DAA – Lab 4

Auteurs :
Thibault Seem
Pascal Perrenoud

Date : 19.12.2022

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

3.1 Arrêt des coroutines lors de scroll

Question : Veuillez expliquer comment votre solution s'assure qu'une éventuelle Couroutine associée à une vue (item) de la RecyclerView soit correctement stoppée lorsque l'utilisateur scrolle dans la galerie et que la vue est recyclée.

On utilise la méthode onViewRecycled() dans laquelle on force l'arrêt de toutes les coroutines du holder qui vient d'être recyclé en appelant la méthode cancel() des jobs.

3.2 Arrêt des coroutines à l'arrêt de l'app

Question : Comment pouvons-nous nous assurer que toutes les Coroutines soient correctement stoppées lorsque l'utilisateur quitte l'Activité ? Veuillez expliquer la solution que vous avez mis en œuvre, est-ce la plus adaptée ?

Il faut "cancel" tous les jobs retournés par les coroutines. Ceci impose à l'utilisateur d'attendre que tous les threads s'interrompent correctement et peut ralentir l'interraction. Le problème est également pour les développeurs qui doivent garder correctement accès aux jobs afin de pouvoir les terminer et les nettoyer, ce qui peut être une tâche fastidieuse et donc pas la meilleure. Il est ainsi préférable de créer des fonctions avec le mot-clé "suspend" qui annote les fonctions asynchrones.

Nous avons décidé de stocker tous les jobs déclenchés par le lancement des coroutines dans une mutablelist et de les cancel manuellement dans la méthode onDestroy().

3.3 Dispatcher adapté

Question : Est-ce que l'utilisation du Dispatchers.IO est le plus adapté pour des tâches de téléchargement ? Ne faudrait-il pas plutôt utiliser un autre Dispatcher, si oui lequel ? Veuillez illustrer votre réponse en effectuant quelques tests.

Les différents dispatchers ont une utilité différente. Le dispatcher IO est particulièrement adapté pour les téléchargements puisqu'ils peuvent être assez long selon le réseau et la taille de l'image. Le main ne doit pas être bloqué afin de permettre à l'UI d'être dessinée et le "default" est plutôt préférable pour les opérations CPU.

4.1 Rafraichissement ponctuel

Question : Lors du lancement de la tâche ponctuelle, comment pouvons-nous faire en sorte que la galerie soit rafraîchie ?

En appelant le notifiyDataSetChange() de l'adapter.

4.2 PeriodicTask unique

Question : Comment pouvons-nous nous assurer que la tâche périodique ne soit pas enregistrée plusieurs fois ? Vous expliquerez comment la librairie WorkManager procède pour enregistrer les différentes tâches périodiques et en particulier comment celles-ci sont réenregistrées lorsque le téléphone est redémarré.

En utilisant enqueueUniquePeriodicWork() au lieu de enqueue(). Grâce à cette méthode, nous pouvons faire en sorte de ne sauvegarder une tâche qu'une fois, grâce à son nom.

Les tâches périodiques enregistrées dans Android sont identifiées et sauvegardées dans une base de données SQLite gérée par l'OS. OS qui s'assure également de réinscrire les tâches au démarrage tout en ne gardant qu'une copie par tâche.