Pokecard

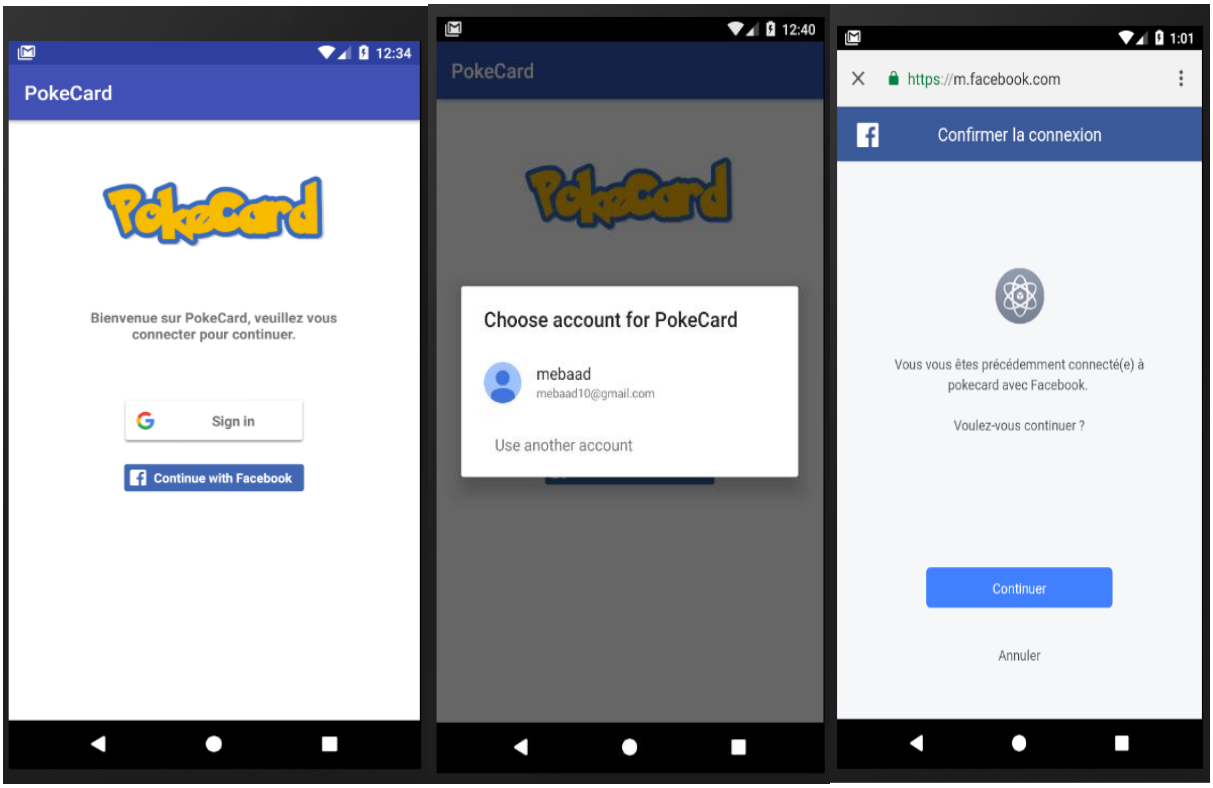
Documentation fonctionnelle et technique

Adam Mebarki / Florian Van Den Berghe

I Partie Fonctionnelle

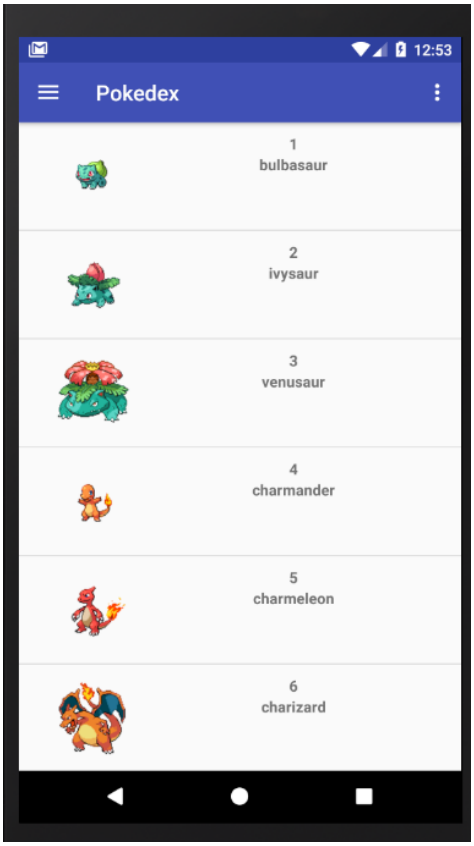
Présentation des fonctionnalités apportées :

L'application possède deux connexions : Facebook et Google. L'utilisateur pourra choisir l'une des deux.

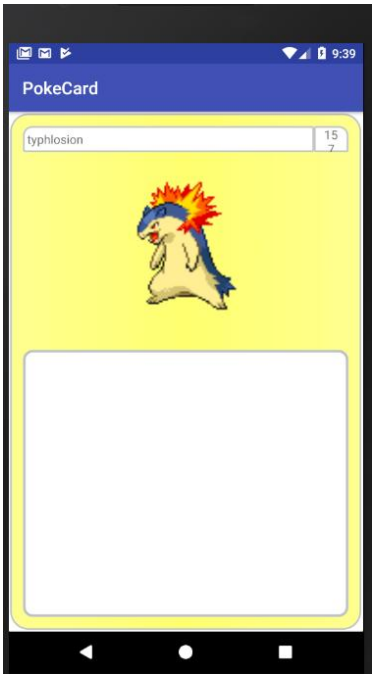


Ces connexions sont gérées par Firebase, qui est un ensemble d'hébergement pour n'importe quel type d'application. Il héberge notamment les authentifications sociales. Elles sont ainsi gérées et sécurisées via l'hébergeur.

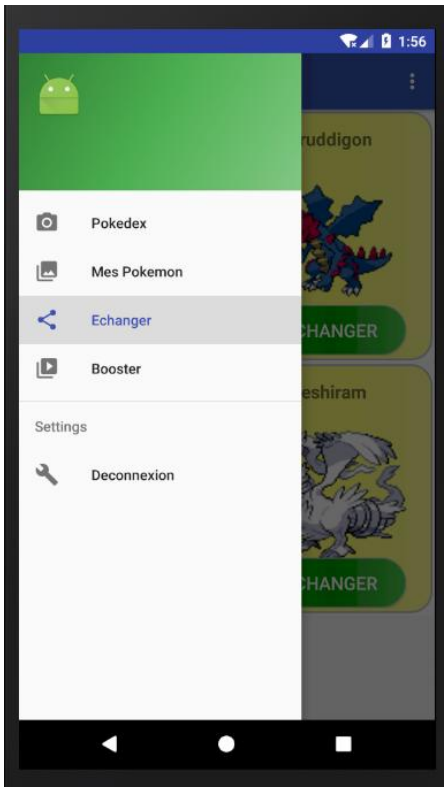
Une fois la connexion effectuée par l'utilisateur, l'application affichera automatiquement le Pokedex avec la liste complète des Pokémons actuellement disponibles.



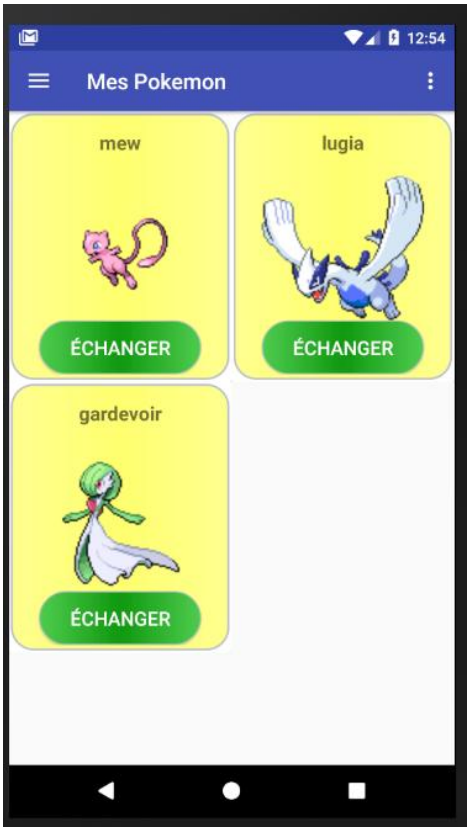
Il pourra notamment cliquer sur un Pokémon en particulier afin d'afficher les détails de celui – ci :

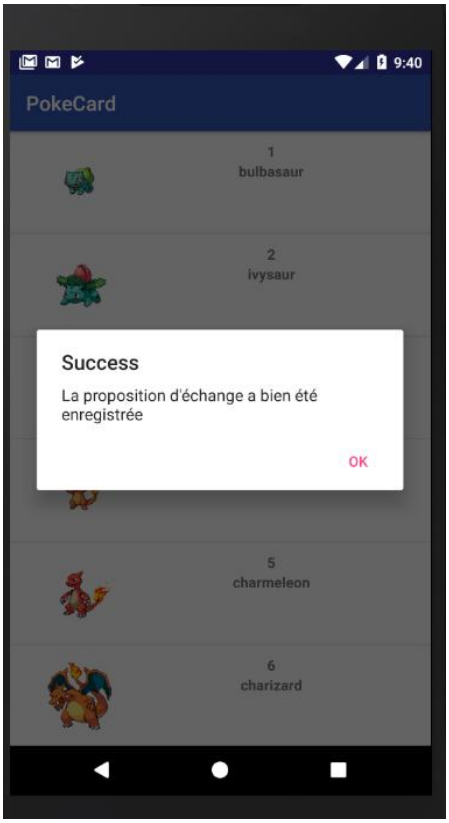


L’utilisateur pourra par la suite, grâce à la navigation drawer, choisir de voir ses Pokémons, de voir la liste des échanges, d’ouvrir un booster ou de se déconnecter.

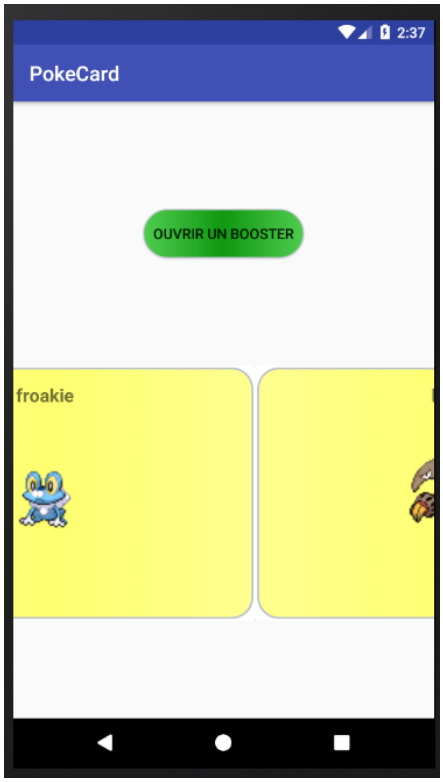


Lorsque l’utilisateur cliquera sur « Mes Pokémon », il aura une page similaire à celle-ci :





L'utilisateur aura la possibilité d'afficher les détails des Pokémons qu'il possède et il pourra proposer un échange avec celui – ci via le bouton « ECHANGER ». Cette dernière action lui permettra de créer une requête d'échange en inscrivant le Pokémon qu'il veut échanger et le Pokémon qu'il souhaite en retour. Il pourra choisir le Pokémon qu'il souhaite grâce à une liste répertoriant tous les Pokémons existants.



Une page « Booster Pack » a été crée. Elle offre la possibilité à l'utilisateur d'ouvrir un booster et de recevoir 5 Pokémons aléatoires qui seront inclus dans sa liste. Il pourra les échanger avec d'autres utilisateurs.

II Partie Technique

L'API est codée en PHP et l'architecture se base sur le MVC (Modèle Vue Controller) qui a été adapté à notre façon pour le projet. Pas de raisons particulières pour ces deux choix. Le tout est géré par Composer qui est un gestionnaire de dépendances libre écrit en PHP. Pas de raisons particulière pour son utilisation.

Le projet se base sur un projet vu en cours. Il y avait clairement de meilleurs solution mais très peu on été proposé en cours par nos professeurs et étant des développeurs mobiles et non web, nous n'avions pas l'expérience pour créer une API performante avec un

langage web. De plus le serveur est difficile à lancer. A l'heure actuelle il ne fonctionne que sur le PC ou il a été mis en place. L'application ne peut pas fonctionner sans le serveur.

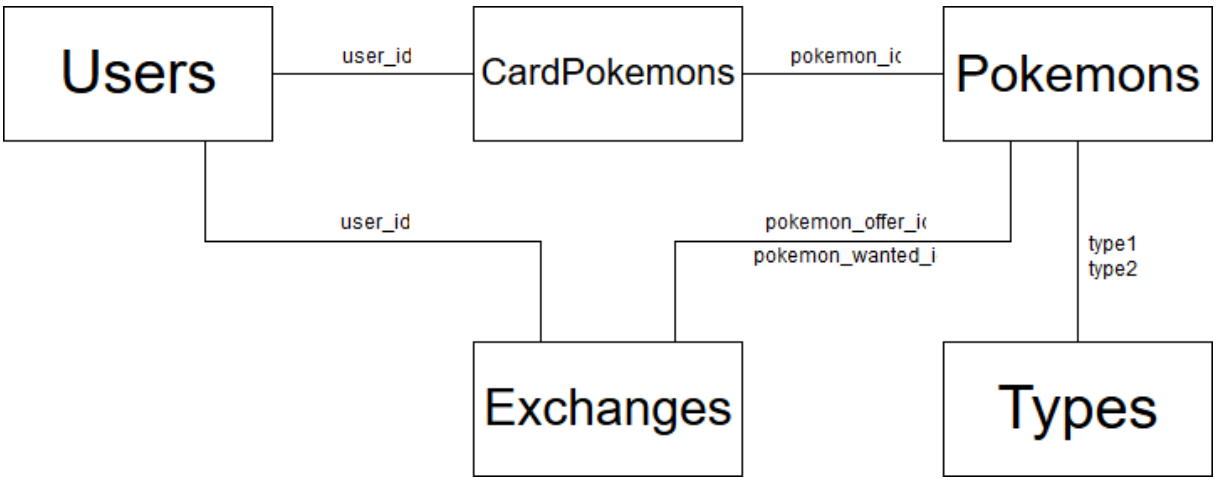
Le projet est réparti en plusieurs dossiers. Les plus importants se trouvent dans le dossier src. Le dossier var contient les logs et les erreurs liés à l'API. Le dossier vendor contient toutes les dépendances.

Le dossier possède deux sous dossiers (Users et Pokemon). Chacun de ces deux dossiers se base sur l'architecture MVC modifiée pour correspondre à nos besoins :

- Controller : possède un fichier php contenant la liste des méthodes utilisées par les routes définies dans le fichier routes.php qui sont appelées par l'application pour recevoir ou envoyé des données.
- Entity : possède un fichier PHP contenant le modèle. Ces modèles sont utiles pour la création, la sauvegarde et l'envoi d'utilisateurs et de Pokémon à l'application ou à la base de données.
- Repository : contient la liste des fonctions appelées par les méthodes du Controller. La plupart des fonctions interagissent avec la BDD. Ces méthodes se chargent de traiter les données reçu par l'application et de les ajouter à la BDD, et inversement.
- Service : contient la liste des fonctions appelées par les méthodes du Controller. Chaque fonction appelle l'API RESTFUL sur Internet Manipule les données reçues et les retourne au Controller qui les envoie à l'application.

Toutes les données sont envoyées et reçu avec le format JSON. L'intérêt de ce format est sa facilité pour le traitement des données.

Afin de stocker les données utilisateurs, une base de données était nécessaire.



La table Users contient la liste des utilisateurs de l'application avec leur identifiant, leur adresse mail et leur nom ou pseudo.

La table Pokemons a été créée. Il était spécifié qu'il n'y avait aucun stockage de Pokémons. Cependant avec cette table, l'utilisateur aura accès aux informations complètes de ces Pokémons sans dépendre de l'API RESTFUL. Il y a eu beaucoup de problème lié à POKEAPI. Le principal était l'erreur 504 Gateway Timeout. Et durant les phases de développement il était difficile de pouvoir montrer la liste des utilisateurs en dépendant uniquement de POKEAPI. De plus, grâce à ce stockage des Pokémons de l'utilisateur, le traitement du JSON est bien plus rapide car les informations utiles sont uniquement stockées dans la BDD.

La table Types regroupe tous les types de Pokémons. Elle était prévu pour être utilisée pour des fonctionnalités diverses comme le tri, la couleur de fond, le design de certains éléments.

La table CardPokemons regroupe les identifiants des utilisateurs et des Pokémons de la BDD. Elle permet de savoir les Pokémons que les utilisateurs possèdent.

La table Exchanges contient la liste de tous les échanges qui ont été créés par les utilisateurs. Elle comprend l'identifiant de l'utilisateur qui a posté une annonce d'échange, l'identifiant du Pokémon qu'il veut échanger et enfin l'identifiant qu'il souhaite en retour. Une variable a été ajoutée afin de savoir si un échange à été conclu ou non.

L'API a été développée sous un Environnement Windows avec le bash d'Ubuntu. En cours, on a appris a installé un serveur sur Mac, cependant, je ne possède pas de mac. Il a fallu trouver un moyen pour pouvoir développer chez nous sans dépendre de la salle de l'IUT.