Anqi WEI
Yanqiu YANG
Siyuan WANG

Introduction

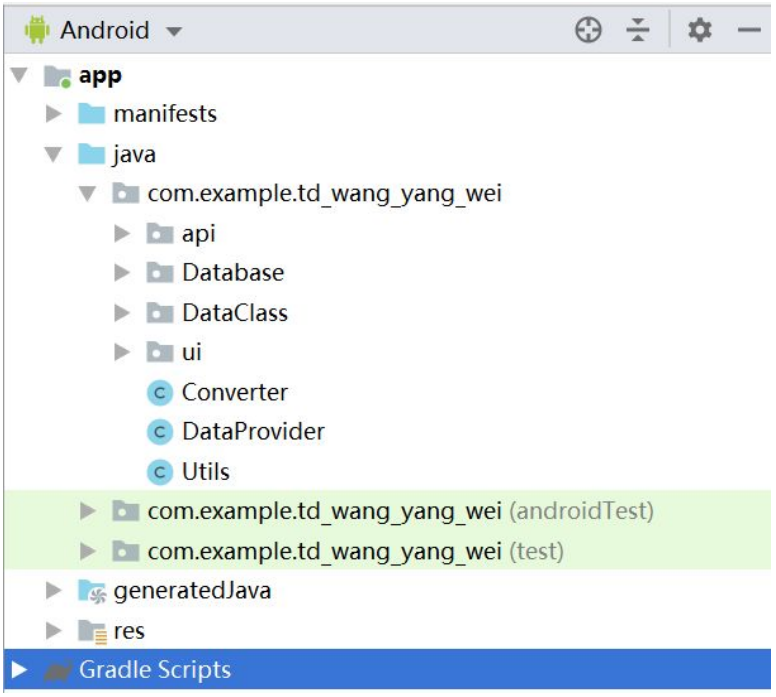
Dans ce TP, on voudrait améliorer l'application développée à la séquence 2 en proposant un mode offline. C'est-à-dire, quand on a de réseau, on récupère les listes et les items de l'api, on les stocke dans un "tableau" dans la mémoire locale. Au démarrage de l'application, si le réseau n'est pas disponible, l'application proposera de manipuler les données localement, une fois le réseau disponible, le nouveau "tableau" sera mis à jour auprès de l'API.

Pour réaliser ce but, on utilise l'ORM Room et met en place d'une persistance de données sur le téléphone à l'aide d'une base SQLite. On transforme tous les modifications dans les requêtes en les modifications dans le SQLite.

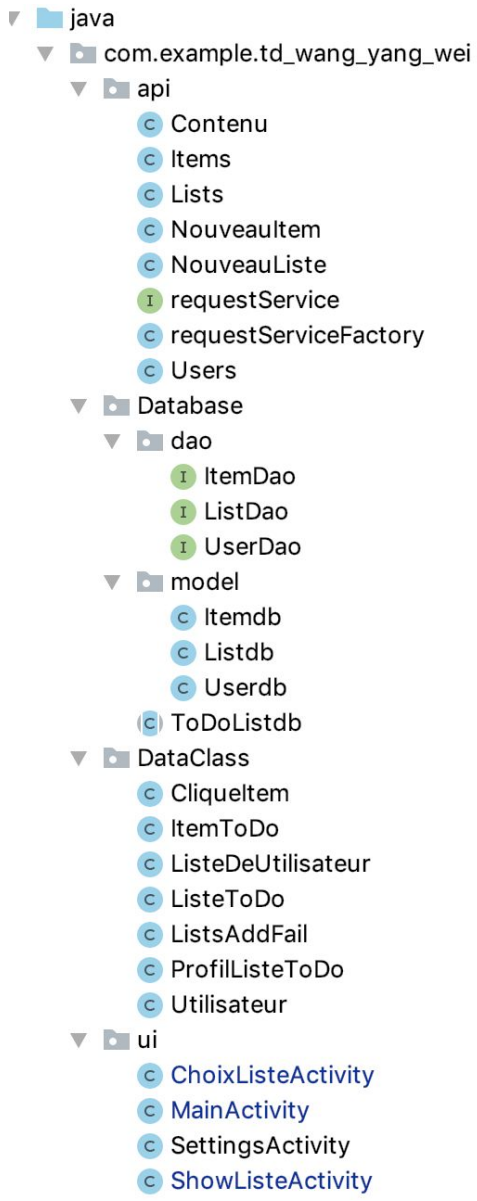
Remarque : le programme de séquence 1 est dans la branche master, le programme de séquence 2 est dans la branche WEI, le programme de séquence 3 est dans la branche sequance3_YANG.

Analyse

En bref, le programme que l'on réalise dans la séquence 3 réalise la même fonction que ceci des deux premières séquences, on change la transmission de données. Donc, la structure de programme ne change pas beaucoup.



Anqi WEI
Yanqiu YANG
Siyuan WANG



Maintenant, notre application devient de plus en plus compliquée : elle contient la partie qui fait l'interaction entre l'utilisateur et l'application (ui), la partie qui nous simplifie le processus de programme (DataClass), la partie qui fait l'interaction entre l'application locale et l'api (api) et la partie qui fait l'interaction entre l'application et database (Database).
On ajoute aussi 3 outils class : Converter, DataProvider et Utils pour nous aider. Dans le Converter, on fait la transformation mutuelle entre les modèles (type utilisé pour database) et les listes et items. Dans le DataProvider, on définit les méthodes de synchronisation syncGetLists() et syncGetItems().

Voici la conception de programme.

Tout d'abord, on change le processus d'obtenir le data et les afficher. Dans le class DataProvider on a créé des méthods en utilisant l'interface requestService pour les data, et puis les enregistrer dans le database locale.

ToDoList Séquence 3 offline

Anqi WEI
Yanqiu YANG
Siyuan WANG

Quand l'utilisateur lance l'application, il est dans l'interface de MainActivity. On construit une fonction getProfiles() pour créer une listeDeUtilisateur qui stock les utilisateurs locals. Comme on transmet les données par l'Internet, il faut vérifier le réseau avant d'action de l'utilisateur. Pour créer un nouveau utilisateur en appuyant sur SIGNIN, il faut un hash code, donc il faut listeDeUtilisateur.getUtilisateurs().isEmpty(). Pour la connexion, On guide l'utilisateur à entrer le pseudo et le mot de passe. Le dernier pseudo saisi est automatiquement renseigné dans le champ de saisie. Si le mot de passe est correct, l'utilisateur entre dans l'interface de ChoixListeActivity en appuyant sur le bouton OK!.

Dans l'event de clique de bouton OK, il y a pluisieurs case. Quand c'est la première fois on connecte, le pseudo n'existe pas dans le ListeDeUtilisateur. On utilise le méthod syncGetUserId() pour obtenir le data user et enregistrer dans le database user. En meme temps on utilise le Id on a obtenu pour obtenir et enregistrer les listes de cet utilisateur en utilisant le méthod syncGetLists. Ensuit on obtient les id de toutes les listes de cet utilisateur. Et puis en utilisant le méthod syncGetItems(), on peut enregistrer les données des items.

Pour l'instant, on a stocké les données de user et les données de liste et les données de items dans notre database. Les executions ensuites vont etre executées dans le database local.

une autre case, ce n'est pas la première fois que cet utilisateur connecté. C'est-à-dire que son data est déjà stocké dans notre database. Ensuite on va entrer l'activité ChoixListeActivity en utilisant méthod ConvertToListe(), Dans la construction de fonction ConvertToListe(), on utilise des Intent.putExtra() pour la transmission de donnés. Une fois que l'on entre cette activité. On affiche le donné que l'on a stocké dans l'activité dernière en utilisant le method syncGetAll().

En cliquant sur le menu mis sur l'interface de MainActivity, l'utilisateur obtient tous les utilisateurs de préférence. Dans cet interface, l'utilisateur peut cliquer un profil pour revenir à l'interface de MainActivity avec ce profil automatiquement renseigné dans le champ de saisie.

En cliquant sur un Liste, l'utilisateur peut entrer dans la liste d'Items de ce Liste. L'application affiche la liste de Items de ce Liste.

A cause de bug et que le temps est un peu limité, on n'a pas réssuit de résoudre tous les bugs. Il y a des problèmes sur comparer et merger les données.

Conclusion

Pendant cette séance, basé sur l'application réalisée dans la séquence 2, on essaye à manipuler les données localement quand le réseau est indisponible. On utilise un SQLite à stocker les listes et items et transforme tous les requêtes en langue de la base de données. Cette approche évite non seulement l'interruption du réseau, mais réduit également la charge de modification de Item à chaque fois.

Anqi WEI
Yanqiu YANG
Siyuan WANG

Jusqu'à présent, nos programmes sont devenus de plus en plus complets et l'expérience utilisateur est devenu de plus en plus excellente.

perspective ?

Bibliographie

https://www.cnblogs.com/dolphin0520/p/3949310.html
http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2017/0726/8249.html
https://developer.android.com/reference/java/util/concurrent/Future