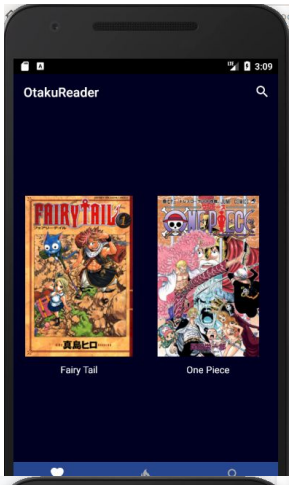
Rapport Android Commun TA/JEUDY

TA Michael
JEUDY Jordan

Dans le cadre de l'UE TPALT, nous avons créé une application Android (**JAVA**) afin de pouvoir lire des mangas, nous utilisons pour ce faire deux API, une API "**Manga Eden**", qui est notre API principale, et notre propre API (hébergé sur un serveur **HEROKU**) afin de pouvoir ajouter deux fonctionnalités, une recherche filtre et la récupération des mangas les plus populaires. Nous avons aussi ajouté la possibilité de changer l'orientation du téléphone.



Dans notre application, nous pouvons ajouter des mangas dans nos **favoris** et ainsi avoir un accès plus rapide aux mangas qui nous plaisent. Nous avons aussi ajouté une recherche de mangas, il s'agit d'une **recherche** telle un **filtre**, en faisant la recherche du mot "piece", on retrouvera tous les mangas contenant ce mot.
On voit aussi dans ce screenshot, qu'il est possible d'accéder aux "**Trending**", il s'agit d'une **liste statique** que l'on récupère depuis notre propre API.
Depuis les favoris, les tendances ainsi que depuis la recherche, il est possible en cliquant sur le manga d'accéder à la liste de ses **chapitres**.

Cette vue correspond à la liste des chapitres, on peut ajouter un manga à ses favoris en cliquant sur **l'étoile** dans le coin droit, un court descriptif du manga (3 lignes) est disponible, il est possible d'avoir la totalité de la description en cliquant sur la flèche sous celui-ci.
On remarque qu'il y a, à côté de certains chapitres, deux icônes, le livre ouvert sert à indiquer à l'utilisateur, qu'il a quitté l'application en **pleine lecture**, la seconde icône (la couverture du livre) représente le fait qu'on a **fini de lire le chapitre**.
Il est bien évidemment possible d'accéder à la lecture du chapitre en cliquant sur celui-ci.

Nous avons décidé que le sens de lecture correspondrait au vrai sens de lecture des mangas, soit de droite à gauche. Lors de la lecture, nous avons ajouté un **slider** (en bas de l'écran), afin de pouvoir choisir de façon rapide la page qu'il souhaite lire.

TA Michael

Dans ce projet, je me suis dans un premier temps occupé des appels API. Pour ce faire, j'ai utilisé le framework "Retrofit" que j'ai trouvé assez intéressant. En effet, il nous permet de faire les appels API de façon simple. Il nécessite uniquement une interface pour dire quels sont les appels possibles, un POJO dans lequel le résultat sera récupéré (désérialiser) et un Builder qui contient un Singleton représentant l'endpoint.

Je me suis après cela occupé de la vue des chapitres, cette vue contient une ListView avec un header, permettant ainsi lors d'un défilement de la liste, que les items ainsi que le header le soit. Celui-ci contient l'image du manga, son nom, l'auteur ainsi qu'un descriptif court du manga.

Nous avons décidé à ce moment-là que lors du clique sur un chapitre, que l'id du chapitre, l'id du manga et la liste des chapitres soient envoyés à l'activité de lecture via un Intent. En effet, nous déléguons l'appel à l'API pour récupérer les pages à la vue suivante. Ainsi, lors d'un clique, nous changeons directement de vue et avons un ProgressBar en attendant que les pages soient récupérées.

Je me suis ensuite occupé de la vue pour les listes des mangas, il s'agit d'un RecyclerView avec un GridLayoutManager afin de pouvoir avoir un affichage tel une grille. En effet, après m'être renseigné sur GridView, je me suis aperçu que celui-ci était legacy, d'où notre choix de RecyclerView. Par la suite, cette activité a été transformée en Fragment. La liste des mangas affiche les mangas les plus populaires, nous récupérons cette liste avec notre propre API.

Après cela, je me suis tourné vers la ToolBar afin d'y ajouter la recherche, Jordan avait dans un premier temps implémenté sur notre serveur la recherche exacte d'un manga, c'est-à-dire qu'on devait connaître le nom exact du manga pour le trouver. J'ai ajouté à cela la recherche par filtre, ainsi nous pouvons maintenant rechercher des mangas plus librement. J'ai cependant ajouté une contrainte sur la taille de la recherche, en effet faire une recherche des mangas contenant "a" pouvait renvoyer une liste de taille conséquente, ce qui est gourmand pour le cache et assez long si la connexion ne suit pas. Pour afficher les résultats d'une recherche, j'ai dans mon layout, "include" celui de la liste pour avoir un affichage similaire.

Nous avons décidé après cela d'ajouter le changement d'orientation, pour ce faire, j'ai empêché de refaire un appel à l'API lors d'un changement d'orientation et j'ai sauvegardé les différents résultats des appels dans l'instance avant le changement, pour ce faire, j'ai dû faire en sorte que nos objets implémentent l'interface "Parcelable" qui est similaire à "Serializable". Dans un second temps, j'ai aussi indiqué à ma GridLayoutManager d'afficher 4 colonnes au lieu de 2 en cas d'orientation paysage.

JEUDY Jordan

La principale fonctionnalité de notre application est de pouvoir lire des manga, je me suis donc occupé de cette partie en premier. J'ai du trouvé une solution techniques au problèmes suivant, changer de page par un "swipe" et pouvoir gérer un grand nombre de pages (jusqu'à 200 pages pour un chapitre). J'ai donc utilisé un ViewPager auquel j'associe un FragmentStatePagerAdapter qui est selon l'API Android l'implémentation de PagerAdapter recommandé pour gérer un grand nombre de page.

Une application de lecture d'ouvrage digne de ce nom offre la possibilité à l'utilisateur de pouvoir zoomer sur la page courante que ce soit en "double tap" ou un "pincement" sur l'écran. Malheureusement l'API Android n'offre pas nativement un tel widget, dans notre cas les pages sont des images et les ImageView ne convenait pas. Des implémentations d'ImageView avec zoom existe néanmoins sur Github et on c'est donc arrêté sur une implémentation appeler PhotoView qui réponds à notre problème.

Une fois la partie API implémenter avec Retrofit, j'ai simplement fais des requêtes pour récupérer les images et les placer dans les PhotoView. Les requêtes sont effectué à l'aide de Glide qui est une librairie de chargement d'image distante. Charger du contenus depuis le réseau implique de gérer les éventuels problème réseau. On ajoute donc au layout un bouton "Refresh" qui permet de réessayer le chargement d'une image.

Ensuite, je me suis occupé de la partie concernant la gestion des favoris et de l'historiques, des données persistantes donc. On devait stocké comme information les manga favoris, ainsi que les chapitres lu ou en cours de lecture. Dans un premier temps, j'avais implémenté une solution qui stockait les données dans un simple fichier Json. Mais les insertions, suppressions, mis à jour des données étaient laborieuse, coûteuse, et ne passerait sans doute pas à l'échelle. J'ai supprimé cette méthode et j'ai hésité entre SQLite et les SharedPreference. On a finalement décidé de prendre SQLite car un véritable Otaku peut lire un très grand nombre de manga et on doit pouvoir stocker ces données efficacement. On a donc choisi d'utiliser la librairie Room qui nous a permis de créer rapidement notre base de données. Dans notre application on a fait le choix de ne pas sauvegarder la dernière pages lu car cela impliquerait de faire un UPDATE à chaque changement de page.

Quand Michael a terminé la page Trending j'ai simplement fait un include pour créer le layout de la page Favorites et remplacé l'appel à l'API mangaeden par un accès à la base de données.

Enfin je suis retourné sur l'activité de lecture d'un chapitre pour y ajouter une SeekBar et ainsi avoir afficher à l'écran le nombre de page ainsi que la possibilité de changer de page plus rapidement que par de multiple "swipe". Puis j'ai terminé en ajoutant le support du changement d'orientation à cette activité.