Compte rendu Application TODO séquence 2

Introduction

J'ai fait le choix d'utiliser Retrofit, à la suite du cours avec Monsieur Boukadir, on avait des trames pour utiliser l'API et une certaine confiance en la solution.
Vous trouverez donc sur le dépôt git associé le code de mon application de TODOList en lie avec votre API, utilisée ici en local.

Notes pour les tests :
Pour les tests de la solution : J'ai utilisé l'API en local, directement à l'adresse sur mon serveur, les adresses sont donc du type **« /todo-api/index.php ?request=authenticate&… »**
De plus, le test sur un émulateur nécessite de faire apparaître l'adresse IP de l'émulateur qui est par défaut « 10.0.2.2 », le port utilisé avec MAMP est le 8888 et l'url de base utilisée pour le service retrofit doit commencer par « http:// ». On aura donc une url complète qui ressemble à cela http://10.0.2.2:8888/todo-api/index.php?request=...

Analyse de la solution :

Framework global :
On retrouve l'utilisation des recyclerview comme dans la version 1.
On retrouve des fonctions communes utilisées dans plusieurs activités :

- loadPrefs permettant de charger les préférences
- loadAPIConnexion permettant de tester le lien avec l'API. Comme je n'ai pas réussi à capter l'attribut « version » ou « status » du json renvoyé lors de l'appel à l'API, je suis parti sur une solution plutot simple, bien que limite en terme de sécurité et d'évolution de la solution : si j'arrive à récupérer le hash de l'utlilisateur tom, je considère que le lien avec l'api est fait.
  Cette fonction est appelée dans les fonctions onCreate et onResume pour vérifier si l'URL de base a été changée par l'utilisateur et si le lien avec l'API est fait. Si le lien n'est pas fait (si on ne peut pas récupérer ce hash), on va alors bloquer partiellement l'application tout en prévenant l'utilisateur par un toast que le lien avec l'API ne foncitonne pas. Ainsi les listes ne seront plus affichées, les items non plus ou bien le bouton OK sera désactivé, l'utilisateur ne pourra plus de logger.
- editHash permet de stocker le hash dans les préférences de l'app.
- getLastUserFromPrefs permet d'obtenir le dernier pseudo renseigné, on l'utilise notamment pour l'auto complétion du champ pseudo dans la main activity lors du clic sur ce champ, et on l'utilise aussi pour l'afficher dans les settings.

L'ensemble des coroutines utilisées pour charger les données de l'API sont dans des « try catch », ce qui permet de ne pas faire crasher l'application et de simplement afficher des toast pour conserver une bonne expérience utilisateur si le problème vient de l'API ou de la connexion internet par exemple mais pas de l'application.

Architecture Retrofit :
On a le data provider, dans lequel est stocké l'URL de base de l'API et qui construit le service retrofit pour appeler les fonctions du fichier serviceAPI qui elles-même appellent l'API.
Dans le data provider, une fonction reStartService permet de changer le service quand l'utilisateur modifie l'url de base dans l'activity settings. Cette url de base est stockée directement dans le fichier dataprovider. Le fait de relancer le service permet de prendre en compte le changement d'url de base et de bloquer ou débloquer les activités en fonction du lien avec l'API qui est vérifié au lancement de chacune d'entre elles.
Dans ServiceAPI on retrouve les fonctions qui définissent directement les requêtes à passer à l'API

Classes du projet :
J'ai adapté les classes du projet pour pouvoir déserializer directement sans préciser de seralizedname, ainsi les items n'ont plus de titre mais un label par exemple.

Affichage du chargement :
Lors de l'appel à l'API, on affiche un gif de chargement, pour des raisons d'Expérience Utilisateur on affiche dans ShowListActivity les items en plus du chargement, sinon on a la liste des items qui apparaît et disparaît à chaque clic. Contrairement à cela, lors du chargement des listes par exemple on n'affiche pas les listes qu'une fois chargées parce qu'on ne peut pas cocher et décocher une liste.

Activité d'affichage des listes :
J'ai désactivé l'ajout de liste par manque de temps, on ne peut donc pas entrer de texte dans le champ texte mais un toast nous indique que le bouton est désactivé lors du clic sur ce dernier.

Conclusions :

On a donc un bon début d'application, avec quelques fonctionnalités manquantes mais je pense avoir fourni un code plutot sérieux en pensant à l'expérience utilisateur tout au long de la création de l'application.

Perspectives :

Il faudrait changer la vérification du lien avec l'api pour que ça ne fasse pas entrer en jeu un utilisateur, peut etre en intégrant cela directement à l'API ou en créant un utilisateur admin qui ne servirait qu'à cela.
Il faudrait biensur avec plus de temps ajouter la fonctionnalité d'ajout/suppression de listes et suppression d'items, mais cela sera proche de la solution développée.

Bibliiographie :

Cours de M. Boukadir pour les recyclerview et la gestion de retrofit, ainsi que pour les coroutines.
Informations sur les url à utiliser pour accéder à l'émulateur :
https://stackoverflow.com/questions/6760585/accessing-localhostport-from-android-emulator