Introduction

1. Pourquoi utiliser un outil multiplateforme pour la programmation mobile ?

Depuis la sortie de l'iPhone, les smartphones sont devenus des ordinateurs à part entière. Et même si avant leur arrivée, il était déjà possible de créer des applications pour les téléphones qui existaient, faire des applications pour les smartphones a ouvert de nouvelles possibilités en termes de créativité. Aujourd'hui, il existe plusieurs systèmes d'exploitation mobiles pour ces téléphones intelligents. Même s'ils proposent tous des fonctionnalités proches, ces systèmes diffèrent totalement les uns des autres sur plusieurs points. Parmi eux, les plus populaires de nos jours sont iOS et Android. À côté d'eux, on trouve d'autres systèmes comme Windows Phone, BlackBerry OS, QNX, Firefox OS, etc.

À cause de cette diversité et des particularités de chaque système, créer des applications mobiles est devenu un vrai challenge si l'on souhaite cibler deux ou plusieurs de ces plateformes. Dans le cas d'une société qui veut créer une application pour diverses plateformes et qui a les moyens de se payer des développeurs spécialisés dans chacune d'elles, le problème ne se pose pas. Dans le cas d'une autre société, d'une petite équipe de développeurs ou encore d'un développeur indépendant qui n'a pas les moyens ni le temps, mais qui veut cibler plusieurs plateformes avec une même application, le travail risque de devenir très fastidieux. Écrire une bonne application pour une seule plateforme demande déjà beaucoup de travail. Si en plus de ça, il faut réécrire la même application pour d'autres plateformes alors on n'est pas sorti de l'auberge. La difficulté de viser différentes plateformes avec la même application réside principalement dans les quatre points suivants :

- *Le langage de programmation et l'API utilisée diffèrent totalement d'une plateforme à une autre*. En exemple, programmer pour iOS se fait en Objective-C ou en Swift, Android et BlackBerry OS se programment en Java (avec des API différentes), Windows Phone se programme en C#.
- *L'environnement de développement utilisé*. Même s'il est possible de nos jours d'utiliser un même environnement de développement pour programmer dans plusieurs langages différents, certains environnements sont souvent dédiés et plus adaptés à un langage. Ainsi, le développeur iOS utilisera de préférence l'environnement Xcode, le développeur Android utilisera Android Studio ou Eclipse, le développeur Windows Phone utilisera Visual Studio, etc.

- *L'interface utilisateur*. Chaque plateforme mobile propose une manière propre à elle de naviguer entre les interfaces, de présenter les menus, d'interagir avec une application, etc.
- *Le système d'exploitation de la machine de développement*. Aussi dommage que cela puisse être, il n'est pas possible de développer pour certains systèmes mobiles si l'on n'a pas le système d'exploitation approprié sur son ordinateur. En exemple, il faut avoir un Mac pour pouvoir créer des applications pour iOS, un PC avec Windows pour créer des applications pour BlackBerry OS et pour Windows Phone.

Depuis quelques années, des outils qualifiés d'outils de développement multiplateforme sont apparus et permettent de s'affranchir de ces quatre sources de difficultés majeures. Ces outils proposent d'utiliser un seul langage pour développer des applications fonctionnant sur plusieurs plateformes mobiles. Avec une promesse aussi alléchante, on ne peut qu'être emballé à la découverte de ces outils qui présentent malheureusement aussi leurs limites. Parmi eux, Codename One est l'un des plus aboutis, innovants et stables. Il propose d'écrire avec un code unique en Java des applications qui s'exécuteront sur cinq plateformes mobiles. Ainsi, il est possible d'affirmer qu'utiliser un outil multiplateforme permet de gagner en temps d'apprentissage, de conception et aussi en coût monétaire.

Les lignes qui suivent vont vous introduire Codename One, qui est un framework Java créé par deux ingénieurs israéliens réputés pour être des spécialistes en développement mobile bien avant même l'arrivée des smartphones. Après un historique du framework, nous ferons un tour d'horizon de ses principales caractéristiques et en examinerons les avantages et les inconvénients, avant d'enchaîner sur les choses sérieuses. Sur ce, bonne initiation à Codename One.

2. Historique de Codename One

Tout commence en 2007 à Sun MicroSystem (la société fondatrice de Java) avec le souci de créer une bibliothèque d'interfaces graphiques riche en J2ME. Le premier objectif de cette bibliothèque était de réduire les problèmes de fragmentation qu'il y avait au niveau des plateformes mobiles J2ME et BlackBerry. En plus de la volonté de résoudre ce problème, la bibliothèque devait aussi être flexible, riche en composants graphiques (ce qui n'était pas le cas de la bibliothèque d'interface fournie par défaut par l'API CLDC de J2ME en Java). Ainsi naquit la bibliothèque LWUIT créée par l'ingénieur israélien Chen Fishbein qui travaillait chez Sun. Étant donné que cette bibliothèque (qui est open-source) apportait une réelle solution à un problème contraignant, d'autres développeurs ont rejoint le projet. L'un d'eux était Shai Almog (développeur Java expérimenté et consultant auprès de Sun MicroSystem à l'époque). Il aidait Chen Fishbein

(l'initiateur du projet) à faire évoluer LWUIT. Peu de temps après, Il laissa sa casquette de consultant pour rejoindre finalement la société. Quelques années plus tard, la société Oracle racheta Sun MicroSystem, un rachat qui allait peser sur l'évolution de LWUIT qui constituera plus tard la base de développement de Codename One.

Nous sommes maintenant en 2011 et les deux ingénieurs et amis Chen Fishbein et Shai Almog ont envie de faire évoluer considérablement leur bibliothèque et de toucher aussi les plateformes mobiles présentes dans les smartphones (Android, iOS, BlackBerry, Windows phone en plus du J2ME d'origine). À cause de sa politique et pour certaines autres raisons, ils décident de quitter Oracle pour créer leur startup et réaliser leur projet. Ils présentent alors leur démission cette même année et commencent leur projet qu'ils nomment Codename One (ce qui sera aussi le nom de leur startup). Puisque LWUIT est open-source, ils clonent son code source, changent le nom des packages, font des modifications intensives, créent des portages vers d'autres systèmes mobiles et ajoutent de nombreuses fonctionnalités. Codename One est né, et sa première version bêta est lancée publiquement en janvier 2012.

Étant open-source, en plus d'avoir un système de plug-ins qui rend son évolution flexible, Codename One n'a cessé d'évoluer depuis le début de sa création. Toujours bien accueilli par les développeurs Java principalement, Codename One est actuellement un outil complet permettant aux développeurs Java de concevoir des applications mobiles multiplateformes de qualité pour les plateformes iOS, Android, Windows phone, BlackBerry et toujours J2ME (pour les développeurs ciblant les téléphones Nokia Asha pour ne citer que ça).

3. Pourquoi Codename One ?

Codename One est un framework écrit en Java permettant de faire de la programmation mobile multiplateforme. Il est open-source et se présente sous la forme d'un plug-in disponible pour les trois environnements de développement majeurs en Java (NetBeans, Eclipse, IntelliJ IDEA). Il permet de cibler cinq plateformes mobiles (iOS, Android, Windows, BlackBerry, J2ME) avec un code unique et a aussi pour particularité d'utiliser le cloud pour la compilation. Cette utilisation du cloud permet aux développeurs de s'affranchir de l'installation de divers SDK ou de posséder un système d'exploitation spécifique pour programmer des applications pour certaines plateformes mobiles. Codename One produit toujours du code natif donc il n'y a aucune raison de se soucier des problèmes de performance. Le plug-in est composé de quatre parties majeures que voici :

Une API

Cette API contient toutes les classes nécessaires à la conception d'une application mobile et est écrite en langage Java.

Un designer

Fourni sous forme de logiciel, le designer permet de concevoir visuellement une interface d'application, de gérer la traduction d'une application en diverses langues, de créer des thèmes, de manipuler des images, etc.

Un simulateur

Il permet de tester ses applications sur son ordinateur. N'étant pas un émulateur, ce simulateur est rapide à l'exécution et embarque des outils pratiques pour tester en profondeur les applications.

Un serveur de compilation dans le cloud

Ce serveur permet de compiler en ligne les applications écrites avec Codename One. Cette manière d'effectuer les compilations a ses avantages et ses inconvénients sur lesquels nous reviendrons.

L'une des grandes particularités de Codename One est son architecture dite *lightweight* qui apporte une meilleure solution aux problèmes de fragmentation des plateformes mobiles. Un composant lightweight dans ce cas-ci est un composant écrit entièrement en Java qui dessine sa propre interface tout en gérant ses propres événements et états. Cette manière de faire apporte un énorme avantage en termes de portabilité puisque le même code est exécuté sur toutes les plateformes en plus d'autres avantages. Les composants graphiques de Codename One sont infiniment personnalisables.

L'API de Codename One couvre une immense catégorie de fonctionnalités. On peut y trouver ce qu'il faut pour faire par exemple les tâches suivantes :

- l'interface graphique ;
- la manipulation de la vidéo et de l'audio (enregistrement comme affichage) ;
- le stockage ;
- l'accès à la caméra ;
- la manipulation d'une base de données SQLite ;

- la manipulation des services web ;
- le réseau ;
- l'accès au cloud ;
- la lecture des QR et Bar codes ;
- l'internationalisation et la localisation ;
- les notifications ;
- la manipulation des contacts ;
- l'accès aux pages web ;
- la monétisation ;
- l'accès aux réseaux sociaux ;
- la géolocalisation ;
- les tests unitaires ;
- la création de thèmes personnalisés ;
- et beaucoup d'autres fonctionnalités à découvrir.

Cette liste n'est pas exhaustive donc pas d'inquiétude si vous ne trouvez pas une fonction particulière non citée ci-dessus.

Codename One est orienté exclusivement vers la conception d'applications métiers et n'a pas du tout été pensé pour créer des jeux. Quelques efforts sont en train d'être faits dans ce sens, mais rien de concret n'est encore disponible de ce côté.

Pourquoi utiliser Codename One et pas les autres outils de développement mobile multiplateforme ? Qu'a-t-il de mieux que les autres outils de ce genre ? Ce qu'il faut d'abord savoir c'est que ce langage n'est pas l'outil parfait et magique sans inconvénients qui permet de tout faire en un clic. Comme pour chaque outil ou framework, celui-ci aussi a sa propre philosophie. Une fois cette dernière acquise, son utilisation devient simple et est un vrai régal.

Voici une liste de quelques avantages et inconvénients.

Avantages

- Simulateur fourni et s'adaptant automatiquement au visuel et comportement des plateformes mobiles supportées. En plus de remplir sa tâche de base, ce simulateur contient un ensemble d'outils pratiques pour les tests avancés.

- Compilation dans le cloud. Ceci a pour avantage de se passer de l'installation et de la configuration de divers SDK sur son ordinateur. Cela évite aussi d'avoir à utiliser un système d'exploitation spécifique pour compiler pour certaines plateformes comme le fait d'avoir un Mac avant de compiler pour l'iOS ou un PC sous Windows avant de compiler pour Windows Phone ou Blackberry. Cette méthode peut aussi faciliter le travail en équipe en permettant à chaque membre d'avoir accès aux compilations des autres.
- Présence d'un éditeur visuel d'interfaces graphiques. Peu d'outils multiplateformes (et même certains outils natifs) fournissent un éditeur graphique. Cet éditeur permet de gagner un temps considérable dans la conception des interfaces d'une application, ce qui aide à se concentrer uniquement sur les fonctionnalités.
- Utilisation du langage Java, qui est un langage stable, structuré, connu et très documenté sur le web et dans les livres est un avantage non négligeable.
- Possibilité d'obtenir une interface au visuel unique sur toutes les plateformes ou une interface propre à chaque plateforme.
- Présence d'un logiciel nommé Codename One Designer. En plus de pouvoir créer une interface graphique à la souris et de créer des thèmes visuels pour une application, ce logiciel fournit d'autres possibilités permettant de simplifier diverses choses. Un chapitre entier lui est consacré dans cet ouvrage et il s'agit du chapitre Codename One Designer.
- Possibilité de créer soi-même des plug-ins liés au framework et permettant d'étendre ses fonctionnalités.
- Contrairement à d'autres frameworks de développement mobile multiplateforme qui utilisent les technologies web pour l'interface graphique ou qui traduisent leur code dans le but d'utiliser les API graphiques d'origine, Codename One dessine ses propres composants graphiques quelle que soit la plateforme visée. Cela permet de résoudre les problèmes de fragmentation liés aux interfaces sur différentes plateformes.

Inconvénients

Certains des avantages de Codename One apportent aussi certains inconvénients. Les voici :

- Le fait de compiler dans le cloud est bien, mais il serait aussi intéressant et plus pratique de compiler en local sur son propre ordinateur. Même si ce n'est pas l'option fournie par défaut, il est quand même possible de le faire après quelques bidouilles qui sont qualifiées de complexes par les créateurs du framework.

- Codename One est fourni avec un simulateur et non un émulateur. N'étant pas un émulateur, les comportements des applications ne sont pas forcément reproduits fidèlement comme sur les plateformes réelles. Cela a pour inconvénient de voir par exemple une fonctionnalité refusant de s'exécuter (ou s'exécutant mal) sur le simulateur mais s'exécutant sur la vraie plateforme et vice versa.
- Le fait que Codename One dessine ses propres composants et y applique ensuite des thèmes au lieu d'utiliser directement les composants des SDK natifs peut être vu par certains comme un inconvénient mais ce point reste quand même très discutable.
- Le fait de ne pas pouvoir effectuer de compilation si le serveur en ligne est indisponible. Ce genre de situation est rare mais peut arriver.

Comme vous pouvez le remarquer, Codename One a aussi ses inconvénients mais le fait d'être un projet open-source et d'avoir une communauté ouverte et généreuse permet d'avoir un outil très évolutif et en constante amélioration.