Compte rendu application de todo list

Luca Jakomulski et Tristan Buscemi

Introduction

Commençons par une description de l'application que nous avons codée. La première fenêtre qui s'affiche permet à l'utilisateur d'entrer un pseudo et d'appuyer sur un bouton afin d'accéder à ses todo lists. La fenêtre qui affiche ses todo list lui permet également d'ajouter des todo list grâce à un champs texte et un bouton en bas de la fenêtre, de supprimer une todo list en cliquant sur l'icone en forme de poubelle à gauche de la todo list et bien sûr d'accéder aux items d'une todo list en cliquant dessus. Quand l'utilisateur clique sur une liste, il a accès aux items composant sa todo list. La fenêtre qui affiche les items d'une todo liste permet également d'ajouter un item grâce à un champs texte et un bouton en bas de la fenêtre, de supprimer un item en cliquant sur l'icone en forme de poubelle à gauche de l'item et de cocher ou décocher un item en cliquant sur la cache à cocher à gauche de l'item. Enfin, les paramètres de l'application sont accessibles depuis n'importe quelle fenêtre de l'activité en cliquant sur l'écrous en haut à droite. Dans les paramètres on retrouve le nom de l'utilisateur qui utilise actuellement l'application, ce dernier peut être modifier.

Analyse

Pour réaliser cette application, nous avons réalisé différentes activités :

- MainActivity qui correspond à la fenêtre dans laquelle l'utilisateur entre son pseudo ;
- ListsActivity qui correspond à la fenêtre qui affiche les todo lists de l'utilisateur ;
- ItemActivity qui correspond à la fenêtre qui affiche les items d'une liste ;
- SettingsActivity qui correspond à la fenêtre qui affiche les paramètres de l'application.

Enfin on remarquera la présence de GenericActivity qui est la classe mère dont hérite les activités MainActivity, ListsActivity et ItemActivity.

Afin de mieux comprendre les problèmes auxquels répondent ces différentes classes, nous allons les parcourir et expliquer nos choix techniques.

La classe GenericActivity a 3 attributs protégés qui sont SharedPreferences (l'espace de stockage des préférences), UserName (le pseudo en cours d'utilisation) et UserProfile (le profil utilisateur associé au pseudo en cours d'utilisation). L'application étant construite autour d'un utilisateur et de son profil, il est assez naturel de placer ses attributs dans cette classe mère. La méthode onCreate va permettre d'affecter aux attributs UserName et UserProfile des valeurs que l'on va récupérer depuis le fichier des préférences. Dans cette classe on va construire le menu qui se compose uniquement de l'élément Paramètres. On retrouve deux méthodes writeJSON et readJSON qui vont nous permettre d'écrire sous forme JSON le profil utilisateur dans les paramètres et de récupérer un profil enregistré sous forme JSON et d'instancier la classe UserProfile à partir des données récupérées. On retrouve également des méthodes qui vont nous permettre d'éviter de répéter du code comme toMain, toLists et toItems qui vont permettre de passer d'une activité à l'autre. Plus anecdotiquement on trouve dans cette classe des méthodes qui vont directement modifier l'attribut SharedPreferences afin de mettre à jour le profil utilisateur (updateProfile) ou de créer un profil utilisateur (createProfileIfNeeded).

Passons à la classe MainActivity. Cette classe dispose d'une méthode toUserLists qui sera accédée lorsque l'utilisateur clique sur le bouton présent dans l'activité. Dans cette méthode on récupère le pseudo de l'utilisateur et on l'enregistre dans les préférences si celui-ci est valide avant de passer à l'activité suivante (ListsActivity) qui va afficher ses listes.

Note : les classes ListsActivity et ItemsActivity étant structurées de la même manière nous nous contenterons d'expliquer simplement la première des deux classe, le lecteur pourra par lui-même expliquer les différences assez simplement en regardant le code source.

La classe ListActivity permet d'afficher les listes de l'utilisateur dont le pseudo est enregistré dans les paramètres. Les élements présents dans l'activité sont un champs texte et un bouton ainsi qu'un RecyclerView. La méthode onCreate va permettre d'initialiser le RecyclerView. Les todo lists à afficher dans ce RecyclerView sont récupérées depuis le UserProfile. C'est également dans cette méthode que l'on instancie notre Adapter que l'on a nommé ListAdapter. Faisons un détour par cette classe contenue dans le package adapter. On y retrouve les méthodes communes aux adapter : le constructeur, onCreateViewHolder, onBindViewHolder et getItemCount d'où le peu de commentaire dans cette classe. Dans la classe ItemViewHolder, qui correspond à un élément particulier du RecyclerView, on va initialise un TextView que l'on va remplir avec le nom d'une todo list et un ImageView qui va afficher l'icône d'une poubelle. Dans le constructeur d'ItemViewHolder, on va mettre un listener sur le click d'une liste qui va faire appel à la méthode onItemClicked et un listener sur le click de l'icône poubelle qui va faire appel à la méthode onDeleteClicked. Si on revient à la classe ListActivity, on retrouve nos méthodes onItemClicked et onDeleteClicked. OnItemClicked va permettre de passer à l'activité ItemsActivity en affichant les items correspondant à la liste cliquée. OnDeleteClicked va ouvrir une fenêtre de dialogue pour demander à l'utilisateur de confirmer la suppression de la liste, s'il accepte la liste est supprimée du profil et retirée du recyclerView. Passons maintenant à la méthode addList qui va récupérer le nom de la liste que l'utilisateur veut créer, vérifier si une liste ayant le même nom n'existe pas déjà et l'ajouter au profil, puis l'utilisateur est directement redirigé vers l'activité ItemsActivity pour créer des items dans sa nouvelle liste. On remarquera aussi la présence de la méthode onStart, dans celle-ci on récupère le pseudo dans les préférences. Cela peut paraître redondant avec le onCreate de la genericActivity mais pas du tout. Si on a la présence de ce onCreate c'est pour vérifier si le pseudo a changé après la création de l'activité. Ce cas survient quand on modifie le pseudo dans les paramètres et que l'on retourne à l'activité précédente nécessitant donc de mettre à jour les todo lists à afficher si nécessaire.

Enfin l'activité SettingsActivity qui fait intervenir un fragment permet l'affichage et la modification du pseudo de l'utilisateur.

Les classes Item, TodoList et UserProfile qui sont dans le package data sont celles représentées dans le diagramme UML de l'énoncé. On notera simplement l'ajout d'une méthode removeList dans la classe UserProfile et removeItem dans TodoList permettant la suppression d'un élément entré en paramètre.

Conclusion

Nous avons essayé de faire une application donc le code est structuré, le plus lisible possible, avec des variables compréhensible (on remarque la convention qui fait que les variables protégées et privées sont préfixées d'un m). Nous avons essayé de factoriser un maximum de code dans la classe genericActivity sans perdre de lisibilité. On a pu mettre en pratique tous les éléments vus dans le cours,

approfondir certains points comme l'utilisation des ConstraintLayout dans l'ensemble des activités ou la notion de Fragment. On a fait de notre mieux pour rendre l'expérience utilisateur le plus simple possible avec l'ajout de la flèche retour en haut à gauche, l'utilisation d'icone pour avoir une application la plus visuelle possible. De même si l'utilisateur fait une action non autorisée, un message lui est forcément affiché.

Perspectives

L'élément le plus contraignant de l'énoncé est le fait de devoir enregistrer les données dans les préférences dans un fichier texte. Etant donnée la structure des données dans l'application, il serait préférable d'utiliser une base de données, mais cela engendrerait forcément de coder beaucoup plus pour un résultat qui ne sera pas visible par l'utilisateur. Cependant si l'application est amenée à évoluer on aurait plus de facilité à coder rapidement les nouveaux changements. De plus on peut toujours imaginer de nouvelles fonctionnalité telle que la modification du titre des todo list, l'ajout de plusieurs langues dans les paramètres… Ou même penser à améliorer le style visuel de l'application afin d'avoir une application que ne ressemble à aucune autre mais ces changements nécessitent beaucoup de temps.

Sitographie

L'ensemble des connaissances nécessaire à la réalisation de cette application sont issus du site pour les développeurs Android ou de StackOverflow. On notera tout de même ces quelques liens pertinents car ils nous ont le plus aidé :

Settings

https://medium.com/@JakobUlbrich/building-a-settings-screen-for-android-part-1-5959aa49337c

https://developer.android.com/guide/topics/ui/settings

Shared preferences

https://developer.android.com/training/data-storage/shared-preferences.html

Gson

https://howtodoinjava.com/library/google-gson-tutorial-convert-java-object-to-from-json/