h e g

Haute école de gestion
Genève

Choix de développement mobile multiplateforme, application native ou hybride ?

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

par :

Andy CHRISTEN

Conseiller au travail de Bachelor :

Rolf HAURI, Chargé d'enseignement HES

Genève, le 30 septembre 2015

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Hes·so GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

Déclaration

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

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

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

Fait à Genève, le 30 septembre 2015

Andy CHRISTEN

Remerciements

Je tiens à remercier vivement M. Rolf Hauri, pour son aide à l'orientation de ce travail de Bachelor, ses conseils avisés ainsi que ses excellents cours de JavaScript, contributeurs essentiels à l'accomplissement de ce projet.

Un grand merci à ma famille également, pour son soutien tout au long de mon cursus, et tout particulièrement à ma mère, Corinne Kneuss, relectrice attentionnée de cet écrit.

Résumé

Ce travail a pour objectif de présenter le panorama actuel des alternatives de développement d'applications mobiles proposant une approche multiplateforme. Se basant sur ce panorama, il vise également à préciser les caractéristiques et possibilités fonctionnelles offertes par celles-ci, par opposition à une approche de développement traditionnelle native.

Trois solutions parmi les plus populaires au moment de la réalisation de ce travail sont présentées, différents critères de choix étant également proposés pour chacune.

La première partie de ce document présente l'évolution des appareils mobiles et de leurs principaux acteurs, puis l'apparition des applications mobiles à proprement parler, pour arriver à l'ampleur du marché actuel et les enjeux impliqués pour la thématique de leur développement.

La suite de ce travail définit ce qu'est une approche de développement multiplateforme ainsi que ses différences par rapport à une approche native. Différentes approches sont présentées ainsi que quelques acteurs principaux du domaine, l'une d'elles étant testée à travers la création d'un prototype d'application, également présenté.

La fin du travail se concentre sur les critères de choix de solution discernés lors des recherches et conclu sur le positionnement actuel du développement mobile multiplateforme, par rapport au développement natif.

Mots clés : développement, mobile, multiplateforme, hybride, applicatif, PhoneGap, Cordova, Appcelerator, Titanium, Xamarin, Mono

Table des matières

Liste des tableaux

Liste des figures

1. Introduction

Notre mode de vie a évolué, depuis un peu moins d'une dizaine d'année, pour intégrer un nombre croissant d'innovations technologique, pour une majorité d'entre elles fortement liées à la mobilité: téléphones, tablettes, bracelets, montres et lunettes connectées…

En les considérant dans leur ensemble, il est apparent que ces supports ont tous trois caractéristiques essentielles en commun : une portabilité et une ergonomie maximales ainsi qu'un grand nombre d'applications ou fonctionnalités disponibles. Leur objectif ? Réussir à faire part intégrante de la vie de leur utilisateur; un tour de force qui explique la vague de succès qu'ont initiée les premiers smartphones à large écran multi-tactile dont le fer de lance, l'iPhone, a été considéré comme une révolution à sa sortie en 2007.

Avec du recul, ce sentiment se comprend aisément : dans un monde tourné vers le rendement, la rapidité et la simplicité de service, présenter un produit à l'ergonomie novatrice et sans contrepartie ne pouvait que difficilement engendrer un échec. Ajoutez à cela la possibilité d'accéder en quelques pressions du doigt à un nombre aujourd'hui incalculable d'applications et vous tenez l'explication de ce franc succès.

S'il est largement reconnu que les smartphones ont, sinon changé, tout du moins impacté le mode de vie de beaucoup d'entre nous, il est moins évident que cet engouement pour le mobile, et surtout pour les applications mobiles, représente des challenges pour les développeurs.

En effet, le nombre croissant d'acteurs métier que ce nouveau marché a attiré a amplifié une problématique qui, sans concerner directement l'utilisateur, constitue un obstacle de taille pour le développeur : il s'agit là de la multiplication des supports et plateformes.

Sans être un phénomène nouveau, différentes machines et systèmes d'exploitation pour ordinateurs personnels coexistaient déjà bien avant l'apparition des smartphones, le besoin de développer des applications similaires sur différentes plateformes a pris ici une dimension plus contraignante en raison du nombre d'acteurs. Qu'il s'agisse d'Android, iOS, Windows Phone ou BlackBerry pour ne citer que les quatre plus grands, développer une même application pour chacun de manière générique requiert une expérience de quatre environnements différents et de plusieurs langages probablement sans possibilité aucune de réutilisabilité de code. Cela représente une quantité de ressources considérable pour une application qui peut être simple au départ.

Depuis quelques années, cependant, différentes solutions sont apparues avec pour optique de traiter cette problématique du développement multiplateforme. Bien qu'utilisant différentes approches, toutes visent à permettre le développement d'une application unique et donc dans un seul environnement et langage. Application qui fonctionnera ensuite indépendamment de la plateforme cible.

L'objectif de ce travail est de présenter les enjeux et les contraintes liés au développement multiplateforme d'applications mobiles ainsi que ces différentes approches existantes. L'analyse de ces solutions ainsi que l'essai pratique de certaines à travers une application simple permettra ensuite de définir si l'une d'entre elles répond à toutes les attentes pour être considérée comme solution idéale ou, si tel n'est pas le cas, de définir les atouts de chacune afin de formuler des recommandations d'utilisation selon les besoins de l'application à développer.

2. L'évolution du mobile applicatif

Les innovations technologiques aidant, un long chemin a été parcouru depuis que Martin Cooper, ingénieur chez Motorola, passa le premier appel depuis un téléphone mobile, en 1973.[1] De fait, en raison de ces innovations, la notion même du mot mobile ainsi que les attentes qui y sont associées ont évolué :

De l'objectif initial très technique qui était de rendre portable la communication téléphonique, nous sommes passés par les premières applications mobiles (calendrier, carnet d'adresses, de notes…) au courant des années 1990 pour finalement arriver aux ordinateurs de poche que nous connaissons. Cette évolution est le signe d'un changement graduel de mentalité dans lequel l'application, avec ses innombrables possibilités, a aujourd'hui pris le rôle central.

2.1 Historique des possibilités de développement

2.1.1 Les débuts et les contraintes technologiques

Malgré un succès très mitigé, l'IBM Simon, lancé commercialement en 1994 soit 13 ans avant l'IPhone, est souvent considéré comme le premier smartphone. Dans le cadre informatique, imaginer que ce concept ait été abordé aussi longtemps avant son avènement peut surprendre, il faut néanmoins réaliser que le contexte de l'époque était très différent, d'un point de vue technologique évidemment mais également au niveau du marché.

De fait, la "vision" du mobile et des applications au début des années 1990 était entièrement restreinte à une utilisation orientée business. Les applications mobiles de l'époque étaient uniquement présentes sur les assistants personnels (PDA ou Personnal Digital Assistant) qui eux-mêmes servaient d'outils de travail encore très exclusifs et ne permettaient pas de lancer d'appels, les téléphones cellulaires y étant entièrement dévoués. En raison de ce marché très spécifique et restreint, le nombre d'acteurs était limité à quelques leaders tels Motorola puis Nokia pour les téléphones et Psion (qui deviendra Symbian Ltd) avant l'arrivée de Palm pour les assistants personnels.

[1] « History of mobile phones ». Disponible à l'adresse : https://en.wikipedia.org/wiki/History_of_mobile_phones

Le développement d'applications mobiles à proprement parler eut tout de même l'occasion de se démocratiser à cette époque, notamment grâce au lancement par Psion du système d'exploitation EPOC16 pour sa ligne d'assistants personnels "Psion Series 3" (lancée en 1991) [2].

Ce système s'est démarqué par son langage de programmation embarqué appelé OPL (Open Programming Language) permettant à l'utilisateur de créer ses propres applications aux performances et à l'apparence similaires aux applications de base. Cette possibilité eu beaucoup de succès et suscita l'apparition d'une communauté autour du développement OPL précurseur de la tendance applicative actuelle.

Même si un mouvement similaire apparu quelques années plus tard grâce au succès des assistants Palm (basés eux sur les langages C/C++), la problématique de développement multiplateforme ne se posait pas encore, le marché étant largement dominé par ces quelques acteurs.

L'explication de cette situation vient évidemment des restrictions technologiques de l'époque qui ne permettaient pas de proposer une ergonomie attrayante dans un contexte autre que professionnel : si certains assistants personnels pouvaient tenir plusieurs dizaines d'heures sur batterie, leur écran monochrome de faible résolution, leur poids et leur taille présentaient peu d'attrait. La situation étant pire pour les téléphones mobiles, qui commençaient seulement à se distancer du standard "brique" avec un poids s'éloignant lentement du kilogramme pour une autonomie de quelques heures à peine.

Le développement global du réseau 2G avec l'apparition des services tels que SMS et MMS contribua par la suite rapidement à la croissance du marché mobile. Il fallut cependant attendre une amélioration des vitesses de transfert de données pour que de réels smartphones proposant notamment le téléchargement d'applications fassent leur apparition au début des années 2000.

[2] « Psion Series 3 ». Disponible à l'adresse : https://en.wikipedia.org/wiki/Psion_Series_3

A lire également si le sujet vous intéresse « Programming Psion Computers » by Leigh Edwards, disponible à l'adresse : http://www.tobidog.com/ppc.pdf

2.1.2 L'apparition de la 3G

La multiplication des réseaux 3G et l'amélioration des performances hardware lors de la première moitié des années 2000 permirent l'apparition rapide de nouvelles fonctionnalités : web mobile, Wireless Lan, services multimédia tels que musique, capture de photos et vidéos ou jeux contribuèrent à populariser les smartphones en les démarquant de l'usage exclusivement professionnel. Nokia fût le principal acteur de ce succès notamment grâce à sa « série N », fabriquant jusqu'à 50% des smartphones vendus dans le monde jusqu'en 2007 (année de l'apparition de l'IPhone).

Cette vague d'innovations technologiques initia le mouvement du développement d'applications tierces qui ne cessera de croître jusqu'à aujourd'hui.

Du point de vue développeur, au premier abord, cette époque suivait la même tendance mono plateforme que lors de la décennie précédente : Le système d'exploitation Symbian, successeur d'EPOC, s'affichait comme dominateur, possédant plus de 70% de parts de marché entre 2005 et 2006[3] avant de décliner à partir de 2007.

L'entreprise encourageait volontiers le développement tiers, lançant en 2005 le programme « Symbian Signed » permettant d'approuver les applications indépendantes sans les forcer à passer par un centre de test.

Un grand nombre d'applications ayant été développées pour ce système (environ 10'000 en 2007), Symbian envisagea même l'ouverture d'un app store. Ce projet, s'il s'était concrétisé, nombreux sont ceux qui considèrent qu'il aurait pu changer le cours de l'histoire ou tout du moins limiter l'effet IPhone à venir.

Quelques nouvelles difficultés commençaient cependant à apparaître : le nombre de constructeurs actifs sur le marché augmentait régulièrement tandis que les modèles de téléphones se renouvelaient continuellement. Même si Symbian en tant que système était la norme, il fallait gérer les interfaces utilisateurs tributaires de quatre distributions possibles du système (S60, S80, S90 pour Nokia et UIQ pour Sony-Ericsson) et différentes selon les modèles.

[3] « Android before Android: … » by Jo Best.
Disponible à l'adresse http://www.zdnet.com/article/android-before-android-the-long-strange-history-of-symbian-and-why-it-matters-for-nokias-future/

Malgré la possibilité de réutilisation de parties du code, basé sur le même langage, il fallait redévelopper plusieurs fois une application pour la rendre disponible sur les différentes interfaces utilisateurs existantes, ce qui compliquait sérieusement la tâche de développement.

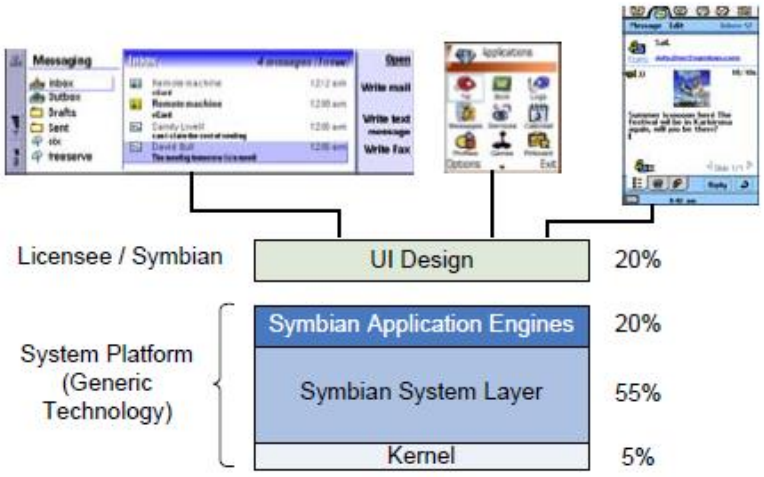

Figure 1 : Architecture d'application Symbian

2.1.3 L'iPhone et l'arrivée des app stores

Etant donnée la domination du système d'exploitation Symbian sur toute la période d'apparition et popularisation des smartphones (encore 65% de parts de marché courant 2007), il est difficile d'imaginer les raisons de sa chute, continue jusqu'à son abandon définitif par Nokia en 2011.

Il existe cependant un certain nombre de pistes expliquant le succès sans précédent de l'iPhone à sa sortie en 2007 ainsi que celui de ses concurrents à venir tels les appareils Android les années suivantes. Nous pouvons en retenir deux principaux[4].

[4] « What factors contributed to the success of Apple's iPhone ? » by John Laugesen et Yufei Yuan. Disponible à l'adresse : http://www.msitmonline.com/media/Cin/compre/1_What%20Factors%20Contributed%20to%20the%20Success%20of%20Apple_s%20iPhone.pdf

Simplicité : Symbian fût développé spécialement dans le but de cibler des appareils mobiles à l'origine simples de fonctionnalités (rappelons qu'il s'agit du successeur d'EPOC, au cœur de la ligne d'assistants personnels de Psion), alors qu'iOS et Android dérivent eux du monde de l'informatique. Cela leur donna dès le départ un avantage de flexibilité qu'ils exploitèrent rapidement par exemple pour l'intégration et l'exploitation des fonctionnalités tactiles telles multi-touch et slide.

En effet, en dehors de son design physique soigné et de bonnes caractéristiques hardware, l'iPhone représenta une révolution par la qualité de son interface utilisateur dont une ergonomie du système hors du commun qui rendait son utilisation réellement plaisante et accrocheuse.

Lors d'une interview en 2013[5], Tony Cripps, analyste pour l'entreprise de consulting informatique et télécommunications Ovum décrit la situation de la manière suivante :

> *« Des décisions avaient été prises pour l'architecture de Symbian et datant de sa création qui impliquaient qu'il serait difficilement flexible en termes de comment adapter l'expérience utilisateur pour accommoder des choses telles que tactile et mouvements. »*

Applications et app stores : Bien qu'Apple n'ouvrit pas son App Store avant juillet 2008, les applications de base et particulièrement le navigateur mobile basé sur safari rencontrèrent un franc succès. L'entreprise réagit rapidement en lançant une première version béta de son SDK (Software Development Kit) en mars, téléchargée plus de 100'000 fois en 4 jours[6]. La fièvre des applications mobiles était déjà bien lancée.

Effectivement, un peu moins d'une année après la sortie de l'App Store, Apple pouvait se vanter de proposer 100'000 applications sur celui-ci, alors qu'il fallut 7 ans à Symbian pour arriver à 10'000 applications, moins faciles d'accès pour les utilisateurs.

[5] « Android before Android: … » by Jo Best.
Disponible à l'adresse http://www.zdnet.com/article/android-before-android-the-long-strange-history-of-symbian-and-why-it-matters-for-nokias-future/

[6] « iPhone SDK downloads surpass 100,000 » by J. D. O'Grady,
Disponible à l'adresse http://www.zdnet.com/article/iphone-sdk-downloads-surpass-100000/

Si, lors d'une interview, David West, ancien cadre chez Symbian, en donna la raison suivante :

> *« Symbian était trop limité par son code legacy et sa base installée pour répondre au challenge d'APIs modernes et des meilleures outils de développement fournis par Apple et Google, tous deux ayant pu démarrer "à neuf". »*

Comme le montrent les graphiques suivants, la mise à disposition des applications à travers un store a sans aucun doute grandement contribué au succès des leaders actuels du marché des smartphones.

Figure 2 : Nombre d'applications par stores (AppAnnie)

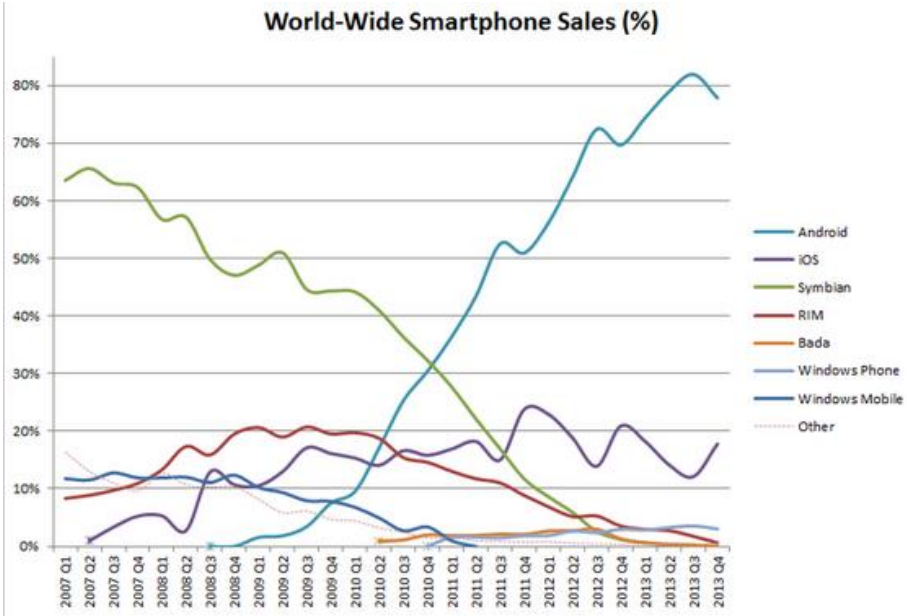

Figure 3 : Parts de marché sur ventes de smartphones (Gartner)

2.2 L'approche du mobile aujourd'hui

Bien que les fonctionnalités principales des appareils mobiles n'aient pas connu de changements majeurs (mis à part l'amélioration constante de leurs performances) depuis maintenant plusieurs d'années, leur popularité n'a cessé d'augmenter, de même que leurs cas d'utilisation se sont diversifiés. Désormais omniprésent, le smartphone a son usage en toutes occasions, comme nous pouvons le voir sur le tableau suivant présentant la répartition de différents lieux selon le taux d'utilisation du smartphone en Suisse.[7]

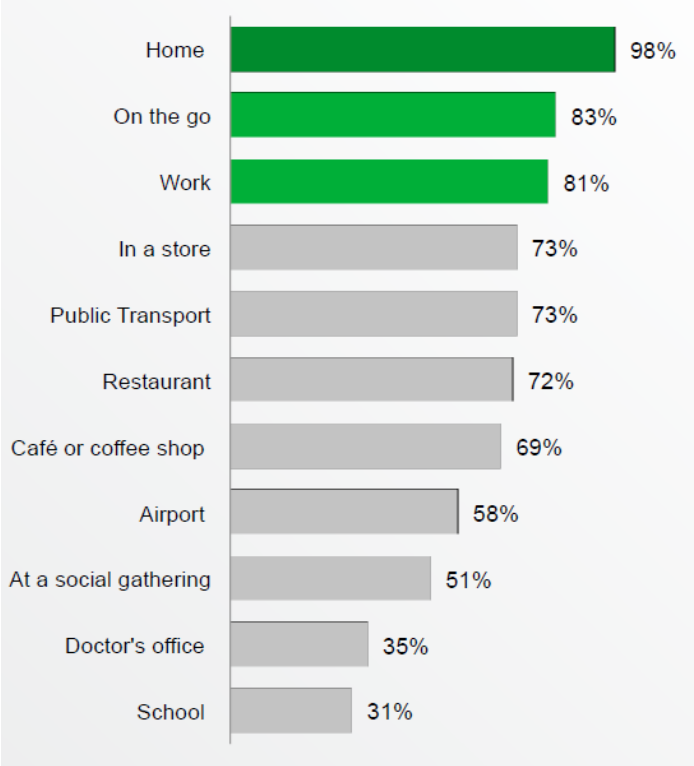

Figure 4 : Taux d'utilisation du smartphone selon localisation

(Ipsos OTX MediaCT)

[7] Table tirée d'un rapport de sondage sur l'utilisation du mobile en Suisse effectué par Ipsos OTX MediaCT « Our Mobile Planet : Switzerland | Understanding the Mobile Consumer ». Disponible à l'adresse : http://services.google.com/fh/files/blogs/our_mobile_planet_switzerland_en.pdf

Si l'intérêt pour les appareils mobiles semble aujourd'hui incontestablement universel, force est cependant de constater qu'il existe toujours une différence entre leur approche professionnelle, soit l'utilité qu'ils représentent pour une entreprise ou comme outil de travail, et leur approche utilisateur/consommateur.

2.2.1 Pour l'utilisateur

Dans les rapports de l'utilisateur à son mobile, les applications qu'il utilise (qu'elles soient basées sur le web ou locales à l'appareil) sont fréquemment classées dans deux catégories différentes :

Les réseaux sociaux et loisirs, tels que jeux, écoute de musique ou visionnage de vidéos représentent l'activité principale, avec plus des deux tiers du temps passé sur notre smartphone qui leur est dévolu.

L'information et la productivité, telles que recherches, médias et communication (de type mail par exemple) pour le temps restant, soit à peu près un tiers.

Les différences de possibilités entre web mobile (applications accédées depuis un browser tel Firefox, Chrome ou Safari) et applications installées directement sur l'appareil tendant à disparaître, l'utilisateur est donc souvent confronté au choix entre ces deux possibilités et va généralement se décider pour l'une ou l'autre selon le type de l'activité concernée[8].

Cependant, et cela est présenté plus en détail à la fin de ce chapitre, il est intéressant de remarquer que ce choix s'effectue très fréquemment en faveur de l'application installée, pour des fonctionnalités qui peuvent être identiques sur les deux supports.

[8] Le développement à suivre présente des critères de choix d'une méthode selon un point de vue utilisateur en termes de facilité et accessibilité du service.
Les différences de possibilités techniques et fonctionnelles sont, elles, présentées dans un chapitre ultérieur.

2.2.1.1 Le ‘’site web’’

Facile d’accès, l’application web, à savoir tout site web accessible dans une version "mobile", est généralement le choix de prédilection pour les activités non régulières voire à usage unique telles que réservations, achats occasionnels voir simple consultation d’un site spécifique ou d’un moteur de recherche pour des besoins d’information.

L’intérêt principal d’une application web tient au fait que l’utilisateur n’a pas à se soucier d’installer celle-ci voir éventuellement de devoir l’acheter au préalable. Il n’a donc également pas besoin de la configurer, de la tenir à jour et, dans une moindre mesure, de se préoccuper des informations privées qui lui seraient accessibles.

Une telle application est cependant entièrement tributaire de l’accès au réseau et peut donc s’avérer totalement inaccessible selon l’état de ce dernier. D’autre part, l’ergonomie de l’application peut être inférieure à ce que l’utilisateur est habitué à utiliser avec des applications installées.

2.2.1.2 L’application installée

De manière similaire à la création de favoris dans un navigateur pour garder facilement accessible une page ou un site internet fréquemment utilisé, l’installation de l’application en local signifie souvent que celle-ci fait partie des préférées de l’utilisateur ou tout du moins qu’il va l’utiliser à fréquence régulière : il veut donc s’en assurer une utilisabilité maximale.

Les avantages de ce type d’application vont dans ce sens : Selon le but de celle-ci, les performances peuvent être supérieures à celles d’une application web tout comme les interactions avec l’utilisateur plus faciles avec une meilleure ergonomie à la clé.

Le fait que l’application puisse fonctionner en mode dégradé, ou au minimum afficher un message de circonstance en cas d’inaccessibilité du réseau nécessaire à toutes ou certaines fonctionnalités peut également rassurer.

Ces caractéristiques contribuent à forger une impression, justifiée, de facilité et de qualité pour l’utilisateur poussant à l’adopter pour ses activités régulières telles que réseaux sociaux ou jeux.

2.2.2 Pour l'entreprise

Si l'apparition des smartphones, leurs applications dédiées et fonctionnalités novatrices telles que boussole, système gps ou appareil photo ont représenté une révolution en raison des innombrables possibilités d'utilisation pour les utilisateurs, l'approche du mobile pour le monde professionnel est, pour une majorité d'entreprises, beaucoup plus uniforme.

2.2.2.1 Un vecteur d'image

Il existe bien évidemment nombre d'utilisations professionnelles spécialisées possibles des appareils mobiles: il serait déplacé d'affirmer à un médecin qu'accéder à ses dossiers de patients en déplacement est inutile tout comme il serait peu avisé d'ignorer le potentiel d'achats effectués sur smartphone pour un shop en ligne.

Cependant, l'engouement de beaucoup d'entreprises ou organisations pour les applications mobiles ne s'explique pas en premier lieu par des intérêts fonctionnels ou productifs mais vient directement du succès de celles-ci et leur très large degré de pénétration auprès des utilisateurs privés.

L'application mobile est devenue, pour beaucoup, une incontournable vitrine à des fins de marketing, en lien direct avec le site web standard qui a longtemps tenu ce rôle. De fait, de plus en plus d'entreprises décident de baser tout ou partie de leur promotion et communication sur ce vecteur, en témoigne l'adoption croissante de techniques telles que l'utilisation de QR codes.

De manière similaire, l'explosion du nombre d'utilisateurs d'appareils mobiles représente une aubaine pour le domaine de la publicité ciblée: un grand nombre d'utilisateurs ne s'adonnant pas à la tâche fastidieuse de vérifier systémiquement les permissions de chaque application à l'installation ou lors de mises à jour, une quantité considérable d'information est exploitable pour les entreprises.

Pouvoir assurer une présence permanente et une communication personnalisée envers sa clientèle est devenu un indispensable vecteur d'image pour les entreprises, en témoigne le graphique suivant présentant les dépenses mondiales de publicité mobile des deux dernières années et des projections pour les années à venir.

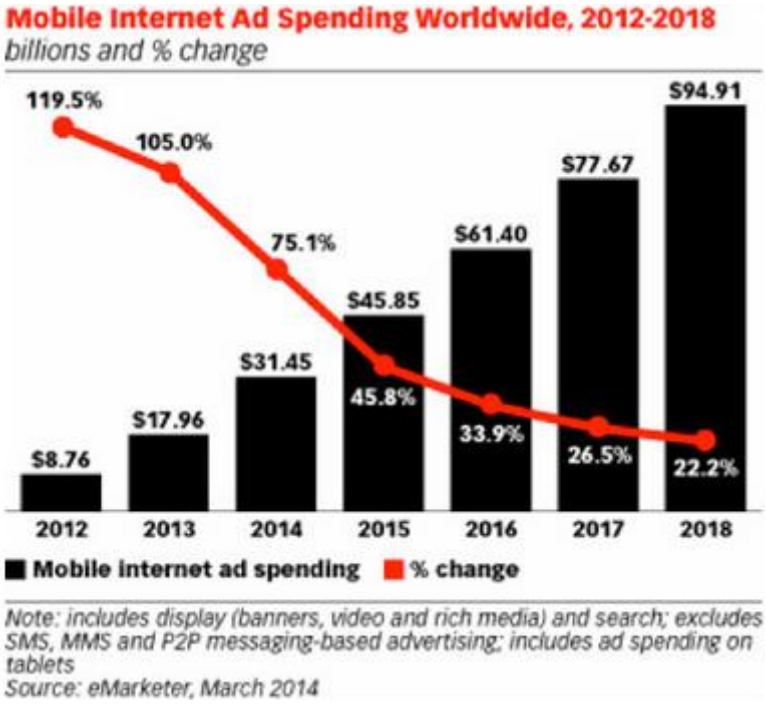

Figure 5 : Dépenses mondiales en publicité mobile

(eMarketer)

2.2.3 Une tendance à l'applicatif

Nous l'avons vu précédemment, un utilisateur souhaitant accéder à un service, ou un service devant être rendu accessible, se retrouve confronté au choix, peut être difficile, entre application mobile installée ou service web.

S'il existe toujours des différences techniques, et donc de performance ou de possibilités, entre les deux technologies (ce sujet est détaillé dans le chapitre suivant), les standards et outils de développement se sont améliorés et beaucoup d'applications sont aujourd'hui disponibles ou envisageables sous les deux éventualités.

Ce fait étant, il est naturel d'imaginer une popularité similaire ou proche entre les deux types d'application, chaque utilisateur ayant ses préférences. Etonnement, tel n'est cependant pas le cas et force est de constater qu'il existe depuis quelques années une tendance à privilégier l'application installée, qui ne cesse de se répandre.

2.2.3.1 Quelques chiffres

Dans un rapport publié début 2014[9], Flurry Analytics, entreprise spécialisée dans les statistiques, monétarisation et publicité mobile, présentait le graphique suivant montrant les habitudes d'utilisation d'appareils connectés aux Etats-Unis.

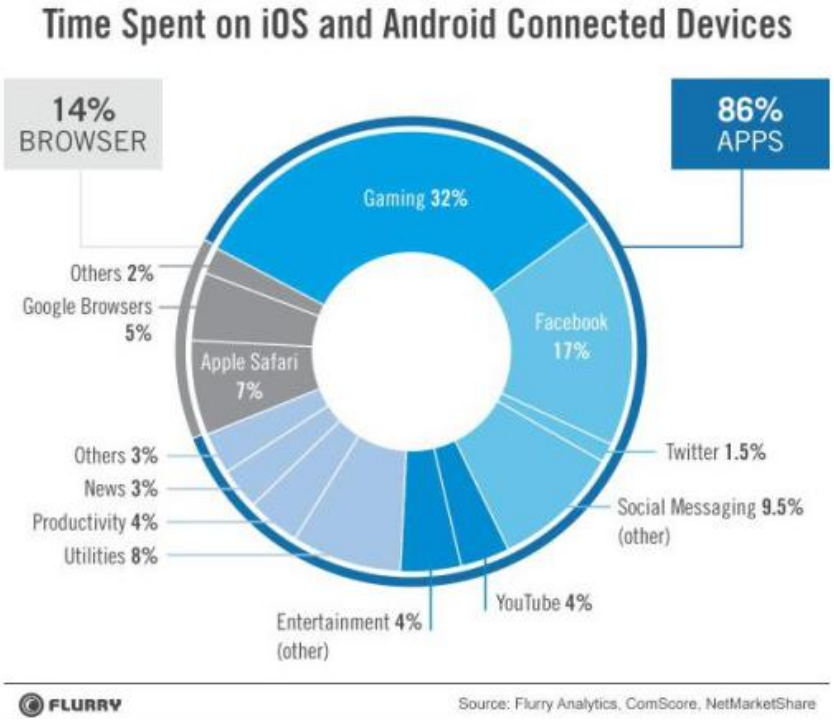

Figure 6 : Répartition du temps d'utilisation d'appareils connectés par activité

La différence est significative: le temps passé sur des applications (non-web) est effectivement plus de 5 fois supérieur à celui passé sur un navigateur. En tenant compte du fait que les données de ce rapport ne comprennent que les 3 premiers mois de l'année 2014 et que le ratio était de 80% contre 20 en 2013, la tendance apparaît très nettement.

[9] « Apps solidify leadership six years into the mobile revolution », by Simon Khalaf, Disponible à l'adresse: http://flurrymobile.tumblr.com/post/115191864580/apps-solidify-leadership-six-years-into-the-mobile

2.2.3.2 Les explications possibles

S'il est difficile d'imaginer une unique explication à ce phénomène, étant donné le nombre de facteurs entrant en ligne de compte, il est cependant possible de cerner certains mécanismes.

En se rapportant au graphique précédent, il apparait clairement que le cumul du temps passés sur des jeux et les différents réseaux sociaux représente la majorité du temps total avec plus de 60%, or ces activités sont pratiquées presque uniquement sur des applications embarquées.

La raison de cette exclusivité est, elle, plus facilement identifiable: s'il est possible de créer la même application web et native, les performances et la qualité resteront tout de même toujours différentes, avec un avantage pour l'application native ; Il peut être à la rigueur supportable d'utiliser une interface légèrement saccadée pour un service de messagerie, cela devient nettement plus dérangeant s'il s'agit d'un jeu vidéo.

Bien que de nets progrès soient continuellement faits pour améliorer les standards web, HTML5 en tête, il est peu probable que les services web puissent atteindre le niveau des applications natives dans un futur proche. De plus, le fait que certains leaders, tel Facebook[10], aient pris il y a quelques années (alors que la différence entre les deux approches était encore forte), la décision de privilégier sur le long terme les applications embarquées aide à comprendre la situation actuelle.

L'important pouvoir, en terme d'image et marketing, que représente le marché mobile peut également expliquer l'engouement générale pour les applications natives: toute entreprise souhaitant faire mieux, ou au minimum aussi bien, que son concurrent, il est facile d'imaginer l'installation d'un effet de mode et la création d'un "besoin" fictif de pouvoir proposer une application embarquée.

[10] « Facebook : "Betting on HTML5 was a mistake" », by Abel Avram, Disponible à l'adresse http://www.infoq.com/news/2012/09/Facebook-HTML5-Native

3. Le développement d'applications natives

Tout comme le succès des smartphones, tablettes et autres appareils connectés s'explique majoritairement grâce au succès des applications natives, ce dernier ne pourrait être possible sur le long terme sans une forte implication des propriétaires de systèmes d'exploitation et plateformes.

La survie d'une plateforme étant tributaire de ses applications, l'on comprend aisément que ses possesseurs aient tout intérêt à soutenir et simplifier le travail de sa base de développeurs tiers pour la fidéliser : Typiquement, cela va consister en la mise à disposition, l'amélioration et la mise à jour fréquentes d'outils et d'environnements de développement spécialisés.

3.1 En opposition aux applications web

De manière générale, il est facile d'imaginer le paradigme du développement d'application native ou d'application web comme une barrière entre un besoin de spécificité d'un côté (natif) et de généricité de l'autre (web).

Une application native va essentiellement cibler une plateforme spécifique et sera donc mise à disposition sur un app store avec des contraintes de développement bien précises. Plus important, étant installée et fonctionnant sur un appareil mobile, elle nécessite des connaissances techniques propres à un système bien défini. Il peut s'agir du langage de programmation : Java pour Android, Objective-C ou Swift pour iOS, C++ pour Windows Phone… mais également des outils et interfaces propres au système visé.

Une application web mobile, de son côté, vise la transversalité dès le départ et doit donc être disponible indépendamment du support de consultation. Elle est interprétée par des navigateurs certes spécifiques mais répondant à des même standards et langages, tels qu'aujourd'hui majoritairement HTML5/CSS3 et JavaScript, et ne requiert donc que peu de connaissances propres à chacun.

Si cette différence de granularité technique représente les principales disparités de l'approche de développement entre les deux types d'applications, il existe cependant toujours des différences fonctionnelles et de possibilités entre celles-ci.

3.1.1 Différence des possibilités et leur évolution

Les applications natives ont longtemps représenté l'unique moyen de tirer profit des fonctionnalités et composants hardware des smartphones et autres tablettes, les outils web tel que le DOM n'ayant pas été conçus dans une optique d'exploitation d'appareils nomades.

Bien que l'arrivée du standard HTML5 ait progressivement changé la donne, visant à permettre l'accès à de nombreuses fonctionnalités telles que GPS, capteurs de mouvements ou caméra, il faut du temps aux éditeurs pour adopter toutes ses spécifications et de ce fait, certaines restrictions plus ou moins contraignantes existent encore :

Par exemple, la création d'un standard supportant l'accès direct au système de fichier depuis un browser a été projetée par le World Wide Web Consortium (W3C) avant d'être abandonnée au printemps 2014[11], présumément en raison du manque d'intérêt des éditeurs principaux. En résultat, très peu de navigateurs offrent aujourd'hui cette possibilité.

L'utilisation des senseurs de luminosité, des indicateurs de charge et niveau de la batterie ainsi que les notifications par vibration de l'appareil sont également des fonctionnalités qui ne sont pas encore supportées par tous les navigateurs[12].

De plus, en raison de la structure et du fonctionnement même du DOM et malgré des améliorations constantes, les applications web ne peuvent encore réellement entrer en compétition avec des applications natives d'un point de vue de performance maximale, ces dernières étant développées en langage ultimement compilé. Les applications et jeux nécessitant des calculs et traitements lourds côté client restent donc généralement développés en natif.

[11] « File API : Directories and System », by W3C Working Group, note du 24 avril 2014
Disponible à l'adresse : http://www.w3.org/TR/file-system-api/

[12] Une liste de suivi de l'adoption des standards HTML5 par les différents navigateurs mobiles est disponible à l'adresse suivante : http://mobilehtml5.org/
Il s'agit cependant d'un service en version beta mis à jour de manière irrégulière.

3.2 La problématique du multiplateforme

Bien que les deux géants que sont Google et Apple représentent une large majorité du marché des smartphones, tablettes, smart-tv et autres appareils connectés (système d'exploitation s'entend), ce dernier n'en est pas moins très fragmenté avec un certain nombre d'autres acteurs dont évidemment Windows Phone mais également BlackBerry OS, Firefox OS, Sailfish OS, Tizen…

La tendance actuelle au "tout natif" n'est donc pas sans poser de sérieuses difficultés : Développer par exemple une même application native sur les 3 systèmes les plus répandus nécessite des connaissances spécifiques de 3 environnements et langages différents avec aucune portabilité possible si ce n'est une éventuelle architecture. Une telle optique peut rapidement revenir à doubler voire tripler les ressources nécessaires à un projet et maintenance comprise, représente un investissement considérable tout au long de la durée de vie de l'application.

4. Les applications hybrides

Observant le gain de notoriété des applications natives, et par conséquent, l'accroissement de la problématique du multiplateforme, des entreprises se sont rapidement rendues compte de l'importance de trouver des solutions permettant d'uniformiser le développement natif.

Des noms tels que PhoneGap, Appcelerator et Xamarin sont ainsi devenus aujourd'hui des incontournables du développement mobile multiplateforme grâce au succès de leurs solutions. Celles-ci sont notamment présentées plus en détails par la suite et sont utilisées ou citées dans le chapitre suivant consacré au développement d'une application prototype.

Le choix de ces solutions a été fait principalement en raison de leur popularité et des différentes approches qu'elles représentent, il est toutefois important de noter que ce travail traite d'un domaine en constante progression dont le nombre d'acteurs a fortement augmenté ces dernières années, il est donc tout à fait possible que les rapports de popularité aient évolué lors de la lecture de ce document.

4.1 Le concept général

4.1.1 Explication du terme « hybride »

Partant du constat que l'utilisation d'un duo environnement de développement / langage spécifique pour chaque plateforme constitue le nœud du problème, les différentes solutions multiplateformes consistent généralement en des outils basés sur un langage ou une combinaison de langages génériques et populaires, qui vont servir d'interface avec les fonctionnalités de l'appareil mobile.

Le développeur n'ayant plus à se soucier de connaître les outils spécifiques à chaque plateforme en développement plusieurs applications, il lui suffit de maîtriser ceux offerts par la solution choisie qui lui compilera une application native spécifique à chaque plateforme à partir de son unique projet de départ.

PhoneGap (Apache Cordova) fût notemment l'un des pionniers du développement mobile multiplateforme, proposant le populaire trio de langage web HTML5/CSS3/ JavaScript comme source pour ses applications.

L’approche utilisée par PhoneGap s’apparentant à une hybridation entre application web et native, le terme "hybride" s’est par la suite démocratisé pour désigner globalement toute application développée avec une solution multiplateforme, il est donc important de noter qu’une "application hybride" n’est pas forcément développée avec une solution exploitant "l’approche hybride".

4.2 Les approches

Si les applications hybrides sont ultimement toutes des applications natives du point de vue du système d’exploitation et de l’utilisateur, il existe effectivement différentes variantes dans la manière dont est créée l’application finale selon la solution utilisée.

4.2.1 L’approche hybride

L’approche hybride, utilisée notamment par le moteur Apache Cordova (et donc PhoneGap), vise à tirer parti des objets de type WebView (tels UIWebView sur iOS et android.webkit.WebView sur Android) disponibles nativement sur le système d’exploitation.

Les solutions exploitant cette approche compilent de manière autonome une application native spécifique à la plateforme ciblée, enveloppant l’application de départ et dont l’intégralité de l’interface utilisateur consiste en une WebView.

Celle-ci va ensuite interpréter, à la manière d’un navigateur, le code et langage web du projet qui aura été développé tout à fait comme s’il s’agissait d’une application web standard. L’accès aux fonctionnalités de l’appareil se fait grâce à des interfaces fournies dans le langage de développement, le développeur n’a donc pas ou très peu à se préoccuper de la plateforme cible, les librairies de la solution gérant les interactions avec l’environnement natif.

Le schéma à la page suivante image plus en détail l’architecture d’une application PhoneGap.

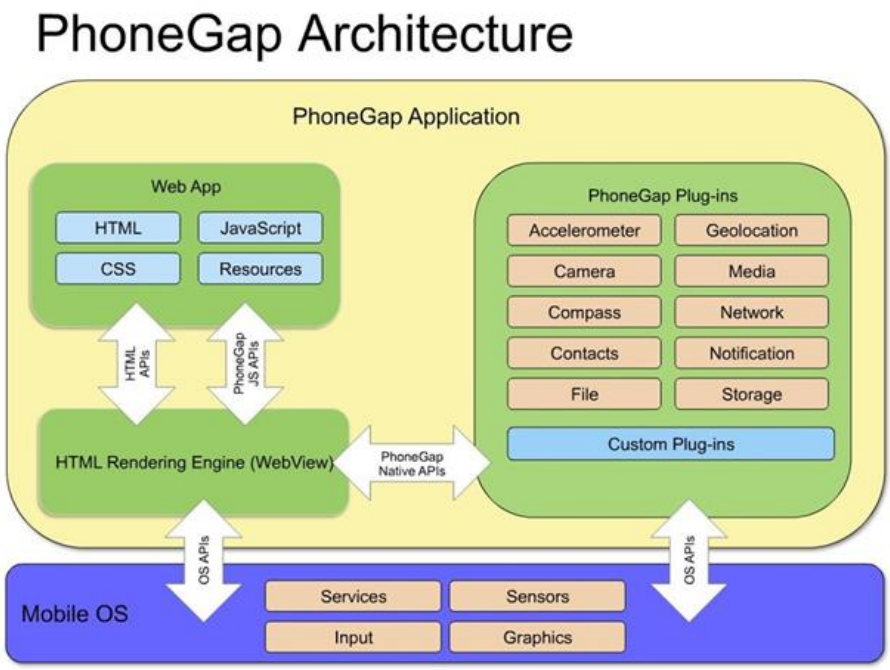

Figure 7 : Diagramme d'architecture d'une application PhoneGap

(Mammoth Digital)

4.2.2 L'approche compilation direct

Contrairement à l'approche hybride, enveloppant un langage web qui sera interprété à l'exécution de l'application finale, l'approche compilation direct tire parti d'un compilateur spécialisé propre à la solution.

Celui-ci est capable de créer directement le code assembleur natif spécifique à l'architecture ciblée à partir d'un langage de départ unique (et donc pas forcément web).

L'avantage par rapport à l'approche hybride est une performance plus élevée, l'architecture étant plus proche d'une réelle application native. Il faut cependant tenir compte du fait qu'un tel système va typiquement nécessiter un environnement d'exécution (runtime) pour l'application finale, celui-ci exposant les API permettant l'accès aux fonctionnalités système.

Cette approche est typiquement utilisée par Xamarin, les applications fonctionnant à l'intérieur du runtime Mono.

Le schéma suivant montre l'architecture d'une telle application sur une plateforme Android : comme nous pouvons le voir, l'environnement Mono accède au Kernel Linux

et l'expose aux applications Xamarin (C#) de la même manière que l'Android Runtime (ART) permet aux applications Androids standards (Java) de fonctionner.

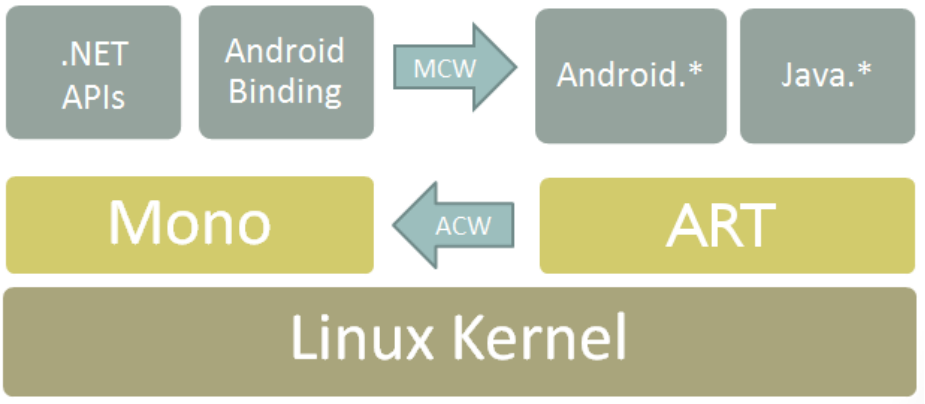

Figure 8 : Diagramme d'architecture, runtime Android - Xamarin

(Xamarin)

4.2.3 Quelques variantes

Le système développé par Apache Cordova étant en licence libre, de nombreuses solutions l'exploitant existent et rencontrent généralement beaucoup de succès auprès des développeurs. Apache Cordova en tant que tel n'étant pas un framework de développement front-end et n'étant pas lié à un IDE particulier, une grande partie des solutions l'exploitant vise à le complémenter sur ces points.

Ionic est l'une des plus populaires, proposant un framework de développement d'applications mobiles HTML, JavaScript et CSS complet tout en étant dérivé du très populaire framework Model-View-Controller, AngularJS.

Microsoft a également décidé d'intégrer des Apache Cordova Tools à son éditeur Visual Studio, qui permet désormais de créer ses projets d'applications web en tant que "solutions Cordova", facilitant la gestion des plugins d'accès aux fonctionnalités mobiles et le déploiement de son application.

D'autres systèmes encore s'apparentent aux approches présentées précédemment avec cependant quelques différences de fonctionnement.

Appcelerator utilise notamment un compilateur de JavaScript permettant un mapping avec du code natif. Si l'application n'est plus réellement développée selon un paradigme web : elle n'exploite plus une WebView à travers une application native, JavaScript n'appelant plus le DOM mais servant principalement à appeler les API natives, un interpréteur est toujours présent à l'exécution tel Mozilla Rhino sur Android et BlackBerry OS ou JavaScript core sur iOS[13].

Marmelade, finalement, est une solution proposant de compiler directement son projet C++ en instructions binaires ciblant une architecture précise (telle ARM) et y combinant un "loader" spécifique à la plateforme voulue afin de créer un package installable.

4.3 Les acteurs principaux

4.3.1 Apache Cordova / Adobe PhoneGap

PhoneGap est un framework de développement mobile multi plateforme initialement développé par Nitobi, entreprise par la suite rachetée par Adobe Systems en 2011.

Parallèlement à ce rachat, le code du projet fût cédé à la Fondation Apache et la solution renommée Apache Cordova, Adobe Systems l'utilisant par la suite pour proposer ses propres services Adobe PhoneGap puis, plus récemment, Adobe PhoneGap Build.

Apache Cordova exploite l'approche hybride présentée au chapitre précédent pour permettre la création d'applications natives à partir de projets web HTML/CSS/ JavaScript. Une très large majorité des plateformes sont supportées tel qu'Android, iOS, Windows Phone, BlackBerry, LG webOS, Firefox OS, Tizen, Ubuntu Touch, Bada et Symbian OS.

Le framework propose notemment l'accès aux fonctionnalités avancées des appareils mobiles au travers de plug-ins (APIs) permettant au code JavaScript s'exécutant dans la WebView de communiquer avec la plateforme native. Sont accessibles via les plug-ins des fonctions telles que l'état de la batterie, l'appareil photo, la boussole, la localisation par GPS[14]…

[13] Jeff Haynie, co-fondateur d'Appcelerator a fourni une explication plus détaillée du fonctionnement interne du système disponible à l'adresse suivante : http://stackoverflow.com/questions/2444001/how-does-appcelerator-titanium-mobile-work/2471774#2471774

[14] Une liste détaillée des fonctionnalités couvertes est effectuée au dernier chapitre.

Il est possible de développer ses propres plug-ins Cordova ou d'utiliser ceux mis à disposition par des développeurs tiers en plus d'utiliser ceux officiellement référencés par Apache Cordova.

Apache Cordova ne propose pas d'environnement de développement avec son framework : la création et gestion de l'arborescence de son projet d'application se fait avec une interface en ligne de commande, de même que la compilation. Les tests de l'application doivent donc également se faire, soit sur un appareil physique, soit sur des émulateurs externes (souvent ceux fournis avec les différents SDK des plateformes visées).

La solution d'Adobe PhoneGap propose elle, en complément, une application mobile, PhoneGap Developer App, permettant de débugger directement son application sur un appareil mobile physique sans besoin de recompiler, recréer et réinstaller le package de celle-ci. Elle offre également un service de compilation basé sur le cloud, Adobe PhoneGap Build, supprimant la nécessité d'avoir installé localement les SDKs de chaque plateforme ciblée ainsi qu'une application à interface graphique pour la création du projet PhoneGap (actuellement en version béta et uniquement pour windows et OS X).

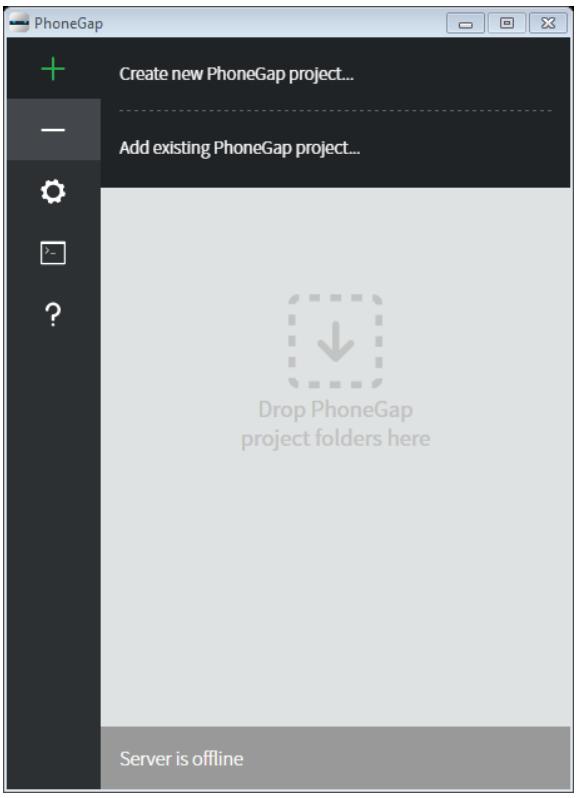

Figure 9 : Adobe PhoneGap Desktop App

4.3.2 Appcelerator Titanium Mobile

Appcelerator est une entreprise américaine spécialisée dans les technologies mobiles et fondée en 2006.

Elle lança la première version de son framework open source, Appcelerator Titanium, fin 2008, ciblant au départ le développement d'applications multiplateforme pour ordinateurs de bureau. L'entreprise commença ensuite à concentrer son activité sur les applications mobiles en ajoutant le support pour développement iPhone et Android l'année suivante puis en lançant son propre environnement de développement Appcelerator Studio en 2010.

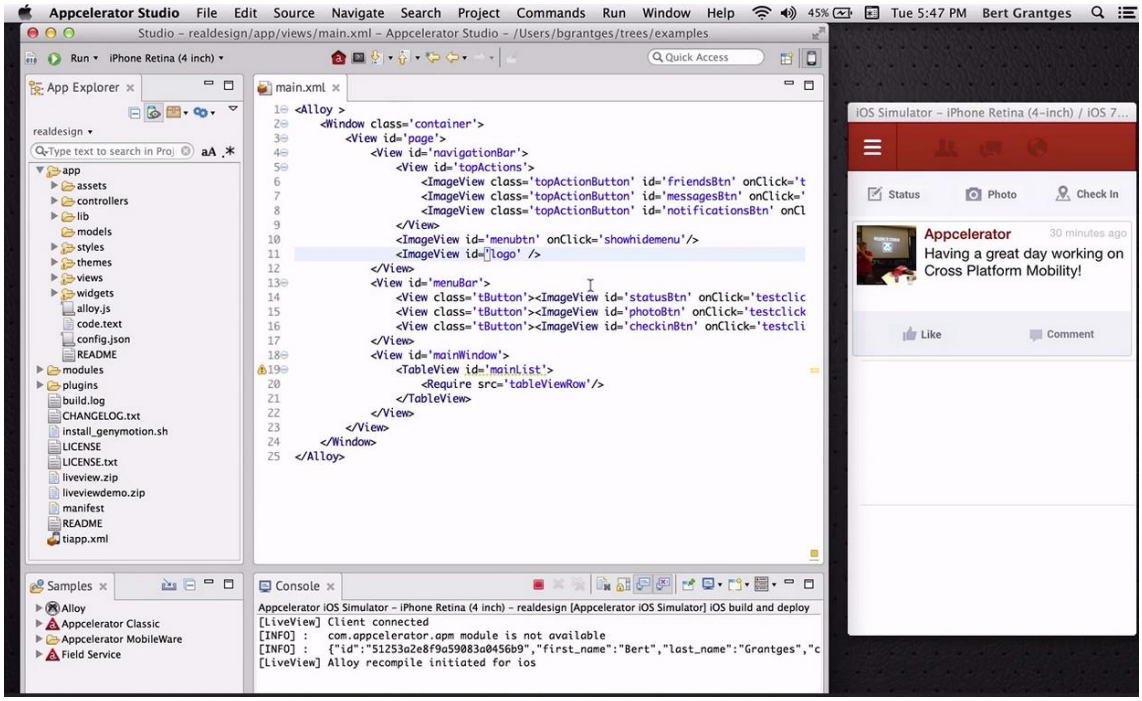

Figure 10 : Appcelerator Studio

Appcelerator Titanium offre la possibilité de développer des applications mobiles multiplateformes à partir de JavaScript, le code source de l'application étant interprété par un moteur JavaScript à l'exécution. Le framework supporte actuellement les plateformes Android, iOS, Windows Phone, BlackBerry OS et Tizen.

Bien que se passant du DOM, la dynamique de création d'interface utilisateur et l'expérience globale se veut intuitive et facile d'accès pour tout développeur familier avec le développement web.

```
var win = Ti.UI.createWindow({
    backgroundColor:'#fff'
});
var redview = Ti.UI.createView({
    top:20,
    left:20,
    width:10,
    height:10,
    backgroundColor:"red"
});
```

Figure 12 : Ajout d'éléments d'interface utilisateur – Appcelerator Titanium

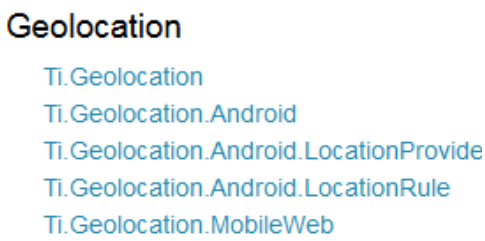

Figure 11 : Exemple d'API – Appcelerator Titanium

Les modules Appcelerator Titanium (API) offrent la possibilité d'accéder de manière standardisée à des fonctionnalités natives de l'appareil mobile telles qu'appareil photo, contacts ou système de fichier, mais également de cibler des spécificités d'une plateforme précise. Il est également possible de développer ses propres modules.

L'environnement de développement Appcelerator Studio propose également un émulateur Android et iOS pour le debugging ainsi que la possibilité de débugger / installer directement sur un appareil physique (à partir de la version 3 de l'IDE).

Bien qu'initialement Open Source et gratuit d'utilisation, les services d'Appcelerator Titanium sont récemment devenu payants (souscription) à partir du moment où une application doit être publiée.

4.3.3 Xamarin

Xamarin est une entreprise américaine fondée en 2011 par l'équipe d'ingénieurs fondatrice du projet open-source Mono.

Mono fût initié en 2004 par Ximian, puis repris par Novell: Il s'agit d'un projet visant à développer des outils respectant le standard Ecma et compatibles .NET comprenant notamment un compilateur C# et un environnement d'exécution Common Language.

Novell lança par la suite MonoTouch en 2009 (renommé Xamarin.iOS) et Mono for Android en 2011 (renommé Xamarin.Android) en tant qu'implémentations mobiles commerciales du framework Mono. Les trois solutions furent ultimement reprises par Xamarin.

Actuellement, Xamarin permet de développer des applications mobiles multi-plateforme à partir du langage C# et cela de différentes manières : Son set d'outils Xamarin.Forms permet la plus grande réutilisabilité de code pour des applications exigeant peu de fonctionnalités spécifiques aux plateformes cibles tandis que Xamarin.iOS et Xamarin.Android promettent des interactions plus spécialisées et un accès plus direct aux API spécifiques des différentes plateformes.

Android, iOS et Windows Phone sont les trois plateformes supportées par les outils de Xamarin, la dernière tirant avantage du fait que C# est le langage utilisé nativement.

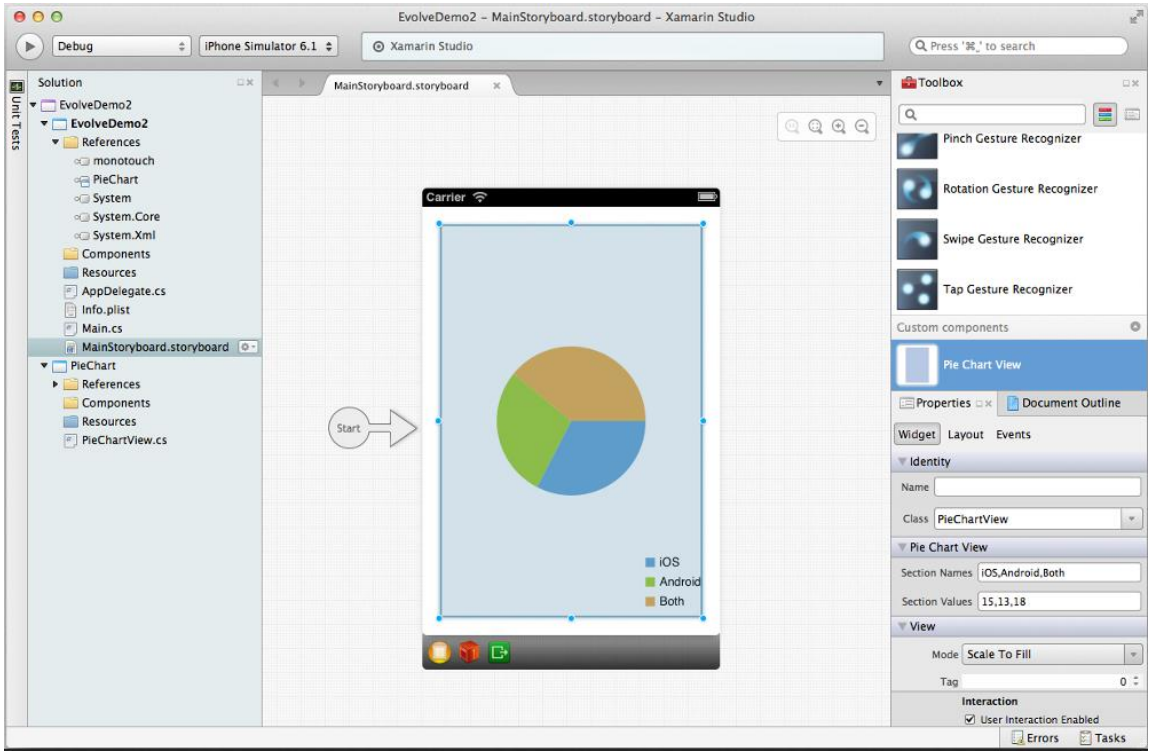

Figure 13 : Xamarin Studio

Xamarin propose son propre environnement de développement, Xamarin Studio, offrant notamment en plus du debugging, la complétion de code en C# et la possibilité de créer l'interface utilisateur graphiquement. Xamarin for Visual Studio, une extension pour l'IDE de Microsoft permet également la création, le déploiement et debugging sur simulateur ou sur appareil physique, de projets utilisant les outils Xamarin.

Tout comme Titanium, Xamarin est une solution de développement multiplateforme payante.

5. Réalisation d'un prototype

L'objectif premier de ce travail étant de déterminer les possibilités qu'offrent les différentes solutions étudiées, ainsi que, principalement, réussir à identifier les avantages et inconvénients de chacune vis-à-vis d'un développement purement natif, il m'est apparu important de pouvoir tester et présenter concrètement le développement d'une application mobile hybride à travers l'une de ces démarches.

5.1 Choix de la démarche

En raison des contraintes de durée et de charge de travail associées au travail de Bachelor, la réalisation du prototype dans chacune des trois solutions présentées précédemment n'a pu être envisagée.

Bien qu'une telle possibilité eut permis une plus grande précision dans l'analyse des trois démarches, la définition des possibilités et l'orientation vers une solution, ou un type de solution, plutôt qu'une autre selon des besoins restent tout à fait possibles au travers des recherches documentées effectuées.

PhoneGap étant l'une des démarches les plus répandues et certainement le nom le plus connu lors de la réalisation de ce travail, le choix de la solution s'est rapidement orienté vers celle-ci.

De plus, Apache Cordova ou PhoneGap n'étant pas liés à un environnement de développement particulier et se basant sur JavaScript, langage ayant une forte popularité, ce choix me permis également de concentrer la présentation du développement du prototype (chapitres suivants) sur les spécificités structurelles et d'utilisation de la démarche et non sur l'utilisation de l'environnement.

5.2 Objectifs et développement

Etant familier avec l'environnement Visual Studio et celui-ci proposant l'extension Visual Studio Tools for Apache Cordova dans sa version la plus récente (2015), le développement du prototype a été fait dans cet IDE.

Ces outils permettent d'automatiser la création de la structure du projet Apache Cordova, normalement à effectuer en ligne de commande, et de simplifier la gestion des plugins (API) ainsi que l'appel des émulateurs des différentes plateformes ou de l'appareil physique pour les tests de l'application.

Il est à noter que l'étape de distribution d'une application hybride n'étant pas concernée par les objectifs de ce travail (et se rapprochant souvent à celle d'une application purement native), la présentation du prototype se rapporte donc uniquement à la démarche de développement.

5.2.1 L'application Easy Locator et ses fonctionnalités

Le but du prototype développé est avant tout de tester la facilité d'utilisation et le fonctionnement des plugins (API) de base mis à disposition par Apache Cordova ainsi que de simuler une application typiquement mobile. A cet effet, celui-ci implémente quelques fonctionnalités usuellement répandues sur les smartphones et présente une interface utilisateur simple sans ambition de proposer une utilité et une ergonomie avancées.

Les fonctionnalités suivantes ont notamment pu être implémentées, testées et fonctionnent à la fois sur émulateurs (Android 4.* - 5.* / iOS iPhone 5) et sur un appareil physique (Android 4.2.2) :

- Etat et utilisation de la connexion internet
- Utilisation du signal GPS
- Utilisation du magnétomètre (boussole)[15]
- Capture de photo
- Notifications natives (vibration, alertes…)

L'absence de fonctionnalités sur cette liste (telle par exemple l'ajout de contacts) signifie que celles-ci n'ont pas été envisagées dans le cadre de ce prototype et non qu'elles ne fonctionnent pas pour une application développée avec Apache Cordova.

[15] Le fonctionnement de la boussole n'a pas pu être testé sur appareil physique, le smartphone utilisé pour les tests ne possédant pas de magnétomètre.

Bien que tentée, l'utilisation de l'accéléromètre n'a pas fonctionné ni sur les différents émulateurs ni sur l'appareil physique. Les deux plugins testés exploitant cette fonctionnalité génèrent présumément des erreurs non rapportées à l'exécution sur les plateformes testées.

Afin de proposer un cadre d'exploitation du réseau ainsi que des données GPS et d'orientation de l'appareil, l'application se base sur l'API Google Maps et le service associé Google Directions Service pour proposer à l'utilisateur de se situer sur une carte et trouver un itinéraire vers une destination de son choix.

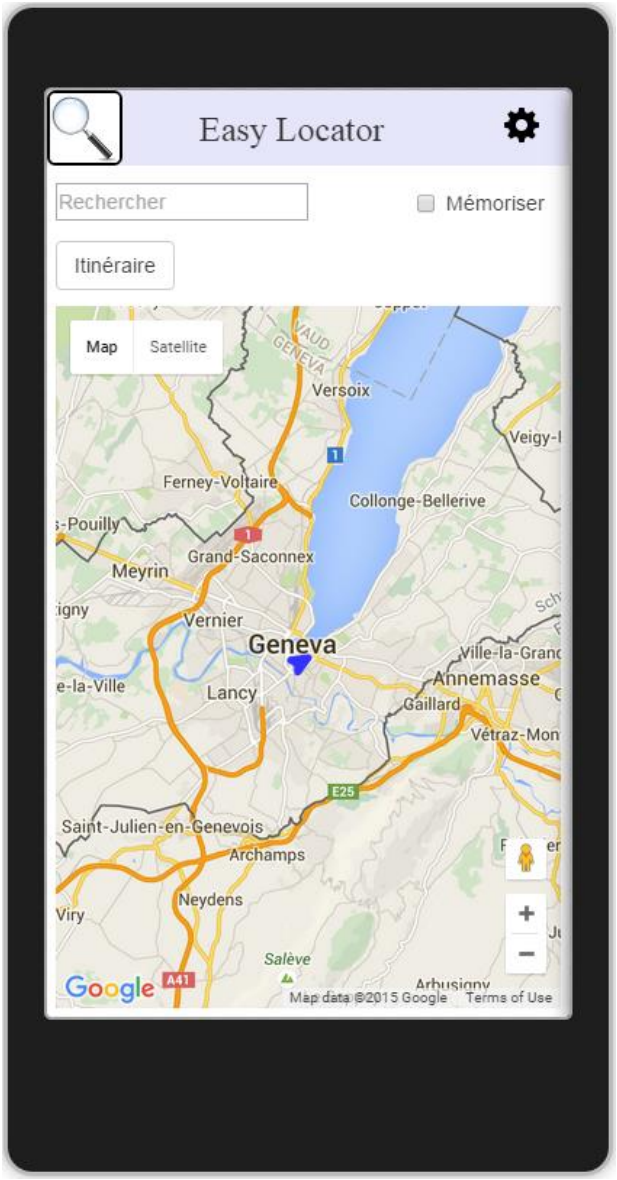

Figure 14 : Ecran de l'application prototype
(émulateur Apache Ripple – iPhone 5)

L’application charge la carte au démarrage et centre celle-ci selon les coordonnées GPS de l’utilisateur. Des alertes dans le style natif de la plateforme informent l’utilisateur en cas de problème au démarrage de même que la vibration de l’appareil signale une perte ou récupération de signal.

L’emplacement ainsi que la direction du marqueur de position sont synchronisés avec les informations GPS et l’orientation de l’appareil.

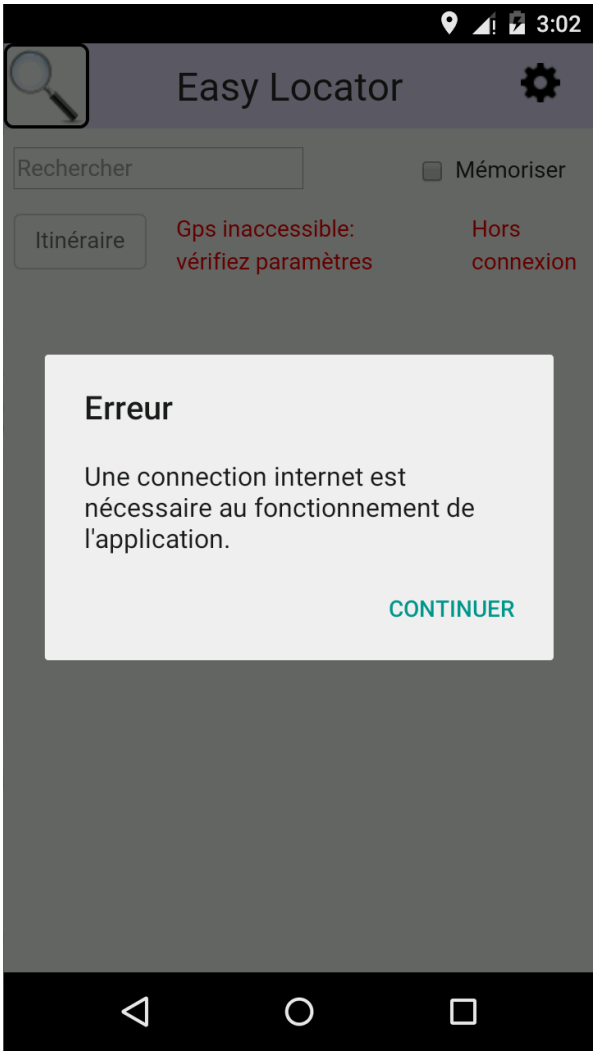

Figure 15 :
Message d’erreur style natif Android
(émulateur AVD – Android 5.1.1)

L'utilisateur peut également cliquer sur le symbole du menu (route dentée) qui lui propose la capture de photo. Cette dernière option présente les applications existantes permettant ce service ou lance l'application d'appareil photo par défaut (le comportement étant tributaire de la plateforme).

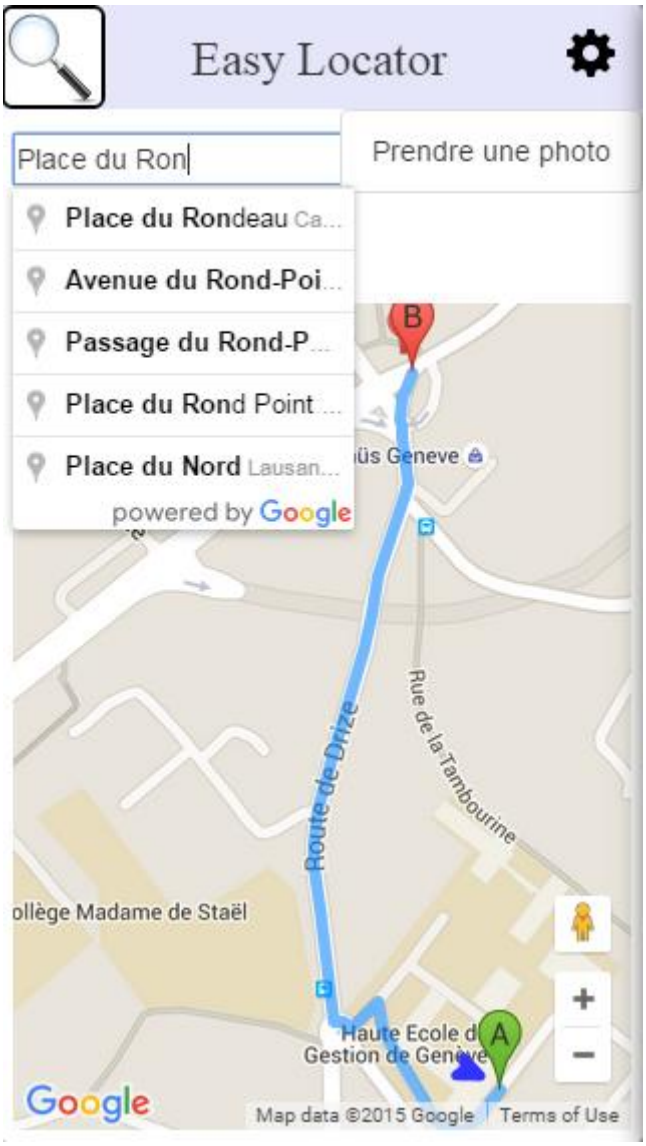

Figure 16 :
Itinéraire et option photo
(émulateur Apache Ripple – Android 4.1)

5.2.2 Structure du projet Apache Cordova

Une application hybride basée sur Apache Cordova étant fondamentalement une application web (exécutée dans une WebView), sa structure est proche de celle d'un projet web JavaScript basique avec l'addition des particularités nécessaires au framework pour la gestion des plugins, le support des différentes plateformes et la définition de l'application native enveloppante.

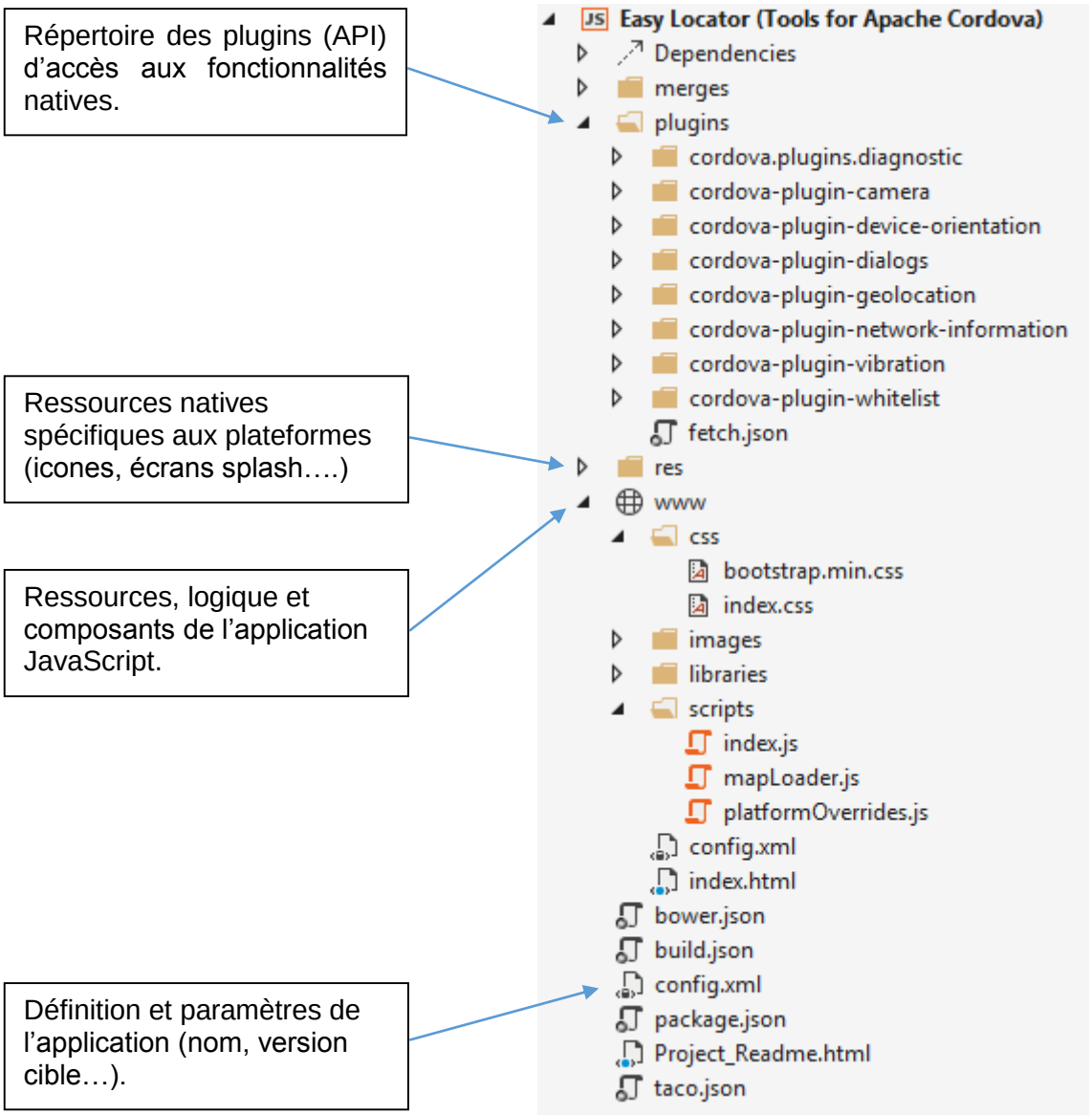

Figure 17 :
Structure d'un projet d'application Apache Cordova

Similairement aux différentes pages que contient un site ou une application internet, les différentes pages de l'application ainsi que leurs composants d'interface sont définis dans des fichiers HTML, le style ainsi que la disposition des éléments étant définis par des règles CSS.

Le ou les fichiers JavaScript comprenant la logique et les traitements de l'application sont référencés dans le fichier HTML principal puis chargés après chargement du DOM et de la libraire principale du noyau Apache Cordova.

Dans le cadre du prototype, l'application ne contient qu'une seule fenêtre, représentée par le fichier HTML principal index.html. Le style et la disposition sont définis par le fichier index.css tout en tirant parti du framework Bootstrap qui a également été utilisé dans ce projet, celui-ci étant populaire pour le développement d'applications web dédiées aux clients mobiles.

La logique de l'application, les traitements ainsi que l'utilisation des plugins, qui est détaillée au point suivant, sont regroupés dans le fichier index.js.

Un fichier de configuration global propre aux projets Apache Cordova, config.xml, regroupe les propriétés de l'application telles que nom, description, version et auteur et défini des paramètres généraux ou spécifiques aux plateformes comme l'orientation de la fenêtre ou le niveau minimum de version de l'OS cible. Ce fichier référence également les plugins utilisés par l'application[16].

5.2.3 Implémentation et utilisation des plugins

Apache Cordova fonctionnant sur deux bases de codes, native et JavaScript, le risque existe qu'au démarrage, l'application appel une fonction JavaScript de Cordova une fois le DOM chargé mais avant que le code natif correspondant soit disponible.

Afin de résoudre ce problème, l'API principale de Cordova lance l'évènement "deviceready" une fois son chargement ainsi que celui des plugins terminés : l'application doit donc typiquement inscrire un listener sur cet évènement en début de son script principal, le reste des traitements devant suivre dans une fonction callback (voir image suivante).

[16] L'intégralité du code des fichiers HTML, CSS et JavaScript mentionnés ainsi que du fichier de configuration est située en annexe à ce document. Le chapitre suivant ne contient que les parties essentielles rapportant à l'utilisation des plugins Cordova et non par exemple la logique ayant attrait à l'utilisation de l'API Google Maps.

Comme nous pouvons le voir également sur le code suivant, deux autres évènements principaux doivent être gérés une fois l'application chargée, "pause" et "resume", qui servent à identifier la mise en arrière-plan et la reprise de l'application par le système.

Cordova n'offre cependant pas la possibilité de détecter la fermeture du processus de l'application (telle qu'un appel sur la fonction onDestroy() d'une Activité principale Android par exemple).

```
(function () {
    "use strict";

    document.addEventListener('deviceready', onDeviceReady.bind(this), false);

    function onDeviceReady() {

        document.addEventListener('pause', onPause.bind(this), false);
        document.addEventListener('resume', onResume.bind(this), false);

        (Traitements JavaScript)

    };

    function onPause() {};
    function onResume() {};

} )();
```

Figure 18 : Apache Cordova – Cycle de vie de l'application

L'application prototype tirant parti d'un certain nombre de capteurs et fonctionnalités natives du smartphone, plusieurs plugins sont exploités par celle-ci. Apache Cordova propose généralement un plugin unique pour chaque fonction ou type de tâches spécifique[17].

L'application ayant besoin d'une connexion pour le chargement de l'API Google Maps, ainsi que pour le streaming constant des données de la carte et des directions, celle-ci est typiquement testée en premier lieu sur le prototype, soit juste après le chargement des API Cordova.

Pour ce faire, le plugin « cordova-plugin-network-information » est utilisé.

[17] L'ajout des plugins au projet n'est pas couvert dans ce chapitre, l'objectif n'étant pas une démonstration exhaustive du fonctionnement de la solution. Cela se fait cependant très facilement via la CLI de Cordova ou à travers Visual Studio. La documentation officielle explique les étapes à l'adresse suivante : http://cordova.apache.org/docs/en/5.0.0/guide_cli_index.md.html#The%20Command-Line%20Interface

Une fois installé, celui-ci offre la possibilité de connaître le type de la connexion lors d'un appel sur "navigator.connection.type". Différentes constantes sont retournées pour son identification telles "Connection.NONE", "Connection.CELL_4G" ou encore "Connection.WIFI".

Deux évènements, "offline" et "online", sont également disponibles et automatiquement lancés lors de la perte ou reprise de connexion.

Afin que l'utilisateur soit averti de la nécessité d'une connexion internet pour le fonctionnement de l'application, celle-ci doit lancer une notification, idéalement dans le style natif, ce qui est rendu possible très simplement grâce au plugin « cordova-plugin-dialogs ».

Un appel sur l'objet "navigator.notification" permet d'ouvrir différents types de fenêtres de notification telle qu'alerte, demande de confirmation et *prompt* ou également de beeper l'appareil.

```
function startupConnectionTest () {
    if (navigator.connection.type == Connection.NONE) {
        navigator.notification.alert(
                        'Une connection internet est nécessaire ' +
                        'au fonctionnement de l\'application.',  // message
                        badConnectionCallback,          // callback
                        'Erreur',               // title
                        'Continuer'             // buttonName
        );
    } else {
        $.getScript("https://maps.googleapis.com/maps/api/js?libraries=plac
    }
}
```

Figure 19 : Apache Cordova – Test de la connexion et alerte native

Un autre plugin « cordova-plugin-vibration » permet de contrôler la vibration du smartphone très facilement en fournissant simplement la durée voulue en millisecondes.

```
function addNetworkListeners() {
    document.addEventListener("offline", onOffline, false);
    document.addEventListener("online", onOnline, false);
}
function onOffline() {
    navigator.vibrate(100);
    document.getElementById("connectionState").style.visibility = "visible";
}
```

Figure 20 : Apache Cordova – Vibration et évènement de connexion

Le prototype a également besoin des informations GPS de l'appareil pour localiser l'utilisateur sur la carte, fonction qui est servie par le plugin « cordova-plugin-geolocation ».

Celui-ci offre principalement deux possibilités : soit interroger l'appareil pour obtenir un objet de type "Position" possédant de multiples attributs dont notamment la latitude et la longitude, soit de souscrire la fonction callback à ce même service, l'objet "Position" étant alors retourné régulièrement, à chaque modification de la position réelle du téléphone.

Des paramètres permettent de déterminer la durée de vie maximale des informations reçues ainsi que forcer le plugin à ne pas se baser sur la position approximative que le smartphone aurait éventuellement pu recevoir via le point d'accès internet.

```
gpsWatchId = navigator.geolocation.watchPosition(
                    gpsOn,
                    gpsOff,
                    { maximumAge: 5000, timeout: 20000, enableHighAccuracy: true });

function gpsOn(position) {
    curLat = position.coords.latitude;
    curLng = position.coords.longitude;
```

Figure 21 : Apache Cordova – GPS

En complément, le plugin « cordova-plugin-device-orientation » permet d'interroger le magnétomètre du smartphone (boussole) et de manière similaire au plugin GPS, il fournit à intervalle régulier (à définir en paramètre) un objet comprenant les données voulues. Celui-ci contient notamment l'orientation en degré relativement au Pôle Nord, qui sert à l'application pour indiquer à l'utilisateur sa direction sur la carte.

```
compassWatchId = navigator.compass.watchHeading(
                    compassSuccess,
                    compassError,
                    {frequency: 100});

function compassSuccess(heading) {
    if (mapIsLoaded) {
        gpsMarkerIcon.rotation = heading.magneticHeading;
        gpsMarker.set('icon', gpsMarkerIcon);
    }
}
```

Figure 22 : Apache Cordova – Boussole

Quant à l'utilisation de l'appareil photo, celle-ci se fait via un appel à l'application camera par défaut du smartphone.

Le plugin « cordova-plugin-camera » gère ce type d'appel : divers paramètres sont possibles tels que taille, qualité d'image, type de retour désiré (URI du fichier ou string base-64) ou encore sauvegarde automatique dans l'album photo. Le plugin permet également de rechercher et obtenir une photo déjà présente dans la galerie d'images de l'appareil.

```
navigator.camera.getPicture(onPictureSuccess, onPictureFail, {
    quality: 100,
    destinationType: Camera.DestinationType.FILE_URI,
    saveToPhotoAlbum: true
});
```

Figure 23 : Apache Cordova – Appareil photo

6. Comparaison et positionnement des solutions

En conclusion au test d'Apache Cordova sur le prototype développé, ainsi qu'aux recherches effectuées sur cette solution et ses deux concurrentes présentées dans ce travail, il m'apparait important de pouvoir maintenant situer chacune dans ses spécialités, ses possibilités, ainsi que vis-à-vis des deux autres.

Ce chapitre n'est pas à considérer comme un classement universel de la meilleure solution existante, car tout projet représente des besoins et attentes spécifiques. Un tel classement serait d'ailleurs injuste au vue de la démarche entreprise : une seule solution ayant été testée pratiquement, sans couvrir l'entièreté de ses possibilités, et toutes ne partageant pas forcément le même langage et donc la même cible de développeurs.

Il s'agit d'avantages, et c'est dans cette direction que la fin de ce travail a été pensée, de découvrir un état des lieux des possibilités de développement d'application mobile hybride et de pouvoir discerner d'éventuels critères de décision dans le choix d'une solution selon des attentes prédéfinies.

6.1 Fonctionnalités natives supportées

Les fonctionnalités typiques des plateformes mobiles telles que géolocalisation, appareil photo ou senseurs ont été regroupées dans un tableau situé à la page suivante. Celui-ci présente pour chacune son accessibilité selon la solution choisie d'après le code de couleur présenté ci-dessous.

Des clarifications s'avérant parfois nécessaires pour une bonne compréhension du degré de support de certaines fonctionnalités, celles-ci sont signalées par un astérisque (*****) et détaillées au point suivant, spécifiquement à chaque solution.

Fonctionnalité accessible facilement en utilisant les APIs / plugins officiels de la solution. Principales plateformes et versions concernées par la solution globalement supportées.

Fonctionnalité plus difficilement implémentable (par exemple nécessitant du code spécifique à chaque plateforme, des APIs tiers…) ou non supportée sur certaines des plateformes principales.

Fonctionnalité actuellement non accessible avec la solution.

Tableau 1 : Répartition des fonctionnalités natives supportées par solution

	GPS	Boussole	Accéléromètre	Appareil photo	Media (audio, vidéo)	Notifications, vibrations	Contacts	Stockage
Apache Cordova / Adobe PhoneGap			Plugin non fonctionnel sur certaines plateformes / versions *		Aucun plugin de lecture de vidéo			- LocalStorage - IndexedDB - WebSQL - SQLite (plugin tiers) *
Appcelerator Titanium Mobile						Différentes API selon la plateforme *		- app.properties - Database (SQLite) *
Xamarin *		Pas directement depuis Xamarin.Forms / Labs *			Pas de lecture vidéo depuis Xamarin.Forms / Labs *		Pas directement depuis Xamarin.Forms / Labs *	SQLite *

6.2 Remarques et conclusions spécifiques à chaque solution

6.2.1 Apache Cordova / Adobe PhoneGap

Remarques concernant le stockage local de données :

Les applications hybrides Cordova peuvent et doivent gérer la persistance des données de différentes manières, selon les besoins et les plateformes visées. Chacune de ces possibilités est généralement dépendante des API accessibles de manière standard à travers la WebView sur laquelle l'application est basée.

Le stockage local (LocalStorage ou encore Web Storage) permet, à travers le référencement de paires clé / valeur, un stockage simple et indépendant de la plateforme. A la manière des sessions ou cookies web, il est utile à la conservation de données peu conséquentes telles que paramètres utilisateurs, sa limite étant fixe à 5MB pour tous les navigateurs excepté le browser standard d'Android, fixant lui celle-ci à 2Mb.

Bien que sa spécification ait été délaissée par le Web Applications Working Group (W3C), l'implémentation du standard WebSQL sur la WebView des systèmes Android, BlackBerry 10, iOS et Tizen permet un stockage sur base de données SQL et représente la solution préconisée par Apache Cordova pour ces plateformes. Une taille limite de 50 MB pour la base est imposée par iOS, les autres systèmes permettant un stockage sans restriction de taille.

La WebView des systèmes Firefox OS, Windows 8 et Windows Phone 8 met elle aussi une API de stockage sur base de données à disposition, basée sur une implémentation de type IndexedDB. Celle-ci ne pose également pas de contraintes de taille.

Finalement, Cordova permettant l'accès au système de fichier, divers plugins tiers existent, créés par la communauté de développeur, qui offrent notamment le stockage sur base de données de type SQLite.

Remarques et impressions générales sur la solution :

Globalement, et après avoir pu tester celle-ci à travers le prototype développé, Apache Cordova constitue l'une des solutions les plus facilement accessibles : les plugins sont généralement simples d'utilisation et nécessitent très peu sinon aucun questionnement sur les spécificités des différentes plateformes visées.

En résultat de cette simplicité, si Cordova propose au total beaucoup moins de plugins que les deux autres solutions, les principales fonctionnalités mobiles sont tout de même bien supportées sur la plupart des plateformes. Les autres framework possédant souvent leurs propres éléments d'interface et de multiples API complémentaires visant plus à faciliter le développement (tel que gestion de collections, objets, requêtes Web…) ou à supporter des fonctions très spécifiques à certaines plateformes.

La documentation des plugins Cordova étant cependant également succincte, il peut être facile de bloquer sur un problème et rencontrer des difficultés à obtenir les informations nécessaires à sa résolution. Cette situation a typiquement été rencontrée avec l'application prototype où je ne pus déterminer précisément les raison du non fonctionnement de l'accéléromètre sur les plateformes de test à ma disposition.

De la même manière, la simplicité du framework et de ses plugins peut rendre difficile l'exploitation d'une fonctionnalité native de la manière désirée : cela fût le cas lors du développement du prototype, ne pouvant parfois pas faire la distinction avec certitude entre un signal GPS inaccessible et la fonction qui serait désactivée dans les options.

La performance générale d'une application hybride Cordova est tout à fait acceptable, même si la différence avec une application native standard, concernant la responsivité des éléments d'interface utilisateur, est perceptible dans l'application développée.

Ce framework est donc peu envisageable pour des applications nécessitant par exemple des traitements et rendus graphiques complexes, ce pour des raisons de performance.

En résumé, Apache Cordova propose des attraits certains pour un développement mobile multiplateforme JavaScript, visant une généricité de code maximale dans un environnement de développement déjà défini.

Il s'agit également d'une solution bien adaptée pour tout projet ciblant des fonctions natives répandues sans forcément vouloir exploiter en sus les particularités de certaines plateformes. Cette dernière option reste cependant tout à fait possible à travers le développement ou l'utilisation (selon existence) de plugins tiers.

6.2.2 Appcelerator Titanium Mobile

Titanium, bien que basé sur le même langage que Cordova (JavaScript), se positionne davantage comme solution de développement complète pour des plateformes spécifiques. Proposant son propre environnement de développement, le support de nombreuses spécificités Android ou iOS et une meilleure performance générale en raison de l'aspect partiellement compilé de son fonctionnement, la conséquence est cependant que la courbe d'apprentissage de ce framework est plus raide.

En résultat et d'après les recherches effectuées, Titanium permet d'exploiter plutôt facilement les principales spécificités mobiles, mais certaines peuvent exiger davantage de travail pour viser un large support multiplateforme. L'utilisation de multiples API aux fonctionnements hétérogènes peut par exemple être nécessaire pour supporter une fonctionnalité sur toutes les plateformes cibles, cela est notamment le cas pour l'implémentation des fonctions de notifications telles que vibrations de l'appareil.

Concernant la persistance des données, Titanium offre plusieurs API permettant soit un stockage "léger" (Titanium.app.properties), similaire au LocalStorage Cordova, soit un stockage sur base de données de type SQLite (Titanium.app.database), sans contrainte de taille.

Finalement, Titanium s'avère être un choix judicieux pour des projets nécessitant des fonctionnalités spécifiques ou une plus forte ressemblance aux applications natives du point de vue de l'interface utilisateur et des performances. Il s'agit cependant d'une solution de développement payante (depuis le printemps 20015) dès la publication d'une application.

6.2.3 Xamarin

De manière similaire à Titanium, Xamarin est une solution misant beaucoup sur le support étendu des spécificités des différentes plateformes, notamment à travers ses deux principales librairies Xamarin.iOS et Xamarin.Android.

La librairie Xamarin.Forms, qui fournit les outils de développement de l'interface utilisateur, est l'unique API à but réellement multiplateforme. Celle-ci étant assez récente (introduite en 2014) et n'ayant pas pour but d'exposer de manière uniforme les fonctions hardware (ce qui est le rôle de Xamarin.iOS et Xamarin.Android), l'exploitation des fonctionnalités mobiles sans passer par les API spécifiques peut donc se révéler compliquée.

La communauté étant cependant active, des addons tiers existent, tel que le projet Open Source Xamarin Forms Labs[18] visant à améliorer le support multiplateforme des fonctions mobiles depuis la librairie Xamarin.Forms, et rencontrent un certain succès.

Xamarin permet de gérer le stockage de données à travers une implémentation de base de données locale de type SQLite-net, virtuellement sans restriction de taille, tout comme le propose Appcelerator Titanium.

En conclusion, Xamarin se positionne très similairement à Titanium, bien que basé en C# et non JavaScript. Si un développement multiplateforme avec unique base de code est possiblement moins aisé avec Xamarin, celui-ci propose cependant une intégration dans Visual Studio, l'un des éditeurs les plus populaires et les recherches effectuées semblent indiquer un meilleur support de la plateforme Windows Phone.

[18] Ce projet est accessible sur son repository GitHub à l'adresse suivante : https://github.com/XLabs/Xamarin-Forms-Labs

7. Conclusion

Nous arrivons maintenant, avec cette conclusion, au terme de ce travail sur le développement mobile multiplateforme. Bien qu'il puisse sembler dommage que je n'ai présenté concrètement, à travers l'application prototype, qu'une seule des solutions les plus répandues mentionnées dans ce document, les recherches effectuées sur celles-ci et sur le sujet général tout au long de ce travail m'ont permis de vous présenter leurs principaux attraits et surtout, de tirer des informations sur le panorama actuel du développement « hybride ». Ce sont ces informations que je vous propose de découvrir désormais, à titre de conclusion et avec l'intention de répondre finalement à la question principale : Qu'en est-il aujourd'hui, des possibilités de développement mobile hybride, par rapport à la traditionnelle approche native ?

L'un des aspects principaux ressortant des recherches effectuées dans le cadre de ce travail est que le choix de solutions ou frameworks disponibles est vaste voir très vaste.

Trois solutions complètes parmi les plus populaires ont été présentées précédemment, ce n'en sont cependant que trois parmi des dizaines de possibilités. Du framework JavaScript à but mobile tel Sencha Touch, en passant par la plateforme dérivée (iFactr, exploitant à la fois Xamarin et PhoneGap) jusqu'à la solution complète autonome, comme par exemple Marmalade, basée sur C++, le choix est réellement large. Cela témoigne une certaine maturité du domaine, sans forcément signifier celle de toutes les solutions individuellement. En résultat, la situation actuelle présente de nombreux attraits mais aussi quelques pièges ou du moins considérations à ne pas négliger.

Du côté des avantages, le nombre d'acteurs et d'approches différentes existants répond directement à l'un des problèmes majeurs posés par le développement natif qui est la nécessité de maîtriser et donc de se former aux différents langages et environnements des plateformes cibles. Outre le très répandu JavaScript, de plus en plus de langages permettent aujourd'hui d'exploiter un framework de développement mobile hybride tel que C# ou C++. Alors qu'il y a quelques années, l'éventualité d'un développement mobile multiplateforme était souvent réservée à un type de projet bien défini, basé sur une application web uniquement, il est maintenant possible de partir de spécifications purement natives tout en imaginant un développement hybride, moyennant le choix de la bonne solution pour les plateformes visées.

Cela nous amène au principal "piège" que peut aisément devenir justement, la décision d'opter pour une solution plutôt qu'une autre, si les attentes sur celle-ci ainsi que sur l'application envisagée n'ont pas été clairement définies ou prises en comptes.

Le chapitre précédent présente des pistes d'aide au choix de trois différents frameworks en se basant sur leurs principales caractéristiques, forces ou faiblesses, qui ont pu être identifiées et se base surtout sur le support de fonctionnalités mobiles spécifiques. Il ne tient pas compte de bien d'autres aspects, comme par exemple de contraintes de suivi de développement ou d'application de patterns particuliers voir même de distribution, tous des aspects non directement concernés par le thème principal de ce travail mais bien évidemment pertinents pour la décision à prendre.

De plus, la relative jeunesse de certaines solutions ainsi que la forte rapidité d'évolution de ce qui est devenu désormais un marché du développement hybride, ajoute à l'importance de la bonne définition des priorités. En effet, le manque de documentation ou de références sur un sujet qui n'est pas idéalement supporté par la solution choisie peut rapidement s'avérer problématique. De la même manière, le temps d'apprentissage des différents frameworks s'avère très uniforme et constitue un point à ne pas négliger.

Une autre conséquence de l'ampleur qu'a prise le marché du développement hybride est que de nombreuses solutions sont désormais payantes, ce qui peut rendre le choix d'une solution hybride plus compliquée pour de petites équipes ou des développeurs indépendants.

En point final, le développement d'applications mobiles hybrides est devenu, à mon avis, une alternative très sérieuse voir un idéal remplaçant du développement natif, selon les cas envisagés. Comme nous avons pu le voir précédemment, le support de spécificités typiquement mobiles est globalement bon : les rares fonctions natives non ou difficilement supportée sont très souvent accessibles à travers des API tiers, celles-ci pouvant être crées soit même le cas échéant.

Les différentes possibilités permettent de répondre à de nombreux scénarios, qu'il s'agisse d'unicité de code maximale ou de l'exploitation de fonctionnements très spécifiques à certaines plateformes, même s'il faut admettre qu'aucune solution n'offre le meilleur des deux mondes. Même la performance, principale critique faite aux applications hybrides, est généralement bonne et souvent l'un des points en constante amélioration par les différents acteurs. Le développement de jeux vidéo est même devenu tout à fait réaliste avec l'apparition de solutions basées sur des approches de code compilé, telle que Marmalade, compilant directement en code machine.

Bibliographie

Les citations présentes dans ce document proviennent des sources suivantes :

BEST, Jo, *Android before Android: the long strange history of Symbian and why it matters for Nokias future*, [en ligne], 4 Avril 2013. [Consulté le 23 juin 2015]. Disponible à l'adresse : http://www.zdnet.com/article/android-before-android-the-long-strange-history-of-symbian-and-why-it-matters-for-nokias-future/

Les sources ci-dessous ont également été utilisées pour la réalisation de ce travail et de l'application prototype associée:

History of mobile phones, [en ligne], [Consulté le 18 juin 2015]. Disponible à l'adresse : https://en.wikipedia.org/wiki/History_of_mobile_phones

Psion Series 3, [en ligne], [Consulté le 22 juin 2015]. Disponible à l'adresse : https://en.wikipedia.org/wiki/Psion_Series_3

EDWARDS, Leigh, *Programming Psion Computers*, [en ligne], Octobre 1997. [Consulté le 22 juin 2015]. Disponible à l'adresse : http://www.tobidog.com/ppc.pdf

LAUGESEN, John & YUAN, Yufei, *What factors contributed to the success of Appple's iPhone ?*, [en ligne], 2010. [Consulté le 24 juin 2015]. Disponible à l'adresse : http://www.msitmonline.com/media/Cin/compre/1_What%20Factors%20Contributed%20to%20the%20Success%20of%20Apple_s%20iPhone.pdf

O'GRADY, J. D., *iPhone SDK downloads surpass 100,000*, [en ligne], 12 Mars 2008. [Consulté le 24 juin 2015]. Disponible à l'adresse : http://www.zdnet.com/article/iphone-sdk-downloads-surpass-100000/

Ipsos OTX MediaCT, *Our Mobile Planet: Switzerland | Understanding the Mobile Consumer*, [en ligne], Mai 2012. [Consulté le 24 juin 2015]. Disponible à l'adresse : http://services.google.com/fh/files/blogs/our_mobile_—planet_switzerland_en.pdf

KHALAF, Simon, *Apps solidify leadership six years into the mobile revolution*, [en ligne], 1 Avril 2014. [Consulté le 23 juillet 2015]. Disponible à l'adresse : http://flurrymobile.tumblr.com/post/115191864580/apps-solidify-leadership-six-years-into-the-mobile

AVRAM, Abel, *Facebook: "Betting on HTML5 was a mistake"*, [en ligne], Septembre 2014. [Consulté le 23 juillet 2015]. Disponible à l'adresse : http://www.infoq.com/news/2012/09/Facebook-HTML5-Native

World Wide Web Consortium, *Technical Reports*, Divers rapports [en ligne], [Consultés en Août 2015], Disponibles à l'adresse : http://www.w3.org/

Mobile HTML5, [en ligne], [Consulté le 4 Août 2015], Disponible à l'adresse : http://mobilehtml5.org/

HAYNIE, Jeff, *StackOverflow,* [en ligne]. 18 Mars 2010, [Consulté le 8 Août 2015]. Disponible à l'adresse : http://stackoverflow.com/questions/2444001/how-does-appcelerator-titanium-mobile-work/2471774#2471774

Apache Cordova, *Documentation v5.0.0*, [en ligne], [Consultée en juillet/Août/Septembre 2015]. Disponible à l'adresse : http://cordova.apache.org/docs/en/5.0.0/

Google Maps Javascript API v3, *Documentation*, [en ligne], [Consultée en Août / Septembre 2015]. Disponible à l'adresse : https://developers.google.com/maps/documentation/javascript/

W3schools, *Bootstrap 3 Tutorial*, [en ligne], [Consultée en Août/Septembre 2015]. Disponible à l'adresse : http://www.w3schools.com/bootstrap/

Appcelerator Titanium, *Documentation*, [en ligne], [Consultée en juillet/Août/Septembre 2015]. Disponible à l'adresse : https://docs.appcelerator.com/platform/latest/#!/api

Xamarin, *Documentation*, [en ligne], [Consultée en juillet/Août/Septembre 2015]. Disponible à l'adresse : https://developer.xamarin.com/api/

KITAMURA, Eiji, *Working with quota on mobile browsers*, [en ligne], 28 Janvier 2014. [Consulté le 12 Septembre 2015]. Disponible à l'adresse : http://www.infoq.com/news/2012/09/Facebook-HTML5-Native

Xlabs / Xamarin-Forms-Labs, *Extention Xamarin*, [en ligne], [Consulté en Septembre 2015]. Disponible à l'adresse : https://github.com/XLabs/Xamarin-Forms-Labs

Annexe 1 : Fenêtre de l'application – index.html

```
<!DOCTYPE html>
<html ng-app="mapApp">
<head>
    <meta charset="utf-8" />
    <meta name="description" content="Apache Cordova discovery and testing app">
    <meta name="author" content="Andy Christen">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!--
        Customize the content security policy in the meta tag below as needed.
        Add 'unsafe-inline' to default-src to enable inline JavaScript.
        For details, see http://go.microsoft.com/fwlink/?LinkID=617521
    -->
    <meta http-equiv="Content-Security-Policy" content="
          default-src 'self' data: gap:
                http://ssl.gstatic.com http://csi.gstatic.com
http://*.googleapis.com
                http://*.gstatic.com https://*.googleapis.com
https://*.gstatic.com 'unsafe-eval';
          script-src 'self' 'unsafe-eval' 'unsafe-inline'
                 https://*.googleapis.com  https://*.gstatic.com;
          style-src 'self' 'unsafe-inline'
                https://*.googleapis.com https://*.gstatic.com;
          font-src
                'self' https://*.googleapis.com https://*.gstatic.com;
    ">

    <!-- EasyLocator references -->
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/index.css">

    <title>EasyLocator</title>

</head>
<body>
    <div class="container-fluid">

        <div id="menu-options">
            <ul class="list-group">
                <li class="list-group-item">Prendre une photo</li>
            </ul>
        </div>
        <!-- App-icon / Title / Options-icon -->
        <div class="row header-size">
            <div class="col-xs-3 header-size pull-left" style="background-
                                                    color:lavender;">
                <img class="img-rounded header-icon"
                    src="images/mdpi_app-logo.png" alt="app_logo">
            </div>
            <div class="col-xs-7 header-size" style="background-color:lavender;">
                <h3 class="header-title-text">Easy Locator</h3>
            </div>
            <div class="col-xs-2 header-size pull-right" style="background-
                                                     color:lavender;">
                <img id="options-icon" src="images/glyphicons-137-cogwheel.png">
            </div>
        </div>
```

```
        <!-- Search-field / Checkbox -->
        <div class="row body-row">
            <div class="col-xs-8 search-line" id="searchfield-container">
                    <input id="pac-input" class="controls" type="text"
                           placeholder="Rechercher">
            </div>
            <div class="checkbox col-xs-4" id="checkbox-storage-div">
                <label><input id="checkbox-storage"
                              type="checkbox">Mémoriser</label>
            </div>
        </div>
        <!-- Button / Textfields / Map-canvas -->
        <div class="row body-row">
            <div class="col-xs-3 search-line">
                <button class="btn btn-default"
                        id="directionsBtn">Itinéraire</button>
            </div>
            <div class="col-xs-6">
                <p id="gpsState">Gps inaccessible: vérifiez paramètres</p>
            </div>
            <div class="col-xs-3">
                <p id="connectionState">Hors connexion</p>
            </div>
            <div class="map-parent">
                <div id="map"></div>
            </div>
        </div>
    </div>

    <!-- Cordova reference, this is added to your app when it's built. -->
    <script src="cordova.js"></script>
    <script src="scripts/platformOverrides.js"></script>

    <script src="libraries/jquery-2.1.4.js"></script>
    <script src="libraries/bootstrap.min.js"></script>

    <script src="scripts/index.js"></script>

</body>

</html>
```

Annexe 2 : Feuille de style – index.css

```
body, html, .container-fluid {
    height: 100%;
}

.header-size {
    height: 52px;
}
.header-icon {
    position: absolute;
    left: 0px;
    border:2px solid #000000;
}
.header-title-text {
    font: 24px bold;
    position: relative;
    top: 35%;
    transform: translateY(-90%);
}
#options-icon {
    position: absolute;
    top: 12px;
}

.body-row {
    margin-top: 6px;
}
.search-line {
    padding-left: 6px;
}

#searchfield-container {
    margin-top: 6px;
}

#gpsState, #connectionState {
    visibility: hidden;
    color: #FF0000;
}
.map-parent {
    position: absolute;
    top:143px;
    left:0px;
    right:0px;
    bottom:0px;
    margin: 6px;
}
#map {
   height: 100%;
}

#menu-options {
    position: fixed;
    top: 52px;
    right: 0;
    z-index: 1;
    display: none;

}
```

Annexe 3 : Script de l'application – index.js

Cette application étant basée sur plusieurs plugins Apache Cordova, il est important de contrôler leur présence dans votre projet en cas de tentative de reproduction de tout ou partie du prototype. Ceux-ci sont listés sur la figure 17 (page 33).

```
/*************************************************************

    * Easy Locator - Apache Cordova discovery and testing app
    *
    * Author:  Andy Christen

*************************************************************/

var initMap;

(function () {
    "use strict";

    document.addEventListener('deviceready', onDeviceReady.bind(this), false);

    // Device & UI state variables.
    var netStatus, gpsWatchId, compassWatchId, compassCompatCheck;
    // Directions and map services variables.
    var map, mapIsLoaded, dirService, dirDisplay, curLat, curLng;
    var targetLatLng = null, gpsMarker, gpsMarkerIcon, moveToMarker;

    function onDeviceReady() {

        // Handles the Cordova pause and resume events
        document.addEventListener('pause', onPause.bind(this), false);
        document.addEventListener('resume', onResume.bind(this), false);

        // TODO: Cordova has been loaded.
        // Perform any initialization that requires Cordova here.
        mapIsLoaded = false; compassCompatCheck = true;

        // Checks local storage if search string needs to be restored.
        storedValuesTest();

        startupConnectionTest();

        // Sensors and network listeners
        addNetworkListeners();
        gpsWatchId = navigator.geolocation.watchPosition(gpsOn, gpsOff, {
                                maximumAge: 5000, timeout: 20000,
                                enableHighAccuracy: true });
        compassWatchId = navigator.compass.watchHeading(compassSuccess,
                                compassError, { frequency: 100 });
        // Motion detection  based on cordova.plugin.device-motion  and
           leecrossley's shake plugin
        // -- not working -- apparently not supported on some devices and Ripple
           emulators, unable to test
        // shake.startWatch(onShakeSuccess, 30 , onShakeError);

        // Binds physical menu button (deprecated) & options icon to options menu
           div visibility.
        document.addEventListener("menubutton", menuToggle, false);
        $('#options-icon').on('click', menuToggle);
```

```
// Stores searchBox String locally on key events.
$('#pac-input').on('keyup', function () {
    localStorage.setItem("searchString", $('#pac-input').val());
});

// Stores "Mémoriser" checkbox state locally for startup check.
$('#checkbox-storage').change(function () {
    if (this.checked) {
        localStorage.setItem("storageEnabled", true);
    } else {
        localStorage.setItem("storageEnabled", false);
    }
});

// Binds "Itinéraire" button to directions service.
$('#directionsBtn').on('click', function () {
    if (targetLatLng != null) { calculateAndDisplayRoute(dirService,
                                    dirDisplay); }
    else {
        navigator.notification.alert(
                        'Aucune destination existante disponible.',
                        function () { },         // callback
                        'Erreur',               // title
                        'Continuer'             // buttonName
        );
    }
});

// Binds menu item "Prendre une photo" to camera plugin service
// (takes a picture with native camera app).
$(".list-group-item").on("click", function () {
    navigator.camera.getPicture(onPictureSuccess, onPictureFail, {
        quality: 100,
        destinationType: Camera.DestinationType.FILE_URI,
        saveToPhotoAlbum: true
    });
});


// Handles search string  value restoration (called once at startup)
function storedValuesTest() {
    var storageEnabled = localStorage.getItem("storageEnabled");
    if ((storageEnabled === null) || (storageEnabled == "false")) {
        localStorage.setItem("storageEnabled", false);
        localStorage.setItem("searchString", "");
    } else {
        $('#pac-input').val(localStorage.getItem("searchString"));
        $('#checkbox-storage').prop('checked', true);
    }
}

// Tests connection and load scripts if available (called once at
   startup)
function startupConnectionTest() {
    if (navigator.connection.type == Connection.NONE) {
        navigator.notification.alert(
            'Une connection internet est nécessaire au fonctionnement de
            l\'application.',
            function () { },         // callback
            'Erreur',               // title
            'Continuer'             // buttonName
        );
    } else {
```

```
$.getScript("https://maps.googleapis.com/maps/api/js?libraries=places&key=AIzaSyD
9ajxdXKQoDbLAV7HFHL2PLVxMTGB0RmI&callback=initMap", function () { });
            }
        }

        // Handle changes in connection state events (going offline/online)
        function addNetworkListeners() {
            document.addEventListener("offline", onOffline, false);
            document.addEventListener("online", onOnline, false);
        }
        function onOffline() {
            navigator.vibrate(100);
            document.getElementById("connectionState").style.visibility =
            "visible";
        }
        function onOnline() {
            navigator.vibrate(100);
            document.getElementById("connectionState").style.visibility =
            "hidden";
            if (mapIsLoaded) {
                google.maps.event.trigger(map, 'resize');
            } else {

$.getScript("https://maps.googleapis.com/maps/api/js?libraries=places&key=AIzaSyD
9ajxdXKQoDbLAV7HFHL2PLVxMTGB0RmI&callback=initMap", function () { });
            }
        }

        // Handles gps callbacks with map centering option.
        function gpsOn(position) {
            curLat = position.coords.latitude;
            curLng = position.coords.longitude;
            document.getElementById("directionsBtn").disabled = false;
            document.getElementById("gpsState").style.visibility = "hidden";
            if (mapIsLoaded) {
                // Centers the map on current location at startup and resume.
                if (moveToMarker) { map.setCenter({ lat: curLat, lng: curLng });
                                    moveToMarker = false; }
                gpsMarker.setPosition({ lat: curLat, lng: curLng });

            }
        }
        function gpsOff(error) {
            document.getElementById("directionsBtn").disabled = true;
            document.getElementById("gpsState").style.visibility = "visible";
        }

        // Handles device compass callbacks for updating the location marker
           orientation.
        function compassSuccess(heading) {
            if (mapIsLoaded) {
                gpsMarkerIcon.rotation = heading.magneticHeading;
                gpsMarker.set('icon', gpsMarkerIcon);
            }
        }
```

```
function compassError(compassError) {
    if (compassCompatCheck) {
        if ((compassError.code == 20) || (compassError.code == 3)) {
            navigator.notification.alert(
                        'Fonctionnalité boussole incompatible avec votre
                          appareil.',  // message
                        function () { },         // callback
                        'Erreur',               // title
                        'Continuer'             // buttonName
        );
            compassCompatCheck = false;
        }
    }
}

// Handles menu visibility changes
function menuToggle() {
    $('#menu-options').toggle();
}

// Handles device shaking callbacks for clearing search textbox.
function onShakeSuccess() {
    var directionSearchBox = document.getElementById('pac-input');
    directionSearchBox.focus();
    directionSearchBox.innerHTML = "";
}
function onShakeError() {
    // alert('shake error');
}

// Displays itinerary on the map.
function calculateAndDisplayRoute(directionsService, directionsDisplay) {
    directionsService.route({
        origin: curLat + "," + curLng,
        destination: targetLatLng,
        travelMode: google.maps.TravelMode.WALKING
    }, function (response, status) {
        if (status === google.maps.DirectionsStatus.OK) {
            directionsDisplay.setDirections(response);
        } else {
            navigator.notification.alert(
                        'La requête de direction a échoué: ' + status,
                        badConnectionCallback,         // callback
                        'Erreur',               // title
                        'Continuer'             // buttonName
            );
        }
    });
}

// Handles picture App callback with Uri of created file.
function onPictureSuccess(imageUri) {
    navigator.notification.alert(
                        'Photo enregistrée sous : ' + imageUri,
                        function () { },         // callback
                        'Capture d\'image',               // title
                        'Continuer'             // buttonName
    );

}
```

```
function onPictureFail(message) {
    navigator.notification.alert(
                        'La capture d\'image n\'a pas réussi : ' +
                         message,  // message
                        function () { },         // callback
                        'Erreur',               // title
                        'Continuer'             // buttonName
        );
}


// Handles the map and binds related DOM elements
// (searchBox \ directions button).
initMap = function () {
    var geneve = { lat: 46.2000, lng: 6.1500 };
    // Directions services
    dirService = new google.maps.DirectionsService;
    dirDisplay = new google.maps.DirectionsRenderer;
    // Loads the map instance
    map = new google.maps.Map(document.getElementById('map'), {
        center: geneve,
        zoom: 11
    });
    dirDisplay.setMap(map);
    // Adds the location marker with Geneva as default value.
    gpsMarker = new google.maps.Marker({
        position: geneve,
        map: map,
        title: 'Ma position',
        icon: {
            path: google.maps.SymbolPath.FORWARD_CLOSED_ARROW,
            strokeColor: '#3333FF',
            strokeWeight: 5,
            scale: 2.5
        },
    });
    gpsMarkerIcon = gpsMarker.get('icon');
    mapIsLoaded = true;
    moveToMarker = true;

    // Creates the search box and links it to the UI element.
    var input = document.getElementById('pac-input');
    var searchBox = new google.maps.places.SearchBox(input);

    // Bias the SearchBox results towards current map's viewport.
    map.addListener('bounds_changed', function () {
        searchBox.setBounds(map.getBounds());
    });

    var markers = [];

    // [START region_getplaces]
    // Listens for the event fired when the user selects a prediction and
       retrieves
    // more details for that place.
    searchBox.addListener('places_changed', function () {
        var places = searchBox.getPlaces();

        if (places.length == 0) {
            return;
        }
```

```
                // Cleares out the old markers.
                markers.forEach(function (marker) {
                    marker.setMap(null);
                });
                markers = [];

                // For each place, gets the icon, name and location.
                var bounds = new google.maps.LatLngBounds();
                places.forEach(function (place) {
                    var icon = {
                        url: place.icon,
                        size: new google.maps.Size(71, 71),
                        origin: new google.maps.Point(0, 0),
                        anchor: new google.maps.Point(17, 34),
                        scaledSize: new google.maps.Size(25, 25)
                    };

                    // Stores the place location for directions usage.
                    targetLatLng = place.geometry.location;

                    // Creates a marker for each place.
                    markers.push(new google.maps.Marker({
                        map: map,
                        icon: icon,
                        title: place.name,
                        position: place.geometry.location
                    }));

                    if (place.geometry.viewport) {
                        // Only geocodes have viewport.
                        bounds.union(place.geometry.viewport);
                    } else {
                        bounds.extend(place.geometry.location);
                    }
                });
                map.fitBounds(bounds);
            });
            // [END region_getplaces]

        }
    };


    function onPause() {
        navigator.geolocation.clearWatch(gpsWatchId);
        navigator.compass.clearWatch(compassWatchId);
        localStorage.setItem("searchString", $('#pac-input').val());
    };

    function onResume() {
        moveToMarker = true;
        gpsWatchId = navigator.geolocation.watchPosition(gpsOn, gpsOff, {
                                maximumAge: 5000, timeout: 20000,
                                enableHighAccuracy: true });
        compassWatchId = navigator.compass.watchHeading(compassSuccess,
                                compassError, { frequency: 100 });
    };


} )();
```