Application TodoList

Electif PMR 2019

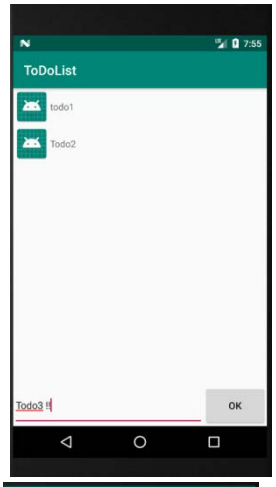
Au cours de ce TP, le but était de coder une application TodoList, avec trois pages principales. La première (MainActivity) permet à l'utilisateur d'entrer son pseudo, la seconde (ChoixListActivity) affiche les TodoList associées au pseudo (aucune si le pseudo n'est pas connu), et la troisième (ShowListActivity) affiche la liste des taches à faire propre à la TodoList sélectionnée. Au cours de ce TP, la difficulté principale qui s'est montrée à moi aura été de correctement sauvegarder les données dans la mémoire interne du téléphone.

Présentation générale

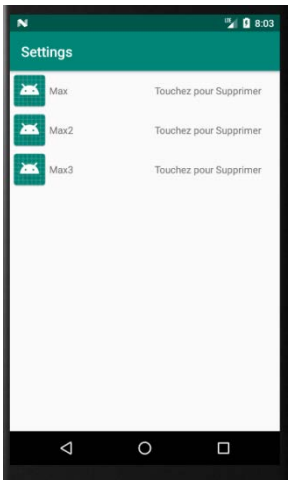
Structure de base :

La première activité avec un EditText pour le pseudo et un Button équipé d'un OnClickListener pour récupérer le pseudo, et mise en place d'un bouton Menu. Dans le layout associé à cette activité, il est écrit pour l'EditText le pseudo précédemment choisi par l'utilisateur.
Le Menu affiche la liste des pseudos enregistrés et permet de supprimer ceux souhaité.
Lors du clic sur le Button OK, on enregistre le pseudo s'il n'existe pas, sinon on charge sur ce pseudo avec les données sauvegardées dans la mémoire interne du téléphone.

La deuxième activité est la page où l'on affiche toutes les todolist associées au pseudo que l'on a rentré précédemment.
Il y a également un EditText ainsi qu'un bouton pour ajouter une todolist à la liste des todolist associées au pseudo entré au début.



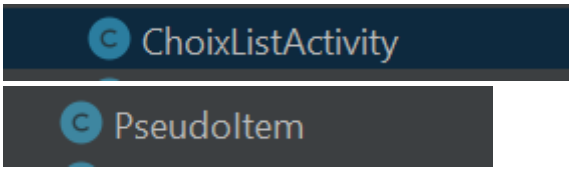
La troisième activité est la page où l'on affiche toutes les taches associées à la liste sélectionnée, associée au pseudo que l'on a rentré précédemment.



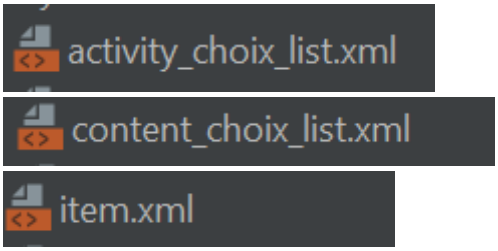
Il y a également un EditText ainsi qu'un bouton pour ajouter une tache à la todolist sélectionnée.
Il y a enfin une activité Settings, qui permet à l'utilisateur de gérer la liste des profils qui ont été entrés. Simplement en cliquant sur le profil que l'on souhaite supprimer, on le supprime et il n'existe alors plus et aucune trace n'est gardée.

Mise en place des RecyclerView :

Nous allons voir la mise en place d'un RecyclerView pour l'activité ChoixListActivity. Le système est le même pour chacune des activités nécessitant un RecyclerView.
Il a été mis en place ces deux classes :



Et ces 3 fichiers xml :



item.xml met en place un « item » qui sera dans le Recyclerview.
content_choix_list.xml est un simple layout contenant un objet Recyclerview, on insérera ce layout au sein de activity_choix_list. Ce dernier est composé également d'un EditText et du Button.

En s'inspirant du cours de l'intervenant Monsieur Boukadir [1], ainsi que de la documentation sur Internet [2], j'ai mis en place la classe PseudoItem qui hérite de « RecyclerView.Adapter<PseudoItem.ItemViewHolder> »
Au sein de cette classe, on se concentre à créer les ViewHolder (qui sont donc les objets qui porteront les item de item.xml). On implémente également des fonctions de listener, afin de pouvoir correctement agir lorsque l'utilisateur appuie sur les items en question.

La gestion de ces fonctions de listener se fait au sein de ChoixListActivity.
Comme nous le voyons sur le prochain document, j'ai ici mis en place une fonction sur onItemClick, donc lorsque l'on clique sur un item affiché (donc ici on choisi une des todolist associée au pseudo) la fonction est exécutée. Par exemple ici, nous mettons dans le Bundle

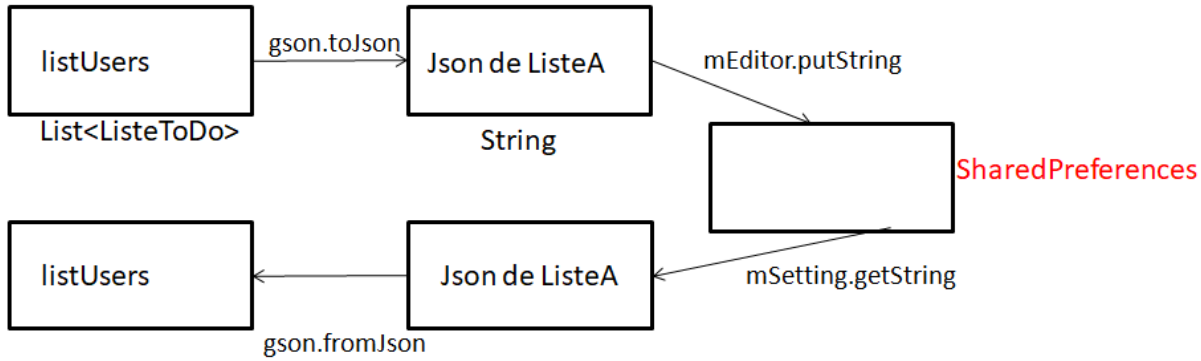
data la position de la todolist que l'on vient de sélectionné et nous lançons l'activité suivante. La position de la todolist correspond à l'indice de la todolist dans la liste des todolist associé au pseudo que l'on a selectionné.

```
mAdapter.setOnClickListener(new PseudoItem.OnItemClickListener() {
    @Override
    public void onItemClick(int position, List<ListeToDo> mList) {
        Intent toShowListActivity = new Intent( packageContext: ChoixListActivity.this,ShowListActivity.class);
        //on transmet la position de la TodoList sélectionnée
        data.putInt("positionSelectionnee",position);
        toShowListActivity.putExtras(data);
        startActivity(toShowListActivity);
    }
});
```

Ce système de groupe de class et de documents xml pour un recyclerview se répète pour chacun des recyclerviews. Cela alourdi un peu l'architecture du projet.

Sauvegarde des données :

Le document qui suit résume la méthode de sauvegarde des données qui a été suivi pour ce projet [3] :



La totalité des données est sauvegardé dans la liste de profil listUsers. Au sein de cette liste, il y a la totalité des utilisateurs qui ont été enregistré. Pour chaque utilisateur, il est associé leur liste de todolist, et pour chaque todolist il y a la liste de tache.

Ainsi, pour chacune des activités qui ont besoin d'information, il faut aller cherche la liste des utilisateurs, trouver le pseudo associé, et aller chercher les infos que l'on souhaite.

Il est donc nécessaire de passer quelques informations supplémentaires entre chaque activité, afin de pouvoir correctement utiliser listUsers. L'utilité du Bundle data que nous avons vu tout à l'heure est donc ici avéré.

Prenons l'exemple où l'utilisateur est connu. On obtient listUsers depuis les SharedPreferences ainsi que grâce à la fonction de Gson fromJson. Il faut cependant pouvoir « repérer » la où se trouve l'utilisateur dans listUsers. On crée donc une variable int positionUser, qui lorsque l'on trouve la position de l'utilisateur connu dans listUser la sauvegarde. On passe cette variable à travers un Bundle et ainsi lors de l'activité suivante, on peut travailler sur le profil en question grâce à listUsers.get(positionUser).

Conclusion :

Pour conclure, le projet est presque entièrement rempli. Cependant, la classe SettingActivity n'hérite pas de PreferenceActivity. En effet j'ai préféré mettre en place un RecyclerView ainsi qu'une fonction de onClickListener afin de supprimer le profil sélectionné, directement sur le document listUser qui sert de liste de stockage de toutes les informations. Il faut également noté que (malgré un code commenté dans ShowListActivity dans la fonction onCheckBoxClick) le code testé pour cocher la checkBox pour attribuer la valeur « faite » à la tache en question n'est pas fonctionnel et fait crasher l'application. L'erreur vient certainement de la sélection faite à partir de variables de position passées dans le bundle. Malheureusement faut de temps et de prise de recul, je ne trouve pas l'origine de ce problème, qui est il me semble, assez mineur. Enfin, il apparait que lorsqu'une nouvelle todolist est ajouté, la mise à jour de l'affichage de l'activité ChoixListActivity ne se fait pas correctement. Afin de contrer cela, j'ai choisi de relancer l'activité (cf. document ci-après).

```
Intent intent = getIntent();
finish();
startActivity(intent);
```

On peut donc dire que cette méthode n'est pas optimale.

Au débogage, il semble également que lorsque listUser est vide, le menu fait planter l'application.

Perspective :

Les perspectives de perfectionnement de l'application sont tout d'abord celles proposées qui n'auront pas été implémentée faute de temps. Nous pouvons penser à ne pas tenir compte des majuscules dans l'écriture d'un nouveau pseudo.

Nous pouvons également imaginer une certaine forme de sécurisation de l'application en demandant non seulement un pseudo, mais également un mot de passe. L'application propose seulement une todolist, donc il ne semble pas vraiment nécessaire de sécuriser à ce point l'application. Enfin, afin d'optimiser le fonctionnement de l'application, il faudrait trouver une solution pour mettre à jour « proprement » l'activité ChoixListActivity. Finalement, il faudrait trouver l'origine du bug dans le menu faisant planter l'application lorsque listUser est vide.

[1] Intervenant Monsieur Boukadir, Ecole Centrale de Lille

[2] https://www.youtube.com/playlist?list=PLrnPJCHvNZuBtTYUuc5Pyo4V7xZ2HNtf4 Liste de tutoriels apprenant à mettre en place des RecyclerView sur lesquelles on peut placer des OnClickListener
[3] Cours Monsieur Bourdeaud'huy, Ecole Centrale de Lille.