PMR - Compte rendu Séquence 1

Johan Manuel

Objectifs

L'objectif de ce projet était la réalisation d'une application de *to-do list* multi-utilisateurs, permettant de cloisonner les listes de chacun. Les objectifs en terme d'UX sont définis par un storyboard, et les objectifs pédagogiques sont l'utilisation de plusieurs outils comme les *RecyclerViews*, la bibliothèque *GSON* et la persistence de données.

Développement

J'ai commencé par la mise en place des différentes *Activities* de l'application et de leur layout, notant les *ids* des éléments utilisés par la suite dans le code. J'ai choisi d'utiliser deux *RecyclerViews* pour la liste des listes d'un utilisateur et la liste des items d'une liste. Un *ListView* classique aurait peut-être suffit pour afficher la liste des listes, mais autant faire les choses proprement, puis il n'est pas impossible d'accumuler des centaines de listes au fil des années, j'espère longues, d'usage de cette application.

Cela fait, j'ai implémenté les classes demandées dans le diagramme UML du sujet, à savoir *ItemToDo*, *ListeToDo* et *ProfilListeToDo*. Je me suis ensuite mis à faire fonctionner les *RecyclerViews*. Dans un premier temps, j'ai testé leur fonctionnement avec des éléments factices, générés par le code. Je n'ai rencontré qu'un bug notable, mes *RecyclerViews* n'affichaient que leur premier élément. Cela était dû au fait que mes *ViewHolders* avaient leur propriété *layout_height* mise à *match_parent* au lieu de *wrap_content*.

L'étape suivante a été de faire fonctionner les préférences, et comprendre comment les utiliser pour stocker les profils utilisateurs. Après avoir compris qu'appeler *editor.clear()* avant chaque modification de préférence était contre-productif, j'ai décidé de stocker le pseudo de l'utilisateur en cours dans une préférence fixe, et chaque profil utilisateur dans une préférence identifiée par son pseudo. Il m'a fallu expérimenter un peu avec *GSON*, par exemple je n'étais pas sûr par les docs et les screencasts que les *MutableLists* se (dé-)sérializeraient sans plus de travail de ma part, mais il s'est trouvé que tout marchait tout seul de ce point de vue là. Pour afficher une liste ou les listes d'un utilisateur, le profil complet de celui-ci est dé-sérializé, puis la bonne liste est identifiée et passée à l'*Adapter* correspondant, et quand la liste est modifiée, elle est re-sérializée et le profil est ré-écrit dans les préférences.

J'ai finalement passé un moment à penser aux *corner cases*, comme par exemple gérer le fait d'avoir deux listes/éléments ayant un même nom (j'ai décidé de ne pas l'autoriser), gérer le changement d'utilisateur par les préférences à partir de la vue d'une liste (l'activité "vue de liste" se termine et l'on revient à la vue des listes du nouvel utilisateur). Il y a un cas particulier que je n'ai pas réglé, celui d'un utilisateur s'appelant "pseudo", qui est le nom de la préférence contenant l'utilisateur courant. J'aurais pu prohiber ce pseudo, mais il m'a semblé qu'exposer ce comportement correspondait plus à l'esprit de la gestion des données dans cette application.

Conclusion

Ce sujet était assez plaisant et suffisamment court, je pense qu'il m'a été très utile pour commencer à comprendre le développement d'une application Android, bien que l'utilité d'avoir plusieurs utilisateurs dans ce genre d'application m'échappe encore.

Perspectives d'amélioration

Certains aspects importants de l'application utilisent des façons de faire qui sont loin de l'état de l'art en développement Android, par exemple l'affichage des préférences et la persistence de données. De ce que je comprends, l'affichage des préférences se fait de nos jours par l'utilisation de *Fragments*, qui, il est vrai, ont l'air relativement compliqués à mettre en place, peut-être trop pour une simple application de todo list. Quant à la persistence de données, le stockage de *JSON* dans les préférences me semble être un détournement de l'usage voulu de celles-ci ; la mise en place d'une base de donnée légère type *SQLite* serait sûrement plus appropriée. Mais là encore, plus coûteuse en temps.

Évidemment, l'ergonomie de cette application pourrait être améliorée, par exemple en affichant une liste des utilisateurs existant, ou en permettant la suppression de listes et d'éléments de liste, la modification de leurs noms, etc...

Annexe

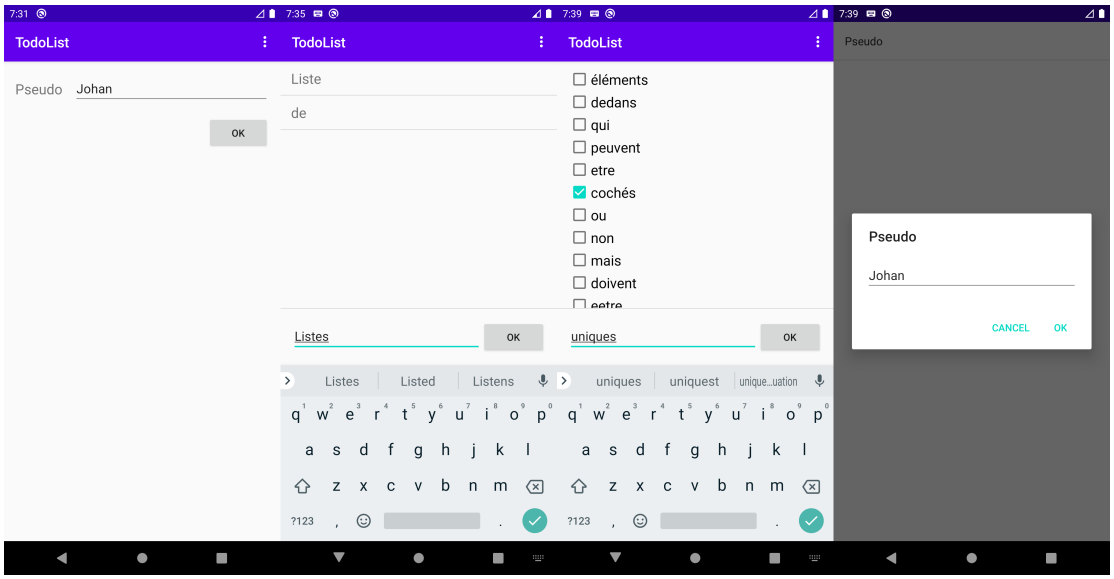

FIGURE 1 – Captures d'écran des activités de l'application réalisée