Frameworks de Développement Cross-Platform Mobile

Thibaud Destouches – Marceau Lacroix

ISTIC

Table des matières

1 Introduction

Le Smartphone est aujourd'hui "le nouvel ordinateur personnel", de plus avec l'arrivée des tablettes, ces terminaux tactiles remplacent de plus en plus les classiques ordinateurs personnels. Le nombre d'appareils nomades va croissant chaque année, et devrait ainsi dépasser le nombre de PC dans les prochaines années.

1.1 Problématique

Le marché des appareils mobile augmente chaque année, et avec lui la fragmentation des systèmes d'exploitations. Chaque OS mobile à en effet un langage de programmation différent (JAVA sur Android, Objective C/Cocoa sur iOS, .NET sur Windows Phone...) et des APIs différentes, à tel point qu'il est très compliqué pour un seul développeur de maîtriser correctement la création d'application sur l'ensemble des plateformes existantes.

OS	Q2 2012	Q2 2012 %	Q2 2011	Q2 2011 %	Evolution
Android	104.8	68.1%	50.8	46.9%	106.5%
iOS	26.0	16.9%	20.4	18.8%	27.5%
BlackBerry OS	7.4	4.8%	12.5	11.5%	-40.9%
Symbian	6.8	4.4%	18.3	16.9%	-62.9%
Windows	5.4	3.5%	2.5	2.3%	115.3%
Linux	3.5	2.3%	3.3	3.0%	6.3%
Others	0.1	0.1%	0.6	0.5%	-80.0%
Grand Total	154.0	100.0%	108.3	100.0%	42.2%

Aujourd'hui, si une entreprise souhaite développer une application avec un maximum de visibilité, et donc fonctionnant sur un maximum de terminaux différents, il faut développer entièrement 3 ou 4 applications différentes. Le temps de développement s'en retrouve donc multiplié, l'équipe de développement augmentée, et donc le Time-to-Market devient trop important.

1.2 Enjeux

Les Frameworks de développement cross-platform mobile cherchent à résoudre ces problèmes. En abstrayant le développement à une instance plus haute, et en générant une application pour chaque système d'exploitation cible, on souhaite ainsi réduire drastiquement le temps de développement en ne programmant plus qu'une seule application, et en générant les instances à la demande. Ce principe est aussi appelé "Write once, run everywhere"

Ces frameworks ont donc une réelle utilité au niveau Marketing, ils permettent de faire de grande économie en réduisant le temps de développement et le nombre de développeur, tout en réduisant le Time-to-Market d'une application.

2 Etat de l'art

De nombreuses solutions sont disponibles aujourd'hui pour répondre à la problématique du développement d'applications mobiles multi-plateforme. Le tableau en annexe regroupe l'ensemble des informations que nous avons pu trouver sur ces différentes solutions. Ce tableau montre, selon les frameworks, quelles sont les fonctionnalités offertes (ou non) aussi bien en terme de support d'un système que d'accès aux fonctions avancées du terminal.

Parmis l'ensemble des solutions existantes pour le développement mobile cross-platform, nous avons retenu quatre solutions (plus ou moins) OpenSource, deux commerciales et une pour laquelle nous n'avons pas d'information sur la licence. Il est assez facile de séparer cet ensemble de frameworks en deux sous ensembles : ceux générant du code natif pour chaque OS et ceux générant une web view de l'application.

2.1 Code natif

L'interet du code natif est sa rapidité d'execution. En effet réaliser des applications en utilisant les standards fourni par le système d'exploitation est bien plus efficace que de construire des interfaces en HTML/css/JavaScript puisque celles-ci sont interprétées à l'exécution par le navigateur (ou équivalent allégé) du périphérique. Cette catégorie regroupe la plupart des solutions qui nous ont intéressé dans cette étude. Pourtant ces frameworks ne génèrent pas entièrement du code natif ; les applications générées via les framework sont des hybrides, c'est à dire que le code produit est un mélange de code natif et de web view. Cette dernière est utilisée pour les composants que le framework ne sait pas gerer pour la plateforme. Une autre approche (celle de Corona) est d'orienter le framework vers une seule utilisation principale ; Corona est orienté vers le jeu sur mobile et fourni ainsi un dsl de développement pour les jeux. Ce code est ensuite "complié" vers du code natif pour chaque OS, voir pour chaque périphérique (les jeux étant bien plus sensibles que la plupart des autres applications aux problèmes de performances)

2.2 Web view

Comme nous l'avons remarqué précédement, la solution "de facilité" pour le développement cross-platform est l'interface web. Celle-ci est en effet supposée identique (ou au moins très similaire) d'un navigateur à l'autre. En contrepartie, l'utilisation de ces technologies impacte les performances de l'application mais impose aussi d'autres limitations. Parmis celles-ci on trouve notamment l'accès au matériel spécifique du périphérique (accéléromètre, GPS, Bluetooth, ...). C'est ici qu'interviennent les frameworks que nous avons étudié (PhoneGap, titanium et Rhodes), ils permettent en effet d'utiliser le matériel spécifique au périphérique au travers d'API spécifiques. Nous avons aussi pris en compte un framework "pur web" dans notre comparatif (Sencha) ; celui ci ne permet pas

d'accès au matériel spécifique du périphérique, en revanche le développement de l'interface est très simple pour un developpeur familier des technologies web

2.3 Voind

La dernière solution que nous avons examiné est "voind" ; il s'agit d'un framework développé par un étudiant lors de sa thèse [2] pour lequel nous n'avons pas d'informations sur la licence étant donné que le site officiel n'est plus accessible. le principe de voind, contrairement aux autres frameworks est de générer un squelette pour chaque application native. Le developpeur doit ensuite remplir ce squelette avec les fonctionnalités non supportées par le framework. On ne retrouve pas, avec cette solution, le principe du "write once run everywhere" mais le temps de développement de chaque application est considérablement réduit. Dans l'exemple pris pour sa thèse (une application permettant de commander au restaurant via son smartphone), le framework génère environ 50% du code de chaque application, mais le developpeur doit connaitre chaque plateforme pour pouvoir y ajouter le code spécifique.

3 Conclusion

Les frameworks de développement cross-platform sont une solution intéressante pour permettre le déploiement d'une application sur un ensemble de systèmes d'exploitations importants. Cependant ces frameworks sont limité du fait de la grande segmentation des matériels et des OS mobiles, il ne faut donc pas espérer concevoir une application complexe à partir de ceux-ci. Il sont toutefois totalement recommandé dans le cas d'application dite "vitrine", ne nécessitant pas de grandes interactions, ni l'accès à des compostant spécifiques, autant matériels (GPS, Accéléromètre, Appareil Photo...) que logiciels (liste des contacts, application téléphonique...).

Finalement, il est encourageant de constater que des frameworks comme Corona, certes spécifique à un type developpement donné, empreintent la voix de la compilation vers du code natif. Ils tirent ainsi pleinement partie des spécificités matérielles et logicielles de chaque périphérique.

Références

[1] Timo Paananen, *Smartphone Cross-Plaform Frameworks*. Jamk University of Applied Sciences, avril 2011.

[2] Mathieu Bruning, *Native Cross-Platform Mobile Application Development Using Voind*. Faculty EEMCS, Delft University of Technology, novembre 2011.

[3] Appcelerator Inc, *http ://www.appcelerator.com/*.

[4] Rhomobile Suite, *http ://www.motorola.com/*.

[5] PhoneGap, *http ://phonegap.com/*.

[6] Sencha, *http ://www.sencha.com/products/touch/*.

[7] Corona, *http ://www.coronalabs.com/*.

Annexe

Nom	**PhoneGap**	**APPcelerator titanium**	**Rhodes**	**Adobe AIR for Mobile Devices**	**Corona**	**Sencha**	**voind**
iOS	oui	oui	oui	oui	oui	oui	oui
Android	oui	oui	oui	oui	oui	oui	oui
Blackberry	oui	oui	oui	oui	non	oui	oui
Windows Phone	oui	non	oui	non	non	oui	oui
Bada	oui	non	non	non	non	oui	oui
Symbian	oui	non	oui	non	non	oui	oui
web mobile	oui	oui	oui	oui	non	oui	oui
ajout de code natif	oui	NC	NC	non	oui	non	impératif
code natif/web view	web view	hybride (orienté natif)	hybride (orienté web)	NC	natif	web view	natif
GPS	oui	oui	oui	oui	NC	non	oui
caméra	oui	oui	oui	NC	NC	non	oui
application native	oui	oui	oui	oui	NC	non	oui
accéléromètre	oui	oui	non	oui	NC	non	oui
Look & feel natif	non	oui	non	non	NA (orienté jeux)	non	oui
Prix	Open Source	Open Source (support payant)	Open Source (support payant)	Commercial (gratuit)	Commercial (350e/an)	Open Source (support payant)	NC
Language de développement	HTML CSS JavaScript	DSL	ruby HTML Javascript	flash flex	DSL	HTML CSS JavaScript	languages multiples