

ÉCOSYSTÈME DES ARCHITECTURES MOBILES

WEB, NATIVE,
CROSS-COMPILÉE OU HYBRIDE

Olivier CATTÀ



LIVRE BLANC

Mars 2017

IPPON

Digital . Technologies . Hosting

TABLE DES MATIÈRES

1	L'auteur	2
2	A propos d'Ippon Technologies	3
2.1.	Nos solutions	4
2.2.	Nous contacter	5
3	Licence	6
4	A qui s'adresse ce livre blanc ?	8
5	Rappel sur les différents environnements	9
5.1.	Le développement natif	9
5.2.	Le développement Web Mobile	12
5.3.	Le développement hybride	17
5.4.	Le développement cross-compilé	19
6	Les environnements techniques de développement mobile	21
6.1.	L'environnement technique Android natif	21
6.2.	L'environnement technique iOS natif	28
6.3.	L'environnement technique Windows natif	34
6.4.	L'environnement technique des sites Web mobiles	38
7	Le back-end	46
7.1.	Les protocoles de communication	46
7.2.	Les formats des données	47
7.3.	Problématiques fréquentes et pistes de solution	49
7.4.	Sécurité applicative	53
8	Industrialisation des développements	55
9	Statut légal d'éditeur	57
10	Perspectives à court terme	58
10.1.	Quelques faits pour commencer	58
10.2.	A partir de ces quelques faits	59

1 — L'AUTEUR



Olivier Catta

Directeur d'agence et responsable du centre de compétences Mobilité chez Ippon Technologies depuis 2013.

Il travaille sur des missions d'architecture et d'audit. Fort de 17 ans d'expérience, dont 15 en Java, Olivier est spécialisé dans les architectures distribuées et les applications mobiles.

Découvrez son livre blanc sur blog.ippon.fr

Comment livrer une application stratégique avec un Time To Market réduit?

www.ippon.fr/livreblanc/comment-livrer-une-application-strategique-avec-un-time-market-reduit/

2 — A PROPOS D'IPPON TECHNOLOGIES

Ippon est un cabinet de conseil en technologies, créé en 2003 par Stéphane Nomis, sportif de Haut-Niveau et un polytechnicien, avec pour ambition de devenir leader sur les solutions Digitales, Cloud et Big Data.

Ippon accompagne les entreprises dans le développement et la transformation de leur système d'information avec des applications performantes et des solutions robustes.

Ippon propose une offre de services à 360° pour répondre à l'ensemble des besoins en innovation technologique : Conseil, Design, Développement, Hébergement et Formation.

Nos équipes tirent le meilleur de la technologie pour transformer rapidement les idées créatives de nos clients en services à haute valeur ajoutée. Elles accompagnent à la fois les grands groupes (Axa, EDF, Société Générale, LVMH,...) et les champions français de l'industrie numérique (CDiscount, LesFurets.com, Aldebaran....) dans leurs innovations.

Nous avons réalisé, en 2016, un chiffre d'affaires de 24 M€ en croissance organique de 20%. Nous sommes aujourd'hui un groupe international riche de plus de 270 consultants répartis en France, aux USA, en Australie et au Maroc.

IPPON VOTRE PARTENAIRE
END TO END



CONSEIL
FORMATIONS



UX
DESIGN



RÉALISATION
AGILITE/DEVOPS



HÉBERGEMENT

2.1. Nos solutions

Dans le cadre de notre expertise technique, Ippon Technologies propose des prestations dans le domaine du développement de sites Web mobiles et d'applications mobiles. Notre équipe pluri-disciplinaire de concepteurs d'interfaces, de designers, d'ingénieurs Etudes et Développement est en mesure de vous accompagner de A à Z, de l'idée au déploiement sur les Stores applicatifs. Sans à-priori, nous préconisons les solutions techniques qui seront les meilleures pour répondre à votre besoin : développement natif ou hybride, application ou site Web. Notre compétence en architectures n-tiers permet aussi d'envisager des passerelles vers votre S.I., permettant de valoriser vos données ou votre savoir-faire métier sur des appareils mobiles.

UNE IDÉE, UN POC, UN PRODUIT...



idée



lean
startup



conseil
POC



product
backlog



projet par
itérations



équipe
agile



projet
run



conseil
DevOps



infogérance
cloud

Vos besoins, nos solutions

2.2. Nous contacter



PARIS
BORDEAUX
NANTES
LYON

RICHMOND, VA
WASHINGTON, DC
NEW-YORK

MELBOURNE
MARRAKECH

*www.ippon.fr/contact
contact@ippon.fr
blog.ippon.fr*

*+33 1 46 12 48 48
@ippontech*

3 — LICENCE

Ce document vous est fourni sous licence Creative Commons Attribution Share Alike. Le texte ci-dessous est un résumé (et non pas un substitut) de la licence.

› Vous êtes autorisé à :

- **Partager** : copier, distribuer et communiquer le matériel par tous moyens et sous tous formats
- **L'Offrant** ne peut retirer les autorisations concédées par la licence tant que vous appliquez les termes de cette licence.

› Attribution :

- **Vous devez créditer l'Œuvre** : intégrer un lien vers la licence et indiquer si des modifications ont été effectuées à l'Œuvre. Vous devez indiquer ces informations par tous les moyens raisonnables, sans toutefois suggérer que l'Offrant vous soutient ou soutient la façon dont vous avez utilisé son Œuvre.
- **Pas d'utilisation commerciale** : vous n'êtes pas autorisé à faire un usage commercial de cette Œuvre, tout ou partie du matériel la composant. Pas de modifications - Dans le cas où vous effectuez un remix, que vous transformez, ou créez à partir du matériel composant l'Œuvre originale, vous n'êtes pas autorisé à distribuer ou mettre à disposition l'Œuvre modifiée.
- **Pas de restrictions complémentaires** : vous n'êtes pas autorisé à appliquer des conditions légales ou des mesures techniques qui restreindraient légalement autrui à utiliser l'Œuvre dans les conditions décrites par la licence.

› Notes :

- **Vous n'êtes pas dans l'obligation de respecter la licence pour les éléments ou matériel appartenant au domaine public** ou dans le cas où l'utilisation que vous souhaitez faire est couverte par une exception.
- **Aucune garantie n'est donnée. Il se peut que la licence ne vous donne pas toutes les permissions nécessaires pour votre utilisation.** Par exemple, certains droits comme les droits moraux, le droit des données personnelles et le droit à l'image sont susceptibles de limiter votre utilisation.



4 — A QUI S'ADRESSE CE LIVRE BLANC ?

Ce livre blanc s'adresse principalement aux directeurs des systèmes d'information, aux architectes techniques et applicatifs, aux chefs de projets mobiles et aux développeurs. En somme à toute personne participant à la construction d'applications mobiles.

Il s'adresse également à toute personne curieuse et désireuse de se faire une idée du développement d'applications mobiles et de l'écosystème technique associé...

5 — RAPPEL SUR LES DIFFÉRENTS ENVIRONNEMENTS

Lors de la création d'une application destinée aux appareils mobiles (principalement smartphones et tablettes), la première question technique qui se pose est celle du choix de l'architecture de développement.

En effet, plusieurs pistes d'architecture sont en lice :

- › Le développement natif
- › Le développement d'un site Web mobile
- › Le développement hybride
- › Le développement cross-compilé

L'une ou l'autre solution sera sélectionnée de manière à répondre au mieux aux différentes contraintes portant sur le projet :

- › le public visé
- › le besoin d'intégration des fonctionnalités avancées du téléphone
- › le budget alloué
- › le besoin de performances
- › les capacités et compétences de l'équipe de développement

5.1. Le développement natif

Le développement dit natif consiste à développer une application dans l'écosystème technique offert par le système d'exploitation d'une famille d'appareils mobiles.

Il existe aujourd'hui trois principaux systèmes d'exploitation utilisés sur les appareils mobiles :

- › Android, un produit de la société Google, est utilisé sous licence par de multiples constructeurs.

- › iOS, un produit de la société Apple, uniquement utilisé sur les appareils construits par Apple.
- › Windows 10, un produit de la société Microsoft, utilisé sur les appareils Microsoft (depuis le rachat de la société Nokia) principalement.

Chacun de ces systèmes permet d'accéder, via des API, aux fonctionnalités de l'appareil mobile, qu'il s'agisse d'éléments graphiques (widgets et menus), de solutions constructeurs (paiement en ligne, gestion des notifications) ou des capteurs embarqués (appareil photo, boussole, GPS...).

- › Les avantages en termes de fluidité et d'intégration des applications natives sont évidents.
- › L'accès aux solutions logicielles des constructeurs (Google Services, iOS Developer Library, Windows SDK) donne accès à de nombreuses fonctionnalités complémentaires.
- › Les fonctionnalités matérielles des appareils mobiles sont accessibles via ces kits de développement ou SDK.
- › Les applications natives (et hybrides) peuvent être ajoutées aux catalogues applicatifs des constructeurs : les «Stores» applicatifs sur lesquels les utilisateurs peuvent aller chercher les applications

Des systèmes d'exploitation tiers ont tenté une percée : FirefoxOS, Bada, webOS... sans parvenir à s'imposer. Ils sont aujourd'hui négligeables et n'ont aucune raison d'être pris en compte.

Un illustre prédécesseur a aujourd'hui disparu des écrans radar : BlackBerry OS.

Un système isolé persiste dans une niche : Chrome OS, uniquement disponible pour des appareils Chromebooks.

Ces systèmes tiers ne seront pas évoqués, leur part de marché est trop faible.

- **Les forces du développement natif**

Les forces du développement natif sont, par ordre décroissant :

- › l'intégration de l'application au système – techniquement en utilisant les API système, graphiquement en utilisant les widgets natifs, fonctionnellement en reprenant «look & feel» et codes d'usage.
- › la facilité d'utilisation des spécificités de l'appareil mobile – bluetooth, NFC, GPS, Audio... sans pour autant que cela ne soit une exclusivité du développement natif.
- › les performances permises par une exécution directe ou pseudo-directe (dans le cas de l'interprétation du bytecode par la machine virtuelle Android) par rapport à une interprétation du JavaScript dans un navigateur ou une WebView, et du coup la qualité de l'expérience utilisateur qui en ressort.

- **Les faiblesses du développement natif**

Le développement natif a un défaut majeur : tout développement réalisé pour un système d'exploitation est limité à ce système.

Comme le marché est partagé entre les trois systèmes Android, iOS et Windows Mobile, il serait donc nécessaire de développer trois fois l'application pour atteindre la totalité des utilisateurs.

De plus, le marché Android est particulièrement fragmenté, car Google ne maîtrise ni le calendrier de déploiement des nouvelles versions d'Android par les constructeurs, ni le calendrier de mise à jour des appareils mobiles par ces mêmes constructeurs.

Il existe de manière courante, dans le catalogue des vendeurs de smartphones

et tablettes, des versions anciennes d'Android et la dernière version.

Cependant, ce défaut est largement réduit par l'importance considérable prise par Android sur le marché. Au niveau mondial, plus de 4 appareils sur 5 sont équipés avec Android.

La prise en charge d'une version minimale d'Android 4.3 permet d'assurer à l'heure actuelle une large compatibilité, alors même que la version 7 « Nougat » vient d'être déployée sur certains appareils.

Des particularismes locaux subsistent : en fonction de la clientèle visée par une application, l'ouverture sur iOS peut être indispensable.

5.2. Le développement Web Mobile

EN RÉSUMÉ

Le développement natif permet une intégration et une expérience utilisateur optimales. En contrepartie, le développement réalisé est spécialisé pour le système d'exploitation hôte.

La réalisation d'un site Web mobile consiste à développer un site web en respectant un certain nombre de règles de création du fond et de la forme afin d'assurer la compatibilité de ce site à une consultation sur appareil mobile.

L'approche aujourd'hui la plus courante est de concevoir un site Web d'abord pour les appareils mobiles (approche «mobile-first») et en élargissant la conception de proche en proche vers les dimensions d'écran plus confortables : on parle de Responsive Web Design ou RWD.

Les principales adaptations de forme sont :

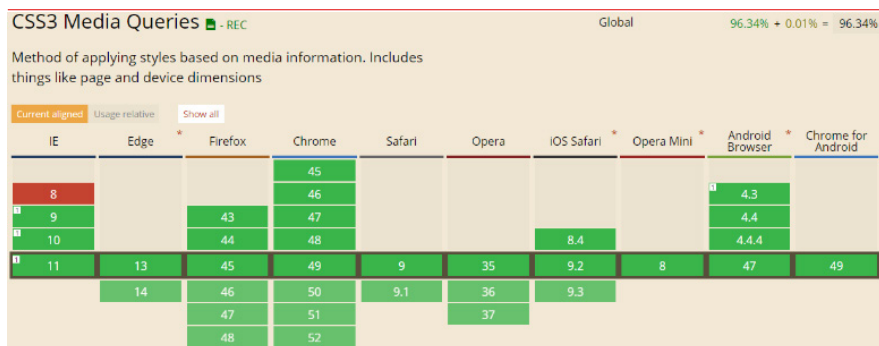
- › gérer les différentes tailles d'écran en introduisant des points de rupture; les balises CSS3 media queries permettent d'adapter la présentation d'une page à l'espace disponible sur l'écran.
- › Ces points de rupture vont permettre de définir la transformation du menu, de menu « burger » vers un menu exposé, l'augmentation du nombre de colonnes pour la présentation du contenu, la dimension des icônes de présentation, etc.
- › définir le contenu en fonction de la place disponible : mise en avant des chapeaux et de résumés d'articles, adaptation quantitative des images...

Les principales adaptations du fond sont liées à la sélection du contenu présenté :

- › suppression ou élagage du contenu d'importance moindre,
- › dégradation qualitative des images présentées.

L'objectif premier étant d'offrir une bonne lisibilité et une bonne navigabilité sur des écrans de taille réduite, sur des appareils n'ayant ni clavier, ni souris, ni trackpad...

Ces balises CSS media queries sont aujourd'hui supportées par l'ensemble des navigateurs mobiles et fixes «majeurs», comme le montre cette capture issue du site CanIUse.



LE SUPPORT DES MEDIA QUERIES SUR LES PRINCIPAUX NAVIGATEURS DU MARCHÉ. SOURCE : <http://caniuse.com/#search=mediaqueries>.

(Ce site www.canIuse.com est fort utile pour déterminer quelles sont les techniques permises par les navigateurs cibles; il intègre en effet les navigateurs mobiles.)

- **Les forces du développement Web mobile**

Un site Web mobile est consultable sur l'ensemble des trois familles Android, iOS, Windows Mobile, ainsi que sur ordinateur portable ou fixe, quelque soit le système d'exploitation : Windows, OS-X, Linux...

Le site Web mobile est accessible sans demander l'installation préalable d'une application sur le device : la saisie d'une URL, un lien sur une page tierce, le scan d'un QR-Code imprimé sur un contrat, un flyer, un magazine... sont autant de moyens simples et rapides d'arriver sur le site.

Il n'y a pas de procédure de recherche sur le store applicatif du système à réaliser par les utilisateurs.

De plus, le contenu du site Web est libre : le site Web n'est pas soumis à la validation pré-publication de son contenu par les stores, ceci permettant :

- › de réduire le délai de mise en ligne et de mise à jour du contenu,
- › de proposer le contenu souhaité sans restriction autre que légale,
- › d'être dégagé des obligations de paiement sur les articles numériques vendus (30% de commission chez Apple et Google).

- **Les faiblesses du développement Web mobile**

Une des faiblesses reprochées au site Web mobile est sa supposée lenteur liée à l'étape d'interprétation du JavaScript.

Cependant, les progrès réalisés dans l'interprétation du langage JavaScript par les moteurs de Chrome (moteur V8), Safari (JavaScriptCore) et Edge (Chakra JS) conduisent à des temps de traitement des scripts JavaScript très acceptables :

L'étape d'interprétation n'est plus un frein au développement d'applications réactives.

Ainsi, Microsoft met en avant les performances de son navigateur avec une démonstration de calcul de fractales, démonstration ensuite reprise et optimisée par Google sous Chrome...

De la même manière, le stockage local a longtemps été un maillon faible des sites Web (mobiles ou non), ce défaut a été résolu par l'usage des éléments HTML5 LocalStorage, IndexedDB et WebSQL (Langage SQL). Attention cependant au support de ce dernier élément, absent sur les navigateurs Microsoft Edge & IE, donc à une large proportion des PC. De fait, la spécification Web SQL n'étant plus maintenue, son support risque d'être abandonné dans le futur. De nouveau, le site CanIUse démontre son utilité.

Reste cependant un inconvénient majeur : Sauf mise en place d'un mécanisme particulier (Application cache, Service Workers), le site Web ne sera accessible à l'utilisateur que si celui-ci est connecté à Internet. Même avec des stratégies de cache étendues (incluant les scripts, les pages HTML, les feuilles de style...), la possibilité d'un effacement des ressources en local par le navigateur, l'utilisateur ou un outil de nettoyage tiers est toujours forte.

Un autre inconvénient, moindre, est celui de l'absence de l'application développée sur les Stores applicatifs : les usagers ont pris l'habitude de chercher les applications dans leur Store, et n'ont pas, sur device mobile, le réflexe Web.

Enfin, malgré quelques passerelles offertes par HTML5, toutes les fonctionnalités des appareils mobiles ne sont pas accessibles. Citons toutefois que l'accès à l'appareil photo, au microphone sont permis par des champs de type input ou la fonction `getUserMedia()`, que la géolocalisation est possible via `getCurrentPosition()`, et que l'usage de la fonction téléphone est rendue possible par un simple lien href.

Les fonctionnalités qui restent inaccessibles sont l'accès au Bluetooth, le NFC, et de manière plus générale tout ce qui implique de sortir de l'environnement cloisonné du navigateur.

EN RÉSUMÉ

Le développement d'un site web mobile permet une compatibilité étendue et un déploiement simplifié. En contrepartie, le développement réalisé est générique et limité dans son accès aux fonctions natives des appareils mobiles.

5.3. Le développement hybride

Offrir une application hybride permet de pallier ces problèmes liés à la restriction des sites web mobiles à l'environnement du navigateur, tout en limitant les coûts de développement en créant une application unique multi-plateformes. Nous avons vu que les sites Web mobiles proposent des possibilités intéressantes; leur offrir l'ensemble des fonctionnalités des applications natives et une présence sur les Stores Applicatifs est possible en les encapsulant dans un conteneur natif. Ce conteneur natif leur donnera accès aux fonctionnalités natives des appareils mobiles, aux API des systèmes d'exploitation ainsi qu'aux stores applicatifs.

- **Les forces du développement hybride**

L'application hybride est une excellente solution pour les applications basées sur des formulaires, ou dédiées à de la communication institutionnelle, ou bien devant offrir une interface sobre et pour lesquelles l'aspect «waouh» n'a pas ou peu d'importance. Il existe de multiples frameworks permettant la réalisation d'applications hybrides; l'outillage de test est étendu, et la mise en ligne est celle des applications natives : les applications hybrides sont disponibles sur les stores applicatifs.

Les coûts de développement d'une application multi-plateformes sont limités par rapport aux coûts de développement d'une application native, sans toutefois être aussi bas que les coûts du site Web mobile, car il faut intégrer les coûts d'intégration dans le conteneur natif et le coût éventuel de développement des fonctionnalités natives.

Il est nécessaire de bien anticiper les problématiques liées au besoin en fluidité (nécessaire à la qualité de l'expérience utilisateur), car :

Si la complexité graphique ou fonctionnelle de l'application dépasse un certain seuil, des ralentissements sont à prévoir, et leur élimination s'avère difficile voire impossible.

- **Les faiblesses du développement hybride**

N'ayant pas accès, depuis le conteneur Web, à l'ensemble des fonctionnalités offertes par les appareils mobiles, l'application hybride qui en a besoin devra passer par des plugins natifs spécifiques à chacun des systèmes d'exploitation cible, ou de manière plus coûteuse par des développements ad hoc.

Dans ce dernier cas, la promesse d'un code unique partagé par tous les terminaux n'est plus tenue. L'autre faiblesse fréquemment reprochée aux applications hybrides est leur manque de réactivité, leur niveau d'expérience utilisateur inférieur. Dans la pratique, cette dernière faiblesse n'est vérifiée que si les étapes de conception de l'UI/UX ont été sous-estimées. Le manque de réactivité peut être un réel problème sur les matériels d'entrée de gamme.

EN RÉSUMÉ

Le développement d'une application hybride permet une compatibilité étendue et une présence sur les stores, ainsi que l'accès aux fonctions évoluées des appareils mobiles. Un effort particulier de conception de l'interface et de l'expérience utilisateur doit être réalisé pour obtenir de bons résultats. Il ne faut pas attendre une fluidité parfaite sur tous les matériels.

5.4. Le développement cross-compilé

C'est une solution technique dont la maturité est assez récente, mais réelle. La promesse est de créer une application en la codant une seule fois, en s'appuyant sur un environnement de développement dédié, fournissant un SDK externe aux différents systèmes d'exploitation des appareils mobiles.

La philosophie de l'application hybride est assez proche, à ceci près que dans le cadre du développement cross-compilé, le code passe par une étape de compilation et de linkage vers les API systèmes.

L'objectif est d'atteindre les niveaux d'intégration et de fluidité des applications natives avec la généralité des sites Web mobiles.

Les principales solutions de développement cross-compilé permettent de générer des applications vers les trois systèmes d'exploitations mobiles majeurs.

- **Les forces du développement cross compilé**

La promesse est globalement tenue ; en s'appuyant sur des SDK tiers, et en utilisant des langages tiers (Javascript, C#) communément utilisés et donc sans qu'il ne soit difficile de monter en compétences dessus.

Le code est plus ou moins compatible avec l'ensemble des plates-formes; certains outils annoncent un taux de réutilisation de 60% à 90% selon les plates-formes, alors que d'autres sont à 100%.

Dans tous les cas, l'accès aux fonctionnalités natives des appareils mobiles est rendu possible via le SDK fourni.

Les performances des applications générées sont comparables à celles développées nativement. Enfin, les plates-formes sont souvent complétées par des outils de test, de déploiement, de profilage, etc.

- **Les faiblesses du développement cross-compilé**

Le développement cross-compilé demande une montée en compétence sur les SDK des plates-formes tierces : ainsi un développeur Javascript ou C# n'est pas immédiatement opérationnel sur les outils tiers, de la même manière qu'un développeur JavaScript n'est pas immédiatement opérationnel sous Ionic ni qu'un développeur Java sous Android. Second point, il existe nécessairement un temps de latence entre la sortie d'une nouvelle version d'un système d'exploitation mobile et l'intégration des nouveautés au sein des SDK tiers; cette latence est cependant réduite car des versions bêta des systèmes d'exploitation mobiles sont généralement diffusées par leurs éditeurs.

Troisième élément à prendre en considération :

Le fait de s'appuyer sur une plate-forme tierce introduit une dépendance envers son éditeur, sa feuille de route propre et sa stratégie commerciale.

EN RÉSUMÉ

Le développement d'une application cross-compilée permet une compatibilité étendue et une présence sur les stores, ainsi que l'accès aux fonctions évoluées des appareils mobiles. Le taux de réutilisation du code d'une plate-forme à une autre est variable.

Trouver des développeurs compétents est difficile, une phase de formation amont est souvent nécessaire.

Le risque majeur est celui de l'introduction d'une dépendance à la feuille de route technique et commerciale d'un acteur tiers.

6 — LES ENVIRONNEMENTS TECHNIQUES DE DÉVELOPPEMENT MOBILE

Développer une application mobile, quelle que soit l'architecture technique sélectionnée, suppose la maîtrise d'un environnement constitué de plusieurs éléments :

- › le langage de programmation
- › l'environnement technique
- › l'environnement de développement intégré
- › les outils de test et de recette
- › le mécanisme de publication

6.1. L'environnement technique Android natif

- **Le langage de programmation**

Une application Android native est programmée en Java. Le code Java est transformé en bytecode, comme en Java «standard», mais en bytecode destiné à la machine virtuelle Dalvik de Google, spécialement optimisée pour les appareils mobiles, moins performants que les PC ou serveurs qui hébergent les applications Java. Les fichiers de bytecode Dalvik ne sont pas exécutables sur une VM Java d'Oracle.

Depuis Android 5, la VM Dalvik est remplacée par ART (Android Runtime). Si Dalvik interprète le bytecode à chaque exécution d'une application (compilation Just-In-Time), ART procède à sa traduction en instructions natives une seule fois. Cette traduction est optimisée pour le matériel détecté.

Ces considérations sont transparentes pour le développeur Java – qui n'a besoin que d'apprendre les spécificités du SDK Android.

Dans certains cas particuliers, une recherche de performances exceptionnelles

peut être nécessaire : le développeur peut alors implémenter une partie de son application en C ou C++. Les cas d'usages proposés par Google sont le développement de moteurs graphiques, de traitement du signal, de simulation physique...

- **Un environnement ouvert**

L'environnement technique Android est relativement aisé à mettre en place et le niveau de compétence requis est moyen.

Le socle est composé d'une JVM (de préférence Oracle), d'un JDK, d'un système de build : initialement Ant, puis Maven, ont été supplantés par Gradle, ce dernier étant l'outil poussé par Google pour le build dans son environnement de développement.

L'environnement de développement de référence est celui fourni par Google : il s'agit d'Android Studio.

Android Studio est basé sur l'IDE Java IntelliJ ; il est proposé en version stable depuis décembre 2014. Cet environnement est fourni gratuitement sous licence Open Source Apache 2.0. Cet environnement a atteint la maturité.

Le JDK Oracle est disponible pour les plates-formes Linux, Mac OS X, Solaris et Windows. Android Studio recommande l'usage du JDK d'Oracle par opposition à l'usage d'OpenJDK, cependant OpenJDK peut être utilisé. L'usage d'OpenJDK n'aura pas d'impact sur la compatibilité du bytecode généré sur les terminaux mobiles.

Gradle est disponible pour ces mêmes environnements (Gradle est un outil écrit en Java et Groovy).

Enfin, Android Studio est disponible sous Windows, Mac OS X et Linux à l'URL [***https://developer.android.com/studio/index.html***](https://developer.android.com/studio/index.html).

- **Un IDE complet : Android Studio**

Android Studio est un IDE dédié au développement mobile pour Android. L'IDE Android Studio est fourni au sein d'un pack qui comprend également le gestionnaire de build (Gradle), le gestionnaire de bibliothèques (le SDK Manager), l'outil de prévisualisation des écrans, un client de gestion de configuration (Git), un outil de création et de gestion de machines virtuelles Android etc.

- **Un test facilité**

Le principal reproche qui peut être fait à la plate-forme Android est la complexité d'assurer un développement de qualité tant la plate-forme est fragmentée.

Le test d'une application Android se fait de différentes manières depuis l'IDE Android Studio :

- › avec un appareil mobile Android physiquement connecté : cette facilité permet d'accéder à des outils de monitoring des ressources physiques (CPU, mémoire, GPU, réseau) et de valider le bon fonctionnement de l'application sur le device cible.
- › sur un émulateur Android : dans le cas d'une application grand public, il est totalement illusoire de vouloir posséder physiquement l'ensemble des appareils mobiles Android cibles (multiplicité des versions d'OS, des constructeurs, des modèles... et des mises à jour). Cependant l'émulateur de terminaux Android permet de définir un jeu d'émulateurs ; ainsi l'application peut être testée sur plusieurs formes (smartphones, tablettes), plusieurs densités d'écran, plusieurs versions d'OS. Il reste que ces tests ne permettent pas de détecter toutes les anomalies et peuvent devenir vite onéreux en temps passé.

Sous Android, le test des applications est un élément indispensable à leur qualité ; c'est cependant toujours un point difficile.

Certaines sociétés se sont d'ailleurs spécialisées dans la création et la maintenance de parcs de smartphones et de tablettes Android, sur lesquels une application peut être déployée et jouée à distance.

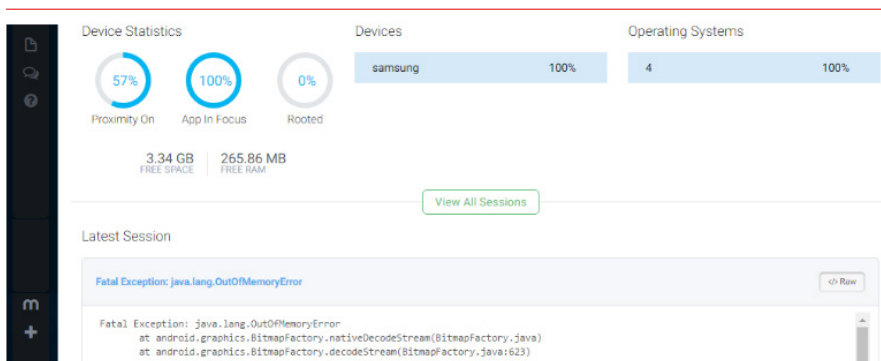
- **La détection des anomalies en production**

Une fois l'application publiée, elle se retrouve exploitée sur l'ensemble du parc cible. Dans le cas d'une application grand public, ou simplement si les appareils mobiles de ses utilisateurs ne sont pas déterminés, l'importante fragmentation de ces appareils va faire apparaître des anomalies auparavant non détectées.

Afin de maintenir l'application développée à un niveau de qualité satisfaisant, il est nécessaire de pouvoir détecter les anomalies, les recenser, obtenir un rapport d'erreur (une «stack trace» dans le monde Java / Android), pour enfin les corriger.

A noter que ce «niveau de qualité» est un curseur qui doit être positionné en fonction des contraintes de budget, de temps disponible, d'image de marque... et de durée de vie prévue de l'application !

Pour cette détection des anomalies, l'outil le plus adapté est «Crashlytics», un



UN EXEMPLE DE RAPPORT CRASHLYTICS MONTRANT UN DÉBUT DE STACK TRACE.

outil disponible pour iOS et Android, qui est une part de l'offre Fabric.io . En quelques lignes de code, l'application est connectée au Cloud Crashlytics qui compile les crashes, leurs causes, et offre des rapports complets sur les erreurs rencontrées. Crashlytics est un produit offert par Twitter.

Les rapports Crashlytics permettent de visualiser :

- › le nombre d'anomalies fatales (crash applicatif) ou non,
- › le nombre d'utilisateurs impactés (facilite la priorisation des correctifs),
- › la trace de l'exception (la stack trace) qui a causé le crash sur l'appareil de l'utilisateur.

Google propose également un mécanisme de suivi des crashes en production mais ce dernier est aujourd'hui moins convivial.

- **Le mécanisme de publication**

Sous Android, le mécanisme de publication sur le store applicatif «Google Play» est assez simple et passe par un outil en ligne, la «Google Play developer console».

- › la première étape est de se connecter à la console avec un compte Google.
- › la seconde étape est d'accepter «*Google Play Developer Distribution Agreement*» la licence de distribution.
- › l'étape suivante est de payer 25 dollars américains. Ce coût est un coût d'entrée non renouvelable, valable sans limitation de temps.
- › enfin il est nécessaire renseigner les informations qui seront présentées aux utilisateurs du Store et les informations de contact pour Google.

A partir de ce moment il est possible de publier une nouvelle application ou une nouvelle version d'une application existante sur le store Google Play. Cette licence n'est pas obligatoire si vous souhaitez diffuser l'application

par un autre canal, par exemple en permettant le téléchargement direct du binaire (fichier APK) de l'application. Cette façon de fonctionner peut être aussi utilisée si vous disposez d'un outil de Management des Terminaux Mobiles (MDM). De plus, la licence donne accès aux services de Google : les notifications natives, le paiement in-app, etc.

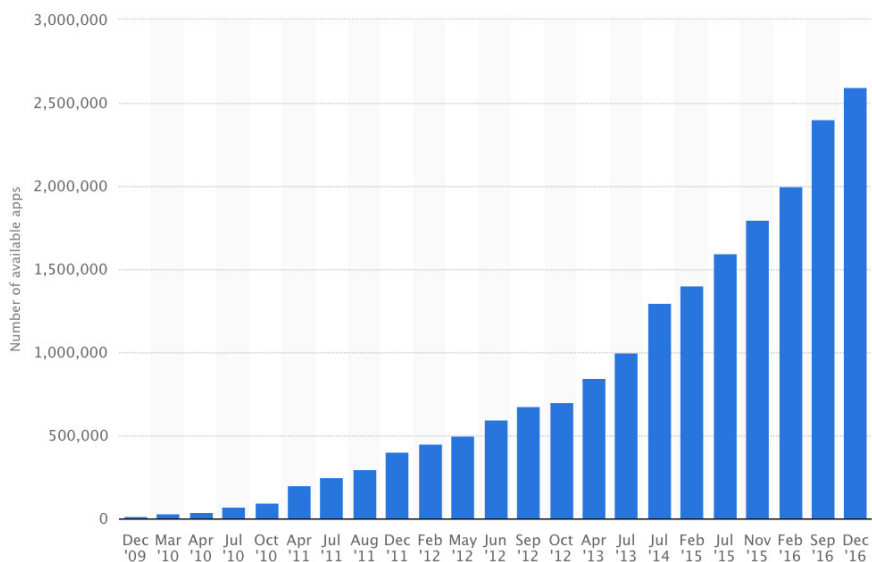
La publication peut se faire sur l'un des trois canaux suivants :

- › canal de test Alpha : peut être ouvert ou fermé,
- › canal de test Bêta : peut être ouvert ou fermé,
- › canal de production.

Les canaux de test permettent la diffusion d'une version (de test) à une liste déterminée d'utilisateurs (canal fermé) ou à tout utilisateur disposant du lien d'accès à l'application (canal ouvert). Ce lien est fourni par la console. Le canal de production met l'application à disposition sur le store applicatif. Les applications passent par une étape de validation par Google: il faut noter que sont interdites les applications comportant des images violentes, incitant à la haine ou portant sur des jeux d'argent ou de hasard.

Le descriptif précis des interdictions est donné par Google à l'URL ***<https://play.google.com/about/developer-content-policy.html>***

La validation d'une application est réalisée automatiquement, cette étape



NOMBRE D'APPLICATIONS PUBLIÉES SUR LE STORE APPLICATIF GOOGLE.

EN RÉSUMÉ

La publication d'une application Android est une opération assez simple, relativement rapide et peu onéreuse. La procédure d'inscription peut être anticipée par rapport à la publication effective en production.

La simplicité de publication a permis au store Google Play d'offrir presque 2 600 000 applications en Décembre 2016 (source Statista.com).

est donc assez rapide.

6.2. L'environnement technique iOS natif

- **Les langages de programmation**

Une application native pour appareils iOS peut être développée suivant deux langages : Objective-C ou Swift.

Swift est un langage de programmation nouvellement inclus par Apple sur leur plateforme de développement. Ce langage moderne est plus facile à utiliser qu'Objective-C et la courbe d'apprentissage initial moins raide. Enfin, toutes les API iOS sont disponibles en Swift (comme elles le sont en Objective-C). Apple indique que Swift est le langage à privilégier dans le développement d'applications natives.

Objective-C est le langage historique des applications Apple, que ce soit pour OS-X ou iOS. En conséquence, le volume des applications écrites en Objective-C est plus important, et il est plus aisé de trouver des ressources compétentes en Objective-C. De plus, une société ayant déjà un parc d'applications Objective-C doit conserver des compétences pour en assurer le maintien en conditions opérationnelles.

Ainsi, le développeur idéal d'applications natives iOS devrait connaître les deux langages : l'un pour maintenir les applications légataires (si tant est qu'on puisse parler de légataire dans les applications mobiles) et l'autre pour développer les nouvelles applications.

Il n'y a pas de différences entre une application développée en Swift et une application Objective-C du point de vue de l'utilisateur final.

Le SDK Apple est disponible pour les deux langages.

La maturité d'Objective-C est excellente, Swift est un langage encore jeune et dont les évolutions sont fréquentes.

- **Un environnement fermé**

La société Apple conçoit, construit et vend dans ses boutiques Apple des ordinateurs et des appareils mobiles Apple fonctionnant exclusivement avec

un système d'exploitation Apple.

Les avantages de ce modèle en termes de stabilité matérielle et applicative, de qualité de design et d'intégration tant matérielle que logicielle sont évidents et contribuent à la notoriété et au succès de la marque Apple.

L'inconvénient principal de cet écosystème clos est subi par le développeur, qui doit s'équiper en matériels et logiciels Apple pour pouvoir développer ses applications pour iOS.

Plus précisément, il est nécessaire pour développer une application mobile native pour iOS de disposer :

- › de l'EDI Xcode, disponible sur le store applicatif iTunes, pour OS-X ; télécharger cet EDI impose la création d'un compte iTunes et donc la fourniture d'un numéro de carte de paiement valide.
- › d'une machine sous Mac-OS (donc d'une machine Apple)

L'avantage de Xcode est de proposer l'intégralité du système de compilation et de build. Le téléchargement du SDK et ses mises à jour se font de manière automatique.

- **Un IDE complet : Xcode**

Xcode est une suite complète d'outils de développement, de test, de prévisualisation, et de publication d'applications.

Conçu pour permettre le développement d'applications via de multiples langages, Xcode permet le développement sous Objective-C et Swift, et avec l'addition du SDK iOS la création d'applications mobiles à destination de l'architecture ARM utilisée par les appareils mobiles d'Apple.

Les principaux composants de la suite Xcode sont :

- › l'EDI lui-même également appelé Xcode,
- › l'outil de construction d'interfaces Interface Builder,
- › le compilateur Clang,
- › le débogueur LLDB,

- › Un simulateur de terminaux iOS, tvOS, watchOS.
- **Le test applicatif**

Xcode permet trois types principaux de test : le test fonctionnel, le test de performances et le test d'interface utilisateur.

Les tests fonctionnels et de performances nécessitent un développement spécifique, le plus souvent sous forme de test unitaire, avec de manière habituelle les méthodes de mise en place (`setUp`), `run` (`testExample`), et remise en état (`tearDown`) de l'application.

De manière intéressante, les tests d'interface utilisateur peuvent être joués par enregistrement d'un processus utilisateur sur un simulateur, le test consistant alors à vérifier en fin de rejeu l'état d'éléments de l'interface.

Cette fonctionnalité n'est offerte que pour les applications développées pour iOS 9 ou plus.

- **La détection des anomalies en production**

Apple a intégré dans la suite Xcode l'outil de collecte et de présentation des rapports de crash des applications mobiles.

Le service Apple de reporting de crash fonctionne de la manière suivante :

- › collecte des journaux de crash des applications déployées
- › remplacement des adresses mémoires par des noms lisibles
- › collation des logs pour identifier et grouper les rapports similaires
- › suppression de toutes les données nominatives des journaux
- › calcul du nombre de terminaux différents victimes du crash
- › fourniture d'un exemple de journal pour chacun des types de crash
- › mise à jour quotidienne des rapports de crash

Ces rapports de crash sont visibles dans Xcode dans la fenêtre « Crashes organizer ».

La consultation et la correction du code source incriminé est facilitée par un survol de la stack trace à la souris puis un clic sur la flèche qui apparaît alors. Ce mécanisme est donc totalement intégré, ce qui le différencie du système

« Crashlytics » d'Android.

Cependant dans un souci d'homogénéisation et de centralisation des rapports, il est également possible d'utiliser Crashlytics pour iOS.

- **Le mécanisme de publication**

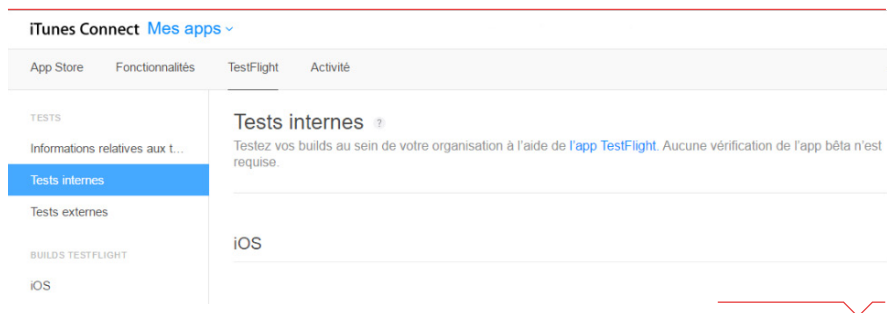
Le déploiement d'une application peut se faire selon plusieurs mécanismes, variables selon qu'il s'agisse d'une application « Grand public » ou d'une application d'entreprise.

Dans les deux cas, un compte développeur Apple est indispensable : ce point doit être pris en compte si votre application a été développée par un développeur tiers disposant de son propre compte.

› Publication pour recette

Depuis le rachat par Apple de l'application Testflight, Xcode permet la publication en version recette de vos applications.

Cette publication pour recette consiste en une pré-publication sur iTunes suivie d'une mise à disposition aux testeurs par le biais de l'application Testflight. Ce mécanisme permet de proposer une application en test à une



LA PUBLICATION POUR RECETTE
DANS LA CONSOLE DÉVELOPPEUR D'APPLE.

population pouvant comprendre jusqu'à 2000 testeurs.

Seule l'adresse e-mail des testeurs doit être connue pour pouvoir leur proposer le test : cette solution est donc assez simple à mettre en œuvre et permet d'avoir des retours sur une population représentative.

Pour une recette dite « interne », aucune validation de l'application n'est réalisée par Apple.

Pour une recette dite « externe », une validation de l'application est réalisée par Apple.

› Publication en production sur l'App Store

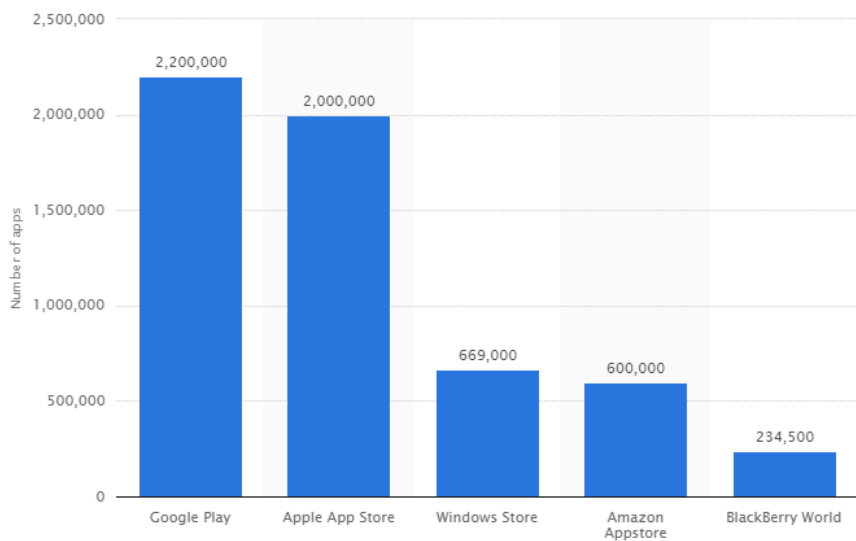
L'application doit être packagée et signée avec le certificat de distribution via Xcode. L'upload de l'application vers iTunes peut se faire depuis Xcode. La console d'administration « iTunes Connect » est ensuite le moyen de suivre l'avancement de la validation de l'application par Apple, puis de déterminer le jour de mise à disposition au grand public.

› Publication en production en dehors du Store

Ce mécanisme de publication est réservé aux entreprises membre du « Apple Developer Enterprise Program ». Il permet la mise à disposition des applications en direct (sans passer par le Store) vers les salariés de l'entreprise titulaire.

La création de l'archive au format d'application iOS (extension **.ipa**) se fait depuis Xcode. La distribution de l'application se fait alors en mettant à disposition le fichier ipa à disposition des utilisateurs.

Pour la première installation d'une application d'entreprise, une étape supplémentaire d'acceptation du certificat d'entreprise devra être réalisée par l'utilisateur final : cette étape n'est pas triviale (elle est de complexité basse à moyenne) et peut bloquer certains des utilisateurs les moins technophiles : elle est donc à prendre en compte dans ce cas particulier d'une première installation.



VUE COMPARATIVE DU NOMBRE D'APPLICATIONS SUR LES PRINCIPAUX STORES APPLICATIFS.

EN RÉSUMÉ

La publication d'une application iOS est une opération de complexité moyenne, relativement rapide et raisonnablement onéreuse. La procédure d'inscription peut être anticipée par rapport à la publication effective en production et il est conseillé de le faire afin d'éviter des délais supplémentaires.

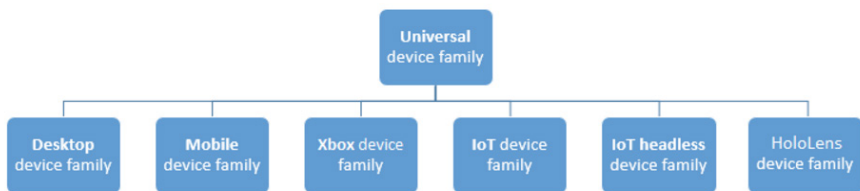
Le store Apple App store offre à date 2 000 000 d'applications (source Statista.com).

6.3. L'environnement technique Windows natif

La présentation est faite ici de manière focalisée sur Windows 10. Cette version du système d'exploitation de Microsoft est en effet portée sur toutes les familles d'appareils, du smartphone à la tablette, et les applications pour Windows 10 sont utilisables sur ces différents appareils – modulo leur adaptation lors du développement.

La plateforme Windows 10 est donc « Universelle » et introduit le socle de développement partagé entre les différents terminaux, appelée la « Universal Windows Platform » (UWP).

UWP fournit à une application non seulement les APIs communes à l'ensemble des terminaux mais aussi les API (y compris les API Win32 et .NET) spécifiques à la famille de device accueillant l'application. Ces API étant spécifiques, il revient à l'application de vérifier leur présence de manière dynamique : l'API « Desktop device family » n'est pas présente dans



TOUTES LES FAMILLES HÉRITENT DU SOCLE COMMUN « UNIVERSAL DEVICE FAMILY ».

l'API

« Xbox device family » par exemple.

- **Les langages de programmation**

Les développeurs d'applications mobiles pour Windows 10 bénéficient d'une grande liberté de choix pour le langage de programmation : Visual Basic, Visual C#, Visual C++, JavaScript.

A noter qu'à l'exception de JavaScript qui est conforme à la spécification ECMAScript 5ème édition, Visual Basic, Visual C# et Visual C++ sont des langages spécifiques à Microsoft.

Le langage Visual C# peut être utilisé pour réaliser des applications Windows Universelles, Android et iOS.

JavaScript peut être utilisé pour réaliser des application Windows Universelles et des applications hybrides via l'intégration de Cordova.

Les langages Visual Basic et C++ peuvent être utilisés pour réaliser des applications Windows Universelles.

Ainsi le choix du langage est assez ouvert ; cependant les développeurs Windows ont l'habitude du C#, ce langage est parfaitement mature et s'intègre parfaitement avec le le framework .NET. Ce langage est assez proche du langage Java.

- **Un environnement entrouvert**

En pratique, l'environnement Windows 10 est à peine plus ouvert qu'un environnement Apple, et nettement moins qu'un environnement Android.

Le développeur d'applications UWP doit être équipé d'un poste de travail Windows obligatoirement, il n'est pas possible de développer, compiler et tester d'applications UWP sous un autre système d'exploitation facilement

(c'est possible mais compliqué). Par contre l'univers des ordinateurs personnels compatibles avec Windows est vaste et il est aisé de trouver des machines adéquates au développement à faible coût.

De plus si l'IDE Visual Studio n'est disponible que sous Windows, il en existe une version communautaire gratuite qui permet le développement d'applications UWP. Les fonctionnalités manquantes les plus gênantes sont celles liées au test automatique des applications. Une grande rigueur lors de la phase de développement permet de pallier ces manques.

L'installation de Visual Studio 2015 s'accompagne des frameworks .NET depuis la version 4.

- **Un IDE complet : Visual Studio**

Visual Studio est l'IDE de référence pour tous les développements à destination de la plate-forme Microsoft Windows. Il existe depuis 1997 et sa version Visual Studio 97, et bénéficie donc d'une grande maturité.

Cet outil est proposé sous plusieurs moutures, la mouture communautaire « Microsoft Visual Studio Community » permet de réaliser des applications complètes et complexes. Elle est téléchargeable gratuitement par les développeurs indépendants. Pour les entreprises, elle est disponible sous Extrêmement modulaire et ouvert à de nombreux langages (C++/C#/F#/JavaScript/Python/VB.Net etc.), cet outil est mis à jour fréquemment et suit les évolutions du système d'exploitation Windows et des services Microsoft.

Enfin, pour le développement cross-plateformes :

- › l'extension Xamarin permet la cross-compilation des applications natives UWP vers iOS et Android.
- › Apache Cordova est intégré et permet la réalisation et la compilation d'applications hybrides pour iOS , Android et Windows.

Visual Studio comprend entre autres :

- › un éditeur de code permettant la complétion automatique,

- › un compilateur adapté au langage utilisé,
- › un émulateur de terminaux Windows (Microsoft device simulator),
- › un émulateur de terminaux Android basé sur Hyper-V, offrant de bonnes performances (également compatible avec Android Studio) et pouvant être utilisé de manière autonome,
- › un déboggeur pas-à-pas,
- › un outil de gestion des extensions permettant d'ajouter facilement des modèles de projet, des modules techniques etc..., classés par langages.

Visual Studio est un outil puissant, riche et performant : dans le cadre du développement d'une application UWP, il n'est pas justifié d'utiliser un autre IDE, sauf cas particulier.

- **Le test applicatif**

Visual Studio apporte une batterie d'outils de tests unitaire :

- › l'explorateur de tests pour l'exécution des tests unitaires et de leur résultat,
- › l'infrastructure de test unitaire pour le code managé, qui fournit une infrastructure pour tester le code .NET,
- › l'infrastructure de test unitaire pour C++, qui fournit une infrastructure pour tester le code natif,
- › des outils de couverture de code,
- › une infrastructure d'isolement « Microsoft Fakes » qui peut créer des classes et des méthodes de remplacement,
- › enfin, la version payante « Enterprise » de Visual Studio fournit l'outil IntelliTest qui génère seul des données de test et une suite de tests unitaires, mais pour C# en 32 bits seulement.

Pour le test fonctionnel, il existe de très nombreux frameworks, les plus courants étant SpecFlow (<http://www.specflow.org/>), Concordion (<http://concordion.org/>) et Fitnessse (<http://www.fitnessse.org/>), trois projets open Source et gratuits.

Concordion existe également pour Java, Specflow est particulièrement bien

intégré à Visual Studio.

Quant à lui, Fitness implique une infrastructure un peu plus riche incluant un Master déployé sur un serveur et des esclaves sur les postes de développement, mais il est plus accessible aux Product Owners qui seraient purement fonctionnels.

6.4. L'environnement technique des sites Web mobiles

- **Les langages de programmation**

Un site Web se développe aujourd'hui avec les langages du Web. Si cette assertion semble être une lapalissade, elle cache en réalité le fait que les multiples frameworks et moteurs de rendu RWD JavaScript/CSS/HTML sont revenus en force, après une période durant laquelle ont été à la mode les outils tels que GWT, Flex, ou même Flash, Silverlight et autres JavaFX. Le précurseur JavaScript, maintes fois annoncé comme mourant voire disparu, est revenu sur le devant de la scène en force.

Le triptyque JavaScript/HTML5/CSS3 constitue le socle de développement des sites Web mobiles, il doit être complété par les frameworks adéquats pour permettre au développeur de se concentrer sur le travail de mise en place du site et non sur la gestion des subtilités d'affichage.

Le choix des frameworks est un travail d'architecte et va être fortement dépendant de la structure du site, des contraintes techniques et fonctionnelles prévues, etc. On peut cependant citer :

- › Bootstrap, un outil développé par Twitter et rendu Open Source. C'est un ensemble de styles CSS et de composants JavaScript qui permet d'accélérer la mise en place d'un site RWD.
- › AngularJS est un outil développé et maintenu par Google et fournit un jeu de composants permettant de structurer et organiser le site. Il fournit à un site web Guideline et architecture.
- › Angular 2 est une version nouvelle – plus qu'une évolution, une révolution – d'AngularJS et apporte davantage de fonctionnalités et d'outils de développement lors de la réalisation. En plus de JavaScript, Dart et TypeScript peuvent être utilisés avec Angular2.

- **Un environnement ouvert**

Il est possible de développer un site Web mobile sur n'importe quel environnement : Windows, Linux, OS-X, et même sous Android et iOS – la différence se faisant au niveau des habitudes de travail de chaque développeur.

Chacun des langages (JavaScript, HTML, CSS) est un langage qui n'est lié à aucun éditeur en propre, à condition d'éviter les déclinaisons apportées par certains (Jscript, ActionScript).



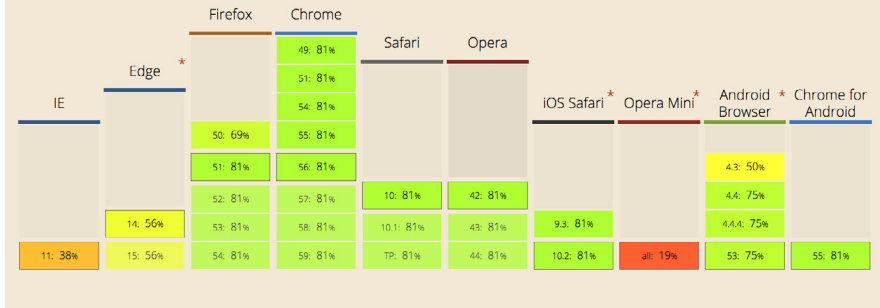
RÉSUMÉ DU NIVEAU DE SUPPORT D'ECMASCRIPT 7 PAR LES PRINCIPAUX NAVIGATEURS DU MARCHÉ. SOURCE : CANIUSE.COM

Les normes actuelles sont :

› pour JavaScript : ECMA Script, 7ème édition, est la spécification de référence la plus récente. CanIUse indique que l'ensemble des navigateurs supporte la version 5 de la spécification, alors que la version 6 n'est encore que partiellement supportée.

Summary

Calculation of support for currently selected criteria



RÉSUMÉ DU NIVEAU DE SUPPORT DE HTML5
DANS LES PRINCIPAUX NAVIGATEURS DU MARCHÉ. SOURCE : CANIUSE.COM

› Pour le HTML : HTML5 est la spécification de référence, publiée en Octobre 2014 par le W3C. Parmi les apports principaux d'HTML 5, on peut citer les options de formulaires étendues («Form features» ou Web Forms 2.0), le support des balises SVG, les éléments sémantiques, les connexions persistantes (interface EventSource), et enfin le WebStorage permettant le stockage local d'éléments. Certains de ces éléments sont parfaitement supportés, d'autres ne le sont que partiellement. En particulier, IE 8 ne supporte pas ces éléments car il est très antérieur à la spécification.

De nouveau, en fonction des besoins fonctionnels et de la cible du site mobile, il conviendra de se rendre sur CanIUse.com pour valider l'usage de telle ou telle fonctionnalité avancée de HTML5.

› CSS3 : la spécification CSS3 est un ensemble de modules, chaque module étant une extension ou un complément de fonctionnalités définies dans la spécification CSS2. Les premiers brouillons de spécifications ont débuté dès 1999 ; chaque module peut être au

niveau « Brouillon avancé » (« Working Draft » : WD), « Candidat » (Candidate Recommendation : CR) , Proposed Recommendation (PR) (équivalent de Release Candidate) et enfin atteindre le statut de recommandation officielle du W3C sous le statut « W3C recommendation » (REC).

L'implémentation des différents modules par les navigateurs dépend de plusieurs facteurs : le statut de la recommandation, la maturité du navigateur, et l'utilité pour les développeurs de la recommandation.

Un élément CSS3 est particulièrement important pour le développement d'un site Web mobile, il s'agit de la méthode Media Queries qui permet d'adapter les styles CSS en fonction de l'information sur le type d'écran qui sert à l'affichage, la taille de la fenêtre, etc...

- **Des IDE multiples**

Il existe une multitude d'outils permettant de développer des sites Web mobiles, pour ce qui est de la partie affichée : le « Front-End ».

Le développement des Web Services est hors périmètre.

Pour le développement Web, on peut citer :

› Aptana Studio (<http://www.aptana.com>) : Open Source et gratuit, cet éditeur est basé sur l'IDE Java Eclipse. Il intègre des fonctionnalités avancées d'édition HTML, CSS et JavaScript. Il propose également l'intégration avec GIT, un Wizard de déploiement , un débogueur JavaScript. Il est disponible pour Windows, Linux et OS-X.

› Visual Studio Code (<https://code.visualstudio.com>) : Open Source et gratuit, cet éditeur est offert par Microsoft. Fonctionnant sous Mac, Linux et Windows, c'est un IDE qui vise à permettre un cycle de release rapide en se focalisant sur l'essentiel : la coloration syntaxique, l'auto-complétion, la gestion de sources. Il permet d'ajouter des plugins pour avoir des fonctions complémentaires.

› Sublime Text (<https://www.sublimetext.com>) est un éditeur de texte riche de fonctionnalités, de raccourcis, qui permet au

développeur de se concentrer sur son code. Un outil de gestion d'extensions, Package Control, facilite la personnalisation de Sublime Text (<http://packagecontrol.io>).

› Brackets (<http://brackets.io>) est un outil Open Source qui intègre également des fonctions intéressantes : Live Preview, support des préprocesseurs CSS, etc. Il est possible d'ajouter des modules pour couvrir tel ou tel besoin. On peut regarder aussi Atom (<https://www.atom.io/>)

› Notepad++, UltraEdit sont des éditeurs de texte enrichis, plus légers, qui peuvent également trouver leur place dans la trousse à outil du développeur de site Web mobile

- **Le test applicatif**

La question du test d'un site Web mobile est une question délicate en raison du nombre important de navigateurs, de systèmes d'exploitation et du nombre de versions déployées.

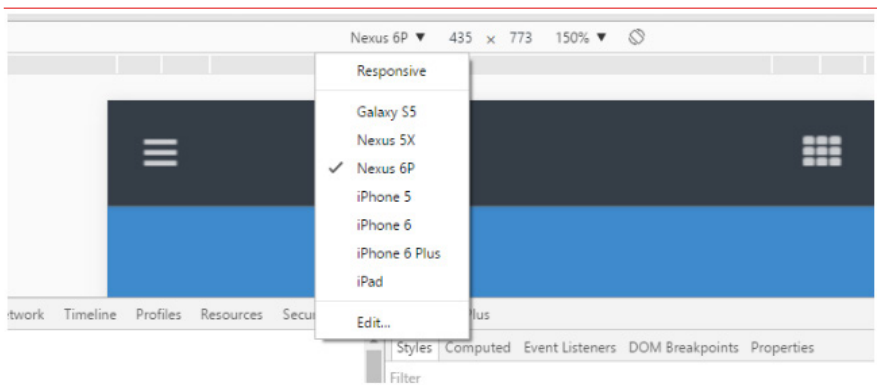
Cependant, dans le cadre d'un site Web mobile, on peut se focaliser sur les navigateurs embarqués sur les smartphones et tablettes, soit principalement (Source StatCounter pour la France, Juillet 2016 <http://gs.statcounter.com/#mobile+tablet-browser-FR-monthly-201607-201607-bar>):

- › Google Chrome et Safari autour de 40 % des utilisateurs chacun, et partagent le même moteur de rendu Webkit,
- › Samsung Internet : presque 10 % des utilisateurs,
- › Android Browser (présent sur les versions d'Android jusqu'à la 4.1 comprise) : 7 % des utilisateurs et en baisse régulière,
- › IEMobile pour 2 % des utilisateurs et en baisse légère,
- › et une multitude de navigateurs alternatifs se partagent les 3 % restant.

De ces statistiques – à nuancer selon la destination France ou Monde du site mobile- on peut retenir que 80 % des utilisateurs surfent sur un moteur Webkit, un effort de test sur Chrome & Safari est donc particulièrement important.

Les outils de test principaux sont :

- › En premier lieu, les navigateurs Chrome et Safari eux-mêmes, sur le poste de travail du développeur. Cependant des manques évidents liés au matériel mobile sont visibles : absence de géolocalisation, orientation de l'écran...
- › Chrome Developer Tools est une fonctionnalité intéressante de Chrome sur PC, tablettes et smartphones qui permet de simuler



LE CHOIX DU RENDU D'UN SITE RWD
DEPUIS LA CONSOLE DEVELOPER TOOLS DE CHROME.

rapidement une interface de dimension réduite, et le « tap » au lieu du « clic ». Accessible facilement, elle permet de simuler une gamme d'appareils mobiles.

Les fonctionnalités des Outils de développement sont multiples : analyse du modèle DOM de la page, visualisation de la console, débogage JavaScript, visualisation des latences réseau, des ressources consommées...

De plus, il est possible de connecter physiquement un appareil mobile à son poste de travail par USB et d'accéder aux mêmes informations à distance

(Remote Debugging).

- › Web Development Tools est un équivalent proposé au sein de Safari par Apple ; il est présent dans Safari sous OS X et sous iOS.
- › BrowserStack (payant) donne accès à un ensemble de terminaux physiques (iOS) et d'émulateurs (Android) sur lesquels le site mobile et son rendu peuvent être testés et validés. Des tests fonctionnels peuvent aussi être réalisés, soit en scripts (Python, Ruby, Java, PHP,...) soit depuis des outils d'intégration continue (Jenkins, Travis,...).

BrowserStack est un outil très complet qui permet de s'assurer d'une bonne couverture de tests sur un grand nombre de terminaux réels et de navigateurs. Un outil alternatif intéressant est le navigateur Blisk (<https://blisk.io>) qui permet de vérifier rapidement le rendu d'un site mobile sur une grande variété d'appareils mobiles (smartphones & tablettes) émulsés au sein de l'outil.



L'INTERFACE DE BLISK PERMET DE VISUALISER CÔTE À CÔTE LE RENDU WEB ET LE RENDU MOBILE

Enfin, un outil comme JSHint (<http://jshint.com>) permet de détecter des erreurs de codage ou de formalisme.

- **La détection des anomalies en production**

Lorsque le code du site Web est interprété côté client, de multiples erreurs peuvent se produire, comme des erreurs liées aux politiques d'accès des scripts aux ressources, les « Cross-Domain Exception », des problèmes de fuite mémoire, des timeouts d'accès aux ressources. La détection de ces anomalies doit se faire côté client et les rapports d'anomalies envoyées/centralisées vers un serveur. Une solution comme Sentry (<https://getSentry.com>) offre un panel de solutions pour différents frameworks JavaScript dont Angular 1 et 2. Sentry est disponible en version hébergée (sur abonnement) et en version Docker pour installation dans un S.I. propre. Sentry est Open Source et peut être téléchargé à l'URL <https://github.com/getsentry>.

- **Le mécanisme de publication**

Publier un site Web est une opération largement documentée par ailleurs. La publication d'un site Web mobile ne présente pas de particularité notable.

7 — LE BACK-END

Les applications mobiles sont rarement autistes.

Elles échangent avec un ou plusieurs serveur tiers les informations montantes et descendantes.

Ce chapitre n'entrera pas dans l'étude du tiers « back-end » mais dans les moyens de communications avec celui-ci :

- › formats d'échanges,
- › protocoles de communication,
- › problématiques fréquentes et solutions.

7.1. Les protocoles de communication

Si, dans l'univers back-end, les protocoles de communication sont multiples et associés à des besoins particuliers (FTP, SSH, HTTP, DNS, SMTP, Telnet, etc.), l'univers mobile est focalisé sur l'un d'entre eux.

Le protocole de base utilisé est HTTP (HTTPS est recommandé) ; FTP et SMTP sont utilisables assez facilement (il existe des bibliothèques pour iOS, Android et Windows 10) ; GitHub fournit des accès à de nombreuses solutions permettant d'utiliser les autres protocoles.

De fait, la prépondérance absolue du protocole HTTP vient de son usage dans les principales philosophies d'échange de données : les WebServices. Concept précisé et mis en œuvre par le W3C, il a été enrichi par le modèle REST.

- › Les Web Services de type REST (pour Representational State Transfer) exposent les fonctions d'accès à une ressource par des URI ; les actions sur les ressources sont déterminées par la syntaxe et la sémantique du protocole HTTP. Ainsi GET pour obtenir une

ressource, DELETE pour la supprimer, etc...

› Les Web Services SOAP (ou WS-*) utilisent le format SOAP pour l'échange de messages, le langage WSDL pour leur description. Ils reposent également sur l'appel d'URL via HTTP, mais avec une approche de type RPC.

Dans le cadre des applications mobiles, les Web Services utilisés sont principalement de type REST ou tendent vers ce modèle. On peut parfois avoir besoin d'accéder à des WebServices SOAP dans le cas d'API légataires. Une API back-end mise à disposition pour une (ou plusieurs) application mobile est à l'heure actuelle exposée au format REST. Cette approche a également été privilégiée par les principaux acteurs du Web comme Google, Amazon, Facebook, Apple, Microsoft... pour mettre à disposition les API d'accès à leurs services.

7.2. Les formats des données

Les données sont envoyées principalement sous deux formats : JSON et XML. Ces formats ont l'avantage de présenter les données sous forme lisible à l'humain, et d'apporter une structure avec ces données. D'autres formats structurés moins répandus existent : BSON, MessagePack (<http://msgpack.org>).

Les Web Services REST peuvent fournir JSON et XML ; les Web Services SOAP également (même si c'est contraire à la spécification) mais le formalisme et les contraintes formelles plus importantes font qu'en pratique, les Web Services SOAP fournissent leurs réponses au format XML.

Lorsqu'une API fournit les deux formats, une application cliente peut spécifier lequel elle demande en envoyant dans la requête HTTP un header de type

```
Accept : application/json ou application/xml
```


Les deux formats étant généralement disponibles, quel est le choix judicieux ?

Dans le cadre strict d'un transfert de données d'un serveur vers une application mobile, les éléments différenciateurs du XML peuvent être mis de côté. Pour rappel, XML n'est pas un simple format de données, mais un langage complet, qui intègre :

- › XPath, un langage de requêtage,
- › les attributs et les Namespaces, permettant de mieux organiser et structurer l'information,
- › les schémas XML, permettant de valider strictement le format d'un document XML,
- › le langage XSLT, permettant de transformer un document XML.

Une fois cette richesse écartée, si l'on considère uniquement leur utilité pour porter les données, XML et JSON sont assez proches.

La syntaxe de JSON est simple, mais ne supporte que quelques types de données. JSON est concis, XML est plus bavard.

Les deux formats proposent des formules de validation formelle : JSON Schema et XML Schema.

Enfin, les deux formats sont bien supportés par les bibliothèques des différents langages de programmation.

Les quelques avantages pour JSON sont :

- › Le JSON est immédiatement transformable en objet JavaScript, mais cet avantage n'a de validité que pour une application hybride ou un site web mobile.
- › Le JSON est moins bavard et le volume du flux transmis est plus faible pour une même quantité d'information utile.

De ce fait, JSON est devenu le standard, mais d'autres solutions peuvent être prises en considération pour des besoins spécifiques, dont BSON ou MessagePack qui proposent une sérialisation très compacte.

7.3. Problématiques fréquentes et pistes de solution

- **Les évolutions des API ou la problématique du versioning**

Les API d'échange avec le back-end posent un contrat avec les applications mobiles qui y accèdent : il s'agit de l'ensemble des signatures des différents services.

Cette signature comprend le protocole d'accès, le nom du serveur et le nom de domaine, le port d'accès, et le chemin d'accès à la ressource, mais aussi le format et la structuration des données retournées.

Tout logiciel étant amené à évoluer, il est nécessaire de prévoir tôt le fonctionnement du back-end qui sera amené à servir des applications clientes qui seront dans différentes versions alors que l'API elle-même aura pu évoluer de manière plus ou moins corrélée.

Dans le cadre d'une entreprise gérant ses terminaux avec un MDM et maîtrisant le calendrier de déploiement des versions clients et back-end, le déploiement de toute nouvelle API peut s'accompagner du déploiement de l'application cliente adaptée.

Sinon, il est nécessaire de mettre en place un double mécanisme :

- › de publication de la version attendue de l'API par l'application mobile,
- › d'adaptation de l'API à la version attendue.

- **La publication de la version attendue par l'application mobile**

L'application mobile est développée en suivant un contrat d'interfaces avec l'API. Ce contrat correspond à une version.

Afin d'indiquer cette version au back-end, plusieurs pistes sont possibles :

- › Par inclusion du numéro de version dans l'URL, comme par exemple `http://be.ippon.fr/api/v2/blog/article/12`
- › Par création d'un request header HTTP spécifique, comme « `api-version:2` »
- › Par modification du Accept header de la requête HTTP pour qu'il inclue la version, comme dans « `Accept:application/be.v2+json` »

La modification de l'URL est une mauvaise solution car une URL doit, dans le respect de la forme REST, représenter l'entité. Aussi l'inclusion de la version introduit-elle une confusion : est-ce la version de l'entité qui change ?

La création d'un request header spécifique est une mauvaise solution car elle conduit à l'introduction d'un élément extérieur à la spécification HTTP alors qu'un élément permettant de répondre au besoin existe.

La modification de l'Accept-header complique le test de l'API car il devient impossible de la tester par copier/coller ou par lien href dans un navigateur basique. Cet argument est très léger car il est facile de compléter les navigateurs par des plug-ins facilitant le test de requêtes HTTP (Advanced REST Client, PostMan, ou autres).

Cependant en complément de la forme (et de la philosophie) REST, il faut également prendre en compte le principe KISS (Keep it short and simple) : l'objectif est d'avoir un contrat d'API stable, compatibles avec de bonnes testabilité, maintenabilité et évolutivité.

Par maintenabilité il faut comprendre la possibilité de corriger des anomalies dans une version d'API sans modifier le contrat d'interface (on reste en v2, mais on corrige la v2), alors que l'évolutivité désigne la facilité à ajouter des points d'accès ou à modifier la signature des points d'accès à l'API.

La testabilité indique la facilité d'un système à réaliser des tests probants. Elle est en relation directe avec la validation de non-régression sur les évolutions, la correction des anomalies en maintenance : elle est maximisée si on utilise la solution de la modification de l'URL.

- **Les tests d'API**

Une API doit être testée avec autant de rigueur que tout autre tiers logiciel. La bonne pratique est d'automatiser ces tests afin de détecter au plus tôt toute régression et de vérifier le respect du contrat d'interface.

Cette automatisation peut être réalisée à l'aide d'outils tels que SoapUI, Jenkins et Maven – comme expliqué dans l'article de blog Ippon <http://blog.ippon.fr/2014/10/08/automatiser-les-tests-fonctionnels-dune-api-rest/>.

- **Expérience utilisateur et performance du back-end**

La qualité de l'expérience utilisateur est très dépendante de la fluidité perçue de l'application mobile. Or il arrive fréquemment que les données à présenter à l'utilisateur proviennent du back-end, via un réseau data non maîtrisé. Il faut donc minimiser le temps d'obtention de ces données.

Il existe plusieurs stratégies côté application mobile :

- › Le chargement à la volée où l'application va chercher les données au moment où elles sont nécessaires. Ce chargement est bloquant.
- › Le chargement masqué des données où l'application va chercher les données avant d'en avoir besoin. Ce chargement est réalisé en tâche de fond et non bloquant.

De plus, les données peuvent être chargées par delta ou par « annule et remplace ».

Cependant la complexité de développement diffère selon les solutions appliquées. Par exemple, une application présente une liste de messages échangés entre un organisme et un utilisateur ; les échanges peuvent être menés depuis différents canaux donc il est nécessaire de synchroniser la liste des messages sur le device mobile en cours d'utilisation.

Chargement		Avantages	Inconvénients
Partiel (mise à jour)	A la volée	Temps de transfert limité / Faible volume échangé	Latence perçue par l'utilisateur
	Masqué	Temps de transfert non perçu / Faible volume échangé	Consommation de bande passante et de données potentiellement inutile / Développement complexe
Complet	A la volée	Développement simple	Temps de transfert important / Expérience utilisateur dégradée / Consommation de data importante
	Masqué	Temps de transfert non perçu	Consommation de bande passante et de données potentiellement inutile et volumineux

De ce tableau il ressort que la solution la plus simple techniquement (tout télécharger à la volée) est aussi la plus dégradante pour l'expérience utilisateur et son forfait data. A l'opposé, la solution la plus complexe techniquement (télécharger uniquement les données nécessaires de manière masquée) est aussi celle qui permet de maximiser l'expérience utilisateur. De plus, il faut prendre en compte la population cible de l'application mobile : majoritairement urbaine et équipée en 4G ou potentiellement tributaire d'une connexion réseau à la qualité aléatoire ?

La solution employée sera donc dépendante des contraintes sur l'expérience utilisateur, du budget, du temps disponible, et des possibilités proposées par le back-end.

7.4. Sécurité applicative

Les problématiques de sécurité dans une application mobile sont liées :

- › au stockage des données sur le device,
 - › au transfert des données entre le terminal et le serveur back-end.
- **La sécurité des échanges sera mieux assurée si les techniques suivantes sont mises en place :**

- › Appel des URLs sur le protocole HTTPS – sur la base d’un certificat délivré par une autorité de certification valide. Sinon, il est possible d’obtenir des certificats auto-signés et gratuits par une variété d’outils tels que OpenSSL, l’outil keytool de Java, Keychain d’Apple, et aussi par le site Web <https://letsencrypt.org>. Let’s encrypt est une solution poussée par des acteurs majeurs (l’Electronic Frontier Foundation, le Mozilla Foundation, l’Université du Michigan) pour promouvoir et faciliter l’usage du HTTPS dans les échanges de données.

- › Valider l’intégrité des messages - On peut se servir du « Hash-based message authentication code » (HMAC : <https://tools.ietf.org/html/rfc2104>). 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 le 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 – par exemple en fournissant un Hash de certains éléments uniques et d’une clef partagée entre le client et le serveur.

- › 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 ré-utilisables que par le serveur. Par contre cette solution complique la scalabilité horizontale en introduisant cette notion de session (il faut alors prévoir un frontal de répartition des requêtes de type Apache + sticky_session).

- › Sécuriser la sauvegarde des données – il faut éviter de sauvegarder des données très sensibles sur le téléphone. Il existe des APIs d'encryption tant pour Android que pour iOS. Cependant, sauf exploit 0-day, les données propres à une application et enregistrées en tant que telles ne sont pas accessibles aux autres applications.
- › 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 prévu. Les plateformes mobiles fournissent des APIs (Security framework sous iOS, TrustManager sous Android) qui permettent de faire ce genre de validation à partir d'empreintes sur le 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 en utilisant la « WebView » qui permet d'avoir une interface Web au sein même de l'application sans devoir quitter son contexte vers le navigateur 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 de rendre difficile cette forme d'attaque. On peut par exemple définir un seuil (délai) à partir duquel une même requête ne doit plus être servie en se basant sur le timestamp (détaillé plus haut) fourni lors de la requête.



En pratique, un bon niveau de sécurité est atteint pour un coût de développement très limité en utilisant des requêtes HTTPS, c'est à dire en ne mettant en œuvre que la première de ces techniques.

8 — INDUSTRIALISATION DES DÉVELOPPEMENTS

En complément des environnements individuels de développement, une équipe peut souhaiter s'équiper d'une plate-forme d'intégration, tout comme pour un développement de projet en dehors du monde de la mobilité.

Les outils les plus fréquemment rencontrés sont Jenkins, Gitlab et Teamcity. Chacun de ces outils a ses forces propres. Tous permettent le build des projets Android, iOS et .Net.

Pour chacun d'eux, la notion de travailleur distant en charge des builds permet de préparer des environnements de travail dédiés à l'une ou l'autre des plates-formes mobiles.

 Jenkins	Jenkins a pour force principale une facilité d'intégration de plugins nombreux et une grande simplicité d'utilisation apparente – la mise en œuvre d'une plate-forme par une équipe de développement peut être envisagée même pour des équipes ayant peu de compétences orientées système. Malgré sa simplicité, cet outil est riche de fonctionnalités via ses plugins multiples : les tâches de builds peuvent être conduites à des niveaux élevés de complexité.
 TeamCity	TeamCity est un outil bien conçu, à l'interface sobre, explicite et dont les fonctionnalités sont exclusivement dédiées à l'intégration continue. Il assiste considérablement le paramétrage des builds : ainsi son analyse des fichiers présents dans l'outil de gestion de configuration pour proposer des choix, ou sa validation de la présence des pré-requis sur les machines hébergeant les Runners. Sa force réside dans son intégration avec Android Studio – ou IntelliJ IDEA du même éditeur.



Gitlab est un outil de gestion de sources intégrant un serveur d'intégration continue. Sa particularité est que les builds sont configurés dans un fichier intégré au sein du projet, et donc potentiellement éditable par l'équipe de développement. Ce choix est très intéressant car il permet d'intégrer une démarche devops au sein de l'équipe : responsable du paramétrage des builds, elle intègre mieux les contraintes associées et peut aussi faire évoluer le projet, les rapports fournis, les phases préalables au build, etc...

En conclusion, il n'y a pas un bon choix unique mais plusieurs possibilités en fonction de la taille des équipes (et des projets) et du degré d'agilité souhaité.

› Pour des équipes ou des projets de petite taille, Gitlab est une solution à privilégier : autonomie et responsabilisation de l'équipe, intégration de la démarche Devops, liberté quant au paramétrage des builds rendent cette solution idéale.

- L'intégration avec un outil de gestion des sources simplifie la mise en œuvre de la forge logicielle.
- Un paramètre important à prendre en considération est l'existence d'une version EE de Gitlab, permettant un déploiement sécurisé par un support professionnel si nécessaire.

› Pour des équipes ou des projets d'envergure, la richesse fonctionnelle de Jenkins, la centralisation du paramétrage des builds, sa possibilité d'intégration d'outils externes (SonarQube, Checkstyle) via les plugins en font une solution plus industrielle – au détriment de l'agilité et de l'approche devops - et plus en mesure de fournir métriques et KPI utiles au pilotage.

Ce choix est plus cohérent avec la présence d'un administrateur en charge d'une Forge Logicielle, comprenant en plus de Jenkins un outil de gestion de sources, un outil de Tracking, etc...

9 — STATUT LÉGAL D'ÉDITEUR

Une application mobile n'est pas un site Web, elle n'est pas hébergée comme un site Web.

Le statut d'hébergeur n'est donc pas important, dans les CGU ou CGV c'est surtout l'éditeur qui faut indiquer car c'est lui qui porte la responsabilité de l'application.

L'hébergeur est Google pour l'application Android et Apple pour l'application iOS. « Les intermédiaires techniques comme Google ou Apple entrent dans la catégorie légale **des hébergeurs de l'application**, mais d'**éditeurs des systèmes d'exploitation** des smartphones.

Adresse de Google :

**Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA**

Adresse d'Apple :

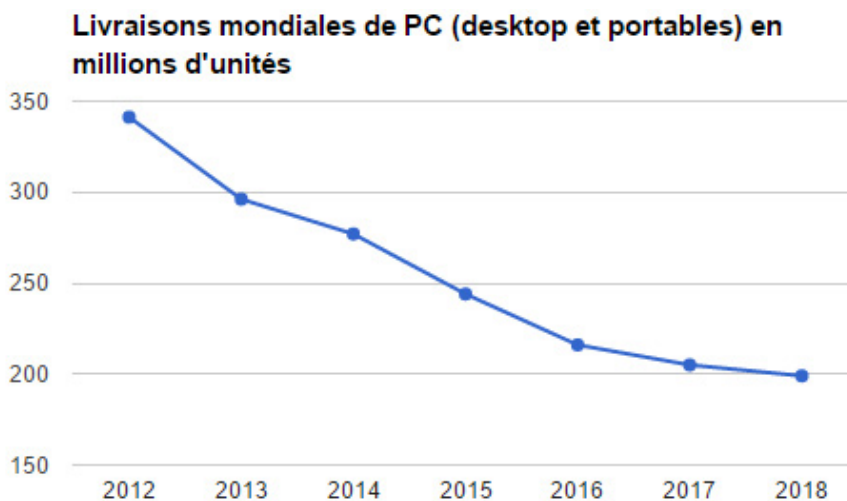
**Apple Inc.
1 Infinite Loop, Cupertino, CA 95014, USA**

**Adresse de Microsoft :
Microsoft Corporation
One Microsoft Way Redmond,
WA 98052-7329, USA**

10 — PERSPECTIVES À COURT TERME

10.1. Quelques faits pour commencer

› Les ventes des PCs traditionnels sont en baisse régulière : de plus en plus et en particulier dans les marchés émergents, les tablettes et smartphones sont les relais de croissance.



Source Gartner - via ZDNet.fr/chiffres-cles

LES VENTES D'ORDINATEUR DEPUIS 2012

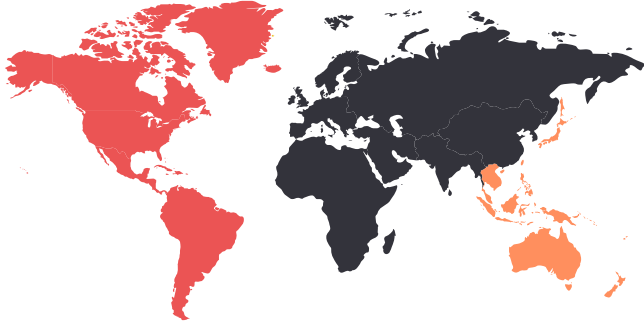
› La puissance de calcul des appareils mobiles est sans cesse croissante. De ce fait, des applications de plus en plus riches et complexes peuvent en exploiter les capacités. Ainsi, des applications comme LightRoom ou Snapseed de traitement de photos y sont aujourd'hui à l'aise.

› Des domaines entièrement nouveaux s'ouvrent encore :

- Par le biais des casques de réalité virtuelle faits pour les smartphones (comme le Samsung Gear VR). Aujourd'hui il n'existe pas d'application à usage professionnel dans ce type d'environnement, mais les solutions logicielles existent pour en créer, comme Unity3D. Microsoft en particulier, avec la dernière version de Windows 10, commence à défricher le terrain de la 3D.
- Gestion des objets connectés, domotique ou autre : le domaine de l'IoT est encore jeune !

10.2. A partir de ces quelques faits

- › La dynamique des appareils mobiles ne sera pas enrayerée.
- › Il n'est plus envisageable d'avoir des sites Web inadaptés à une consultation sur appareil mobile (50,7 % des Français de 15 ans ou plus favorisent la connexion sur mobile, contre 46,9 % sur ordinateur - Source Mediamétrie).
- › Toutes les applications "métier" ne sont pas transposables au monde mobile, mais la réflexion sur l'opportunité de leur adaptation au monde mobile est d'actualité. Cela implique d'ailleurs de les adapter au mode déconnecté.
- › Certes, il existe déjà plus de 2.000.000 d'applications mobiles, mais la 2.000.001ème peut encore être révolutionnaire !



PARIS
BORDEAUX
NANTES
LYON

RICHMOND, VA
WASHINGTON, DC
NEW-YORK

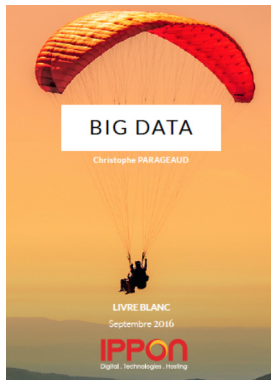
MELBOURNE
MARRAKECH

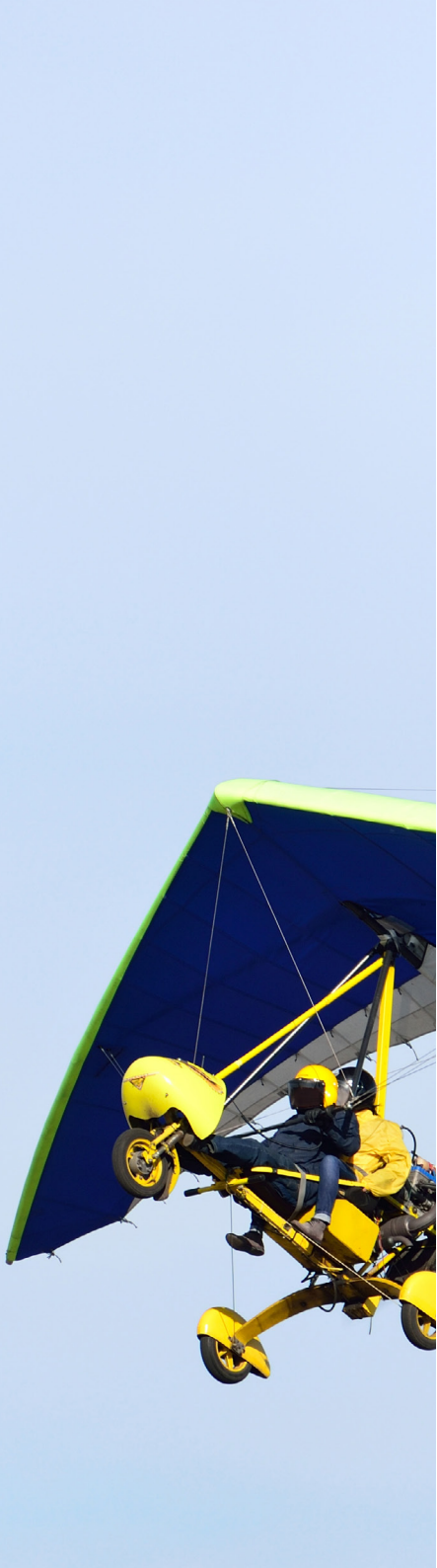
www.ippon.fr/contact
talents@ippon.fr
blog.ippon.fr

+33 1 46 12 48 48
@ippontech

Découvrez aussi nos autres ressources

sur <http://www.ippon.fr/ressources/>





IPPON
Digital . Technologies . Hosting

www.ippon.fr

blog.ippon.fr

Paris
Nantes
Bordeaux
Lyon
Washington, DC
Richmond, VA
New York, NY
Melbourne
Marrakech