Synthèse Article et étude des technologies PhoneGap & Titanium Mobile(Appcelerator)

Evaluating Cross Platform Devellopment and Approches for Mobile Application

Henning Heitk¨otter, Sebastian Hanschke, and Tim A. Majchrzak

Ce que nous définissons de nos jours à travers le terme de "smartphone" est un téléphone muni de capteurs avec une capacité de calculs beaucoup plus avancée. La liste des fonctionnalités offertes à travers les différents modèles de smartphones ne fait que croître.
Actuellement sur le marché du mobile, nous retrouvons différentes plateformes de développement, les plus populaires sont, Android, iPhone, Windows Phone et BlackBerry. Elles possèdent chacune son propre environnement de développement.

Pouvoir développer son application une seule fois et la porter sur différentes plateformes, est possible de nos jours grâce à plusieurs solutions cross-plateformes qui existent, avec des approches et des processus de développement différents. Face à ces différentes solutions, les développeurs sont amenés à choisir la solution qui leur est la plus adaptée.

Le but de notre d'étude dans un premier temps est d'apporter une meilleure compréhension des cross-plateformes, dans une seconde partie, fournir un modèle de critère aux développeurs, et enfin nous nous appuierons sur deux exemples de technologies cross-plateformes.

I. Approche générale des cross-plateformes

Nous rappelons qu'ils existent différentes plateformes mobile de nos jours, avec différentes versions d'API, différentes tailles et résolution d'écrans. Les développeurs souhaitent souvent avoir un public le plus large possible à travers leurs applications. Développer une application pour chaque plateforme nécessite la maîtrise d'une part les différents langages de programmation et d'autre part des différents environnements. Cela devient très long en termes de temps de développement.

Contrairement au web, le mobile n'est pas aussi flexible, il est très différent d'une plateforme à une autre, auxquels s'ajoute les contraintes du mobile en général, c'est à dire, la gestion des écrans, la batterie, les capacités limitées et le réseau. Comme on l'a dit précédemment, ce qui diffère d'un téléphone ordinaire à un smartphone, c'est l'ensemble des capteurs qui le constituent et les fonctionnalités propres au mobile qui sont offertes par le système. C'est un point très important à considérer, car selon le type d'application que l'on veut élaborer, certaines cross-plateformes ne permettent pas d'avoir accès aux composants hardware du mobile.

Les applications natives

Dans le cas des applications natives, le développeur implémente une application pour une plateforme spécifique en utilisant le SDK de développement correspondant. Par exemple, les applications Android, sont programmées en Java et donne accès aux fonctionnalités du Framework Android fourni ses éléments

d'interface graphique. Or les applications iOS utilisent le langage objective-C et le Framework d'Apple. Dans le cas de plusieurs plateformes, il faut développer chaque plateforme séparément, d'où les solutions de cross-plateformes.

Le principe des cross-plateformes

Le principe des cross-plateformes est d'offrir la possibilité aux développeurs d'implémenter leur application une seule fois et qui peut être exécuté sur plusieurs plateformes. Le principe est le même pour les différentes approches, le code source est indépendant des plateformes ciblés.
On peut identifier trois environnements d'exécution différents, le navigateur web, la combinaison entre le web et le native, et les environnements indépendants.

Les solutions cross-plateformes

Les applications web sont implémentées à base de HTML, CSS, JavaScript et utilisent le navigateur web comme environnement d'exécution. Quand on utilise cette approche, on développe une seule fois notre application comme un site web optimisé pour le mobile. L'optimisation doit prendre en compte les différentes tailles d'écrans et respecter la philosophie du mobile.
On peut accéder à l'application web via une url par le biais d'un navigateur web. Néanmoins, les applications web ne peuvent pas être installés sur le mobile et ne peut pas accéder aux composants hardware du téléphone, comme le capteur GPS ou l'appareil photo.
Les nouvelles versions de Html5 n'apportent pas de solutions concernant ces contraintes. Les applications web ont l'apparence d'un site web et sont différents au niveau du comportement des applications natives.

Pour résoudre ces problèmes d'accessibilités aux fonctionnalités hardware et d'installation, l'approche hybride est une solution possible, c'est la combinaison entre le web et les fonctionnalités natives. Le principe est d'encapsuler une page web par le composant webview dans une application native, en offrant donc les accès à l'API et aux fonctionnalités de la plateforme.
Le code source est toujours à base de HTML, CSS, Javascript. A l'exécution, la webview se lance par le navigateur et tous les appels hardware de l'API sont délégués à la partie native. Le fait d'encapsuler le code source dans un corps native, permet aussi l'installation de l'application.

Enfin nous avons les solutions qui ont leur propre environnement d'exécution, complètement indépendant. Ils utilisent uniquement leurs l'API, qui se charge de faire les appels aux composants natives, cela demande beaucoup plus de travail mais offre plus de liberté. Le processus de développement est assez long, les applications sont encapsuler par le Framework et ensuite déployé comme application native.

Le marché du cross-plateformes

On a vu brièvement les différents types de cross-plateformes, voici la liste non exhaustive des cross-plateformes sur le marché de nos jours:

- PhoneGap: http://phonegap.com
- sencha Touch: http://www.sencha.com/products/touch/
- Mono iOS/Android: http://xamarin.com/
- Appcelerator: http://www.appcelerator.com/
- Adobe AIR: http://www.adobe.com/devnet/devices.html
- Qt : http://www.qt-project.org/
- RhoMobile: http://rhomobile.com
- Marmelade: http://madewithmarmalade.com

- Corona: http://www.coronalabs.com
- MoSync: http://www.mosync.com
- jQuery Mobile: http://jquerymobile.com/
- jQTouch: http://www.jqtouch.com/

Problématique

Face à toutes ces technologies de cross-plateformes, le choix n'est pas évident, il n'y a pas de bonne ou mauvaise cross-plateformes. Il faut que la cross-plateforme soit adapté au besoin de l'application. Le choix dépend de diverses critères, le but dans la suite de notre étude, sera de fournir un moyen au développeur de réaliser un choix parmi les différentes solutions de cross-plateformes présentes sur le marché.

L'objectif n'est pas de réaliser une comparaison entre les différentes solutions, mais de fournir un outil pour choisir une cross-plateforme en se basant sur un ensemble de critères.

II. Modèle de critère

- Démarche

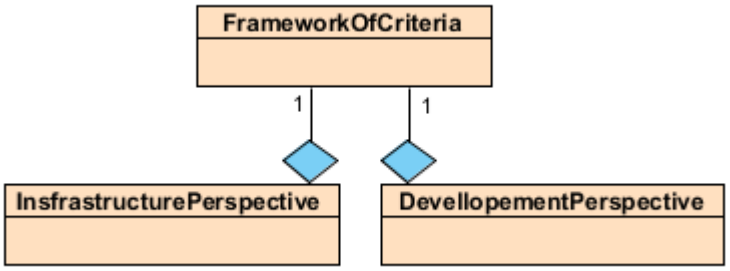
Notre travail dans cette partie vise à fournir un modèle de critères sur lequel les développeurs pourront se baser et réaliser un choix en fonction de leur besoins.

Cette approche peut-être très utile dans le cas où l'on souhaite migrer d'une plateforme à une autre, et où on voudrait prendre en considération tous les changements et pouvoir limiter les problèmes d'intégration.

Nous avons vu précédemment qu'il y'a différentes formes de cross-plateformes, qui se différencie par rapport à l'environnement d'exécution. C'est intéressant de savoir dans le processus de développement et d'apprentissage ce qui convient le mieux à un débutant. Les applications natives serviront de point de référence.

- Modélisation

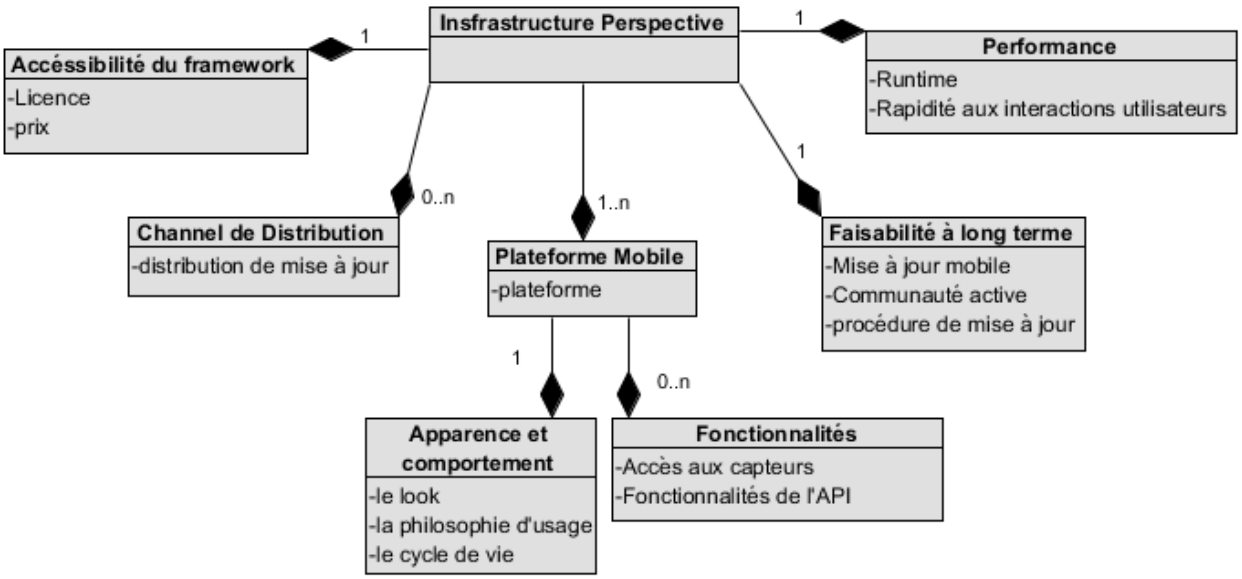
Les critères utilisés sont principalement issues des problèmes typiques rencontrés par la communauté de développeurs en ligne. Les différents critères ont été structurés selon deux perspectives, infrastructure et développement.



Infrastructure perspective

Ce que nous définissons par le terme de "infrastructure", c'est l'ensemble des critères relevant du cycle de vie d'une application cross-plateforme, son usage, les opérations et fonctionnalités qu'elle possède.

Le modèle suivant défini sous forme de diagramme la partie infrastructure:

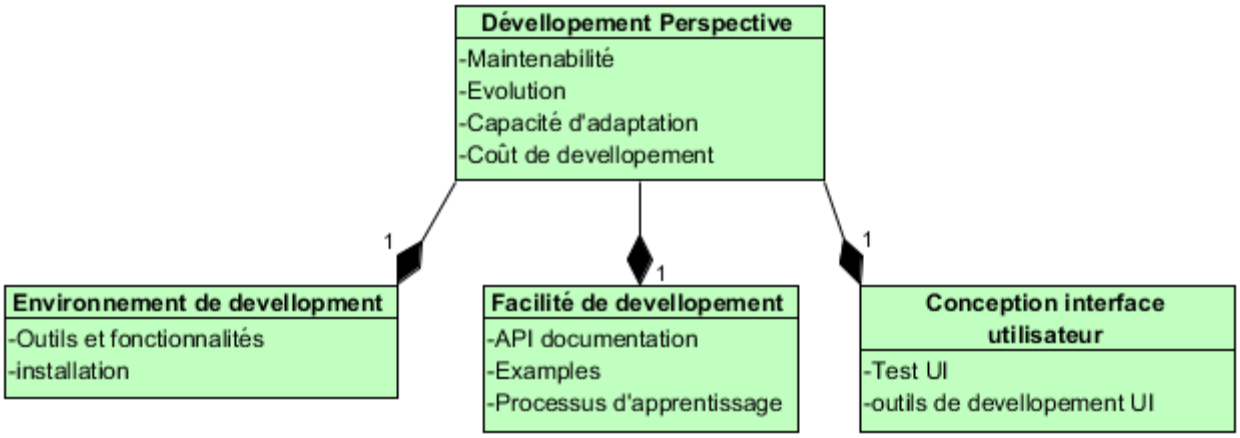


Les éléments qui composent le digramme sont détaillés selon le tableau suivant:

Accessibilité du Framework	Si il s'agit d'un Framework gratuit ou open source, si le développeur est libre de créer une application commerciale.
Channel de distribution	Les moyens de distribution possible, la possibilité de publier une application sous le store de la plateforme, et les moyens supplémentaires que propose le Framework.
Plateforme Mobile	Le nombre de plateformes mobile que supporte le Framework
Apparence et comportement	Il est important que les utilisateurs retrouvent le style native à travers le look de l'application, la philosophie d'utilisation ainsi que le cycle de vie des applications par rapport au comportement native.
Fonctionnalités	La possibilité d'accéder aux capteurs du mobile et l'accès aux fonctionnalités de l'API de la plateforme.
Faisabilité à long terme	Pour le développement sur du long terme, il est important de considérer le fait de pouvoir supporter les prochaines versions d'API des plateformes. Avoir des cycles de mise à jour rapide, une communauté de développeurs active, des supports commerciaux qui contribuent au développement de la cross-plateforme.
Performance	La fluidité et la rapidité des applications, l'efficacité de l'environnement d'exécution, réactivité face aux interactions utilisateurs.

Développement perspective

Cette partie regroupe les critères relatifs au processus de développement. Le processus de développement a été modélisé de la forme suivante:



Environnement de développement	Evaluation de la maturité et des fonctionnalités de l'environnement de développement. Les outils de développement offert par le framework (IDE, débuggeur, émulateur, auto-complétion, les tests). C'est aussi la facilité d'installation et les configurations requises pour arriver à un environnement prêt à l'emploi.
Facilité de développement	La facilité dans le processus d'apprentissage grâce aux supports à disposition, une API bien documentée, des exemples à dispositions, des références vers les problèmes les plus connus. Les concepts intuitifs dans le développement et les compétences qu'un développeur doit acquérir.
Conception Interface Utilisateur	Le processus pour la création des interfaces utilisateurs, les outils à disposition comme un éditeur de WYSIWYG ou bien la possibilité de tester son interface utilisateur sans avoir à déployer à travers un téléphone ou un émulateur.
Maintenabilité	La maintenabilité du code source est défini par la complexité, visibilité et la facilité à comprendre un programme.
Capacité d'adaptation	La modularisation du Framework, c'est à dire sa capacité à prendre en charge de grands projets, et permettre à des équipes larges de travailler ensemble.
Evolution	Détermine la réutilisabilité du code source pour des développements futurs, si on souhaite changer d'approche ou apporter des améliorations par la suite.
Coûts de développement	Les coûts de développement dépendent de la rapidité du processus de développement mais aussi du type d'investissement comme par exemple entre un développeur Javascript et un développeur Java.

Grâce à ce modèle de critères, on va pouvoir évaluer les différentes solutions, dans la suite de notre étude nous nous intéresserons à deux cross-plateformes différentes PhoneGap et Titanium Mobile.

IV. PhoneGap & Appcelerator

Nous avons évoqué les différentes solutions de cross-plateformes précédemment et défini un modèle de critères. On va se concentrer sur deux technologies de cross-plateformes, PhoneGap et Appcelerator que l'on appelle aussi Titanium Mobile, pour réaliser une analyse selon notre modèle de critères. On va évaluer sur une échelle de 1 à 6 les critères, 1 pour très bien et 6 pour très pauvre.

1. PhoneGap

Phone Gap est une approche dites "hybride" créé par Nitobi Software puis acquis par Apache Foundation, a été développé sous une licence open source, par une communauté de développeurs issu de grandes entreprises informatiques comme IBM ou Microsoft. En utilisant PhoneGap, les développeurs continuent à implémenter leurs applications en Html, Css et Javascript. Actuellement, PhoneGap possèdent un moteur Cordova, c'est donc une distribution de Cordova. PhoneGap fourni une API JavaScript, le code source est exécuté dans une webView à l'intérieur d'une application native qui permet l'accès aux fonctionnalités hardware. Pour notre étude nous utilisons la version 3.5.0 de PhoneGap et la version jQuery Mobile 1.4.5.

Evaluation - Infrastructure perspective

Accessibilité du Framework	*2*

PhoneGap avec JQuery Mobile est Framework open source sous la licence Apache 2.0, respectivement une licence GPL/MIT. Commercialiser une application ne coûte rien.
Nitobi propose des formules de support allant de 20 à 1600 euro par mois, qui inclut du support téléphonique et de la résolution de bugs.

Channel de distribution	*2*

Contrairement aux applications web mobiles, PhoneGap construit un corps d'application native, les applications peuvent être téléchargées et installé, on peut soumettre une application via le store de la plateforme, ainsi que des mises à jour. Néanmoins Apple, garde le droit de décliner les applications qui sont à base de web.

Plateforme Mobile	*1*

Aujourd'hui PhoneGap supporte 7 plateformes différentes : iOS/Android/BlackBerry OS,Windows Phone, HP WebOS, Symbian et Bada. JQuery Mobile supporte toutes ces plateformes également, les différentes versions de plateformes sont généralement bien supportées à l'exception de veilles versions pour les plateformes BlackBerry et Windows Phone. PhoneGap couvre 96% des plateformes environ en termes de parts de marché.

Fonctionnalités	*2*

PhoneGap permet l'accès à la majorité des fonctionnalités de l'API :
Accéléromètre, appareil photo, camera, boussole, contacts, évènements, géolocalisation, globalisation, inAppBrowser, accès multimédias, notification, splashScreen. Cependant, il reste un certain nombre de processus offert par les API propre au mobile qui ne peuvent pas être utilisés.

Apparence et comportement	*3*

PhonesGap n'a pas accès aux composants natifs, tout est réalisé à partir de CSS, en cherchant à imiter l'apparence native, mais il existe des librairies qui fournisent des composants en javascript/Css très similaire au composant native comme Onsen UI. Le cycle de vie est bien meilleur que les applications web car on a la possibilité de détecter des évènements par rapport au statut de l'application, il faudra par contre gérer les actions à réaliser pour chaque évènement.
D'un point de vue expérience utilisateurs, ça reste pauvre par rapport à ce qu'on peut obtenir avec une application native, la philosophie d'usage est relativement long et compliquer à reproduire, par exemple la pile de navigation. Il faut aussi considérer que d'un navigateur à un autre on peut avoir des différences, on retombe sur les problématiques du web. Il faut absolument traiter est le mode offline qui n'est pas adapté du tout au webview.

Faisabilité à long terme	*2*

Depuis que PhoneGap est devenu un projet Apache, il y'a régulièrement des nouvelles versions et des bugs résolus. Il y'a une grande communauté active, car elle est lié à la communauté du web, on trouve beaucoup de ressources et de librairies open source qui servent à enrichir ou compléter l'API. On peut toujours recourir à un support de la part de Nitobi, seule contrainte est que ce support est payant.

Performance	*3*

Le déploiement vers les plateformes est relativement facile et rapide. Par contre, les performances d'une application sont aléatoires selon la performance des appareils cibles, et des navigateurs, la gestion du cache et le chargement des pages n'est pas très optimale.

Evaluation - Développement perspective

Environnement de développement	*2*

Comme un développeur web, on a le choix de choisir son environnement de développement, pas tous les IDE offre une auto-complétion de l'API PhoneGap. On retrouve les problématiques du web en terme de débugage. On peut utiliser les émulateurs classiques pour nos plateformes mobiles comme Genymotion ou des extensions peuvent être installées pour bénéficier d'un émulateur dans un navigateur.
Il existe des plugins pour pouvoir réaliser des tests unitaires comme Test Studio ou QUnit. Si on ne souhaite pas installer tous les SDK de toutes les plateformes que l'on veut on peut utiliser PhoneGap Build, c'est un service qui compile l'application pour les différentes plateformes dans le Cloud, et ensuite on peut facilement télécharger l'application.

Facilité de développement	*2*

La documentation de PhoneGap est clair, structurée et complète. Elle offre dans la plupart des cas un exemple long et un exemple court, et pour les problèmes connus, une méthode spécifiques aux plateformes pour y remédier.
La documentation de JQuery est équivalente en termes de qualité. Mise à part l'API PhoneGap il n'y a pas de compétence supplémentaire à acquérir. Pour un développeur web, c'est très intuitive comme développement. On peut aussi développer des plugins natifs idéals pour les développeurs mobiles, mais cela ne vaut pas un développement spécifique.

Conception Interface Utilisateur	*1*

Comme pour des sites web, concevoir une interface utilisateur est facilement réalisable en utilisant un navigateur standard et un éditeur WYSIWYG comme Adobe Dreamweaver.

Maintenabilité	*1*

Excepté le code additionnel pour accéder aux fonctionnalités hardware, contrairement aux applications web, les applications hybrides ne nécessitent pas de lignes de code supplémentaires.
Implémenter une application avec PhoneGap, généralement les codes sources sont bien synthétisés et clairs grâce à jQuery Mobile.

Capacité d'adaptation	*2*

Dans le cadre d'applications mobile, ce critère est respecté du moment où les modifications sont très minimes.

Evolution	*2*

Un projet PhoneGap peut-être, à condition qu'aucunes fonctionnalités hardware ne soient utilisées, exécuté comme un site web mobile. On sait que le web est une technologie qui évolue très vite et donc les applications PhoneGap peuvent très bien aborder une nouvelle approche.

Coûts de développement	*1*

C'est plus ou moins équivalent à du développement web, avec un petit temps additionnel consacré à l'implémenté aux accès hardware.

2. Appccelerator, Titanium Mobile

Titanium Mobile présente une approche différente, il possède son propre environnement d'exécution. On n'utilise pas de Html/Css pour implémenter l'interface utilisateur. Le principe de Titanium est de fournir une machine virtuelle JavaScript permettant d'accéder au système natif, et ainsi développer des applications natives en JavaScript. L'interface utilisateur est complètement construite via le code, les développeurs utilisent du JavaScript pour construire l'interface et pour implémenter tout le modèle de données en utilisant l'API Titanium. La version étudiée est la 3.4 .

Evaluation - Infrastructure perspective

Accessibilité du Framework	*5*

Titanium Mobile est une communauté d'édition open source, qui est limitée en fonctionnalités. Les fonctionnalités supplémentaires sont accessibles via une souscription. La souscription inclus également un support en termes de documentation avec un accès à Appcelerator developer center.

Channel de distribution	*1*

Les applications Titanium Mobile peuvent être distribuées via les différents app store sans difficultés.

Plateforme Mobile	*4*

Titanium Mobile supporte seulement trois plateformes, iOS, Android et BlackBerry. Android et BlackBerry sont moins bien supportés, une grande majorité des méthodes de l'API sont seulement pour iOS.

Fonctionnalités	*2*

Le spectre de fonctionnalités qu'offre Titanium Mobile est équivalent à celui de PhoneGap.

Apparence et comportement	*2*

Contrairement à PhoneGap, Titanium intercepte une application en JavaScript et va créer des composants graphiques natives. Pour créer des éléments graphiques, il est nécessaire de maitriser l'API Titanium Javascrip. Construire une application avec un look natif nécessite beaucoup moins d'effort qu'une application web. Le cycle de vie d'une application peut-être facilement implémenté.

Faisabilité à long terme	*3*

La communauté semble être beaucoup moins active que PhoneGap, les réponses aux questions se font chaque semaine. Les mise à jours et la résolution de problèmes sont réguliers, Titanium Mobile est entretenu par une seule compagnie, leurs perspectives sur du long terme dépendent de leur stratégie.

Performance	*5*

Au démarrage, l'application ne diffère pas de ceux créés avec les autres solutions. Les performances du runtime sont lourdes dût au moteur Javascript, à chaque interaction utilisateur, on passe par les éléments natives au Javascript.

Evaluation - Développement perspective

Environnement de développement	*2*

On dispose d'un IDE dédié fourni par Appcelerator, Titanium Studio qui est basé sur Eclipse, il offre l'auto-complétion sur toute l'API Titanium Mobile, il prend en charge le déploiement sur émulateur ou téléphone. L'installation de l'environnement de développement est simple mais il faut installer séparément le SDK des différentes plateformes.

Facilité de développement	*3*

La documentation de Titanium Mobile est complète, il y'a une multitude d'exemples. Cependant, la progression en termes d'apprentissage est relativement longue au début, car il faut prendre en main l'API Titanium.

Conception Interface Utilisateur	*2*

Titanium Studio n'offre pas d'éditeur WYSIWYG pour créer l'interface, toute l'interface est créer via le code, ce qui est très long en temps de développement.

Maintenabilité	*3*

Une application peut-être facilement modularisé, généralement on a affaire à des codes relativement long.

Capacité d'adaptation	*2*

La modularisation d'une application, permet à celle-ci d'être très adaptable. On peut séparer les fichiers de code et faire un simple appel pour les utiliser, on peut avoir différentes fenêtres d'exécutions traitant deux morceaux indépendant de l'application. Néanmoins, communiquer des données entre deux fenêtres d'exécutions est un peu lent.

Evolution	*5*

Le code source d'une application ne laisse pas la possibilité d'évoluer vers une autre approche car il y'a généralement trop d'appel aux méthodes de l'API Titanium Mobile. Les évolutions envisageables sont uniquement celles que propose Titanium Mobile.

Coûts de développement	*5*

Le développement avec Titanium Mobile requière beaucoup de compétence sur le Framework, et demande beaucoup d'expérience. La conception des interfaces utilisateur est seulement possible avec un émulateur ou un appareil, ce qui est très pénible.

V. Synthèse

L'intérêt principal de l'utilisation d'une plateforme de développement multiplateforme comme PhoneGap et Titanium Mobile réside en deux points :

- Pouvoir déployer l'application sur les magasins afin de bénéficier de ce canal de distribution et de communication.
- Réduire les coûts de développements en mutualisant du code. Cela est bien assuré par ces deux outils à condition de bien les utiliser et de limiter la part des développements spécifiques à chaque plateforme. On rogne sur l'expérience utilisateur et la richesse fonctionnelle.

Si on regarde l'ensemble des notations que l'on a attribuées précédemment, PhoneGap se distingue de Titanium face à notre modèle de critères. Ce modèle de critères nous aide à poser les bonnes questions vis-à-vis d'une plateforme.

PhoneGap supporte aujourd'hui 7 plateformes contre 3 pour Titanium, par contre Titanium fournit un environnement de développement complet, là ou PhoneGap manque d'outillage.
Les deux plateformes sont extensibles, Titanium plus facilement que PhoneGap, mais c'est déconseiller d'aller dans cette direction car cela devient très coûteux à maintenir (on estime en général à 20% la part maximale de code spécifique tolérable), et nécessite en outre encore plus de compétence que de faire des applications natives à chaque plateforme.
En pratique, les développements PhoneGap nécessitent des adaptations pour chaque plateforme, dont les capacités et l'expérience utilisateur diffèrent fortement. Alors que Titanium Mobile utilisent des composants natives, au niveau expérience utilisateurs, les applications semblent être natives, BlackBerry est encore récent et disponible uniquement sur Windows. En terme de fonctionnalités, autant Titanium que PhoneGap présentent un panel assez riches de nos jours.

D'un point de vue personnel, j'ai une expérience confirmée en Android, débutante en iOS et intermédiaire pour Windows Phone, la solution des cross-plateformes est très pratique dans le cadre de petites applications. Sur le marché des applications, un grand nombre applications sont issues de cross-plateformes. Je reste quand bien attaché aux processus natives, mes pratiques de développement ne sont pas vraiment applicable dans le cadre d'applications cross-plateforme. Néanmoins, c'est une solution qui satisfait le besoin de couvrir la grande majorité des plateformes. J'ai tout de même une préférence envers les plateformes basé sur des langages standards comme PhoneGap, ce sont des langages déjà connus, aucune compétence supplémentaire n'est requise.