Frameworks mobiles cross-platform

(Évaluation réalisée au 13/01/2016)

Une application cross-platform, en opposition à une application native, est une application qui une fois développée est capable de s'exécuter sur plusieurs plates-formes comme Android, iOS, ou encore Windows Phone. Dans l'idéal on ne développe qu'une seule fois et on exécute partout. Ce qui n'est pas tout à fait vrai, on mutualise une grande partie du code mais jamais la totalité. Les applications cross-platform sont généralement développées à l'aide des technologies web, à savoir le HTML, le CSS et le JavaScript.

Tout n'est évidemment pas parfait, on reproche souvent aux applications cross-platform, bien que ça soit de moins en moins vrai, d'être moins performantes que les applications natives. En réalité, tout dépend de la complexité et des fonctionnalités de votre application. L'autre chose que l'on reproche souvent, c'est d'être moins immersives d'un point de vue purement graphique. En effet les guidelines graphiques d'une application Android, iOS et Windows Phone sont très différentes ce qui permet de les identifier au premier coup d'oeil. Dans le cas d'une application cross-platform, on a souvent tendance à créer un design standard qui sera exactement le même pour toutes les plates-formes.

- Apache Cordova (ex PhoneGap)
- Titanium
- Sencha
- Xamarin
- Ionic
- React Native

Apache Cordova (ex PhoneGap)

https://cordova.apache.org/

Surement le plus connu des outils cross-platform car également un des plus simple d'utilisation. En effet l'écriture d'une application Cordova se fait en HTML, JavaScript et CSS. L'outil s'occupe ensuite d'encapsuler ce code dans une WebView et génère une application native qui sert de conteneur pour lancer la WebView. De plus, et c'est là tout l'intérêt de Cordova, le JavaScript peut accéder aux fonctionnalités natives du mobile (contacts, photos, notifications...), ce que les applications web classiques ne peuvent pas.

Pour

- Développement en langage web donc pas de nouveau langage à apprendre, vous êtes déjà prêt à commencer !
- Cordova utilise une architecture en plugin pour les accès aux fonctions natives. C'est à dire qu'il suffit de piocher dans une bibliothèque les plugins dont nous avons besoin pour rajouter des fonctionnalités.
- Open source et gratuit, que demander de plus ?
- Supporte avec plus ou moins de fonctionnalités quasiment tous les OS mobile (iOS, Android, FfOS, BB, WP...).
- Produit mature, un des premier dans ce secteur, et il est toujours là.

Contre

- Les développeurs devront prêter particulièrement attention aux performances, ainsi qu'à la bonne adaptation de leur UI aux mobiles.
- La majorité des modules dépends de la communauté, il peut arriver que certain plugins ne s'exécutent pas sur un OS en particulier, ou qu'ils ne soient plus à jour.

Titanium

http://www.appcelerator.org/

Titanium est un acteur de longue date dans le développement cross-platform. Il a subit beaucoup de mutations avant de devenir ce qu'il est aujourd'hui, une plate-forme complète d'outils et services pour le développement d'applications natives. Titanium utilise Alloy, un framework MVC, les modules créés ainsi sont facilement réutilisables dans différentes apps, réduisant le temps de développement. Tout le code est écrit en JavaScript, celui est combiné avec l'API Titanium afin d'être interprété en tant que code natif dans l'environnement d'exécution du mobile. Titanium n'utilise pas de WebView comme ses concurrents. L'interface de l'application est 100% native et vous pouvez accéder aux fonctionnalités natives du mobile.

[18/01/2016] : L'éditeur français Axway rachète Appcelerator

Pour

- L'outil Titanium de base est open source -- ce n'est pas très clair, dans leur option commerciale il est écrit qu'il faut un compte payant pour pouvoir déployer -- ...
- Supporte iOS, Android, Windows Phone.
- Utilisation de l'UI natif de chaque OS, ce qui se traduit par une rapidité / fluidité des interfaces et un visuel adapté à chacun d'eux.
- Apps réellement natives

Contre

- ...cependant pour bénéficier de tous les outils de la plateforme Appcelerator (push, statistique, tests unitaires), il faut payer !

Sencha

https://www.sencha.com/

Sencha utilise une API JavaScript et une approche MVC pour créer des apps. La programmation s'effectue exclusivement en JavaScript, le HTML / CSS étant généré par des widgets Sencha que l'on configure en Js. Le tout est ensuite compilé vers l'OS mobile choisi grâce à ...PhoneGap.

Pour

- Facilité de développement grâce aux widgets
- Concepteur d'interface visuel

Contre

- Prix exorbitant !

Xamarin

https://xamarin.com/

Xamarin est un framework qui permet le développement d'applications natives pour Windows Phone, iOS et Android en utilisant le C#. En sortie de la compilation, nous obtenons un binaire natif pour chaque plateforme cible. Le développeur commence par créer une base de code commune. Elle contient notamment la logique métier, le stockage en base de données, les appels réseaux, les éléments d'interface communs. Ensuite, un projet est créé par plateforme cible. Il contient l'interface graphique, la navigation et les composants propres à chaque SDK. Ainsi, on peut tirer parti des spécificités propres à chaque OS sans réduire l'expérience utilisateur.

Pour

- Chaque méthode d'un SDK (Android, iOS, Windows Phone) est encapsulée à l'identique en C#. Sur chaque plate-forme toutes les fonctionnalités sont conservées. Un développeur expérimenté en Android ou iOS s'y retrouve très rapidement car il écrit les mêmes noms de fonctions et utilise les mêmes classes.
- Performance égale au natif, UI natif.
- Concepteur d'interface visuel.

Contre

- Un développeur natif Java doit apprendre le C# pour utiliser Xamarin
- La nécessité de connaître les SDK de chaque plate-forme, un développeur Android ou iOS s'y retrouve facilement dans les SDKs mis à disposition. Il conçoit ses écrans et sa navigation comme il a l'habitude de faire. En revanche, un développeur sans expérience du mobile doit apprendre les subtilités propres à chaque plate-forme cible.
- Payant ! Xamarin nécessite obligatoirement une licence mensuelle / annuelle pour utiliser le produit.

Ionic

http://ionicframework.com/

Ionic est un framework HTML5 développé avec Sass et optimisé pour AngularJS. Il utilise Cordova pour packager le HTML / JavaScript dans une application native qui servira de conteneur à la WebView. Il inclus des composants et des contrôleurs spécifiques mobile. Et Cordova permet l'accès aux fonctionnalités natives du mobile.

Pour

- Développement en AngularJS.
- Open source et gratuit.
- Lignes de commandes pour démarrer un projet très rapidement.
- Client graphique disponible pour la gestion des apps, le LAB (création, compilation, test et déploiement).

Contre

- Projet encore jeune, mais en forte croissance, ne supporte que iOS et Android. Windows Phone et FireFox Os sur la roadmap.
- Concepteur d'interface visuel drag&drop mais payant pour l'export natif.

React Native

https://facebook.github.io/react-native/

Les développeurs de chez facebook sont partis du constat que les applications mobiles faites avec des technos web ne sont pas encore au niveau des composants natifs. De plus, il existe des éléments d'interfaces conçus spécifiquement pour ces plateformes (iOS et Android) et qui sont adaptés en terme de performance, d'interaction et de visuel. Plutôt que de réécrire un nouveau framework mobile, ils ont décidé d'utiliser Javascript et React comme couche d'abstraction sauf que au lieu de créer des éléments de DOM, on va ici piloter des composants natifs ! Et ce qui est assez intéressant, c'est que la syntaxe est exactement la même que dans un navigateur, seul les composants changent. Par exemple sous iOS, une div devient une View, un paragraphe un composant Text etc… Le système tire partie des processeurs multi-coeurs en mettant en place un moteur javascript en arrière plan. Celui ci va communiquer avec le thread principal de manière asynchrone, évitant ainsi de gêner le rendu graphique. Pour chaque plateforme un serveur permet de transformer les appels issus de React vers les composants natifs. Pour le style, pas de navigateur, pas de CSS. Toutes les bonnes pratiques largement éprouvées sont mises à rude épreuve puisque le style est donné aux composants directement en ligne et en javascript.

React Native est encore très jeune, la version Android vient tout juste d'être annoncée (sept 2015). Mais avec son approche complètement différente, il pourrait bien apporter sa petite révolution dans le monde des apps natives développées à partir de technologies web. A surveiller !