DA MOTA Anthony - Comparaison de technologies : PhoneGap VS Cordova

I. Introduction

Dans une période où la plasticité peut aider à réduire les coûts de développement de projets comme des applications mobile, il peut être intéressant pour une société de miser sur l'utilisation d'un Framework pouvant faciliter le portage d'une application d'un système à un autre. C'est le cas de Adobe PhoneGap notamment, ainsi que de Apache Cordova.

PhoneGap est une technologie qui a été d'abord développée en 2009 par la société Nitobi SoftWare, puis par la suite été donnée à Apache, qui en fera Apache CallBack, puis Apache Cordova. Nitobi a été acquise par Adobe et devient Adobe PhoneGap, qui se base maintenant sur Cordova, cette dernière servant en quelque sorte de noyau, de moteur, et lui ajoute des outils. Le nom de PhoneGap évoque naturellement l'univers du développement mobile, tandis que Cordova n'est ni plus ni moins que la ville d'origine du local de Nitobi SoftWare.

Dans le cadre du module *plasticité et adaptabilité des interfaces,* nous allons comparer ces deux technologies, en expliquant au mieux les divers principes utilisés en commun, et les quelques différences qui les séparent.

II. Principe

Le principe des deux technologies est de packager des applications web et d'utiliser le moteur de rendu du navigateur natif du téléphone pour pouvoir les exécuter dans un composant étant une WebView. L'intérêt devient alors de pouvoir écrire du code utilisant le couple HTML/CSS ainsi que le Javascript, afin d'écrire le code d'applications compatibles à la fois avec Android, iOS, Windows Phone ou d'autres systèmes d'exploitation destinés aux mobiles.

Apache Cordova et Adobe PhoneGap couvrent la majorité des fonctionnalités offertes par une application native, mais ont besoin de plugins spécifiques pour pouvoir gérer certains services (caméra, boussole). En réalité, ces plugins sont des bouts de code natifs dédiés à chaque OS, et l'appel des fonctions associés en Javascript va en fait appeler le code approprié au système (Android, iOS...).

Les limites sont alors notre capacité à programmer en langage natif. En effet, lorsqu'un plugin ne livre pas le résultat voulu, ou n'existe tout simplement pas, il faudra alors développer la brique manquante dans les différents langages voulus, et de l'intégrer au projet PhoneGap comme étant un nouveau plugin.

III. De nombreuses similarités

III.1. Avantages et inconvénients

L'utilisation de ces deux Frameworks peut avoir plusieurs avantages pour un développeur. Par exemple, une société souhaitant développer rapidement une petite application affichant des informations sur une entreprise va préférer cette solution. En effet, le code sera rapidement porté sur les différentes plateformes mobiles, et la simplicité de l'application est bien adaptée aux possibilités offertes par les deux technologies.

Un autre avantage est la faciliter avec laquelle jouer avec certains capteurs par exemple, pour peu que le plugin associé existe bien. La simplicité de syntaxe pour certaines fonctions comme déclencher les vibrations du téléphone est un gain de temps considérable.

Des inconvénients peuvent cependant apparaitre et devenir rapidement contraignants pour le développeur. Par exemple, une fonction très simplifiée permet de prendre une photo. Néanmoins, si le programmeur veut développer un appareil plus customisé et avoir plus de contrôle sur les différentes fonctionnalités, il devra se tourner vers une solution d'application native.

Les projets PhoneGap et Cordova sont également très lourd (de l'ordre de 2MO), sans compter les ajouts qui pourront être faits (pages, images, ...).

De plus, selon la charge de traitement à réaliser par l'application, comme la quantité d'animations, une application PhoneGap va être moins robuste qu'une version native. Enfin, le moteur utilisé pour afficher l'application dans une WebView reste un peu moins performant qu'un navigateur classique, ce qui peut donner une impression de manque de fluidité à l'utilisateur.

Il est également important de prendre en compte le fait que les différents navigateurs des téléphone peuvent produire un résultat différent les uns des autres. Par exemple, un effet visuel pourra s'afficher d'une façon imprévue, ou un geste tactile pourra être mal interprété.

Enfin, Cordova est PhoneGap seront un jour amenés à devenir obsolète du fait que la plupart des API développées seront dans le futur proposées directement par les navigateurs (accès aux capteurs du téléphone par exemple).

III.2. Compatibilité

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 6.0+	Blackberry 10	Windows Phone 8	Ubuntu	Firefox OS
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓
Compass	X	✓	✓	X	✓	✓	✓	✓
Contacts	✓	✓	✓	✓	✓	✓	✓	✓
File	✓	✓	✓	✓	✓	✓	✓	X
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	X	✓	✓	✓	X
Network	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✓

Figure 1 : Cordova - Extrait du tableau des services gérés en fonction des OS pris en charge

	Amazon-fireos	Android	blackberry10	iOS	Ubuntu	WP7 (Windows Phone 7)	wp8 (Windows Téléphone 8)	WIN8 (Windows 8)	paciarelli
Cordova CLI	✓ Mac, Windows, Linux	✓ Mac, Windows, Linux	✓ Mac, Windows	✓ Mac	✓ Ubuntu	✓ Windows	✓ Windows	✓	✗
Embedded WebView	✓ (voir détails)	✓ (voir détails)	✗	✓ (voir détails)	✓	✗	✗	✗	✗
Plug-in Interface	✓ (voir détails)	✓ (voir détails)	✓ (voir détails)	✓ (voir détails)	✓	✓ (voir détails)	✓ (voir détails)	✓	✗
API de la plate-forme									
Accéléromètre	✓	✓	✓	✓	✓	✓	✓	✓	✓
Appareil photo	✓	✓	✓	✓	✓	✓	✓	✓	✓
Capture	✓	✓	✓	✓	✓	✓	✓	✗	✗
Boussole	✓	✓	✓	✓ (3 G +)	✓	✓	✓	✓	✓

Figure 2 : PhoneGap - Extrait du tableau des services gérés en fonction des OS pris en charge

On remarquera que la plupart des principaux systèmes d'exploitation pris en charge sont communs aux deux technologies, ce qui ne fait pas de la compatibilité un argument de comparaison. Cette compatibilité est d'ailleurs ce qui fait la force de ces deux choix technologiques, car elle permettra de faciliter la vie à un développeurs voulant porter son application sur beaucoup de périphériques mobiles différents.

III.2. Intérêts en terme d'adaptabilité et plasticité

Ces deux techniques présentent un intérêt évident en terme de plasticité. En effet, elle permet au développeur de faciliter son travail pour réaliser des applications s'adaptant aux différents supports et systèmes d'exploitation. Nous avons donc là affaire à une adaptation à la conception.

En effet, PhoneGap comme Cordova permettent d'utiliser du code HTML/CSS et Javascript qui se lance sur la plupart des systèmes d'exploitation mobile grâce aux principes évoqués précédemment. Par exemple, un scénario-type pourrait être l'implémentation de la prise de photo par un développeur. Le seul travail qu'il aura à faire sera d'intégrer le plugin existant à son projet, appeler une fonction très simple d'utilisation et laisser le Framework transposer le tout en code natif selon la plateforme. D'autre part, le code CSS permet également d'adapter le design en fonction des différentes tailles d'écran.

III. Principales différences

III. 1. Une comparaison qui fait débat

Le fait que Cordova serve de base à PhoneGap rend les deux technologies très similaires. Beaucoup d'articles sur le Web débâtent d'ailleurs de l'existence de réelles différences entre les deux techniques. La technologie PhoneGap appartenant à Adobe, celle-ci peut intégrer des outils qui lui sont dédiés. Dans ce contexte, beaucoup d'accordent à dire que Apache Cordova est plus ou moins la version "communautaire" de PhoneGap, même si les deux restent open source et gratuites.

III. 2. Des ajouts qui peuvent faire la différence

La syntaxe étant similaire à quelques appellations près, c'est du côté des outils fournis que l'on peut trouver des différences. En effet, l'outil le plus utilisé par les développeurs PhoneGap est Adobe PhoneGap Build, qui aussi le vrai seul outil exclusif départageant les deux technologies. Celui-ci permet de compiler directement le code dans le Cloud, celui-ci pouvant se lancer comme une application Android, Windows Phone ou autre. Sont seulement nécessaires une archive .zip par exemple, ou l'outil GitHub. AdobePhoneGap Build permet également aux développeurs de profiter de fonctions facilitant le travail collaboratif, la gestion automatique de dépendances, la mise à jour des plugins.

L'outil se chargeant de compiler dans le Cloud, cela permet également l'installation des SDK nécessaires sur sa machine par exemple. L'inconvénient d'un tel outil est qu'il devient payant si l'utilisateur ne veut pas que son code soit public.

Un autre ajout de PhoneGap est l'outil PhoneGap Developper app, qui permet de voir directement les modifications apportées sur un téléphone mobile.

Enfin, PhoneGap a annoncé fin Mars l'outil PhoneGap Enterprise, qui est destiné aux professionnels et permet de gérer plus facilement gérer les différentes versions de l'application (test, production). L'intégration de Adobe's Marketing Cloud est également un plus permettant d'avoir un jeu d'outils destinés à gérer efficacement les statistiques d'utilisation de son application, ou encore leur monétisation.

D'après les retours utilisateurs, Cordova a à son avantage une interface légèrement plus simple, et des mises à jour plus fréquentes.

III. 3. Quelle technologie choisir ?

Au vu du peu de différences existant à l'heure actuelle, difficile de départager PhoneGap et Cordova lorsque l'on veut produire une application cross-platform. La majorité des développeurs s'accordent simplement à dire qu'il suffit de considérer le fait de vouloir ou non utiliser un ou plusieurs des outils Adobe cités précédemment. Si la réponse est oui, la solution est donc Adobe PhoneGap.

Par exemple, une application nécessitant de travailler efficacement en équipe pourra peut-être bénéficier de l'outil Adobe PhoneGap Build, qui fournira un outil de travail collaboratif, et évitera à tous les développeurs de s'accorder sur des versions de SDK.

IV. Conclusion

Pour conclure cette comparaison de technologies, nous pouvons dire que peu de différences apparaissent entre Adobe PhoneGap et Apache Cordova. Du fait que PhoneGap utilise Cordova comme noyau et y rajoute des outils Adobe, il suffira de se demander si ceux-ci pourraient oui ou non apporter quelque chose à notre projet avant de sélectionner une technologie.

Même si les deux solutions sont un gain de temps pour beaucoup de petits projets d'applications simples, il ne faut pas oublier les différentes limites qu'elles posent. Le tableau qui suit est en quelque sorte un récapitulatif de tout ce qui est ressorti de cette étude.

	Apache Cordova	**Adobe PhoneGap**
Compatibilité avec les principaux OS mobile		
Simplicité de syntaxe		
Cross-platform		
Adobe PhoneGap Build		
Adobe PhoneGap Enterprise		

Robustesse des applications		

Bibliographie

http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name/

http://ionicframework.com/blog/what-is-cordova-phonegap/

https://build.phonegap.com/

http://www.smile.fr/

http://www.wikipedia.com/