BOYER Nathan
BRISSON Ulysse

PMR Séquence 1 : Compte-rendu - Application ToDoList

Sommaire

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

BOYER Nathan
BRISSON Ulysse

Introduction

L'objectif du projet de cette séquence était de concevoir une application Android de gestion de TodoLists sous Android Studio, cette application devant respecter certaines contraintes et offrir différentes fonctionnalités, obligatoires ou facultatives.

Pour rappel, une TodoList est une liste de tâches (des items) qui peuvent être dans un état "fait" ou "non-fait". Un utilisateur doit pouvoir rentrer son pseudo sur l'activité principale de l'application, et se verra alors transporté sur une nouvelle activité qui contient l'ensemble des TodoLists qu'il s'est créées. Lors d'un clic sur l'une des TodoLists, une nouvelle activité est générée et affiche l'ensemble des tâches associées à la TodoList sélectionnée, ainsi que leur état. Ces deux affichages se basent sur une implémentation de type RecyclerView, avec Adapters.

L'utilisateur doit en plus avoir la possibilité d'ajouter une nouvelle TodoList, mais aussi de rajouter des tâches à n'importe laquelle de ses TodoLists. De plus, la persistance des données est primordiale; il nous faut stocker dans les préférences de l'application le profil associé à l'utilisateur, c'est-à-dire un fichier contenant les données de l'utilisateur (pseudo, ensemble des TodoList, de leurs items et de leurs états). On utilise pour cela la technologie JSON et la sérialisation.

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

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 permet à l'utilisateur de saisir son pseudo, tout en proposant par défaut le dernier pseudo saisi. Une toolbar permet d'accéder à une seconde activité, SettingsActivity. On sauvegarde le pseudo saisi dans les préférences de l'activité, et ce même paramètre est transmis à l'activité ChoixListActivity, ouverte lors du clic sur le bouton OK.
- SettingsActivity : cette activité affiche le pseudo affiché par défaut (dernier pseudo saisi par l'utilisateur) en utilisant des fragments (classe SettingsFragment).

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. Elle permet de plus de générer une nouvelle activité, ShowListActivity, lors d'un clic sur le titre d'une des TodoLists.
- 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". Lors d'un clic sur une des tâche, inverse l'état de la checkbox et de la tâche correspondantes.

Ces deux activités travaillent en étroite collaboration : en effet, lors de l'ajout d'une tâche à une TodoList, il faut sauvegarder le nouveau profil associé au pseudo dans les préférences de l'application pour tenir compte des mises à jour, et recréer le profil associé dans l'activité précédente afin de satisfaire à la persistance des données. De même lors de l'ajout d'une nouvelle TodoList. Nos deux activités font donc appel aux mêmes fonctions de sauvegarde d'un profil au format JSON, et d'instanciation de profil à partir de données JSON : nous avons donc choisi de placer ces fonctions dans une classe Library, dont hérite nos deux activités.

Les fonctions de cette classe sont appelées à chaque mise en pause et à chaque reprise d'une activité (cf. cycle de vie des activités).

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".

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 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…);
    - Gérer l'UTF-8 en JSON (problème d'écriture des caractères spéciaux).

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'`

- Code correspondant à l'export d'un profil sous le format JSON, à la création d'un profil à partir d'un fichier JSON, et à la persistance des données dans les préférences de l'application : Isabelle Le Glaz (Moodle)