DAA - Laboratoire 05

Développement d'Applications Android

Martins Alexis, Saez Pablo

18 décembre 2023

Table des matières

1 Introduction

Ce rapport détaille le développement d'une application de galerie d'images pour le cours de Développement d'Applications Android. Notre projet se concentre sur l'implémentation de fonctionnalités clés telles que l'affichage efficace des images, et la gestion des interactions utilisateur.

2 Questions

2.1 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 `lifecycleScope` dans MainActivity.kt pour aligner les coroutines avec le cycle de vie de l'activité principale, garantissant ainsi une gestion sûre et efficace des coroutines à ce niveau. En ce qui concerne la RecyclerView, nous avons choisi une méthode distincte. Chaque élément de la liste gère ses propres coroutines, qui sont annulées et relancées dans la méthode bind de l'adaptateur chaque fois qu'une vue est recyclée. Cette stratégie assure que les opérations asynchrones associées aux éléments de la liste, telles que le téléchargement et l'affichage d'images, sont interrompues de manière appropriée lors du recyclage des vues, optimisant la gestion de la mémoire et les performances.

2.2 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 mise en œuvre, est-ce la plus adaptée ?

Notre application Android utilise lifecycleScope dans MainActivity pour s'assurer que les coroutines sont arrêtées lorsque l'activité est terminée. Ce même lifecycleScope est transmis comme argument au RecyclerViewAdapter. En conséquence, les coroutines associées à ce scope sont automatiquement interrompues lors de la destruction de l'activité, comme lorsqu'un utilisateur quitte cette activité.

2.3 Est-ce que l'utilisation du Dispatchers.IO est la plus adaptée pour des tâches de téléchargement ? Ou faut-il plutôt utiliser un autre Dispatcher, si oui lequel ? Veuillez illustrer votre réponse en effectuant quelques tests.

Alors je trouve que le meilleur moyen de répondre à cette question est de directement consulter la documention en ligne du guide de dévleoppement android officiel. On peut lire ci-dessous que les Dispatcher IO sont en effet le meilleur type de Dispatcher à utiliser dans le cas d'opéraitons en réseaux et/ou dans la manipulation de données sur le disque. Ce qui correspond totalement à notre cas.

> Dispatchers.Main - Use this dispatcher to run a coroutine on the main Android thread. This should be used only for interacting with the UI and performing quick work. Examples include calling suspend functions, running Android UI framework operations, and updating LiveData objects.
>
> Dispatchers.IO - This dispatcher is optimized to perform disk or network I/O outside of the main thread. Examples include using the Room component, reading from or writing to files, and running any network

> operations.
>
> Dispatchers.Default - This dispatcher is optimized to perform CPU-intensive work outside of the main thread. Example use cases include sorting a list and parsing JSON.

Pour les tests réalisés de notre côté, on a simplement reamrqué que le programme ne marchait pas avec un Dispatcher Main. Et qu'il était plus lent lorsque l'on utilisait le Dispatcher par défaut.

2.4 Nous souhaitons que l'utilisateur puisse cliquer sur une des images de la galerie afin de pouvoir, par exemple, l'ouvrir en plein écran. Comment peut-on mettre en place cette fonctionnalité avec une RecyclerView? Comment faire en sorte que l'utilisateur obtienne un feedback visuel lui indiquant que son clic a bien été effectué, sur la bonne vue.

On ne sait pas s'il fallait complètement implémenter cette feature ou juste montrer qu'on savait faire un petit toast qui servait de pop-up à l'écran pour montrer qu'on cliquait sur le bon. Mais bon qui peut le plus, peut le moins donc on a décider de au pire faire trop que de faire pas assez :).

Voici ci-dessous les ajouts qui ont été fait afin de pouvoir cliquer sur icones et qu'elles s'affichent en grand. Si ce n'était pas demander d'aller aussi loin, on espère avoir un petit bonus de fin d'année ;).

- **Modification de l'Adaptateur RecyclerView** : Implémentez une interface dans `RecyclerViewAdapter` pour gérer les clics, avec une méthode transmettant la position de l'élément cliqué. Configurez un gestionnaire de clics dans `ViewHolder` pour chaque élément d'image.
- **Implémentation du Gestionnaire de Clics dans l'Activité** : Dans `MainActivity`, implémentez l'interface de gestionnaire de clics. Lors d'un clic sur un élément, lancez `FullScreenImageActivity` pour afficher l'image en plein écran.
- **Affichage en Plein Écran** : Créez `FullScreenImageActivity` avec un `ImageView` en plein écran. Transmettez les informations de l'image (URL) à cette activité pour l'afficher.

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

Pour actualiser la galerie lors de l'exécution d'une tâche ponctuelle, nous avons utilisé WorkManager avec un PeriodicWorkRequestBuilder. Cette approche permet de déclencher une mise à jour régulière des données affichées. Lorsque cette tâche s'exécute, elle notifie le changement de données au RecyclerViewAdapter en utilisant la méthode notifyDataSetChanged. Cela a pour effet de rafraîchir la visualisation des éléments, assurant ainsi que la galerie affiche les informations les plus récentes.

2.6 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 ré-enregistrées lorsque le téléphone est redémarré.

WorkManager assure l'unicité des tâches périodiques grâce à des identifiants uniques, évitant ainsi les enregistrements multiples. Les tâches persistent même après un redémarrage du téléphone, car WorkManager gère leur réactivation automatique. Cela garantit que les tâches périodiques continuent de fonctionner sans nécessiter une nouvelle programmation après chaque redémarrage de l'appareil.

3 Détails d'implémentation

- **RecyclerView et Adaptateur** : Utilisation d'une `RecyclerView` pour l'affichage des images. Le `RecyclerViewAdapter` gère le rendu et le cache des images pour une performance optimale.
- **Gestion Asynchrone avec Coroutines** : Les Coroutines facilitent les opérations asynchrones, améliorant l'expérience utilisateur grâce à un chargement rapide et fluide des images.
- **Interactions Utilisateur** : Fonctionnalité de clic pour ouvrir les images en plein écran, offrant une interaction intuitive et engageante.
- **Persistante et Gestion des Tâches avec WorkManager** : WorkManager assure la continuité des tâches en arrière-plan, même après un redémarrage de l'appareil, augmentant la fiabilité de l'application.
- **Conception de l'Interface Utilisateur** : Interface utilisateur simple et réactive pour une expérience utilisateur améliorée.

4 Conclusion

En conclusion, notre projet reflète une combinaison réussie de connaissances théoriques et de compétences pratiques dans le domaine du développement Android. Les défis rencontrés et les solutions apportées témoignent de notre compréhension approfondie des concepts clés et des meilleures pratiques.