Programmation Mobile
et Réalité Augmentée

SEQUENCE 1 – TP

Application To-do List

Hadrien SALEM

INTRODUCTION

Pour ce TP, mon rendu contient :

- Le présent compte-rendu PMR_TP1_HSALEM_CR.pdf
- Le projet Android Studio de mon application PMR_TP1_HSALEM
- Le fichier .apk de mon application pmr_tp1_hsalem.apk

Pour faire fonctionner l'application, il faudra soit passer par Android Studio (en ouvrant le dossier PMR_TP1_HSALEM) , soit installer l'APK via un explorateur de fichiers.

Il semble qu'essayer de faire les deux successivement soit parfois problématique (conflit dans les fichiers internes), je préconiserais donc plutôt de passer uniquement par Android Studio.

Remarques :

Le code est commenté à 95% en anglais pour éviter les problèmes dûs à l'utilisation de caractères spéciaux. S'il reste des commentaires en français, il s'agit d'un oubli de ma part. Il est également possible que j'ai oublié de supprimer certains commentaires « TODO ».

ANALYSE

Dans ce TP, il s'agissait de créer une application permettant de gérer des To-do lists. Il fallait pouvoir créer et afficher des listes et leurs items, ainsi que pouvoir cocher ces items. Chaque utilisateur, distingué par un login, peut posséder plusieurs listes. Il fallait également gérer la persistence des données.

I. Architecture

Pour ce qui est de la **gestion des listes**, j'ai mis en œuvre l'architecture exigée par le sujet. Aux objets ItemToDo, ListeToDo et ProfilToDo, je n'ai rajouté que des méthodes permettant la conversion au format JSON (xxxToJSONObject() et xxxToJSONString()).
Ces objets sont regroupés dans le **package « lists »**. Dans ce package, j'ai également ajouté des fichiers .json qui m'ont servi au débuggage.

Les activités se trouvent toutes dans le **package « activities »**.

Le **package « adapter »** contient des adapteurs pour les RecyclerView.

Le **package « utils »** contient la classe UserManager, qui permet de manipuler les données utilisateur. Son fonctionnement sera détaillé dans la suite (II.2).

Pour la gestion de fichiers .json, j'ai utilisé la librairie org.json qui existe déjà dans Android Studio, et non pas la libraire GSON préconisée par l'énoncé, par familiarité avec la première. Ce choix ne change rien au fonctionnement de l'application.

II. Mise en œuvre et différences par rapport au sujet

1. Utilisation d'un fichier pour la gestion des utilisateurs

En commençant à développer l'application, j'avais l'intention de mettre en place une persistance des données utilisateur. Ainsi, plutôt que de passer par les préférences pour sauvegarder les pseudos/listes, j'ai décidé de passer par les fichiers de l'application pour qu'ils soient immédiatement sauvegardés à la création. Tous les logins et les listes correspondantes sont donc stockées dans un unique fichier list_data.json. La structure de ces objets JSON figure en annexe.

2. Utilisation de la classe « UserManager ».

Ce paragraphe est en lien direct avec le précédent. La classe « UserManager » que j'ai créée permet de faire le lien entre les objets java de gestion des listes, et les objets JSON correspondants, qui figurent dans le fichier list_data.json.

Un objet « UserManager » est créé pour gérer un unique utilisateur (son constructeur requiert un login). À sa création, l'UserManager :

- **Crée le fichier list_data.json s'il n'existe pas déjà**. Pour donner sa structure au fichier, un profil par défaut est créé (défini dans un companion object DataInitialise). Ce profil par défaut n'apparaît pas dans le choix du login (voir le paragraphe sur les préférences) et restera donc inconnu de l'utilisateur (à moins qu'il ne décide d'utiliser le login DEBUG_123456789, possibilité que j'ai décidé de ne pas gérer dans le cadre de ce TP).
- **Vérifie si l'utilisateur qu'il gère existe via son login**. Il lit pour cela le fichier list_data.json.
- **Si l'utilisateur n'existe pas, il est créé et ajouté au fichier.**
- **Le profil correspondant est construit**. On utilise pour cela la méthode buildProfile(), qui retourne un objet de la classe ProfilListeToDo.

Une fois que l'utilisateur existe dans les fichiers, son **UserManager peut être utilisé pour ajouter des listes et des items à ces listes, ainsi que pour sauvegarder l'état fait/non fait des items**. Ainsi, la classe UserManager est appelée dans ChoixListeActivity et ShowListActivity, lors de l'utilisation des boutons d'ajout.

Enfin, tout objet UserManager permet d'accéder à la liste des clés de list_data.json, autrement dit à la liste des logins.

En un sens, l'application est construite comme si list_data.json était une base de données, et UserManager fait le lien entre cette base de données et l'interface utilisateur.

3. Gestion des préférences

L'écran de préférences (correspondant à SettingsActivity) ne sert qu'à choisir un pseudo parmi ceux qui existent déjà pour qu'il s'affiche dans l'EditText de MainActivity. La liste de pseudos affichés ignore le pseudo stocké dans la variable DEBUG_LOGIN de UserManager.DataInitialise.

4. Autres remarques

Les listes et les items sont affichés grâce à des RecyclerView. Pour leur implémentation, j'ai repris la structure présentée par M.Boukadir en cours (séparer l'Adapter, y inclure le ViewHolder comme classe interne, définir l'interface Listener dans le même fichier).

III. Perspectives d'amélioration

En dehors des améliorations facultatives déjà données par le sujet, voici quelques améliorations que je pourrais apporter à l'application avec plus de temps :

- **Permettre à l'utilisateur de supprimer et renommer des listes et des items**. Cette amélioration rendrait l'application utilisable au quotidien. Ces fonctionnalités n'étant pas demandées par le sujet, je ne les ai pas implémentées, mais c'est facilement faisable avec l'architecture actuelle : il suffit d'ajouter les fonctions correspondantes dans UserManager et de les faire appeler par des boutons qu'il faudrait ajouter aux items des RecyclerView.
- **Faire en sorte que l'application s'adapte à différents écrans et tailles d'écriture.** En essayant l'application sur le téléphone de ma mère, je me suis rendu compte que si la police d'écriture définie par défaut dans le téléphone est agrandie, les textes sortent de leur cadre. Il faudrait étudier ce problème plus en détail et voir comment mettre en place un design « responsive » sur ce type d'application. Peut-être est-il possible de contraindre une taille de police, par exemple.
- **Éviter certaines redondances de code**. Notamment en ce qui concerne l'écriture dans les fichiers. La documentation spécifie qu'il peut être lourd d'ouvrir et de fermer des fichiers de façon répétée. Pour cette application de To-do List simple, j'ai décidé de ne pas trop m'en soucier, sachant en particulier que les fichiers JSON sont extrêmement légers.

CONCLUSION

Ce TP m'a permis de mettre en pratique l'architecture vue en cours pour l'utilisation de RecyclerView, d'apprendre à utiliser les préférences et les fichiers internes de l'application (via le format JSON). J'ai également acquis de l'expérience en programmation orientée objet via ce projet complexe.

BIBLIOGRAPHIE

- Le cours de M.Boukadir
- Cette page openclassrooms pour des rappels sur les RecyclerView : https://openclassrooms.com/fr/courses/4568576-recuperez-et-affichez-des-donnees-distantes/4893781-implementez-votre-premiere-recyclerview
- Des posts sur le forum https://stackoverflow.com/ pour débugger des erreurs mineures
- La documentation trouvée https://developer.android.com/, en particulier pour l'utilisation des préférences et la gestion des fichiers internes
- Pour utiliser des fichiers JSON sous Android : https://medium.com/@nayantala259/android-how-to-read-and-write-parse-data-from-json-file-226f821e957a

ANNEXE

Structure de list_data.json :

```
{
  "user1":[
    {"titreListe": "Liste1",
      "items": [
        {"description": "item11","fait": false},
        {"description": "item12","fait": false}
      ]
    },
    {"titreListe": "Liste2",
      "items": [
        {"description": "item21","fait": false},
        {"description": "item22","fait": false}
      ]
    }
  ],

  "user2":[
    {"titreListe": "2Liste1",
      "items": [
        {"description": "2item11","fait": false},
        {"description": "2item12","fait": false}
      ]
    },
    {"titreListe": "2Liste2",
      "items": [
        {"description": "2item21","fait": false},
        {"description": "2item22","fait": false}
      ]
    }
  ]
}
```

Les noms d'utilisateur sont des clés, et les valeurs associées sont des tableaux (correspondant à mesListesToDo). Dans l'exemple ci-dessus, il n'y a que deux logins enregistrés : user1 et user2, qui ont chacun deux to-do lists contenant 2 items.