ARCHITECTURE MOBILE

WEB, NATIVE OU HYBRIDE :

CONSEILS TECHNIQUES DE LA RÉALISATION
À L'INDUSTRIALISATION DE VOTRE SERVICE MOBILE

Lorys Pognon,
Février 2013

// FORCES ET FAIBLESSES DES DIFFÉRENTS ENVIRONNEMENTS TECHNIQUES

L'écosystème mobile fait face à des questionnements concernant :

La prolifération des applications
La fragmentation des terminaux
L'industrialisation des développements mobiles
La sécurité en termes de stockage de données et d'échange d'informations
Le coût de réalisation

Toutes ces problématiques nous montrent à quel point, il est judicieux de mettre en place aujourd'hui et rapidement au sein d'une entreprise, un environnement technique mobile entièrement maîtrisé.

D'une manière générale nous disposons de trois approches pour développer une application mobile :

Ecrire des applications entièrement natives basées sur les plateformes existantes comme celles d'Apple, de Google ou de Microsoft…

Faire des applications web mobiles

Combiner les deux premières approches pour en faire des applications hybrides

Dans les lignes suivantes, nous allons présenter chacune des ces catégories d'application tout en mettant en évidence les outils que chaque plateforme met à disposition des responsables de projets mobiles.

1/ LES APPLICATIONS NATIVES

Les forces des applications natives

Les applications mobiles sont grossièrement de simples applications qui doivent accomplir des tâches spécifiques pour l'utilisateur en se basant parfois sur les capacités matérielles du terminal.
On aura toujours un avantage considérable à écrire une application native lorsqu'on désire s'intégrer totalement à une plateforme donnée. La force d'une application native réside donc dans sa rapidité, sa fluidité, sa performance pour autant qu'elle soit bien écrite.

Les trois plateformes majeures que sont iOS (Apple), Android (Google) et Windows Phone (Microsoft) facilitent énormément le développement, l'intégration et la visibilité des applications via leur système de boutique en ligne (Apple Store, Google Play et Windows Marketplace). Une fois l'application dans les stores, elle pourra aussi intégrer facilement les systèmes de paiement (achat en une fois ou souscription), de transaction et de gestion des utilisateurs mis en place par chaque plateforme.

Les boutiques deviennent le point central pour les utilisateurs désireux de chercher une application. Ces boutiques offrent des fonctionnalités de recherche, de catégorisation et de gestion de paiement. Même si certaines plateformes ne se portent pas garantes de la qualité des services disponibles sur leur boutique, la mise en exergue des avis d'utilisateurs tient une place primordiale dans le processus d'achat et de valorisation des meilleures d'entre elles.

Les applications natives offrent une expérience utilisateur uniforme et partagent la plupart du temps des composants utilisés par les applications installées par défaut sur le terminal par le constructeur. Ce qui permet d'avoir des applications avec un niveau d'expérience utilisateur supérieur aux applications web mobiles. Les utilisateurs des applications natives se retrouvent rapidement d'une application à une autre tout simplement parce que ces applications sont très souvent écrites à partir des composants natifs mis à disposition par la plateforme ciblée. Dernier point mais pas des moindres, la rapidité de navigation est rendue possible par le matériel et les composants natifs optimisés et mis à disposition par les différentes plateformes. Même si les accès réseaux peuvent introduire des délais lors de l'acquisition des données, ces derniers n'affecteront pas les temps de navigation ou d'affichage de l'interface utilisateur.

Les faiblesses des applications natives

Leur spécificité est une force mais c'est aussi une des faiblesses des applications natives. Les systèmes d'exploitation mobiles sont des environnements isolés à mémoire limitée. Il est donc difficile voire impossible qu'une application mobile native développée pour une plateforme s'exécute sur un terminal muni du système d'exploitation d'une autre plateforme.

Développer une application native pour plusieurs plateformes suppose la maîtrise de différents langages de programmation, la connaissance de différents frameworks, outils et l'achat des matériaux spécifiques dans certains cas (pour le développement sous iOS par exemple) comme le résume le tableau qui suit.

PLATEFORME	SYSTÈME D'EXPLOITATION	OUTILS	LANGAGE DE PROGRAMMATION
IOS	Mac Os	Mac OS, Windows, Linux	Mac OS, Windows, Linux
ANDROID	Mac OS Windows Linux	Eclipse IntelliJ IDEA	Java et Android SDK C# et MonoDroid
WINDOWS PHONE	Windows	Microsoft Visual Studio	C# et Microsoft .NET Framework 4
SYMBIAN	Mac OS, Windows, Linux	Qt Quick	C++ et Qt SDK
BLACKBERRY	Mac OS, Windows, Linux	Eclipse	Java et BlackBerry SDK

Lorsque l'on essaie d'améliorer la visibilité de son application, notamment à l'aide de stratégies SEO (search engine optimization) l'aspect natif peut poser des problèmes de référencement sur les moteurs tels que Google, Bing et autres.
Contrairement à un site web mobile où chaque page peut être indexée et exposée aux moteurs de recherche, les applications natives distribuées sur les différentes plateformes mobiles ne sont pas indexées. Néanmoins, on peut toujours créer un mini site web pour promouvoir l'application mobile native afin de rendre le lien de téléchargement disponible depuis les moteurs de recherche.

Dans quel contexte faut-il privilégier une application native?

Il est parfois difficile de faire le choix entre une application native ou web mobile.
Les lignes suivantes décrivent les scenarios à partir desquels prendre le parti de développer une application native peut être plus intéressant :

---- Le niveau de graphisme est assez important

---- Lorsque l'on désire utiliser la caméra du téléphone

---- Lorsque l'on désire utiliser le micro du téléphone

---- Un accès au répertoire des contacts du téléphone

---- Un accès à la librairie des médias du téléphone

---- Pour tirer avantage du système de paiement des différentes plateformes

---- Pour exploiter le système de notifications mis en place par certaines plateformes

---- Lorsque l'application utilise des composants qui doivent faire du traitement en arrière plan

---- Pour écrire des jeux....

2/ LES APPLICATIONS WEB MOBILES

Le développement web sur mobile n'est pas un sujet récent, on se souvient des premiers sites web mobiles réalisés avec les langages de balisage que sont le WML, le XMHTL-MP ou même encore le CHTML. Il est important de noter que les navigateurs d'un ordinateur de bureau et les terminaux mobiles sont deux outils opérant dans des environnements totalement différents. Un site web mobile peut être aussi facile à mettre en place qu'un site web classique pour autant qu'il reprenne les fonctionnalités simples et importantes du site web destinées aux navigateurs bureautiques.
Ainsi, un site web mobile peut être vu comme la version du site web bureautique designée et optimisée pour les terminaux à petites résolutions d'écran ou terminaux tactiles.

Les forces des sites web mobiles

La courbe d'apprentissage des bonnes pratiques du développement web mobile n'est pas aussi élevée, pour un développeur, car elle reprend une partie importante des techniques de développement classiques utilisées dans le cadre d'un site internet pour desktop. Les développements web mobiles sont en général des développements faits côté serveur. Ainsi on a, à disposition, un site déployé et totalement accessible par tous les terminaux mobiles munis d'un navigateur adapté. Ce qui répond au problème de développement spécifique qu'on peut être amené à résoudre dans le cadre des applications natives. De plus, un site web mobile nous permet de cibler plusieurs terminaux utilisés par les clients mobiles.
Par ailleurs, les sites web mobiles ne sont pas soumis aux conditions et processus de validation des plateformes mobiles. Il n'y a donc pas de délai entre le moment où le produit est fini et sa mise en ligne. Les nouvelles versions ou les mises à jour du site pourront par la suite être déployées à chaud et disponibles en temps réel.

Mais qu'est-ce qui nous empêche alors de faire du web mobile partout ?

En informatique, quelque chose de simple n'est pas forcément simpliste et le web mobile a aussi ses limites et ses faiblesses.
Premièrement, avec un site web mobile, il est difficile d'accéder aux fonctionnalités matérielles du téléphone même si aujourd'hui avec le HTML5 nous avons à disposition de nombreuses API d'intégration de données plus ou moins abouties pour y parvenir (Contact API, Calendar API, Gallery API, HTML5 Media Capture…).
Le problème de fragmentation cité plus haut ne concerne pas que le téléphone en tant que matériel mais aussi certains composants tels que les navigateurs. Ainsi un site web mobile peut s'afficher ou se comporter différemment d'un terminal mobile à un autre même s'ils sont tous basés sur le même moteur de rendu. Un site web mobile étant d'une manière générale une solution côté serveur (même si aujourd'hui il existe des pratiques permettant de déléguer une bonne partie des fonctionnalités entièrement côté client), l'accès à ces pages demande un accès au serveur de la part du téléphone. Ce qui génère des latences au niveau du réseau et provoque parfois des délais non négligeables lors de l'affichage des pages. Contrairement à une application native qui, même en cas d'absence réseau peut répondre de manière fluide en terme de navigation, la latence du réseau sur une application web mobile peut dégrader l'expérience utilisateur et donner l'impression d'une navigation pas forcément optimale.

Un site web mobile n'étant pas soumis au processus de validation de certaines plateformes et surtout n'utilisant pas les APIs qu'elles mettent à disposition, il est impossible, par exemple, de tirer profit d'un système de paiement et de gestion des utilisateurs offerts par ces plateformes.
Un site web mobile qui intègre du m-commerce doit implémenter entièrement (ou utiliser d'autres services tierces) et gérer tous le problèmes liés à la gestion des utilisateurs, aux transactions, aux contrats avec les différents acteurs du commerce (banques, plateformes de transaction, les systèmes de moyens de paiement - Visa, Mastercard, American Express, les contraintes et réglementations financières par pays…).

Pour cibler un nombre important de plateformes mobiles avec un budget limité, le choix d'un site web mobile est certainement le meilleur. Il faut tout de même garder à l'esprit que les fonctionnalités développées ne pourront pas toujours exploiter toutes les ressources matérielles du téléphone.
Par ailleurs, il sera plus complexe d'offrir une expérience utilisateur optimale et satisfaisante en terme de navigation.

Les mythes du web mobile

Une simple page web pourra faire l'affaire ! Peut-être, mais servir les terminaux mobiles avec une seule page web n'est pas une bonne idée. Déjà un site web mobile n'est pas toujours, contrairement à ce qu'on peut penser, un simple site avec une ou deux pages – ça peut aller bien au delà. Nous conseillons d'éviter les modèles single page interface (spi) qui sont très inspirés des applications bureautiques dans lesquelles une seule interface utilisateur primaire est ajustée et reconfigurée via des appels ajax (eg. Google web toolkit) pour faire du web mobile. Privilégiez plutôt le modèle full page request (fpr) où à chaque requête correspond le chargement d'une nouvelle page.
On peut bien évidemment faire du cache.

Le navigateur mobile se chargera du reste ! Même si le site internet bureautique peut être affiché sur le navigateur du téléphone moyennant des efforts de la part de l'utilisateur (scroll, pinch, zoom…) qui dégradent l'expérience utilisateur, il est nécessaire d'avoir un site mobile dédié, plus adapté aux smartphones et même aux téléphones classiques (oui, il y a encore beaucoup de personnes qui n'ont pas de smartphones).

Un seul site peut servir tous les terminaux ! Oui il est possible d'écrire une seule version d'un site web qui pourra servir tous les terminaux (ordinateur de bureau, tablette, smartphone, …) sauf que ce choix peut s'avérer mauvais par la suite tout simplement parce que l'architecture de l'information peut varier d'un support à un autre, ce qui peut rendre compliqué la maintenance du site. Il est important de penser à l'expérience qu'on veut offrir et si possible dissocier complètement le site web classique du site web mobile.
Si l'on dispose d'un site web ou d'une application web avec une quantité importante de contenus et de données (images, vidéos, publicité, flash…), il est conseillé d'opter pour un site web pour poste de travail, dissocié d'un site web mobile.

Quelles options du web mobile choisir ?

Le choix technologique n'est pas toujours évident et dépend parfois du site web bureautique existant et de ce qu'on aimerait offrir aux utilisateurs web mobiles.
Un site web mobile peut être abordé de plusieurs façons :

La stratégie dite « Adaptative mobile website » : Lorsque votre site web bureautique actuel est au départ basé sur des technologies web standards et modernes (CSS externalisé, usage de « div », web sémantique), on peut envisager la stratégie « Adaptative » tout simplement parce que l'idée derrière ce concept est d'ajuster le site web actuel aux écrans de terminaux mobiles en faisant usage de CSS (pour cacher, afficher, redimensionner, espacer certains éléments de la page). Si votre site web bureautique actuel est basé sur un CMS (Content Management System), il est judicieux de voir dans un premier temps si le CMS utilisé ne fournit pas de facto une version mobile du site. La plupart des CMS comme WordPress, Drupal, DotNetNuke offrent cette possibilité. D'une manière générale, si vous voulez que les utilisateurs web mobiles accèdent à la même information que sur leur navigateur bureautique, la stratégie « Adaptative » est le meilleur choix à faire pour autant que les technologies du site web actuel le permettent.

La stratégie « Dedicated mobile website » :
Si vous savez que les utilisateurs mobiles n'auront pas besoin d'accéder à la même quantité importante de contenus du site web bureautique, la stratégie du site web mobile dédié est la meilleure.
Elle vous permet de construire un site web mobile qui s'adresse et répond aux besoins spécifiques de votre stratégie marketing mobile. Elle permet aussi d'utiliser des contenus optimisés et ciblés pour les terminaux mobiles.

La stratégie « Mobile web app » : elle est plus adaptée aux sites web mobiles qui veulent offrir une expérience semblable aux applications natives des plateformes mobiles.

	ADAPTIVE MOBILE WEBSITE	DEDICATED MOBILE WEBSITE	MOBILE WEB APPLICATION
Pros	• Maintain only one website • Quick and least expensive to implement • Provide improved mobile user interface	• Good website performance • More perfected mobile UI • Fairly inexpensive to implement	• A website that can behave like an app • Less development cost than native app • Works across platforms
Cons	• No use of native device functionality • Not optimal performance • Some layout restrictions	• No use of native device functionality • Maintain two websites	• Not in app stores • Can't use all mobile features

Comment planifier un site web mobile?

Une fois la stratégie mobile adoptée (Responsive Web Design, Dedicated Mobile Website ou Mobile Webapp), il est temps de se demander s'il faut une seule version du site web mobile ou plusieurs. La fragmentation des terminaux mobiles, surtout au niveau des tailles d'écran, nous invite à poser ce genre de questions car le site web mobile sur un iPhone peut s'afficher différemment sur une tablette iPad ou sur un téléphone d'ancienne génération.

La réponse est simple – si vous êtes limité par le coût et que vous voulez cibler uniquement les terminaux mobiles modernes (smartphones et tablettes actuelles du marché), il ne sert à rien de créer différents sites web mobiles par profil de terminaux.
Un seul site web mobile suffira largement pour couvrir votre cible.

Par contre si vous ciblez tous les terminaux mobiles et surtout si vous déployez plusieurs versions du site web mobile aux différents terminaux mobiles, il est important de suivre les étapes suivantes :

Développer le profil des terminaux – en d'autres termes il s'agit de déterminer quels sont les terminaux mobiles sur lesquels le site web mobile doit être chargé. La meilleure méthode consiste à catégoriser les terminaux mobiles en groupes de profils à partir d'un certain nombre de paramètres tels que la taille des écrans ou le support multimédia de chacun.

Développer ou acquérir une base de données (base de connaissance) de terminaux mobiles – pour délivrer un contenu adapté au terminal mobile, il est nécessaire d'avoir un fichier de configuration ou une base de données qui contient des informations détaillées sur chaque terminal. Plusieurs entreprises créent et maintiennent leur propre base de données de terminaux mais vous pouvez aussi utiliser aussi des solutions open source comme WURFL (Wireless Universal Resource File) et bien d'autres.

Ecrire un programme qui peut délivrer la bonne version du site mobile au bon terminal mobile – le challenge à ce niveau est d'écrire un programme performant qui permet de faire correspondre les informations issues de la base de connaissance de terminaux avec les différentes versions ou le contenu du site web mobile. D'une manière générale ce programme est souvent créé par chaque entreprise en fonction des différents profils de terminaux qu'elle a identifié ou qu'elle veut cibler. Néanmoins il est possible d'utiliser des solutions open-sources telles que « Detect Mobile Devices » d'Andy Moore, « Open Source Mobile Phone Detection », « Apache Mobile Filter » ou bien « Device Altlas ». Sur le site de « Detect Mobile Devices » vous avez une page de génération de fonction de détection de terminaux mobiles dédiée à ce rôle. Pour les passionnés de Java vous avez des APIs sur le site Mobile ESP.

Développer des templates ou des «designs» qui peuvent être adaptés, aux différents terminaux ou groupes de profils de terminaux mobiles à la volée.

Tester, tester, tester et faire des ajustements si besoin

3/ LES APPLICATIONS HYBRIDES

L'application mobile hybride est à la frontière du monde natif et web mobile.
En d'autres termes, une application hybride est une application native qui offre la plupart de ses services via le web mobile.
Vous pouvez utiliser vos expériences et outils en développement web et services pour mettre en place une application web mobile qui peut répondre à des contraintes métiers spécifiques.

L'intérêt d'une application hybride est de réduire les coûts de réalisation des applications mobiles. Plusieurs organisations ou entreprises préfèrent développer en interne toute la partie métier ou « back » et déléguer la réalisation des applications mobiles (couche présentation) à des sociétés de services ou des agences mobiles.
Même si cette approche permet à court terme d'avoir des équipes dédiées et qualifiées pour réaliser les applications mobiles, il n'en demeure pas moins que le problème de coût persiste surtout quand on veut cibler toutes les plateformes mobiles.

	Expérience utilisateur		Maintenabilité		Coût		
	Visibilité de l'application sur le Market	Rendu des composants graphiques	Compétences nécessaires	Pérennité de la solution	Prix de la licence d'utilisation	Charge de développement côté Mobile	Charge de développement coté serveur
Site web mobile	☹	☹	☺	☺	☺	☺	😐
Application native	☺	☺	😐	☺	☺	😐	😐
Application hybride WebView	😐	😐	😐	😐	☺	☺	😐
Application hybride PhoneGap	☺	☹	😐	😐	☺	☺	😐

4/ L'ENVIRONNEMENT TECHNIQUE D'UNE APPLICATION NATIVE

Nous n'allons pas dans ce paragraphe aborder toutes les plateformes mobiles et frameworks qui aident au développement d'applications mobiles natives.
L'objectif est de se concentrer sur les trois majeures qui sont les plus utilisées dans un contexte « B2C » et qui ont sû faire leur preuve – il s'agit bien entendu de la plateforme d'Apple (iOS), celle de Google (Android) et enfin celle de Microsoft (Windows Phone).

L'environnement technique iOS

Lorsque l'on choisit d'écrire une application mobile native pour les terminaux munis de l'iOS, il faut au moins avoir un environnement comprenant les outils suivants :

Posséder un Mac OS X: Attention, éviter l'utilisation de machines virtuelles (VM) car c'est une violation de la licence Apple.

Télécharger et installer le SDK de l'iOS : Pour éviter tout souci, il est conseillé d'installer Xcode qui s'occupera de l'installation du SDK

Télécharger et installer Xcode : C'est l'environnement intégré de développement (IDE) le plus conseillé pour le développement des applications iOS.

Rejoindre le programme des développeurs iOS :
Il existe trois types de programme (développeur, entreprise et université).

---- Un développeur désireux de faire des tests sur un téléphone, doit payer une somme de $99/an, lors de la phase de développement ou de distribuer son application sur l'Apple Store.

---- Le programme Entreprise permet de mieux gérer votre équipe de développement en interne et vos distributions moyennant une somme de $299/an.
En souscrivant à ce programme vous avez la possibilité de rattacher autant de développeurs que vous voulez à votre compte. Ce qui vous permet d'avoir une gestion maîtrisée de votre équipe de développement. Ce programme vous permet de faire des tests sur des téléphones durant la phase de développement, de faire de la distribution « ad hoc » et surtout de pouvoir soumettre votre application à Apple pour validation. Dans un scénario d'entreprise, vous avez la possibilité de déployer votre application via votre intranet ou via internet sur les terminaux dotés au préalable d'un certificat de l'entreprise.

---- Le programme Universitaire est gratuit et permet aux universitaires ou institutions éducatives d'introduire le développement iOS au sein de leur équipe. Ce programme n'autorise pas la distribution des applications sur l'Apple store mais permet aux professeurs et étudiants, dans le même groupe de développement de pouvoir distribuer des applications depuis un serveur web ou via email.

Coder en utilisant l'Objective-C comme langage de programmation :
On peut bien entendu écrire une application mobile iOS à partir d'un langage autre que l'objective-C (Java, C#...) via des frameworks.
Un développeur Java peut se convertir à de l'objective-C sans aucun problème – le concept objet n'étant pas aussi différent que l'on peut penser même si ce n'est pas la même syntaxe.

L'environnement technique Android

Posséder un ordinateur muni d'une machine virtuelle Java : Il vous faut donc installer la JDK (Java Développement Kit).

Installer ANT comme système de « build » : Il est possible de compiler des applications Android en utilisant « Maven » – ce qui est le plus adapté pour certains systèmes d'intégration continue comprenant des projets qui sont à la base sous « Maven ».

Télécharger le SDK android et un IDE (Eclipse, Netbeans, Intelli J)

S'enregistrer au programme Android Developer pour un montant de 25$ payé une seule fois si l'on veut distribuer sur Google Play et bénéficier d'un certain nombre de services tels qu'une interface de gestion des ses applications, le « in-app billing » et bien d'autres…. On peut toujours distribuer une application Android sur les téléphones sans passer par Google Play. Il suffit tout simplement de signer l'application avec un certificat valide même auto généré.

Configurer son environnement

Environnement technique Windows Phone

Windows Phone est le successeur de Windows Mobile. C'est le nouveau système d'exploitation pour mobile de Microsoft. Son écosystème est similaire à celui de iOS (Apple) contrairement à celui d'Android (Google) en terme d'exigence et de contrôle sur la manière dont les applications doivent s'exécuter sur la plateforme. Même si la notion de fragmentation n'est pas pour le moment aussi flagrante dans l'univers Windows Phone, on imagine que, dans les années à venir, beaucoup de constructeurs de téléphone n'hésiteront pas à adopter cette plateforme mobile. Pour le moment, le développeur d'applications sous Windows Phone n'a pas beaucoup de résolutions et de variations en terme de capacité matérielle à gérer. La plateforme mobile est basée sur .Net (plus précisément sur Silverlight) avec pour langage de programmation le C#. Notons qu'avec Windows Phone 8, le développeur a le choix d'écrire son application en Visual Basic, C++ ou C#. Pour écrire des applications sous Windows Phone, le développeur a besoin au moins de :

Visual Studio Express comme IDE suffit largement :
Ceux qui possèdent Microsoft Visual Studio peuvent toujours l'étendre avec des additifs mobiles.

Installer Microsoft Expression Blend si l'on désire plus en terme d'expérience utilisateur et de design.

S'enregistrer au programme Windows Phone Developer (App Hub), un programme qui coûte $99/an et permet de tester les applications sur de vrais terminaux, et de publier les applications gratuitement ou moyennant un certain montant sur Microsoft Windows Phone Marketplace.

Développer vos applications en optant pour l'un des frameworks Silverlight ou XNA (Xbox New Architecture) qui est beaucoup plus adapté pour le développement des jeux.

Chaque plateforme propose ses outils, son modèle économique et un langage de programmation différent. Des trois plateformes, Apple est celle qui a atteint une maturité remarquable avec un système bien intégré et des outils de développement efficaces qui rendent le développement des applications beaucoup plus rapide. Le système de validation des applications mis en place par Apple et Microsoft est nécessaire même si contraignant.
Il permet de sécuriser davantage les applications mises à disposition pour les utilisateurs et surtout d'assurer une visibilité de l'application et de l'éditeur.

Tips… pour mieux entreprendre un projet d'application mobile natif

Stratégie de mise à jour – même si les plateformes mobiles
mettent à disposition des mécanismes de notifications pour les mises à jour d'une application, rien n'empêche de rajouter dans les applications, des alertes de mise à jour, surtout pour les applications qui ne passent pas par le canal de distribution des plateformes mobiles.

Cibler les versions d'OS les plus installées – par exemple à partir de l'iOS 4, ou à partir d'android 2.1, qui couvrent déjà 93% des terminaux Android.

Privilégier plutôt des applications universelles – c'est à dire un seul exécutable (.ipa) pour iPhone et iPad. La même logique s'applique aux applications Android. Il faut donc éviter les apks multiples si
possible. Il est vrai que l'écriture d'une application universelle peut être un vrai challenge pour le développeur mais cela vaut le coup dans certaines circonstances.

Bien connaître les guides de développement de chaque plateforme mobile.

5/ L'ENVIRONNEMENT TECHNIQUE D'UNE APPLICATION WEB MOBILE

Contrairement au développement d'application mobile native, nous ne sommes pas soumis à ce niveau, aux contraintes ou outils des différentes plateformes détaillées plus haut. D'une manière générale, une application web mobile doit avant tout prendre en compte les technologies et techniques standards du développement web tout en gardant en vue le côté mobile à ressource limitée. Nous n'allons pas donner ici une liste précise des outils à utiliser mais plutôt vous orienter sur les différentes possibilités et le choix à faire :

HTML 5 & CSS3 & JavaScript

Le HTML5 n'est pas seulement un ensemble de nouveaux éléments et attributs HTML. Un lifting est apporté au CSS et au JavaScript dans l'optique de toujours améliorer l'expérience utilisateur et de réduire la complexité du code au développeur. La plupart des smartphones et tablettes sont munis de navigateurs basés sur le moteur de rendu « WebKit » qui supporte de manière optimale ces trois technologies.

Faire du spécifique site web mobile

On a toujours grand avantage à avoir un site web mobile dédié lorsque les fonctionnalités et les ressources qu'on veut mettre à disposition ne sont pas négligeables. Avoir un site web mobile dédié facilite la maintenance et l'évolution des fonctionnalités répondant à une expérience utilisateur adaptée au mobile. Pour finir on peut contrôler de manière plus fine le SEO (Search Engine Optimization)

Usage de technologies et templates standards et matures

Plusieurs technologies et frameworks web qui font tout pour satisfaire les besoins du web mobile existent. Faire le choix de technologies simples ayant fait leur preuve et qui respectent des normes et standards, est le point de départ d'une application web mobile pérenne et de qualité. Comme pattern de développement, on a le choix entre le MVC (Model View Controller) ou le MVVM (Model View View Model). On pourrait s'orienter par exemple sur du HTML 5 mobile boilerplate comme template web de base avec du Twitter Bootstrap pour les pages HTML 5. L'avantage c'est que Twitter Bootstrap utilise déjà du CSS3 pour l'aspect Responsive Web Design et est facilement personnalisable. En ce qui concerne les librairies ou frameworks javascript, vous pouvez utiliser AngularJS, Backbone.js ou Knockout.js sans oublier JQuery ou zepto.
Pour finir, optez plutôt pour du simple JSP que pour du JSF.

Attention
Nous ne disons pas dans ce paragraphe de ne pas utiliser les frameworks web mobiles tels que (JQuery Mobile, Sencha, JQTouch, Rhodes et bien d'autres). Pour la mise en place d'un site web mobile d'une certaine envergure avec beaucoup fonctionnalités et qui doit répondre à une certaine charge, ces solutions ne sont peut-être pas le bon choix en terme de pérennité.

Éviter le model SPI et privilégier le FPR

Le modèle Single Page Interface (SPI) est un modèle très inspiré des applications bureautiques mais aussi du web dans lequel une seule interface utilisateur principale est ajustée et reconfigurée via des appels Ajax. L'exemple typique est le Google Web Toolkit. L'idée même d'avoir une seule page HTML qui servira de manière dynamique les différentes sections visuelles du site est séduisante mais l'expérience montre que l'on atteint très vite les limites ; Surtout sur les terminaux mobiles car on assiste rapidement à la dégradation des performances de la lisibilité... A contrario, un modèle basé sur une vraie navigation entre les différentes pages du site web mobile est le plus conseillé.
Ce modèle classique du web, connu sous le nom Full Page Refresh (FPR), est plus adapté pour du web mobile tout simplement parce qu'il est plus rapide de charger de petite pages HTML sans mettre en péril l'expérience utilisateur contrairement au modèle SPI qui conduit au chargement d'une seule unique et énorme page HTML. Le modèle SPI est plus adapté pour faire des prototypes ou des proofs of concept (PoC) des applications web mobiles.

Eviter de vouloir tout cibler

Lorsqu'on fait du web mobile, il est important de connaître ses clients et d'avoir si possible des statistiques sur le parc mobile qu'on voudrait adresser. Il est conseillé ensuite de catégoriser les terminaux mobiles à cibler en trois ou quatre grands groupes (Touch, Advanced, Basic, Legacy).
Aujourd'hui le message qui consiste à vouloir tout cibler n'est qu'un leurre et souvent une cause de dette technique. L'équipe d'Ippon Mobile conseille, dans un premier temps, de cibler les terminaux de la catégorie « Touch » et « Advanced » car c'est surtout ces clients qui accèderont plus facilement à vos services. C'est aussi ceux qui pourront vous aider à améliorer votre service et pourquoi pas drainer par la suite les deux catégories restantes vers l'expérience et le service que vous offrez.

Mise en cache des données et des pages

L'accès à un site ou à une page web mobile passe souvent par le réseau opérateur ou par internet. Très souvent, l'utilisateur est mobile et donc en déplacement. Les conditions d'accès ne sont donc pas toujours optimales. Il revient au fournisseur du service ou au développeur de définir des stratégies de cache pouvant aider à accéder rapidement aux ressources. En fonction de la technologie serveur adoptée, les mécanismes de mise en cache peuvent varier même si le principe est le même. La mise en cache peut se voir à trois niveaux : au niveau des bases de données, au niveau des services métiers mais aussi au niveau du client (le terminal mobile). Avec le HTML 5, la mise en cache côté client est devenue possible avec les API tels que le « Web Storage », le « Web SQL Database » et « Indexed DB ». Pour finir on peut aussi définir une politique d'accès des pages en mode déconnecté via le mécanisme « Offline Application Caching » introduit par le HTML 5.

Détection des terminaux

L'objectif principal d'un site web mobile est avant tout de servir plusieurs terminaux mobiles sans utiliser les technologies ou API d'une plateforme mobile dédiée.
Avec la prolifération des smartphones et tablettes, il est clair qu'un site web mobile devra s'adapter aux nombreux terminaux.

Il existe deux approches en termes de détection d'un terminal mobile

Une détection côté client et une détection côté serveur. Sans détection, on sera contraint à servir la même page web à tous les terminaux sans prendre en compte leurs caractéristiques.

---- La détection côté serveur passe par l'usage d'un Device Description Repository (DDR), qui est nécessaire si l'on veut connaître et vérifier un certain nombre de fonctionnalités des terminaux mobiles. Le plus connu et le plus utilisé demeure WURFL (Wireless Universal Resource File).

---- La détection côté client passe par l'usage de JavaScript et de CSS via l'usage des « media queries » par exemple avec le CSS 3. Si vous ciblez les terminaux de catégorie « Touch » et « Advanced », vous n'êtes plus obligés d'utiliser un DDR pour adapter vos contenus. La détection côté serveur va un peu plus loin car elle peut vous aider à connaître d'autres spécificités du terminal que vous ne pourriez pas avoir avec du JavaScript ou du CSS côté client.

Multiserving vs. Responsive web design (RWD)

Le « multiserving » implique que, pour une requête donnée, le serveur compose une page, qui est un assemblage de diverses vues optimisées pour un profil ou une catégorie de mobiles donnée. C'est donc une logique de génération des pages portées par le serveur. Le « Responsive Web Design » quant à lui implique qu'on a un seul site web qui fournit la même expérience utilisateur quel que soit le terminal mobile de l'utilisateur. C'est donc une logique de génération des pages portées par la page servie. Aujourd'hui la phase d'adaptation privilégie le RWD car il est simple à mettre en œuvre et se base sur les technologies côté client plus précisément le CSS3. L'intérêt du RWD réside dans le fait que différentes structures d'une page peuvent être servies à différents terminaux, en cachant ou déplaçant certains éléments de la page mais surtout, en utilisant des conteneurs (layouts) fluides et flexibles.

En conclusion

Lorsque que votre cible porte sur des terminaux mobiles de catégorie « Touch » et « Advanced », le RWD n'est pas un mauvais choix. Par contre pour les autres catégories de terminaux mobiles, ce n'est pas le choix idéal, il vaut mieux s'orienter vers une détection côté serveur avec du « multiserving ».

Comment améliorer la rapidité et la performance de son site web mobile ?

---- Miser sur la simplicité

---- Aller à l'essentiel et savoir prioriser les fonctionnalités

---- Faire du web sémantique en utilisant les balises sémantiques du HTML5

---- Ne pas faire trop d'Ajax

---- Eviter de faire du JSF pour des applications web mobiles. Le support HTML5 avec JSF n'est pas de très bonne qualité. JSF est à la base fait pour du web classique avec des composants qui ne sont pas forcément pensés ni optimisés pour le mobile.

---- Séparer les codes JavaScript des pages HTML5 en externalisant le code JavaScript et le CSS. Certains serveurs CDN mettent à disposition la plupart des librairies javascript qu'on utilise. Ce qui facilite la mise en cache des javascript et CSS par le navigateur. On peut citer par exemple le site cdnjs.

---- Compression des images, CSS sprite, Image map. Il existe des services qui permettent de redimensionner et de compresser à la volée des images en ligne.

---- Compression du JavaScript en utilisant des librairies spécifiques de compression (minify)

---- Compression de l'application (GZIP compression)

---- Placer les CSS en haut et les scripts JS en fin de fichier.

---- Utilisation des CDN (Content Delivery Network) pour mettre vos sites en cache et les rendre accessibles rapidement à tous. On peut citer Akamai Technologies, Limelight Network et bien d'autres.

---- Cacher les appels Ajax si possible

Avantages et inconvénients de JQuery Mobile

jQuery Mobile est un framework léger, optimisé pour le les terminaux mobiles. Il est basé sur le framework populaire jQuery. JQuery mobile présente quelques avantages qui permettent de :

---- Passer moins de temps pour le design et l'implémentation des pages web mobiles

---- Rendre plus disponible des éléments de layout, de widget et de contrôle qui sont déjà pré-désignés et prêts à l'emploi.

---- Apporter des effets de transition dans une application web mobile qui sont à la base disponibles sur des applications natives.

---- Garantir que votre application web mobile pourra s'adapter sur un nombre important de terminaux mobiles

Il existe néanmoins quelques inconvénients, à savoir :

---- Utiliser certains éléments pré-désignés de jQuery Mobile dans un site web mobile existant n'est pas toujours facile et peut dans certains cas se révéler impossible.

---- L'utilisation du jQuery mobile influence l'apparence et le comportement du site web mobile écrit au préalable en HTML ou HTML 5.

---- Prendre du temps pour apprendre et comprendre les frameworks CSS et JavaScript d'autrui plutôt que d'écrire seul ses propres scripts.

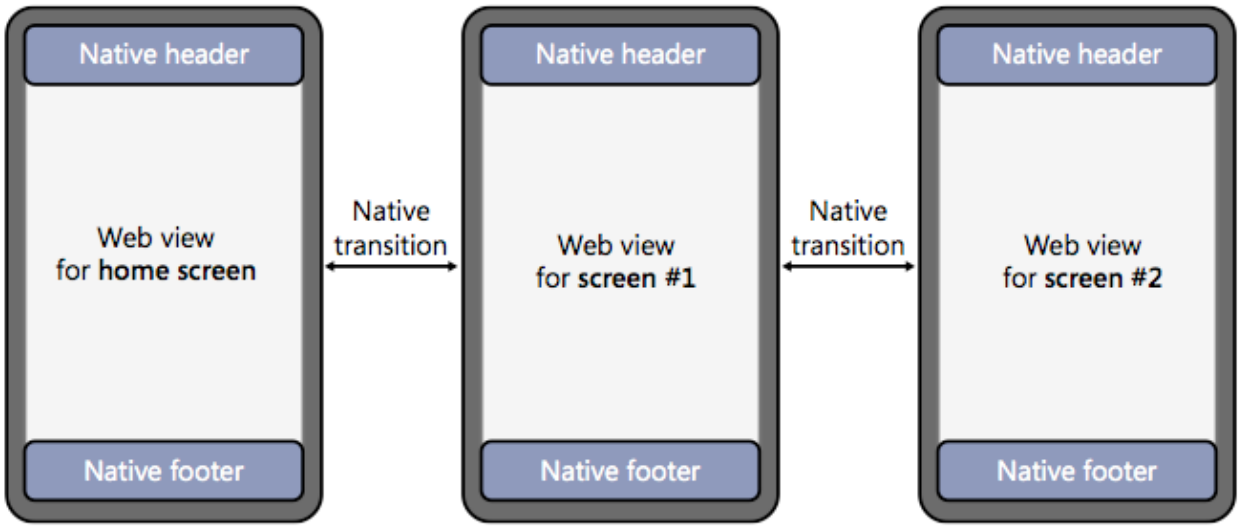
---- L'usage et l'application du jQuery Mobile à plusieurs sites web mobiles peut les rendre très semblables et similaires.

---- L'utilisation du jQuery Mobile dans une logique Responsive Web Design (RWD) est compliqué pour ne pas dire impossible. En d'autres termes, on utilisera le jQuery Mobile dans le cas d'un site web mobile dédié où l'on sait au préalable comment rediriger les utilisateurs sur la version mobile du site.

6/ L'ENVIRONNEMENT TECHNIQUE D'UNE APPLICATION HYBRIDE

Il existe aujourd'hui plusieurs solutions pour faire des applications hybrides.
On peut noter des frameworks comme Titanium Mobile, PhoneGap, Flash Builder, Oracle ADF Mobile, IBM Mobile Foundation.... La mauvaise nouvelle c'est qu'il n'y a aucune solution qui pourra répondre entièrement à vos besoins sur des projets de grandes envergures. Néanmoins, PhoneGap demeure le framework le plus utilisé comme solution de développement rapide des applications hybrides.
Avec PhoneGap, vous pourrez rapidement écrire une application mobile en utilisant des technologies familières telles que JavaScript et CSS. Nous n'allons pas rentrer dans ce paragraphe dans le développement avec PhoneGap.

Il n'est pas obligatoire d'utiliser un framework particulier pour écrire une application hybride. Les APIs et les composants (plus précisément les « WebView ») permettant de facilement intercepter les liens et donc de gérer les transitions entre les vues pour avoir une expérience utilisateur proche sinon identique d'une application mobile native. Il est d'ailleurs conseillé d'aller dans ce sens car ces frameworks hybrides peuvent être sources de lenteur et de manque de fluidité dans les transitions lorsque les ressources à servir deviennent importantes.
On parlera donc d'une coquille applicative comme le montre le schéma suivant :



// LA DISTRIBUTION DES APPLICATIONS

La distribution des applications mobiles dépend du choix de développement et des plateformes mobiles adoptées.

1/ DANS LE CAS D'APPLICATIONS NATIVES OU HYBRIDES

Stratégie de distribution B2C et B2B

Toutes les plateformes mobiles permettent aux développeurs et aux entreprises de mettre leur application à disposition des utilisateurs via des boutiques (App Store, Google Play, Windows Phone Marketplace).

Test, distribution et déploiement iOS

Les applications natives iOS sont packagées en un fichier « bundle » et installées sur les terminaux mobiles depuis iTunes.

---- Pour tester l'application sur un vrai terminal mobile connecté à Xcode - s'enregistrer comme développeur Apple, générer un certificat de développement, enregistrer le terminal à partir de son User Device IDentifier (UDID), créer une application depuis le portail d'approvisionnement et générer le profil d'approvisionnement.

---- Pour tester l'application en mode « ad hoc » - il suffit de générer le fichier bundle .ipa qui sera envoyé à chaque testeur qui, à partir de iTunes pourra installer l'application sur son téléphone. Il faut au préalable générer un certificat de distribution « ad hoc », enregistrer les terminaux des testeurs et générer un profil d'approvisionnement « ad hoc ». Le profil d'approvisionnement fait le lien entre l'application, les terminaux mobiles et le certificat. Durant cette phase de test « ad hoc », il est conseillé de mettre en place dans l'application un système de bug report permettant de remonter en temps quasi réel les différents bugs rencontrés par les testeurs. On peut se servir par exemple du service « BugSense » pour atteindre cet objectif.

---- Pour faire des tests Over The Air (OTA) - L'ipa généré peut être rendu disponible sur internet via un site de service de test comme « TestFlightApp » depuis lequel tout testeur enregistré peut installer l'application et faire des tests. Dans le cadre d'une licence entreprise, cette procédure peut aussi se mettre en place en interne. Il suffit tout simplement de maintenir un site web comportant un lien qui pointe sur une url avec le protocole « itms-services » et un fichier manifeste contenant les informations sur l'ipa (url, display-name, title,....). Apple autorise au maximum 100 distributions « ad hoc » par application même pour un programme Entreprise.

---- Soumettre son application via iTunes Connect à Apple pour validation. Le temps de réponse est en moyenne de deux semaines. Une fois l'application validée par Apple, elle est rendue disponible sur l' AppStore à tous les utilisateurs. Avant de faire cette soumission, il faut s'assurer d'obtenir un profil d'approvisionnement pour distribution valide puis compiler et packager l'application avec ce profil.

---- Distribution interne d'une application – Le principe de distribution est le même, en tout cas au niveau des tests. Apple permet aux entreprises ayant souscrit à un programme de Développeur Entreprise de distribuer leur application à un nombre illimité d'utilisateurs sans passer par App Store. Pour une distribution interne (« in-house ») il vous faut donc l'ipa de l'application, le fichier manifest, et le profil d'approvisionnement de distribution entreprise (Enterprise distribution provisioning profile). Cinq méthodes de déploiement d'une application iOS en entreprise sont disponibles :

1 Distribution et installation via iTunes
2 Avoir un administrateur système qui installera l'application sur chaque terminal avec l'outil « iPhone Configuration Utility »
3 Mettre en place un serveur sécurisé à partir duquel les utilisateurs peuvent installer l'application via internet
4 Se servir d'un MDM (Mobile Device Management) pour gérer les applications.
5 Dans le cadre d'une télédistribution, on peut installer l'application sur l'ordinateur des utilisateurs et leur demander de synchroniser leur terminal mobile, via une connectique adaptée.

La méthode 3 est la plus utilisée :

- Avoir un serveur web pouvant authentifier les utilisateurs et accessible via https
- A partir de Xcode créer une version archivée de l'application (.ipa + fichier manifest). Placer ces deux fichiers dans des répertoires cachés sur votre serveur.
- Le fichier manifest est un fichier XML au format .plist. Il est utilisé par le terminal mobile pour trouver, télécharger et installer les applications depuis votre serveur. Ce fichier est généralement créé depuis Xcode et contient les attributs décrits dans le tableau ci-dessous :

Item	Description
URL	The fully qualified HTTP or HTTPS URL of the app (.ipa) file.
display-image	A 57 x 57-pixel PNG image that's displayed during download and installation. Specify the image's fully qualified URL.
full-size-image	A 512 x 512-pixel PNG image that represents the app in iTunes.
bundle-identifier	Your app's bundle identifier, exactly as specified in your Xcode project.
bundle-version	Your app's bundle version, as specified in your Xcode project.
title	The name of the app, which is displayed during download and installation.

- Écrire et maintenir un petit site web (très souvent d'une page) qui contient un lien vers le fichier manifest :

Install App

- Configurer votre serveur web en définissant les MIME types suivants : application/octet-stream ipa et text/xml plist
- Envoyer l'url du mini site web à vos utilisateurs via email, sms ou bien via l'usage d'un QRCode.

---- Mise à jour - Les applications distribuées en interne ne sont pas automatiquement mises à jour. Il revient au développeur de mettre en place un mécanisme de notification de mises à jour. Le plus simple est de forcer l'application, à chaque lancement, à scruter le serveur pour une mise à jour disponible. Si c'est le cas, l'application peut afficher une alerte à partir de laquelle l'utilisateur sera redirigé sur le mini site web de mises à jour. Attention, pour que l'application soit mise à jour et conserve des données stockées sur le terminal, il faut s'assurer que la nouvelle version de l'application utilise le même « bundle-identifier » et surtout demander aux utilisateurs de ne pas manuellement supprimer l'ancienne version de l'application avant d'installer les nouveautés.

Pour ceux qui sont intéressés par des stress tests, des tests fonctionnels, ou de l'analyse statique du code vous avez respectivement l'utilitaire « MonkeyTalk », KIF (Keep It Functional), et OCLint.

Test & distribution sous android

---- Les applications mobiles natives sous android sont packagées en un fichier .apk qui est installé sur le terminal de l'utilisateur. Un peu comme pour une distribution iOS, il faut évidement bien tester son application.

---- Android met à disposition plusieurs outils de tests, de débogages et de profilages des applications. On peut citer adb, ddms, traceview, dmtracedump, systrace...
On peut aussi utiliser des services comme « BugSense » pour mieux gérer les bugs lors des tests « ad hoc »

---- Contrairement à l'iOS, il n'y a pas de restriction au niveau du système pour tester les applications sur un terminal. Seul l'utilisateur peut décider d'utiliser son téléphone pour des tests lors des phases de développement. Il suffit d'aller dans les paramètres du téléphone pour activer cette option. Pour faire des tests en mode « ad hoc », il suffit de signer l'application avec un certificat auto signé et déposer l'apk sur un serveur web pour téléchargement et installation. On peut donc installer l'application sur un nombre illimité de terminaux mobiles pour autant qu'ils soient compatibles avec les versions des OS Android ciblées par l'application.
On peut utiliser des services comme « apkudo » pour tester nos applications sur plusieurs types de terminaux Android en ligne. Pour éviter d'invertir énormément dans l'achat de plusieurs terminaux mobiles pour faire des tests en interne.

---- Distribution de l'application pour le grand public peut se faire via Google Play. Ceci n'est pas une obligation, surtout pour une entreprise qui désire faire un déploiement interne de ses applications. Il suffit tout simplement de suivre la même procédure de distribution « ad hoc ». Il faut aussi maintenir un mini site web à partir duquel les utilisateurs peuvent amorcer le processus d'installation de l'application. Il faut simplement ne pas oublier de rajouter le MIME type application/vnd.android.package-archive apk et surtout demander aux utilisateurs d'activer, dans les paramètres de leur téléphone, l'option « Installer depuis une source inconnue »

Pour ceux qui sont intéressés par des stress tests, des tests fonctionnels, ou de l'analyse statique du code vous avez respectivement les utilitaires « Monkey », Robotium, et Lint.

Test, soumission et distribution Windows Phone

---- Test des applications sur émulateur sans contrainte particulière si ce n'est avoir les outils de développement listés dans les paragraphes précédents. Pour faire des tests sur un vrai téléphone, il faut s'enregistrer en tant que développeur Windows Phone et payer pour.

---- Enregistrer les terminaux que l'on prévoit pour les tests : d'une manière générale, Microsoft permet à tous les terminaux sous Windows Phone de pouvoir installer des applications depuis sa boutique. Par contre seuls les terminaux enregistrés pour des tests peuvent installer du code lors de la phase de développement. Une fois le terminal de test enregistré, il est débloqué et l'on peut y installer n'importe quelle application. L'enregistrement passe par un outil sous Windows (Zune Client Software). Le développeur doit donc connecter les téléphones à son ordinateur pour amorcer le processus d'enregistrement. Cette technique manque un peu de souplesse comparée à Apple qui permet d'enregistrer un terminal seulement à partir du UDID. Un développeur est limité à l'enregistrement de trois téléphones et chaque téléphone ne peut se voir installer plus de trois applications ne venant pas du marketplace.

---- Passer par le programme Beta du marketplace - cette procédure consiste à soumettre une application sur marketplace uniquement pour des beta testeurs. Cette application en beta test ne sera accessible et installable que via le web que par des testeurs dont l'identifiant « Windows Live » correspond à ceux définis dans le package de l'application. Les conditions d'utilisation du beta programme sont nombreuses.

Déjà en tant que développeur, il est impératif de rajouter la liste des identifiants « Windows Live » des utilisateurs qui veulent tester l'application. Tous les utilisateurs dont l'identifiant figure dans le package de l'application recevront automatiquement un mail venant de marketplace avec un lien vers l'exécutable de l'application en Beta test. Windows limite à 100 beta testeurs et toute application soumise en Beta test expirera après 90 jours. A tout moment le développeur peut aussi déclencher l'expiration de l'application. Dépasser le délai des 90 jours, le beta testeur ne pourra plus démarrer l'application et il lui sera demandé de la désinstaller. Pour finir, une application en Beta test ne peut être mise à jour; il faut terminer le processus de test actuel et en démarrer un autre.

---- La soumission sur le marketplace en mode Beta n'est pas disponible dans l'immédiat (cela peut prendre des heures) . Il est donc conseillé, pour ceux qui veulent faire des mises à jour fréquentes en Beta test, d'associer le terminal d'un utilisateur avec un compte « App Hub » afin de procéder à un chargement direct de l'application sur le terminal mobile.

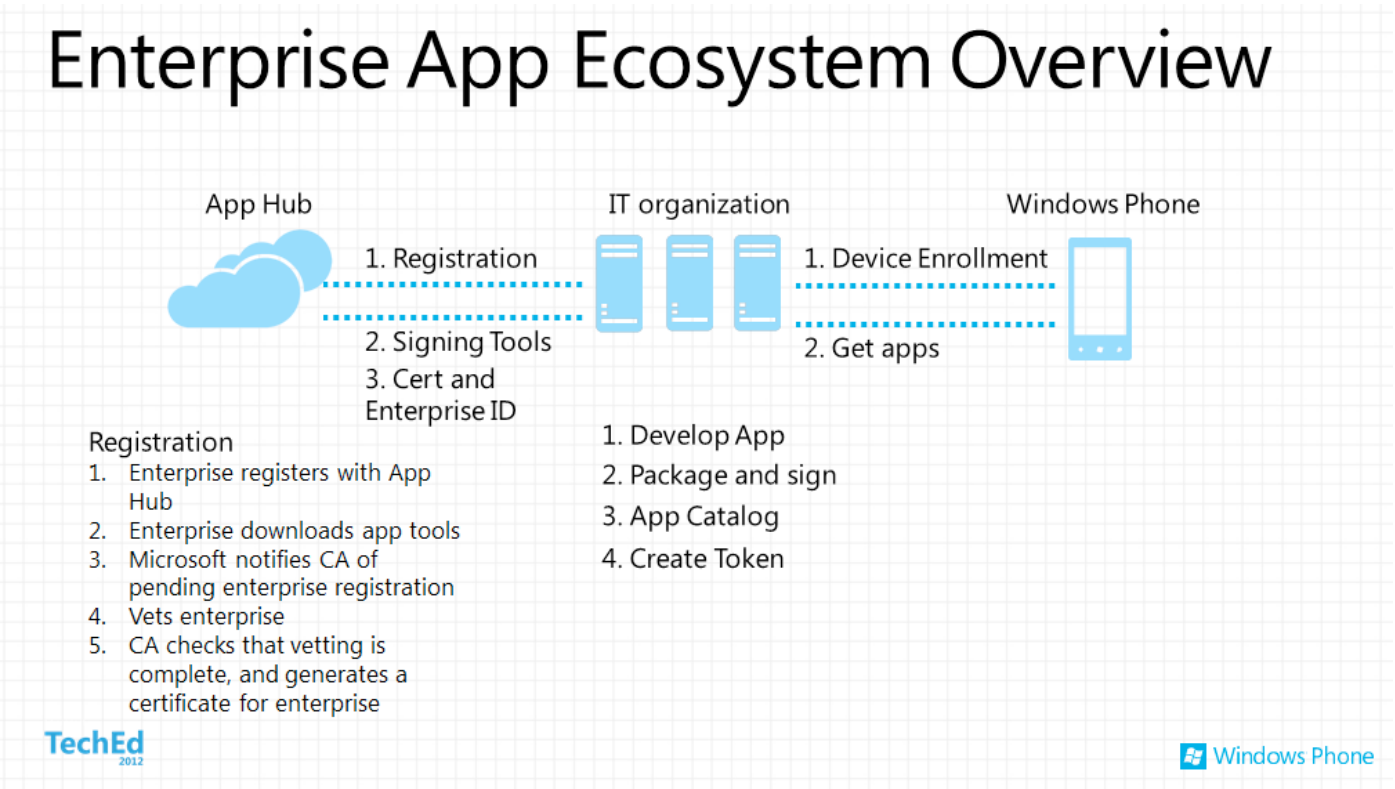
---- La seule manière de distribuer une application Windows Phone est de passer par le marketplace, qui ressemble beaucoup aux autres plateforme (Apple et Android) même si Google permet d'avoir d'autres boutiques Android. On peut soumettre un nombre illimité d'applications sur le marketplace de Windows, avec pas plus de 100 applications gratuites par développeur. Le partage de revenu est de 30% pour Microsoft et 70% pour le développeur. C'est le même type de partage pour Apple et Google Play.
Le processus de soumission est très semblable à celui des autres plateformes : uploader le package final (.xap) et fournir les metadata. Une phase de validation de l'application ainsi que des ses mises à jour sont faites, un peu comme sur Apple. Cela peut prendre quelques jours.

Pour finir la plateforme fournit une API permettant à une application de faire, par exemple, des recherches directement depuis le marketplace. Une fonctionnalité intéressante que les autres plateformes n'offrent pas de-facto est le mode « try before you buy » d'une application. Ainsi un utilisateur peut tester une application avant de l'acheter. Il revient au développeur de mettre en place cette fonctionnalité dans l'application via des APIs spécifiques mais aussi de choisir les bonnes options lors de la phase de soumission.

---- Pour finir il est possible de rajouter de la pub via des composants natifs un peu comme sous iOS avec « iAd » et sous Android via « AdMob »

Au départ, Microsoft n'avait pas prévu un programme Entreprise pour Windows Phone. La version Windows Phone 8 permet aux entreprises de déployer leurs applications sans passer par le marketplace comme expliqué ici :

http://channel9.msdn.com/events/TechEd/Europe/2012/WPH205



Avant de clore ce paragraphe sur la distribution et le déploiement des applications mobiles natives ou hybrides, nous pouvons parler des solutions de MDM (Mobile Device Management) qui peuvent aider les administrateurs au sein d'une entreprise à mieux gérer un parc mobile. Android met à disposition l'API DevicePolicyManagement qui permet au développeur de contrôler d'une manière ou d'une autre certains terminaux mobiles.
Le tableau suivant regroupe les services et outils à disposition :

Produit	Editeur	Fonctions	Systèmes mobiles supportés
PushManager	Ibelem	Déploiement d'applications, inventaire, effacement à distance, gestion des politiques de sécurité.	Android, iOS
Good for Enterprise	Good Technology	Outils de travail collaboratif et navigateur maison placés dans un conteneur. MDM dédié à ces outils.	Android, iOS, Windows Phone
McAfee Enterprise Mobility Management	McAfee	Déploiement d'applications, détection des iPhones jailbreakés, authentification, accès sécurisé aux données.	Android, iOS, Windows Phone, Nokia
Landesk Mobility Manager	LanDesk	Déploiement d'applications, inventaire, gestion des politiques de sécurité…	Android, iOS, Windows Phone
Virtual Smartphone Platform	MobileIron	Déploiement d'applications, visibilité et contrôle en temps réel sur les contenus, l'activité et les applications. Gestion des politiques de sécurité…	Android, iOS, Windows Phone, BlackBerry, Nokia
Afaria	Sybase/SAP	Déploiement d'applications, inventaire, détection des iPhones jailbreakés, gestion des politiques de sécurité…	Android, iOS, Windows Phone
Zenprise MobileManager	Zenprise	Déploiement d'applications, prise de contrôle, gestion des politiques de sécurité…	Android, iOS, Windows Phone

2/ LE CAS DES SITES MOBILES

Le déploiement des sites web mobiles n'est assujetti à aucune contrainte liée aux plateformes mobiles. N'étant pas soumis à une phase de validation, le développeur maîtrise la phase de déploiement. Tout dépend des technologies serveurs adoptées par l'entreprise pour développer sa solution web mobile (php, .net, java).

Le déploiement d'un site web mobile n'a aucune spécificité particulière comparé à celui d'un site web classique. C'est tout simplement une application web en plus.

Très souvent l'utilisateur ne connaissant pas l'url du site web mobile, il revient à l'équipe technique de bien configurer le serveur (Apache, Nginx,…) à l'aide de certaines directives afin que tous les utilisateurs mobiles venant sur le site web classique soient redirigés vers le site web mobile. Vous pouvez utiliser les scripts disponibles sur **ce site** pour configurer ce routage :

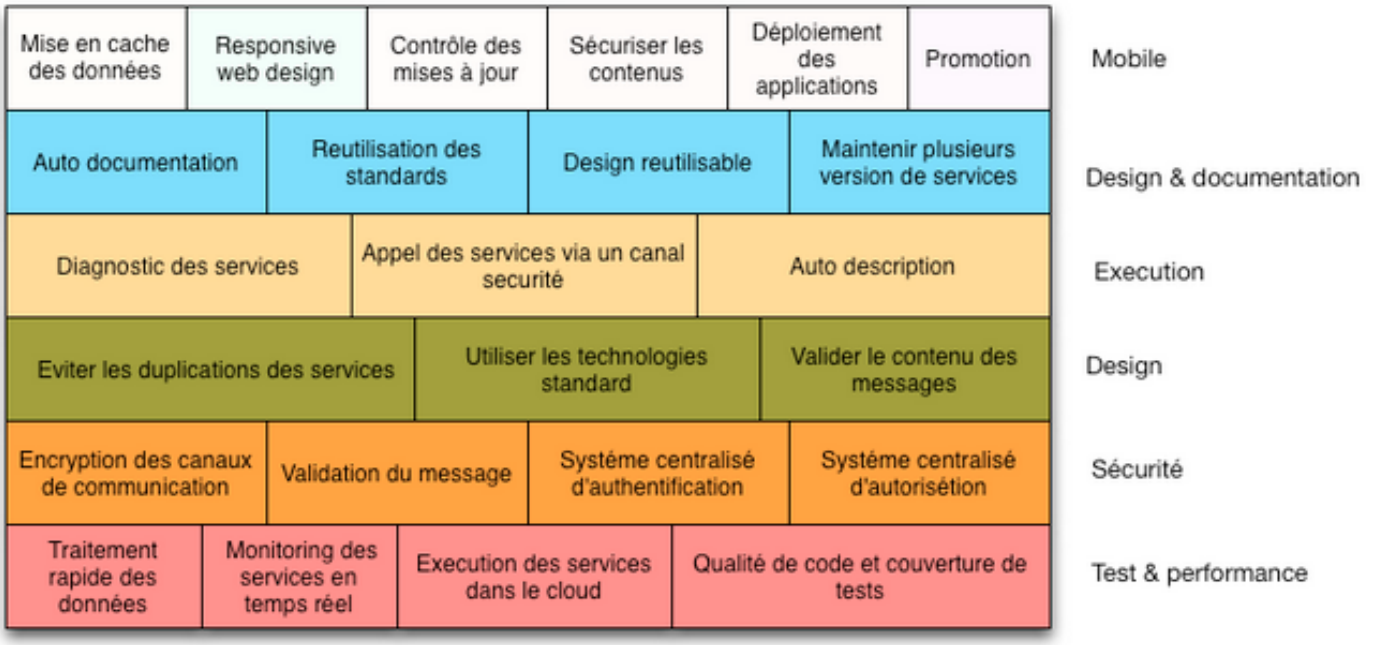
N'hésitez pas à appliquer les règles d'amélioration de performances listées sur le site de **yahoo**. Vous pouvez aussi utiliser les outils **YSlow** et **Blaze** pour tester vos applications web mobiles.

// L'INTEGRATION DES APPLICATIONS AVEC LE SYSTEME D'INFORMATION

Les systèmes d'information ont été longtemps pensés pour les applications web. Très souvent, il s'agit de systèmes et d'infrastructures rigides et peu flexibles pour intégrer facilement de nouvelles applications mobiles basées sur les services existants.

Dans ce chapitre, nous allons essayer de voir quelles sont les règles et bonnes pratiques à adopter pour intégrer un service ou une application mobile dans le Système d'Information.

Le schéma suivant montre quelles sont les questions à poser lors d'une mise en place du service mobile au sein du SI.



1/ LES CONTRAINTES DE SÉCURITÉ

Les problématiques de sécurité dans une application mobile sont liées entre autres au stockage des données sur le terminal, à l'échange de l'information entre le terminal et le serveur avec lequel elle communique.
Lorsqu'on écrit une application mobile qui doit consommer des services web ou web services au sein du SI, il est important de mettre en place les solutions suivantes pour garantir la sécurité de l'application:

---- Accès sécurisé via HTTPS – Il est fortement conseillé lors de la mise en service que le canal de transmission des informations et des données entre le mobile et le serveur soit entièrement crypté via l'usage d'un certificat délivré par une autorité de certificat valide.

---- Privilégier la méthode POST pour passer des informations, dans le corps de la requête, qui ne peuvent être interceptées lorsque le canal est encrypté.

---- Valider l'intégrité des messages - On peut se servir du « Hash-based message authentication code » (HMAC). Le principe est de permettre au serveur de vérifier qu'une requête n'a pas été altérée en comparant certains paramètres envoyés par le client. Il est nécessaire dans ce cas d'assigner un secret à chaque utilisateur devant consommer votre service. Ce secret devrait être envoyé séparément via mail ou autre moyen sécurisé au développeur. Il servira à signer et générer le HMAC.

---- Authentifier les consommateurs des services web – Tout consommateur de services REST/SOAP doit s'authentifier en fournissant par exemple le triplet (WSCallerId, Timestamp, Hash).

- wsCallerId - l'identifiant de l'appelant du Web service
- timestamp – le timestamp au moment de l'envoi de la requête
- hash- la signature de la requête sous forme Hexa (SHA1(wsCallerSecret + timestamp))
- Une clef secrète (wsCallerSecret) est en plus partagée entre le serveur et le service.

---- Maintenir un « token » de session auto porteur – ce token sera une donnée encryptée par le serveur et qui doit contenir un certain nombre d'informations qui ne sont réutilisables que par le serveur.

---- Sauvegarde sécurisée sur terminaux mobiles – il faut éviter de sauvegarder des données très sensibles sur le téléphone. Il existe des APIs d'encryption autour des « keychain ». Sous iOS, vous avez par exemple « KeychainItemWrapper ».

---- Valider le certificat du serveur applicatif depuis l'application mobile – ceci afin d'être certain que l'application mobile communique bel et bien avec le serveur avec qui l'application devra communiquer. Les plateformes mobiles fournissent des APIs qui permettent de faire ce genre de validation à partir des informations (empreintes) du certificat serveur.

---- Utiliser le protocole OAuth – lorsqu'on veut exposer des données utilisateurs du système vers d'autres systèmes ou services, il est conseillé de passer par ce protocole. En effet, c'est un protocole qui est à la base fait pour l'univers du Web mais qui peut aussi s'utiliser facilement lorsqu'il s'agit des applications mobiles natives tout simplement parce que la plupart des plateformes mobiles à disposition fournissent ce composant génial qui est le « WebView » et qui permet d'avoir une sorte de navigateur au sein même de l'application sans devoir passer par le celui du téléphone.

---- Limiter la répétition des requêtes (services web) - les « replay attacks» sont une forme d'attaque réseau dans laquelle la transmission d'une donnée valide peut être frauduleusement répétée. Dans le cadre des services web, il est conseillé, en tout cas durant la mise en production du service, de rendre difficile cette forme d'attaque. On peut par exemple définir un seuil (délai) à partir duquel une même requête ne doit être traitée par le serveur en se basant sur le timestamp (détaillé plus haut) fourni lors de la requête et l'instant où la requête est reçue par le serveur.

2/ LES SERVICES WEB

La couche de services web ou de web services permet d'exposer les services métiers aux requêtes qui ne s'exécutent pas dans le même domaine applicatif que ces services métiers. On peut par exemple donner la possibilité aux terminaux mobiles de communiquer avec les services métiers via cette couche. Elle définit un ensemble de messages et formats d'échange pouvant permettre la transmission d'informations entre deux entités de manière fiable.

Les choix d'implémentation

Parce que plusieurs applications web ont par le passé utilisé des services SOAP, certains pensent que les mêmes services SOAP sont le bon choix pour les applications mobiles – ce qui n'est pas forcément le cas et les analyses de performance l'ont bien démontré. Les experts mobile d'Ippon Technologies pensent que les services REST avec des échanges de données au format JSON demeurent le bon choix pour les clients mobiles basés sur les technologies iOS, Android et même pour du web mobile. Il est vrai que certains portent des réserves quant à l'usage intensif des services web (REST/SOAP) pour certaines applications web. Ceci peut se comprendre car on peut solliciter des ressources importantes au niveau du réseau. Néanmoins si vous avez une architecture modulaire et orientée service, vous pouvez réduire de manière considérable le nombre d'appels au web services avec une analyse en amont des services que vous voulez offrir. L'usage des mécanismes de cache peut être d'une grande utilité.

Des web services SOAP et des services web en REST sont disponibles. Pour l'accès aux services via les téléphones mobiles, il est plutôt conseillé de privilégier l'utilisation du REST/JSON pour les raisons suivantes :

---- La productivité du développeur – consommer un service REST et traiter du message au format JSON est plus rapide à mettre en œuvre.

---- La performance – les études l'ont montré, le REST est beaucoup plus performant quand il s'agit des applications web mobiles et n'introduit pas autant d' « overhead » que le SOAP.

---- L'utilisation efficace de la bande passante

---- Contrairement à ce que l'on pense, on peut rendre un service REST sécurisé. On peut facilement sécuriser la couche transport via du HTTPS mais aussi définir un certain nombre de mécanismes bien connus du protocole http pour sécuriser l'authentification et l'accès aux ressources.

---- La robustesse

---- La simplicité – il a été pensé pour des clients légers, ce qui implique une simplicité et une flexibilité dans l'implémentation

---- La possibilité de faire du cache et la scalabilité – l'accès aux ressources étant basée sur le principe de l'URL, il est très facile de mettre en place des mécanismes de cache sur certaines ressources et d'utiliser les CDN (Content Delivery Network) pour du cache au niveau des infrastructures réseau.

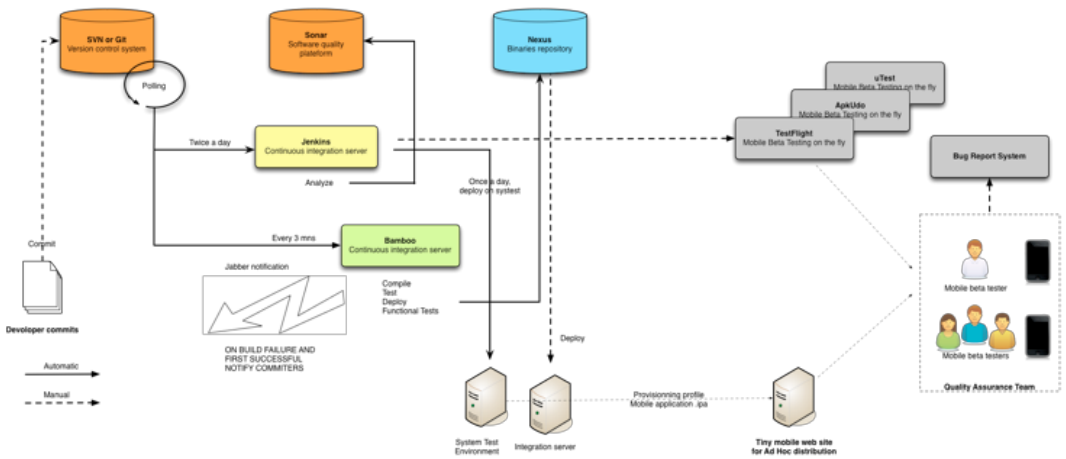
---- SOAP n'est pas plus sécurisé que le REST. Quand les services SOAP retournent uniquement du flux XML, les services REST fournissent une certaine flexibilité au regard du type de message retourné. Le format de base du REST demeure le JSON même si on peut toujours utiliser le XML. Les données au format JSON sont très souvent moins verbeuses et facilement manipulables par la plupart des terminaux mobiles à partir des API mises à disposition par les différentes plateformes mobiles.

// L'INDUSTRIALISATION DES APPLICATIONS MOBILES

Aujourd'hui, il est complètement possible d'industrialiser ses développements mobiles. Concrètement, vous pouvez opter pour des systèmes d'intégration continue que vous avez déjà presque l'habitude d'utiliser sur vos projets web. L'avantage d'un tel système c'est de pouvoir tout automatiser ou presque et de rendre disponible les exécutables mobiles aussi rapidement que possible à l'équipe de tests.

Ainsi, un bon système d'intégration continue dans l'univers mobile peut ou doit aider les équipes techniques à automatiser la compilation, les tests unitaires, les tests fonctionnels, les tests sur les terminaux ou simulateurs mobiles, la couverture de code et le déploiement.

Le bénéfice sera d'avoir des applications mobiles iPhone ou Android sans erreur, sans crash, d'éviter des tests manuels qui déjà peuvent mobiliser beaucoup de ressources et surtout ne sont pas toujours efficaces quand ce n'est pas bien orchestré.
Attention, il n'est pas question d'abandonner les beta testeurs. Au contraire, ils doivent jouer un rôle complémentaire à celui des tests sur simulateurs ou terminaux réalisés par le système d'intégration continue. Enfin, avec un système d'intégration continue, on peut mieux gérer les tests de non régression sur les applications mobiles.



Très souvent, la mise en place d'un système d'intégration mobile est beaucoup moins complexe car ce sont des petits projets avec une petite équipe (rassurez vous ça ne va plus être le cas dans les années à venir).

Plusieurs additifs permettent d'étendre les fonctionnalités du système d'intégration continue vers une gestion des projets mobiles.

Dans l'univers informatique, le système d'intégration « Jenkins » est devenu une référence et aujourd'hui il permet de gérer et d'automatiser entièrement un projet mobile depuis la phase de compilation jusqu'à la distribution aux beta testeurs. Jenkins est bien pour les projets mobiles sous iOS, Android et les applications web mobiles.
Il offre son écosystème de « plugins mobiles » qu'on peut utiliser pour mieux faire configurer les scripts d'automatisation. Par exemple, certains additifs permettent de configurer plusieurs types d'émulateurs pour tester les projets Android. D'autres additifs, permettent d'envoyer automatiquement l'exécutable final (.apk) du projet aux beta testeurs.

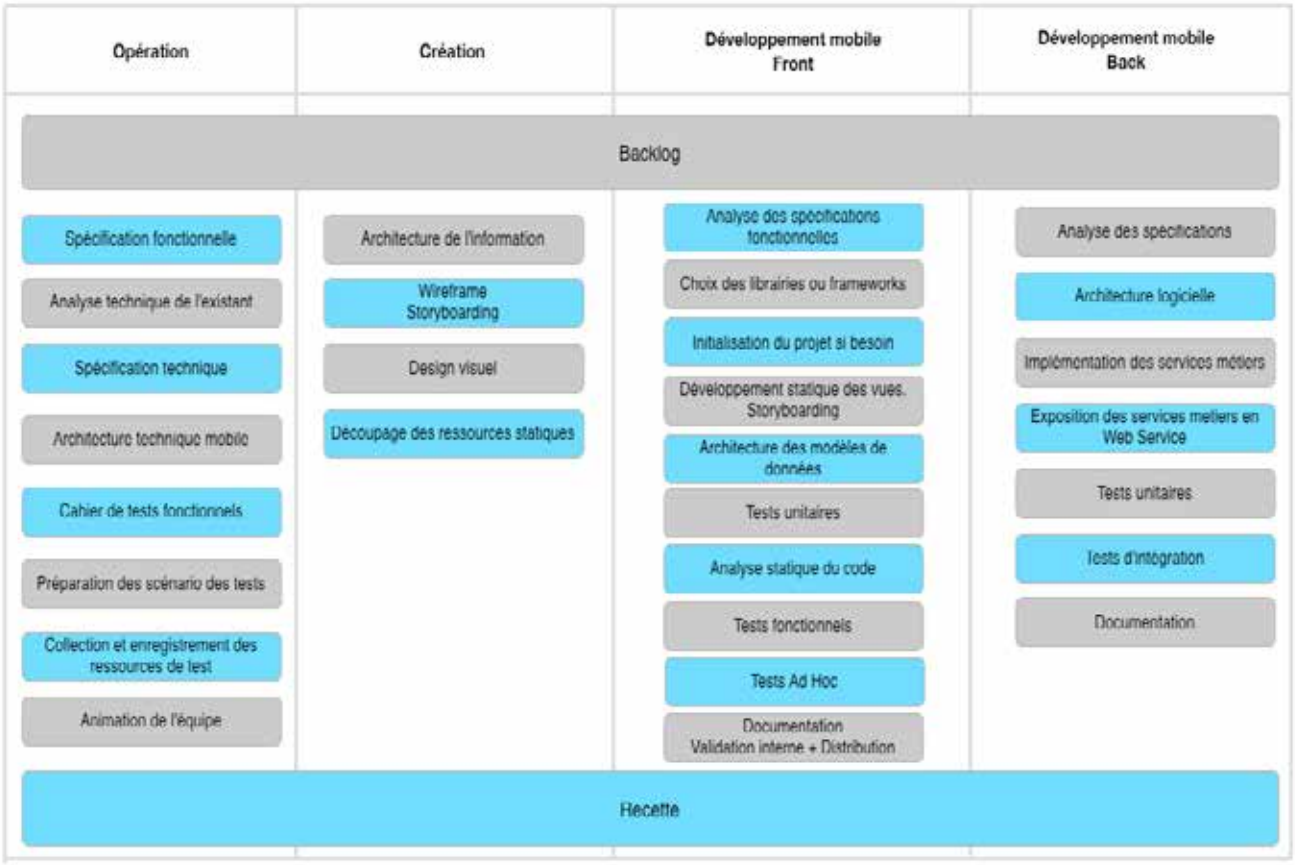
En ce qui concerne les projets iOS, on peut par exemple définir plusieurs cibles (targets) pour les projets où chaque cible a une configuration particulière (pourquoi pas une cible qui pointe sur la pré-prod et l'autre sur la prod.). A partir de ces cibles, Jenkins peut vous aider à générer les .ipa associés à chaque cible et les déployer automatiquement sur le site « TestFlight » pour les Beta Testeurs.

// EQUIPE ET ORGANISATION

L'écosystème mobile est en perpétuel mouvement, un projet mobile a besoin d'une équipe dédiée, flexible qui connaît techniquement le fonctionnement des différentes plateformes mobiles. Le développement mobile contrairement à ce qu'on peut penser ne se limite plus à ce qui devrait être normalement une application mobile, c'est-à-dire une application permettant de faire une tâche et une seule.
Aujourd'hui, développer une application mobile demande un spectre assez large de connaissances.

Ippon Technologies encourage les porteurs de projets mobiles à orienter les équipes dans une organisation Agile qui comprend la notion de changement de besoin. Pour cela, il faut une équipe bien structurée, pragmatique qui ne doit jamais perdre de vue que l'application mobile qu'elle développe aujourd'hui aura au maximum six mois à vivre. Au delà des ces six mois, il est important, voir urgent de mettre à jour l'application sur les différentes boutiques avec de nouvelles fonctionnalités.

D'une manière générale, une équipe mobile doit comprendre au moins un ergonome, un designer, un développeur mobile, un développeur côté serveur, un chef de projet, un testeur. Chaque acteur aura à jouer des rôles spécifiques et fournir des livrables qui devront servir aux autres membres de l'équipe.



// CONTACT

Vous pouvez retrouver toutes nos coordonnées sur www.ippon-mobile.fr

Nous joindre par mail à l'adresse accueil@ippon.fr,

Ou contacter une de nos 4 agences directement par téléphone :

Agence Paris : + 33 (0)1 46 12 48 48
Agence Nantes : +33 (0)2 40 48 28 06
Agence Bordeaux : +33 (0)6 19 65 35 06
Agence Marseille : +33 (0)4 86 68 58 48

Twitter : @ippontech
Facebook : https://www.facebook.com/ippon.technologies
Viadeo : www.viadeo.com/fr/company/ippon-technologies

L'ÉQUIPE

Références : **Architecting Mobile Solutions for the Enterprise**.
Jenkins peut vous aider à générer les .ipa associés à chaque cible et les déployer automatiquement sur le site «TestFlight» pour les Beta testeurs.


ippon
TECHNOLOGIES