GUIDE PRATIQUE

PROJET MOBILE AVEC REACT NATIVE

Les 5 erreurs à éviter

KALIOP
INTERACTIVE MEDIA

INTRODUCTION 3

POURQUOI CHOISIR REACT NATIVE ? 3
A. Un framework efficace
B. React Native, technologie clé du marché

1. MÉCONNAÎTRE LES SPÉCIFICITÉS PROPRES AU MOBILE 5
A. Utiliser au maximum les composants par défaut
B. S'appuyer sur des librairies stables
C. Faire les bons choix d'architecture React

2. MAL ANTICIPER LA GESTION DU OFFLINE ET DES NOTIFICATIONS 9
A. Gérer le contenu offline
B. Faire les bons choix d'architecture pour les notifications

3. NÉGLIGER LA QUALITÉ DE CHAQUE VERSION DE VOTRE APPLICATION 11
A. Mettre la priorité sur la qualité
B. Analyser les performances en amont
C. Monitorer les performances en temps réel
D. Avoir une gestion fine des versions

4. SOUS-ESTIMER L'IMPORTANCE DES CHOIX DE TECHNOLOGIES BACKEND 13
A. Les questions à se poser dans le choix du backend
B. Exemple d'un backend on premise : Kuzzle

5. MAL ANTICIPER LA CONSTITUTION DE L'ÉQUIPE ET L'ORGANISATION DU PROJET 15
A. Constituer une équipe cohérente
B. Faire les choix de standard de code
C. Installer la CI et le déploiement en amont

CONCLUSION 16

INTRODUCTION

En 2016, dans le monde, la part de l'utilisation de la tablette et du smartphone a dépassé celle des ordinateurs pour se connecter à Internet. Même si la France n'a pas encore atteint ces proportions, il est indéniable qu'elle s'en approche au fil des ans.

Ainsi, avec 170 milliards d'entre elles téléchargées mondialement, le **développement des applications mobiles** est devenu stratégique pour tous les acteurs du marché. Pour les créer, des technos comme Flutter ou React Native sont des alternatives solides aux développements Kotlin / Objective C pour fabriquer une application mobile. Ce type d'infrastructure logicielle en JavaScript permet de concevoir des applications natives, à la fois sur iOS et Android.

Pour vous accompagner dans vos choix, vous trouverez ci-dessous un guide pratique, basé sur un retour d'expérience de plusieurs projets d'envergure que nous avons menés avec le framework React Native. Il vous permettra d'anticiper les **5 erreurs classiques à ne pas commettre dans le lancement d'un projet mobile en React Native**.

POURQUOI CHOISIR REACT NATIVE ?

A. UN FRAMEWORK EFFICACE

React Native est un framework JavaScript qui permet de réaliser des applications natives, sur iOS comme sur Android - et, sous conditions, sur Windows Phone.Son principal avantage est qu'il propose l'utilisation **d'un seul code source JavaScript pour construire deux applications**, au lieu de développer séparément une application iOS et une application Android.

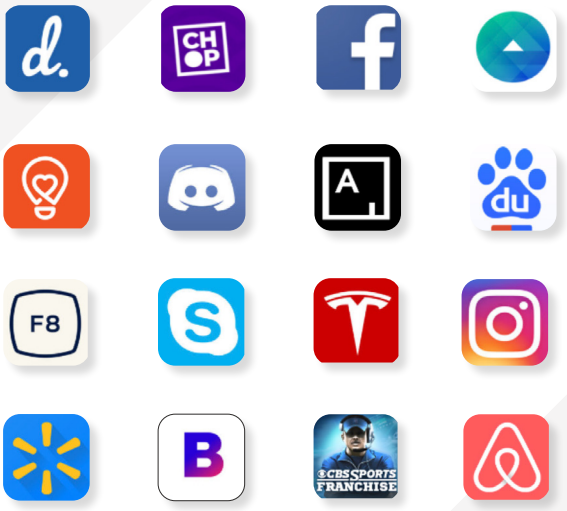
Les avantages de React Native

- Un framework JavaScript unique permettant de capitaliser sur des développeurs full stack, front et back end.
- Des coûts de développement divisés par deux.
- Des développements plus rapides qu'avec d'autres technologies grâce à une communauté riche et des librairies directement réutilisables.

- Une seule mise en place de tests automatisés pour iOS & Android.
- Une meilleure homogénéité entre les deux versions iOS et Android, car une seule équipe est mobilisée.

B. REACT NATIVE, TECHNOLOGIE CLÉ DU MARCHÉ

Développé par Facebook à partir de 2015 et **utilisé par des acteurs majeurs du marché de l'application** comme Airbnb, SoundCloud ou encore Instagram, ce framework est donc largement plébiscité et reconnu pour sa robustesse.



Avec tous ses avantages, React Native est une technologie attractive, qu'il est important de connaître. Son utilisation implique néanmoins d'éviter certains écueils.

1 - MÉCONNAÎTRE LES SPÉCIFICITÉS PROPRES AU MOBILE

Par rapport au monde du Web traditionnel, le monde du mobile implique notamment l'existence d'un **vaste parc d'appareils** disponibles sur le marché. Cette caractéristique entraîne donc une grande diversité de résolutions d'écrans, capacités de stockage, connectivités, capteurs ou encore fonctionnalités disponibles. Ainsi, se lancer dans la création d'une application nécessite de bien appréhender toutes les spécificités propres au mobile et à la prise en charge des deux systèmes d'exploitation (OS) que sont iOS et Android.

A. UTILISER AU MAXIMUM LES COMPOSANTS PAR DÉFAUT

Utiliser les composants proposés par défaut par React Native est très sécurisant et donne la certitude de bien gérer toutes les spécificités propres au mobile, tout en assurant parfaitement le portage sur les deux OS. Ainsi, le développement d'une application est grandement accéléré, car on prend pour bases des solutions déjà fonctionnelles.

La bibliothèque de composants React Native permet de gagner du temps dans les développements, en ajoutant seulement une touche de développement personnalisée.
Côté internautes, ceux-ci retrouvent les fonctionnalités standard qu'ils connaissent, dans le respect du design basé sur l'expérience utilisateur (UX-Design).

B. S'APPUYER SUR DES LIBRAIRIES STABLES

Pour répondre à des besoins plus complexes, non couverts par les composants par défaut, l'**écosystème React Native propose de nombreuses librairies**, très stables, sur lesquelles il est recommandé de s'appuyer.

Exemple : afin de créer un calendrier, on dispose de multiples possibilités, implémentations et designs différents

1. React Native Calendar Components
2. CalendarPicker Component for React Native
3. React Native calendar strip

1

July 2016
Sun Mon Tue Wed Thu Fri Sat
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

2

Previous
May 2017
Next
Mon Tue Wed Thu Fri Sat Sun
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
ELECTED START DATE:Tue May 02 2017 00:00:00
MT-0700 (PDT)
ELECTED END DATE:Sat May 20 2017 00:00:00
MT-0700 (PDT)

3

2:28 PM
August / September 2016
MON TUE WED THU FRI SAT SUN
29 30 31 1 2 3 4

Les critères de choix d'une librairie :

- la couverture fonctionnelle par rapport aux besoins du projet,
- la notoriété de la librairie (symbolisées par des étoiles ou stars),
- le nombre de problèmes signalés… et résolus,
- la communauté et les entreprises qui la développent – Wix, Facebook et Airbnb sont souvent à l'origine de librairies de qualité.

Très fréquent, le questionnement autour du choix d'une librairie implique de faire l'inventaire de ses besoins, étudier le champ des possibles (en sélectionnant et testant des librairies répondant aux besoins sur les deux OS), puis, personnaliser et adapter la librairie choisie.

C. FAIRE LES BONS CHOIX D'ARCHITECTURE REACT

Certains projets nécessitent de se **questionner sur l'architecture en amont**. C'est particulièrement le cas lorsqu'une solution sera déployée dans plusieurs pays ou qu'elle devra répondre à plusieurs cas d'usage.

Selon les fonctionnalités de l'application à développer, on peut créer :

- une application commune avec un code de base unique couvrant tous les besoins, en masquant les fonctionnalités spécifiques à certains usages lorsqu'elles ne sont pas requises,
- plusieurs applications, chacune comportant des fonctionnalités différentes.

- Une étude en amont avec l'équipe de développement est recommandée.
- Pour développer des applications multiples, il est préférable d'employer des composants réutilisables et suivis, puis de les déverser dans la communauté Open Source afin d'assurer leur maintenance.

Faut-il utiliser la technologie Expo ?

✓ **Pour une petite application, les fonctionnalités sont pratiques et accélèrent le développement en amont.**

x **Pour une grosse application, Expo ne permet pas de maîtriser tout le développement : si on en a le temps, il est préférable de le réaliser soi-même.**

Faire les bons choix de composants et de librairies est essentiel pour développer une application React Native solide. De même, la stratégie et l'architecture à adopter conditionnent grandement la réussite d'un projet, tout comme la gestion du offline et des notifications.

2 - MAL ANTICIPER LA GESTION DU OFFLINE ET DES NOTIFICATIONS

A. GÉRER LE CONTENU OFFLINE

A Kaliop, nous estimons que la gestion offline présente un surcoût de + 20 % si elle est pensée en amont des développements, et + 60 % si elle est subie à l'issue des développements.

Différentes stratégies peuvent être adoptées selon les fonctionnalités désirées. Aucun choix n'est plus indiqué qu'un autre et **chaque cas doit être étudié préalablement** en confrontant les besoins du projet et les propositions de l'équipe de développement.

Le principal choix qui s'offre, hormis l'absence de gestion, est de stocker le contenu en offline (*offline first*) ou sur le serveur (*online first*) :

	Avantages	Inconvénients	Coûts
Pas de gestion	_	Pas de contenu si pas de réseau	_
Offline first	Contenu très rapide	Peut « clipper » ou ne pas être à jour	++
Online first	Contenu à jour	Temps d'accès au contenu, car attend l'échec du réseau	++

Les critères de choix d'une librairie :

- Choisir le offline nécessite d'effectuer plusieurs tests, et ce, tout au long de la réalisation du produit.
- Pour ce faire, l'outil Charles Proxy est très intéressant : il simule la réponse du serveur avec différents scénarios d'instabilité du réseau (bande passante réduite, réseau coupé, etc.) auxquels pourront être confrontés les utilisateurs.

B. FAIRE LES BONS CHOIX D'ARCHITECTURE POUR LES NOTIFICATIONS

Les *push notifications* « réveillent » une application, même lorsque celle-ci est éteinte ou que le smartphone est en veille, afin d'avertir l'utilisateur. Elles sont donc cruciales dans le fonctionnement et la vie des applications.

Pour générer ces notifications, il est nécessaire d'utiliser les **services de notifications d'Android et iOS, par exemple, Firebase et Batch**. Encore une fois, il importe d'étudier la maturité de ces SDK (*Software Development Kit*), qui réunissent plusieurs outils (bibliothèques, langages de programmation…) permettant de développer rapidement les applis.

Étudier les fonctionnalités et l'architecture des notifications en amont est essentiel.
Des tests sur iOS et Android s'imposent donc, notamment sur **l'évolutivité des solutions en cas de gestions de notifications avancées** : squash notifications, regroupements par canaux ou notifications riches (par exemple avec des images).

**Bien étudier sa stratégie au sujet du offline et des notifications impacte le développement comme l'expérience utilisateur.
Il en va de même pour la gestion des versions d'une application React Native.**

3 - NÉGLIGER LA QUALITÉ DE CHAQUE VERSION DE VOTRE APPLICATION

Si React Native facilite le développement mobile pour iOS et Android, il est néanmoins indispensable d'analyser et de tester en amont, puis de monitorer, la qualité et les performances de chaque version.

A. METTRE LA PRIORITÉ SUR LA QUALITÉ

Le développement mobile implique obligatoirement la soumission des applications aux stores d'iOS et Android. **Les notes des utilisateurs et leurs commentaires**, positifs ou négatifs, conditionneront alors grandement l'échec ou la réussite d'une application.

Pour un développement mobile de qualité :

- analyser les performances en amont,
- automatiser les tests end to end, afin d'éviter les bugs,
- intégrer la gestion de différentes versions,
- réagir très vite aux commentaires et livrer rapidement des corrections.

Par rapport au Play Store de Google, **l'Apple Store se distingue par un système de validation manuelle**, beaucoup plus exigeant sur la qualité des applications. Quelques points méritent donc une vérification approfondie pour éviter les refus : crash et bugs, liens morts, *placeholder* à la place du contenu, teasings (« Bientôt disponible ») qui peuvent laisser croire à une version non terminée, etc.

B. ANALYSER LES PERFORMANCES EN AMONT

Afin de s'assurer que l'application ne présente aucun défaut, temps de latence ou bug, quel que soit l'appareil utilisé, il est important d'effectuer **différents tests en amont, non seulement sur des émulateurs, mais aussi sur de véritables appareils**, en conditions réelles. On portera particulièrement attention à la consommation d'énergie des librairies choisies, à la vitesse du rendu (en optimisant le poids des images) ou encore au poids de l'application en vue de son téléchargement.

Pour ce faire, **l'automatisation des tests** est hautement recommandée pour garantir une qualité irréprochable. Plusieurs outils permettent de réaliser des tests automatiques sur les applications, en couvrant l'ensemble des étapes requises, à l'instar de GitLab CI ou Detox.

C. MONITORER LES PERFORMANCES EN TEMPS RÉEL

Lorsque l'application est en production, pour se prémunir des inévitables crashs, il faut se montrer **réactif et être bien équipé afin de corriger les bugs** le plus rapidement possible.

- Un outil comme Bugsnag permet d'agréger les erreurs et les crashs, mais aussi de les corriger, avant que les utilisateurs n'aient le temps de faire des commentaires négatifs.

D. AVOIR UNE GESTION FINE DES VERSIONS

La gestion de version est encore plus importante sur le mobile que sur le Web classique, d'autant que **plusieurs acteurs sont impliqués**. Outre les multiples appareils et les deux OS, il existe différentes versions de React Native, des librairies utilisées, du backend et donc, du code de l'application.

Il en résulte un **grand nombre de combinaisons possibles** pour chaque utilisateur, dont il faut absolument assurer la maintenance pour une application fonctionnelle, même si elle n'a pas été mise à jour. D'où l'importance également, de choisir des librairies fiables et bien maintenues, en évitant le plus possible de créer des forks (dérivés du code originel).

L'App Center CodePush

Hébergé sur le réseau, CodePush récupère à intervalle régulier le code de l'application en développement. L'App Center intègre des fonctionnalités très pratiques pour résoudre des problèmes rapidement, notamment en cas de bug à résoudre.

De nombreux process et outils permettent de tester, analyser et monitorer les applications de façon réactive, afin de fournir des versions de qualité. Une démarche cruciale pour s'assurer de la meilleure notation possible dans les stores et donc, de la réussite de l'application.

4 - SOUS-ESTIMER L'IMPORTANCE DES CHOIX DE TECHNOLOGIES BACKEND

LE BACKEND, PARTIE CACHÉE DE L'ICEBERG DE L'APPLICATION, DOIT FOURNIR UN CERTAIN NOMBRE DE FONCTIONNALITÉS :

- Gestion de l'authentification

- Stockage des données des utilisateurs (identifiants, données RGPD ou métier, etc.)

- API riche et rapidement exploitable (en GraphQL par exemple)

- Temps réel

- Geofencing

A. LES QUESTIONS À SE POSER DANS LE CHOIX DU BACKEND

Pour travailler sur le backend dans React Native, il est possible soit d'utiliser directement des SDK si elles existent, soit une API, en réécrivant des couches de service communiquant avec le backend.

Côté déploiement, **le serveur du backend se doit d'être performant**, surtout si l'application vise un large public. On en attend ainsi une haute disponibilité, une scalabilité dans plusieurs pays avec la capacité de monter en charge en cas de pic, sans oublier le respect des contraintes RGPD et données de santé.

Il existe différentes solutions pour développer un backend, comme AWS et Google Cloud Platform, mais aussi Kuzzle.

B. EXEMPLE D'UN BACKEND ON PREMISE : KUZZLE

Kuzzle est un backend développé par un éditeur Open Source. Directement installable, il permet d'accélérer le déploiement d'une application : il propose des SDK qu'on peut embarquer dans du React Native, mais aussi du Cordova ou du Xamarin.

Cette solution sur étagère donne un véritable **coup d'accélérateur à toutes les problématiques de backend** : il permet d'affranchir l'équipe de toute la partie DevOps / scalabilité et de se concentrer uniquement sur ce qui a de la valeur : le code métier.

Kuzzle présente également l'avantage d'être installable on Premise, ce qui permet de gérer l'hébergement de données sensibles sur les serveurs de votre choix.

S'il importe de bien réfléchir au choix de son backend pour développer une application performante, la composition de l'équipe et l'organisation du projet sont également des facteurs clés de réussite.

5 - MAL ANTICIPER LA COMPOSITION DE L'ÉQUIPE ET L'ORGANISATION DU PROJET

A. CONSTITUER UNE ÉQUIPE COHÉRENTE

POUR MENER UN PROJET D'ENVERGURE AVEC REACT NATIVE, **DIFFÉRENTS PROFILS DE DÉVELOPPEURS** SONT REQUIS :

- un noyau de développeurs React Native maîtrisant le JavaScript et sensibilisés à la culture mobile,
- un développeur frontend, orienté utilisateur, se concentrant sur le rendu graphique (animation, qualité des écrans, etc.),
- un devops sur la partie CI, pour optimiser le build des applications, afin d'effectuer des livraisons fréquentes, des patchs rapides, etc.,
- un développeur iOS et un développeur Android, disponibles à la demande, afin notamment de faire évoluer les librairies.

B. FAIRE LES CHOIX DE STANDARD DE CODE

Dans le cas d'applications complexes, les choix de code structurants importent dès le début du projet. Ainsi, **les librairies standard sont recommandées**, car elles améliorent l'accessibilité et l'organisation du code, garantissant la stabilité et le maintien de l'application. On préconisera de bien évaluer le niveau de l'équipe et sa capacité à manipuler ECMAScript, Redux ou TypeScript.

C. INSTALLER LA CI ET LE DÉPLOIEMENT EN AMONT

La CI et le déploiement continu doivent être gérés en amont, bien avant même les livraisons. De fait, une fois que le processus de déploiement est défini et stable, il permet de **rapidement soumettre l'application aux stores**, afin de continuer à développer le produit sereinement.

Structurer ses versions permet de gagner en clarté : faire figurer des informations sur les écrans des applications au niveau des environnements et des versions évite le risque de confusion.

CONCLUSION

React Native est un framework complet, permettant de **gagner du temps dans le développement** d'un projet, tout en construisant des applications robustes, à la fois pour iOS et Android.

Notons qu'il existe des alternatives à React Native :

Xamarin, framework historique, est en langage C# (C sharp) : son mode de compilation, plus compliqué, diminue la productivité. Néanmoins, il peut être intéressant lorsque l'on possède un environnement C# très fort par ailleurs.

NativeScript est une initiative Open Source basée sur les frameworks Angular et Vue.js. Ce framework est moins soutenu en raison de son caractère Open Source.

Flutter est une réplique de React Native développée par Google. A terme il n'est pas impossible que Google remplace le traditionnel Java par Flutter (DART) Prometteuse, elle est plus récente, donc moins avancée et comportant moins de librairies disponibles. l'effort semble plus porté sur Android que iOS.

Les PWA (Progressive Web Apps ou applications web progressives) sont une alternative permettant de créer des sites web installables sur mobile, en gérant le mode offline et le système de notifications.

Il ressort néanmoins de la comparaison que React Native présente l'avantage de réduire le *time to market* et les coûts de développement... sous réserve de respecter les cinq conseils précités.

Les mots d'ordre pour un projet React Native réussi : anticiper et tester !

Vous avez prévu de lancer une application mobile, mais vous hésitez entre une native app, une hybride app ou une Progressive Web App ? Alexis Trancart, Directeur Conseil chez Kaliop et expert mobile, vous présente en vidéo les avantages et inconvénients de chaque format, et vous donne les clés pour choisir celui qui conviendra le mieux à votre projet.

Voir la vidéo

Rationaliser le sourcing des équipes, augmenter la qualité des produits, réduire le time-to-market, construire des architectures plus scalables... Et si ce bon vieux JavaScript était devenu une solution ultra performante pour relever ces enjeux ? Dans ce webinar, nous vous proposons un tour d'horizon des technos et concepts JavaScript : Node.js, Serverless, React, Angular, React Native, Vue.js...

Voir la vidéo

À PROPOS DE KALIOP

Le groupe Kaliop est un spécialiste de la transformation numérique et de l'innovation. L'excellence technique est à la source de notre ADN, l'agilité est au cœur de notre méthodologie. L'open-source est une valeur centrale de nos choix technologiques. Grâce à ces engagements, Kaliop a mené à bien depuis plus de 15 ans des centaines de projets digitaux stratégiques, en associant l'expertise technique à une vision fonctionnelle centrée sur la création de valeur pour l'utilisateur final.

tech.kaliop.com - 01 80 49 30 00

Copyright © 2019 Kaliop

Tous droits réservés. Cet ouvrage ne peut en aucune manière être reproduit en tout ou partie, sous quelque forme que ce soit ou encore par des moyens mécaniques ou électroniques, y compris le stockage de données et leur re-transmission par voie informatique sans autorisation de Kaliop.
La citation des marques est faite sans aucun but publicitaire. Les erreurs ou les omissions involontaires qui auraient pu subsister dans cet ouvrage malgré les soins et les contrôles de Kaliop ne sauraient engager leur responsabilité.

KALIOP
INTERACTIVE MEDIA