LOUVAIN
School of Management

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

LOUVAIN SCHOOL OF MANAGEMENT

Le paradigme « Write once, run everywhere » est-il l'avenir du développement mobile ?

Promoteur : **Jean Vanderdonckt**

Mémoire-recherche présenté par
Thibaut Bombey en vue de l'obtention du titre de
Master en sciences de gestion

ANNÉE ACADÉMIQUE 2012-2013

REMERCIEMENTS

L'élaboration de ce mémoire m'a forcé à me repousser dans mes limites. Malgré un sujet peu adapté à mon master en sciences de gestion, de nombreuses personnes m'ont poussé à persévérer dans mes recherches.

Tout d'abord, je remercie mon promoteur, Monsieur Jean Vanderdonckt, pour son implication dans mon travail de recherche. Ses conseils et orientations m'ont permis d'obtenir un travail complet et innovant.

Je remercie également, tout particulièrement, les personnes qui ont accepté de m'aider lorsque j'en avais le plus besoin et sans lesquelles cette étude n'aurait pas été possible. Obtenir différents points de vue m'a permis d'élaborer plus aisément certains chapitres.

Finalement, je souhaite conclure sur un élément plus personnel et remercier ma famille, ainsi que ma petite amie et mes amis pour m'avoir soutenu et remonté le moral dans les étapes plus difficiles.

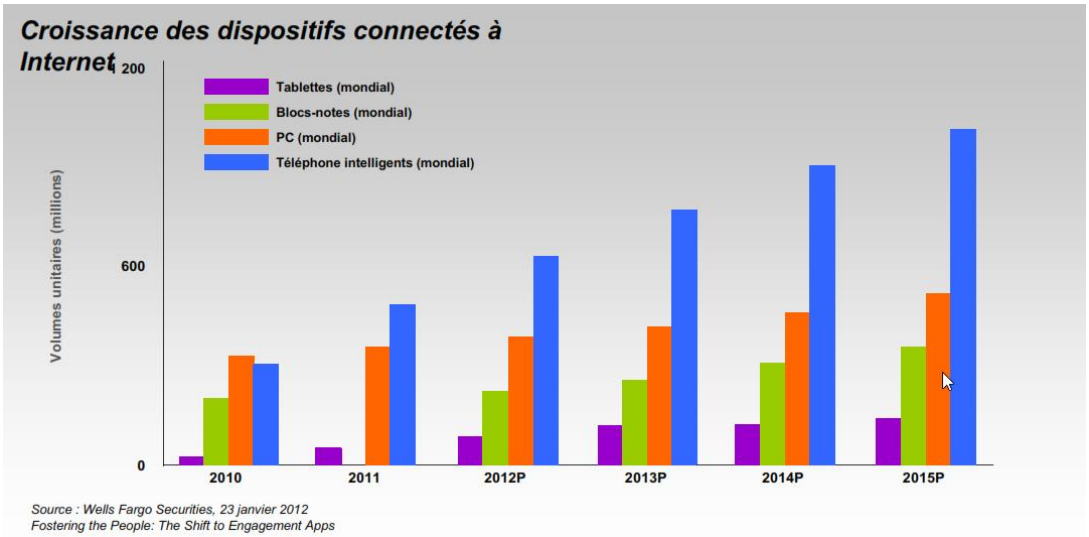
Table des matières

1 Introduction

1.1 Contexte de la recherche

Depuis le lancement du premier téléphone intelligent d'Apple en 2007, une véritable révolution mobile s'est mise en place. Qu'on le veuille ou non, cet évènement a radicalement changé notre quotidien. Comme nous le montre le graphique 1, la croissance de ce type d'appareil est tellement forte que ces téléphones intelligents (smartphones) ont dépassé le nombre d'ordinateurs classiques connectés à internet.

Graphique 1 - Croissance des dispositifs connectés à internet de 2010 à 2015



Source : Wells Fargo Securities (2012)

La principale raison de cette révolution peut être expliquée par le célèbre slogan d'Apple « There's an app for that ». Celui-ci relate le fait qu'il existe une application pour, à peu près tout. En effet, le nombre d'applications destinées aux appareils mobiles n'a cessé d'augmenter de manière exponentielle depuis ces cinq dernières années. Si l'on en croit les statistiques du cabinet Canalys, le nombre d'applications téléchargées au premier quadrimestre 2013 serait de 13,4 milliards, soit 150 millions par jour. Dès lors, il est normal de penser que de nombreux acteurs vont continuer à se concentrer sur ce marché en pleine expansion.

Cette forte croissance a créé de nouvelles opportunités pour de nombreux secteurs puisque ce type d'application est capable de trouver un intérêt partout.

Que ce soit dans le domaine de la publicité, des réseaux sociaux, des loisirs, des jeux ou de la productivité, il est plus que propice de créer une application compatible sur les appareils mobiles.

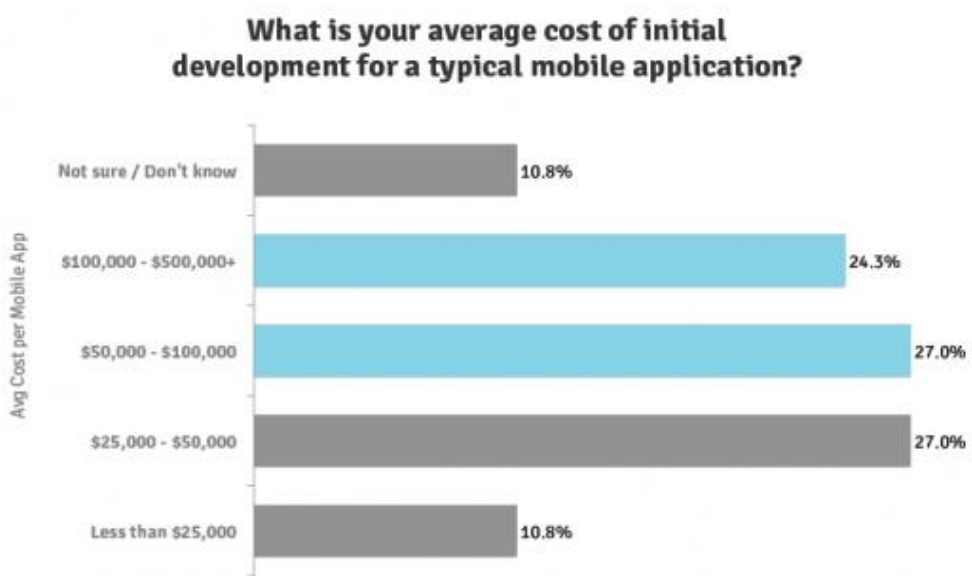
Par ailleurs, ce nouveau marché contente un groupe d'acteurs particuliers : les développeurs informatiques. Ceux-ci voient leur offre de travail augmenter d'une manière considérable. Le développement d'applications et de sites Web n'est plus uniquement destiné aux ordinateurs classiques, mais également à ce nouvel univers mobile. Ils disposent donc de la possibilité de se diversifier ou de se spécialiser dans la création d'applications mobiles.

Pour ce faire, le développeur doit faire face aux obstacles que le monde mobile engendre. Effectivement, il faut prendre en considération les différents facteurs liés à ces nouveaux appareils. Il est dorénavant primordial de faire attention à ces appareils de taille réduite, disposant de divers capteurs et d'une connexion internet limitée. Mises à part ces différences liées aux facteurs mobiles de ces téléphones intelligents, la plus grande difficulté est le fait que ce marché est extrêmement fragmenté. Il existe de nombreux systèmes d'exploitation mobiles ainsi que des centaines d'appareils différents. Ceux-ci peuvent énormément varier sur de nombreux critères.

Cette hétérogénéité est le challenge le plus complexe lors de la création d'applications mobiles. L'industrie des logiciels mobiles a besoin d'une technologie permettant la création d'application complexe à l'aide d'un seul code compatible sur les différentes plateformes mobiles (Miravet & al. 2009). Prenons l'exemple d'une entreprise désirant déployer son application sur les appareils Apple, Windows et Google. Celle-ci doit engager un développeur pouvant coder l'application à l'aide de trois langages différents et donc, réaliser trois fois le même travail. Elle devra également tester l'application sur des centaines d'appareils différents afin de la rendre complètement compatible avec les plateformes visées.

D'après une étude de l'entreprise AnyPresence, 51,3 % des entreprises débourseraient plus de 50,000 $ par application mobile. Cette forte fragmentation représente donc un challenge crucial pour les entreprises devant réitérer l'opération sur chaque système d'exploitation désiré. Une solution permettant de réduire ces coûts serait de réussir à développer une application directement compatible sur les différents supports. Cependant il n'y a encore aucune solution permettant d'égaler le développement individuel d'application pour chaque plateforme.

Graphique 2 – Coût moyen de développement pour une application mobile classique



Source : ANYPRESENCE (2013) The state of enterprise mobile readiness 2013

1.2 Définition de notre objet d'étude

À travers ce mémoire, nous allons tenter de mieux comprendre les raisons de cette fragmentation afin d'analyser les différentes solutions réalisables permettant au développeur de passer outre cet obstacle.

Notre question de recherche s'exprime donc comme suit : *« Le paradigme "Write once, run everywhere" est-il l'avenir du développement mobile ? »*

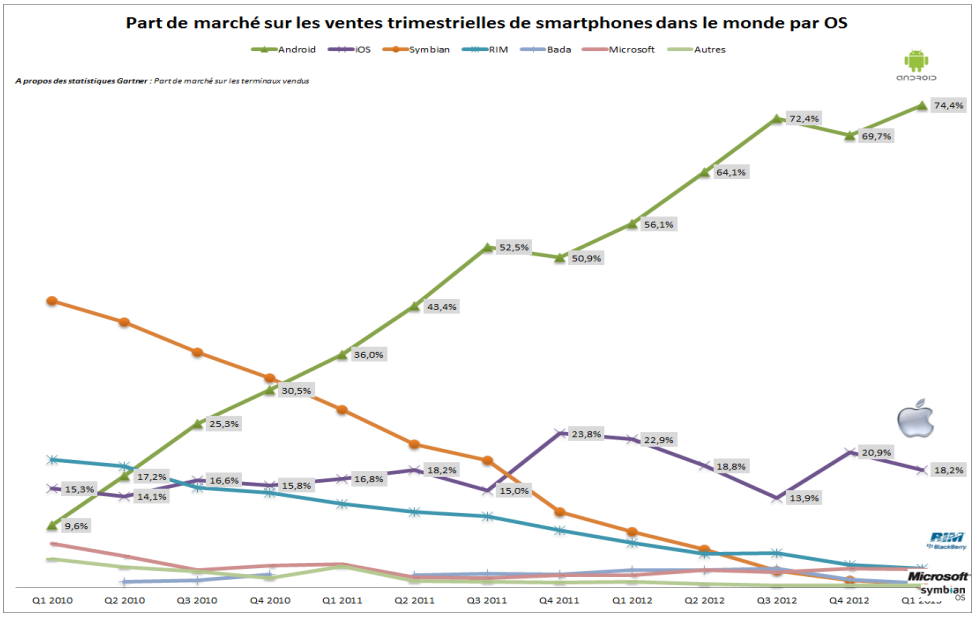
Tout d'abord, le paradigme *« Write once, run everywhere »* est un slogan lancé par Sun Microsystems. Il était destiné au célèbre langage informatique JAVA. Ils désiraient illustrer le fait que l'on puisse développer son application à l'aide de ce langage pour qu'il soit ensuite possible de l'exécuter sur tous les appareils disposant de la machine virtuelle Java. En d'autres mots, une fois l'application développée, elle serait capable de fonctionner, sans modifications, sur un smartphone, un ordinateur ou une tablette.

Aujourd'hui, ce slogan est devenu un paradigme explicitant la possibilité d'écrire une application à l'aide d'un code unique pour la déployer sur tous les appareils disparates (Balkan, 2012).

Ce paradigme symbolise un idéal dans le monde du développement. Il représente un gain énorme en temps, en coût, en investissement et réduit également le nombre de compétences requises. Dans notre recherche, nous allons donc analyser les méthodes se rapprochant ou non de ce paradigme afin d'essayer de déterminer s'il s'agit d'un mythe ou d'une destinée.

Pour ce faire, nous allons émettre différentes suppositions. Premièrement, nous allons appliquer cette recherche en fonction des quatre systèmes d'exploitation les plus présents dans le secteur des smartphones (voir annexe 1). Comme on peut l'observer sur le graphique 2, Android, iOS, RIM et Microsoft sont les systèmes d'exploitation disposant de plus de 98 % du marché. Notre question de recherche ne s'étendra donc pas au-delà de ceux-ci. Deuxièmement, nous allons concentrer nos recherches sur les différentes façons de créer une application mobile. Celles-ci seront jugées en fonction de diverses comparaisons et études telles que l'étude de VisionMobile qui a interrogé plus de 2400 développeurs en 2012.

Graphique 3 – Part de marché sur les ventes trimestrielles de smartphones dans le monde par système d'exploitation de 2010 à 2012



Source : Gartner (mai 2013)

Les critères que nous allons prendre en considération sont axés tant sur la facilité de développement que sur l'expérience de l'utilisateur. Ces critères ont été choisis puisqu'ils définissent parfaitement ce que l'on cherche. C'est-à-dire que le but de cette recherche est de trouver une solution facilitant le développement d'applications mobiles sans perdre, pour autant, le confort d'utilisation du consommateur final.

1.3 Structure du mémoire

Notre recherche sera divisée en plusieurs parties permettant une approche progressive.

Dans un premier temps, nous allons en apprendre un peu plus sur ce marché en pleine expansion afin d'obtenir une vision claire et précise tout au long de la recherche. Pour ce faire, nous commencerons par définir les différentes notions et défis présents dans le monde du développement mobile.

Ensuite, nous prendrons le temps d'analyser minutieusement les différents systèmes d'exploitation mobiles repris dans notre recherche. Cette observation nous permettra d'identifier les principales différences entre ces plateformes, mais également de mieux comprendre les difficultés du développement mobile.

Nous examinerons aussi les différents moyens de développement mobile dans leur fonctionnement général. Ce chapitre nous démontrera les avantages et inconvénients de chaque méthode de développement mobile. Nous pourrons en déduire les solutions se rapprochant de notre paradigme et sélectionner ainsi différentes méthodes à analyser plus en profondeur.

D'autre part, nous analyserons un échantillon d'outils prônant au mieux notre paradigme. En les examinant, nous serons capables de conclure différentes choses sur le fonctionnement et leurs réponses quant au problème que les développeurs rencontrent. De plus, différents critères feront leur apparition et nous permettront de les comparer méticuleusement.

À la suite de ce chapitre, nous serons en mesure d'extraire les critères majeurs qu'une entreprise ou un développeur doit prendre en considération lors de la création d'une application. Nous pourrons ainsi les développer pleinement. Après cette description, nous serons capables d'identifier les outils nécessaires en fonction des besoins du créateur d'applications.

Finalement, ces différentes recherches nous permettront d'orienter une entreprise ou un développeur dans le choix d'une technologie de développement mobile. Ce chapitre permettra de se diriger vers la solution la plus adaptée quant aux besoins et compétences requises.

2 Préalable indispensable à la compréhension de la recherche

2.1 Qu'est-ce qu'une application mobile ?

Une application mobile est un type de logiciel spécialement développé pour fonctionner sur les appareils mobiles tels que les smartphones ou les tablettes. Ces applications sont faites de codes permettant l'utilisation des différentes fonctionnalités de ces appareils mobiles comme l'antenne GPS, l'accéléromètre ou l'appareil photo (Anne Salz & al 2013). Au commencement, celles-ci avaient les mêmes fonctions, bien que plus limitées que celles des ordinateurs classiques comme le traitement de texte, un agenda ou une boite de réception mail. Ce type d'application a ensuite enregistré un énorme succès avec l'« App Store » d'Apple qui proposait des milliers d'applications pour iPhone, iPad et iPod touch. On peut les retrouver sous différentes catégories telles que les jeux, les réseaux sociaux, les navigateurs Web, les utilitaires ou même des applications de divertissements.

Aujourd'hui, l'App Store d'Apple et le Google Play d'Android disposent chacun de plus de 800.000 applications et il est devenu possible de trouver une application pour à peu près tout (IDC, 2013). Par exemple, si nous voulons connaître le nombre de calories que contient son petit-déjeuner ou retrouver où nous avons parqué notre voiture, il y a des applications conçues pour cela. Le marché des smartphones augmente d'environ 4 % par an et a déjà dépassé le marché des téléphones portables classiques (IDC, 2013). Ces applications font dorénavant partie intégrante de la vie quotidienne et ne sont pas prêtes de disparaître. Une application mobile peut être préalablement installée sur l'appareil ou bien se télécharger via un marché d'applications. Elles peuvent se retrouver sous différentes formes telles que l'application native, Web ou hybride.

2.2 Défi des applications mobiles

Les applications mobiles sont davantage un challenge que les applications pour ordinateur classique. Il faut prendre en compte le fait que l'on ne développe plus avec une mémoire vive qui dispose de multiples Giga-octets de RAM ou des téraoctets de disque dur. Ceci est dû au facteur de forme réduit des appareils mobiles (IBM corporation, 2012). Il en va de même pour la taille diminuée des écrans ainsi que de leurs résolutions (Lutes, 2012). Cela peut paraître logique, mais de telles caractéristiques changent radicalement la composition d'une application. Sur un ordinateur classique, le développeur peut se contenter de fournir la totalité des informations sur l'écran principal.

C’est différent sur un appareil mobile où il faut sélectionner l’information pour qu’elle soit condensée et pertinente afin que l’utilisateur se retrouve aisément dans l’index de l’application. Une approche progressive permet à l’utilisateur de passer de niveau en niveau de l’application, c’est-à-dire mettre les informations les plus synthétiques au premier abord pour ensuite accéder à celles de plus en plus détaillées. Une réflexion profonde sera nécessaire pour parvenir à une application de qualité.

La connexion internet sur les appareils mobiles ne possède pas un débit aussi élevé que sur les plateformes fixes. Il faut, dès lors, penser à limiter le transfert de données pour gagner en productivité, mais également économiser la batterie du terminal ainsi qu’éviter le dépassement du forfait tarifaire pour le consommateur final.

Le mode de saisie est un nouveau défi que le développeur se doit de surmonter. Le clavier traditionnel est devenu tactile et bien entendu réduit. Il faut penser à employer les différents raccourcis gestuels possibles à mettre en œuvre sur un écran tactile plutôt que de frustrer le consommateur à utiliser ce clavier restreint (IBM corporation, 2012).

Un appareil mobile dispose de fonctionnalités qui permettent de recevoir des données via le GPS, la caméra, l’accéléromètre ou le gyroscope. Il faut que ces différentes possibilités soient prises en considération lors de la création de l’application (IBM corporation, 2012).

Il y a également la façon dont un utilisateur interagit avec son appareil mobile. Le fait que l’appareil soit tenu entièrement dans sa main force le développeur à rendre son application interactive. De plus, l’utilisateur se retrouve couramment dans un milieu où le monde extérieur le distrait. Il faut donc que le développeur prenne en compte que le temps où l’utilisateur final reste attentif est plus restreint que sur un ordinateur classique. Il est donc important de faire une application claire et concise afin que la tâche soit effectuée rapidement.

Les échéances sont très courtes dans le domaine des applications mobiles (IBM corporation, 2012). Il y a une pression à devoir délivrer son application très rapidement. De plus, le développement doit être continu et lorsqu’une amélioration est apportée, celle-ci doit être portée sur tous les systèmes d’exploitation en même temps. Avec des échéances aussi réduites, la traçabilité est primordiale. Il faut pouvoir tracer chaque étape d’un projet, partager et enregistrer chaque information afin que le projet soit complet, mais surtout qu’il soit possible de retrouver facilement les erreurs possibles (Lutes, 2012).

Il est important d'avoir un bon outil pour tester et déboguer son application. En effet, il est impossible d'écrire un programme d'une traite, sans qu'il y ait des erreurs à corriger. Il est donc primordial de disposer d'un débogueur fonctionnel. Le processus de test est un point très complexe et peut devenir très coûteux pour le développeur qui veut présenter son application sur différentes plateformes. La matrice du projet contient tellement de variables qu'il est impossible de tout y insérer. Le nombre de variables est bien plus grand dans le monde mobile que dans celui des applications pour ordinateur classique. Le nombre conséquent d'appareils, d'opérateurs mobiles, de systèmes d'exploitation ainsi que de fonctionnalités fait que le développeur doit prendre plus de choses en considération. C'est ce nombre considérable de variables qui fait que le processus de test n'est pas évident. Pour savoir si son application fonctionne sur différents appareils, il existe différentes solutions (IBM corporation, 2012). La première est de tester son application manuellement sur chaque appareil. Pour se faire, il faut disposer de tous les terminaux disponibles sur le marché. L'achat de tous ces appareils tests représenterait un coût inconcevable. La deuxième solution est l'utilisation d'émulateurs et de simulateurs permettant de simuler une application mobile sur un ordinateur classique. Même si ces outils sont en amélioration constante, ils ne permettent pas toujours de répéter exactement la situation réelle et tester sur un appareil réel reste plus objectif. De fait, en simulant une application sur un ordinateur classique, différentes fonctionnalités comme celles liées aux capteurs sensoriels ne seront pas correctement testées. Il se pourrait alors que, lors du déploiement, l'application ne soit pas complètement fonctionnelle, voire non utilisable.

2.3 Qu'est-ce qu'un environnement de développement intégré ?

Un environnement de développement intégré (EDI) est un environnement de programmation qui a été compacté en application et permet à son utilisateur d'éditer du code, de le compiler, le déboguer ou même de construire l'interface graphique afin de développer des logiciels (Kingesle-Hughes A & al. 2005). Les différents environnements de développement sont souvent associés à un langage de programmation. Leur tâche principale est de fournir aux développeurs un environnement de travail ergonomique et intégré permettant la mise en œuvre des bibliothèques logicielles qui sont fournies par les Frameworks.

2.4 Qu'est-ce qu'un Framework ?

La traduction française du mot « Framework » est « cadre de travail ». Comme son nom l'indique, il représente le cadre où le développeur effectue son job. En d'autres mots, c'est un ensemble de composants qui permet de créer l'architecture d'un logiciel ou d'un site Web (Degani & al., 2011).

Une métaphore intéressante afin d'expliquer le fonctionnement d'un « Framework » est celle d'une recette de cuisine. Imaginons que le développeur soit un cuisiner et que tous les ingrédients nécessaires pour les différentes recettes soient fournis par un grossiste, ici le « Framework ». Le cuisinier n'a plus qu'à ajouter les ingrédients dans l'ordre ainsi que de les préparer pour obtenir le résultat désiré. La charge de travail se voit davantage réduite par rapport au cuisinier qui devrait faire pousser ses légumes et élever différents animaux pour en obtenir leurs viandes. C'est identique pour le développeur qui emploie un « Framework », il n'a plus qu'à se servir des différents composants déjà présents pour créer ce dont il a besoin. Grâce à ce cadre de travail, le développeur devient plus aisément capable d'écrire les lignes principales du programme ou de la page Web qu'il souhaite construire. Cela engendre un gain énorme en temps, car le développeur ne doit pas réinventer ce qui existe déjà. Il peut se concentrer essentiellement sur le fond de ce qu'il développe, sans se préoccuper de la forme. Le « Framework » aide en forçant les développeurs à créer un code bien organisé et automatiquement de qualité via les différents outils qu'il met à disposition. Le gain en productivité s'y voit décuplé via son usage.

Les développeurs hautement expérimentés, concepteurs de ces environnements de travail, rendent ces « Frameworks » robustes et flexibles. De plus, une grande communauté se trouve derrière ceux-ci. Cela permet aux concepteurs d'obtenir de nombreux rapports sur les différents bugs probants venant d'un « Framework » et qui peuvent, dès lors, être vite réglés.

Il existe un grand nombre de « Frameworks » disponibles et ceux-ci sont destinés à un ou plusieurs langages de programmation différents. On peut voir le « Framework » comme la boite à outils du développeur (Bacco, 2013). Celle-ci n'est pas fournie avec la panoplie complète d'outils nécessaires au développement de chaque plateforme ou application. Il peut être nécessaire d'y ajouter certaines choses comme les bibliothèques logicielles propres à chaque plateforme par exemple.

2.5 Qu'est-ce qu'une interface de programmation ?

Une interface de programmation, plus connue sous son abréviation anglaise « API » (application Programming interface), est un langage de programmation permettant à deux programmes logiciels différents de communiquer ainsi que de fournir des données ou autres dont ils auraient besoin (Gunelius 2011).

Dans notre cas, l'interface de programmation va permettre à une application tierce de demander différentes informations au système d'exploitation de l'appareil. Ces informations peuvent être liées à des données ou des fonctionnalités propres à l'appareil telles que l'accès aux fichiers, aux photos, aux différents capteurs ou toutes autres données liées à celui-ci. Ce type d'interface agit donc en tant qu'intermédiaire afin de permettre une communication plus aisée entre une application mobile et le système d'exploitation visé (Anne Salz & al 2013). Prenons, par exemple, l'installation d'une application de géolocalisation sur un iPhone. Celle-ci devra communiquer avec le système d'exploitation d'Apple (iOS) afin d'obtenir les informations de localisation nécessaires à son utilisation.

2.6 Conclusion

Via ce chapitre, nous avons pu en apprendre un peu plus sur l'existence des applications mobiles et les défis qu'elles engendrent aux développeurs. Nous pouvons en déduire que les applications mobiles sont relativement différentes des applications classiques. Nous avons pu définir les difficultés liées à la mobilité de ces appareils. Il était également intéressant de définir certains termes dont nous aurons recours tout au long de cette recherche. Nous sommes dès à présent prêts pour analyser les divergences entre les quatre plateformes mobiles choisies pour notre étude.

3 Analyse des différents systèmes d'exploitation mobiles

Avant d'entrer pleinement dans les solutions de développement d'applications mobiles, nous allons examiner plus en détail les quatre systèmes d'exploitation mobiles dominants le marché. Précisons d'abord que le système d'exploitation permet aux smartphones de faire tout ce dont ils sont capables tels que la navigation internet ou les appels téléphoniques. Ils sont donc des facteurs essentiels quant au développement d'une application.

3.1 iOS

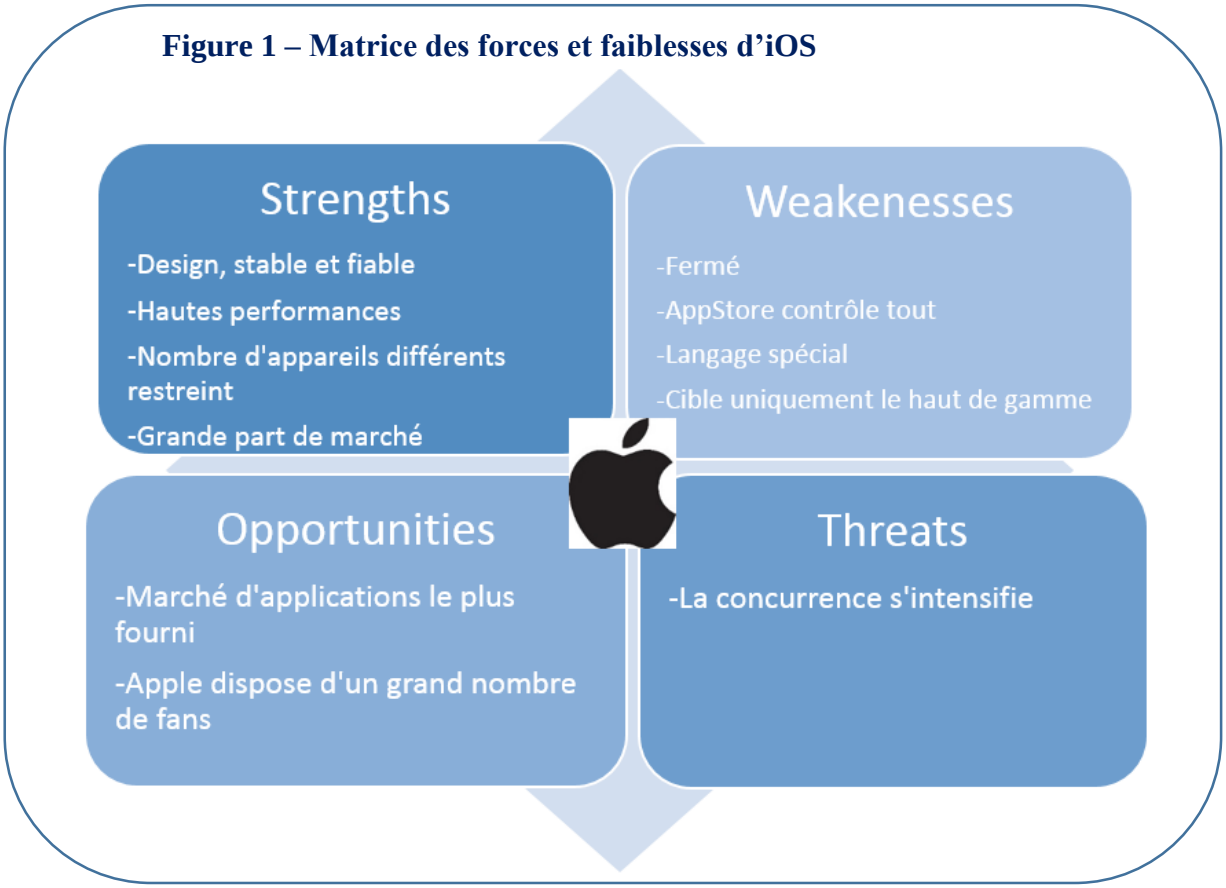
La plateforme d'Apple est de loin la plus prestigieuse et la plus populaire des plateformes mobiles, et pour cause, elle a mis en route le marché des *smartphones* avec l'introduction du célèbre iPhone en 2007. Celui-ci s'est fait connaître grâce à son grand écran tactile et son système d'exploitation (iOS) qui est une version réduite de son grand frère, Mac OS X, présent sur les ordinateurs de bureau de la marque Apple (Haroon, 2011). De plus, elle a le mérite d'être intuitive et ergonomique. Son magasin d'applications, l'App Store, est le plus fourni qui existe sur le marché. Elle n'est disponible que pour les différents produits de la marque Apple tels que l'iPad, l'iPhone et l'iPod. Ce système d'exploitation est facile d'emploi et irréprochable au niveau de la conception. En effet, la plateforme de la pomme est tellement simple que même de très jeunes enfants peuvent se servir d'un iPad. C'est en quelque sorte une des clés de leur succès.

Cependant, elle est assez refermée sur elle-même et pour cause, elle ne laisse que très peu de place à la créativité de l'utilisateur. Apple ne permet pas de vendre ou distribuer des applications ailleurs que sur son magasin et garde donc un contrôle total sur les applications distribuées (Lutes, 2012). Apple se permet également de rémunérer les développeurs à hauteur de 70 % du profit que leur application génère (CGV-EXPERT, 2012). Par rapport aux autres plateformes, elle est celle qui est le moins personnalisable. Elle n'est pas compatible avec Flash, ce qui cause un défaut de fonctionnement à l'accès de certaines informations Web. En contrepartie, iOS reste leader dans les domaines d'applications tels que l'industrie de la musique, la médecine et les applications axées sur le design (Rothman W, 2013).

Son célèbre marché d'applications s'appelle « App Store » et dispose de plus d'un million d'applications (Fiegerman, 2012). Avec l'iPhone qui est devenu le smartphone le plus populaire du marché, la communauté de développeurs est devenue très répandue.

Sur les appareils Apple, la programmation s'effectue à l'aide de l'environnement de développement intégré X-code en employant son propre langage (Lutes, 2012). Celui-ci n'est ni plus ni moins que l'Objective-C. Ce dernier est une version évoluée du langage de programmation C, mais qui rajoute certaines fonctionnalités. Ce langage est uniquement employé par Apple. Pour se lancer dans le développement iOS, le développeur doit obligatoirement disposer d'un support MAC puisque X-code fait partie de l'environnement de développement de MAC OS X. X-code comprend différents outils permettant de développer, déboguer et émuler des applications. Les différents outils de développement sont gratuits, mais il est nécessaire d'acquérir une licence de développeur iOS qui revient à 99 $ pour les particuliers et à 299 $ pour les entreprises (APPLE.COM, n.d.). Il est également nécessaire de préciser que le test d'une application sur un vrai terminal ainsi que la publication de celle-ci via Apple, est un processus très long et contraignant (Duffy, 2012). Il faut que le développeur ainsi que le matériel qu'il utilise, soient validés et enregistrés auprès d'Apple. Ces différentes requêtes d'Apple peuvent être source de conflit et cela peut prendre du temps à se résoudre. De plus, la validation d'une application prend en moyenne cinq jours durant lesquels le développeur ne peut intervenir.

Figure 1 – Matrice des forces et faiblesses d'iOS



La matrice SWOT d'iOS nous permet d'évaluer le système d'exploitation d'Apple. Celui-ci est très développé et concurrence le leadeur du marché. Malgré tout, ses faiblesses rendent ce système d'exploitation vulnérable vis-à-vis de la concurrence.

3.2 Android

Android est un système d'exploitation « open source » créé par Android Inc., racheté par la société Google en 2005 (Beavis, 2008). La base d'Android est fondée à partir du kernel de Linux et de la plateforme de programmation Java. Elle est la plateforme la plus courante sur les différents smartphones et tablettes du marché. On peut la comparer à Windows sur les ordinateurs classiques. De fait, la plupart des terminaux des différents fabricants disposent de cette plateforme. Les versions peuvent être remodelées en fonction de la marque de l'appareil ou bien même de son opérateur de base. Ceci engendre une compétition acharnée sur le prix, le design, les caractéristiques techniques ainsi que sur la qualité. Cette flexibilité accrue permet aux différentes catégories d'utilisateurs d'y trouver leurs intérêts. À titre d'exemple, il permet le support d'une large gamme de tailles et de résolutions d'écran sur les différents appareils. C'est un système d'exploitation qui bénéficie d'une énorme communauté de développeurs ainsi que de la documentation bien fournie.

Cette plateforme offre une expérience comparable à celle d'iOS quant aux fonctionnalités et caractéristiques. Le changement vient principalement de la flexibilité et de la personnalisation énorme de cet OS. C'est d'ailleurs le seul système d'exploitation à proposer des « widgets ».

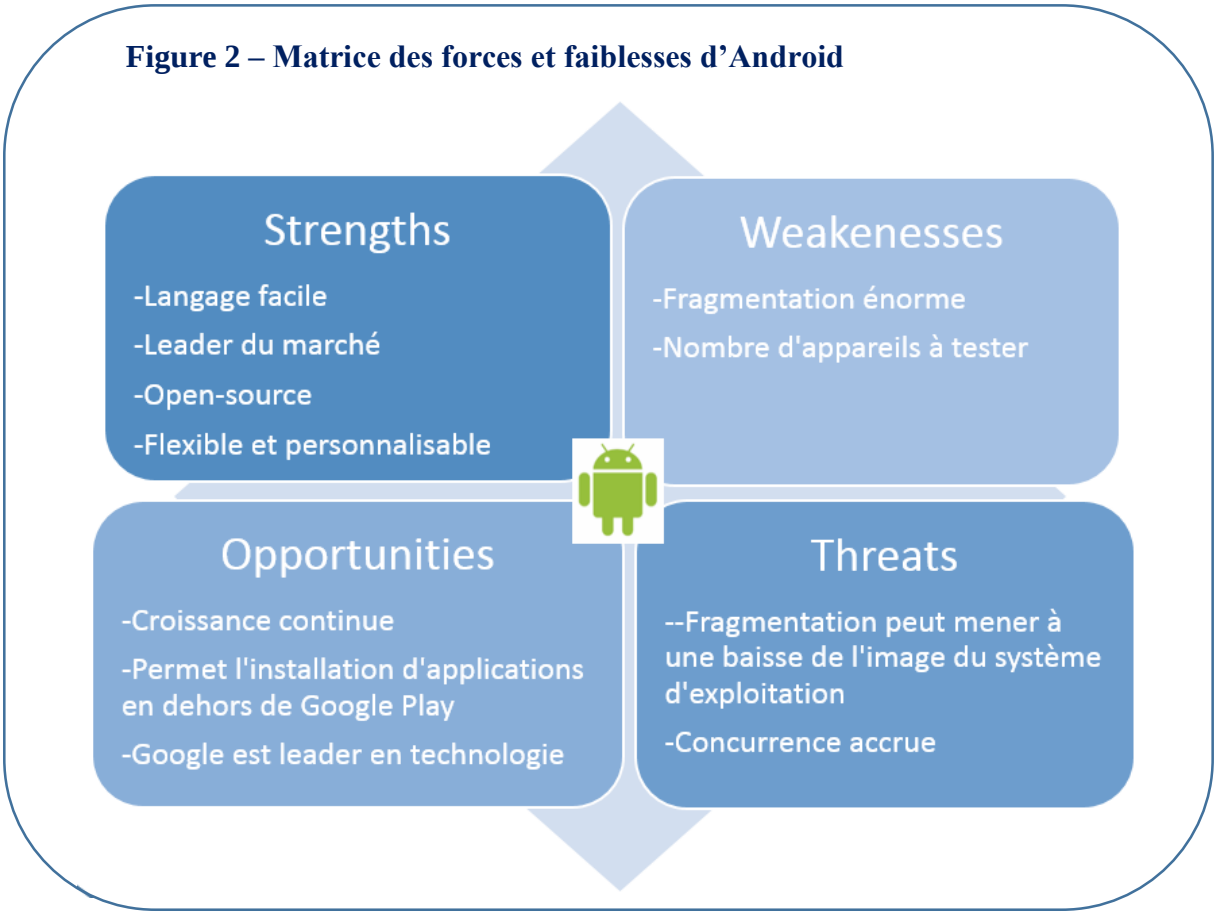
Cet avantage de généralisation de la plateforme n'est pas sans défaut. Le grand nombre de modèles aux différentes caractéristiques techniques ne permet pas de développer parfaitement la plateforme en fonction de ceux-ci. Android se retrouve donc avec un système d'exploitation moins stable et moins bien dessiné que l'iOS d'Apple. En d'autres mots, Android n'est pas autant affiné ni aussi parfaitement adapté à ses appareils que ne l'est son concurrent direct.

Les applications exécutées sur les terminaux Android sont développées en langage JAVA. Le marché d'applications disponibles sur cette plateforme est « GooglePlay ». Il faut noter que ce système d'exploitation ne restreint pas ses utilisateurs à n'installer que des applications à partir de « GooglePlay », mais également via la distribution directe du fichier d'installation d'une application Android (Android Package files).

Si le développeur d'une application décide de distribuer son application à travers « GooglePlay », celui-ci recevra 70 % des revenus et les 30 % restants seront dus à Google, pour la distribution et le coût d'exploitation (GOOGLEPLAY, 2012).

Sur Android, l'environnement de développement intégré le plus utilisé pour le développement natif est Eclipse. Celui-ci intègre différents outils destinés à Android ainsi qu'un émulateur d'applications. Il faut y ajouter le SDK Android qui représente le kit de développement permettant de développer une application pour une version d'Android déterminée. Ces différents outils ont l'avantage d'être complètement gratuits et disponibles au téléchargement. Néanmoins, une licence développeur revient à 25 $ par an (GOOGLEPLAY, 2012). Le langage utilisé est le célèbre langage Java. Ceci permet d'avoir davantage de documentation et une large communauté déjà présente sur le marché.

Figure 2 – Matrice des forces et faiblesses d'Android



Android est devenu le numéro 1 sur le marché grâce à son système d'exploitation « open source » permettant au constructeur de le modifier à leur guise. Cette force est à double tranchant puisqu'elle a créé une fragmentation démesurée.

3.3 Windows Phone 8

Windows Phone 8 est bien entendu la plateforme développée par Microsoft succédant à Windows Mobile. Cette plateforme a un taux de pénétration loin derrière celles d'Android et d'iOS et concerne donc un nombre inférieur d'utilisateurs. Elle est cependant en pleine expansion et Microsoft n'hésite pas à déployer les grands moyens pour revenir sur la concurrence.

Sa caractéristique principale est l'interface utilisateur qu'elle propose. Celle-ci est similaire à l'interface Metro disponible sur le système d'exploitation Windows 8. C'est une interface très minimaliste basée sur la typographie. Elle se résume à une série de tuiles qui représentent les différentes applications ainsi que l'accès aux différentes fonctionnalités de l'appareil. Tout est accessible via le bureau de la plateforme et les différentes notifications, informations nécessaires sont intuitivement intégrées aux icônes propres à chaque application. Elle se veut très simpliste, personnalisable, mais également énormément réactive.

Le marché d'applications disponibles sur Windows Phone 8 se nomme « Windows Phone Store ». Il propose cinq fois moins d'applications que ses concurrents, qui disposent d'ores et déjà d'une avance considérable en cette matière (Rubino, 2013). Les applications qui s'y retrouvent sont développées en langage C#, VB.NET et également en C++ depuis sa dernière version. Cependant les applications peuvent également être installées via le fichier XAP d'installation ou en utilisant le déploiement « Over The Air » (Kramer, 2012).

Cette nouvelle version de Windows Phone a procédé à un changement radical qui est la compatibilité hardware des différentes plateformes Windows. En effet, les différentes plateformes disposent du même noyau qui n'est, ni plus ni moins, que le noyau NT déjà disponible depuis longtemps sur Windows classique. C'est un grand pas en avant pour unifier tablettes, smartphones et pc traditionnels. C'est également un avancement technologique pour les développeurs. Ils peuvent désormais échanger le code créé entre les différentes plateformes. De plus, Microsoft dispose du meilleur support et de la plus grande communauté de développeurs qui existent. La société de Richmond offre aux développeurs de nombreuses conférences et des outils suivant exactement leurs besoins.

La plateforme de développement pour Windows Phone 7 est Silverlight et XAML/C# pour la version 8 du système mobile d'exploitation Windows. XAML/C# est en fait une mise à jour de Silverlight. Ainsi le code XAML permet d'écrire l'interface graphique de son application tandis que le langage C# offre la logique à celle-ci. Les différentes documentations techniques sont de qualité et le développement est très facilité pour ceux ayant déjà travaillé sur des applications pour PC. La plupart des outils de développement Windows Phone sont gratuits dans leur version express, mais pas dans leurs versions professionnelles ou standards (MICROSOFT, 2013).

Figure 3 – Matrice des forces et faiblesses de Windows Phone

Nous constatons que Windows Phone dispose de forces et d'opportunités permettant de garder, voire d'augmenter sa part de marché. Pour se faire, il sera nécessaire d'augmenter le nombre d'applications compatibles ainsi que de se faire connaître auprès d'un plus grand nombre d'utilisateurs.

3.4 RIM

RIM (Research In Motion) est la compagnie qui développe le système d'exploitation incorporé à tous les appareils BlackBerry. Depuis 1999, leurs appareils se sont focalisés sur l'envoi, la réception d'email et un clavier physique QWERTY.

RIM a fait connaitre les appareils BlackBerry dans le secteur des entreprises principalement. Il détenait une part de marché supérieur à 15 % jusqu'au milieu de l'année 2010. Ensuite, le système d'exploitation n'a fait que céder des parts de marché à Android et iOS (Gartner, 2013). Il faut dire qu'avant 2013, les terminaux BlackBerry n'étaient pas tactiles et disposaient d'un OS très vieillissant. Malgré la cohérence et les fonctionnalités, le système d'exploitation et son matériel étaient loin d'égaler la montée en puissance d'iOS et d'Android. Ils ont également pris un retard considérable quant au nombre d'applications disponibles sur leur marché d'applications « BlackBerry World ». De plus, la barrière entre les appareils destinés aux entreprises ou aux particuliers s'est grandement amoindrie. BlackBerry se devait donc de revoir sa politique de vente et son ciblage. Ils ne pouvaient plus se limiter à des terminaux destinés uniquement aux entreprises. Il fallait donc réagir à cette nouvelle révolution qui requiert un système disposant d'un écosystème riche en applications, en interactivité tactile et en nouveautés (Fingas, 2013).

La réaction de RIM fut BlackBerry 10. Ce nouveau système d'exploitation apporte un grand nombre de nouveautés et s'aligne sur la concurrence quant aux fonctionnalités et à l'écran tactile. Une autre amélioration est la quantité d'applications disponibles sur le BlackBerry World. Ce nombre est passé de 70.000 à plus de 120.000 applications en moins d'une année (Nguyen, 2013). Ce chiffre a pu être atteint en rémunérant les développeurs créant des applications approuvées par BlackBerry, mais également en permettant aux développeurs de porter aisément leurs applications développées pour Android. Le nouveau système d'exploitation BlackBerry 10 a fait son entrée avec panache. Son nouveau kit de développement laisse accès à une panoplie d'outils permettant le développement natif, mais également Web (Hinault, 2012). En effet, il est dès à présent possible de développer une application à l'aide du kit de développement natif C/C++ et son Framework « cascades » qui permet de créer l'interface graphique.

On peut également développer à l'aide du langage JAVA et l'environnement de développement Eclipse, l'Android « runtime » permettant le portage des applications Android ou à l'aide du trio de langages Web « HTML5/CSS/JavaScript ». L'environnement Visual Studio de Microsoft peut également être employé pour développer en C/C++. Cette grande offre de moyens de développement est le nouvel atout de RIM. Il vise à satisfaire le plus grand nombre de développeurs afin d'acquérir rapidement un nombre d'applications compatibles important.

Figure 4 – Matrice des forces et faiblesses de BlackBerry

En regardant cette matrice des forces et faiblesses, on peut imaginer que BlackBerry peut stopper l'hémorragie. En offrant autant de possibilités aux développeurs, la communauté ne peut que s'agrandir et voir son public cible augmenter.

3.5 Conclusion

Après avoir analysé les quatre acteurs principaux du secteur des systèmes d'exploitation mobiles, nous constatons que le marché est constitué de multiples plateformes hétérogènes sur différents points. En effet, bien que les capacités et fonctionnalités des appareils disponibles soient similaires, les systèmes d'exploitation présents sur ceux-ci n'ont que très peu de points communs. En principe, cela est une bonne chose pour le marché. La concurrence stimule les sociétés à mettre sans cesse à jour leurs applications, car les utilisateurs ont le choix entre différentes interfaces graphiques, différentes cultures, etc.

C'est une « autre paire de manches » pour les développeurs ayant pour but de fournir maintes applications sur le plus d'appareils possibles. La segmentation de ces systèmes d'exploitation engendre pour eux l'utilisation de plusieurs langages et outils de programmation, mais aussi la compréhension de chaque fonctionnalité présente sur chacun des systèmes d'exploitation. De fait, là où Android requiert de développer en « Java » à l'aide d'Eclipse, IOS demandera d'utiliser X-code avec le langage inhabituel « Objective-C ».

De la même manière, lorsqu'il est possible d'installer des applications en dehors du magasin respectif pour les appareils Android et Windows Phone, cela n'est pas le cas pour iOS et RIM. Il y a un tas de petits détails qui font que ces systèmes d'exploitation n'ont que très peu de compatibilité entre eux. Le développeur est le premier lésé par la segmentation conséquente du marché des plateformes mobiles de par l'investissement qu'il doit faire afin de rendre son application présente sur tout le marché.

Tableau – 1 Comparaison des principaux systèmes d'exploitation mobile :

Caractéristiques/Plateforme :	Android	iOS	Windows Phone	BlackBerry 10
Langage :	Java	Objective-C	C#, VB.NET, C++	C/C++, JAVA, HTML5/CSS/JS,
Environnement de développement :	Eclipse	X-code	Visual Studio	Cascades, Visual Studio, Eclipse
Open source :	oui	non	non	non
Déploiement :	PlayStore, apk	App Store	Windows Phone Store, XAP, OTA	BlackBerry World
Part de marché (GARTNER, mai 2013) :	74,4 %	18,2 %	2,9 %	3 %

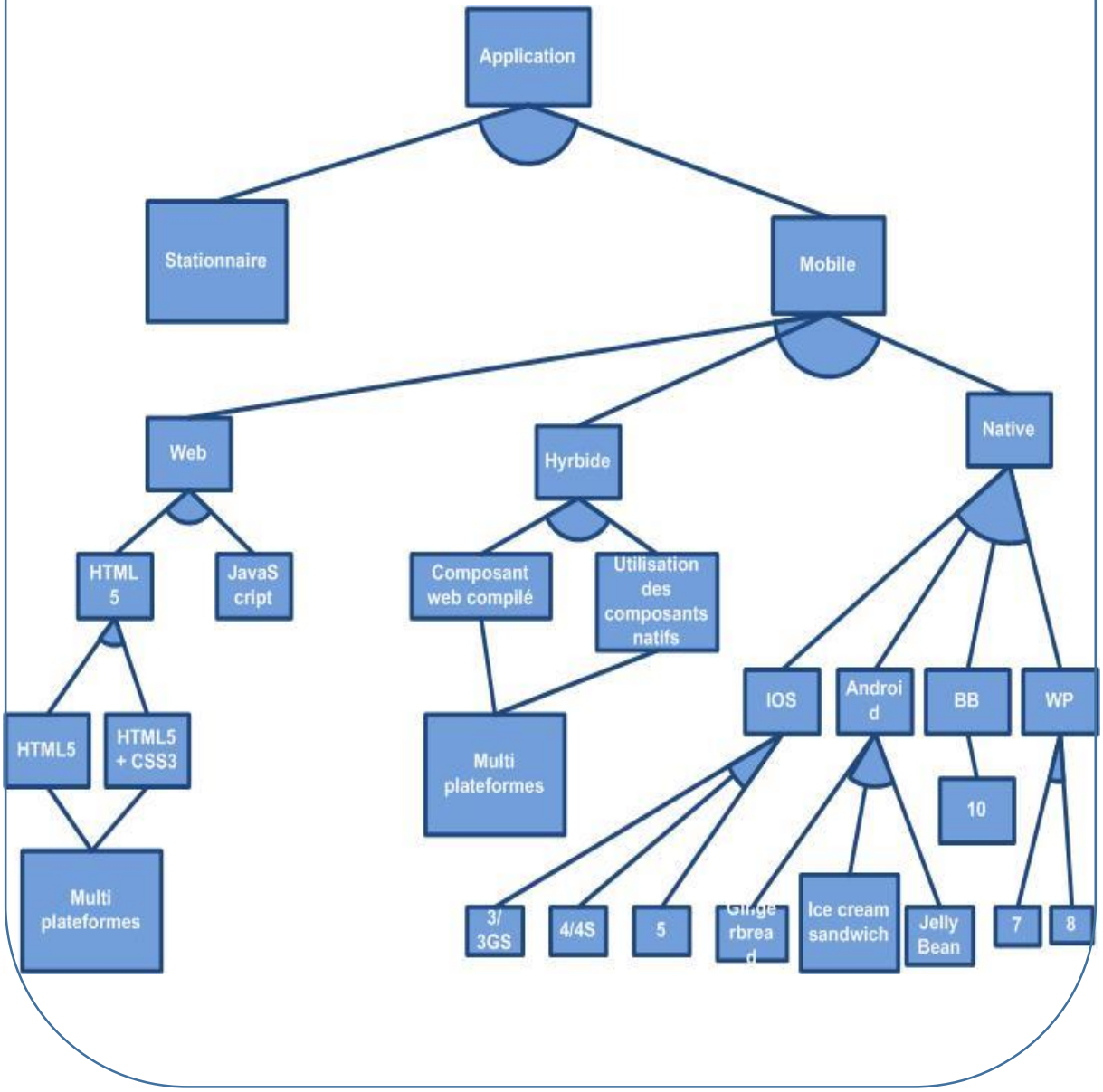
Le tableau 1 compare les différences liées au développement de ces quatre systèmes d'exploitation mobiles.

Grâce à ce chapitre, nous avons pu observer ces divergences et nous sommes dorénavant capables de mieux comprendre l'hétérogénéité du marché des téléphones intelligents. Dès lors, nous sommes capables d'observer les différentes méthodes possibles afin de fournir une application sur ces différents terminaux.

4 Les différentes possibilités de développement d'une application mobile

Nous allons maintenant analyser les différentes solutions possibles lorsqu'un développeur se lance dans la création d'une application mobile. Comme on peut le voir sur la figure 5, le choix est restreint à trois solutions différentes : Native, Web et Hybride. Nous y constatons également que la solution native subit une fragmentation conséquente.

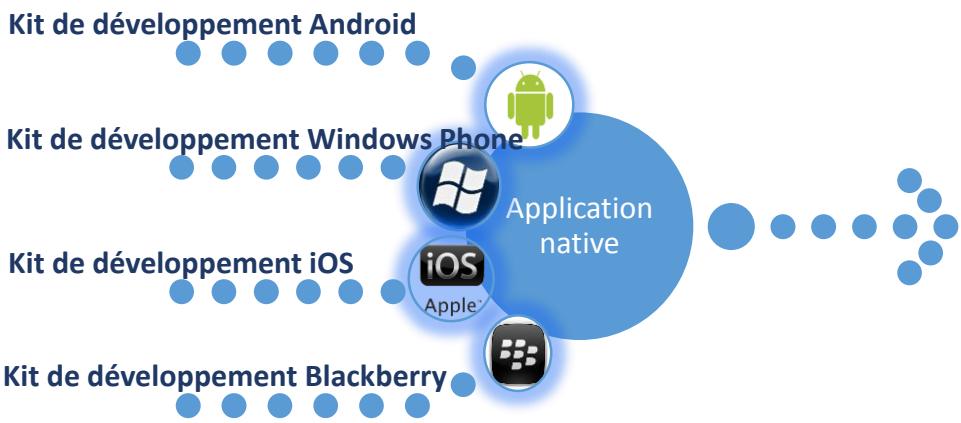
Figure 5 – Arbre de fragmentation du développement d'application mobile



4.1 L'application native

L'application native est l'application la plus courante sur les différentes plateformes mobiles. Elle est programmée dans le langage de l'appareil et ne fonctionne que sur la plateforme de celui-ci. C'est-à-dire qu'une application native *iPhone* ne pourra fonctionner sur un téléphone *Android* et inversement (CHA, YUN 2013). Ce type d'application est principalement distribué à travers un magasin d'applications propre à chaque plateforme. On devra donc se rendre sur l'*App Store* d'Apple, le *Play store de* Google, le *BlackBerry store* de RIM ou le *Windows Phone Store* de Windows Phone pour télécharger et installer l'application souhaitée. Elles sont principalement téléchargées à l'aide d'une connexion internet, mais, une fois installées sur l'appareil, la connexion à internet n'est plus utile. Les applications de base sur un appareil mobile sont des applications natives. À titre d'exemple, les différents navigateurs internet sont des spécimens parfaits de ce type d'applications (Balkan, 2012).

Figure 6 – Développement d'applications natives



4.1.1 Ses avantages

L'application native bénéficie d'avantages non négligeables par rapport à l'application Web mobile ou hybride.

Premièrement, elle permet une optimisation de chaque appareil en fournissant un accès aux propriétés avancées de ceux-ci. Nous pouvons, dès lors, profiter pleinement des fonctionnalités de l'appareil telles que l'antenne GPS, la liste de contact, la caméra, le microphone, le gyroscope et l'accéléromètre (Stangarone, 2013). Cela permet également d'accéder à certains diagnostics de l'appareil tels que ceux de la batterie ou du réseau mobile.

Pour de nombreuses applications, l'accès à ces fonctionnalités peut être très important. De plus, les bibliothèques logicielles sont mises à jour couramment et directement disponibles aux développeurs (CHA, YUN 2013).

Deuxièmement, l'application est créée de manière à convenir exactement à la dimension et à la résolution de l'écran et permet donc de fournir un rendu graphique élevé (CHA, YUN 2013). Cela rend les applications davantage fluides et réactives. La qualité d'image, l'interface utilisateur et l'ergonomie sont mises en avant et permettent d'exploiter au mieux l'écran du terminal utilisé.

Troisièmement, les applications natives sont distribuées à travers un magasin d'applications qui rend l'application facilement trouvable en fonction de ce que recherchent les clients (CHA, YUN 2013). C'est une manière de rendre son application payante par le développeur ou l'entreprise, de recevoir aisément un feedback de l'utilisateur, mais, également, à le rassurer quant à la sécurité de l'application (Grigsby 2009). Cette dernière nécessite un contrôle méticuleux avant d'être disponible sur le magasin d'applications. À titre d'exemple, l'App Store d'Apple détient plus de 500 millions d'utilisateurs actifs sur leur marché d'applications (YAROW 2013). Cet énorme écosystème permet de monétiser ses applications, de se faire un nom parmi les développeurs, mais également de se faire connaître du grand public.

Finalement, l'installation de l'application sur l'appareil crée automatiquement un raccourci sur le bureau du smartphone et se retrouve ainsi présente dans le portefeuille d'applications de ce dernier. L'accès y est aisé pour l'utilisateur et est également bénéfique pour le créateur de l'application. En effet, cela lui permet de maintenir un contact avec l'utilisateur grâce à la nouvelle icône présente sur l'appareil de celui-ci, l'icône étant présente jusqu'à la désinstallation de l'application. Cette dernière est, dès lors, autorisée à envoyer des notifications à l'utilisateur afin de le prévenir d'une nouvelle mise à jour ou de lui rappeler qu'elle existe.

Figure 7 – Avantages de l'application native

4.1.2 Ses inconvénients

Les inconvénients de l'application native sont davantage axés sur le développement que sur l'expérience utilisateur.

Pour commencer, nous avons le fait qu'il faille développer l'application sur chaque plateforme avec des langages complètement différents (CHA, YUN 2013). Le développeur doit donc installer des langages disparates ainsi que les apprendre. Par exemple, il ne sera pas capable de récupérer les codes utilisés pour créer une application destinée à l'iPhone lors de la conception d'une application Windows Phone et inversement. Une application native ne sera pas nécessairement compatible sur chaque appareil d'une plateforme. Il est donc probable que le développeur doive également modifier son application en fonction de l'appareil, car ils ne disposent pas tous des mêmes caractéristiques. En effet, la taille de l'écran, la puissance et le processeur des terminaux sont très variables.

Les applications natives requièrent des compétences spécifiques pour chaque plateforme, ce qui engendre des difficultés à trouver des développeurs adéquats sans devoir payer des prix exorbitants pour leurs services (Anne Salz & al 2013).

Il devient, dès lors, très fastidieux et coûteux de développer une application disponible sur toutes les plateformes et encore davantage sur tous les appareils présents sur le marché. Précisons également que certaines plateformes requièrent un développement à l'aide d'outils non « open source », c'est un coût qu'il faut également prendre en compte en se lançant dans du développement natif.

D'après une étude de Forrester Research, une application native prend au moins six mois de développement et représente un investissement entre 20.000 $ et 150.000 $ en fonction de son niveau de complexité (Stangarone, 2013). Cette somme est calculée pour le développement d'une seule plateforme et doit donc se multiplier par le nombre de plateformes que l'entreprise ou le développeur veut conquérir. Étant donné que quatre systèmes d'exploitation touchent une grande partie des consommateurs et sont eux-mêmes déclinés en smartphone et en tablette, une entreprise – si elle décide d'atteindre tous les consommateurs via une application native - devra débourser entre 160.000 $ et 1.200.000 $ pour être disponible sur les différents smartphones et tablettes présents sur le marché (Stangarone 2013). Il est difficile, mais avantageux, de trouver des développeurs ayant les capacités de développer sur chacun de ces différents systèmes d'exploitation.

La variable « temps » est aussi à prendre en considération. Imaginons qu'une application prend six mois par plateforme, deux ans seront nécessaires à un développeur afin d'avoir son application disponible sur les quatre plus grosses plateformes présentes sur le marché (Stangarone, 2013). À cette dépense et cette période de temps de développement, viennent s'ajouter le coût de maintien de ces applications. Une application native nécessite beaucoup de maintenance ainsi que des mises à jour pour assurer leurs pérennités. Un système d'exploitation évolue avec le temps et rend ainsi les applications de moins en moins adaptées, voire incompatibles. C'était déjà le cas sur les ordinateurs classiques lors du passage de Windows 98 à Windows XP où une mise à jour des applications était souvent de mise pour être à nouveau compatible. Dans le monde mobile, ce type de changement est plus fréquent, que ce soit sur la plateforme iOS, Android, Windows Phone ou BlackBerry. Cela requiert, en fait, un travail régulier pour garder son application compatible sur le long terme.

Ensuite, l'application native se voit forcée de passer par un magasin d'applications tel que l'App Store, Google Play, Windows Phone Store… Bien que ce point se retrouve également dans les avantages, il peut aussi être désavantageux. Mis à part le fait que le développeur ou l'entreprise doive attribuer une commission d'environ 30 % sur les ventes aux différents magasins, il y a toute une procédure à suivre avant de voir son application disponible au grand public (YANG, 2013). En effet, les dirigeants des différents marchés d'applications se donnent le droit de contrôler l'application proposée par son postulant avant la mise à disposition de celle-ci aux utilisateurs finaux. La durée de contrôle peut être relativement longue et se répète à chaque mise à jour de l'application. La perte de temps et, par conséquent d'argent, lors de l'attente de validation doit être prise en compte lorsque l'on décide de créer une application. De plus, les gérants des marchés d'applications se donnent également le droit de refuser ou de bannir une application de leur marché sans qu'aucun recours ne soit disponible pour le créateur de l'application. Il se peut donc qu'une application ne convienne pas à leur règlement ou simplement qu'une autre entreprise dénonce des droits de copyright bafoués pour que l'application soit supprimée (Friedeman, 2013). C'est donc un système qui peut être risqué pour les entreprises qui sont, de cette façon, à la merci d'une tierce partie. L'entreprise, mettant d'énormes ressources en jeu, prend le risque de voir ses efforts réduits à néant (Stangarone, 2013).

Pour terminer, le monde mobile est un secteur en plein essor dont la stabilité n'est guère acquise. De très grands acteurs se sont retrouvés suiveurs, voire carrément supprimés de ce marché. À titre d'exemple, en 2005, seulement RIM, Windows Mobile, Palm et Symbian étaient alors fort présents. Aujourd'hui, Palm n'existe plus du tout et les autres ne représentent que 8 % du marché total actuel. L'arrivée d'Android et d'Apple a complètement chamboulé le marché et, en 2012, ils représentent à eux deux 92,6 % de celui-ci (GARTNER 2013). Ceci est une preuve de l'instabilité et de la variabilité du monde mobile et de ses plateformes respectives. L'avenir est donc incertain : peut-être que les acteurs d'aujourd'hui ne seront plus là demain. En créant une application native, on prend le risque de dépenser du temps et de l'argent sans pour autant être certain de la rentabilité de celle-ci.

Figure 8 – Inconvénients de l'application native

4.1.3 Les langages et environnements de développement natif

Chaque plateforme dispose de son propre langage de développement afin d'obtenir une application native profitant des différentes fonctionnalités de l'appareil telles que le GPS, le gyromètre, la liste de contact, etc. (CHA, YUN 2013).

4.2 L'application Web mobile

Les applications Web mobiles sont des sites Web adaptés pour les appareils mobiles. Elles ne doivent pas être téléchargées ou installées sur l'appareil même, mais sont accessibles via le navigateur internet de ceux-ci, par exemple Internet Explorer, Chrome ou Mozilla (Anne Salz & al 2013). Elles permettent d'être accessibles sur les différentes plateformes sans devoir passer par leurs magasins d'applications respectifs. Quant à leur programmation, nous ne sommes plus obligés de passer par l'environnement de développement intégré propre à chaque plateforme. Nous sommes dans la capacité de programmer à l'aide d'un langage basique, tel que le html5, pour créer son application. Le développeur peut, en effet, créer une application mobile avec un langage commun, également utilisé pour les pages Web classiques. Elles sont aptes à tourner sur tous les appareils disposant d'un navigateur internet compatible et ne nécessitent pas de modification au sein même de l'application.

Il est important de préciser qu'une Web App est différente d'un site Web mobile. En effet, ce dernier n'est rien d'autre qu'un site internet allégé. L'accès se fait via le navigateur, le contenu y est minimisé et s'adapte à la taille de l'écran du téléphone mobile. La forme est optimisée de manière à être plus facile à utiliser depuis un écran tactile. Ici, l'accès est interrompu dès qu'il n'y a plus de connexion à internet.

La Web App se différencie de par les avantages dont elle dispose. Elle permet l'accès aux fonctionnalités telles que l'accès au GPS, gyroscope ou l'envoi de fichier. Lors de la première visite sur l'URL, une installation s'effectue dans le cache du navigateur et l'utilisateur devient capable d'utiliser l'application sans connexion internet.

Figure 9 – L'application Web mobile

4.2.1 Ses avantages

Les applications Web sont aujourd'hui moins présentes que les applications natives, pourtant elles détiennent de nombreux avantages non négligeables.

Elles ne représentent pas un énorme obstacle pour les développeurs qui sont déjà habitués à travailler avec des langages tels que le HTML, JAVASCRIPT ou CSS lors de la création de pages Web traditionnelles. Une entreprise qui développait auparavant des applications Web n'éprouverait que très peu de difficultés pour s'étendre et s'adapter au développement mobile. Ceci grâce aux compétences précédemment acquises par ses développeurs.

Une autre facilité de développement est le fait qu'il ne faut presque pas, voire pas du tout, tenir compte des différentes plateformes (IBM corporation, 2012). L'application Web, se lançant à travers un navigateur Web, ne nécessite que d'un appareil disposant d'un navigateur Web compatible. En théorie, le développeur ne devrait effectuer le travail qu'une seule fois pour être compatible sur les différents terminaux tournant sur Android, iOS, Windows Phone, BlackBerry… Non seulement l'entreprise réduit ses coûts en travaillant avec des développeurs déjà forts présents et disponibles depuis longtemps sur le marché, mais également en développant une application compatible sur les différentes plateformes (Stangarone, 2013). Ceci signifie que si le marché des plateformes mobiles change et que de nouveaux entrants arrivent sur le marché des systèmes d'exploitation mobiles, l'application Web sera déjà présente et apte à tourner sur ceux-ci. Une application Web mobile ne représente dorénavant pas un énorme danger pour les développeurs ou les entreprises.

Un autre point essentiel concernant ce type d'applications est leur maintenance et leur facilité de mise à jour. Lorsque l'on déploie une application Web mobile, les mises à jour destinées à l'utilisateur final se mettront en place immédiatement sur son appareil (SAVITZ 2013). Du fait qu'elles soient accessibles via le Web et non uniquement installées sur les appareils, leur mise à jour se fait automatiquement à la première connexion au serveur (Balkan, 2012). L'utilisateur ne doit, pour cela, rien faire et disposera immédiatement de la dernière version en date, peu importe son terminal. La maintenance de l'application s'y voit, dès lors, facilitée. Tous les utilisateurs se retrouvent sur la même version de l'application et il ne peut y avoir de quiproquo au niveau du support de la compagnie la concernant.

Le dernier point s'intéresse à la distribution et à la liberté de développement de l'application. Ici, l'application est distribuée à travers le Web et non via un magasin d'applications comme l'« App Store » d'Apple. Il n'y a donc pas de tierce partie qui vient se mettre en travers de notre chemin. Personne ne peut exiger une commission ou risque de ne pas accepter ou de supprimer l'application. La liberté de développement est un atout considérable de l'application Web.

Figure 10 – Avantages de l'application Web mobile

4.2.2 Ses inconvénients

Les applications Web mobiles ont globalement peu de désavantages, mais ceux-ci peuvent s'avérer contraignants pour l'utilisateur qui est la cible décisive du marché.

Premièrement, étant donné que l'application est développée pour être compatible sur tous les terminaux, cela provoque forcément des différences de fonctionnement sur l'un ou l'autre appareil. Ils n'ont pas tous la même taille d'écran, la même résolution, les mêmes processeurs, une quantité de mémoire identique, la même version de système d'exploitation, ni le même navigateur internet. Le développeur se doit donc de dessiner une interface utilisateur adaptable à tous les écrans des différents appareils. De plus, cette énorme fragmentation nécessite d'effectuer une période de test assez conséquente sur de nombreux terminaux afin de découvrir si l'application s'adapte à tous les modèles.

Deuxièmement, l'intégration à l'appareil est davantage modérée (Stangarone, 2013). Bien que le HTML5 ait apporté de nombreuses fonctionnalités, il reste des fonctions propres aux appareils dont on ne peut profiter. Ces fonctions sont l'appareil photo, le microphone et la liste des contacts. Malgré les nombreux efforts accomplis, il faudra attendre encore un moment avant de gérer aussi bien ces fonctionnalités que l'application native. Une application Web ne créera pas, non plus, de raccourcis sur le bureau de l'appareil et la seule manière d'en avoir un est de faire un « marque-page » internet. Cette technique n'est pas connue de tous les utilisateurs et peut-être contraignante par son manque d'ergonomie. De plus, elle ne supporte pas les processus en fond de tâche et n'enverront donc pas de notification en dehors de l'application même.

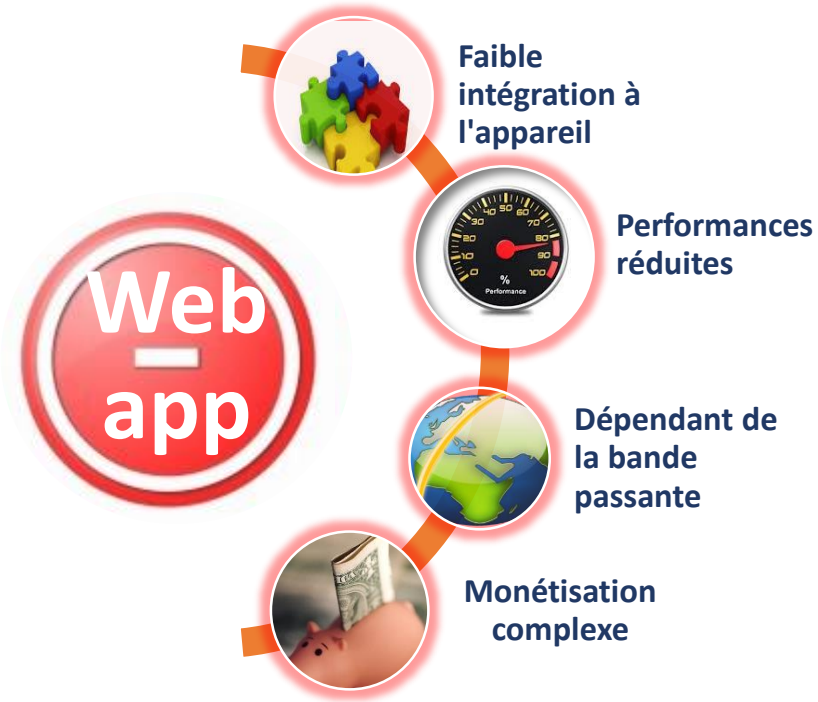
Troisièmement, elles sont dépendantes de la connexion internet. Le trafic du réseau est davantage sollicité puisque l'application ne doit pas qu'acquérir les données brutes, mais également tous les éléments qui concernent la mise en forme de l'application. Même s'il est vrai qu'avec le HTML5, l'application peut être utilisée en mode hors connexion tant que le cache a été précédemment enregistré. De plus, ce type d'application est lié à la qualité de la bande passante qui peut varier selon la situation et la rapidité d'exécution peut s'y voir détériorée (DEGANI & al. 2011). Les applications qui utilisent une connexion internet doivent également être adaptées aux différents protocoles et aux différents navigateurs (SAVITZ 2013). De ce fait, le développeur ne peut prendre l'avantage de certaines fonctionnalités du navigateur d'une plateforme donnée et espérer à la fois que cela fonctionnerait sur un autre appareil qui ne dispose pas des mêmes capacités.

Quatrièmement, l'application n'étant pas contrôlée par une tierce partie est plus à même de subir des failles de sécurité rendant l'utilisateur davantage vulnérable. Elles sont moins sécurisées dû au fait que le code source de l'application peut être facilement observé par une personne malveillante. Celle-ci pourrait injecter du contenu dans l'application à des fins peu scrupuleuses (MICROSOFT). Elle pourrait ainsi récupérer les cookies afin de voler la session de l'utilisateur ou de le rediriger vers une autre page Web. De plus, le HTML5 ne dispose pas encore de solution afin de crypter les données enregistrées dans le cache.

Cinquièmement, l'absence de marché d'applications pour ce type d'application rend la monétisation et la promotion plus ardue.

Finalement, les applications Web mobiles ne brillent pas dans les applications qui ont un besoin graphique élevé.

Figure 11 – Inconvénients de l'application Web mobile



4.2.3 Les langages de développement Web

4.2.3.1 Html5

HTML5 procure les fondations pour construire des applications sophistiquées qui réagissent bien et se lancent directement depuis un navigateur internet. Elles sont, en théorie, capables d'offrir une expérience utilisateur qui égale celle des applications natives. Ce langage a le mérite de posséder différentes technologies permettant aux développeurs de construire une application Web mobile puissante.

HTML5 détient un nombre de spécifications clés qui sont très utiles aux applications Web mobiles.

Tout d'abord, « *Canvas Drawing* » est une fonctionnalité implémentée dans presque tous les navigateurs que ce soit sur mobile ou tablette. Grâce à celle-ci, le développeur est capable d'insérer des images interactives, des graphiques, ainsi que des composants de jeu en deux dimensions sans que l'utilisateur n'ait à télécharger un plug-in additionnel (Selvadurai, 2013).

« *The touch events* » procure à l'application Web le contrôle de celle-ci par des gestes simples. Cela offre une expérience utilisateur plus interactive et semblable à celle de l'application native. Cette touche ergonomique aide les utilisateurs à naviguer rapidement dans l'application ainsi que d'y créer un côté ludique et agréable (Selvadurai, 2013).

Une interface de programmation liée à la géolocalisation permet d'employer l'antenne GPS, les données mobiles ou le réseau wifi afin de déterminer les coordonnées latitudes et longitudes de l'utilisateur (Verrechi, 2011). Cela procure à l'application la possibilité d'adapter son contenu, mais aussi l'expérience utilisateur en fonction du lieu où celui-ci accède à l'application.

L'application enregistrée dans le cache du navigateur est une autre de ses fonctionnalités. C'est une grande amélioration pour les applications Web mobiles qui permet d'accéder à celles-ci sans avoir recours à une connexion internet (Verrechi, 2011). De plus, en permettant l'accès au cache enregistré, on accélère le temps de réponse de l'application. En effet, le montant de données à télécharger est extrêmement réduit puisque ce qui a été précédemment téléchargé est réemployé à chaque utilisation et seulement les données mises à jour (ou nouvelles) nécessitent de passer par le serveur.

L'option du stockage local est une autre fonctionnalité de l'HTML5. L'application est autorisée à garder et à récupérer des informations internes afin d'optimiser l'expérience utilisateur. Le principe est identique aux cookies sur les sessions HTTP, mais il offre une capacité de stockage bien plus grande à ceux-ci (Selvadurai, 2012). En enregistrant les données clients de l'utilisateur dans la base de données du navigateur, l'application verra sa fluidité grandement améliorée et l'utilisateur pourra enregistrer des données sans la nécessité d'une connexion internet. Le mode hors-ligne devient nettement plus accessible grâce à cette fonction.

La fonctionnalité du support multimédia est aussi un apport intéressant de l'HTML5. L'audiovisuel occupant une place gigantesque dans le « Nouveau Monde » du Web, il fallait l'intégrer directement à ce nouveau format de données.

Le HTML5 est capable de supporter de l'audio et de la vidéo sans pour autant avoir le besoin d'installer un quelconque plug-in supplémentaire (Verrechi, 2011). Ces propriétés étaient auparavant assurées par Adobe Flash Player qui n'était compatible que sur certains appareils et qui se voit dorénavant oublié au profit du HTML5. Par rapport aux appareils mobiles non compatibles avec Flash Player, comme les iPhone, c'est une grande avancée, car dorénavant les vidéos peuvent être lues sans ce dernier plug-in.

Malgré cela, le HTML5 fait face à quelques difficultés. Le support pour toutes les interfaces de programmation HTML5 n'est malheureusement pas universel. De plus, beaucoup de navigateurs n'ont implémenté qu'une partie des spécifications du W3C (World Wide Web Consortium). Les développeurs doivent donc trouver une solution lorsque certaines interfaces de programmation ne sont pas disponibles (Selvadurai, 2012).

4.2.3.2 CSS3

CSS est l'abréviation de « Cascading Style Sheets ». C'est un langage qui est d'usage pour la présentation des documents HTML. C'est-à-dire qu'il s'occupe des polices, des couleurs, des marges, des lignes, de la hauteur, de la largeur, des images de fond, des divers positionnements et de tout ce qui peut encore être lié à la présentation (Kyrnin, n.d.). Ce langage offre beaucoup plus de possibilités que le HTML quant à l'habillage d'une application ou d'une page Web. De plus, il est pris en charge par les différents navigateurs internet présents sur le marché. Il est très intéressant de coupler ce langage au HTML. Ce dernier permet de structurer le contenu de votre page Web ou application tandis que le CSS servira à formater le contenu. Cela permet au développeur de séparer le contenu et le style de présentation de ce qu'il développe. Il est donc plus aisé de maintenir son application ou sa page Web à jour sans devoir sans cesse reconstruire ce qui est d'ores et déjà présent. Il apporte également la possibilité d'offrir des présentations différentes selon le type de média utilisé. Il peut donc être très séduisant pour créer des applications cross-plateforme puisqu'il permet à celles-ci de s'adapter en fonction de la taille de l'écran, du type de périphériques, etc.

4.2.3.3 JavaScript

JavaScript a été conçu en 1995 par la société Netscape. C'est le premier langage de script de l'histoire et son objectif est d'incorporer son script dans les documents HTML afin d'y apporter des améliorations (Kumar, 2013). Il est le langage le plus connu et le plus employé par les développeurs sur les différentes pages Web.

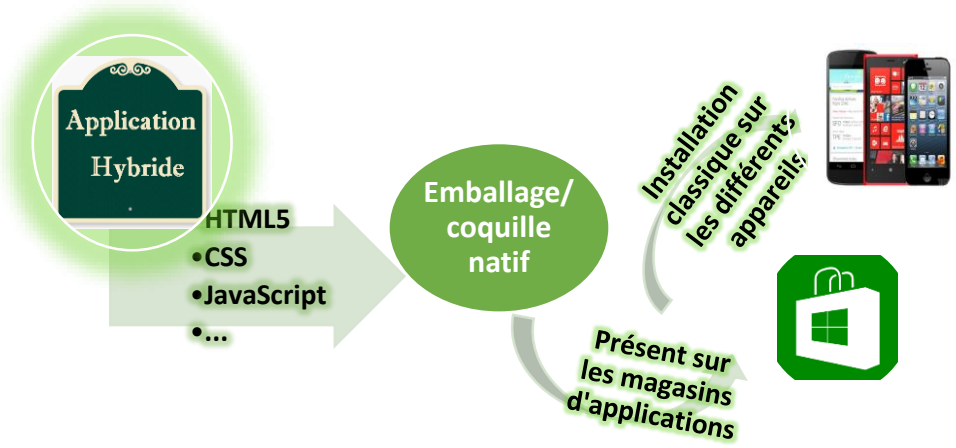
Son principal avantage étant de permettre d'exécuter des commandes au niveau du navigateur et non au niveau du serveur Web. Il permet d'ajouter de l'interactivité aux pages internet en manipulant le code de la page HTML. Par exemple, il est possible de créer des animations, de détecter le navigateur de l'utilisateur, de créer des cookies ou de détecter des données de localisation.

Il est important de ne pas le confondre avec le langage de développement Java qui nécessite un compilateur. C'est-à-dire qu'il aura besoin d'un programme informatique pour transformer son code source en langage cible. JavaScript, lui, ne nécessite pas de compilateur, mais a le désavantage d'être dépendant du navigateur Web puisque le script est intégré à même la page Web (Duffy, 2012). De par sa popularité, il est capable de tourner sur la plupart des navigateurs Web différents. Une autre caractéristique désirable est son niveau de protection. Sa conception ne permet aucune opération qui aurait pour but de mettre en danger la sécurité de l'utilisateur. Il n'est donc pas possible d'accéder à la base de données du client et de lui dérober des informations.

4.3 L'application hybride

L'application hybride est, comme son nom l'indique, un mélange entre l'application Web mobile et l'application native. C'est donc une application qui est programmée dans un langage courant comme le HTML5, JavaScript ou C# pour ensuite être compacté dans une application spécifique à la plateforme de destination (CHA, YUN 2013). Il existe deux niveaux d'application hybride. Le premier se contente d'être une application Web compactée dans un emballage natif tandis que le deuxième niveau traduit littéralement le langage Web classique en langage natif. Dans ce dernier niveau, l'application aura non seulement l'aspect d'une application native, mais également son comportement, son interface et ses fonctionnalités. Dans le premier niveau, il est également possible de se servir du langage informatique CSS3 et JAVASCRIPT pour que la présentation et l'interactivité de l'application correspondent à celles des différentes plateformes (Degani & al., 2011). Une métaphore amusante qui pourrait être employée pour ce type d'application serait celle d'un œuf. La coquille l'entourant représenterait la partie native de l'application qui a donc l'aspect de cette dernière. À l'intérieur, une application Web classique se cache derrière la coquille. Ceci permet à l'application de profiter des avantages de l'une et de l'autre. L'intégration à l'appareil, l'installation de l'application et sa distribution peuvent se faire via un marché d'applications.

Figure 12 – L'application hybride



4.3.1 Ses avantages

Tout d'abord, l'intégration à l'appareil est irréprochable. Nous avons accès à l'intégralité des fonctions de l'appareil complet, comme la liste de contacts, le GPS, l'appareil photo, etc (Stangarone, 2013).

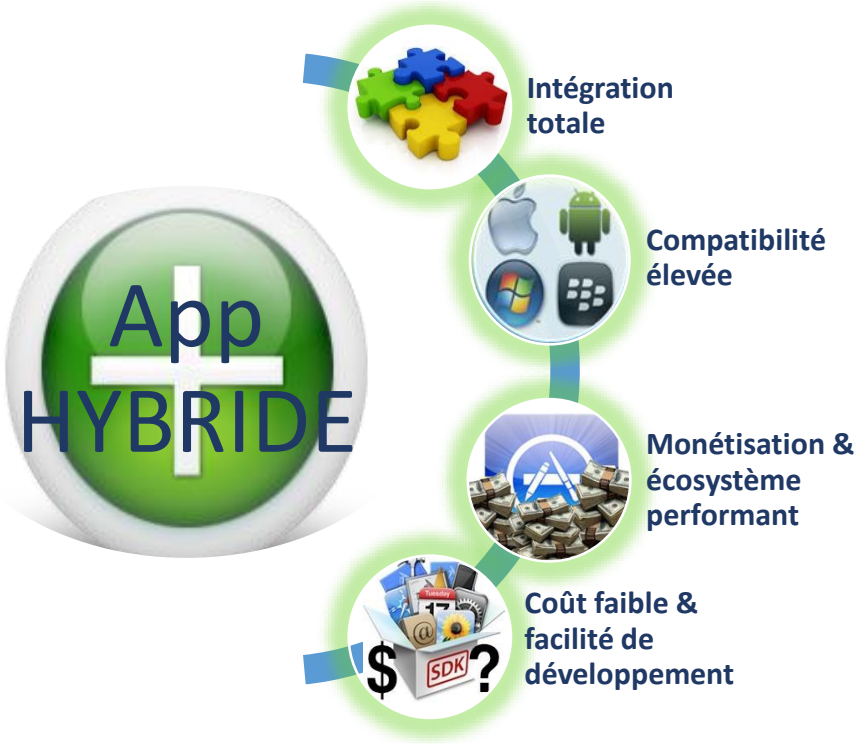
Ensuite, elle permet l'accès à la distribution classique des marchés d'applications. Ceci autorise donc le développeur à promouvoir et monétiser son application.

Elle autorise le développeur à choisir la partie du code qu'il souhaite commune sur toutes les plateformes ainsi qu'à laisser la quantité de codes spécifiques à la plateforme voulue (IBM corporation, 2012). Il peut dès lors choisir la quantité du code qui est incluse dans la partie native de l'application et celle qui devra être téléchargée à l'aide d'internet. Il est donc capable de mettre uniquement les premiers éléments servant à exécuter l'application sur le système de l'appareil dans le code natif de l'application. De telle manière, les différents contenus qui peuvent couramment varier seront mis à disposition à travers le Web sans devoir modifier ce contenu selon la plateforme d'exécution.

L'application hybride permet de développer une application multiplateforme sans engendrer de coût trop important (Anne Salaz & al, 2013).

De plus, l'application disposera globalement de la facilité de développement en utilisant ses connaissances en développement Web, tout en apportant l'aspect et la convivialité de chaque système d'exploitation.

Figure 13 – Avantages de l'application hybride



4.3.2 Ses inconvénients

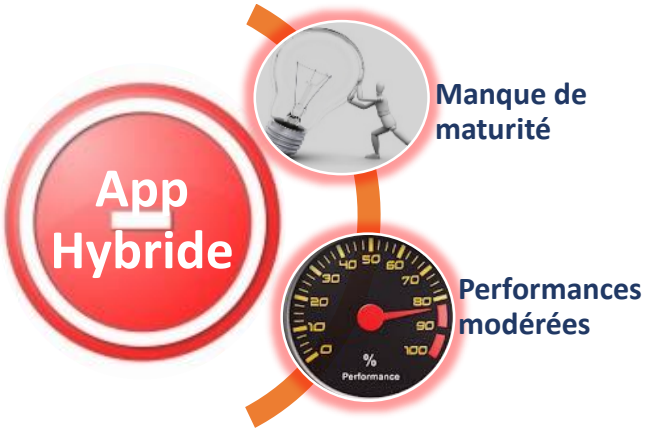
L'application hybride est trop parfaite pour n'avoir aucun inconvénient. Bien qu'elle apporte des qualités de la Web App et de l'application native, elle n'est pas sans défauts.

Au niveau graphique, elle garde principalement les capacités de la Web App et ne peut rivaliser avec une application native (DEGANI & al. 2011).

Le fait qu'elle soit également distribuée à travers un magasin d'applications engendre les inconvénients de ceux-ci. C'est-à-dire les différents problèmes d'approbations, de mises à jour ainsi que de maintenances.

Bien qu'elle soit développée en langage Web courant, le développeur requiert une certaine familiarité avec l'environnement mobile. Il est donc nécessaire de se familiariser avec le cadre de travail qu'impose le développement hybride.

Figure 14 – Inconvénients de l'application hybride



4.4 Conclusion

Après avoir comparé les grands axes permettant le développement d'applications mobiles, nous sommes en mesure de construire le tableau 2 à l'aide des annexes 2, 3 et 4. Celui-ci est basé sur une échelle de type « Likert » en cinq points allant de « médiocre » à « excellent » (MALHOTRA & al. 2010). Nous avons choisi cette échelle non comparative pour sa facilité de compréhension et de manipulation.

Premièrement, nous nous rendons rapidement compte que l'application native détient plus de notes se situant au niveau de la notation « excellent ». Cependant, elle dispose de trois résultats « médiocre » résultant d'un développement complexe, coûteux et inconciliable. Cette solution est extrêmement axée sur les performances, mais n'est cependant pas du tout adaptée à un développement multiplateforme. Il est donc normal de ne pas développer davantage cette solution qui ne peut pas répondre à notre question de recherche.

Deuxièmement, nous observons que l'application Web dispose de résultats totalement contraires à l'application native. En effet, lorsque l'application native obtient une note « médiocre », l'application Web reçoit la plus haute appréciation et inversement. L'application Web n'est donc pas satisfaisante sur de nombreux points liés aux performances et à son manque d'intégration (voir figure 15).

Elle permet malgré cela d'être portée aisément sur 100 % des plateformes mobiles. Ce type d'application répond donc à une partie de la question de recherche puisqu'il suffit d'écrire un code et le déployer sur un serveur pour que l'application soit accessible sur tous les appareils. Cependant, nous pouvons considérer que l'application Web mobile n'est pas assez performante pour suffire aux exigences des utilisateurs. Malgré sa forte compatibilité, elle ne permet pas d'atteindre les consommateurs et perd ainsi beaucoup de son intérêt de développement.

Finalement, nous constatons que l'application hybride se révèle comme étant le meilleur compromis. Elle ne détient aucune note « médiocre » et comble une grande partie des défauts de l'application Web sans avoir recours à une nette diminution de ses vertus. Elle permet un développement sur plusieurs plateformes, des performances raisonnables, mais également, un développement relativement simple. De par ses résultats, ce type d'application retient particulièrement notre attention et nécessite un approfondissement plus poussé.

Ce chapitre nous aura permis d'analyser les différentes méthodes possibles dans la création d'applications mobiles. Nous sommes maintenant capables de les distinguer, mais nous avons également trouvé la solution se rapprochant de notre paradigme. L'application hybride fera donc l'objet de nos futures recherches.

Figure 15 – Comparaison des interfaces graphiques Web et hybride/native de l'application *Facebook*

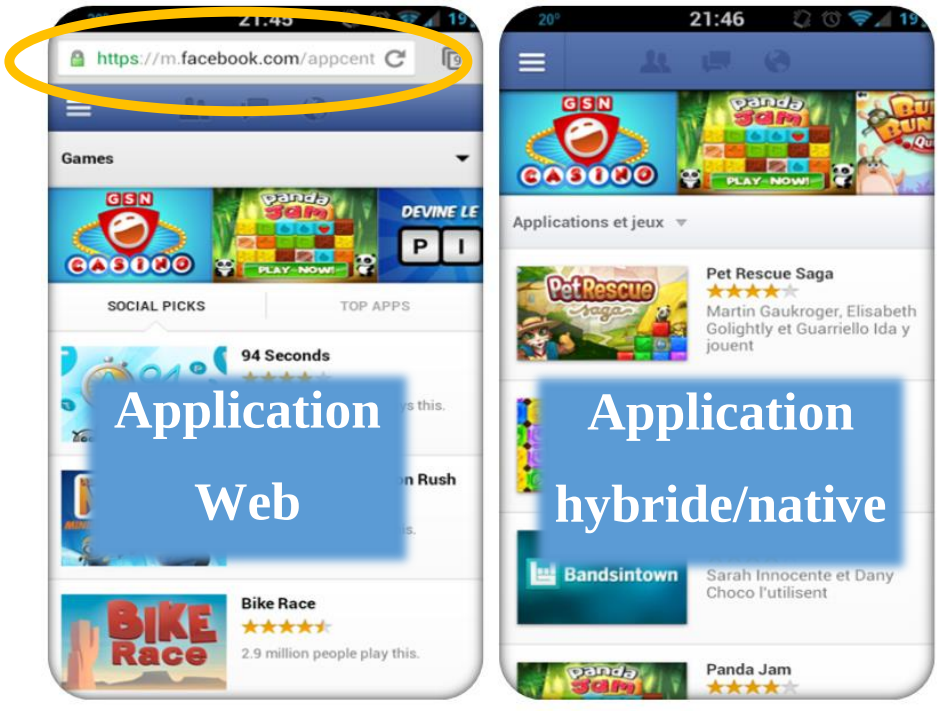


Tableau 2 – Comparaison des différentes solutions de développement mobiles

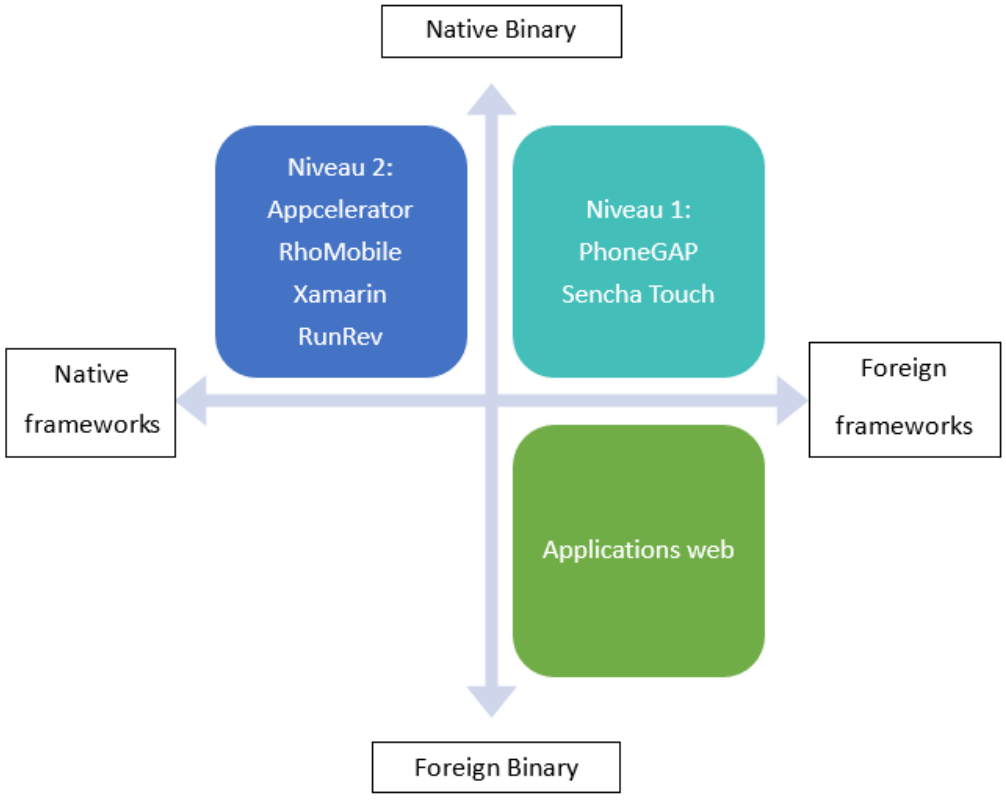
Solution / Caractéristiques	Natives	Web	Hybrides
Compatibilité			
Accès aux Fonctionnalités			
MAJ bibliothèques logicielles			
Qualité de l'interface utilisateur			
Temps de développement			
MAJ et maintenance			
Performances			
Graphisme			
Disponibilité des développeurs			
Intégration			
Sécurité			
Utilisation hors connexion			
Distribution			
Monétisation			
Popularité du Langage			

5 Analyse d'un échantillon d'outils hybrides

On ne peut nier que, depuis quelques années, le nombre de Framework destinés à procurer une solution cross-plateforme a relativement explosé. De grands acteurs et de nouvelles startups ont compris le potentiel qu'avait ce type d'outil et ceci explique la croissance rapide du marché. On assiste à une vraie lutte afin de proposer une solution rapide, innovante et aisée. De ce fait, de nombreux outils existent, mais ne permettent pas les mêmes fonctions, la même compatibilité, voire les mêmes performances. Une première différence se situe au niveau du langage utilisé pour le développement d'une application. Certains outils développeront en langages Web courants (HTML5, CSS, JavaScript), d'autres en langages plus spécifiques (C++, Ruby, JAVA..) (VISIONMOBILE, 2012).

Nous constatons également une distinction quant à la forme de l'intégration de l'application. Il existe, en effet, au minimum deux niveaux dissociables d'applications hybrides. Ceux-ci sont représentés dans la matrice présente dans la figure 16.

Figure 16 – Matrice Frameworks/Composants



Source : BALKAN 2012 pp276

Le premier niveau comprend les outils qui créent du « native binary », c'est-à-dire un pack qui peut être directement exécuté à travers le système d'exploitation (Balkan, 2012). L'installation se fait comme une application native, mais l'accès aux fonctionnalités de l'appareil reste géré par des interfaces de programmation liées à l'outil hybride employé et non aux systèmes d'exploitation natifs de l'appareil. De nombreux outils permettent ce type de développement et se retrouvent dans la partie supérieure droite de la matrice. Le deuxième niveau représente les outils qui utilisent à la fois le « native binary » et les Frameworks natifs. Ici, nous avons des applications qui disposent, non seulement de l'emballage natif de la plateforme mobile, mais proposent également l'accès natif des fonctionnalités de l'appareil. Grâce à ces différents Frameworks, l'application dispose de la culture, des gestes et symboles formels d'un système d'exploitation. Une application hybride de niveau deux sera donc nettement plus proche de l'application native pure. Il existe également un grand nombre d'outils permettant ce type de développement comme *Appcelerator Titanium*, *Xamarin* ou *RhoMobile*. Ceux-ci sont représentés dans la partie supérieure gauche de la matrice.

Maintenant que nous en savons un peu plus sur ces deux principaux niveaux représentant la portion native attribuée selon l'outil de développement, nous pouvons sélectionner un échantillon d'outils représentant non seulement ces deux niveaux, mais également des outils prometteurs quant à leur avenir. Pour ce faire, il était primordial de privilégier des outils ayant déjà fait leurs preuves et disposant de critères de sélection bien précis. Ces derniers sont leur maturité, la quantité de développeurs présents (voir annexe 5), la taille de la communauté ainsi que le nombre d'ouvrages attribués à l'outil. Après avoir minutieusement observé ces différents critères, nous en avons retenu six : RhoMobile, PhoneGap, Appcelerator Titanium, RunRev, Sencha Touch et finalement Xamarin.

5.1 RhoMobile

RhoMobile est une suite d'outils « open source » développée en 2008 et rachetée par Motorola en 2011 (VISIONMOBILE, 2012). Il procure un Framework cross-plateforme compatible sur presque toutes les plateformes mobiles présentes sur le marché (voir annexe 6). Le projet de RhoMobile est clairement d'écrire une application pour qu'elle soit portable sur les différents appareils mobiles possibles et prône donc le paradigme « *Write once, run everywhere* ». La suite d'outils est exécutable à partir de Linux, Windows ou MAC.

RhoMobile comprend RhoConnect qui permet de synchroniser les données de l'application avec celles disponibles sur un site Web. Quant à RhoStudio, c'est un environnement de développement se basant sur le célèbre IDE Eclipse. Finalement, RhoElements est la plateforme de développement HTML5 de la suite RhoMobile. Le développement se fait à l'aide de langages classiques comme le HTML5, CSS ou JavaScript, mais également à l'aide du langage de programmation libre Ruby (DEGANI & al. 2011). Celui-ci autorise l'écriture d'actions de l'application tout en permettant de modifier facilement celle-ci à tout moment. À noter que ce langage n'est pas à l'avantage de RhoMobile puisqu'il n'est pas très répandu.

Ce Framework a la particularité de disposer de l'approche « Model View Controller » (Cimitile & al, 2011). Celle-ci permet de séparer les données logiques de l'affichage, ainsi que des actions. Grâce à cette séparation, un développeur voulant ajouter du nouveau contenu à son application ne sera pas contraint de modifier la structure complète de son application. Un autre avantage que d'être capable de travailler sur une partie sans toucher aux autres est de mieux s'organiser pour la répartition des tâches entre développeurs. Chacun peut dorénavant se concentrer sur ce qu'il gère le mieux. Un webdesigner peut s'occuper de la partie concernant la vue tandis que deux autres développeurs se répartissent les actions et les données sans pour autant devoir collaborer. Il est donc plus facile de comprendre et de maintenir une application.

RhoMobile vise le segment des entreprises et des applications Business-to-Business tout en permettant le portage de celles-ci sur toutes les plus grosses plateformes mobiles présentes sur le marché.

Une fois les différentes parties terminées, elles sont compilées dans un exécutable natif afin d'être lancées sur un appareil physique ou virtuel. Techniquement parlant, RhoMobile permet de compiler l'application en se servant d'une machine virtuelle (Degani & al, 2011). Une fois le code source écrit, celui-ci se lance à travers un interpréteur du langage Ruby. C'est un outil informatique permettant d'interpréter le code Ruby plutôt que de le convertir dans le langage propre à l'appareil. Ce dernier se retrouve donc intégré à l'application native développée et permet également d'avoir un accès aux différentes fonctionnalités de l'appareil via les nombreuses interfaces de programmation supportées. Il est également possible de reproduire l'interface utilisateur native des divers appareils via différentes bibliothèques logicielles. Nous avons donc un outil de niveau 1 puisque celui-ci n'utilise pas directement les composants natifs des différents appareils.

Compatibilité mobile	Langage (s)	Fonctionnement	Performances
-iOS -Android -Windows phone 7 -BlackBerry 5/6 -Symbian -WebOS	HTML5, JavaScript, CSS, Ruby	Machine virtuelle	Limitées
Interface native	**Support**	**Licence**	**Applications types**
Non	Bon	Gratuite sous licence MIT	Super Trainer HQ, Pilsner Urquell

5.2 PhoneGap

PhoneGap est un outil « open source » permettant de créer des applications en utilisant HTML5, CSS3 et JavaScript (Wargo 2012). Cet outil a été conçu en 2005 par Nitobi et racheté par Adobe Systems. Il est disponible gratuitement au téléchargement pour tous les projets, qu'ils soient « open source » ou commerciaux. Selon une étude de VISION MOBILE en 2012, il serait l'outil le plus répandu auprès des développeurs recherchant une solution multiplateforme.

Sa fonction n'est pas de convertir ces différents codes Web en codes natifs, mais de les inclure dans une application ayant l'aspect d'une application native. Cette dernière est en fait uniquement basée sur des technologies Web. L'application finie fonctionne via un navigateur incorporé qui permet d'exécuter du code JavaScript (Degani & al 2011). C'est typiquement ce que l'on peut appeler une application hybride de niveau 1. Son principe est basé sur des interfaces de programmation JavaScript fournies aux navigateurs Web standards des différents appareils. C'est de cette manière que l'application a accès aux fonctionnalités natives telles que l'accéléromètre, l'appareil photo ou le système de fichiers. Le code source CSS/HTML/JavaScript doit donc être incorporé dans une application native. Cette manipulation est faisable grâce à un composant permettant une vue Web dans les applications natives. Pour ce faire, ce composant doit être présent dans les différents SDK des différentes plateformes. L'environnement de développement intégré peut être choisi par le développeur malgré que l'environnement Eclipse soit conseillé (VISION MOBILE, 2012).

Pour développer une application avec PhoneGap, il est nécessaire d'avoir des connaissances en HTML5, CSS3 ainsi que JavaScript.

Tout ce qui est design, rendu, modèle est géré par les deux premiers langages. Le langage JavaScript est, lui, destiné à s'occuper de la logique de l'application (PHONGEGAP, 2013).

Le gros avantage de cet outil est qu'il permet de récupérer le code de base employé pour les applications d'ordinateurs classiques afin de l'utiliser pour d'autres types d'appareils tels que les tablettes ou smartphones. Il autorise l'accès à toutes les fonctions des appareils tournant sous Android, Apple et Windows Phone (Lute, 2012). Mises à part quelques fonctionnalités, il permettra également le port sur toutes les autres plateformes disponibles sur le marché telles que BlackBerry et Symbian (voir annexe 7).

La facilité a bien entendu un prix. JavaScript est connu pour être souvent lent, non naturel et peut cacher de nombreuses erreurs (Fairhead, 2011). Les applications créées avec cet outil ne sont pas à la hauteur de la finition ainsi que de la fluidité d'une application hybride de niveau 2 et loin d'une application compilée en code natif. Bien que CSS3 permette aux applications d'avoir une interface identique aux applications natives, elles n'auront que l'aspect et ne se comporteront pas comme telles. Il est dès lors conseillé de ne pas perdre son temps à faire ressembler son application à la culture de la plateforme, mais bien de créer son propre monde (Balkan, 2012).

Compatibilité mobile	Langage (s)	Fonctionnement	Performances
-iOS -Android -Windows phone 7/8 -BlackBerry 5/6 -Symbian -Palm -WebOS -Tizen	HTML5, JavaScript, CSS	Langage WEB compilé	Limitées
Interface native	**Support**	**Licence**	**Application type**
Non	Bon	Gratuit	NetFlix, Linkedin

5.3 Appcelerator Titanium

Appcelerator Titanium est une plateforme de développement d'applications « open source » pour ordinateurs classiques, tablettes et smartphones (Degani et al 2011).

Cette plateforme de développement a été créée par Appcelerator Inc. en 2008 et propose une solution de développement multiplateforme basée sur des technologies Web. Elle est, pour l'instant, uniquement compatible avec les appareils mobiles Apple et Android (voir annexe 8).

Son objectif est de permettre aux développeurs d'appliquer leurs compétences acquises afin de créer une application native tournant sur les différentes plateformes supportées (Whinnery, 2012). La différence avec les autres solutions basées sur des langages Web courants est qu'elle ne permet pas de lancer simplement du code JavaScript à travers l'application, mais de le compiler avec le langage natif des différents appareils. Ce type d'application est hybride de niveau 2. C'est-à-dire qu'elle emploie, non seulement, le « native binary », mais également, les Frameworks propres à chaque plateforme (Balkan, 2012). À l'instar des différents outils hybrides, il n'introduit pas de navigateur dans la compilation de son application. Son interface de programmation procure la majorité des éléments de l'interface utilisateur et est ensuite convertie en langage natif lors de l'exécution de l'application sur la plateforme voulue. De par ses fonctionnalités, cet outil cible les applications interactives, riches en informations et en médias.

Le Framework Appcelerator Titanium fonctionne grâce à une interface de programmation JavaScript qui contient les fonctionnalités spécifiques aux différentes plateformes ainsi que les technologies natives de celles-ci. Les fonctionnalités natives sont pleinement intégrées à l'application et les performances observées sont supérieures à une technologie où l'application est simplement du code Web emballé dans une coque native (Whinnery, 2012). Cette approche est également intéressante, car elle ne permet pas seulement de dessiner une application ressemblante aux applications natives, via le langage CSS3 par exemple, mais également d'y intégrer l'interface utilisateur propre aux différents terminaux (Lutes, 2012). Le code JavaScript écrit par le développeur est donc bien interprété sur le téléphone, mais la réalisation des opérations natives, elle, est déléguée aux parties compilées du Framework. L'expérience utilisateur s'y retrouve complètement améliorée et est semblable à l'application native.

Ce Framework accepte donc le code JavaScript pour le combiner avec son propre code qui est, lui, écrit dans le langage natif de l'appareil visé par le développeur (DEGANI & al 2011).

Ce n'est pas pour autant que l'application est native. Elle n'est, en effet, pas compilée dans le langage de la plateforme, mais le code JavaScript y est transformé lors de l'utilisation de l'application.

Un autre point positif pour ce Framework est qu'il permet, une fois l'application terminée, de la compacter, de la tester pour enfin la distribuer directement à travers le magasin d'applications correspondant à la plateforme.

Cependant, il peut y avoir un manque de flexibilité et de solidité. L'application sera entièrement dépendante de la qualité du code fourni et de sa correspondance avec le Framework d'abstraction d'Appcelerator Titanium. Qui plus est, le développeur n'a pas un accès direct au code de l'application et il peut être très compliqué de retrouver une erreur. Une fois compilé, il devra également connaître quelques notions du langage natif pour s'y retrouver. D'autre part, les plateformes sont couramment mises à jour et de nouvelles interfaces de programmation font apparition. Il faut un temps pour que l'équipe de développeurs Appcelerator les implémente dans le Framework. Ce temps d'adaptation par rapport aux applications purement natives peut s'avérer long et l'on risque de prendre du retard par rapport à la concurrence.

Compatibilité mobile	Langage (s)	Fonctionnement	Performances
-iOS -Android	HTML5, JavaScript, CSS	Code Web interprété par un moteur d'exécution Web incorporé dans l'application + opérations natives interprétées nativement pas l'appareil.	Élevées
Interface native	**Support**	**Licence**	**Application type**
oui	Bon	Gratuit (options payantes)	NBC iPad, LEGOLAND, Zipcar

5.4 Xamarin

En 2009, Xamarin développe MonoTouch pour iOS et Mono for Android afin de proposer une solution multiplateforme. Ces outils sont basés sur le célèbre projet libre : « Mono ». Aujourd'hui ce Framework ne se nomme plus que « Xamarin » pour plus de facilité (Bright, 2013).

MonoTouch est un kit de développement logiciel permettant de développer des applications mobiles pour iOS et Android en utilisant le langage .NET et C#. Sa grande particularité consiste dans le fait que l'application créée à l'aide de Xamarin est une application « native » (TAFT 2012).

En effet, Xamarin procure les différentes interfaces de programmation ainsi qu'utilisateurs liées à chaque plateforme. La différence est que Xamarin ne se contente pas d'employer une couche d'abstraction matérielle pour atteindre les fonctionnalités de l'appareil.

C'est-à-dire que ce n'est pas via un utilitaire logiciel que le développeur accède aux fonctionnalités de l'appareil, mais grâce à un mécanisme permettant de lier le code directement aux fonctionnalités natives de l'appareil. Cette différence est énorme par rapport aux autres outils de développement hybrides. Cela veut dire que l'application créée pour iOS, par exemple, ne contient plus le langage classique .NET et C#, mais bien le code Objective-C, propre à la plateforme de ce cas-ci (THOMPSON, 2013).

Il est possible de développer à l'aide de l'environnement propre à Xamarin, mais il est également capable d'être utilisé avec l'environnement de développement de Microsoft : Visual Studio. Il a l'avantage d'être le seul outil capable de développer une application native iOS à l'aide de cet environnement de développement, mais surtout sur le système d'exploitation Windows. Les différents outils de Visual Studio sont donc pleinement utilisables et il est possible de tester et déboguer son application via ce dernier, que ce soit directement sur le matériel ou via le simulateur de Visual Studio. Malheureusement le développeur nécessitera tout de même l'accès à un Mac afin de déployer son application sur l'App Store.

Compatibilité mobile	Langage (s)	Fonctionnement	Performances
-iOS -Android	HTML5, JavaScript, CSS	Natif	Excellentes
Interface native	**Support**	**Licence**	**Application type**
Oui	Très Bon	Gratuite, payante dans sa version commerciale	AOL, HP, Target, Monster

5.5 RunRev - LiveCode

RunRev LiveCode a été fondé en 1997 par Runtime Revolution. Celui-ci est basé sur MetaCard, un outil déjà disponible pour créer des applications Windows et Macintosh (RUNREV.COM, n.d.).

Depuis 2010, le programme s'est également spécialisé dans le développement d'applications mobiles (VISION MOBILE, 2012). En 2013, LiveCode Community Edition est devenu un outil de développement gratuit. Il est identique à la version commerciale à l'exception de la monétisation de l'application créée. En effet, si celle-ci est créée à l'aide de la version Community, elle devra être « open source » et ne pourra pas être déployée à travers l'App Store d'Apple.

Le langage utilisé est « LiveCode » propre à RunRev. Ce dernier le décrit comme un langage naturel, plus facile à apprendre et à utiliser que d'autres langages traditionnels. C'est-à-dire que les lignes de codes sont fort ressemblantes à de l'anglais traditionnel. Ce langage est orienté créativité et vise une clientèle recherchant un moyen facile d'apprendre à développer sur plusieurs plateformes (VISWANATHAN 2013).

L'outil est très complet. Il permet de glisser/déposer facilement les différents éléments formant l'interface de l'application. Un testeur et un débogueur sont disponibles dans l'environnement de développement de RunRev afin de tester et de réparer à tout moment le programme en cours de création (RunRev.com). Par ailleurs, il est possible de faire des changements dans le code ou l'interface lorsque l'application est en train d'être testée.

À ce jour, il permet le développement d'applications mobiles uniquement sur Android et iOS. Le code créé est ensuite intégré dans un interpréteur d'exécutions qui est lui-même intégré à un exécutable natif. Son environnement de développement intégré procure différents outils afin de s'occuper du design, de tester et déboguer l'application.

LiveCode permet aux développeurs d'avoir accès aux différentes fonctionnalités natives. Le niveau graphique des applications créées reste raisonnable, mais limité une fois que l'on s'attaque à de la 3D.

RunRev est donc un outil puissant pour les personnes créatives qui recherchent à présenter leur application sur plusieurs systèmes d'exploitation. Cet outil pourrait cependant intégrer davantage les interfaces de programmation natives.

Compatibilité mobile	Langage (s)	Fonctionnement	Performances
-iOS -Android	LiveCode	Natif	Très bonnes
Interface native	**Support**	**Licence**	**Application type**
Oui	Très bon	Gratuite	The Forest guide, EuroTalk

5.6 Sencha Touch

Sencha Touch a vu le jour en 2010 suite au rassemblement de trois bibliothèques d'interfaces utilisateurs célèbres : Ext JS, JQtouch et Raphaël (Abraham, 2010). Ce Framework a été spécialement conçu pour développer à l'aide de JavaScript, HTML5 et CSS (SENCHA.COM). Il contient un set de composants lié à l'interface utilisateur permettant l'optimisation totale des écrans tactiles. Il est principalement utilisé pour développer l'interface d'applications mobiles basées sur les technologies Web.

Il permet de développer rapidement des applications fonctionnant sur Android, iOS, BlackBerry et Windows Phone. Malgré que l'application soit basée sur une technologie Web, Sencha Touch lui offre une interface utilisateur riche et lui permet de ressembler complètement à une application native. En effet, ce Framework offre la possibilité très pratique de choisir un thème en fonction de l'appareil destiné.

L'environnement de développement intégré est orienté sur le design, la qualité et la facilité d'utilisation. Tout comme RhoMobile, il encourage le développeur à employer une architecture « Mobile View Controller ».

Sencha a récemment voulu démontrer la puissance du HTML5 en créant « Fastbook ». Ce dernier démontre le fait que si l'application HTML5 de *Facebook* n'a pas eu de succès, c'est parce qu'elle n'a pas été écrite correctement (Avins & al. 2012). Après avoir comparé la version HTML5 officielle de la version de Sencha, il en ressort d'énormes différences de performances et de fluidité. Nous avons pu constater un résultat très proche de l'application native du célèbre réseau social *Facebook.*

Sencha Touch est disponible gratuitement pour les développeurs désireux de construire une application « open source ». Il existe également une version prémium permettant de monétiser son application (SENCHA.COM, n.d.).

Compatibilité	Langage (s)	Fonctionnement	Performances
-iOS -Android -Windows Phone -BlackBerry	HTML5, JavaScript, CSS	Web compilé	Limitées
Interface native	**Support**	**Licence**	**Application type**
Oui	Bon	Gratuite	XERO, Direct TV app

5.7 Conclusion

Nous venons d'en apprendre un peu plus sur ces différents outils méritant leur titre d'applications multiplateformes. Nous avons pu constater que, malgré leur but commun de fournir une solution facile et rapide de développement, ces outils sont bien distincts.

À l'aide des différents critères de ces outils de développement hybrides, nous sommes capables de fournir différentes variables permettant leur comparaison. Il est important de préciser que le tableau 3 a également été réalisé en examinant minutieusement différentes études comparatives citées dans l'annexe 9.

Nous constatons, tout d'abord, qu'ils ne permettent pas tous une compatibilité complète avec les différents systèmes d'exploitation mobiles. De plus, ils divergent fortement en termes de fonctionnement. Via ces deux critères, nous sommes capables de définir deux types distincts parmi ces outils étudiés.

Premièrement, il y a les procédés se basant principalement sur les technologies Web. C'est-à-dire les solutions qui incorporent le code Web dans un exécutable natif et où l'application est lancée à travers un navigateur Web incorporé. Ces solutions permetten une compatibilité maximale et ne nécessitent aucune compétence dans les langages natifs des systèmes d'exploitation. Cependant, les performances ne sont pas excellentes et l'interface utilisateur ne réagit pas comme elle le devrait. Ce type d'outil rentre dans notre paradigme puisqu'il permet le développement sur les quatre systèmes d'exploitation dominant le marché.

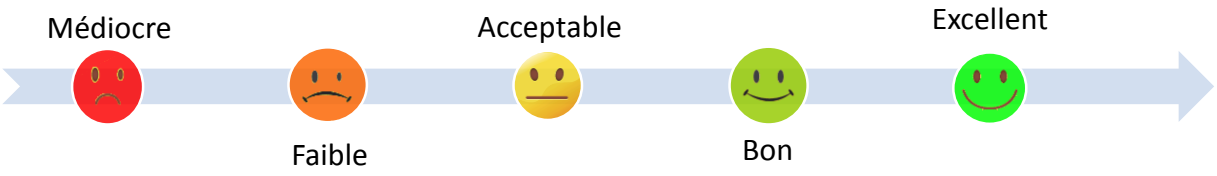
Deuxièmement, nous avons les solutions permettant de générer du code natif, soit à l'exécution de l'application sur le smartphone, soit lors de la compilation. Les performances sont supérieures grâce à l'utilisation des composants natifs formant l'interface utilisateur.

Malheureusement, ces outils perdent en contrepartie au niveau de la compatibilité multiplateforme et ne permettent pas de répondre à notre question de recherche. Cependant, ces outils sont en période de croissance selon le cycle de vie d'un produit et nous savons d'ores et déjà qu'une compatibilité plus avancée est prévue dans les années à venir (Griesel & al. 2004).

Il est également intéressant d'observer qu'il n'y a que très peu de caractéristiques obtenant une notation « faible » et aucune recevant la moins bonne note « médiocre ». Ceci est dû au fait que ce type d'outil est un bon compromis entre l'application Web et l'application native. Dès lors, il est normal qu'ils fassent le maximum pour obtenir un outil conciliant dans pratiquement tous les domaines.

Tableau 3 – Comparaison des différents outils de développement hybride de notre échantillon

Outil / Caractéristiques	RhoMobile	PhoneGap	Appcelerator T.	RunRev	Sencha	Xamarin
Compatibilité	🙂	🙂	😐	😐	🙂	😐
Performances	😐	☹	🙂	🙂	☹	🙂
Popularité du langage	☹	🙂	🙂	🙂	🙂	🙂
% code partagé entre les plateformes	🙂	🙂	🙂	🙂	🙂	😐
Interface utilisateur	🙂	☹	🙂	🙂	😐	🙂
Communauté & nombre de guides utilisateurs	😐	🙂	🙂	😐	🙂	🙂
Coût/temps d'apprentissage	😐	🙂	🙂	🙂	🙂	😐
Coût	🙂	🙂	🙂	🙂	🙂	🙂



Tout au long de ce chapitre, nous avons décomposé différents outils hybrides. Cette recherche nous permet d'observer différentes qualités et défauts selon l'outil choisi. Nous pouvons dès à présent analyser les critères liés non seulement à ces outils, mais également au monde des applications mobiles.

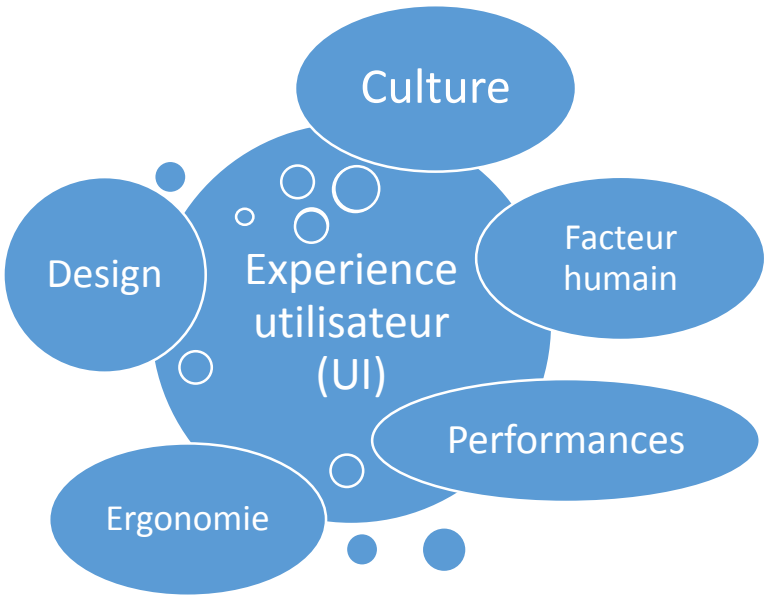
6 Quels critères une entreprise/un développeur doit-elle/il prendre en compte pour choisir une solution de développement ?

Après avoir comparé ces différents outils dans leurs caractéristiques détaillées, nous sommes prêts pour définir les critères à prendre en compte lors de la création d'une application. Via la description de ces caractéristiques, nous serons en mesure de classifier les outils de développement en fonction de leurs capacités.

6.1 L'expérience utilisateur

L'expérience utilisateur est ce que ressent ce dernier lorsqu'il lance et interagit avec l'application ou le site internet. L'interaction de l'utilisateur avec l'application est ce qu'il y a de plus important lors de la création d'une application.

Figure 17 - Relation expérience utilisateur



Pour satisfaire cette expérience, il faut se pencher sur la facilité d'utilisation, la perception de la valeur du système, l'utilité, l'efficience à performer dans différentes tâches, etc. Lorsque nous souhaitons évaluer l'expérience utilisateur, il faut se mettre à la place de celui-ci.

C’est-à-dire utiliser l’application développée dans les différentes circonstances possibles et imaginables que l’utilisateur risque de rencontrer. L’expérience utilisateur est maximisée lorsque la manipulation de l’application est facile et joviale. Il faut constamment penser à l’utilisateur lors de la création d’un produit, car ce dernier est le client final (Balkan 2012). Au moment de choisir le type d’application que le développeur va créer, il doit penser aux problèmes que l’utilisateur risque de rencontrer plutôt que ceux auxquels il doit faire face.

Il faut savoir que les utilisateurs sont différents selon l’appareil et la plateforme qu’ils emploient. En effet, chaque plateforme dispose d’une certaine culture, de normes et de certaines conventions (Balkan 2012). Ceci est particulièrement vrai pour le système d’exploitation d’Apple qui dispose d’une plateforme très fermée sur elle-même. L’utilisateur est habitué aux différentes manipulations, boutons, interfaces et interactions avec les différentes applications fournies avec la plateforme. Cette culture se retrouve sur tous les appareils de la marque et permet à Apple d’avoir un contrôle absolu sur l’expérience de l’utilisateur.

Un utilisateur n’a aucun intérêt à avoir une application disponible sur les différentes plateformes, ce qui l’intéresse est le fait que celle-ci fonctionne correctement sur son smartphone. Pour ce faire, il est intelligent d’insérer dans son application la culture, les normes ainsi que les conventions déjà présentes sur les différents appareils. L’utilisateur de l’application ayant un comportement natif s’y retrouvera plus facilement dès sa première utilisation et aura le sentiment de ne pas avoir quitté son environnement habituel. Il est dès lors naturel de penser qu’au plus la culture d’une plateforme est consistante, au plus il est bénéfique de créer son application avec des outils natifs qui permettront de reproduire cette culture à travers celle-ci.

Les performances d’une application font également partie de l’expérience utilisateur. Elle se retrouve sous deux formes différentes qui sont le chargement et le rendu (Escallier, 2011). La première est le processus qui implique une demande au serveur, la génération d’une réponse pour finalement l’envoyer à l’appareil lui-même. En général, les paramètres pouvant affecter le chargement sont la quantité à télécharger sur le serveur, le type de connexion impliquée et le comportement des composants externes. Le rendu est, lui, un processus qui implique la forme de la mise à disposition de la réponse venant du serveur. Les paramètres qui affectent principalement ce dernier sont le type de plateforme, la configuration de l’appareil utilisé ainsi que le comportement des utilisateurs finaux. C’est très important pour le succès d’une

application. Pour mesurer les performances d'une application, il suffit d'observer le temps de réaction lorsque l'on clique sur une action, que l'on effectue une rotation avec l'appareil mobile ou que l'on effectue différents gestes que le système doit prendre en compte.

L'interface personne-machine détient également un rôle important à la fois dans la création d'une application et la satisfaction de l'utilisateur. Ce sont les lignes de conduite que les développeurs doivent suivre pour développer une application cohérente autant dans l'aspect que dans les différentes fonctions et interfaces graphiques sur une plateforme donnée (ELEMENTARYOS, 2012). En d'autres mots, ce sont des recommandations que les différents éditeurs des systèmes d'exploitation fournissent aux développeurs afin d'avoir un environnement cohérent pour les utilisateurs. Ces différentes règles peuvent être liées à l'ergonomie telles que la police de caractères, les icônes, les boutons, les fenêtres et les menus utilisés sur un système d'exploitation (Balkan, 2012). Elles peuvent également être liées à l'interaction de l'utilisateur avec l'application comme pour des actions telles que la saisie d'informations ou les différentes manipulations courantes sur une plateforme. Elles ont pour but d'améliorer la prise en main des applications par les utilisateurs finaux. Les développeurs qui suivront ces différentes pratiques auront un avantage concurrentiel conséquent. En effet, les utilisateurs pourront utiliser leurs connaissances acquises pour la nouvelle application créée et auront l'agréable surprise de rester dans un environnement qu'ils connaissent.

Expérience utilisateur	Performances: Fluidité, rendu, chargement,...
	Facilité d'utilisation
	Design
	Culture de la plateforme : Boutons, interfaces, menus, manipulations,...
	Niveau graphique
	Intégration : Icône, notifications,...

6.2 Le genre d'application

Le choix de développement d'une application doit naturellement être fait en fonction du type d'application que nous souhaitons créer. Un journal en ligne n'aura pas les mêmes besoins qu'un jeu 3D par exemple. Afin de mieux distinguer le type d'application que l'on désire créer, nous allons nous servir du continuum « Documents à applications » et du niveau d'immersion que l'on souhaite lui attribuer.

Le continuum « Document-Applications » est une façon séduisante de classifier des applications. Celles-ci peuvent être orientées sur leurs contenus, mais également sur leurs comportements (Balkan, 2012). Lorsque l'on est capable d'utiliser une application après y avoir retiré tout ce qui se rapporte au comportement, nous avons une application focalisée sur le contenu. Si l'application n'a plus de sens sans ses interactivités, celle-ci est concentrée sur le comportement. Bien entendu une application n'est pas forcément l'un ou l'autre, mais comprendre où l'application se situe sur ce continuum peut aider le développeur à se focaliser sur l'une ou l'autre technologie (voir figure 18).

Si une application se situe davantage du côté du contenu, on favorisera une amélioration progressive (Balkan, 2012). C'est-à-dire qu'on développera l'application en séparant complètement le contenu destiné à l'utilisateur et la présentation de celle-ci. Grâce à cette technique, le développeur sera capable de présenter son contenu basique à un grand nombre d'utilisateurs et d'améliorer progressivement l'affichage en fonction du terminal de l'utilisateur. Une entreprise diffusant du contenu pour générer son revenu nécessite d'atteindre le plus de clients possible. Ce n'est pas en créant une application native pour une seule plateforme qu'elle parviendra à partager son information avec tous les utilisateurs. Il est donc plus intelligent pour ce type d'entreprise de commencer par diffuser son contenu le plus simplement possible pour ensuite se spécialiser selon la taille de l'écran, sur les fonctionnalités et finalement à la culture et capacité des différents appareils.

Figure 18 – Le continuum Documents à applications

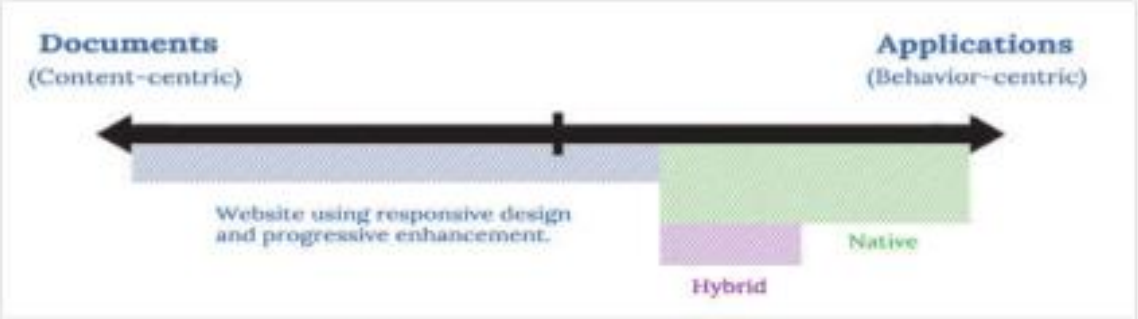


Figure 9.12. The documents-to-applications continuum.

Source: BALKAN A. (2012) Mobile Considerations in UX Design : Web or Native, Smashing Book #3 – Redesign the web pp274

Pour ce qui est de l'application centrée sur le comportement, il est primordial d'optimiser l'expérience utilisateur. Pour ce faire, il faut prendre en compte chaque appareil et chaque plateforme existante. Bien que l'on ne soit pas capable d'atteindre promptement chaque appareil, l'entreprise détiendra la satisfaction de la fraction d'utilisateur possible. Tant bien qu'il soit possible de faire tourner son application directement sur chaque appareil, elle ne sera pas sensible à la culture de ceux-ci et ne pourra pas tourner de manière optimale sur chacun d'eux. Au final, les utilisateurs des différents terminaux ne seront pas satisfaits et se tourneront probablement vers un concurrent prenant davantage en compte la qualité de l'expérience utilisateur.

Les applications peuvent également être cataloguées en fonction du fait qu'elles soient immersives ou non (Balkan, 2012). L'application immersive est celle qui aura un avantage à faire plonger l'utilisateur dans son propre monde, sa propre culture et donc sa propre interface. Des applications telles que les jeux ou ebooks, par exemple, sont souvent considérées comme immersives. Celles-ci n'utilisent que très rarement les composants natifs, car elles ont tout intérêt à créer leurs propres interactions et cultures.

L'application non immersive est le type d'application qui ne se doit pas de plonger l'utilisateur dans un nouvel univers. De nombreuses applications bénéficient de l'incorporation des différents standards natifs de l'interface utilisateur tels que les applications liées à la productivité. Elles ont donc intérêt à s'associer au système d'exploitation et à la culture de l'appareil de destination.

6.3 La couverture espérée

La couverture est un point étudié puisqu'il représente la cible que l'on souhaite atteindre. Il y a de nombreux systèmes d'exploitation sur le marché. Nous savons d'ores et déjà que les plateformes d'Apple et de Google représentent à elles seules plus de 92 % du secteur (Gartner, 2012). Notre question de recherche se limite d'ailleurs à ces différentes plateformes. Cependant, une compatibilité plus poussée ne peut être que bénéfique.

6.4 L'investissement financier

Ce critère représente les différents aspects financiers du développement d'une application. Cela va des coûts de conception jusqu'aux coûts de déploiement.

De fait, créer une application engendre différents frais qui doivent être pris en compte avant même de commencer quoi que ce soit (Microsoft, 2012). Nous avons, premièrement, le coût de conception. Celui-ci est lié au prix de l'outil utilisé, des honoraires du développeur requis ainsi que du temps nécessaire à la création. Effectivement, un développeur disposant de compétences dans des langages et environnements de développement complexe représentera un coût plus conséquent qu'un développeur Web. Nous avons ensuite, le coût lié au déploiement et à la maintenance. Comme expliqué précédemment, une application subit couramment des mises à jour et celles-ci peuvent être plus ou moins coûteuses.

6.5 La facilité de développement

Nous avons ici un critère très complexe puisqu'il peut dépendre de beaucoup de choses. Premièrement, nous avons les compétences requises chez le développeur. Si ce dernier a uniquement étudié des langages Web classiques, il serait préférable de trouver une solution de développement mobile permettant le développement dans ces langages. Les langages classiques permettent donc une facilitation du développement d'application (Balkan, 2012). Ensuite, nous avons la facilité de déploiement. Avant d'être disponible pour l'utilisateur, il faut que celle-ci lui soit proposée. Le déploiement sur certains magasins d'applications peut être laborieux et il serait donc plus facile de la distribuer via un serveur Web. Finalement, nous avons la facilité de maintenance. Il peut être très pénible puisqu'il est nécessaire de faire des modifications sur les différentes versions de notre application ainsi que d'effectuer à nouveau le déploiement sur les appareils de l'utilisateur. À titre d'exemple, l'application Web permettra la maintenance la plus aisée puisqu'il suffit de modifier une seule fois le code et de la déployer directement sur tous les appareils.

6.6 Conclusion

Ce chapitre a été développé en fonction des grands critères de sélection lors de la création d'une application. Ces critères ont chacun une grande importance et leur définition nous permet de créer deux outils offrant la possibilité de se diriger vers différentes méthodes de développement que nous retrouverons dans le chapitre 7.

7 Outils d'orientation à la décision du développement d'applications mobiles

Dans ce chapitre, nous allons tenter de construire deux outils d'orientation. Le premier nous permettra d'orienter le choix du type de développement par rapport aux grands axes. C'est-à-dire se poser la question de savoir s'il est préférable de développer une application Web, hybride ou native selon une série de critères. Nous concevrons ensuite un deuxième outil approuvant le choix d'une technologie hybride. Nous ne développerons pas davantage les différentes solutions natives ou Web puisque celles-ci sont fort limitées en choix et les outils ne seraient pas très efficaces.

7.1 Outil permettant la décision d'une solution générale de développement

Grâce aux critères principaux que nous venons d'expliquer, nous allons pouvoir créer un outil permettant une orientation approximative vers l'une ou l'autre solution de développement. Pour ce faire nous allons remettre six de ces principaux critères sur une échelle de type « Likert ». Ceci nous permettra de mieux évaluer les besoins et préférences d'un développeur ou d'une entreprise lors de la création d'une application. De plus, nous avons la chance d'avoir des solutions distinctes qui permettent de se situer aisément sur cette échelle à la fin du questionnaire. L'échelle est constituée de cinq symboles distincts allant de l'étoile jaune au cœur rouge. Ces symboles ont été choisis de manière totalement aléatoire puisqu'à l'encontre d'une échelle « Likert » classique, il n'y a pas de réponses positives ou négatives (MALHOTRA & al. 2010). Nous aurions pu faire une échelle de 1 à 5, mais cela aurait faussé les résultats du répondant. Par ailleurs, le sens de l'échelle peut changer en fonction du critère. Il était donc plus intelligent d'attribuer des symboles ne représentant pas directement un montant.

Afin de garder une pondération optimale, nous avons regroupé les critères similaires. Notre premier critère, l'investissement financier, représente les coûts pouvant être engendrés par la conception de l'application, les honoraires du développeur ainsi que la difficulté de maintenance. Le deuxième critère, la couverture, permet de savoir si l'on désire développer notre application sur une ou plusieurs plateformes mobiles à la fois. C'est-à-dire que si le développeur ne veut pas écrire plus qu'une seule fois son code pour créer son application, il devra viser une couverture maximale afin d'atteindre différentes plateformes mobiles.

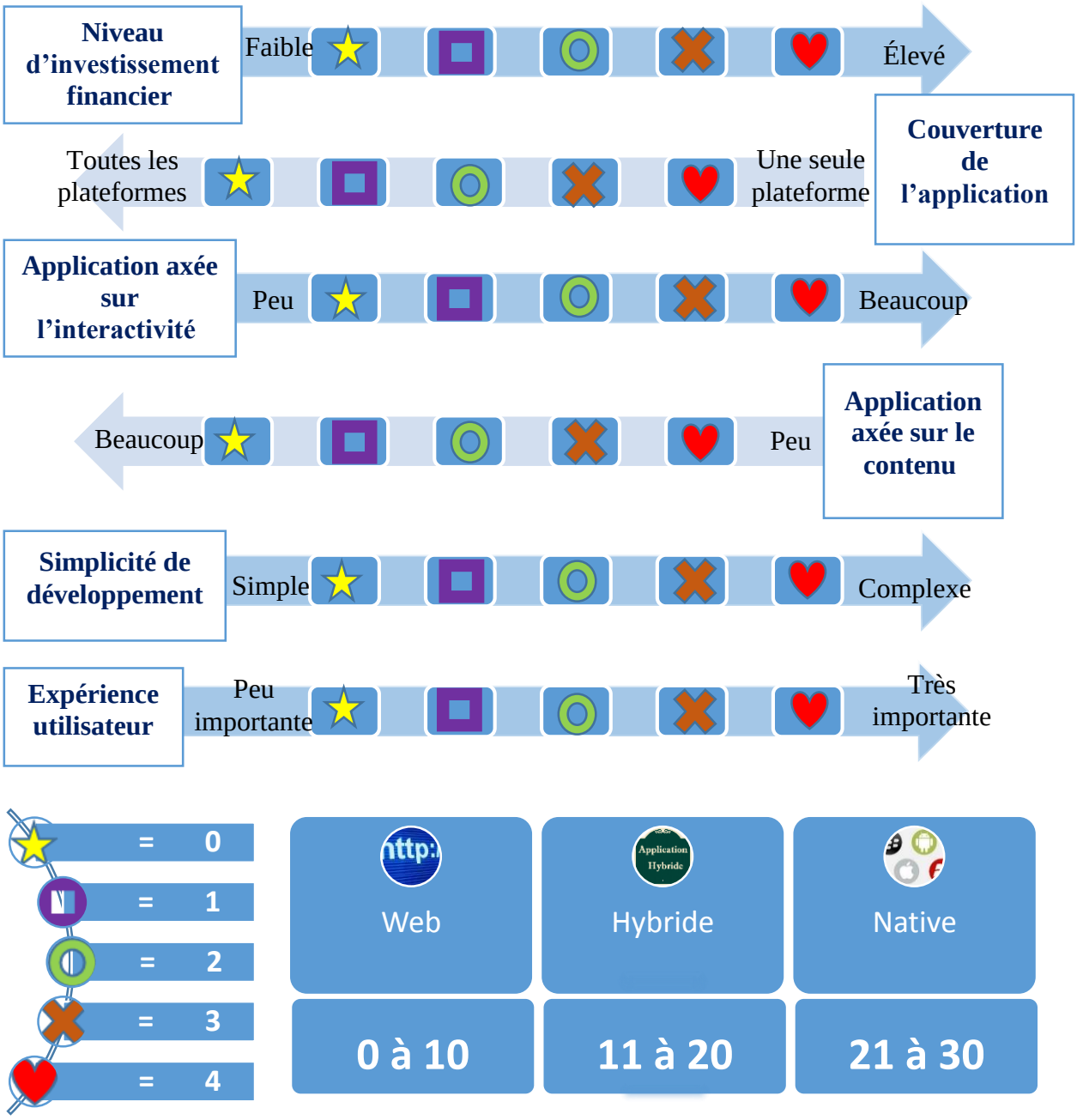
Les troisième et quatrième critères permettent de savoir la quantité de contenu ou d'interactivité désirée dans l'application.

À savoir, une application multimédia ou un jeu sera plus interactif qu'une application fournissant de l'information qui sera, elle, davantage axée sur le contenu. Nous avons ensuite la simplicité de développement qui regroupe la facilité de trouver un développeur ainsi que l'aisance à concevoir et mettre à jour l'application. Finalement, nous avons l'expérience utilisateur comme dernier critère. Celui-ci comprend la facilité d'utilisation, la qualité de l'interface utilisateur et de son design.

Après avoir répondu à chaque critère, un nombre précis de différents symboles en ressort. Il suffit ensuite de les convertir en chiffre à l'aide du tableau de conversion afin d'obtenir le résultat. Celui-ci se situera entre 0 et 30. S'il n'excède pas un total de 10, nous pouvons nous orienter vers une solution Web. C'est-à-dire que la personne cherchant à développer l'application désire davantage une application compatible, facile à développer, pas trop axée sur l'interactivité et les performances. Si le total des points se situe entre 11 et 20, il sera plus intéressant de développer son application à l'aide d'un outil hybride, car les besoins en performances et en qualité d'expérience utilisateur sont de mises. Finalement, si le total excède 20 points, nous conseillerons de développer son application à l'aide des outils natifs. Dans ce cas-ci, le développeur recherche un maximum de performances, une application axée sur l'interactivité ainsi qu'une expérience utilisateur irréprochable, mais en étant forcé à se rabattre sur une solution coûteuse et/ou à n'avoir qu'une compatibilité limitée.

Cet outil permet de s'orienter globalement vers un type de solution de développement. C'est-à-dire qu'une fois le résultat obtenu, nous sommes capables de déterminer la solution disposant du meilleur compromis par rapport à nos besoins. Il est cependant possible de se retrouver dans un intervalle proche de deux solutions différentes. Dans ce cas, il est recommandé de repasser les critères en vue tout en insistant bien sur les besoins indispensables.

Outil 1 - Orientation au choix d'une solution de développement générale



7.2 Outil permettant la décision d'une solution de développement hybride

Après avoir conçu un outil permettant de se diriger vers l'une ou l'autre solution de développement mobile, il serait intéressant d'élargir cette orientation aux différents outils hybrides étudiés précédemment. Dans ce cas-ci, nous sommes dans l'impossibilité de répertorier ces outils sur différentes échelles comme pour l'outil 1, car ceux-ci divergent énormément selon le critère.

Nous avons donc pensé à un outil permettant d'orienter le choix de l'utilisateur en fonction de ses critères absolus. Tout comme pour l'outil 1, nous avons regroupé les différentes variables présentes dans notre tableau 3 afin d'avoir un résultat moins indécis. Les critères ressortant de cette association de variables sont listés dans notre tableau 4 et sont respectivement la compatibilité, les performances et la facilité de développement.

Tableau 4 – Comparaison des différents critères des solutions hybrides

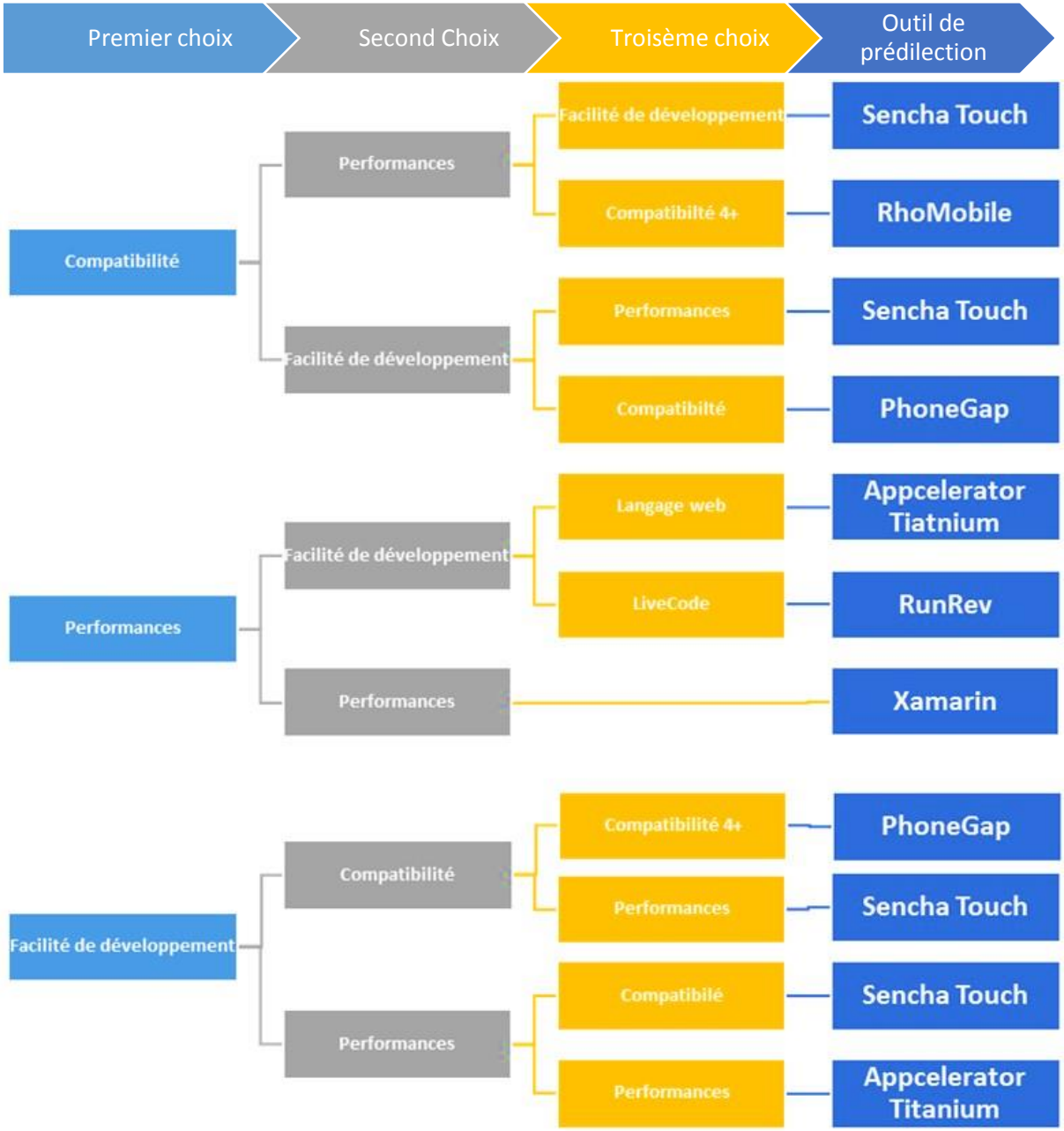
Outil / Caractéristiques	RhoMobile A	PhoneGap B	Appcelerator T. C	RunRev D	Sencha E	Xamarin F
Compatibilité	**5**	**5**	**3**	**3**	**4**	**3**
Performances (performances + interface utilisateur)	3+4 = **7**	2+2 = **4**	4+4 = **8**	4+4 = **8**	2+3 = **5**	4+5 = **9**
Facilité de développement (Popularité du langage + communauté + temps d'apprentissage + % du code partagé)	2+5+3+3 = **13**	5+5+5+4 = **19**	5+4+4+5 = **18**	4+5+3+4 = **16**	5+5+5+4 = **19**	5+3+4+3 = **15**

Après avoir additionné le résultat de ces différentes variables par rapport à chaque outil hybride, nous sommes en mesure de constater que nos trois critères permettent de formuler un arbre à décision. Nous allons par conséquent constituer ce dernier avec, comme racine, les trois critères repris dans le tableau 4.

Ces racines seront le critère représentant le premier choix de l'utilisateur. C'est-à-dire le critère auquel il désire accorder le plus d'importance. À partir de son premier choix, un second choix doit être fait pour déterminer son deuxième critère de prédilection. Le troisième et dernier choix permet à l'utilisateur de départager les solutions restantes en fonctions de leurs différences.

Après avoir orienté ses choix en fonctions de ses préférences, l'utilisateur sera guidé vers l'outil correspondant le mieux à ses besoins.

Outil 2 - Orientation au choix d'une solution de développement hybride



8 Conclusion

8.1 Résumé des contributions/Élément de réponse ?

Le paradigme « *Write once, run everywhere* » est un sujet actuellement très populaire dans le domaine de l'informatique. À la suite d'une lecture de nombreux ouvrages et articles disponibles sur ce thème, nous avons pu nous faire une idée sur l'existence et l'avancement de ce principe. La rapide montée en puissance des technologies mobiles est telle que l'univers des concepteurs de logiciels s'est fameusement élargi. Les moyens de développement mobile sont, par conséquent, en pleine expansion et il peut être difficile de s'y retrouver.

L'analyse de ces différentes solutions nous a fourni des informations précises sur les différents moyens de développement mobiles disponibles. En effet, les nombreux avantages et inconvénients attribués à chaque solution nous ont permis de guider tout individu désirant créer une application vers une solution adaptée à ses besoins.

Après avoir analysé ces solutions, il en ressort que les applications Web et hybrides sont les seules à proposer des réponses à notre illustre paradigme. La première, l'application Web mobile, respecte entièrement ces règles puisqu'après avoir écrit un code, celui-ci est interprétable sur tout appareil disposant d'un navigateur internet. Elle n'aura ainsi pas besoin de modifications à réaliser afin de fonctionner sur les appareils des systèmes d'exploitation mobiles visés. Cependant, cette solution ne permet pas encore des performances adéquates. La deuxième, l'application hybride, est plus contestable puisqu'il existe différents procédés ne permettant pas tous, le partage du code entre nos quatre systèmes d'exploitation choisit. Nous remarquons également qu'au plus l'outil de création s'oriente vers des performances relativement élevées, au plus celui-ci s'éloigne de notre paradigme.

De cette manière, un individu désirant créer une application se doit de faire des choix engendrant obligatoirement certaines concessions. Cette décision doit être focalisée sur trois critères essentiels qui sont la compatibilité, les performances et la facilité liée à la conception de l'application.

La réponse à notre question de recherche dépend aujourd'hui de ces trois critères. Il est en effet possible de développer certaines applications à l'aide d'un seul code pour les rendre compatibles sur les principaux appareils mobiles du marché. Nous ne pouvons cependant pas encore en faire une généralité.

La croissance et les améliorations constantes des langages et outils de développement multiplateformes nous obligent à croire au futur prometteur du paradigme « *Write once, run everwhere* ».

8.2 Apports et limites de la recherche

Notre contribution académique est intéressante puisqu'elle porte sur un thème récent et prometteur. Les applications mobiles sont plus présentes que jamais dans notre vie quotidienne et étudier leurs façons de développement semble novateur.

Notre recherche confirme que ce monde mobile est en pleine expansion. Une course à l'obtention d'un outil de développement multiplateforme est, plus que jamais, lancée. Nous avons pu constater que le nombre d'outils présents ne signifie pas qu'ils sont de qualité. Cette étude nous a permis de mieux comprendre l'avancement des technologies multiplateformes. Nous avons pu en déduire que ce type d'outil ne rivalise pas avec les solutions natives. Le paradigme « *Write once, run everywhere* » n'est possible que dans certains cas et dans certaines circonstances.

En outre, nos résultats nous permettent d'informer les créateurs d'applications sur l'une ou l'autre technologie en citant leurs avantages et restrictions. Cet apport est principalement séduisant pour de petits particuliers ou entreprises désirant se lancer dans la création d'applications. En effet, la recherche permet de comprendre les différents éléments pouvant intervenir lorsque l'on entreprend ce type de développement.

De plus, nous avons pu tirer deux outils de cet apprentissage. Ceux-ci permettront pleinement d'orienter une entreprise ou un particulier vers un type de solution de création d'applications mobiles.

Bien que cette étude ait permis de répondre à notre question de recherche, nous avons pu identifier certaines limites freinant la progression de notre aboutissement ainsi que certains aspects que nous aurions pu développer différemment.

Tout d'abord, la recherche requiert couramment des connaissances poussées en langages et outils de programmation. Celles-ci sont très limitées pour un étudiant visant l'obtention d'un master en science de gestion. Bien que cette limite puisse également représenter un atout par une vision plus simplificatrice et une perception plus objective, elle peut être contraignante.

Effectivement, les différentes solutions hybrides auraient mérité d'être testées une à une, à l'aide d'un cas précis d'application, afin d'avoir un avis personnel et une comparaison de leurs vertus.

Deuxièmement, les changements constants dans le monde de la technologie mobile provoquent des disparités fréquentes. Bien que l'utilisation de sources très récentes fût de mise, ces changements rapides font que certaines informations deviennent rapidement dépassées.

Troisièmement, notre recherche est étendue au développement d'applications mobiles compatibles sur les quatre plateformes mobiles les plus répandues. Or, le développement multiplateforme est un sujet qui peut s'étendre à d'autres plateformes telles que les ordinateurs classiques, Smart-Tv ou consoles de jeux.

Finalement, la méthode inductive a souvent primé dans cette recherche. C'est-à-dire que nous avons principalement comparé les différentes solutions en généralisant leurs caractéristiques pour en obtenir des réponses pouvant être mesurées. Cette généralisation était nécessaire afin d'obtenir des résultats sur une échelle identique, mais pourrait malheureusement manquer de précision.

8.3 Recherches futures

Ces limites nous ouvrent les portes vers des recherches futures. Il serait intéressant de réaliser une étude de cas reprenant les différents outils de développement un par un, pour réaliser une quelconque application. Le chercheur pourrait alors, soit confirmer, soit réfuter certaines théories émises, afin de donner un avis plus subjectif par rapport à ces solutions de développement mobile.

Une autre piste de recherche pourrait être l'acheminement des étapes de développement sur chacun des outils. Cette étude permettrait de décrire minutieusement la facilité de développement de ceux-ci. Par exemple, examiner le nombre de lignes de codes à modifier pour permettre d'être compatible sur les quatre plateformes les plus populaires. Différentes analyses sur la facilité de développement permettraient de mieux orienter le futur créateur d'application.

Finalement, une étude axée sur le développement d'applications sur différentes plateformes informatiques serait très séduisante. En effet, le développement commun d'applications sur smartphones, pc, consoles de jeux ou TV intelligentes commence à faire leur apparition.

9 Bibliographie

ABRAHAM E. (2010) Ext JS + JQtouch + Raphaël = Sencha. http://www.sencha.com/blog/ext-js-jqtouch-raphael-sencha (consulté le 05/07/2013)

AL-SUBAIHIN A, AL-KHALIFA H. (2012) Introducing Mobile Widgets Development in an advanced web technologies course. *SIGITE 2012.* pp61-64.

ANNE SALZ P, MORANZ, (2013) *The everything guide to mobile Apps : A practical guide to affordable mobile app development for your business. USA.* Adams Media Corporations.
ANTILA V, LUI A. (2011) Challenges in disigning inter-usable Systems. *INTERACT 2011 part 1*, septembre. pp396-403.

ANYPRESENCE (2013) The state of enterprise mobile readiness 2013. pp1-16

APPCELERATOR (n.d) http://www.appcelerator.com/products/ (consulté le 03/03/2013)

APPLE (n.d) https://developer.apple.com/programs/which-program/ (consulté le 22/03/2013)

APTARA EDITORIAL (2013) Hybrid native mobile apps « the future of HTML5 », expert claims. http://www.aptaracorp.com/digital-content-news/mobile-devices/hybrid-native-mobile-apps-the-future-of-html5-expert-claims/ consulté le (23/04/2013)

AVINS J, NGUYEN J. (2012) The making of fastbook : an HTML5 love story. http://www.sencha.com/blog/the-making-of-fastbook-an-html5-love-story/ (consulté le 04/03/2013)

BACCO A. (2013) Qu'est-ce qu'un framework ? http://www.siteduzero.com/informatique/tutoriels/developpez-votre-site-web-avec-le-framework-symfony2/qu-est-ce-qu-un-framework-1 (consulté le 10/02/2013)

BALKAN A. (2012) Mobile Considerations in UX Design : Web or Native. *Smashing Book #3 – Redesign the web.* pp256-284.

BALKAN A. (2013) Sites vs. Apps defined : the Documents-to-Applications Continuum. http://aralbalkan.com/notes/the-documents-to-applications-continuum/ (consulté le 15/05/2013)

BASTIDE A. (2012) Site Web mobile ou application native ? http://www.indexel.net/infrastructure/choisir-entre-site-web-mobile-et-application-native-3563.html (consulté le 09/05/2013)

BELFIORE G. (2012) Développement mobile : Web ou natif ? Retours sur les enjeux. http://pro.clubic.com/creation-de-site-web/langage-programmation/actualite-497418-developpement-mobile-web-natif-retours-enjeux.html (consulté le 09/05/2013)

BEAVIS G. (2008) A complete history of Android. http://www.techradar.com/news/phone-and-communications/mobile-phones/a-complete-history-of-android-470327 (consulté le 02/07/2013)

BOSOMWORTH D. (2013) Mobile Marketing Statistics 2013. http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/, (consulté le 21/05/2013)

BRIDGWATER (2013) The future for HTML5 is hybrid native mobile apps, http://www.computerweekly.com/blogs/cwdn/2013/03/the-future-for-html5-is-hybrid-native-mobile-apps.html (consulté le 27/05/2013)

BRIGHT P. (2012) Windows Phone 8 review : Microsoft lays foundation for success. http://arstechnica.com/gadgets/2012/11/windows-phone-8-review-microsoft-lays-a-foundation-for-success/ (consulté le 22/04/2013)

BRIGHT P. (2013) Xamarin 2.0 reviewed : iOS development comes to Visual Studio. http://arstechnica.com/information-technology/2013/02/xamarin-2-0-reviewed-ios-development-finally-comes-to-visual-studio/ (consulté le 05/07/2013)
CANALYS (2013) 11 % quaterly growth in downloads for leading App Store. http://www.canalys.com/newsroom/11-quarterly-growth-downloads-leading-app-stores (consulté le 03/07/2013)

CGV-EXPERT (2012) Les conditions de vente de l'Appstore pour les éditeurs d'application. http://www.cgv-expert.fr/article/conditions-vente-appstore-editeurs-application_46.htm (consulté le 03/02/2013)

CHA S, YUN Y. (2013) Smartphone Application Development using HTML5based Cross-Platform Freamework. *AST 2013*. pp151-153.

CHENG B. (2012) Virtual Browser for Enabling Multi-device Web Applications. *Midlleware 2012*. Article No 3

CIMITILE, RISI, TORTORA (2011) Automatic Generation of multi platform Web Map Mobile Applications. *DMS 2011*.

CORRAL, GARIBBO, RAMELLA, SILLITI, SUCCI (2011) Evolution of Mobile Software Development from Platform-specific to Web-based Multiplatform Paradigm. *Onward ! 2011*. pp181-183.

CSOMOR, HOCK, SMART (2005) *Cross-Platform GUI Programming with wxWidgets*. Pearson education.

DAHAN O. (2012) Stratégie de développement cross-plateform. http://www.e-naxos.com/Blog/post/Strategie-de-developpement-Cross-Platform-Partie-1.aspx (consulté le 05/05/2013)

DAHAN O. (2012) Stratégie de développement Cross-Plateforme 2012-2013. http://www.e-naxos.com/download/Stratgie-de-dveloppement-Cross-Platform-_1411C/Stratgie-de-dveloppement-Cross-Plateforme.pdf (consulté le 02/02/2013)

DEGANI, HARTMANN, STEAD (2011) Cross-platform mobile development. *Medical development projet : D4*. pp2-18.

DELLER M, EBERT A (2011) ModControl – Mobile Phones as a versatile Interaction Device for Large Screen Applications. *INTERACT 2011 part 2*, septembre. pp289-296.

DEWAN P, OMOJOKUN O. (2007) Automatic Generation of Device User-Interfaces ? *PERCOM 07*. pp251-261.

DEW-JONES S. (2012) HTML5 the future for web-based trading apps. http://www.waterstechnology.com/sell-side-technology/analysis/2222844/html5-the-future-for-webbased-trading-apps (téléchargé le 02/03/2013)

DOKOUPIL T. (2009) Strinjing It Rich : Is there an app for that ? http://www.thedailybeast.com/newsweek/2009/10/05/striking-it-rich-is-there-an-app-for-that.html (consulté le 03/02/2013)
DUFFY T. (2012) Programming with Mobile Applications : Android, iOS and Windows Phone 7. *1st Edition, Cengage Learning.*

DUJARDIN R, LORENCE F. (2013) Application native, web ou hybride : 6 points à considérer pour faire le bon choix. http://www.journaldunet.com/developpeur/expert/53610/application-mobile-native--web-ou-hybride---6-points-a-considerer-pour-faire-le-bon-choix.shtml (consulté le 04/04/2013)

ELEMENTARYOS (2012) *HUMAN INTERFACE GUIDELINES.* http://elementaryos.org/docs/human-interface-guidelines (consulté le 09/04/2013)

ERTL FALB KAINDL POPP RANEBURGER (2011) Automated generation of device-specific WIMP UIs : Weaving of structural and Behavioral Models. *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computings systems* pp41-46.

ESCALLIER P. (2011) Quel navigateur sous Android ? http://www.tomshardware.fr/articles/Android-Web-navigateur,2-806-2.html (consulté le 24/04/2013)

FAIRHEAD H. (2011) JavaScript is Slow. http://www.i-programmer.info/news/86-browsers/3492-javascript-is-slow.html (consulté le 09/03/2013)

FERRELL J. (2013) Dos and don’ts of cross-platform mobile design. http://www.netmagazine.com/features/dos-and-donts-cross-platform-mobile-design (consulté le 29/05/2013)

FIERGERMAN S. (2012) Apple has approved one million apps for the App Store. http://mashable.com/2012/11/19/apple-app-store-1-million-submissions/ (consulté le 08/03/2013)

FINGAS J. (2013) RIM a brief history from Budgie to BlackBerry 10. http://www.engadget.com/2013/01/28/rim-a-brief-history-from-budgie-to-blackberry-10/ (consulté le 23/04/2013)

FRIEDMAN N. (2013) 9 surprising reasons mobile apps get rejected from Apple App Store. http://venturebeat.com/2013/02/08/9-surprising-reasons-mobile-apps-get-rejected-from-the-apple-app-store/ (consulté le 03/06/2013)

GAHRAN A. (2012) A future with fewer mobile apps ? http://edition.cnn.com/2012/07/27/tech/mobile/mobile-apps-decline (consulté le 09/04/2013)

GARTNER (2013) Gartner says Asia/Pacifi led worldwide mobile phone sales growth in first quarter of 2013. http://www.gartner.com/newsroom/id/2482816 (consulté le 20/07/2013)

GOOGLE PLAY (2012) Transaction fees. https://support.google.com/googleplay/android-developer/answer/112622?hl=en (consulté le 21/04/2013)

GRIGSBY J. (2009) Native vs Web vs Hybrid Mobile development choice. *Web vision 2009*

GRISEL L, OSSET P. (2004) *L'analyse du cycle de vie d'un produit ou d'un service : Applications et mise en pratique. AFNOR.* pp31-75

GUBE J. (2010) What is user experience design ? Overview, Tools and resources. http://uxdesign.smashingmagazine.com/2010/10/05/what-is-user-experience-design-overview-tools-and-resources/ (consulté le 14/03/2013)

GUNELIUS S (2011) What is an API and why does it matter ? http://sproutsocial.com/insights/2011/10/api-definition/ (consulté le 09/05/2013)

HAROON R. (2011) An introduction to modern mobile operating system. http://www.addictivetips.com/mobile/an-introduction-to-modern-mobile-operating-systems/ (consulté le 29/03/2013)

HINAULT R. (2012) Le SDK BlackBerry 10 disponible en version finale : c/c++, Java, Web, Adobe AIR, RIM veut séduire un maximum de développeurs. http://www.developpez.com/actu/50238/Le-SDK-BlackBerry-10-disponible-en-version-finale-C-C-Java-Web-Adobe-AIR-RIM-veut-seduire-un-maximum-de-developpeurs/ (consulté le 12/06/2013)

IBM CORPORATION (2012) A mobile application development primer – A guide for enterprise teams working on mibile application projects, systems and software development. pp1-12.

IDC (2013) More Smartphones were shipped in Q1 2013 than feature phones, an industry first according to IDC. http://www.idc.com/getdoc.jsp?containerId=prUS24085413 (consulté le 02/05/2013)

JIA, WANG, XU, ZHANG (2011) Research and Implementation of Cross-platform Development of Mobile Widget. *Communication Software and Networks (ICCSN) 2011 IEEE 3rd International conference*. pp146-150.

JONES A. (2012) Native, Hybrid or Web Apps ? Janvier 2012, http://www.sitepoint.com/native-hybrid-or-web-apps/ (consulté le 15/04/2013)

KAVALDJIAN POPP RANEBURGER (2011) Optimized GUI Generation for small screens. *Model-Driven Development of advanced user interfaces*. pp107-122.
KINGSJEY-HUGHES A, KINGSJEY-HUGHES K. (2005) Beginning Programming. *John Wiley & Sons*

KOETSIER J. (2012) Mobile app development : 94% of software developers bet ont HTML5 winning. http://venturebeat.com/2012/11/07/mobile-app-development-94-of-software-developers-betting-on-html5-winning/ (consulté le 17/02/2013)

KOETSIER J. (2013) 5,000 developers say HTML5 is real, it's now, and yeah, it's also the future. http://venturebeat.com/2013/02/26/5000-developers-say-html5-is-real-its-now-and-yeah-its-also-the-future/ (consulté le 25/05/2013)

KRAMER H (2012) Windows Phone 8 APP deployment. http://blogs.tieto.com/enterprisemobility/2012/08/28/windows-phone-8-app-deployment/ (consulté le 06/06/2013)

KRILL P. (2012) Native mobile app dev vs. HTML5 :Why not both ? Info World tech watch (novembre 2012) http://www.infoworld.com/t/mobile-development/native-mobile-app-dev-vs-html5-why-not-both-206167 (consulté le 23/04/2013)

KUHN P. (2012) Windows 8 : Introduction to the platform. http://www.silverlightshow.net/items/Windows-Phone-8-Introduction-to-the-Platform.aspx (consulté le 29/06/2013)

KUMAR S. (2013) History of JavaScript. http://www.javascriptstyle.com/history-of-javascript (consulté le 09/05/2013)

KYRNIN J. (n.d.) What is CSS3 ? An introduction to the modularization of Cascading Style Sheets (level 3) http://webdesign.about.com/od/css3/a/aa061206.htm (consulté le 21/05/2013)

LAWTON K. (2010) Cross-Device Continuity : The Future of User Interface. http://seekingalpha.com/article/208913-cross-device-continuity-the-future-of-user-interface (consulté le 08/05/2013)

LIN J. (2005) Early-stage design and prototyping of Cross-Device User Interfaces. *WORKSHOP AT CHI 2003*. Fort Lauderdale

LIVECODE (n.d.) http://livecode.com/how-it-works/features/ (consulté le 17/04/2013)

LUTES K. (2012) Cross-Platform Mobile App Software Development in the Curriculum. *Issues in Informing Science and Information Technology Volume 9*. pp115-124.

MALHOTRA N, DECAUDIN J-M, BOUGUERRA A, BORIES D. (2010) *Etude marketing*. Ed : France, Pearson Education : pp239-274

MESKENS, LUYTEN, CONINX (2010*)* D-Macs : Building Multi-Device User Interfaces by Demonstrating Sharing and Replaying Design Actions. *UIST 2010 Proceedings of the 23nd annual ACM symposium on User interface software and technology*. pp129-138.

MICROSOFT .NET FRAMEWORK 4 (n.d.) Overview of Web application Security threats. http://msdn.microsoft.com/en-us/library/f13d73y6(v=vs.100).aspx (consulté le 22/07/2013)

MICROSOFT MSDN LIBRARY (2012) Developing mobile Web Apps, patterns & practices. pp6-109

MIRAVET, MARIN, ORTIN, RIONDA (2009) A framework for automatic generation of mobile applications for mutliple platforms. *MOBILITY 2009 Proceedings of the 6th International Conference on Mobile Technology, Application & Systems, Article No. 23*. Septembre

MOBILESTATISTICS (2012) Quaterly device sales in 2012. http://www.mobilestatistics.com/mobile-statistics/ (consulté le 03/04/2013)

MOTOROLASOLUTIONS (n.d.) http://www.motorolasolutions.com/US-EN/Business+Product+and+Services/Software+and+Applications/RhoMobile+Suite (consulté le 17/04/2013)

NGUYEN L. (2013) BlackBerry déclare que 120 000 apps sont disponibles dans Blackberry World. http://fr.ubergizmo.com/2013/05/blackberry-120000-apps-disponibles-blackberry-world/ (consulté le 09/05/2013)

OINAS-KUKKONNEN, SEGERSTAHL, VAANANEN-VAINIO-MATTILA, WALJAS (2011) Cross-platform service user experience : a field study and an initial framework, *MobileHCI 2010*. Septembre. pp219-228.

OLSON J. (2012) How to succed with your mobile app. http://mobile.smashingmagazine.com/2012/11/07/succeed-with-your-app/ (consulté le 18/03/2013)

PALMIERI M, SINGH I. (2012) Comparison of Cross-Platform mobile development tools. *Conférence on Intelligence in Next Generation Networks (ICIN 2012)*. pp179-186.

PARKER J. (2013) Pros and cons of cross-platform mobile app developments tools. http://www.techrepublic.com/blog/android-app-builder/pros-and-cons-of-cross-platform-mobile-app-development-tools/ (consulté le 01/06/2013)

PHANOURIOU C. (2000) UIML : A Device-Independent User Interface Markup Language. pp1-16

PHONEGAP (n.d) http://phonegap.com/about/faq/ (consulté le 17/04/2013)

PIERRAT O. (2012) *Le développement multiplateforme : enjeux, promesses et réalité.* http://www.journaldunet.com/developpeur/expert/51048/le-developpement-multi-plateforme---enjeux--promesses-et-realite.shtml (consulté le 04/03/2013)

RAJA R. (2011) An introduction to mobile operating systems. http://www.addictivetips.com/mobile/an-introduction-to-modern-mobile-operating-systems/ (consulté le 09/04/2013)

ROTHMAN W. (2013) App showdown : Android vs. iPhone. http://www.nbcnews.com/id/38382217/ns/technology_and_science-wireless/t/app-showdown-android-vs-iphone/#.Uf-GV5Io6K8 (consulté le 09/04/2013)

RUBINO B. (2013) With 160,000+ apps, Microsoft breaks down the numbers for the Windows Phone Store. http://www.wpcentral.com/160000-apps-microsoft-windows-phone-store-numbers (consulté le 06/07/2013)

RUNGTA A. (2012) Cross Platform Mobile App Development. http://www.indusnet.co.in/blog/cross-platform-mobile-app-development/547/ (consulté le 18/06/2013)
RUSCHER S (2012) WINDOWS Phone 8 : le test. http://www.clubic.com/os-mobile/windows-phone-8/article-523453-1-windows-phone-8.html (consulté le 09/04/2013)

SAVITZ E. (2013) *HTML5 Vs.* Native Mobile Apps : Myths and Misconceptions. http://www.forbes.com/sites/ciocentral/2013/01/23/html5-vs-native-mobile-apps-myths-and-misconceptions/ (consulté le 23/04/2013)

SELVADURAI S. (2013) HTML5 for mobile developers. http://mobiforge.com/developing/story/html5-mobile-developers (consulté le 02/06/2013)

SENCHA (n.d.) http://www.sencha.com/products/touch (consulté le 17/04/2013)

SILVERMARN L. (2013) Property cross Android : Implementation Shoutout. http://blog.trackabout.com/2013/03/05/propertycross-android-implementation-shootout/#xamarin (consulté le 03/04/2013)

STANGARONE J. (2013) Native apps : The wrong choice for business ? http://www.mrc-productivity.com/Research/mobile-guide.html (consulté le 02/02/2012)

STARDUST (2012) Découvrez les résultats de la première enquête en ligne menée sur la qualité des applications mobiles. http://www.afjv.com/news/1936_enquete-qualite-applications-mobiles.htm (consulté le 23/04/2013)

TAFT D. (2012) HTML5 vs Native : What's a mobile developer do ? http://www.eweek.com/c/a/Application-Development/HTML5-vs-Native-Whats-a-Mobile-Developer-to-Do-648997/ (consulté le 12/03/2013)
THOMPSON T. (2013) Review :Xamarin 2.0 works mobile development magic – Impressive Xamarin SDK brings native iPhone and Android development to C# programmers, Visual Studio. http://www.infoworld.com/d/application-development/review-xamarin-20-works-mobile-development-magic-217236 (consulté le 04/06/2013)

VERRECHI J. (2011) Introduction à l'HTML5. http://www.html5-css3.fr/html5/introduction-html5 (consulté le 09/04/2013)

VISION MOBILE (2012) Cross-platform developper tools 2012 : bridging the worlds of mobile app and the web. Février. pp1-97

VISWANATHAN P. (2013) RunRev Introduces the LiveCode 6.0 Community Edition. http://mobiledevices.about.com/od/thirdpartyservices/a/Runrev-Introduces-The-Livecode-6-0-Community-Edition.htm (consulté le 05/07/2013)

W3C WORKING GROUP NOTE. (2004) Authoring techniques for device independence. http://www.w3.org/TR/di-atdi/ (consulté le 09/04/2013)

WARGO J. (2012) *PhoneGap essentials, building cross-platform Mobile Apps, Adisson-Wesley*, édition 1. pp3-22.

WARREN C. (2012) The pros and cons of cross-platform App Design. http://mashable.com/2012/02/16/cross-platform-app-design-pros-cons/ (consulté le 09/04/2013)

WHINNERY K. (2012) Comparing Titanium and PhoneGap. http://developer.appcelerator.com/blog/2012/05/comparing-titanium-and-phonegap.html (consulté le 02/02/2013)

WOODS B. (2013) The road to BlackBerry 10 : The evolution of RIM's OS and BES. http://www.zdnet.com/the-road-to-blackberry-10-the-evolution-of-rims-os-and-bes_p2-7000009899/ (consulté le 09/06/2013)

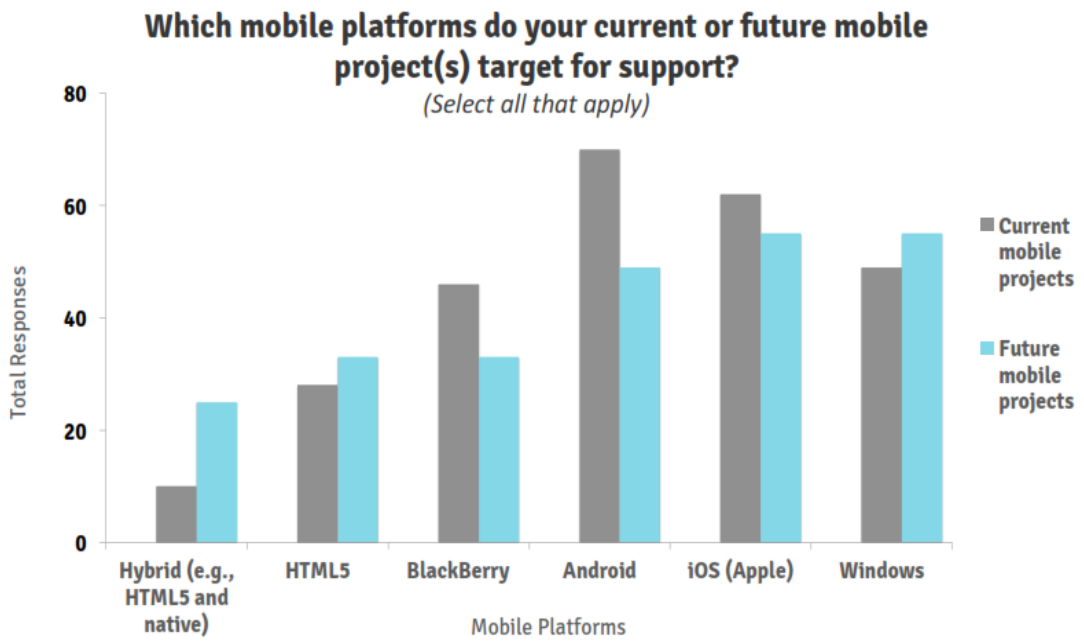
XAMARIN (n.d) http://xamarin.com/ (consulté le 17/04/2013)

YANG S. (2013) Google Play revenue surges, but still far short of Apple : data. http://www.reuters.com/article/2013/04/22/net-us-mobile-revenues-apps-idUSBRE93L05U20130422 (consulté le 04/06/2013)

YAROW J. (2013) This is the only number that matters for Apple in its big App Store press release. http://www.businessinsider.com/active-app-store-users-2013-1 (consulté le 29/05/2013)

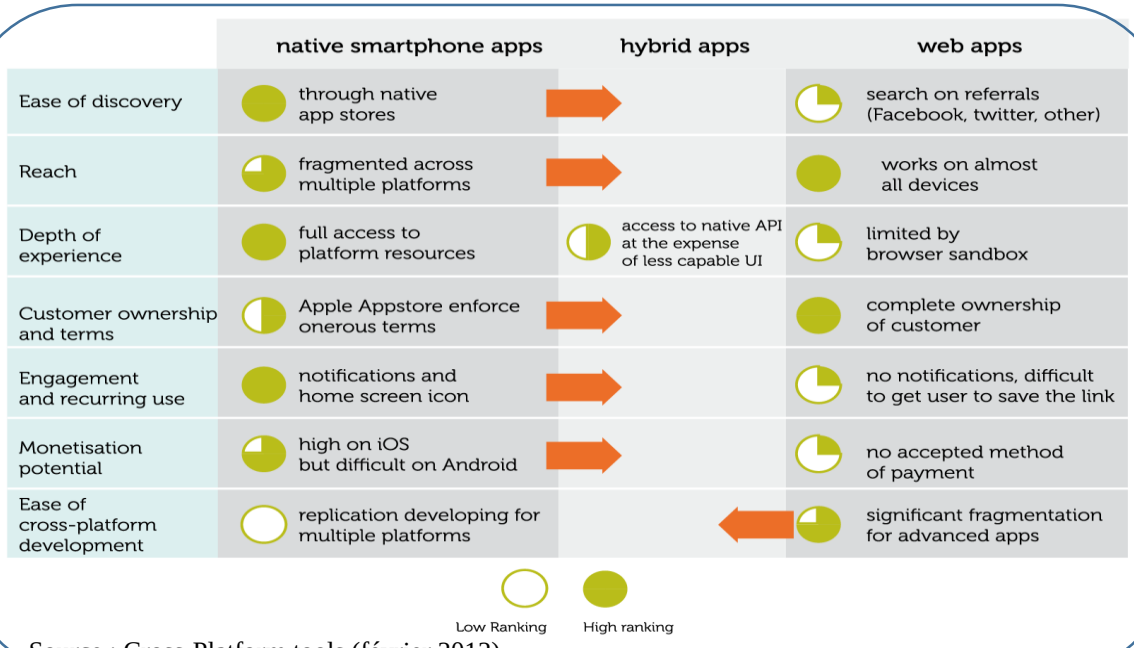
10 Annexes

10.1 Enquête sur les plateformes visées lors d'un projet en cours ou futur d'applications mobiles



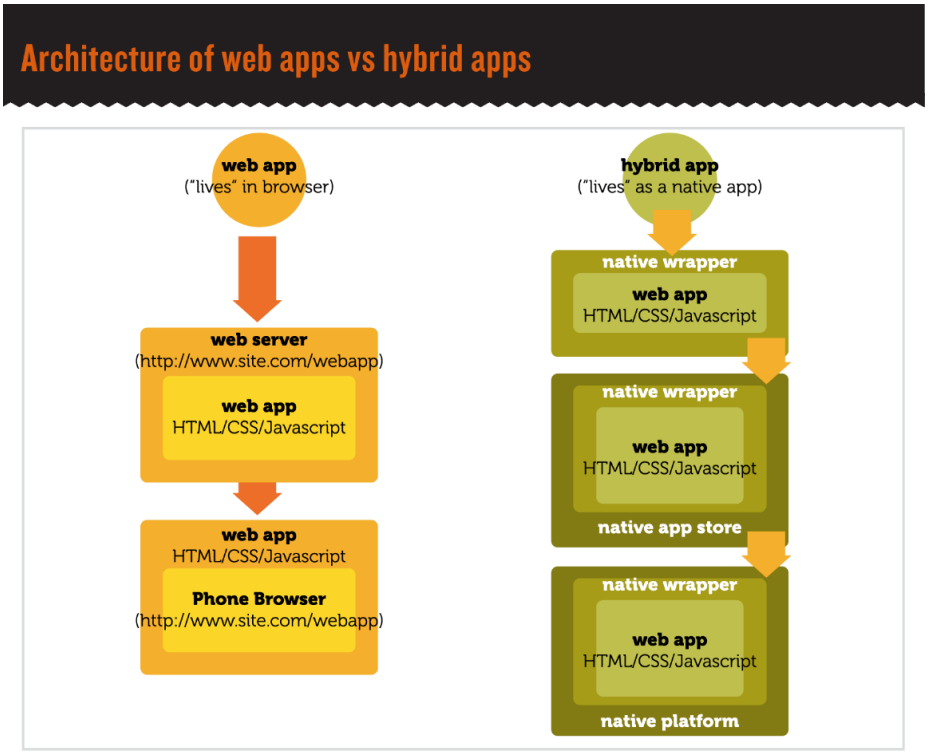
Source : ANYPRESENCE (2013) The state of enterprise mobile readiness 2013

10.2 Tableau de comparaison des applications native, hybride et web relatant le fait que l'application hybride soit un compromis



Source : Cross-Platform tools (février 2012)

10.3 Comparaison de l'architecture d'une application web mobile et hybride



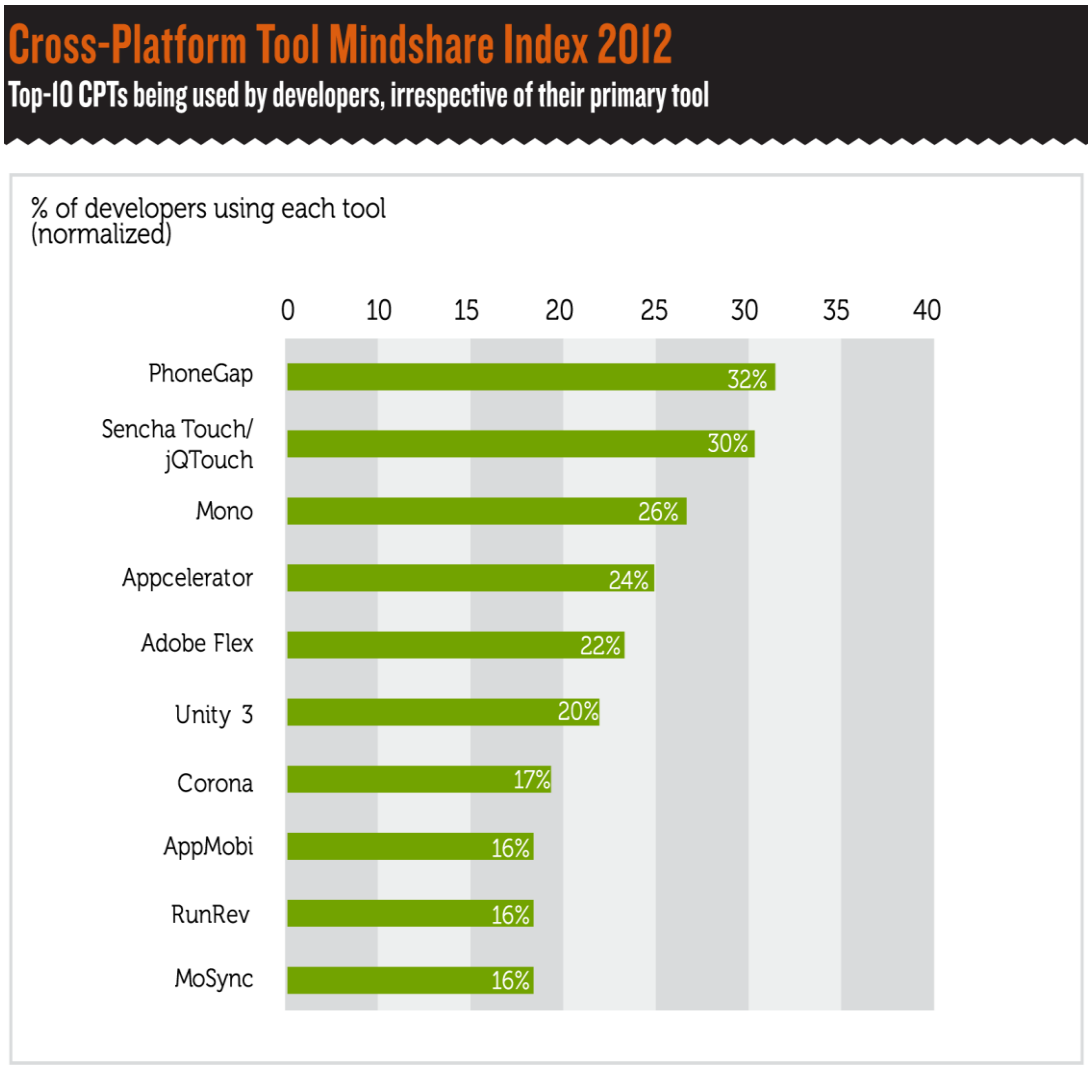
Source : Cross-Platform tools (février 2012)

10.4 Tableau de comparaison des différents types d'outils selon divers critères

Solution / Caractéristiques	Natives	Web	Hybrides
Compatibilité	Uniquement avec le système d'exploitation visé	Compatible avec tous les appareils disposant d'un navigateur internet sans modification	Compatible avec de nombreux appareils selon l'outil utilisé (requiert un minimum de modification pour chaque appareil)
Accès aux Fonctionnalités	Toutes	Réduite	Majorité
MAJ bibliothèque logicielles	Directe	Lente	Lente
Interface utilisateur	propre aux systèmes d'exploitation	Interface web	Interface web, copie de l'interface utilisateur native, voire propre aux systèmes d'exploitation selon l'outil de développement
Temps de développement	Long	Court	Moyen
MAJ et maintenance	Longue et complexe	Rapide et facile	Plus ou moins rapide/facile
Performances	Elevés	Réduites	Médiocre à élevé selon l'outil utilisé
Graphisme	Fort	Bas	Moyen
Disponibilité/coût des développeurs	Faible/Elevé	Forte/faible	Forte/faible
Intégration	Complète (notification & icone de bureau)	Aucune	Complète (notification et icone de bureau)
Sécurité	Elevée	Faible	Elevée
Utilisation hors connexion	Oui	Non	Partielle
Distribution	-Magasin d'applications (Appstore, GooglePlay,…)	-Serveur web	-Magasin d'applications (Appstore, GoogePlay,…)
Monétisation	Elevé	Limité	Elevé
Langage	-Spécifique à la plateforme	-Web (HTML, JS, CSS,…)	Web (HTML, JS, CSS,…)
Environnement de développement intégré	-Spécifique à la plateforme	-Vaste choix	-Vaste choix

Sources : MICROSOFT MSDN LIBRARY (2012) Developing mobile Web Apps, patterns & practices. pp6-109
GRIGSBY J. (2009) Native vs Web vs Hybrid Mobile development choice. *Web vision 2009*
VISION MOBILE (2012) Cross-platform developper tools 2012 : bridging the worlds of mobile app and the web. Février. pp1-97
STANGARONE J. (2013) Native apps : The wrong choice for business ?

10.5 Comparaison du pourcentage d'utilisation des différents outils hybrides en fonction du nombre de développeurs



Source : Cross-Platform tools (février 2012)

10.6 Compatibilité et support de RhoMobile quant aux différentes fonctionnalités et plateformes

API Name	Languages	iOS	Android	Windows Embedded			Windows		BlackBerry	RIM	
Click this row for sorting		3.0+	2.3+	WM6.1+	WM6.5+	CE6.0+	XP/7	7.0+	4.6+		Ver
Accelerometer		★	★								2.2
Alert		★	★	★	★	★	★	★	★	★	2.0
AppApplication		★	★	★	★	★	★	★	★		2.0
AsyncHttp		★	★	★	★	★		★	★		2.0
AudioCapture		★	★	★	★	★		★			2.2
Barcode		★	★	★	★	★		★	★		2.0
BluetoothManager		★	★	★	★			★	★	★	2.0
BluetoothSession		★	★	★	★			★	★	★	2.0
Camera		★	★	★	★	★		★	★	★	2.0
Database		★	★	★	★	★		★	★	★	2.0
DateTimePicker		★	★	★	★	★	★	★	★	★	2.0
File		★	★	★	★	★		★	★	★	2.0
GeoLocation		★	★	★	★	★		★	★	★	2.0
Magnetometer		★	★								2.2
MapView		★	★	★	★	★		★	★	★	2.0
MediaPlayer			★								2.2
Menu Apis		★	★	★	★	★	★	★	★	★	2.0
Menu-Toolbar-Tabbar		★	★	★	★	★	★	★	★	★	2.0
Model Controller		★	★							★	2.0
NativeTabbar		★	★				★			★	2.0
NativeToolbar		★	★	★	★	★	★				2.0
NavBar		★									2.0
NdefMessage			★								2.0
NdefRecord			★								2.0
NFCManager			★								2.0
NFCTags			★								2.0
RawSensors		★	★								2.2
RhoApplication		★	★	★	★	★	★	★	★	★	2.0
RhoConfig		★	★	★	★	★	★	★	★	★	2.0
RhoContact		★	★	★	★	★		★	★	★	2.0
RhoController		★	★	★	★	★	★	★	★	★	2.0
RhoError		★	★	★	★	★	★	★	★	★	2.0
RhoEvent		★	★	★	★	★		★	★	★	2.0
RhoLog		★	★	★	★	★	★	★	★	★	2.0
Rhom		★	★	★	★	★	★	★	★	★	2.0
Rhom Source		★	★	★	★	★	★	★	★	★	2.0
RhoUtils		★	★	★	★	★	★	★	★	★	2.0
RingtoneManager		★	★	★	★	★		★	★	★	2.0
SignatureCapture Inline		★	★	★	★	★		★	★	★	2.0
SyncEngine		★	★	★	★	★	★	★	★	★	2.0
System		★	★	★	★	★	★	★	★	★	2.0
Timer API		★	★	★	★	★		★	★	★	2.0
VideoCapture			★	★	★	★					2.2
WebView		★	★	★	★	★	★	★	★	★	2.0

Source : http://docs.rhomobile.com/rhoelements/apicompatibility#api-matrix-all-

10.7 Compatibilité et support de PhoneGap quant aux différentes fonctionnalités et plateformes

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 5.x	Blackberry OS 6.0+	WebOS	Windows Phone 7 + 8	Symbian	Bada
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓	✓
Compass	x	✓	✓	x	x	✓	✓	x	✓
Contacts	✓	✓	✓	✓	✓	x	✓	✓	✓
File	✓	✓	✓	✓	✓	x	✓	x	x
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	x	x	x	✓	x	x
Network	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	x	x

✓ - supported feature
X - unsupported feature due to hardware or software restrictions

Source : PHONEGAP (n.d) *http://phonegap.com/about/feature/*

10.8 Compatibilité et support d'Appcelerator Titanium quant aux différentes fonctionnalités et plateformes

Capability	iPhone	Windows M	Blackberry	Symbian	Android	Palm
Geo Location	Yes	NA	NA	NA	Yes	NA
PIM Contacts	Yes	NA	NA	NA	Partially	NA
Camera	Yes	NA	NA	NA	Yes	NA
Native menu/Tab bar	Yes	NA	NA	NA	Yes	NA
Barcode	No	NA	NA	NA	No	NA
Audio/Video Capture	Yes	NA	NA	NA	Yes	NA
Bluetooth	No	NA	NA	NA	No	NA
Push/SMS	Partially	NA	NA	NA	Partially	NA
Calendar	No	NA	NA	NA	Yes	NA
Screen Rotation	Yes	NA	NA	NA	Yes	NA
Native Maps	Yes	NA	NA	NA	Yes	NA
Ringtones	No	NA	NA	NA	No	NA
Storage	Yes	NA	NA	NA	Yes	NA

Source : RUNGTA A. (2012) Cross Platform Mobile App Development, http://www.indusnet.co.in/blog/cross-platform-mobile-app-development/547/

10.9 Tableau de comparaison des différents outils hybrides de notre échantillon

Outils/ Caractéristiques	RhoMobile	PhoneGap	Appcelerator T.	RunRev	Sencha	Xamarin
Compatibilité	-iOS -Android -Windows phone 7 -Blackberry 5/6 -Symbian -WebOS	-iOS -Android -Windows phone 7/8 -Blackberry 5/6 -Symbian -Palm -WebOS -Tizen	-iOS -Android	-iOS -Android	-iOS -Android -Windows Phone -Blackberry	-iOS -Android
Langage(s)	HTML5, JavaScript, CSS, Ruby	HTML5, JavaScript, CSS	HTML5, JavaScript, CSS	-LiveCode	HTML5, JavaScript, CSS	C#, .NET
Fonctionnement	Machine virtuel	WEB compilé	Génération de code natif	Natif	WEB compilé	Natif
Interface native	Non	Non	Oui	Oui	Oui	Oui
Performance	Médiocre	Médiocre	Elevé	Elevé	Moyenne	Elevé
MAJ des API	Rapide	Rapide	Lente	Rapide	Rapide	Normal
Support/communauté	Important/Grande	Important/Grande	Important/ Grande	Important/Grande	Important/Grande	Moyen/Grande
Licence	Payante	Gratuite	Gratuite (API de base)	Gratuite/Payante	Gratuite/Payante	Gratuite/Payante
Application de référence	Super Trainer HQ, Pilsner Urquell	NetFlix, Linkedin	NBC ipad, LEGOLAND, Zipcar	The forest guide, EuroTalk	XERO, Direct TV app	AOL, HP, Target, Monster

Source : VISIONMOBILE Cross-Platform tools (février 2012)
XAMARIN (n.d.)
PHONEGAP (n.d.)
APPCELERATOR (n.d)
RUNREV (n.d.)
SENCHA (n.d)
MOTOROLASOLUTIONS (n.d.)
PALMIERI M, SINGH I. (2012) Comparison of Cross-Platform mobile development tools. *Conférence on Intelligence in Next Generation Networks (ICIN 2012)*. pp179-186.
DEGANI, HARTMANN, STEAD (2011) Cross-platform mobile development. *Medical development projet : D4*. pp2-18.