

PROJET INDUSTRIEL

par Diallo Moustapha et Mourey Arthur



Introduction	4
Cahier des charges	4
Objectifs du projet	4
Spécifications du cahier des charges	4
Fonctionnalités attendues	5
Conception	6
Maquettage	6
Diagrammes UML	6
Use Case	7
Diagramme de classes	7
Diagramme d'activité	8
Diagramme de séquence	9
Diagramme de déploiement	9
Réalisation	11
Client	11
Android Studio	11
Lecteur de musique	12
Connexion au serveur et à la base de données	12
Google Maps API	13
Firebase Cloud Messaging	14
Firebase Realtime Database	15
Serveur	16
PHP et MySQL sur serveur OVH	16
Firebase Cloud Messaging	16
Sécurité	17
Coordination du groupe	17
Problèmes rencontrés	18
Aperçu de l'application	18
Connexion et inscription	18
Lecteur de musique	19
Carte de partage	19
Conclusion	20

Introduction

Le but de ce projet a été de concevoir et réaliser une application mobile permettant aux utilisateurs d'écouter et de partager de la musique. Afin de mener à bien ce projet, nous allons dans une première partie, intitulée "Cahier des charges", décrire les fonctionnalités attendues et faire une étude des besoins.

La seconde partie intitulée "Conception du projet" permettra de présenter la conception de l'application à l'aide d'une maquette et de diagrammes UML, afin d'identifier les différentes tâches à réaliser dans la partie suivante.

Enfin, nous consacrerons la troisième et dernière partie à la "Réalisation du projet" où nous présenterons, les choix techniques et outils de réalisation, puis en second lieu, le résultat de notre réalisation à travers des captures d'écran des différentes fonctionnalités réalisées.

Cahier des charges

Objectifs du projet

L'objectif premier de ce projet étant l'analyse, la conception et la réalisation d'une application mobile, celle-ci permettra aux utilisateurs d'écouter de la musique et de la partager avec les personnes qui les entourent grâce à la géolocalisation. Les autres usagers pourront alors savoir qui, autour d'eux, écoutent telle ou telle musique et ainsi découvrir de nouveaux artistes, de nouvelles musiques et même rencontrer des personnes avec les mêmes goûts musicaux qu'eux. Une fois cet objectif atteint, il n'est pas exclu d'apporter de nouvelles fonctionnalités à l'application.

Spécifications du cahier des charges

- Toute personne, dite "anonyme", c'est à dire qui ne possède pas de compte, peut s'inscrire pour avoir accès aux fonctionnalités de l'application.
- Une fois inscrite, elle pourra alors accéder à tous les albums et les musiques disponibles.

- Elle pourra également, grâce à la géolocalisation voir les personnes actuellement en ligne et les musiques qu'elles écoutent.
- Elle pourra aussi partager la musique qu'elle écoute avec les autres utilisateurs.

Fonctionnalités attendues

- Enregistrement des utilisateurs en renseignant les différentes informations (nom, prénom, adresse...).
- Un lecteur pour écouter les musiques stockées sur le serveur du système.
- Partager les musiques avec les autres utilisateurs.
- Géolocalisation des utilisateurs sur une carte en indiquant la musique actuellement écoutée.

Conception

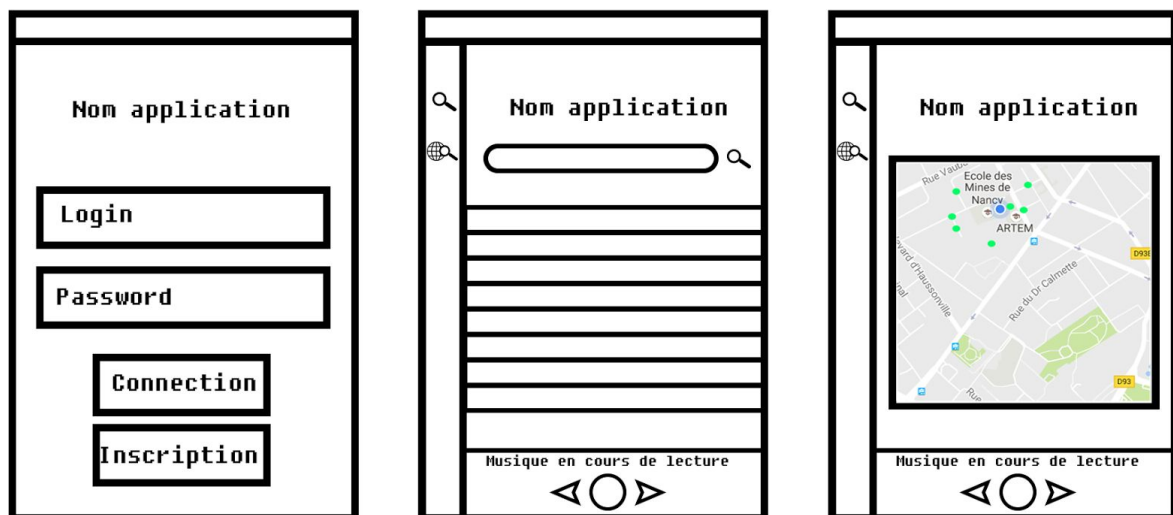
Dans un premier temps, nous avons commencé par concevoir l'application à l'aide de maquettes et de diagrammes UML qui nous ont permis de faciliter la compréhension et la communication au sein du groupe.

Maquettage

Les maquettes réalisées ci-dessous présente la réflexion que nous avons eu par rapport à l'interface de l'application. Celle-ci est structurée de manière à être intuitive et ergonomique pour l'utilisateur.

Dans cette application, il n'y a pas un très grand nombre d'activités, ce qui permet de ne pas perdre l'utilisateur et qu'il puisse se repérer facilement grâce à des icônes qui ne troublent pas sa navigation.

(Une autre vue concernant l'inscription n'est pas représentée ici, car elle est similaire à celle de l'authentification.)



Diagrammes UML



Pour la réalisation des diagramme UML nous avons utilisé Visual Paradigm.

Use Case

Premièrement, nous avons identifié les possibilités d'action de l'utilisateur en réalisant un diagramme de Use Case.

L'utilisateur est alors capable de s'inscrire, de s'authentifier et, suite à cela, de rechercher et d'écouter de la musique enregistrée sur une base de données commune.

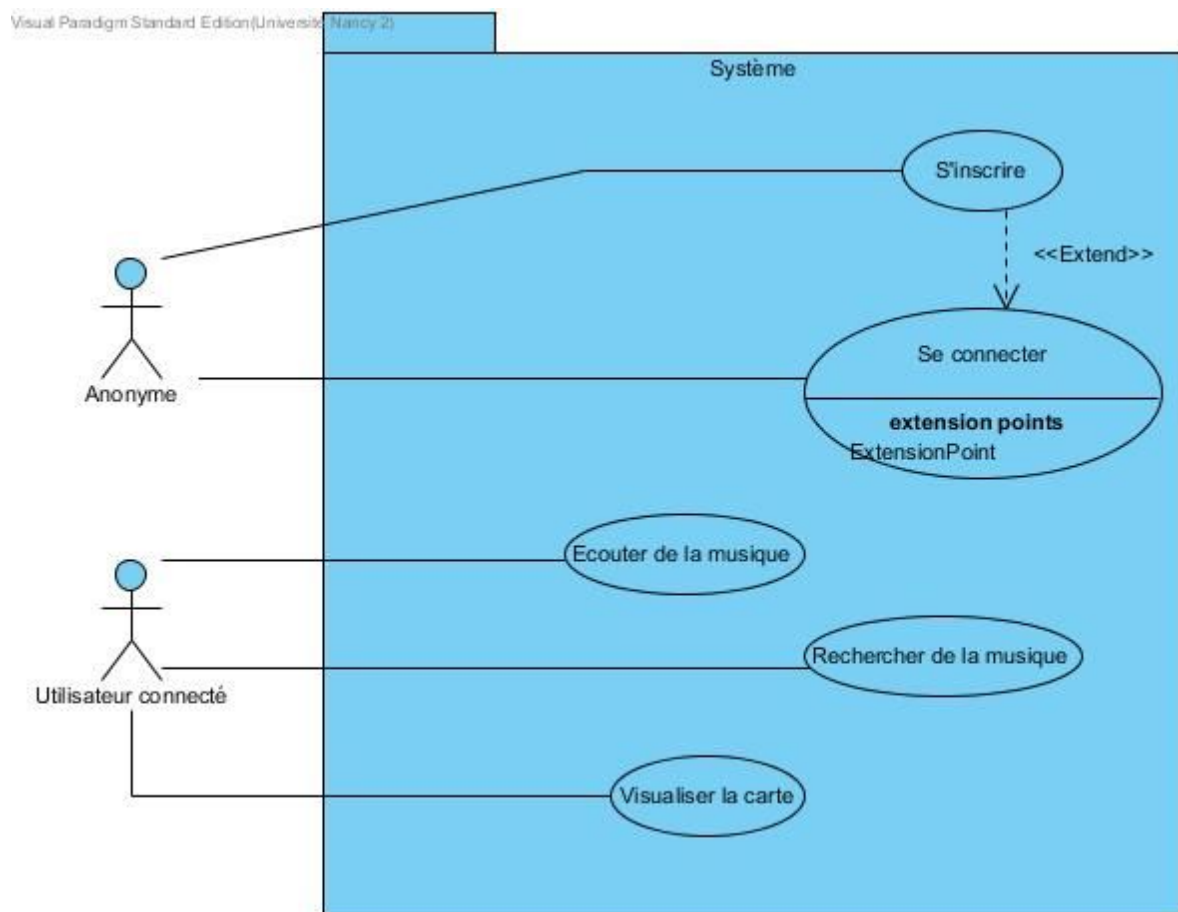
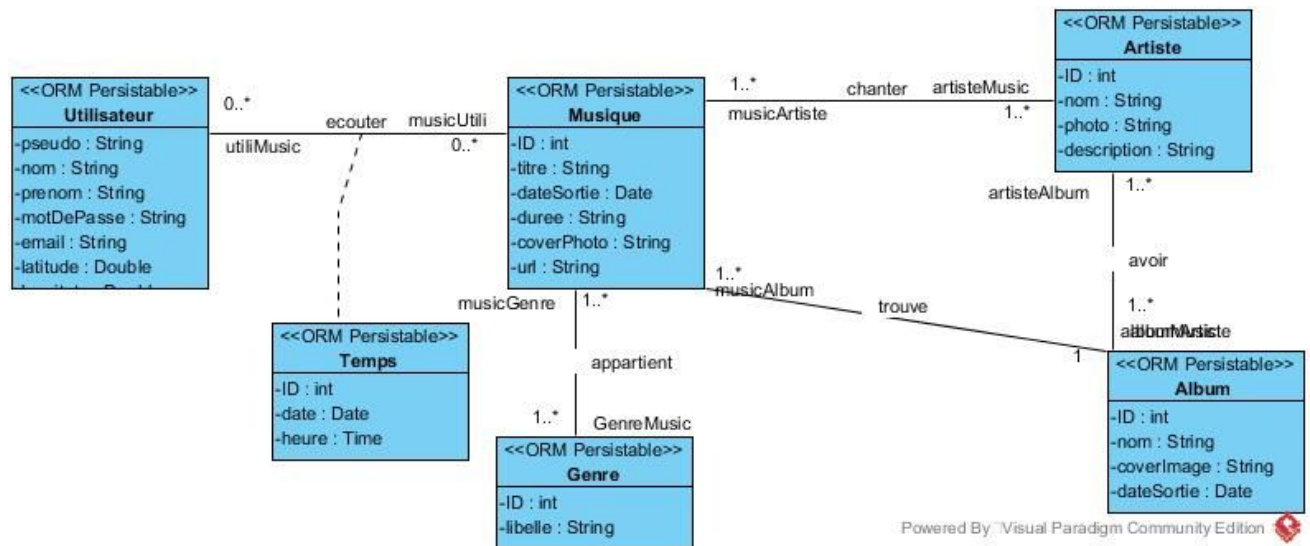


Diagramme de classes

Ensuite, nous avons modélisé les différentes entités de l'application par un diagramme de classes pour mieux nous représenter leurs attributs et les liens entre elles.

On peut alors constater qu'un Utilisateur peut écouter plusieurs musiques à des instants différents. Il nous renseigne également sur ses informations personnelles et sa position (qui

peut être à null) pour pouvoir le géo-localiser. La musique possède aussi de nombreuses informations telles que l'artiste, le genre et l'album auquel elle appartient.



Et à l'aide de ce diagramme, nous avons pu créer le diagramme suivant afin de générer la base de données :

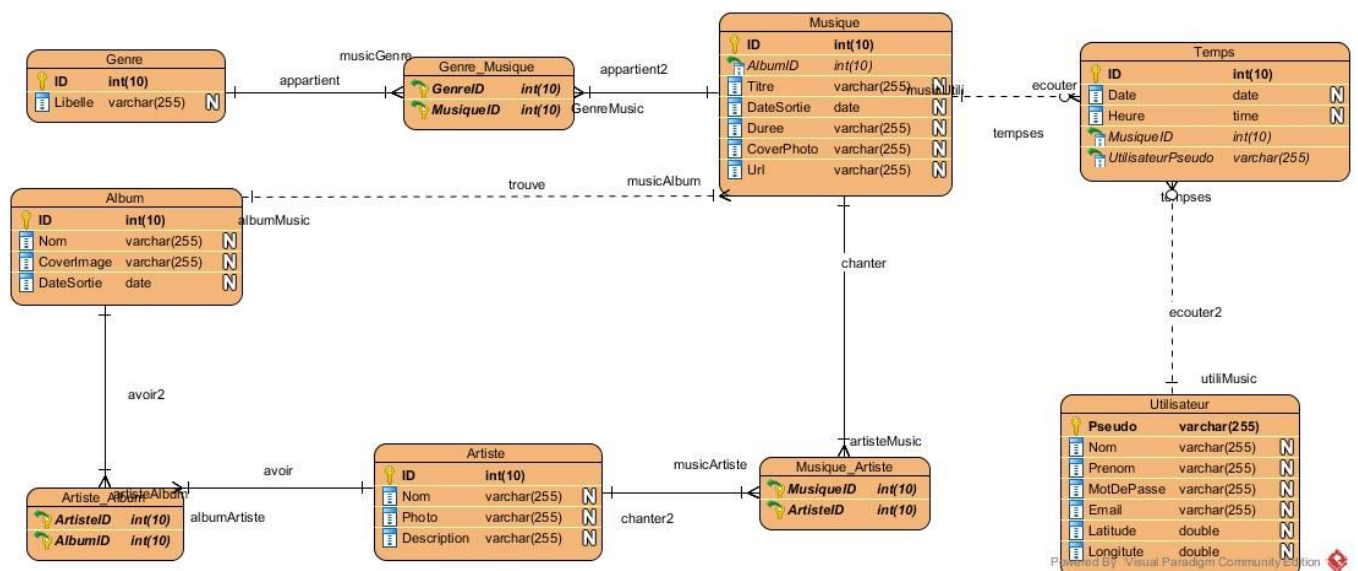
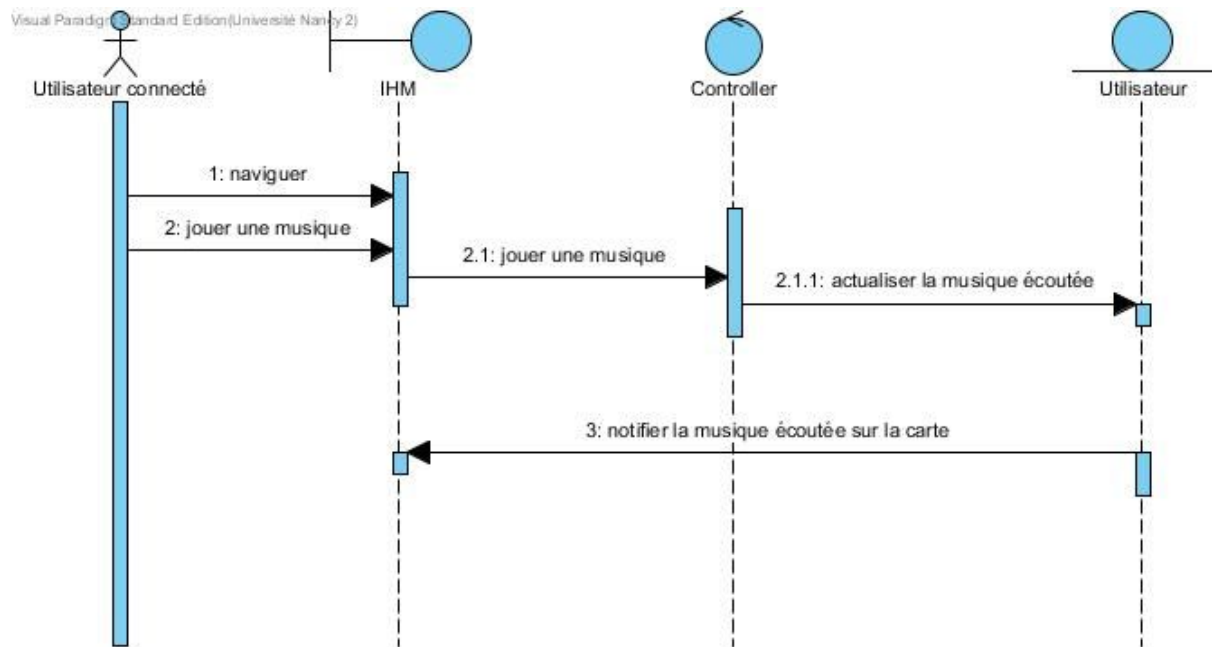


Diagramme d'activité

L'application ne comportant essentiellement que des actions simples de type CRUD (Create Read Update Delete), l'ensemble des diagrammes d'activités possibles restent trop simples ou sans intérêt. C'est pourquoi nous avons choisi de ne pas les représenter dans ce rapport.

Diagramme de séquence



Nous avons choisi de représenter ici le déroulement de l'action de jouer de la musique. Lorsque l'utilisateur connecté décide de jouer une musique, il va naviguer dans l'IHM afin de trouver la musique qu'il souhaite écouter, puis taper dessus pour la lancer.

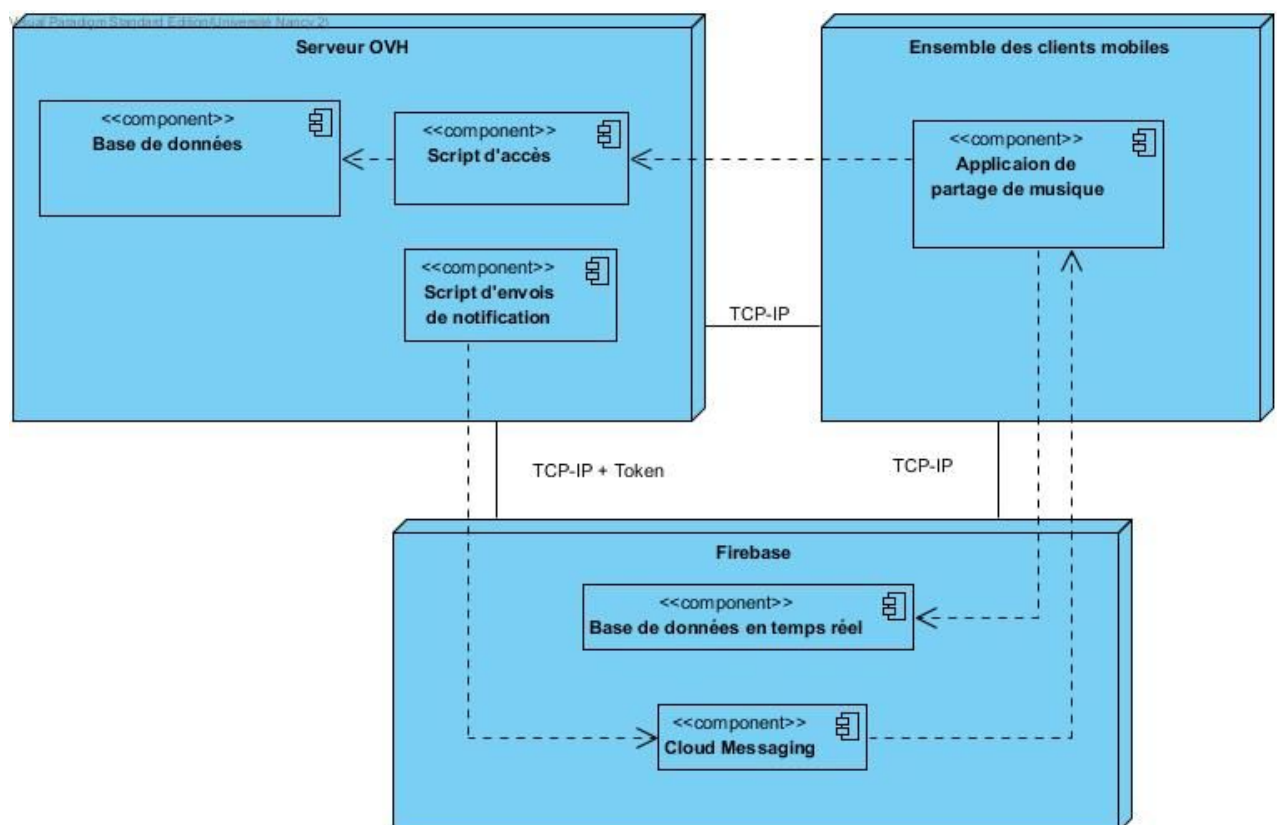
C'est alors au contrôleur de jouer la musique en question et d'actualiser le modèle de l'utilisateur pour modifier la musique courante. Une fois l'actualisation effectuée, l'IHM va être notifiée de ce changement et s'actualiser à son tour pour afficher la musique écoutée par l'utilisateur sur la carte.

Diagramme de déploiement

On constate sur ce dernier diagramme que l'ensemble des clients mobiles sont représentés dans un seul noeud, pour permettre une meilleure compréhension et lisibilité de celui-ci. Chez les clients, nous avons donc l'application de partage de musique qui va pouvoir interroger les scripts d'accès en PHP du serveur pour différentes actions telles que l'authentification ou la lecture de musique.

Sur le serveur, on peut distinguer un autre composant Script d'envoi de notification, qui, comme son nom l'indique, va nous permettre de créer un message qui sera reçu comme notification par tous les utilisateurs.

Ce service étant possible par le composant/service proposé par Firebase (Cloud Messaging) qui nécessite les tokens associés aux différents comptes utilisateurs. Et dans ce même noeud Firebase, nous avons une base de données en temps réel qui permettra à l'application de tous les utilisateurs d'être notifié d'un changement dans le modèle, tel que la localisation d'un utilisateur ou la musique qu'il écoute actuellement.



Réalisation



Pour réaliser l'application, nous avons utilisé un support mobile Android pour les clients et le serveur, hébergé chez OVH, d'Arthur (www.madpumpkin.fr) pour le côté serveur et gestion de base de données.



Client

Au niveau du client, les fonctionnalités à implémenter sont le login, l'écoute de musique et la visualisation des utilisateurs aux alentours et de leurs informations. Il y a également la présentation des vues à réaliser.

Android Studio



Lors de la déclaration des sujets de projets industriels, nous avons décidé de réaliser l'application côté client à l'aide d'Ionic qui est framework open source basé sur AngularJS et Apache Cordova. Mais plus tard, le choix d'Android Studio s'est avéré plus intéressant.

Notamment, pour des raisons d'apprentissage plus rapide, car Android Studio utilise le langage JAVA, connu des membres de notre équipe, et que Moustapha avait déjà utilisé cette technologie dans un précédent projet.

Du fait d'un changement de technologie de base, nous ne pouvions plus utiliser MeteorJS comme convenu, alors nous avons cherché d'autres technologies qui se sont avérées être en plus performantes et mieux adaptées à nos besoins.

Aussi, comme nous avons commencé notre phase de conception par du maquettage, Android Studio nous a également permis de créer les vues de manière plus rapide et de pouvoir travailler directement sur le visuel de l'application.

Lecteur de musique

Le lecteur de musique est légèrement différent de celui dessiné sur les maquettes, car nous nous sommes rendu compte que les musiques apparaissaient trop petites et n'étaient pas assez mises en valeur pour une application dont le thème principal est la musique. De plus, nous avons opté pour un design plus moderne et épuré.

La lecture des musiques s'effectue à l'aide d'un requêtage de la base de données où sont stockés les liens des musiques en questions.

Connexion au serveur et à la base de données

La connexion au serveur s'effectue grâce à la classe DAO. Celle-ci permet de normaliser les envois et la réception des messages dans une structure connue du client et du serveur.

Google Maps API



Google Maps

Pour utiliser la géolocalisation, nous avons utilisé l'API Google Maps qui est un service proposé par Google. Elle nous a permis d'implémenter la carte sur laquelle nous pouvons placer des marqueurs afin d'identifier la position des différents utilisateurs de l'application.

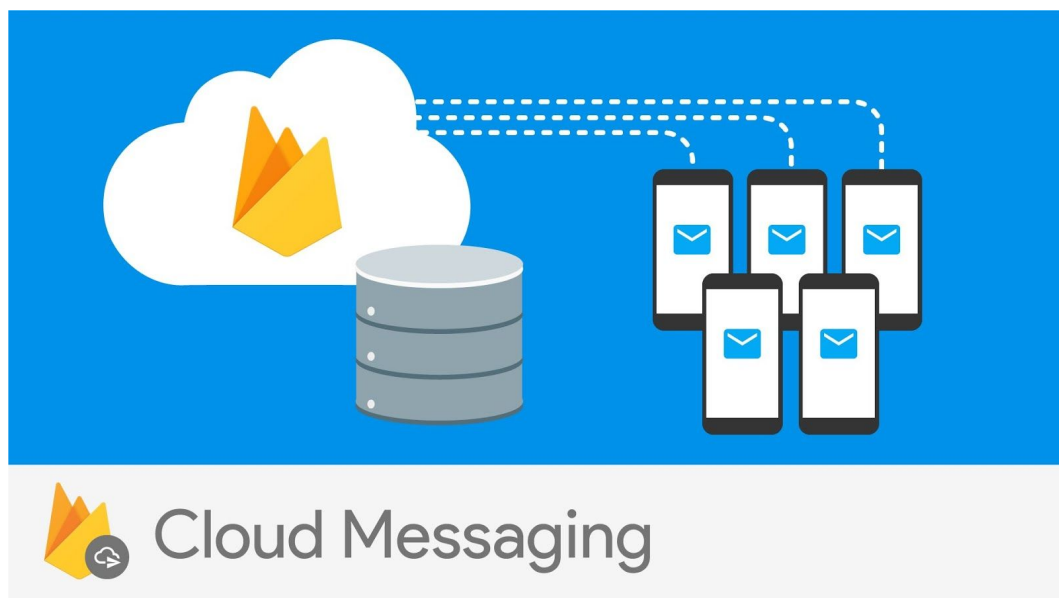
Afin de profiter de ce service, il a fallu générer une clé d'API. Pour cela, nous avons informé le service Google de nos SHA-1 et du package du projet. La clé ensuite récupérée, nous l'avons introduite dans le manifest de l'application via les ressources et ajouté certaines dépendances à compiler dans le Gradle.

Tout d'abord, il était impératif de générer une demande à l'utilisateur pour lui demander les droits de l'application à accéder à sa géolocalisation. Sans cela, le programme irait à l'encontre de certaines valeurs morales et légales. Pour ce faire, nous avons dû rajouter des permissions dans le manifest et créer une pop-up de demande qui redirige l'utilisateur vers les propriétés de l'application afin qu'il puisse changer ses paramètres système.

Ensuite, il a fallu ajouter un marqueur à l'emplacement de l'utilisateur concerné en fonction de sa latitude et longitude et effectuer un zoom sur la carte pour qu'il puisse visualiser les zones proches de lui.

Enfin, et surtout à l'aide de Firebase Realtime Database que nous aborderons plus tard dans ce rapport, nous avons pu faire de même pour les autres utilisateurs.

Firestore Cloud Messaging



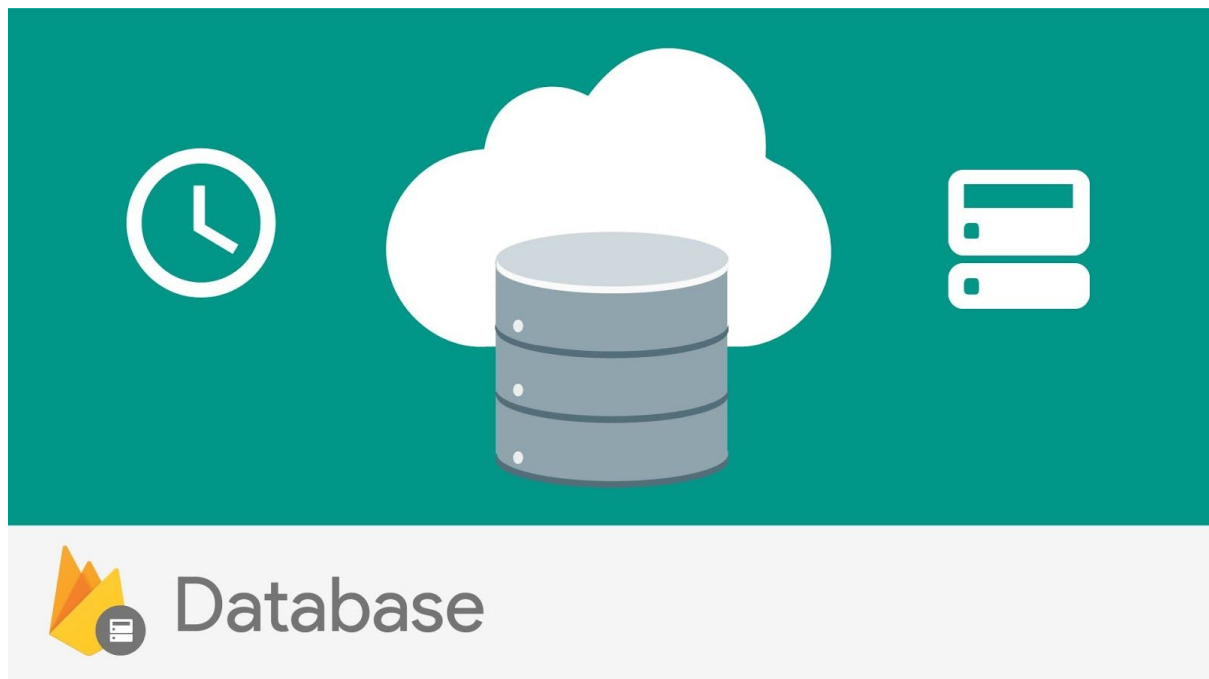
Firebase Cloud Messaging (FCM) est une solution de messagerie multiplateforme qui nous a permis de fournir des messages fiables sans frais.

Pour accéder à ce service, il nous aura fallu faire de même que pour l'API Google Maps. En effet, les services Firebase demandent aussi une clé d'API générée à l'aide d'un SHA-1.

Côté client, il a fallu que nous implantions les classes `FirebaseInstanceIdService` et `FirebaseMessagingService`. L'une permettant de générer un token (jeton) propre à l'utilisateur qui se sert de l'application et l'autre permettant de recevoir les messages envoyés par le serveur.

Les tokens sont créés lors de la connexion de l'utilisateur sur un appareil défini. Ainsi, même si l'application est fermée, le mobile peut recevoir des notifications provenant du serveur pour être informé, par exemple, d'une mise à jour ou de la sortie d'un nouvel album.

Firestore Realtime Database



Firestore Realtime Database (FRD) est une base de données hébergée sur un Cloud. Les données sont stockées en tant que JSON et synchronisées en temps réel à chaque changement de donnée client connecté. Tous les clients partagent une instance Realtime Database et reçoivent automatiquement les mises à jour avec les données les plus récentes.

Simplement en utilisant des instances de `FirestoreDatabase` et `DatabaseReference`, on peut mettre à jour la base de données sur le cloud en modifiant seulement certains éléments du JSON. La méthode `addValueEventListener()` permet de détecter si un changement a été effectué sur une des données que nous avons indiqué.

Nous nous en sommes servi pour les positions des utilisateurs. A chaque fois que ces derniers se déplacent, leur longitude et latitude sont mises à jour et changées dans le JSON. Ainsi tous les autres utilisateurs les voient se déplacer en temps réel.



Serveur

Côté serveur, les fonctionnalités à implémenter sont le requêtage à la base de données, les vérifications, l'envoi de réponses et de notifications.

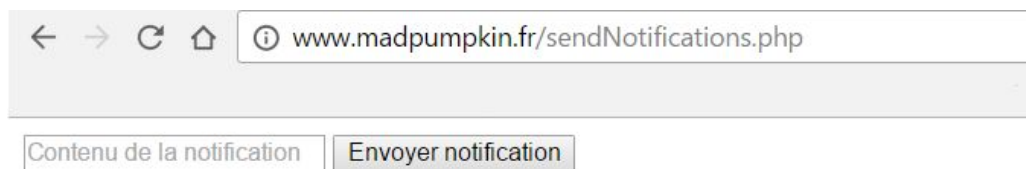
PHP et MySQL sur serveur OVH



L'utilisation de PHP côté serveur à simplifier les connexions à la base de données. Ayant choisis d'utiliser un serveur de ligne plutôt que de laisser l'application tourner en local nous a appris à nous servir de ce genre de service, ce qui était une première pour nous deux. Au début, cela n'a pas été simple, nous n'arrivions pas à savoir si nos accès à la base de données se faisait bien, ou si nos requêtes étaient bien valides.

Ensuite, une fois que la connexion aux scripts PHP via l'application côté client et les interactions avec la base de données furent terminées, nous avons procédé à certaines vérifications, notamment pour l'inscription et le login des utilisateurs.

Firestore Cloud Messaging



← → ↻ 🏠 ⓘ www.madpumpkin.fr/sendNotifications.php

Contenu de la notification Envoyer notification

Nous avons implémenté un script donnant lieu à une page web assez simple. Celle-ci contient un formulaire composé d'un champ de texte et un bouton de validation. Derrière ce formulaire se cache une fonction qui utilise la technologie Firestore Cloud Messaging. Ceci nous permet alors d'envoyer une notification à tous les utilisateurs de notre application au même instant, sous un format JSON dans lequel on peut modifier le titre, le corps du message et certains autres aspects de la notification. Dans notre cas, nous avons choisi de faire vibrer le smartphone de l'utilisateur lors de la réception de la notification.

Sécurité



Pour commencer, nous avons utilisé une technique de cryptage Bcrypt pour stocker les mots de passe dans la base de données. Cette méthode est plus sécurisante que les types de cryptages MD5 utilisés généralement, car il est très facile de trouver des décrypteurs de mot de passe MD5 sur internet. Alors que Bcrypt, en plus de l'utilisation d'un sel pour se protéger des attaques par table arc-en-ciel, il utilise une fonction adaptative qui lui permet d'augmenter le nombre d'itérations pour le rendre plus lent. Ainsi il continue à être résistant aux attaques par force brute malgré l'augmentation de la puissance de calcul.

Coordination du groupe

Pour communiquer et rester en contact même quand nous ne pouvions pas nous voir, nous avons utilisé Slack et Messenger.

Et afin d'accéder à la dernière version du code source de l'application, à n'importe quel moment, nous avons utilisé Github en créant un dépôt distant, accessible par tous, étant publique.



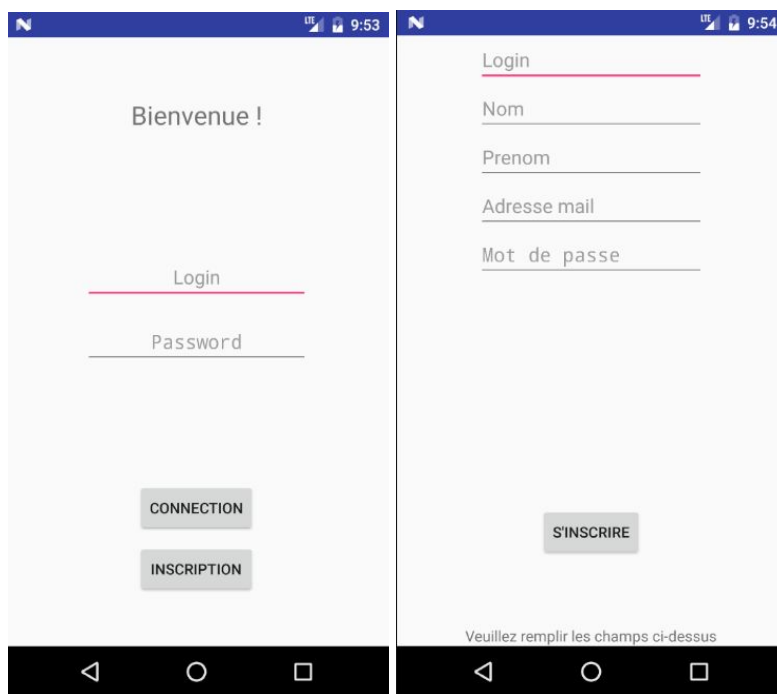
Problèmes rencontrés

Concernant la phase de développement mobile, Moustapha n'en avait fait que pour un précédent projet et était inconnu d'Arthur. C'était l'un des principaux challenges de ce projet. Mais nous nous sommes très vite confrontés à un grand nombre d'autres technologies, telles que les API Google et Firebase ou encore l'utilisation d'un serveur distant avec OVH. Le changement de technologie en cours de projet nous a également un peu déstabilisé face à la charge de travail qu'il restait à accomplir.

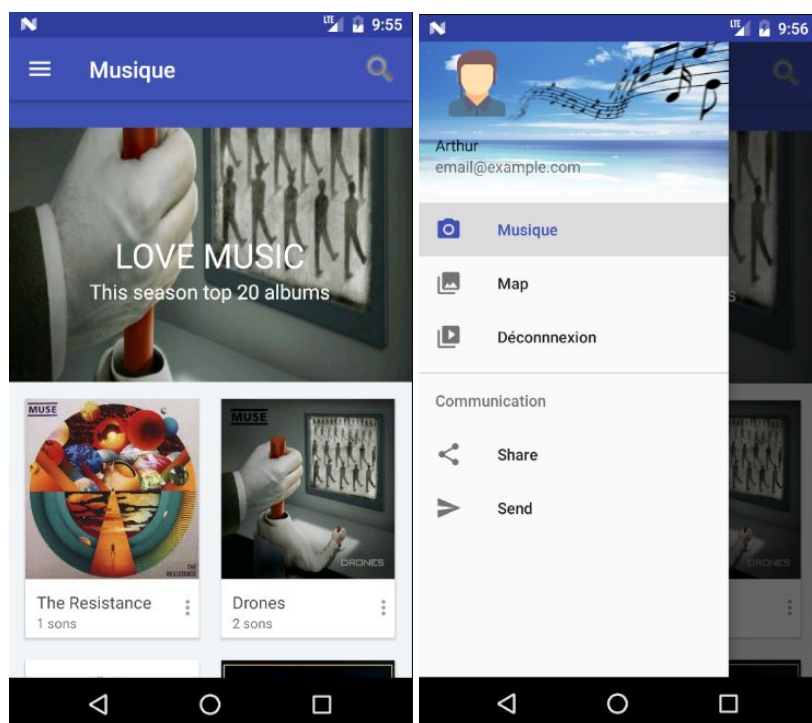
Aussi, nous avons passé beaucoup de temps à apprendre à générer les codes SHA-1 nécessaires pour obtenir les clé d'API et comprendre le fonctionnement de ces frameworks. Il y a eu également une phase d'intégration entre nos travail qui a pris longtemps à se résoudre et se solder par le résultat que nous avons aujourd'hui.

Aperçu de l'application

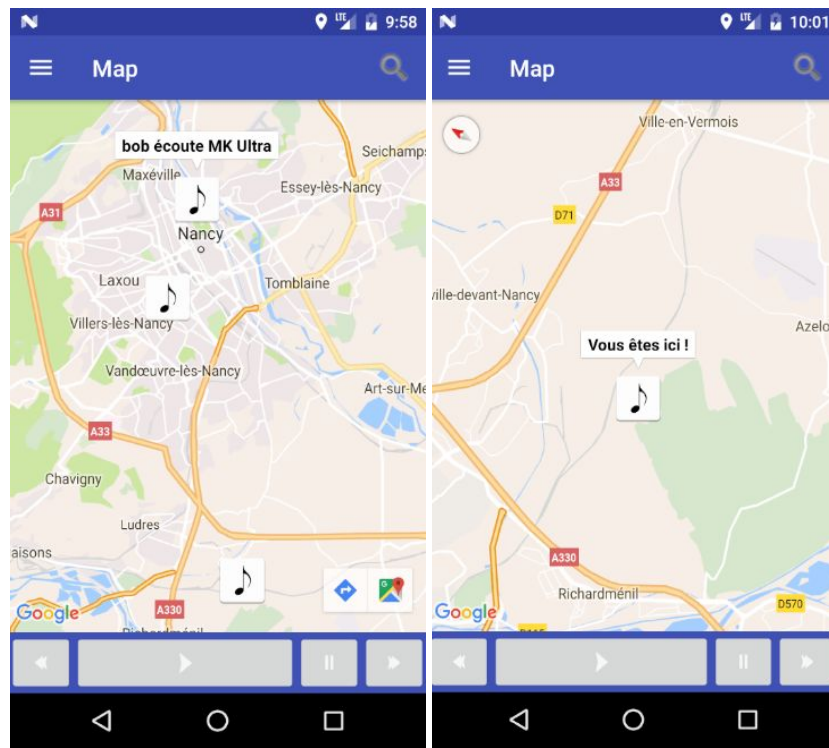
Connexion et inscription



Lecteur de musique



Carte de partage



Nous avons essayé le plus possible de coller aux maquettes réalisées lors de la phase de conception. Certains éléments, comme la page de lecture de musique sont légèrement différents pour un rendu visuel et une navigation plus agréable et ergonomique.

Conclusion

Ce projet nous a permis d'appliquer certains concepts et méthodes vues en cours dans le cadre de la réalisation d'une étude concrète, comme dans l'un des cours d'application de systèmes distribués où nous avons pu aborder le développement mobile avec Android Studio. Mais il nous a surtout appris à aller chercher des informations concernant de nombreuses et diverses technologies à utiliser, pour s'adapter au mieux à nos besoins, et à apprendre à nous en servir.

Ce projet nous a particulièrement intéressé du fait que les technologies mobiles sont de plus en plus présentes dans le monde professionnel et pourtant très peu étudiées en cours.

Nous espérons avoir à nouveau l'occasion de les utiliser à l'avenir.