Mobizel
Think Mobile First

Quel choix technologique pour le développement de mon **application mobile ?**

Livre blanc

Sommaire

Introduction p.2

Panorama des interfaces numériques p.4
Interfaces accessibles depuis un navigateur web p.6
Interfaces accessibles depuis les stores p.8

Partie 1

Quelle interface choisir pour mon projet d'application ? p.10

Webapp ou application mobile ? p.12
Application mobile : native ou cross-plateforme ? p.14
Quelle technologie cross-plateforme ? p.16

Partie 2

Le développement d'une application p.20

Développement d'une application mobile native p.22
Développement d'une application native générée p.24
Développement d'une application mobile hybride p.28
Développement d'une application web native p.30
Flutter : une nouvelle approche p.32

Introduction

Webapp, site mobile, responsive design, cross-platform, application hybride ou native… On peut facilement se perdre parmi ces notions. Langages web et mobiles, des technologies en constante évolution, une multitude de plateformes de développement technique… Il est difficile d'y voir clair.

Entre un site web et une application mobile, la différence est simple. Mais lorsque l'on creuse un peu plus loin… quelle est la différence entre un site mobile et une webapp ? Qu'est-ce qui différencie une application hybride d'une application native ?

Et quelles sont les technologies permettant du cross-platforme ?

Et surtout dans tout ça, qu'est ce qui convient le mieux à mon projet, mon budget, mes contraintes et mes objectifs ?

© Photo : Sebastiano Giuseppe Garilli via unsplash.com

Panorama des interfaces numériques

Site web

Accessible depuis un navigateur

Développement web

HTML, CSS, JavaScript,
PHP, etc.

Site web

Contenu standard, consultatif.

Ex. : blog, site institutionnel, e-commerce, etc.

Webapp

Contenu interfactif.

La webapp ou la PWA est conçue comme un logiciel de bureau ou une application.

Ex. : Google Drive, Pixlr

Application mobile

Accessible depuis un magasin d'applications (store)

Développement spécifique à un système d'exploitation mobile

iOS, Android, etc.

Application native

Réalisée à partir de langage natif.

Technologies utilisables :
- Objective C ou Swift pour iOS
- Java ou Kotlin sur Android

etc.

Développement Cross-Platforme

Génération d'applications sur plusieurs systèmes d'exploitations à partir d'un socle de développement commun.

Application native générée

Un socle commun est écrit dans un langage tel que C#, Ruby, JavaScript, etc.

Technologies utilisables :
- Xamarin
- Titanium
- Ruby Motion

etc.

Application hybride

Le contenu web (HTML, CSS, JavaScript...) est «encapsulé» dans une sur-couche applicative.

Technologies utilisables :
- PhoneGap
- Apache Cordova
- Ionic

etc.

Application web native

À partir de langages web (JavaScript, TypeScript, Vue JS, Angular...), elle s'appuie aussi sur les technologies natives mobiles.

Technologies utilisables :
- NativeScript
- React Native

Interfaces accessibles depuis un navigateur web

Depuis un navigateur web, vous pouvez accéder aux sites responsive ou mobile et aux webapp. Ces interfaces sont développées en HTML, CSS, JavaScript, PHP, etc ; et ne fonctionnent qu'après une première connexion internet. Certaines fonctionnalités pourront être accessibles hors connexion.

Webapp

Il s'agit d'un logiciel applicatif pouvant être exécuté soit :

- depuis un navigateur web (webapp desktop). Ils se présentent comme un logiciel de bureau.

  Exemple
  Les messageries web ou Pixlr

- depuis un smartphone (webapp mobile)

Site

Il existe 2 types de site :

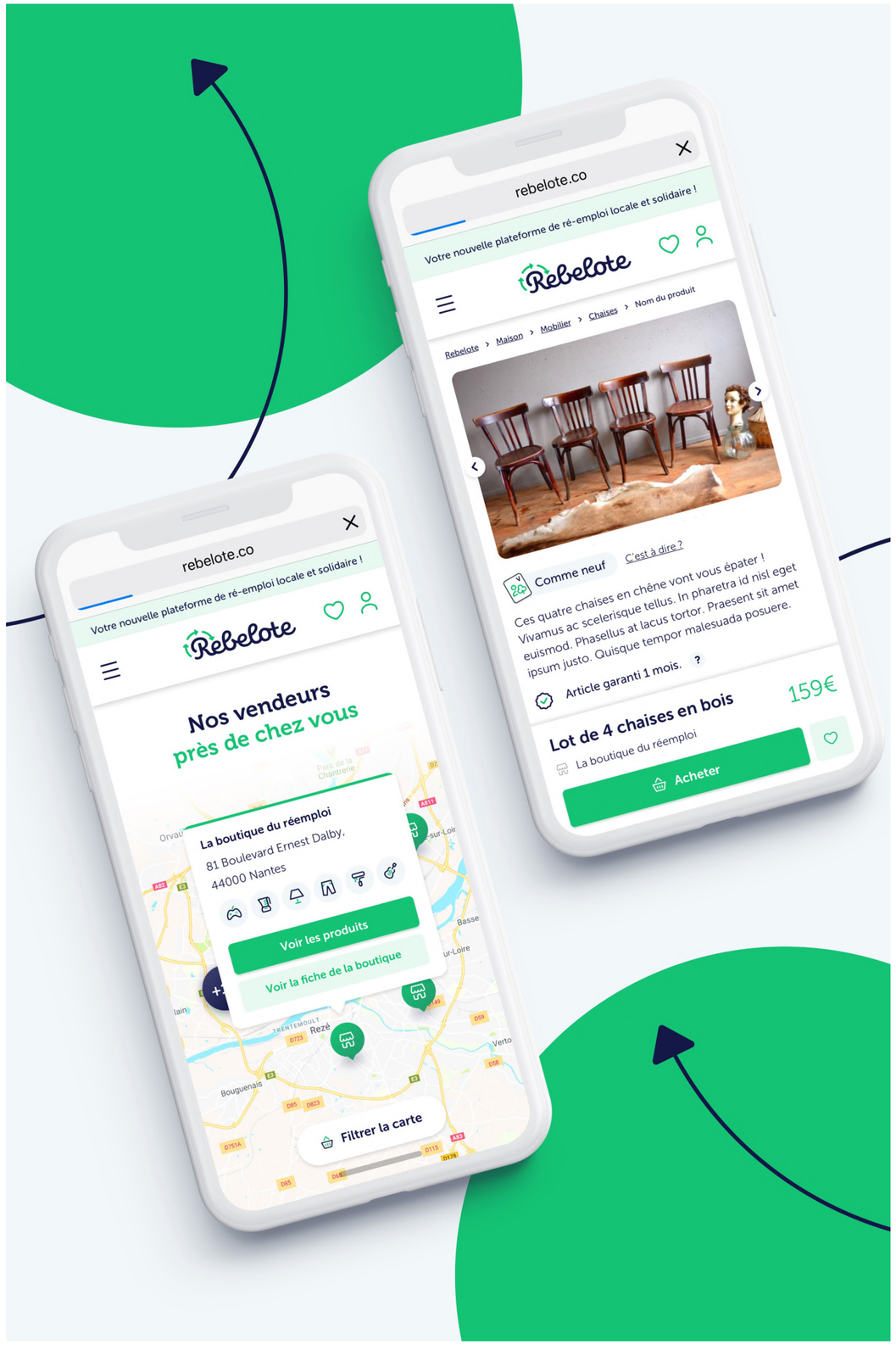
- **Mobile**

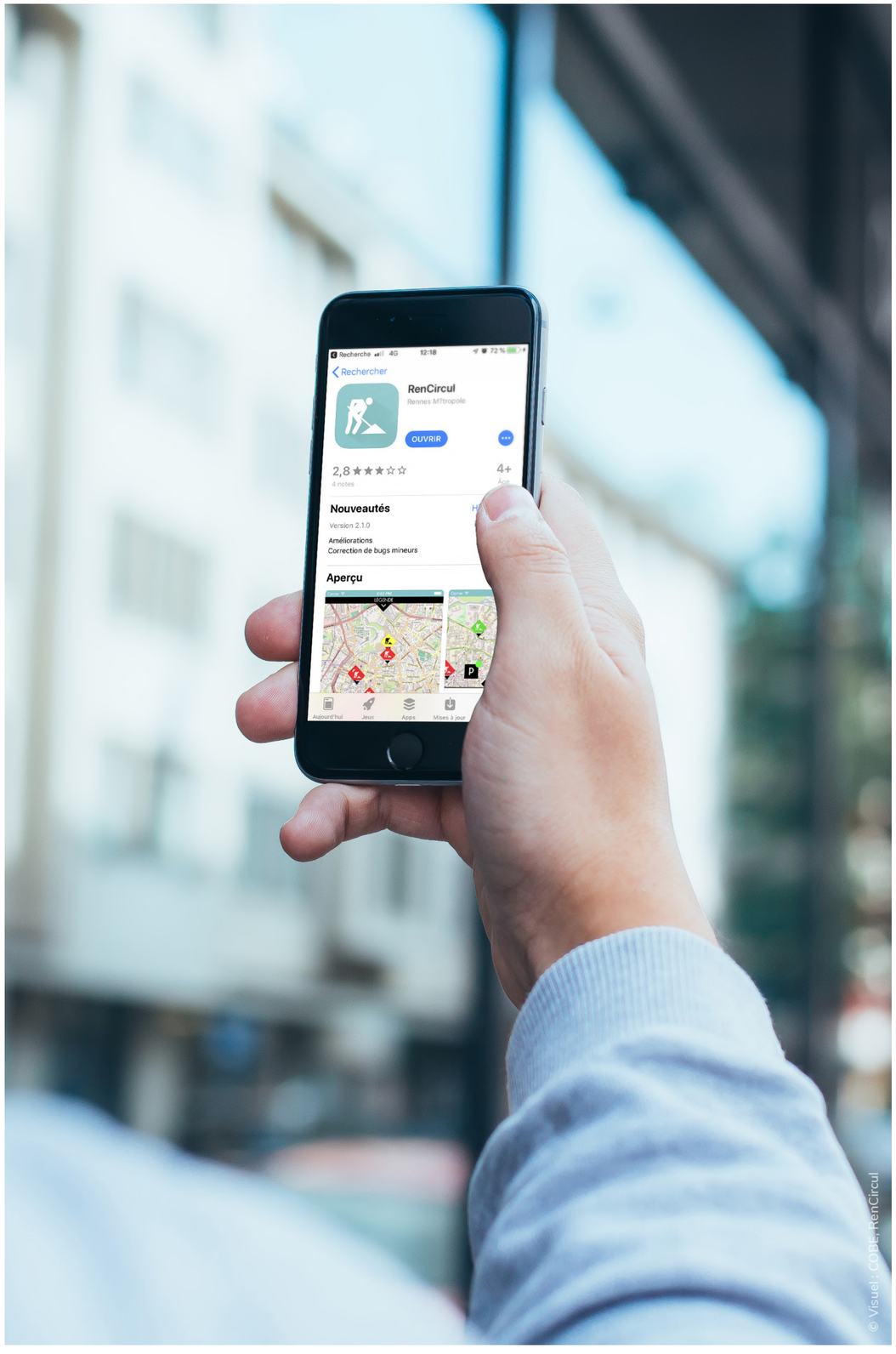
  C'est une interface accessible depuis un navigateur via une URL et optimisée pour une consultation mobile : hiérarchie du site, ergonomie orientée tactile, médias adaptés à l'écran, etc. L'URL d'un site mobile commence généralement par « m. ». Le site mobile tend à disparaître depuis l'arrivée du responsive design.

  Exemple
  m.starbusmetro.fr

- **Responsive Design**

  Il utilise une technique de développement qui permet la réorganisation des contenus d'un site web en fonction du terminal utilisé (desktop, tablette ou smartphone). Pour reconnaître facilement un site responsive design, réduisez la taille d'une page web depuis votre ordinateur et constatez les contenus qui s'adaptent à la taille de la fenêtre.





Interfaces accessibles depuis les stores

Pour obtenir ce type d'applications, smartphone ou tablette, il faut les télécharger depuis un magasin d'applications et les installer sur son mobile comme pour un logiciel.

Il existe deux façons de développer une application mobile :

Développement natif

Le développement d'un applicative dite native est spécifique au système d'exploitation souhaité : pour développer une application iOS il faudra utiliser le langage Objective C ou Swift, pour Android Java ou Kotlin.

Développement cross-platforme

C'est à dire un développement qui va « générer » plusieurs applications mobiles en même temps pour des systèmes d'exploitations différents. Il existe diverses méthodes de développement cross-plateforme :

- des **applications natives générées** si vous utilisez des technologies telles que Xamarin, Titanium, Ruby Motion ou Qtmobile. Celles-ci vont vous permettre de concevoir un socle commun à vos applications avec du langage C#, Ruby, JavaScript, etc.
- des **applications dites « hybrides »**. Elles sont basées sur un contenu web commun qui est ensuite « encapsulé » dans une sur-couche adaptée aux différents systèmes d'exploitations souhaités. Pour cela, vous pouvez utiliser des technologies telles que PhoneGap, Rho Mobile ou Apache Cordova.
- Les **applications web natives** telles quel NativeScrpt ou React native. Elles sont développée à partir de langages web (JavaScript, TypeScript, Vue JS, Angular...) en utilisant des composants natifs.
- Flutter, le langage de Google ayant un fonctionnement spécifique que nous décrirons plus tard.

Partie 1

Quelle interface choisir pour mon projet d'application ?

Choisir la technologie de développement de son projet d'application mobile est une étape importante. Il faut non seulement évaluer ses contraintes et ses besoins immédiats, mais également se projeter et anticiper les évolutions dont vous pourriez avoir besoin à l'avenir. Cependant, il ne s'agit pas seulement d'un choix technique, mais surtout d'un périmètre fonctionnel et de l'expérience utilisateur qui en découlera.

© Photo : Annie Spratt via unsplash.com

Webapp ou application mobile ?

Définitions

Web app ou application web

C'est un logiciel applicatif hébergé sur un serveur et accessible depuis un navigateur Internet. Techniquement, il s'agit donc d'un site web disposant d'une ergonomie et de fonctionnalités spécifiques adaptées aux contraintes des terminaux mobiles. La webapp peut intégrer toutes les fonctionnalités disponibles du navigateur. Son principal avantage est sa compatibilité multiplateforme. Hébergée sur un serveur, elle peut afficher son contenu sur plusieurs systèmes d'exploitation et plusieurs navigateurs.

> **Et la PWA, qu'est-ce que c'est ?**
>
> La PWA, ou Progressive Web App, est tout simplement une forme de webapp profitant des dernières technologies afin d'améliorer l'expérience utilisateur (rapidité, hors-connexion...).

Application mobile

Une application mobile c'est un logiciel développé spécifiquement pour un système d'exploitation mobile (iOS, Android). Elle doit être téléchargée depuis un magasin d'applications mobiles (store) puis installée sur le périphérique. L'application mobile peut fonctionner hors-ligne et se baser sur toutes les fonctionnalités natives du terminal (appareil photo, bluetooth, répertoire...).

Quels avantages et inconvénients ?

Tableau comparatif des avantages et contraintes de la webapp et de l'application mobile.

	Webapp	Application mobile
Compatibilité multiplateforme	+ Compatibilité avec tous les terminaux − Les fonctionnalités disponibles dépendent du navigateur utilisé	− Une application par plateforme (iOS, Android), même si le code peut en partie être mutualisé grâce à un langage cross-plateforme
Ergonomie et graphisme	− Le graphisme est le même pour tous les OS − Ergonomie limitée : le rendu graphique est dépendant du navigateur	+ Sur-mesure et donc adaptés aux usages mobile de chaque OS
Fluidité et performances	− Dépend de la qualité du réseau et du navigateur utilisé. Possibilité de temps de chargement plus longs, lags...	+ Puissance, qualité et fluidité optimales
Fonctionnalités	− N'accède pas à l'ensemble des fonctionnalités native du téléphone − Expérience utilisateur inégale en fonction de l'OS et du navigateur	+ Accès à 100% des fonctionnalités natives du téléphone
Distribution	+ Accessible dès la mise en ligne	− Doit être soumises au stores − 24 à 72h de temps de validation
Monétisation	− Impossible de vendre le téléchargement de l'app + Vente en ligne + Monétisation par la publicité	− Commission des stores + Possibilité de faire payer le téléchargement + Achat intégré à l'application via le store (achat in-app) + Monétisation par la publicité
Référencement	+ SEO : optimisations possibles pour se référencer sur les moteurs de recherche	− ASO : optimisation limitées sur le store + SEO : possibilité de créer une landing page sur son site pour renvoyer vers le store
Mise à jour du contenu	+ Simple et rapide	+ Possible via un back-office − Nécessite une re-soumission sans back office
Évolution et maintenance	+ Peu coûteuse	+ À réaliser pour chaque OS (sauf s'il s'agit d'une application cross-platefome) − Re-soumission obligatoire

Dans quels cas choisir...
la webapp ?

Si vous privilégiez :

- la compatibilité multiplateforme,
- la rapidité de réalisation,
- la simplicité du langage technique et donc d'un prix de production plus accessible,
- l'accessibilité au plus grand nombre en phase d'acquisition d'utilisateur.

Que vous acceptez :

- d'abandonner certaines fonctionnalités (au moins dans le cadre de la 1ère version),
- d'avoir une expérience utilisateur mobile moins personnalisée.

Et que votre projet a pour objectif :

- de tester votre produit sur toutes les plateformes à moindre coût,
- de proposer une application à des utilisateurs qui ne sont pas encore en phase de fidélisation.

Dans quels cas choisir...
l'application mobile ?

Si vous privilégiez :

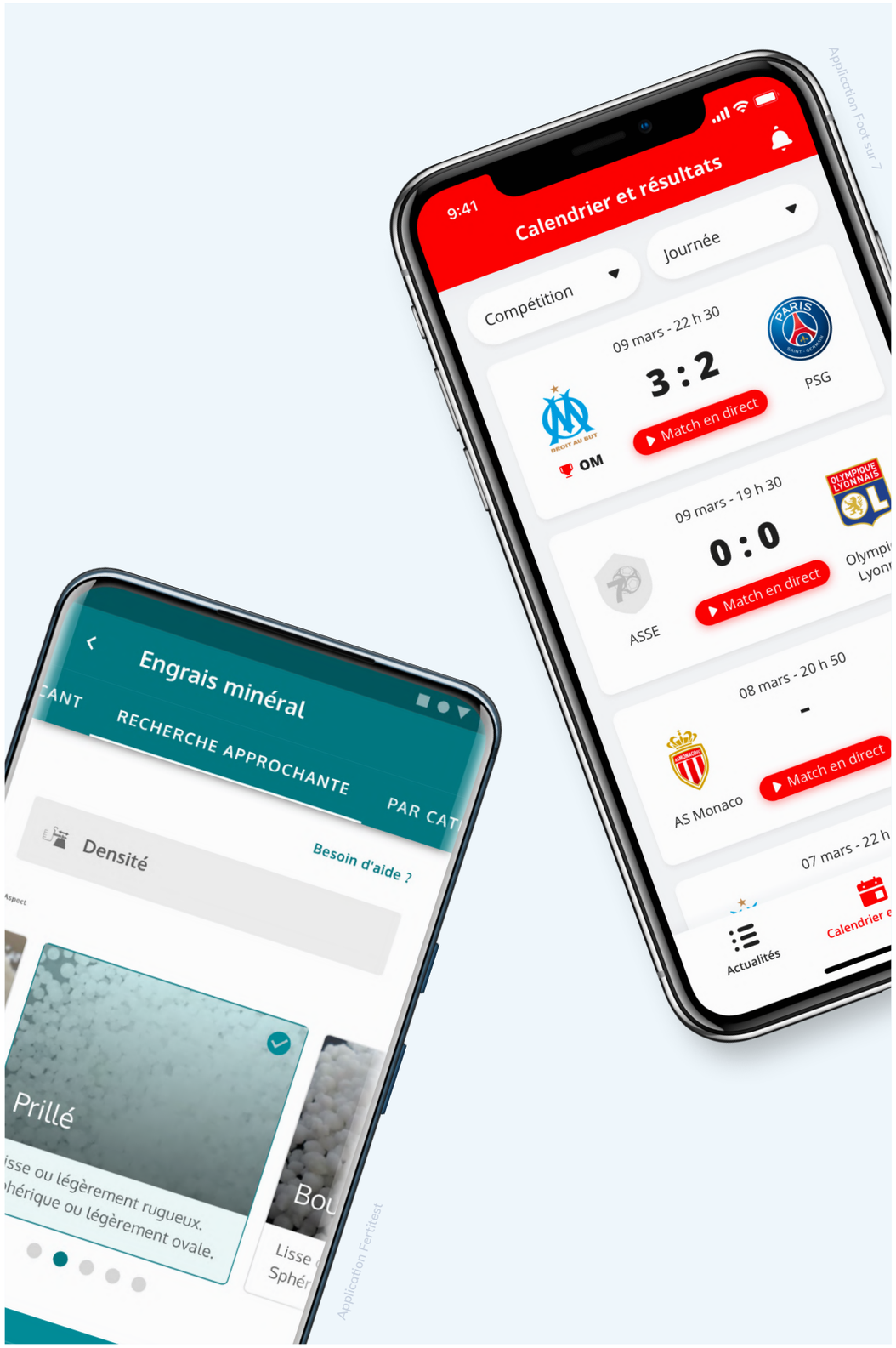
- une expérience utilisateur optimale car adaptée aux usages de chaque OS,
- la liberté dans les fonctionnalités.

Que vous acceptez :

- un développement technique plus long, plus complexe et donc plus coûteux,
- un coût d'acquisition de l'utilisateur plus important.

Et que votre projet a :

- pour objectif de retenir et fidéliser vos utilisateurs,
- besoin de certaines fonctionnalités indispensables qui ne sont pas disponibles via le web.



Application mobile : native ou cross-plateforme ?

Vous avez désormais validé pour votre projet que l'application mobile était plus appropriée qu'une webapp. Mais le choix n'est pas encore fini ! Il y a plusieurs types de développement d'application : le développement natif et le développement cross-plateforme.

La recommandation numéro un des plateformes (Apple, Google, et avant Microsoft ou BlackBerry…) a toujours été d'utiliser leurs solutions propres, c'est à dire leur langage de programmation, leur SDK, leurs outils de développement… En suivant régulièrement les évènements majeurs pour développeurs mobiles (Apple WWDC, Google I/O), on voit bien que le message ne change pas, et c'est aussi notre recommandation.

« Si vous en avez la possibilité, le développement natif est toujours à privilégier car il sera le plus pérenne »

Ces solutions propres à chaque OS sont bien sûr incompatibles entre elles, ce qui nécessite un développement spécifique pour iOS et un autre pour Android, et précédemment un troisième pour Window. Sont apparues progressivement différentes solutions – appelées le plus souvent « hybrides » ou parfois « cross-plateformes » – pour éviter de multiplier les développements plateforme par plateforme (comme PhoneGap, Ionic, Titanium, Xamarin, NativeScript, ReactNative…). Pendant longtemps, toutes les solutions que nous avons testées ne nous permettaient pas de garantir un résultat de qualité (expérience utilisateur, rapidité, fiabilité, évolutivité…). Donc, si vous souhaitiez une application pour iOS et Android de bonne qualité, nous avons jusqu'à présent toujours conseillé deux développements spécifiques (avec un budget et un délai en conséquence).

Une application native est préférable si :

- vous souhaitez investir dans un développement le plus pérenne possible,
- votre budget vous le permet,
- vous souhaitez une interface ou des fonctionnalités différentes entre iOS et Android,
- votre application peut n'être disponible que pour un seul OS (pour une application interne à l'entreprise par exemple),
- votre application nécessite certaines fonctionnalités comme la connexion à un objet via bluetooth.

L'application cross-plateforme, l'idéal si :

- vous souhaitez réaliser un POC à petit budget,
- votre application est simple techniquement et/ou fonctionnellement.

Quelle technologie cross-plateformes ?

Lorsque l'on choisit de développer une application cross-plateforme, il reste encore un choix à faire : quelle technologie utiliser ? En d'autres termes, est-ce que j'utilise les technologies web ou natives ?

Applications hybride

PhoneGap, Cordova, Sencha, Ionic

Il s'agit de langage HTML et Javascript.

Avantages

Adapté aux équipes ayant principalement des compétences web (mêmes langages)

Inconvénients

Fonctionne sur la base d'un navigateur internet embarqué dans l'application. Parfois très lent et présente souvent de nombreux problèmes de stabilité et de fiabilité avec de nombreuses fonctionnalités avancées du terminal mobile. Lors de la maintenance, le diagnostic est souvent rendu compliqué du fait des surcouches logicielles.

Notre avis

Nous déconseillons fortement ces approches.

Applications natives générées et web natives

Titanium, Xamarin, NativeScript, React Native

À mi-chemin entre le web et le natif, avec Titanium (Appcelerator, depuis 2009) et Xamarin (Microsoft, depuis 2011) pour les plus anciennes, puis récemment NativeScript (Telerik depuis 2014) et React Native (Facebook, depuis 2015).

Avantages

Un seul code pour les 2 plateformes et des performances bien meilleures que les solutions à base de navigateur internet (hybrides).

Inconvénients

Le choix de la technologie est stratégique car il engage le projet sur la durée alors que la stabilité du code et la reprise par un autre développeur peuvent être complexes pour l'évolution du projet.

Notre avis

Bien que ces technologies soient intéressantes, elles manquent de stabilité et de pérennité. Nous ne conseillons pas d'investir sur ces technologies, suite à nos expérimentations et celles d'autres plateformes, comme Airbnb qui a fait marche arrière sur React Native.

Flutter

Flutter est la technologie multi-plateforme annoncée par Google en 2015 et introduite en mai 2017.

Avantages

Un seul code pour les 2 plateformes, des performances égales au développement natif et une adoption très rapide du marché.

Inconvénients

C'est une technologie récente (version 1.0 stable en novembre 2018, 1.2 en mars 2019), sur laquelle nous avons moins de recul que sur les autres technologies.

Notre avis

Flutter offre un rendu et des performances de qualité en proposant un fonctionnement différent des technologies précédentes.

Partie 2

Le développement d'une application

Il existe différentes technologies pour développer une application mobile : les applications natives, les applications cross-plateforme natives ou hybrides. Chacune présente des particularités : langage, temps de développement, performances, fluidité... On vous présente les spécificités techniques ou UX pour comprendre les enjeux des différentes technologies.

© Photo : Andrew Neel via unsplash.com

Développement d'une application mobile native

INFO CHRONO

Existe depuis le début du smartphone
Technologie de référence

Une application mobile native c'est...

...une application qui a été conçue spécifiquement pour un système d'exploitation (OS) avec un langage et une/des technologie(s) appropriés.

Les trois étapes de conception d'une application mobile

1 Rédaction du code source

Dans un premier temps, un développeur rédige le code source de la future application dans un langage approprié.

Dans un deuxième temps, pour développer une application mobile qui exploite les fonctionnalités natives du terminal (géolocalisation, contact, médias, appareil photo...) il faut communiquer avec le SDK natif via le code source. C'est-à-dire programmer des « ponts » qui vont aller chercher des morceaux de codes du SDK du système d'exploitation visé. Ces derniers vont permettre d'appeler des fonctionnalités natives pour les intégrer dans l'application mobile.

Définition SDK
Ensemble d'outils d'aide à la programmation pour concevoir des logiciels, jeux, applications mobiles, etc. pour un terminal et/ou un système d'exploitation spécifique.

2 Compilation du code source

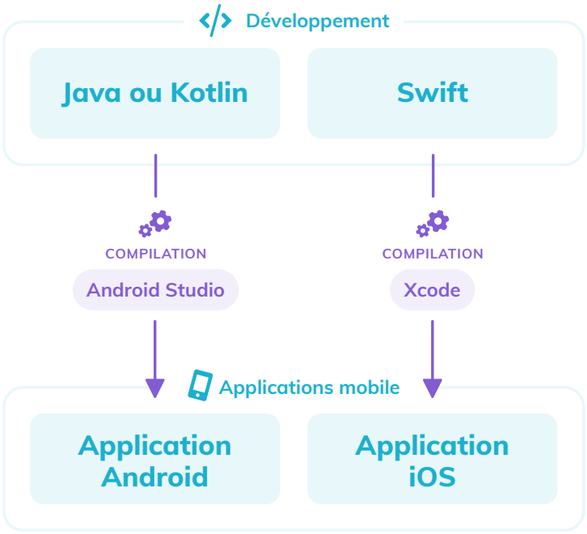
La compilation a un rôle primordial car les logiciels fonctionnent avec du code binaire qui est trop complexe pour être conçu tel quel. Il faut donc rédiger un code plus simple "human readable" (le code source) qui va passer dans un compilateur et être transformé en code binaire "machine readable". Les IDE contiennent un compilateur.

3 Génération d'un code binaire natif

Le compilateur transforme le code source en code binaire et génère un fichier .ipa, .apk ou .aab.

Ce fichier doit être téléchargé (via les magasins d'applications mobiles ou stores), installé puis exécuté sur le terminal pour fonctionner.

En compilant les librairies appelées du SDK natif en même temps que le code source, le code binaire obtenu peut ainsi communiquer avec le système d'exploitation du terminal et exploiter ses fonctionnalités natives.



Récapitulatif des 3 étapes

SYSTÈME D'EXPLOITATION	LANGAGES	IDE	EXTENTION DE FICHIER
iOS	Swift et Objective C	Xcode	.ipa
Android	Java et Kotlin	Android Studio	.apk ou .aab

Définition IDE
Integrated Development Environment (en français « environnement de développement »), est un logiciel qui rassemble un certain nombre d'outils permettant de développer d'autres logiciels.

Notre avis

Les langages du développement natif sont les seuls reconnus officiellement par les stores. Le développement natif est donc préconisé aussi bien par Apple qu'Android. Il garantit également une expérience utilisateur optimale et ne connaît pas de contraintes techniques ou fonctionnelles que pourraient résoudre les autres types de développement.

Dans l'idéal, il est toujours à privilégier.

Développement d'une application native générée

INFO CHRONO

Existe depuis 2009
Technologie en perte de vitesse

Pour concevoir une application mobile native sur des systèmes d'exploitation différents, il n'est pas obligatoire de développer un logiciel pour chaque OS. Vous pouvez utiliser des technologies cross-platform qui permettent (en théorie) de réaliser un code source unique qui sera transformé en plusieurs logiciels adaptés aux OS visés.

Magique ? Pas si simple...

Quel est le fonctionnement générique d'une technologie de développement mobile cross-platform ?

1

Rédaction du code source

La rédaction du code source unique doit tout de même anticiper les spécificités de chaque OS visé et faire appel à tous les SDK nécessaires.

2

Compilation du code source

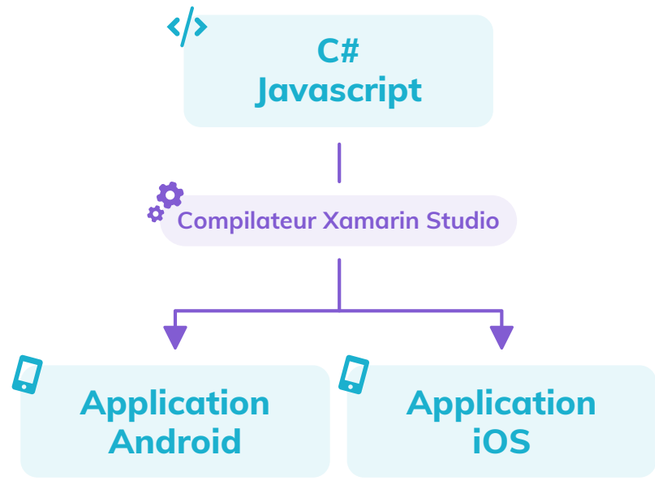
Ce code source est ensuite compilé par la technologie cross-platform choisie. Celle-ci va être capable de reconnaître les parties de votre code source qui doivent être intégrées dans tous les logiciels et celles spécifiques à un OS.

3

Génération d'applications mobiles natives

Après compilation, vous obtenez différents fichiers d'exécution de votre application mobile.

Exemple de technologie cross-platform : Xamarin



1

Rédaction du code source

La technologie Xamarin emploie le langage de programmation C# (dit "C sharp"). Contrairement à Titanium qui permet de réaliser un socle de code commun important, Xamarin nécessite de réaliser en amont d'importantes parties de codes spécifiques à chaque OS. Le travail préliminaire est donc plus important.

2

Compilation du code source

Une fois le code source réalisé, l'IDE de Xamarin (Xamarin Studio) va le compiler et générer deux fichiers exécutables pour les terminaux iOS, et Android.

3

Génération d'applications mobiles natives

L'inconvénient de Xamarin est la rédaction du code source en amont qui est plus longue et plus complexe. En revanche, le code obtenu après compilation est beaucoup plus optimisé car il ne contient que du code binaire, même si celui-ci est tout de même un mélange de code natif et cross-platform.

Cela a l'avantage de générer des logiciels plus performants et plus adaptés aux différents terminaux visés.

Particularités

Avantages

Une seule compétence de développement est nécessaire (maîtrise du langage et des outils appropriés), mais il est presque indispensable de connaître le développement natif de l'un des stores.

Inconvénients

Il existe plusieurs technologies cross-platform telles que Xamarin, Titanium, Qt, etc. qui ont chacune leur propre langage, IDE, avantages et inconvénients…

Elles nécessitent donc des compétences de développement spécifiques ; plus leur structure est complexe et plus le développement et la maintenance des interfaces seront délicats. En revanche, ces technologies peuvent aussi être utilisées pour réaliser une application sur un OS unique.

Notre avis

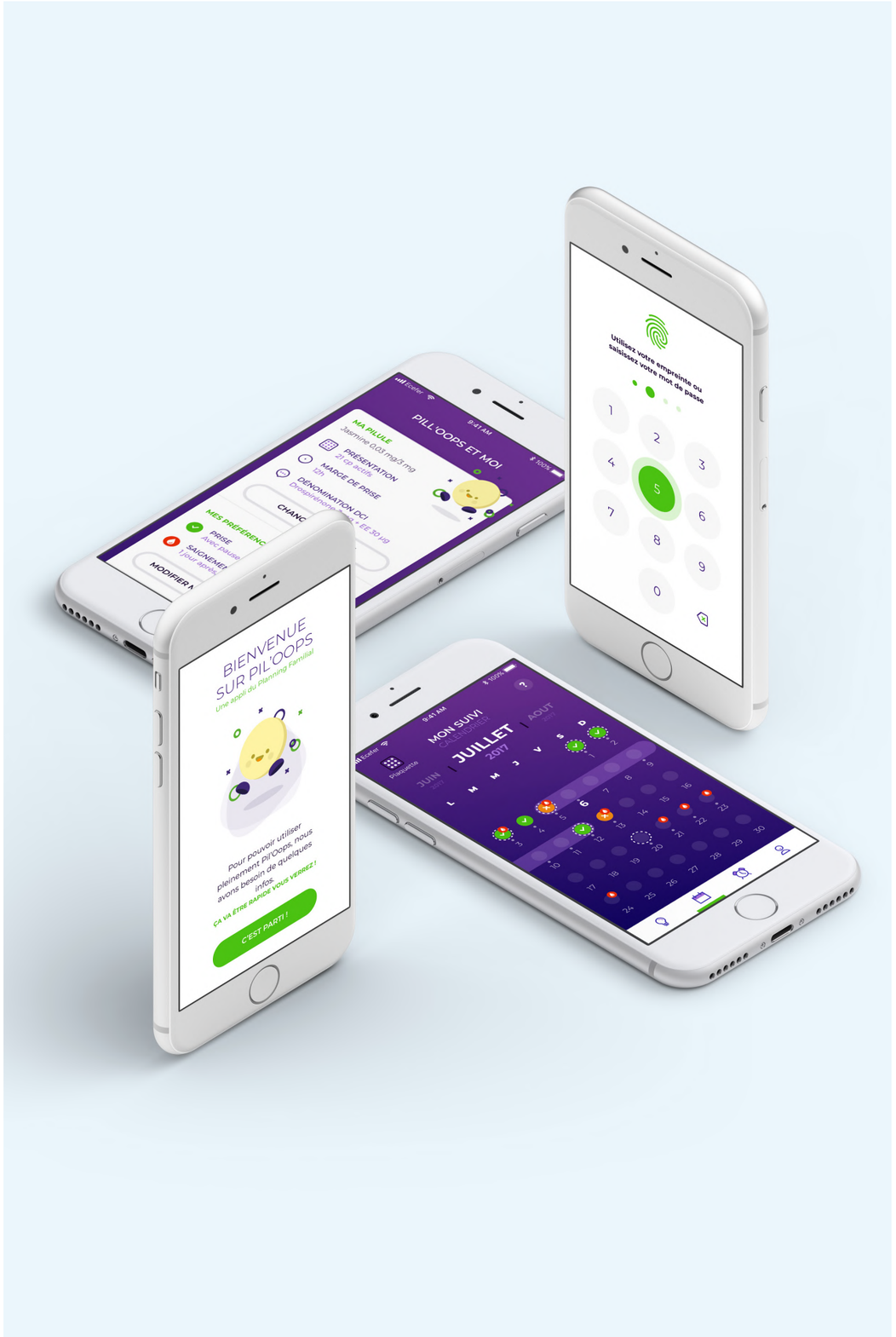
Par définition le développement d'applications natives générées semble être une solution polyvalente, efficace et de qualité, mais en étudiant de plus près le fonctionnement et les différentes étapes de ces technologies cross-platforme, on s'aperçoit qu'il ne s'agit pas d'une solution miracle.

Tout d'abord le développement du code source n'est pas qu'un socle commun, il faut également prévoir des parties de code spécifiques pour s'adapter aux éléments natifs de chaque OS. Cela peut parfois prendre plus de temps que de passer par du développement natif.

Ensuite le diagnostic des bugs et la maintenance sont plus délicats à cause des différentes étapes de compilation, de la structure plus complexe et moins native du code source et de la technologie cross-platforme en elle-même qui laisse des traces dans les codes binaires générés.

Cependant le développement natif généré peut être parfaitement adapté pour un projet mobile :

- qui vise a minima deux systèmes d'exploitation (sachant que cette solution apporte un véritable gain pour trois OS) ;
- qui a besoin de puissance et de performance liées au natif, que des langages web ou des technologies dites "hybrides" ne peuvent pas apporter ;
- et dont les fonctionnalités n'ont pas besoin d'être parfaitement adaptées au hardware du téléphone et n'interagissent pas avec des services spécifiques à une plateforme, ce qui est par exemple indispensable dans certains domaines tels que les jeux, le m-commerce ou les beacons.



Développement d'une application mobile hybride

INFO CHRONO

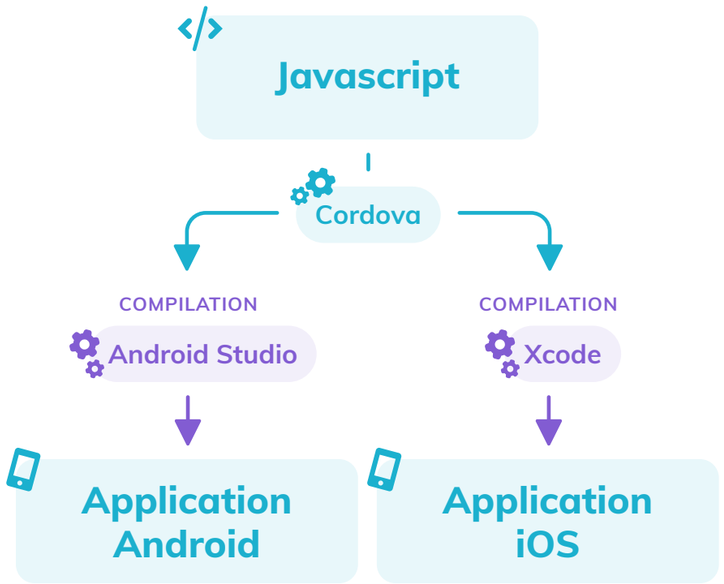
Nombreuses évolutions depuis 2011
Technologie toujours utilisé par les développeurs web

Application mobile hybride c'est...

... une application mobile développée à partir de langages web (HTML5, JavaScript, CSS...). Cependant, elle s'appuie également sur des technologies natives mobiles pour utiliser certaines fonctionnalités du smartphone.

Bien que développée avec du web, il s'agit tout de même bien d'une « application » dans le sens ou elle sera téléchargée depuis les magasins d'applications et installée sur le mobile, contrairement à la web app qui n'est consultable que depuis un navigateurvv.

Il existe différentes technologies de développement d'application hybride : PhoneGap, Rho Mobile, Sensa, Ionic, Apache Cordova. Dans le cas de ce dossier, nous aborderons plus particulièrement Apache Cordova.



1 Rédaction du code source

L'application est développée à partir de langages web. Le contenu web est alors « encapsulé » avec une interface mobile. Dans le code source, le développeur fait d'abord appel au SDK de la technologie qu'il utilise. Ce dernier va ensuite appeler les SDK des OS mobiles souhaités pour adapter le code source selon le système d'exploitation.

2 Compilation du code source

Ce code source est ensuite compilé par Cordova (ou une autre technologie hybride). Selon les parties du code source et les SDK appelés en amont, le logiciel va générer des applications spécifiques aux systèmes d'exploitations visés.

3 Génération d'une ou plusieurs applications mobiles natives

Comme pour les applications mobiles natives générées, on obtient différents fichiers d'exécution de l'application mobile après compilation, mais le contenu est très différent. Le code généré dans ces fichiers est mélangé : une partie du code est toujours en langages web ; certaines parties restent en JavaScript ; enfin, seule une petite partie du code est du code binaire (nécessaire pour utiliser les fonctionnalités du téléphone).

Avantages

- C'est une technologie multi-plateforme, c'est un gain de temps si le projet n'est pas trop complexe et ne nécessite pas le développement de plugins spécifiques ;
- Les compétences web sont plus répandues que les compétences mobiles, les coûts de développement sont donc généralement moins importants

Inconvénients

- Le code source est mélangé et la compilation se fait en de multiples étapes, ce qui peut fragiliser le fonctionnement de l'app et rendre la maintenance complexe ;
- Tout n'est pas réalisable en terme de fonctionnalités, il faut s'assurer en amont de la compatibilité de votre projet (et de ses évolutions futures) avec les technologies hybrides ;
- Des contenus trop lourds peuvent avoir de forts impacts sur les performances ;
- Le mode hors-ligne est plus limité et plus délicat à concevoir ;
- Un développement d'app hybride, nous rend dépendant d'un logiciel tiers et donc de sa compatibilité avec les nouvelles versions d'OS.

Ainsi, l'application mobile hybride est techniquement assez limitée, mais elle est tout de même à considérer dans un projet mobile simple qui fait appel à peu de fonctionnalités et ne supporte pas beaucoup de contenus.

Développement d'une application web native

INFO CHRONO

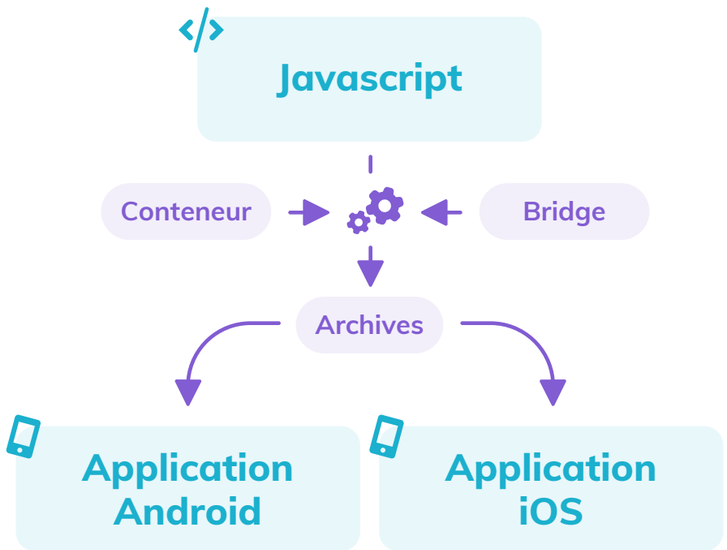
Depuis 2014 (NativeScript) et 2015 (React Nativ)
Utilisées par des app telles que Airbnb ou Facebook

Application mobile web native c'est...

... une application mobile développée à partir de langages web (JavaScript, TypeScript, Vue JS, Angular...) qui s'appuie sur les technologies natives mobiles pour utiliser certaines fonctionnalités du smartphone.

Par vulgarisation technique, nous pourrions dire que ce type d'application est un développement plus moderne à mi-chemin entre l'application native générée et l'application hybride. Elle permet d'améliorer les performances du modèle hybride en utilisant des composants natifs là où l'hybride n'affiche que des composants web. Ces technologies profitent donc de l'expérience des précédentes technologies pour en garder les avantages et pallier au maximum à leurs défauts.

Il existe différentes technologies dont les deux framework open source les plus connus sont React Native (Facebook) et NativeScript.



Notre avis

Bien que ces technologies soient intéressantes, elles présentent une architecture de code comportant beaucoup de couches de langages différents à maintenir, manquent de stabilité et présentent une dégradation de l'expérience utilisateur et/ou un coût technique important.

On peut constater que des plateformes, telles que AirBnB ont fait marche arrière sur cette technologie.

Flutter : une nouvelle approche

INFO CHRONO

Depuis 2017
En pleine accélération

Flutter, c'est le framework dont tout le monde parle, mais pourquoi ? Alors que nous avons pendant longtemps conseillé des développements purement natifs, faute d'avoir des résultats satisfaisants avec les développements multi-plateformes, Flutter rebat les cartes. Qu'apporte t-il pour réussir à souffler un vent nouveau dans le monde du développement mobile ?

Flutter, qu'est-ce que c'est ?

Flutter est le framework de Google permettant un développement multi-plateforme. Il permet de ne développer une application qu'une seule fois pour les différentes plateformes : iOS et Android.

Flutter s'appuie sur le langage de programmation DART (à l'origine appelé Dash), créé également par Google et présenté au public en 2011.

Quelles particularités ?

Flutter présente deux spécificités principales :

- **Les widgets :** ils permettent de décrire simplement le rendu final. Chaque objet est défini indépendamment des contraintes parentes. C'est son emplacement dans le code qui permettra de définir ses contraintes extérieures. Cela permet de construire facilement son interface ; le code est alors plus facilement lisible et maintenable.
- **Les composants :** ils ont été recréés par Google. Les développeurs disposent d'une galerie de composants s'adaptant à IOS comme Android, et aux différentes versions d'OS.

Grâce à la fonctionnalité Hot Reload du langage DART, le build des applications est très rapide, ce qui rend quasiment invisible le temps de compilation. Un autre gain de temps pour les développeurs ! Les avantages sont donc nombreux, mais il existe aussi des inconvénients.

Flutter : points positifs et négatifs

Voici la liste des avantages et inconvénients pour savoir si, oui ou non, votre projet se prête à utiliser Flutter ?

	Points positifs	Points négatifs
UX	+ Très bonnes performances, proches du natif + On peut tout faire ! Les éléments qui n'existent pas en Flutter peuvent être créées via des ponts vers du code natif en Swift ou Kotlin	- Les nouveautés annuelles de chaque OS ne sont pas forcément implémentables au moment de leur sortie
Design	+ Possibilité d'intégrer plus facilement des animations + Beaucoup de composants Material sont disponibles + Tout est possible	- Un design iOS/Android presque identique - Certains composants peuvent être difficiles à personnaliser
Développement	+ Utilisation d'Android Studio, Visual Studio Code ou de n'importe quel autre IDE + Moins de code pour le même résultat en natif + Intégration continue via Codemagic et annonce de Bitrise également + La documentation est très bien faite + Compilation extrêmement rapide : l'application se recharge automatique quand le code est modifié de manière presque instantanée	- Un langage supplémentaire - La connaissance d'au moins un des OS natifs est nécessaire - Le poids des applications, et notamment des APK, est plus important. Une phase d'optimisation est à prévoir
Spécificité Google	+ Montée en compétence très rapide en venant d'Android + Toutes les API Firebase sont très bien gérées	
Maintenance	+ Les corrections de bugs sont rapides et fréquentes + Les anciennes versions d'OS sont supportées - Android 4.1 et + - iOS 8 et + + Maintenance moins coûteuse : une seule application à corriger en cas de bug	- Evolution très rapide : les packages sont rapidement obsolètes et à mettre à jour
Pour l'avenir	+ Fortement poussé par Google + Projet Flutter pour le web (Hummingbird)	- Technologie encore jeune

Contacts

Laurent Bertaux
Dirigeant
02 23 20 15 75
laurent@mobizel.com

Émilie Clément
Marking et clientèle
02 23 20 15 75
emilie@mobizel.com

mobizel.com
ressources.mobizel.com