BOYER Nathan
BRISSON Ulysse

PMR Séquence 3 : Compte-rendu - Application ToDoList

Sommaire

Sommaire **0**

Introduction **1**

Analyse **2**

Perspectives **4**

Bibliographie **5**

Lien vers le dépôt Github : https://github.com/Shysto/ToDoList/tree/sequence3

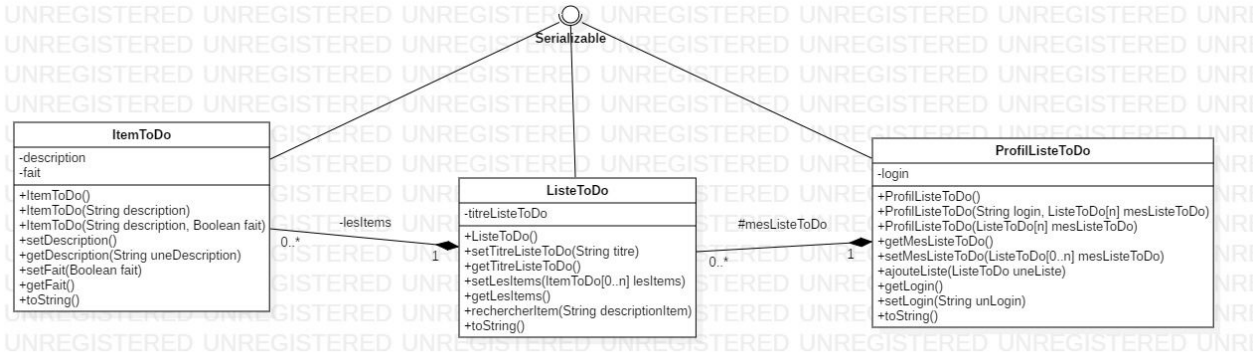
BOYER Nathan
BRISSON Ulysse

Introduction

L'objectif du projet de cette séquence était d'améliorer notre application Android de gestion de TodoLists sous Android Studio, cette application ayant été réalisée lors des deux séquence précédente.

Pour ce faire, nous devons désormais stocker et gérer les données dans un "cache" organisé sous la forme d'une base SQLite.
Si le réseau n'est pas disponible, l'application doit proposer de manipuler les données en cache à son démarrage. Une fois le réseau disponible, l'état des items de toutes les listes modifiées doit être mis à jour auprès de l'API.

Pour rappel, notre application devait respecter le diagramme de classe UML suivant pour la gestion des objets (profil utilisateur, TodoList et tâche) de notre application :



L'ensemble des données associées à un utilisateur sont désormais stockées auprès d'une API Rest, hebergée sur le réseau et accessible à l'adresse de base : http://tomnab.fr/todo-api/
La documentation de l'API est accessible depuis ce lien :
https://documenter.getpostman.com/view/375205/S1TYVGTa?version=latest

Analyse

L'ensemble des fichiers sources implémentant nos objets tels que décrit par le diagramme de classes sont regroupés sous le package **`modele`** de notre application. Chaque classe implémente l'interface Serializable afin de pouvoir utiliser la sérialisation/désérialisation de données en JSON.

Afin de répondre aux différentes contraintes qui doivent être satisfaites par notre application, nous pouvons la découper en quatre activités :

- MainActivity : cette activité représente l'activité principale de l'activité, qui s'ouvre lors du lancement de l'application. Elle vérifie l'état du réseau en continu (ConnectivityManager). Elle permet à l'utilisateur de saisir son pseudo et son mot de passe, tout en proposant par défaut le dernier pseudo saisi. Une toolbar permet d'accéder à une seconde activité, SettingsActivity. On sauvegarde le hash de connexion auprès de l'API dans les préférences de l'activité, et ce même paramètre est transmis à l'activité ChoixListActivity, ouverte lors du clic sur le bouton OK. Ce dernier bouton n'est utilisable que lorsque l'appareil est connecté à un réseau, et lors d'un clic, il envoie une requête de connexion auprès de l'API en utilisant le pseudo et le mot de passe saisi.
  Si la requête réussit, la prochaine activité s'ouvre. Si aucun réseau n'est détecté, un bouton permettant de gérer les données en "cache" apparaît. Les données sont alors modifiées depuis la BDD locale. Pour que cette BDD soit remplie, il faut avoir récupéré des infos depuis l'API au moins une fois. Les listes de la BDD sont mises à jour lors de l'affichage de l'activité ChoixListActivity (avec réseau), et les items d'une liste lors du clic sur la liste concernée (avec réseau).

- SettingsActivity : cette activité affiche le pseudo affiché par défaut (dernier pseudo saisi par l'utilisateur) en utilisant des fragments (classe SettingsFragment), et permet de modifier l'URL de base dirigeant vers l'API.

Les fichiers sources MainActivity.java, SettingsActivity.java et SettingsFragment.java sont placés dans le package **`accueil`**.

- ChoixListActivity : cette activité représente la liste des TodoLists affectée au pseudo rentré, dans une RecyclerView. Elle permet à l'utilisateur de saisir le titre d'une nouvelle TodoList afin de la créer et de l'ajouter à l'ensemble des ses TodoLists lors du clic sur le bouton OK (seulement si le réseau est détecté). Elle permet de plus de générer une nouvelle activité, ShowListActivity, lors d'un clic sur le titre d'une des TodoLists. Toutes les TodoLists sont récupérées depuis l'API si l'utilisateur est connecté à un réseau, ou depuis la BDD sinon, et l'ajout d'une nouvelle liste s'effectue aussi au sein de l'API (si réseau détecté), en temps réel.

- ShowListActivity : cette activité représente la liste des tâches (items) que contient la TodoList sélectionnée sur l'activité précédente, dans une RecyclerView. Chaque tâche est représentée par une checkbox, dont la description correspond à celle de la tâche associée, et dont la case est cochée si et seulement si la tâche associée est considérée comme dans l'état "fait". Un clic sur une des tâches inverse l'état de la checkbox et de la tâche correspondante. Il est aussi possible d'ajouter un item à la liste courante (seulement si connecté au réseau), et toute modification (modification d'un état ou ajout d'un item) se répercute en temps réel au sein de l'API (si réseau) et de la BDD. De même la liste des items est récupérée depuis l'API ou depuis la BDD le cas échéant.

Les fichiers sources ChoixListActivity.java, ShowListActivity.java et Library.java sont placés dans le package **`recycler_activities`**.

Ce package contient un sous-package nommé **`adapter`**, qui contient les deux fichiers sources correspondant à l'implémentation des Adapters associés aux items de nos RecyclerViews, et qui gère dont les fonctions propres à cette classe.

Nous avons finalement choisi de passer par l'implémentation d'une interface pour gérer les écouteurs d'événements sur les items de nos RecyclerViews, une solution qui nous paraissait plus "propre".

Le package **`api`** contient les deux fichiers sources correspondant à l'interface de connexion auprès de l'API (TodoApiServiceFactory.java) et à la liste des requêtes disponibles vers cette API (TodoApiService.java). Il contient aussi un autre package **`response_class`** implémentant des classes correspondant aux différents objets renvoyés par l'API dans ses réponses, afin de pouvoir récupérer les éléments importants de ces réponses, et initialiser nos propres objets. Elles permettent aussi de définir les tables de notre BDD.

Le package **`bdd`** contient la classe permettant d'implémenter la base de données en local, ainsi que trois interfaces instanciant chacune des méthodes (récupération, ajout et mise à jour de données) de requêtes SQL auprès de la BDD.

Perspectives

- Gérer les TodoLists partagées entre utilisateurs → pour l'instant, chaque utilisateur a un ensemble de TodoLists qui lui est propre, mais on pourrait avoir envie que deux utilisateurs ou plus soient inscrits à une même TodoList (dans le cadre d'un projet par exemple).

- Implémenter des fonctionnalités facultatives, proposées par l'énoncé ou non :
    - Ajout de champs dans les préférences de l'application (vider l'historique, onglet "à propos"...);
    - Proposer une complétion automatique pour les pseudos;
    - Permettre la suppression de TodoLists, de tâches (items) ou d'utilisateurs;
    - Développer l'application en plusieurs langues;
    - Proposer des interfaces hommes-machines différentes selon l'orientation portrait ou paysage;
    - Utiliser des champs de saisie texte vérifiant la validité des données saisies (input type, casse…);
    - Connecter automatiquement l'utilisateur sans passer par l'activité de connexion en stockant pseudo/passe dans les préférences
    - Ajouter un menu "déconnexion" dans toutes les activités, qui fera revenir à l'activité de connexion
    - Améliorer les items des listes pour leur associer des liens, permettre de cliquer sur un item pour afficher le lien dans un navigateur au choix de l'utilisateur à l'aide d'un intent implicite..
    - Permettre la création de listes ou d'items hors ligne.

Bibliographie

- Documentation sur les fragments pour les préférences
  https://developer.android.com/reference/androidx/preference/PreferenceFragmentCompat

- Liste des dépendances nécessaires ajoutées à notre application
  - Pour les fragments `'com.android.support:preference-v7:28.0.0'`
  - Pour les fichiers JSON `'com.google.code.gson:gson:2.8.2'`
  - Pour les RecyclerView `'androidx.recyclerview:recyclerview:1.0.0'`
  - Pour l'utilisation de Retrofit `'com.squareup.retrofit2:retrofit:2.6.0'` et `'com.squareup.retrofit2:converter-gson:2.4.0'`
- Code correspondant à la vérification de l'état du réseau : Thomas Bourdeaud'huy (Moodle)
- Code correspondant à la gestion de l'interface de connexion à l'API, à l'envoi de requêtes et au traitement de la réponse : Mohammed Boukadir (Github)
- Code correspondant à l'implémentation, à la gestion et à l'accès à une base donnée (ROOM) : Mohammed Boukadir (Github + diapo)