IT360

GLRT

Rapport

Architecture Logicielle

Etudiants :
Ahmed Ezzarzari
Nicolas Blanc
Kylian Ferron
Alexandre Baudry

Enseignant :
Charles Consel

14 décembre 2022

Table des matières

1 Introduction

2 Problème

Lors du développement d'un nouveau logiciel, il faut prendre en compte certains critères afin que les technologies soient adaptés au projet.

2.1 Plateforme ciblée

Tout d'abord, il faut prendre en considération les plateformes sur lesquelles le projet sera déployé. Par exemple, si l'application est destinée à du desktop est qu'elle est compatible Windows et Linux Mac, certains langages n'utiliseront pas le même type de compilateur (ce qui est le cas pour le C), alors que par exemple Java va être déployable sur toutes les plateformes dotées de JVM. AUssi, pour du développement web et mobile, si le produit est dédié à la fois pour du web et du mobile, utiliser Flutter serait sûrement plus adapté car il suffirait de produire qu'un seul code pour être sur la quasi-totalité des plateformes. React et React Native pourrait également être un exemple. A l'inverse, si l'on veut se concentrer que sur une plateforme, iOS par exemple, Swift est plus adapté.

2.2 Performances

Aussi, les performances de la technologie sont à prendre compte. Si on veut développer un logiciel très performant, il faudrait utiliser un langage plus bas niveau avec moins de surcouche logicielle en contrepartie. Pour la partie backend, il faudrait prendre en compte aussi la scalibilité : la rapidité d'accès à la base de données, la rapidité des requêtes avec la partie client etc. Pour le frontend, si l'on a besoin d'animations, le choix du langage est également important.

2.3 Durée et taille du projet

En fonction de la taille du projet, les besoins sont différents. Pour un projet de petite taille et de courte durée, il est plus logique d'utiliser des outils clés en mains (outils Google, Amazon, Microsoft etc.) ou bien des bibliothèques. Au contraire, pour des projets de plus grande envergure, il serait plus judiciable de développer sa propre bibliothèque afin qu'elle soit plus adaptée aux besoins du projet.

2.4 Communauté

La popularité du langage est également un élément à prendre en compte, car la communauté autour d'un langage, peut aider au développement, ce qui peut être le cas par exemple avec un grand nombre de bibliothèques développées, une meilleure documentation ou encore des forums tels que StackOverflow plus complets et qui facilitent l'étape de développement.

2.5 Élasticité du langage

Lors de l'ajout de fonctionnalités, il faut réfléchir à l'élasticité du langage, c'est à dire qu'il faut se poser les questions suivantes. Peut-on utiliser une capacité du langage sans inclure une nouvelle bibliothèque ? Sinon, la fonctionnalité est-elle facile à développer à partir de la bibliothèque du langage ? Sinon, il faut prendre en compte le coût de développement d'une bibliothèque spécifique au projet.

2.6 Maintenance et choix du paradigme

Enfin, le paradigme du langage est aussi un élément important pour le projet et sa maintenance. Un paradigme fonctionnel est plus rapide à implémenter mais plus compliqué à maintenir. Au contraire, un paradigme objet est plus long à mettre en place mais plus facile à maintenir.

3 Frontend

3.1 Comparaison entre Flutter & React Native

Afin de définir les critères permettant à un développeur mobile de choisir son framework de mise en place du frontend, nous allons effectuer une etude comparative des frameworks "Flutter" et "React Native". Il est dans un premier temps intéressant de prendre en considération les différentes applications mobiles ayant été réalisées à partir des deux différents frameworks. On compte parmi les application mobiles ayant été réalisées avec Flutter, Google Pay et Ebay contre Facebook, Tesla ou encore discord pour React Native.

3.1.1 Comparaison de la documentation

D'un point de vue purement esthétique la documentation de flutter est plus conviviale et facile à lire que celle de React native. De plus Flutter possède une documentation complète sur l'intégration continue / déploiement continu (CI/CD) qui se fait directement via le Flutter CLI contrairement à React Native qui ne fournit pas une documentation officielle pour le déploiement automatisé d'applications vers une plateforme, ni sur la configuration CI/CD. Cette documentation facile d'accès, très complète et agréable à lire était un point très important dans la réalisation de notre projet qu'il a fallut réaliser en peu de temps.

3.1.2 Comparaison de la communauté

Dart est beaucoup moins utilisé que JavaScript et que les contributeurs de ce framework sont relativement moins expérimentés que ceux de React Native. En effet React Native possède une communauté grande, active et expérimentée. Ceci revient au fait que la base de ce framework est JavaScript, un langage très utilisé, à cela s'ajoute l'ancienneté relative de ce framework par rapport à Flutter. Comme notre projet ne possédait pas de fonctionnalités complexes à implémenter la communauté n'était pas un critère déterminant pour le choix du framework.

3.1.3 Comparaison des performances

Dans le cas de Flutter, la majeure partie du travail est effectuée sur le GPU (unité de traitement graphique) ; c'est pourquoi, l'interface utilisateur de Flutter est fluide et offre 60 images par seconde. React Native quand à lui est compilé en code natif, il se base sur un pont entre le code JavaScript et l'écosystème natif de l'appareil afin d'exécuter le code source, ce qui le rend moins performant que Flutter.

3.1.4 Comparaison des Langages

Le language utilisé par le framework Flutter est Dart, un langage de programmation orienté objet développé par Google, mais qui est beaucoup moins adopté et utilisé comparé à JavaScript. Pour React Native on utilise JavaScript qui est un langage de programmation de scripts, c'est l'un des langages les plus utilisés avec une très large communauté.

3.1.5 Comparaison de la maturité

En dépit de l'importante évolution réalisée par Flutter et de la stabilité qu'il a réussi à atteindre en aussi peu de temps, le framework a encore du retard à rattraper en termes de maturité en comparaison avec React Native. Ceci est notamment du au fait que l'équipe Facebook a largement eu le temps de le stabiliser depuis mars 2015.

3.1.6 Comparaison des bibliothèques

Même si Flutter bénéficie d'un important soutien de la part de Google, comparé à React Native, le Framework est moins riche en bibliothèques. Cependant dans le cas de notre projt il n'était pas utile d'avoir à notre disposition toutes les bibliothèques existantes pour React Native.

3.1.7 Comparaison des UI

Flutter fournit ses propres composants UI et widgets qui sont subjectivement aux yeux du groupes plus beau que ceux de base proposés par React Native. Par ailleurs, la compatibilité de Flutter avec tous les appareils est sans doute sa plus grande force.

4 Backend

Dans cette partie, nous allons se concentrer sur la comparaison de la technique utilisée pour stocker toutes les données de notre application "Zouique" avec d'autres technologies qui sont populaires au sein du marché.

Pour commencer, nous allons essayer de comprendre l'architecture utilisée par Firebase, ensuite la comparer avec celle de mongoDB. En deuxième axe, nous allons comparer les performances de ces deux derniers afin de répondre aux questions de scalabilité, en avant-dernier nous allons voir tous les services que Firebase & MongoDB proposent. Au final, la popularité au sein des développeurs.

4.1 Comparaison entre FireBase & MongodB

Firebase & MongoDB sont deux technologies qui permettent de créer une database moderne robuste et fiable pour les applications. Avant de commencer leur comparaison, nous allons parler des similarités entre les deux systèmes, les deux sont des posts-relational databases basés sur JSON document datèrent models et schémas. elles sont construites pour faciliter l'accès aux ingénieurs back-end aux données et la scalabilité en grande échelle.

4.1.1 Comparaison d'architectures

Dans cette partie, nous allons comparer l'architecture de FireBase avec celle de MongoDB.

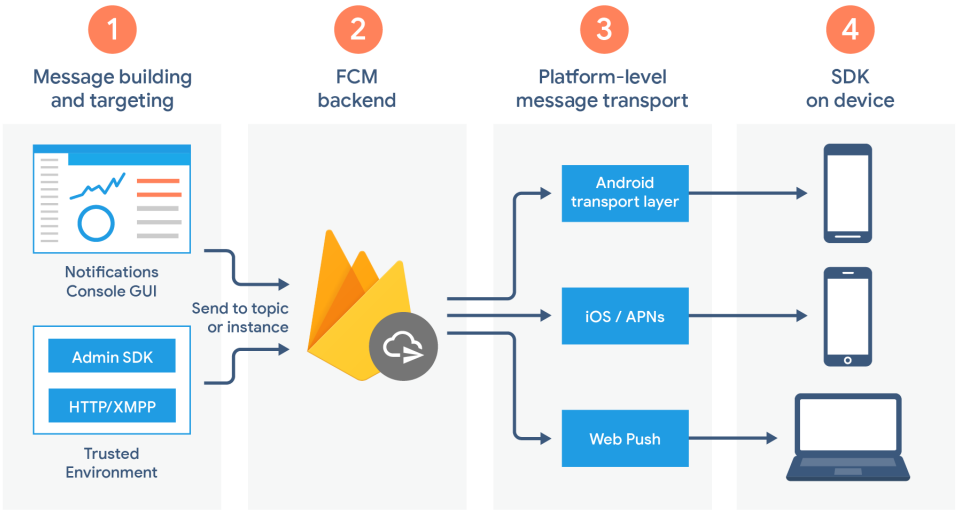

FIGURE 1 – Architecture FireBase

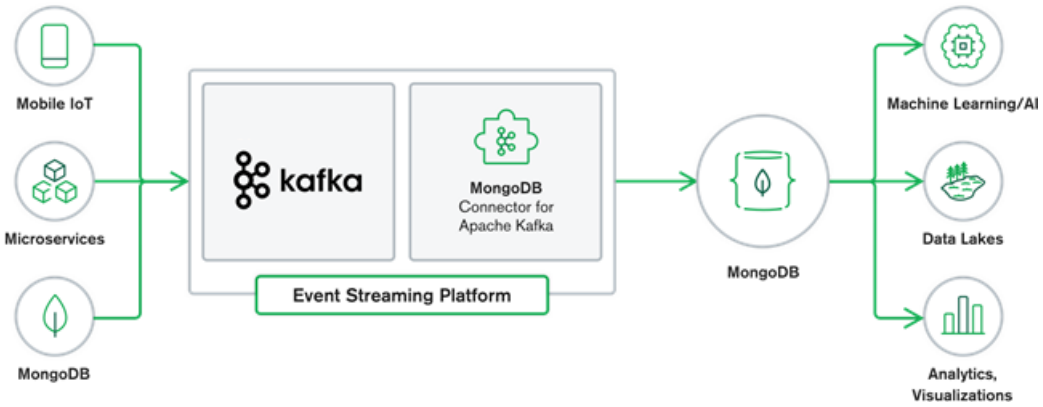

FIGURE 2 – Architecture MongoDB

MongoDB est une base de données de documents plus robuste connue pour ses hautes performances et sa sécurité de premier ordre, et présente plusieurs avantages par rapport à Firebase. Par exemple, MongoDB peut être exploité sur site ou dans le cloud (à l'aide de MongoDB Atlas

ou du cloud autogéré MongoDB), tandis que Firebase est purement un service de base de données cloud.

4.1.2 Comparaison des performances

Critères	FireBase	MongodB
Performances	Inférieur à mongodB	Meilleur
Cloud Support	Oui	Non
API et autres access methods	Android, iOS, JavaScript API, RESTful HTTP API	JSON Protocoles
Securité	Inférieur à mongodB	Meilleur
Adaptabilité	Small-scale application	Large-scale application

FIGURE 3 – Comparaison de performances

4.1.3 Comparaison des services

FireBase	Mongodb
Auth & Realtime DataBase	Data analytics support
Cloud Messaging & Storage	Dynamic Schemas support
Remote Control	Adjustable and document-based
Notifications & Crash Reporting	x

FIGURE 4 – Comparaison de services

4.1.4 Comparaison de popularité au sein du marché

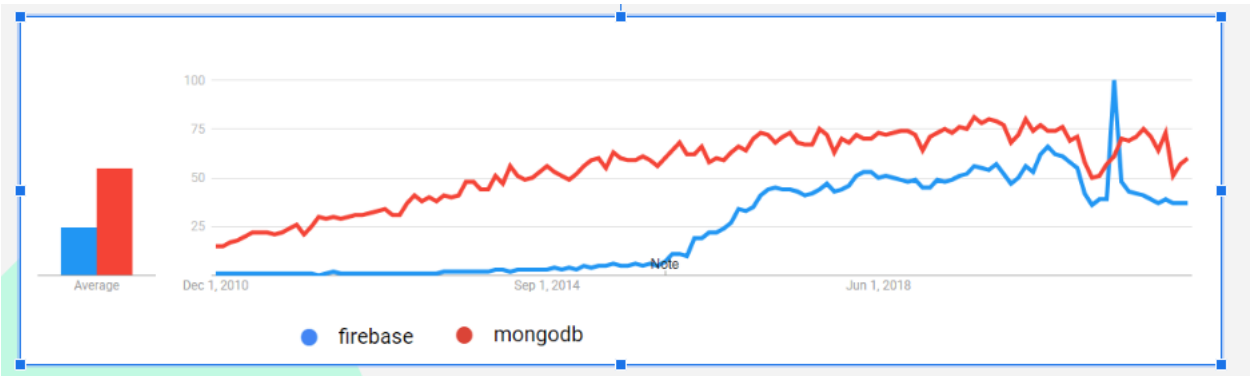

FIGURE 5 – Popularité au sein du marché mondial

5 Conclusion

Concernant le projet Zouïque, les choix qui nous ont porté à utiliser une architecture Flutter / Firebase sont les suivants :

Tout d'abord, notre application destinée à plusieurs plateformes. De plus, le projet doit être développé sur une courte durée donc nous avons besoin d'outils déjà présents pour faciliter le développement de nouvelles fonctionnalités. Avec ses nombreuses bibliothèques développées nativement via les widgets, Flutter semble être la solution idéale pour notre projet. Son importante communauté et sa documentation en font également des arguments importants étant donné que nous sommes novices. Enfin, Flutter n'est pas forcément le plus performant (car il est basé sur Dart qui est basé sur Javascript qui est lui-même un niveau d'abstraction au dessus) mais ce n'est pas un problème pour notre projet.

Du côté de Firebase, c'était une solution tout à fait légitime car c'est dans le package Google et donc cela fonctionne bien avec Flutter. Aussi, nous n'avions pas besoin de serveur backend car aucun traitement lourd n'était à faire. En plus, Firebase est très efficace pour tout ce qui est authentification d'utilisateur et donc cela simplifiait grandement notre travail de ce côté là.

Au final, nous avons pu voir que de nombreux critères sont à prendre en compte pour le choix des technologies utilisées lors du développement d'une application. Ces choix ont une incidence directe sur la structure du projet et les fonctionnalités disponibles.