouvelle interface utilisateur. L'utilisateur, à ce niveau, a le choix de choisir un forfait préféré ou de sélectionner un forfait proposé et basculer à l'interface utilisateur qui en précise les détails.

Les deux autres fonctionnalités restantes sont accessibles à partir du menu principal de l'application et permettent soit d'afficher les forfaits marqués ou montrer les forfaits qui sont en promotion. Le menu principal peut être affiché à tout moment soit en actionnant le bouton correspond ou en faisant un glisse de doigt sur l'interface utilisateur pour basculer instantanément d'une fonctionnalité à une autre. Toutes les interfaces utilisateurs correspondantes à cette conception sont présentées dans l'Annexe II de ce rapport.

3.6 Implémentation

3.6.1 Préparation de l'environnement de développement

Cette section est consacrée à la description du matériel et des logiciels utilisés pour le développement de l'application multiplateforme. Concernant le matériel, un Mac doté d'un système d'exploitation OS X est nécessaire au développement, car c'est la seule plateforme qui permet le développement des applications mobiles iOS, et permet aussi le développement des applications Androïde avec le cadriciel Xamarin. Le tableau suivant présente la configuration matérielle et logicielle du Mac utilisé.

Tableau 1 - Configuration de l'ordinateur de développement

Nom	**Description**
Mac	Mac mini Server Version 2012
Processeur	Intel Core i7 2.6 GHz
Mémoire	16 GB DDR
Système d'exploitation	OS X El Capitan version 10.11.1
iOS IDE	Xcode Version 7.2 avec iOS 9
Plateforme Androïde	Android SDK avec API 22 (v 5.1.1)
Java	Kit de développement Java version 1.7.0_71

Après la préparation du Mac, le cadriciel Xamarin a été installé est configuré. Les versions utilisées sont résumées dans le tableau suivant :

Tableau 2 - Les versions des logiciels de développement utilisés

Nom	**Version**
IDE Xamarin Studio	5.9.7
Xamarin.Forms	1.5.0.6447
Xamarin.iOS	9.0.1.20
Xamarin.android	5.1.6.7
Mono (version libre du cadriciel .NET)	4.0.4

Afin de permettre le contrôle des versions du code source et de créer des sauvegardes continues, un dépôt dans le système « Bitbucket » a été utilisé pour assurer cette fonction. Au niveau du Mac, le logiciel « SourceTree » a été installé et configuré afin de permettre de gérer le dépôt du code source créé à partir de l'ordinateur en local.

3.6.2 Détails d'implémentation

Cette section présente les détails importants pris en considération lors de l'implémentation de l'application. Le langage de programmation utilisé est le C# de Microsoft. Ce langage de programmation a été utilisé pour exploiter les API du cadriciel libre Mono qui est la version libre du cadriciel « .Net » de Microsoft. Un ensemble de « plug-ins » sont nécessaires afin de permettre la maximisation de l'ensemble du code partagé de l'application, ces Plugins sont les suivants :

- **Device Info Plugin :** Ce plug-in a été utilisé pour extraire des informations sur l'appareil mobile, à l'exemple de son Modèle, le Système utilisé et sa version, son identificateur unique;
- **Geolocator Plugin :** Ce dernier a été utilisé pour produire un seul code de gestion des informations de la localisation pour les deux plateformes (c.-à-d. Androïde et iOS);
- **Json.net :** Ce plug-in a été utilisé pour toute la partie services web, la responsabilité de cette partie et d'échanger les informations sous format JSON avec le serveur Snoobe;

Aussi, l'iconographie de l'ancienne application a été réutilisée dans la nouvelle application multiplateforme et les interfaces utilisateurs d'accueil ont été implémentés nativement étant donné que Xamarin n'offre actuellement aucune possibilité de réaliser cette même implémentation en code partagé.

La réalisation des interfaces graphiques a été réalisée en se basant sur un ensemble de pages prédéfinies dans le cadriciel Xamarin. Ces types sont :

- **MasterDetailPage :** Ce type de page est l'implémentation du patron de conception graphique maitre-détails et a été utilisé pour la réalisation de la page principale;
- **CarouselPage :** Ce type permet l'implémentation d'une séquence d'interfaces dans la même page, le passage entre les interfaces se fait soit à l'aide d'un bouton ou avec un glisse du doigt sur l'interface utilisateur. L'implémentation des formulaires a été faite avec ce type;
- **ContentPage :** Ce type de page permet l'implémentation d'une page avec différents types de contenu.

3.6.3 TESTS

Les tests de l'application multiplateforme ont été effectués sur des appareils mobiles pour la plateforme Androïde et sur des logiciels simulateurs pour la plateforme iOS, le tableau suivant résume les appareils et logiciels utilisés :

Tableau 3 - Les appareils mobiles utilisés pour les tests

Nom	**La version du système d'exploitation**
Google Nexus 5	Androïde v 6.0.1
Samsung S3	Androïde v 4.9
iPhone 4S (Simulation)	iOS v 8.0
iPhone 5S (Simulation)	iOS v 9.0
iPhone 6S (Simulation)	iOS v 9.0

La diversité dans les versions des systèmes vise à tester que toutes les fonctionnalités de l'application s'exécutent correctement sur une multitude de versions des systèmes d'exploitation mobiles. Aussi, la diversité des tailles des écrans vise à tester l'affichage des interfaces graphiques.

3.7 Conclusion du chapitre 3

Ce chapitre a été consacré à décrire les étapes les plus importantes de la réalisation de la nouvelle application multiplateforme. Un résumé sur l'étude faite sur les fonctionnalités a été présenté en premier, suivi des améliorations proposées et apportées à la nouvelle application. Les différentes étapes de développement entreprises ont été présentées en dernier lieu ainsi que leurs aspects les plus importants.

4.1 Présentation des résultats

Le résultat principal obtenu dans le cadre de ce projet est une application mobile multiplateforme qui couvre les fonctionnalités du prototype existant qui ne fonctionnait que sur Androïde. Cette nouvelle preuve de concept offre également de nouvelles fonctionnalités qui ont été validées avec Snoobe.

L'interface graphique, de son côté, a aussi subi des améliorations afin d'améliorer l'expérience utilisateur. Ces améliorations ont rendu l'application plus intuitive, plus simple à utiliser et les fonctionnalités sont plus accessibles. Les deux versions (c.-à-d. Androïde et iOS) sur lesquels s'est basée cette étude ont été testées et sont fonctionnelles.

4.2 Récapitulatifs

Un récapitulatif de ce projet de migration est présenté dans cette présente section. Les avantages et les inconvénients d'utiliser un cadriciel multiplateforme sont discutés en premier, suivi d'un résumé des efforts déployé pour effectuer cette conversion.

4.2.1 Avantages apportés par le cadriciel multiplateforme

Les avantages d'utilisation d'un cadriciel multiplateforme sont multiples pour Snoobe. Le premier avantage porte sur l'aspect maintenabilité de l'application. En effet, une nouvelle application compilable et exécutable sur les deux plateformes, 1) iOS 2) Androïde, a été générée avec un seul code source et en se basant sur un seul langage de programmation (c.-à-d. C#). Il y a maintenant une seule technologie de développement (c.-à-d. Mono, la version libre du cadriciel .Net), ce qui implique que les coûts et les efforts seront réduits significativement lors de l'exploitation et des modifications futures de l'application.

De plus, l'application a bénéficié de l'amélioration de son expérience utilisateur par l'uniformité de son interface graphique sur les deux plateformes grâce à la richesse des

composants graphiques du cadriciel Xamarin.Forms. En s'appuyant sur le cadriciel Xamarin, qui va être acquis par le géant de l'industrie logiciel Microsoft au mois d'avril 2016, la nouvelle application Snoobe multiplateforme aura la chance d'utiliser une technologie appuyée par l'industrie et pourra bénéficier de toutes les améliorations futures à Xamarin. Finalement, Snoobe à la possibilité d'utiliser d'autres plateformes mobiles et de bénéficier des extras proposés par Xamarin comme, par exemple, leur plateforme de test contenant des milliers d'appareils virtuels, ainsi que leur système de surveillance « Xamarin Insight ».

4.2.1 Inconvénient du cadriciel multiplateforme

L'inconvénient majeur que l'application Androïde actuelle de Snoobe a subi ne vient pas du cadriciel Xamarin lui-même, mais du fait que les deux autres plateformes mobiles visées ne proposent aucune API d'accès à l'historique d'utilisation du téléphone. De ce fait, toutes les fonctionnalités de récupérations des données d'utilisation du forfait mobile ne sont pas implémentables. Ceci représente un inconvénient et un avantage à la fois, car il a mené à repenser cette fonctionnalité et en proposer une nouvelle qui donne à l'utilisateur la chance de mieux paramétrer ses recherches pour qu'elles ne soient pas juste limitées aux recherches correspondantes seulement à son utilisation.

Un second inconvénient vise le code partagé de la nouvelle application multiplateforme et plus précisément le code partagé de l'interface graphique. Xamarin se base sur un ensemble de fonctionnalités de correspondances (le terme utilisé dans ce cadriciel est : « Renderers »). Les fonctionnalités de « renderers » assurent la correspondance entre les éléments graphiques de Xamarin et leurs correspondants dans les deux plateformes mobiles visées (c.-à-d. Androïde et iOS). Ces références ne correspondent pas toujours à ce que le développeur veut implémenter, d'où la nécessité d'implémenter des méthodes natives personnalisées « Renderers » qui nécessitent du surcharger celles qui existent par défaut pour obtenir le résultat voulu.

Ce projet de migration d'une application Androïde native, vers une application multiplateforme en s'appuyant sur le cadriciel Xamarin démontre finalement que les désavantages sont plus faibles que les avantages.

4.3 L'effort déployé dans le projet

Ce projet de migration est composé de deux principaux sous-projets. Le premier consiste à effectuer la rétro-ingénierie du système existant tandis que le deuxième consiste à redévelopper une nouvelle application multiplateforme. Le premier sous-projet exige des compétences du domaine de la rétro-ingénierie, à savoir la recherche des indices qui conduisent à la reconstruction du système et à sa compréhension, l'établissement du modèle théorique et de son architecture, l'extraction des technologies et services du système existant ainsi que la récupération des concepts qui se cachent derrière un tel système.

Le deuxième sous-projet exige en premier un esprit analytique afin de pouvoir analyser, comparer et choisir le meilleur cadriciel de développement multiplateforme qui correspond aux besoins de Snoobe. Ensuite, les connaissances acquises lors de la maîtrise, d'un ingénieur logiciel, sont requises afin de définir les exigences, concevoir une architecture suivie d'une conception et, procéder à l'implémentation et aux tests par la suite.

Comme il est important de définir les tâches d'un projet de génie logiciel, il est aussi important d'être capable de planifier et mesurer l'effort consacré à chaque tâche. Cette mesure sert de plan et de bilan pour ce présent projet et permet de communiquer ses résultats à tire de point de repère pour des projets similaires.

Le tableau suivant (Tableau 4) présente une synthèse des efforts consacrés à chaque tâche. Ces efforts sont présentés en Personne/Mois (P/M), ce qui signifie les heures consacrées par une personne, moi-même, durant un mois.

Tableau 4 - Tableau récapitulatif des efforts du projet de migration de Snoobe

Tableau récapitulatif de l'effort consacré au projet Snoobe			
Sous-Projet	**Mois(Année)**	**Tâches**	**Effort (P/M)**
Rétro-ingénierie	Mai 2015	Collecter les différentes parties du système et les entrevues avec les anciens membres de l'équipe de développement	120
Rétro-ingénierie	Juin 2015	Définition du modèle théorique de la rétro-ingénierie convenable au modèle de Snoobe	40
Rétro-ingénierie	Juin 2015	Analyse des parties recueillies pour définir les technologies de développement utilisées et leurs versions.	45
	Juillet 2015		38
Rétro-ingénierie	Aout 2015	Compilation et l'exécution de tout le système en local	35
Rétro-ingénierie	Aout 2015	Analyse statique et dynamiques du système Snoobe (client mobile, serveur, base de données).	105
	Septembre 2015		35
		Effort total de l'activité de la rétro-ingénierie	**418**
Développement	Septembre 2015	Recherche du cadriciel de développement multiplateforme le plus convenable aux cas d'étude Snoobe.	40
Développement	Septembre 2015	Mise en place de l'environnement de développement (Installation d'OS X, Xamarin Studio, création et configuration du compte sur Bitbucket, incluant l'essai échoué d'installation d'OS X sur une machine virtuelle dans l'environnement Windows).	30
Développement	Septembre 2015	Repenser l'application et définir les fonctionnalités à réaliser (définition des exigences à satisfaire)	28
Développement	Octobre 2015	Mise en place de l'architecture et de la conception de l'application.	84
Développement	Octobre 2015	Apprentissage de développement avec le cadriciel Xamarin et l'implémentation de la nouvelle application.	35
	Novembre 2015		130
	Décembre 2015		85
Développement	Janvier 2016	Développement de l'application (Suite)	25
		Recherche bibliographique pour la rédaction du rapport.	32
		Rédaction et révision de l'état du chapitre 1 « état de l'art » du rapport.	65
Développement	Février 2016	Rédaction et révision du chapitre 2 « rétro-ingénierie» et chapitre 3« développement de l'application mobile».	125
Développement	Mars 2016	Rédaction et révision du chapitre 4 « récapitulatif du projet ».	93

		Rédaction et révision finales du rapport.	
		Effort total de l'activité de développement	**772**
		Effort total du projet de migration	**1190**

Ce tableau démontre que 35,12% de l'effort a été consacré à des activités de rétro-ingénierie du système existant. Cet effort inclut la collecte des composants du système, leur exécution ainsi que leur analyse. Le pourcentage de l'effort restant qui représente 64.88% de l'effort total du projet a été consacré à l'activité de développement de la nouvelle application multiplateforme. Cela inclut le choix du cadriciel, l'étude des fonctionnalités, les différentes étapes de développement et la rédaction du rapport. Les tâches qui ont eu le plus haut pourcentage de l'effort dans cet ensemble sont l'analyse du système existant, l'implémentation de la nouvelle application et la rédaction du rapport telles qu'elles ont pris respectivement 11.76%, 21%, 26.45% de l'effort total consacré au projet.

4.4 Conclusion du chapitre 4

Un récapitulatif du projet de migration de Snoobe a été présenté. Les résultats obtenus sont décrits en premier lieu, suivi des avantages et des inconvénients de l'utilisation d'un cadriciel multiplateforme pour le cas d'étude : Application mobile Snoobe. Une analyse des efforts fournis tout au long du projet a été finalement présentée.

CONCLUSION

Ce projet de recherche appliquée consiste à investiguer la possibilité d'utiliser un cadriciel multiplateforme pour la migration de l'application Snoobe actuelle sur plateforme Androïde vers une application multiplateforme qui soit exécutable sur les deux plateformes : 1) Androïde et 2) iOS. La première étape du projet consiste à faire la rétro-ingénierie de l'application existante afin d'extraire le maximum d'indices et d'informations pertinentes qui permettent la conversion. Pour réaliser cette étape, un modèle théorique a été utilisé pour guider le processus de rétro-ingénierie. Les modèles identifiés dans la littérature n'offrent pas d'indications pratiques, un nouveau modèle de rétro-ingénierie a été proposé. En se basant sur ce modèle, la rétro-ingénierie a été effectuée ce qui a permis d'obtenir un ensemble de connaissances sur le logiciel existant.

La deuxième étape du projet consistait à choisir un cadriciel de développement multiplateforme le plus adapté pour la conversion de Snoobe. Une étude comparative a identifié Xamarin comme le plus avantageux. Une fois ce choix effectué, la conception de la nouvelle application multiplateforme a été réalisée, incluant des améliorations et des nouvelles fonctionnalités. La préparation de l'environnement de développement a été réalisée en parallèle avec la conception du nouveau logiciel. Ceci inclut l'obtention des licences pour utiliser Xamarin, et la mise en place de l'environnement de développement. Le nouveau prototype multiplateforme de l'application a été réalisé, donc un seul code source et deux applications exécutables sur les deux plateformes mobiles : 1) Androïde 2) iOS. Ce projet de recherche appliquée a comporté plusieurs défis. La première difficulté consistait à faire une étude pour rassembler toutes les parties du système existant de Snoobe, comprendre les technologies existantes et le faire fonctionner en mode local. La seconde difficulté consistait à faire le choix du cadriciel multiplateforme. La troisième difficulté provient du fait que toutes les technologies utilisées durant les étapes de rétro-ingénierie et développement étaient difficiles à utiliser. Beaucoup d'effort a été consacré à apprendre le cadriciel Grails du côté serveur, Gradle et Android du côté application mobile existante. Il a été aussi nécessaire de maîtriser le langage de programmation C# pour développer avec le cadriciel Xamarin ainsi que d'apprendre iOS et toutes les « api » de la plateforme Xamarin.

Ce projet a nécessité l'apprentissage de nouvelles compétences. Du point de vue technique, il m'a permis d'utiliser des compétences en rétro-ingénierie acquises durant ma formation à la maîtrise à l'ETS. De plus, des compétences en développement mobile ont été acquises sur les deux plateformes Androïde et iOS, ainsi qu'avec le cadriciel Xamarin.

ANNEXE I

CAPTURES D'ÉCRAN DU PROTOTYPE SNOOBE

Figure 18 – Interface utilisateur d'accueil

Figure 19 - Notification du choix du pays quand le fournisseur mobile n'est pas reconnu

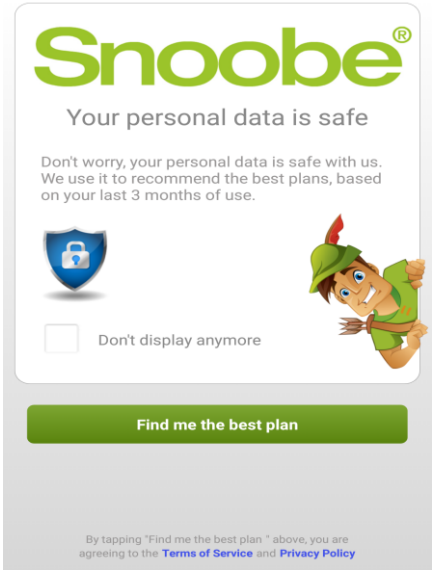


Figure 20 - Interface utilisateur de contrat du Snoobe

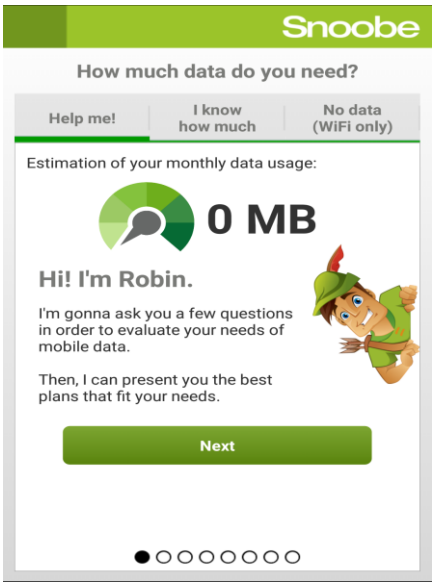


Figure 21 - Première interface utilisateur de la recherche assistée : Accueil

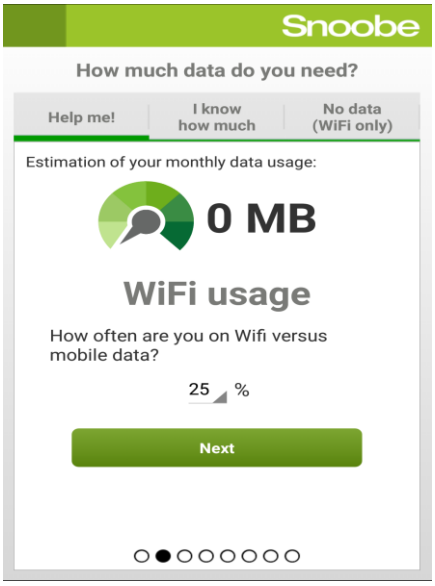


Figure 22 - Deuxième interface utilisateur de la recherche assistée : définition assistée d'usage de données

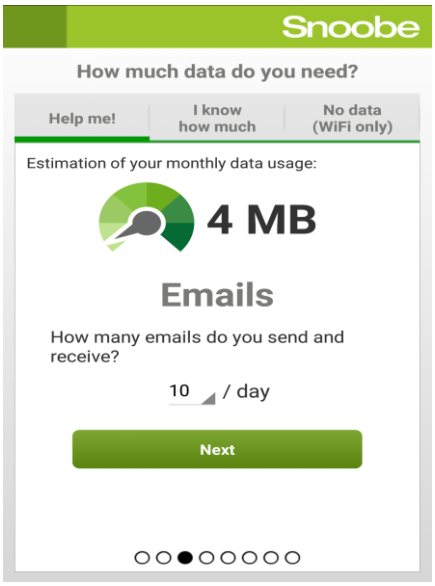


Figure 23 - Troisième interface utilisateur de la recherche autorisée : Choix de nombre de courriels

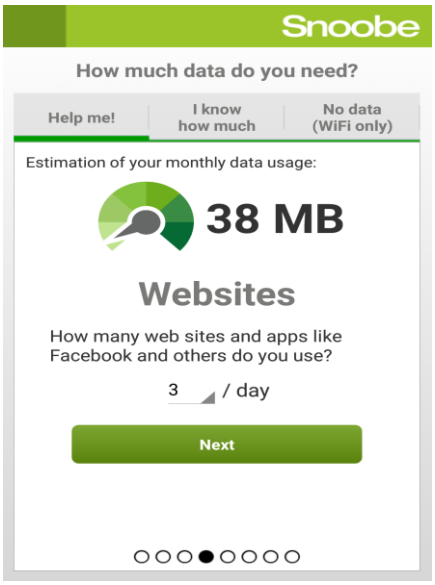


Figure 24 - Quatrième interface utilisateur de la recherche assistée : Choix de nombre de sites web

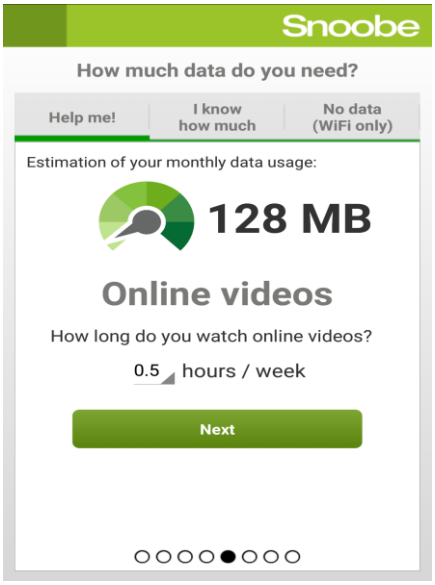


Figure 25 - Cinquième interface utilisateur de la recherche assistée : Choix de la durée des vidéos YouTube

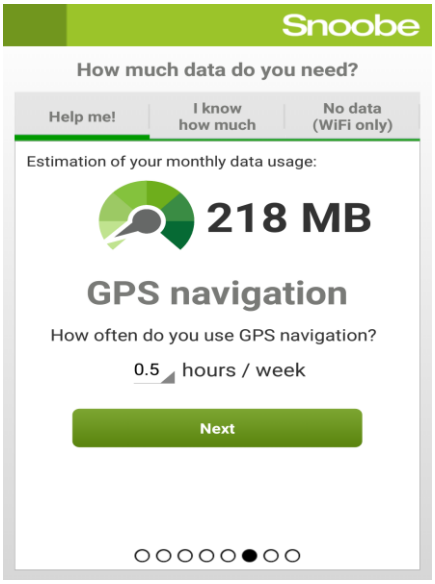


Figure 26 - Sixième interface utilisateur de la recherche assistée : Choix du nombre d'heures GPS

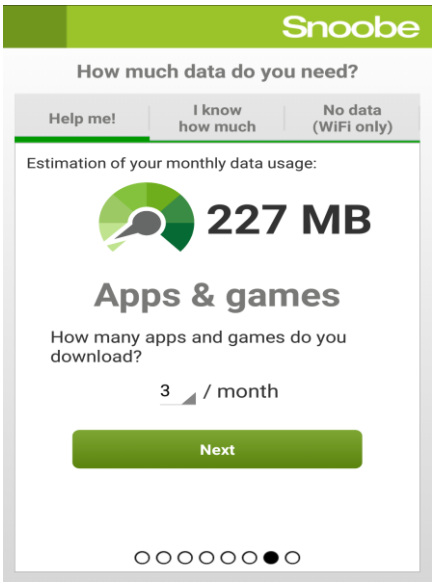


Figure 27 - Septième interface utilisateur de la recherche assistée : Choix du nombre des Apps et jeux vidéos à télécharger

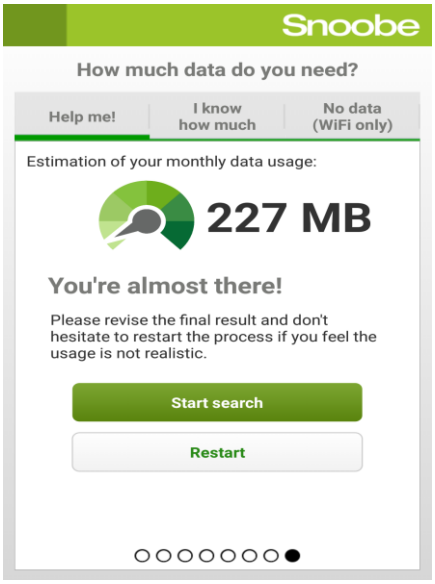


Figure 28 - Huitième interface utilisateur de la recherche assistée : interface utilisateur de validation et lancement de la recherche

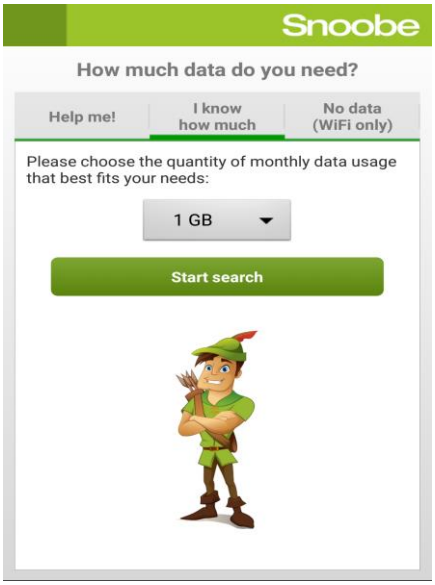


Figure 29 - Interface utilisateur de la recherche avec connaissance de la quantité de données

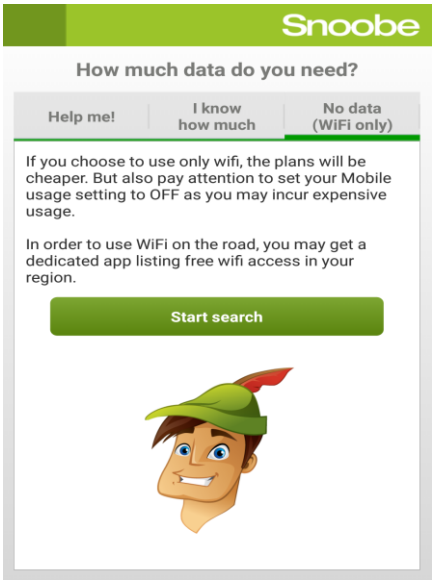


Figure 30 - Interface utilisateur de la recherche des forfaits sans données

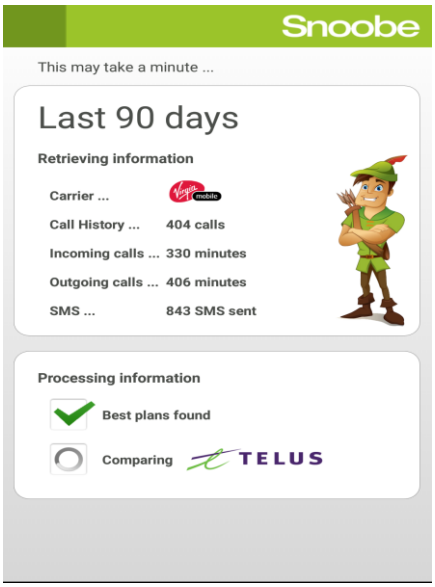


Figure 31 - Interface utilisateur d'attente lors de la recherche des forfaits

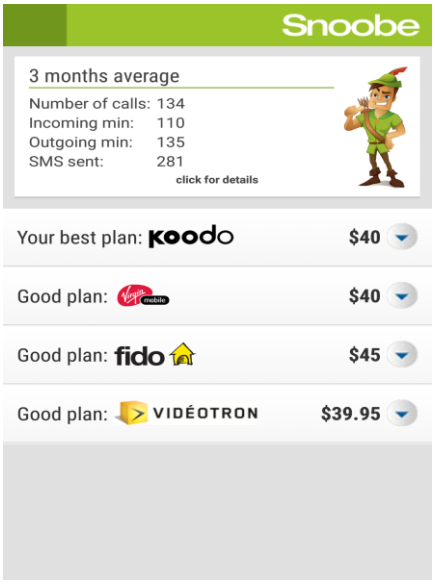


Figure 32 - Interface utilisateur des plans proposés

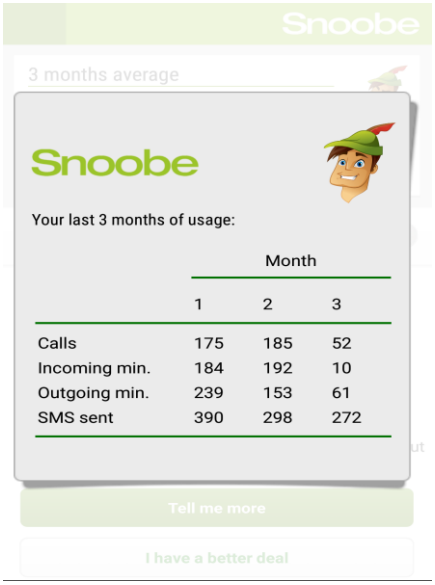


Figure 33 - Interface utilisateur de détails d'utilisation du forfait des trois derniers mois

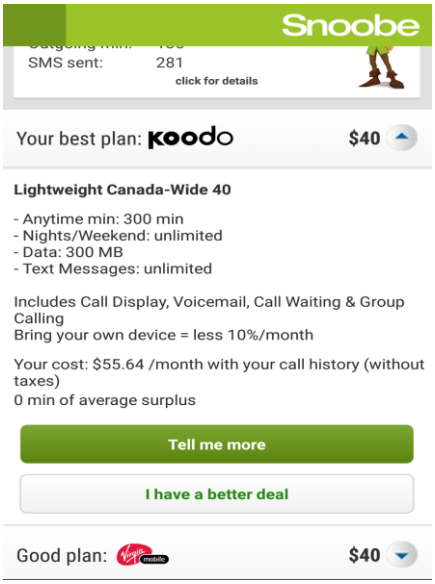


Figure 34 - Interface utilisateur de détails de l'un des plans proposés

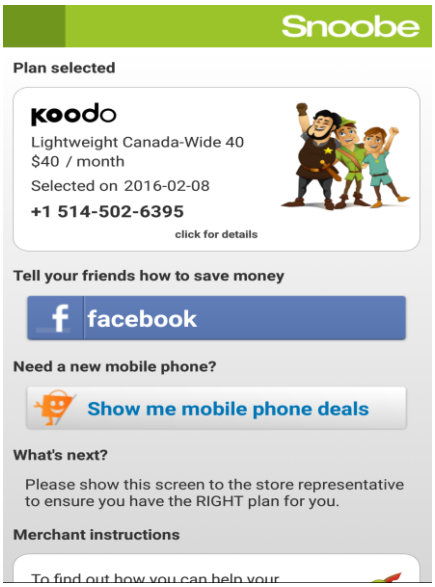


Figure 35 - Interface utilisateur de validation du plan choisi

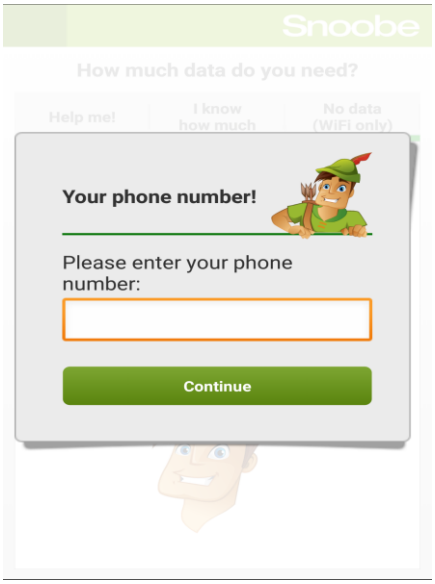


Figure 36 - Interface utilisateur de demande de numéro de téléphone avant de démarrer la recherche quand le numéro n'est pas connu

ANNEXE II

CAPTURES D'ÉCRAN DE L'APPLICATION MULTIPLATEFORME

L'interface utilisateur d'accueil est présentée en premier, elle est composée de plusieurs parties qui ont nécessité beaucoup d'efforts pour qu'elles gardent leur homogénéité.

Figure 37 – Interface utilisateur d'accueil de Snoobe multiplateforme

Le menu principal qui résume les fonctionnalités offertes par l'application Snoobe multiplateforme est présenté

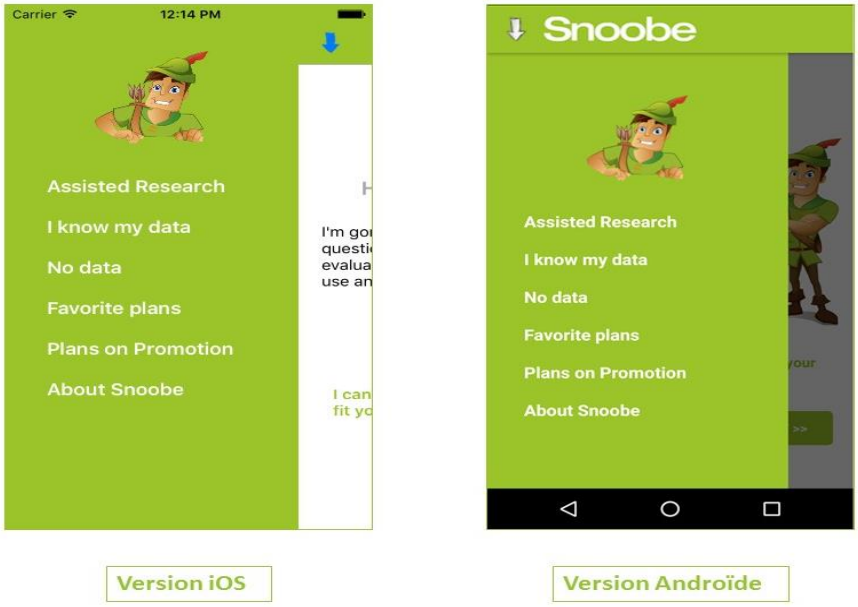


Figure 38 - Menu principal de Snoobe multiplateforme

Les interfaces utilisateurs du premier type de recherche (c.-à-d. la recherche assistée) sont montrées en premier, elles sont regroupées dans deux parties, la première partie regroupe l'interface utilisateur d'accueil de la recherche ainsi que l'interface utilisateur de personnalisation du forfait mobile souhaité.

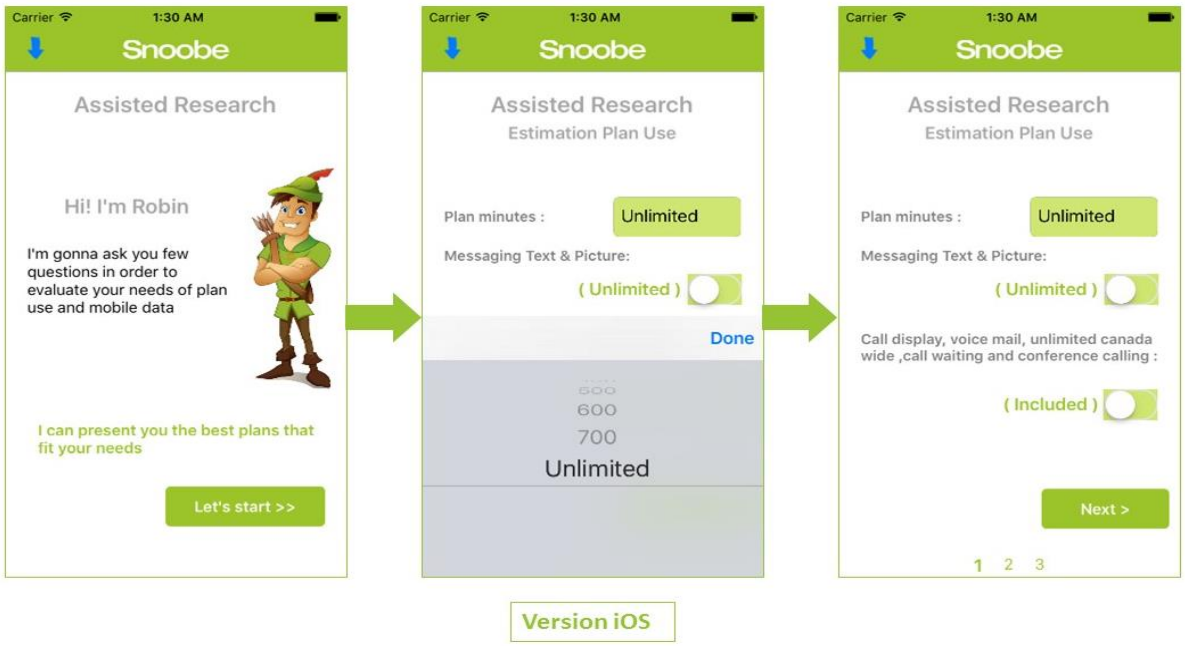


Figure 39 – Trois premières interfaces utilisateurs de la recherche assistée version iOS

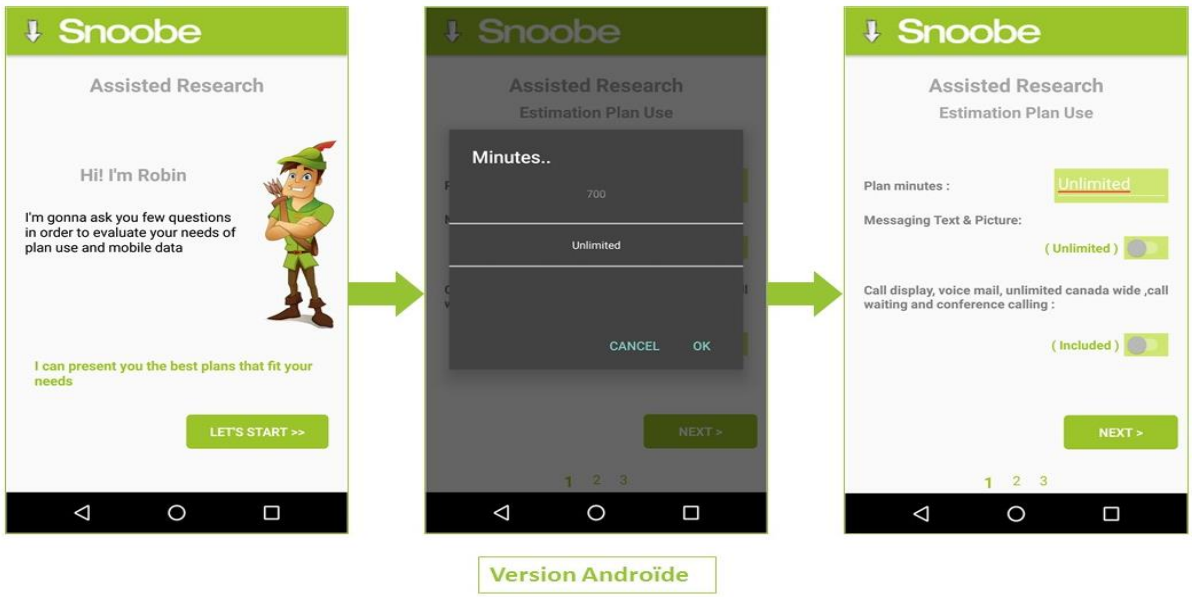


Figure 40 - Trois premières interfaces utilisateurs de la recherche assistée version Androïde

Tandis que la deuxième partie regroupe les interfaces utilisateurs qui permettent à l'utilisateur de définir son besoin de données mobile. Cette partie inclut l'interface utilisateur de validation de la recherche qui permet à l'utilisateur soit de lancer la recherche ou de recommencer la personnalisation de sa recherche.

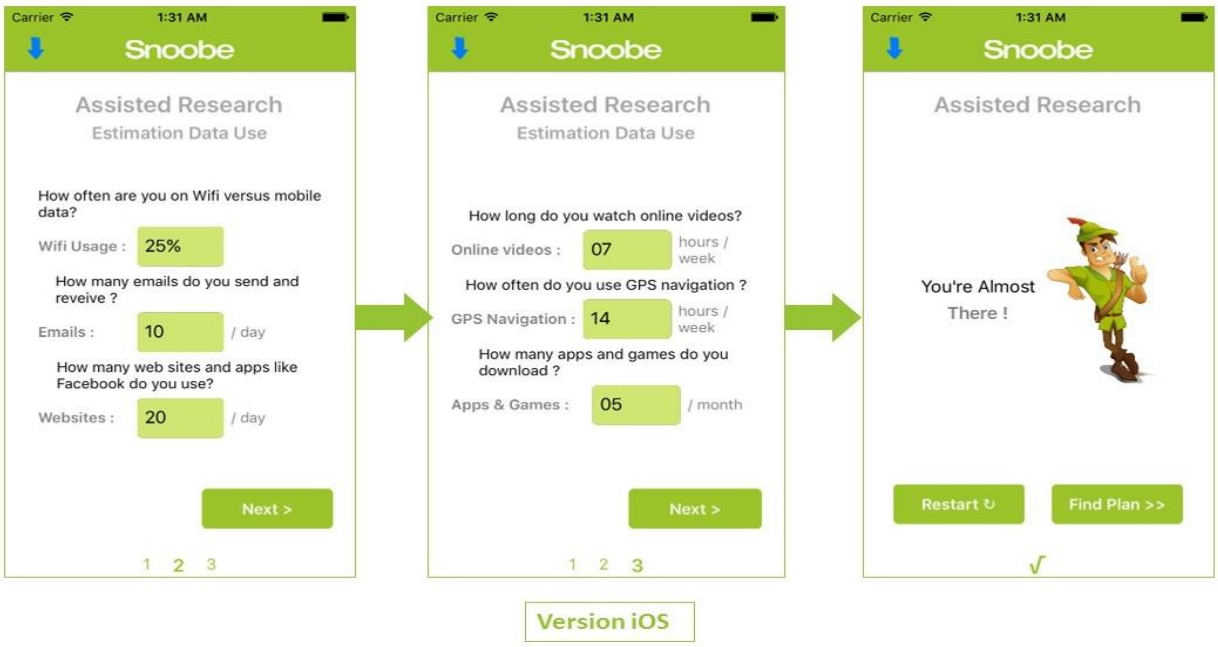


Figure 41 - Trois dernières interfaces utilisateurs de la recherche assistée version iOS

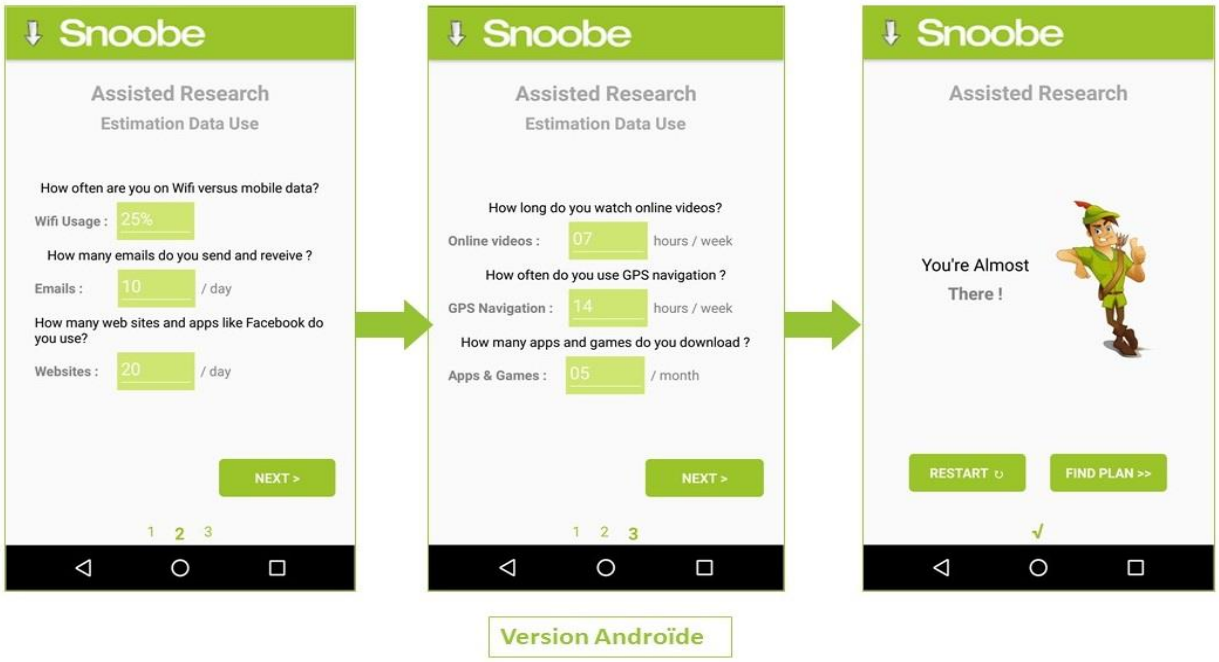


Figure 42 - Trois dernières interfaces utilisateurs de la recherche assistée version Androïde

Les interfaces utilisateurs de la recherche de type « Je connais mon utilisation » sont montrées dans ce qui suit :

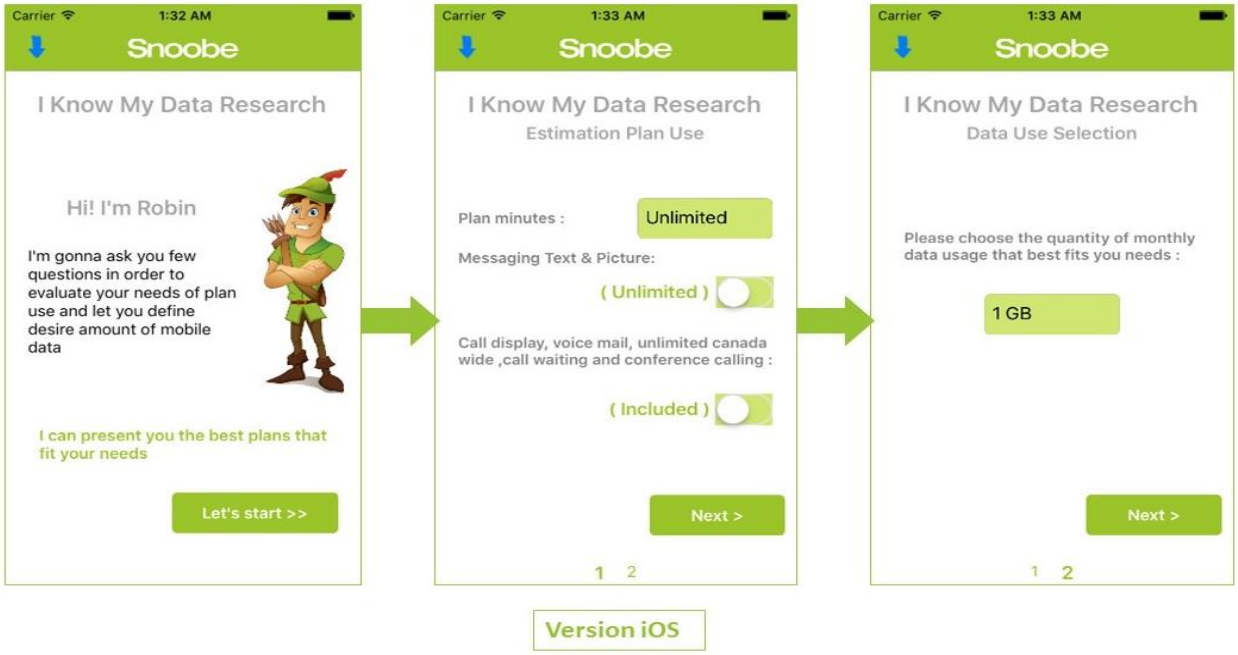


Figure 43 - Trois premières interfaces utilisateurs de la recherche je connais mes données version iOS

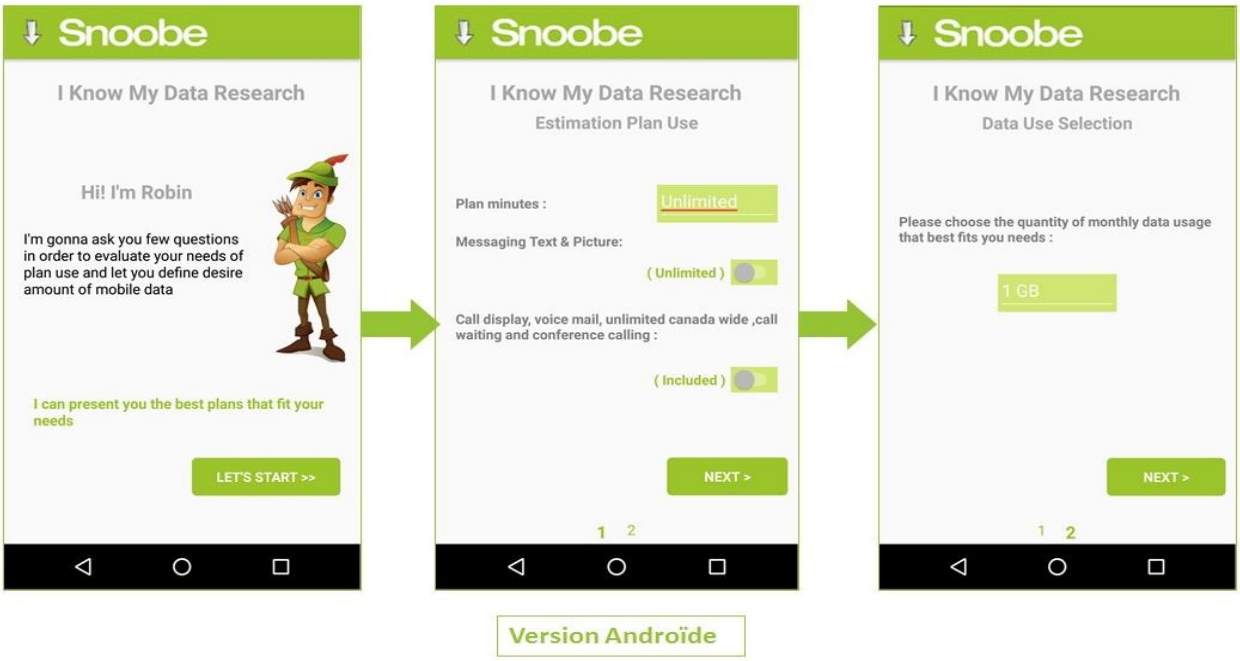


Figure 44 - Trois premières interfaces utilisateurs de la recherche je connais mes données version Androïde

Le type de recherche "je connais mon data" présente aussi des interfaces utilisateurs de validation une fois que la personnalisation est terminée, cette interface utilisateur permet soit de lancer la recherche ou de relancer la personnalisation de cette recherche. La figure suivante présente cette interface utilisateur :

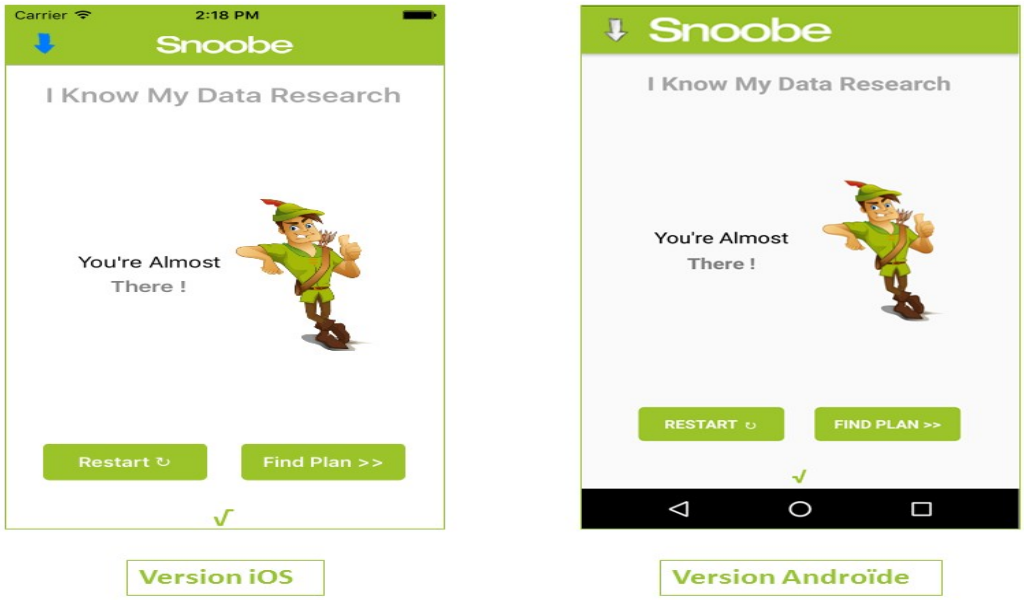


Figure 45 - L'interface utilisateur de validation de la recherche version Androïde et iOS

Les interfaces utilisateurs suivantes montrent le type de recherche sans données, ce type de recherche montre seulement l'interface utilisateur de personnalisation du forfait pour passer directement à l'interface utilisateur de validation par la suite.

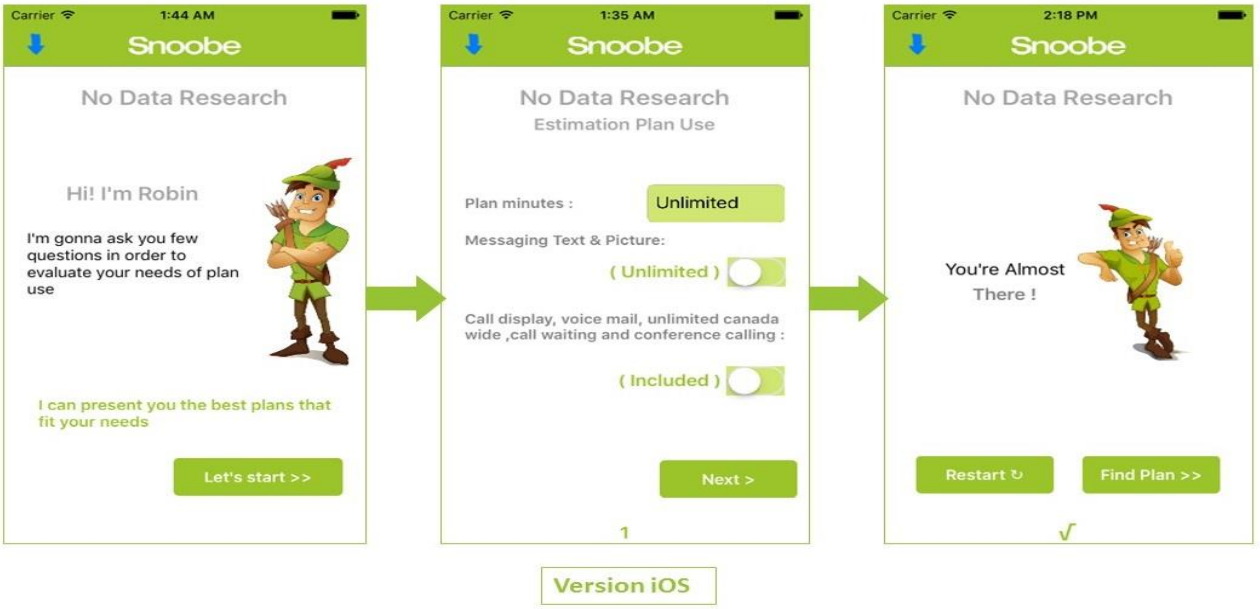


Figure 46 - Les interfaces utilisateurs de la recherche des forfaits sans données version iOS

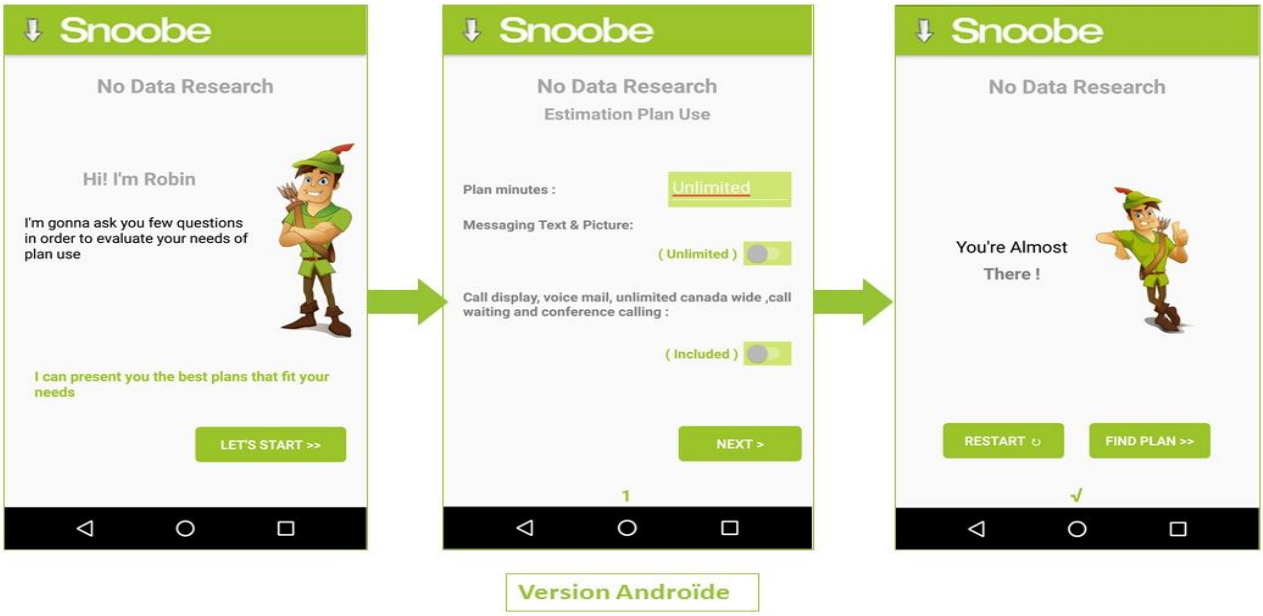


Figure 47 - Les interfaces utilisateurs de la recherche des forfaits sans données version Androïde

Une fois que la recherche est lancée, une interface utilisateur qui affiche un résumé de la configuration choisie est montrée et l'état des deux processus d'envoi et de comparaison des forfaits. La figure suivante montre cette interface utilisateur :

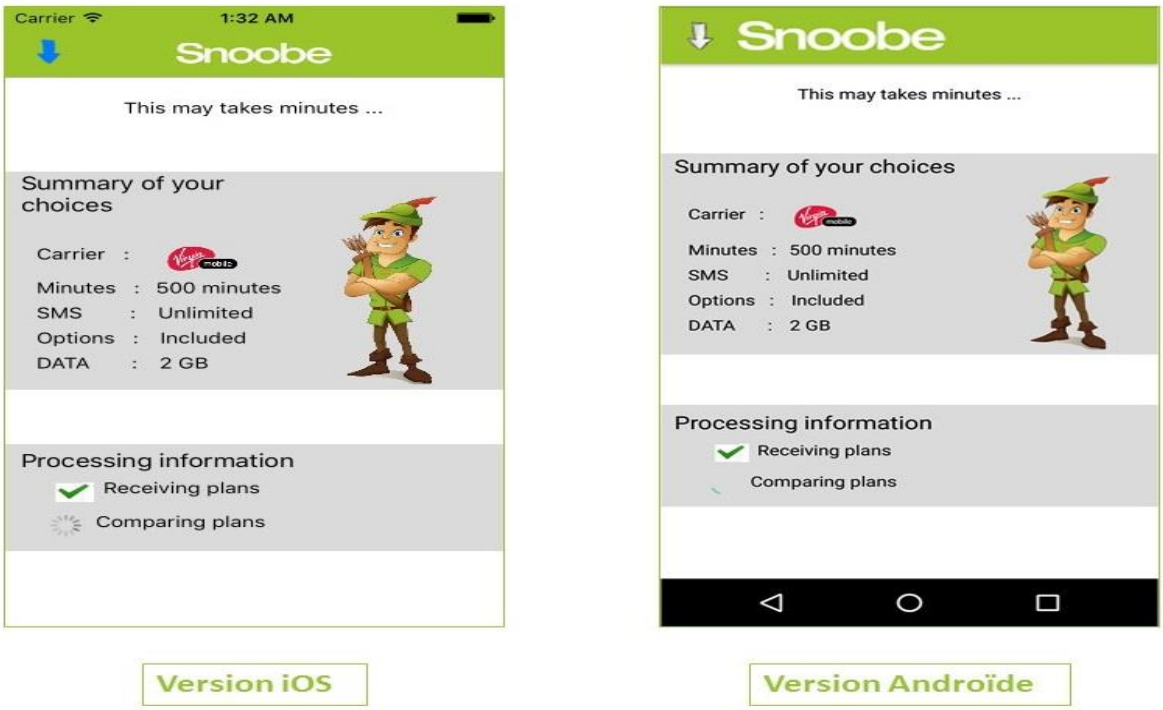


Figure 48 - Interface utilisateur de la recherche des forfaits pour les deux versions : iOS et Androïde

Une fois que le résultat est prêt, une interface utilisateur résultat affiche les 4 meilleurs forfaits qui correspondent à la configuration choisie. Les quatre forfaits sont montrés sous forme d'une liste ou chaque forfait peut être étendu en cliquant sur lui-même. Le menu étendu d'un forfait affiche sa description détaillée ainsi que la promotion (si le forfait est en promotion) affichée en caractère gras. Ce menu donne la possibilité de marquer un forfait comme favoris en cliquant sur l'étoile se trouvant dans le coin haut droit du menu, cette étoile change sa couleur vers le doré pour confirmer que le forfait a été marqué. Ce menu donne également la possibilité de choisir un forfait ou de mentionner que l'utilisateur à une meilleure proposition, ces deux possibilités sont accessibles à partir de deux boutons en bas du menu étendu. Les interfaces utilisateurs du résultat sont montrées dans les figures suivantes :

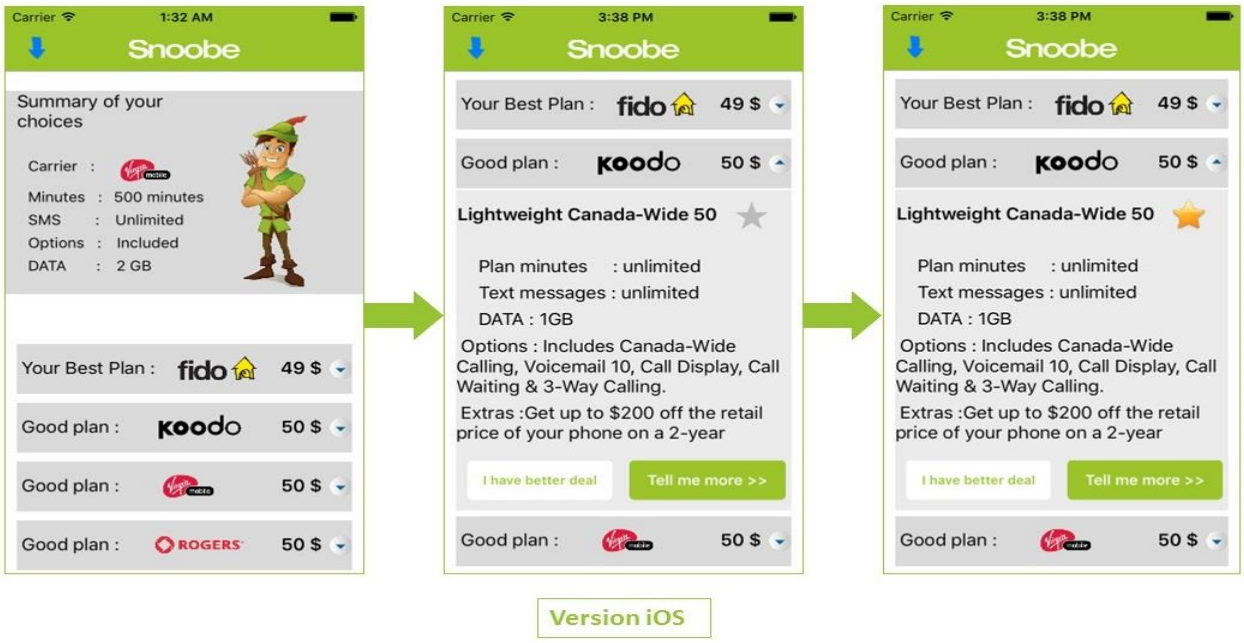


Figure 49 - Les interfaces utilisateurs résultats de la recherche des forfaits vers iOS

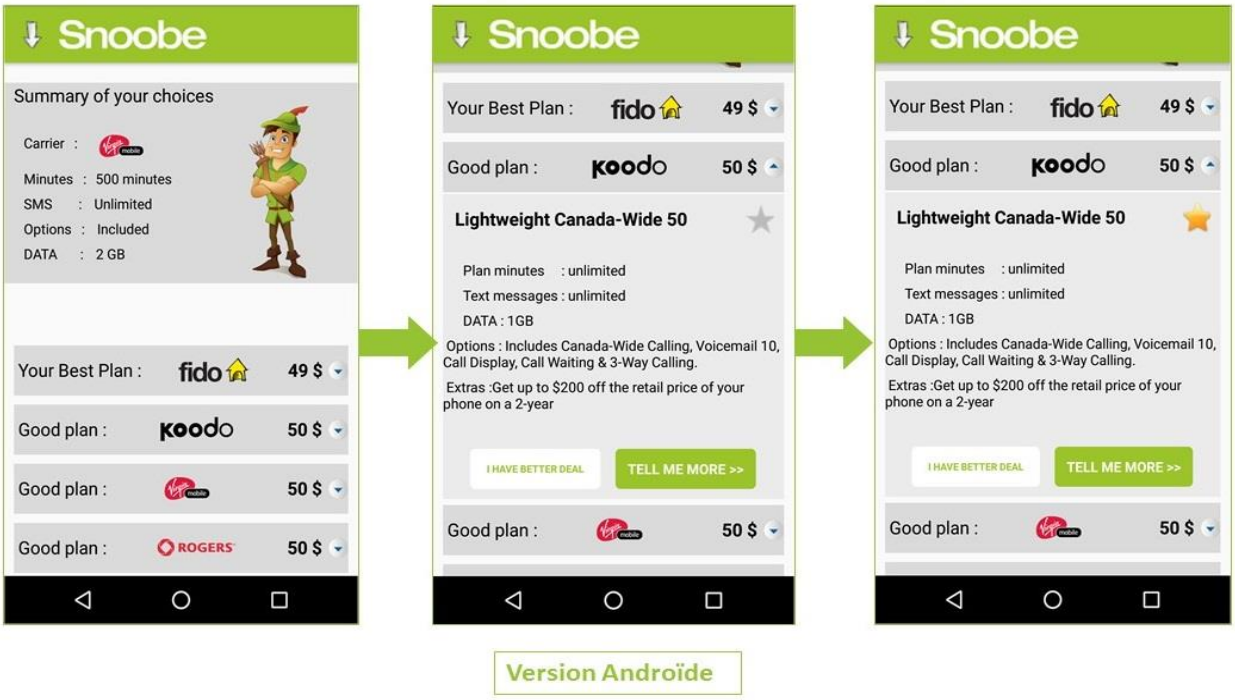


Figure 50 - Les interfaces utilisateurs résultats de la recherche des forfaits vers Androïde

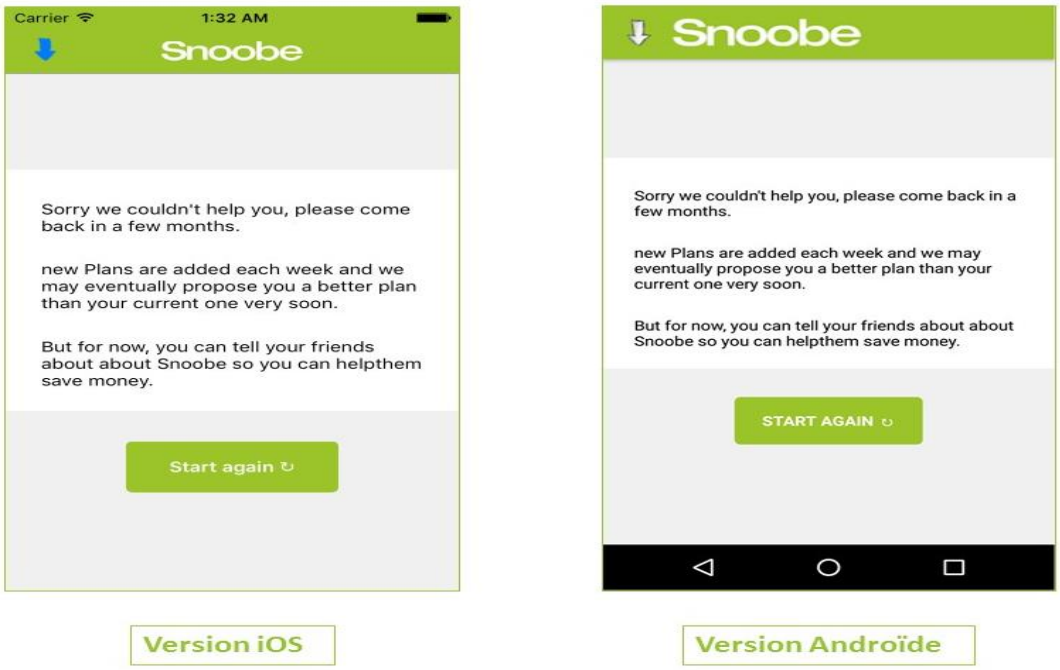
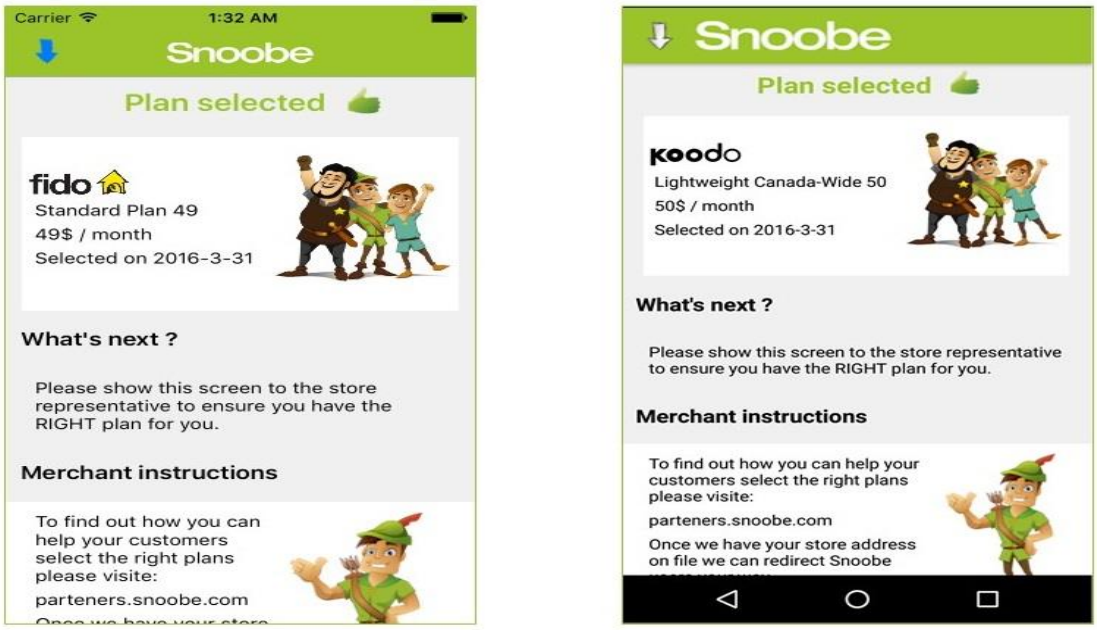


Figure 51 - Interface utilisateur j'ai une meilleure proposition

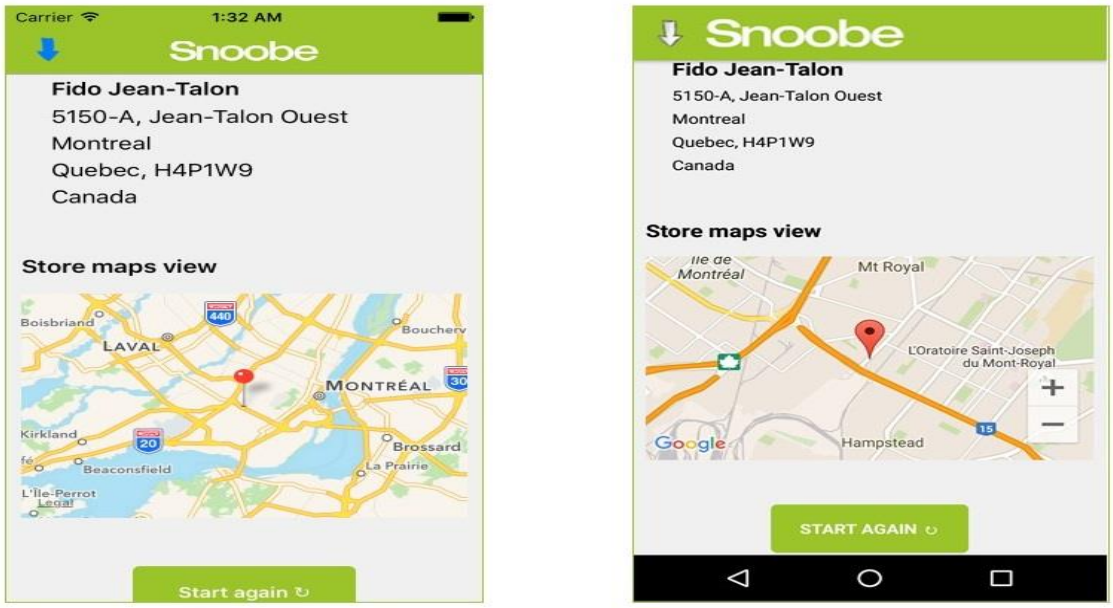
L’interface utilisateur du choix du forfait affiche les informations du forfait, quoi faire pour l’obtenir ainsi que le magasin le plus proche qui offre ce forfait et son adresse.



Version iOS

Version Androïde

Figure 52 – Interface utilisateur du forfait sélectionné - partie 1



Version iOS

Version Androïde

Figure 53 - Interface utilisateur du forfait sélectionné - partie 2

La figure suivante montre l'interface utilisateur qui permet d'afficher les forfaits sélectionnés comme favoris :

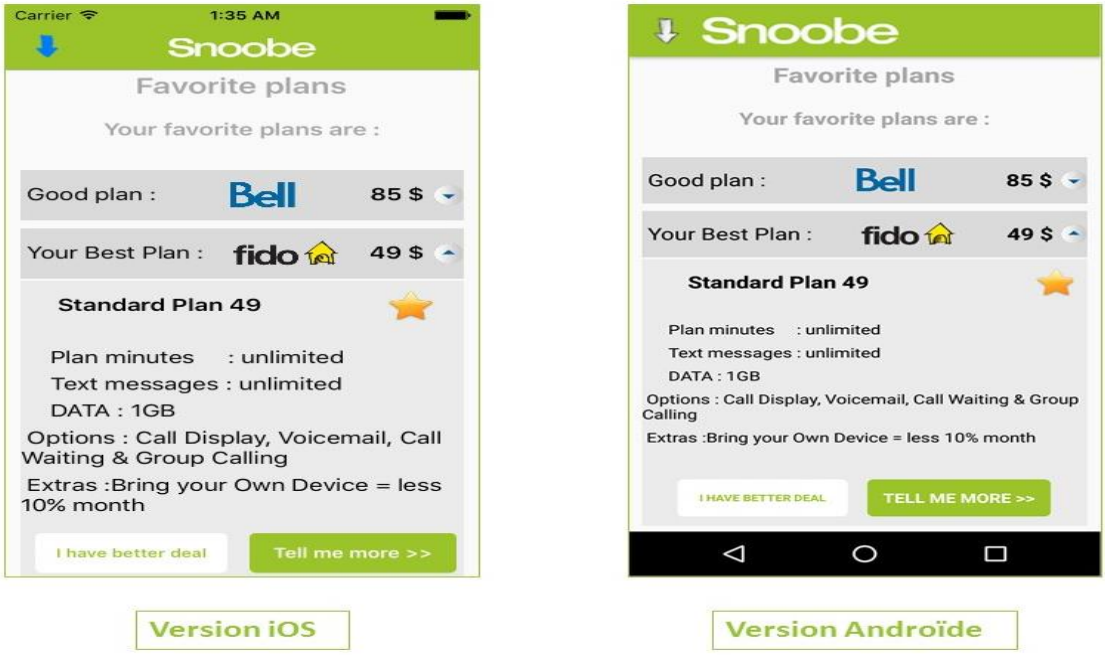


Figure 54 - Interface utilisateur des forfaits marqués favoris

La figure suivante montre l'interface utilisateur qui permet d'afficher les forfaits qui sont en promotion :

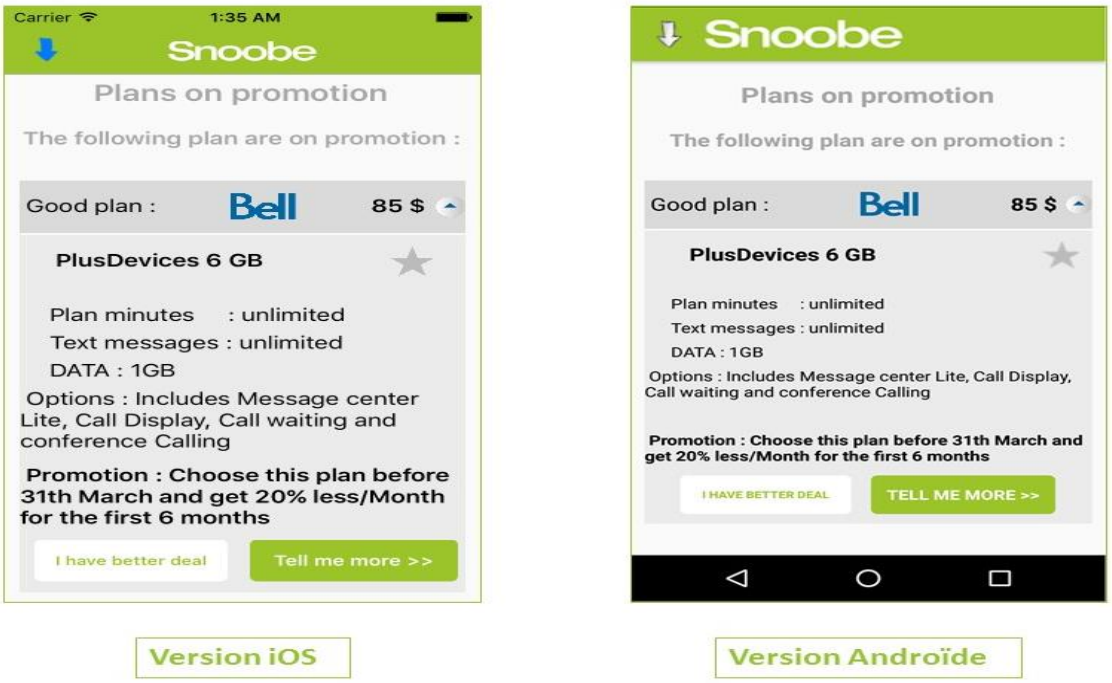


Figure 55 - Interface utilisateur des forfaits en promotion

La dernière interface utilisateur de l'application illustre le concept de Snoobe et donne une bonne description aux utilisateurs de ce que l'application Snoobe peut offrir à ces utilisateurs.

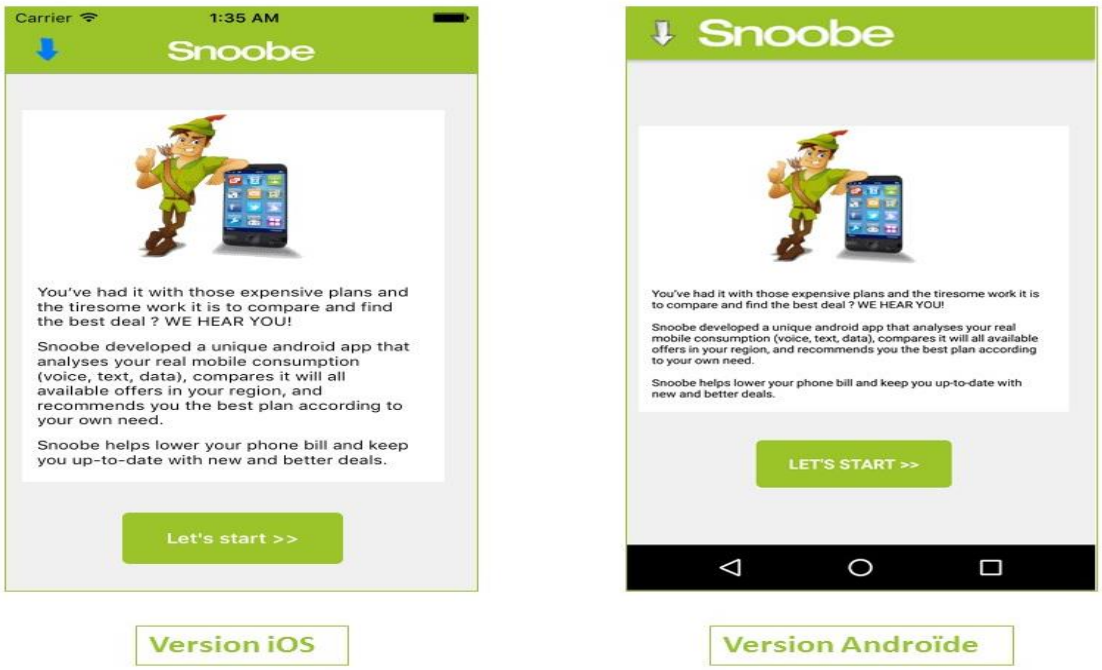


Figure 56 - Interface utilisateur d'à-propos de Snoobe

BIBLIOGRAPHIE

[1] Yiwen Huang, R.and Symonds, J. "*Mobile Marketing Evolution: Systematic Literature Review on Multi-Channel Communication and Multi-Characteristics Campaign*", pp. 157 - 165, 2009.

[2] Bamba, F. and Barnes, S.J. "*SMS advertising, permission and the consumer: A study,"* Business Processes Management Journal, vol. 13, pp. 815-824, 2007.

[3] Tsang, M.M. Ho, S.C. and Liang, T.P. *"Consumer attitudes toward mobile advertising: An empirical study,"* International Journal of Electronic Commerce, vol. 8, pp. 65-78, 2004.

[4] Application de M-Marketing pour divers produit : « http://www.bonial.fr/».

[5] Application de M-Marketing pour divers produit : « http://www.prixing.fr/ ».

[6] Dhillon, B.S. "*Engineering and Technology Management Tools and Applications*", Artech House, 2002.

[7] Byrne, E.J. "*A Conceptual Foundation for Software Re-engineering*", Software Maintenance, 1992. Proceerdings, Conference on, 1992.

[8] Telea, A. C. "*Reverse engineering –recent advances and applications* ", InTech, 2012.

[9] Petzold, C., *"Creating Mobile Apps with Xamarin.Forms"*, Microsoft Press, 2015.

[10] Charkaoui, S. Adraoui, Z., Benlahmar, E.H. "*Cross-platform mobile development approaches* ", Information Science and Technology (CIST), 2014.

[11] Tian, L., Du, H., Tang, L. Xu, Y. "*The Discussion of Cross-Platform MobileApplication Based on Phonegap* ", Software Engineering and Service Science (ICSESS), 2013.

[12] Site web official d'Apache Cordova (Phonegap) "http://www.phonegap.com/", *2016.*

[13] Hiren Dave, "Instant Sencha Touch", Packt Publishing, 2013.

[14] Site web officiel de Sencha Touch : " *https://www.sencha.com/products/touch*", 2016.

[15] Williams, D. "*Corona SDK Application Design*", Birmingham : Packt Publishing, Limited, April 2013

[16] Site official de Corona SDK, " *https://coronalabs.com/products/corona-sdk*", 2016.

[17] Outil de statistique d'utilisation des logiciels informatiques, "http://gs.statcounter.com", 2016

[18] Grønli, T.M., Hansen, J., Ghinea, H., Younas, M. "*Mobile Application Platform Heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS*", Advanced Information Networking and Applications (AINA), 2014.