Programmation Mobile

Et Réalité Augmentée

SEQUENCE 2 – TP

Amélioration de l'application To-do List grâce à une API REST

Hadrien SALEM

INTRODUCTION

Pour ce TP, mon rendu contient :

- Le présent compte-rendu « PMR_TP2_SALEM_CR ».
- Le projet Android studio de mon application PMR_TP2_HSALEM.

Pour faire fonctionner l'application, il suffit d'ouvrir le projet (dossier) PMR_TP2_HSALEM dans Android Studio. L'adresse de l'API peut *a priori* être entrée directement via les préférences de l'application. L'adresse que j'ai entrée par défaut correspond à une IP permettant d'accéder au localhost via un émulateur Android.

ANALYSE

Il s'agissait d'améliorer l'application du TP1 en utilisant non plus des fichiers JSON, mais une API permettant la gestion de To-do Lists.

L'enjeu majeur de ce TP est donc la communication avec l'API todo. Dans les versions actuelles d'Android, la communication par internet ne peut pas avoir lieu dans le main thread. L'autre difficulté de ce TP est donc la gestion de plusieurs threads.

Les solutions que j'ai choisies sont :

- La librairie retrofit pour la communication avec l'API.
- Les coroutines pour les appels asynchrones et le multi-threading.

I. Architecture

Pour l'architecture de l'application, j'ai réutilisé ce que nous avions vu à la séquence 2 avec M.Boukadir : j'ai employé retrofit via un DataProvider, qui lui-même fait appel à des fonctions d'une interface ToDoAPIService (annotées pour être intelligibles par retrofit).

Afin de gérer les réponses asynchrones, ces méthodes sont des méthodes « suspend », appelées depuis une coroutine « activityScope » définie comme en cours. Elles sont par ailleurs exécutées dans un autre thread (grâce à withContext(Dispatchers.Default)).

J'ai ajouté à cela une classe UserManagerAPI, qui fait le lien entre les data renvoyées par l'API via le DataProvider, et les objets créés en java (ItemToDo, ListeToDo). L'intérêt est de séparer tout ce qui concerne la connexion à l'API, de tout ce qui concerne les « XxxActivity », ainsi que de simplifier la mise à jour de ce que j'avais fait pour le premier TP.

II. Implémentation

J'ai implémenté toutes les fonctions demandées par le sujet de la façon préconisée. J'ai également mis en œuvre la première amélioration facultative (ajout d'une nouvelle liste pour l'utilisateur connecté).

Lorsque l'utilisateur n'est pas reconnu par l'API, un Toast s'affiche : « Identifiants incorrects ».

Pour vérifier si la connexion internet est établie j'ai utilisé le post suivant :

https://stackoverflow.com/questions/5474089/how-to-check-currently-internet-connection-is-available-or-not-in-android

Dans l'état actuel de l'application, il faut recharger l'activité pour que le bouton OK s'active si on n'était pas connecté. Il existe des moyens un peu plus complexes pour vérifier si l'état de la connexion change, que je n'ai pas mis en place :

https://stackoverflow.com/questions/25678216/android-internet-connectivity-change-listener

III. Problèmes rencontrés et solutions apportées

1) Connexion à l'API

J'ai commencé par tester l'API sur Postman. Je n'ai pas réussi à faire fonctionner de requêtes sur le serveur de Centrale (qui rencontrait apparemment des problèmes au moment où j'ai travaillé), et j'ai donc réalisé tous mes tests en localhost.

Pour que les requêtes fonctionnent en localhost, j'ai dû faire appel directement à la page index.php, et donc les URLs que j'utilise sont différentes de celles données dans la documentation.

En clair, j'ai par exemple remplacé :

http://tomnab.fr/todo-api/authenticate?user=tom&password=web

par

localhost/todo-api/index.php?request=authenticate&user=tom&password=web

La seule différence étant l'utilisation d'un paramètre request.

Cela ne change pas le fonctionnement de l'application, mais je préfère le préciser pour expliquer la différence d'URL.

Autre problème mineur, lors de la connexion à l'API, j'ai rencontré une erreur « Exception from call site #4 bootstrap method », que j'ai résolue grâce à la solution proposée sur ce post (ajout de compileOptions dans le build Gradle):

https://stackoverflow.com/questions/55034848/exception-from-call-site-4-bootstrap-method-code-doesnt-work-in-android-studi

2) Affichage des items

Lors de l'affichage de ShowListActivity, il arrive que la fonction onItemClicked (qui doit cocher les items) soit appelée. Une erreur était engendrée si on essayait de cocher un item déjà coché. Pour contourner ce problème, j'ai créé un deuxième bloc de vérification dans le UserManager, qui

détermine si oui ou non il faut réellement cocher l'item. Cette solution semble avoir complètement réglé le problème.

3) Autre remarque

En lançant l'application, il y a parfois (très rarement) une exception « unexpected end of stream » que je ne m'explique pas et qui disparaît en relançant l'application. D'après mes recherches, il pourrait s'agir d'un problème backend venant de mon PC et de la façon dont fonctionne mon serveur local. Je n'ai pas réussi à reproduire le problème suffisamment pour le résoudre.

IV. Perspectives d'amélioration

Tout comme pour le premier TP, l'implémentation de fonctions supplémentaires rendrait l'application utilisable au quotidien : renommer les items et les listes, les supprimer, les réorganiser. Des améliorations visuelles rendraient également plus agréable l'expérience utilisateur.

CONCLUSION

Ce TP m'a permis de comprendre l'utilisation des coroutines et des API REST, avec notamment des bonnes pratiques de debuggage (utilisation de Postman). J'ai également fait face à différents codes d'erreur en faisant des requêtes (400, 401 par exemple) et ai appris à trouver leur origine pour éliminer ces problèmes.

BIBLIOGRAPHIE

https://stackoverflow.com/questions/5474089/how-to-check-currently-internet-connection-is-available-or-not-in-android

https://stackoverflow.com/questions/25678216/android-internet-connectivity-change-listener

https://stackoverflow.com/questions/55034848/exception-from-call-site-4-bootstrap-method-code-doesnt-work-in-android-studi