

# Développement Android

## Laboratoire n°5

### Tâches asynchrones et Coroutines

#### *Galerie d'images*

28.11.2024

## Introduction

Ce laboratoire consiste au développement d'une application *Android* représentant une galerie d'images. Nous allons utiliser le cas d'une galerie d'images à télécharger en ligne pour illustrer les concepts de tâches asynchrones (*Coroutines* et *WorkManager*) ainsi que la gestion du cache d'une application.

## Manipulations

### 1. Mise en place

Pour ce laboratoire vous allez partir d'une application vide (Projet Android template) à laquelle vous ajouterez et implémenterez les différents composants architecturaux permettant la réalisation de l'application décrite dans ce document.

Nous vous fournissons l'élément suivant :

Un serveur, accessible depuis Internet, permettant de générer des images « à la volée ». Les images disponibles sont accessibles via l'url <https://daa.iict.ch/images/xxx.jpg> où xxx est un entier entre 1 et 10'000. La Fig. 1 représente une des images générées :



Figure 1 - Exemple d'image retournée par le serveur

## 2. Conception du squelette de l'Activité

Cette application consistera en une seule *Activité* accueillant une *RecyclerView* avec un *GridLayoutManager* pour afficher les images sous la forme d'une galerie. La Fig. 2 représente une proposition d'interface pour cette application. Vous noterez le bouton dans le *Menu* qui servira à vider le cache local des images ainsi que les *ProgressBars* servant à indiquer à l'utilisateur que certaines images sont en cours de téléchargement.

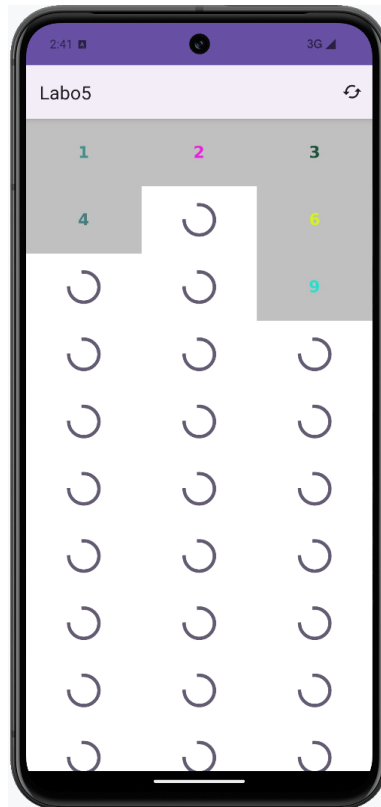


Figure 2 - Proposition d'interface pour l'application

## 3. Mise en place de l'Adapteur et des Coroutines

Vous allez devoir, dans cette partie, concevoir et développer l'*Adapteur* qui servira à populer la *RecyclerView* avec les images téléchargées. Pour des raisons de performance, nous garderons une copie des images dans le cache de l'application. Celles-ci ne seront téléchargées que si elles n'existent pas dans le cache ou alors si la copie dans le cache est plus ancienne qu'un certain nombre de minutes (par exemple 5 minutes pour vos tests).

Toutes les opérations sur le cache et les images devront se faire dans des *Coroutines* s'exécutant dans le pool de threads le plus adapté (*Dispatchers.Main*, *Dispatchers.IO*, *Dispatchers.Default*), vous veillerez donc à bien séparer les différentes étapes sous la forme de méthodes suspensives. En cas de réutilisation d'une vue il faudra veiller à stopper une éventuelle *Coroutine* existante et si l'utilisateur quitte l'*Activité* il faudra alors également s'assurer de stopper toutes les *Couroutines* en cours.

### Questions associées:

Veuillez répondre aux deux questions suivantes. Pour chacune d'entre elles, vous développerez votre réponse et l'illustrerez, si demandé, par des extraits de code.

- 3.1 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.
- 3.2 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 ?
- 3.3 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.
- 3.4 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.

#### 4. Nettoyage automatique du cache

Nous souhaitons à présent pouvoir vider le cache local des images, soit ponctuellement lorsque l'utilisateur appuie sur le bouton du *menu* ou alors de manière régulière (par exemple toutes les 15 minutes). Pour réaliser cela, nous allons utiliser la librairie *WorkManager* de *Jetpack*, avec une tâche (un *Worker*) pouvant s'exécuter soit à la demande *OneTimeWorkRequestBuilder*, soit de façon périodique *PeriodicWorkRequestBuilder*.

##### Questions associées:

Veuillez répondre aux deux questions suivantes. Pour chacune d'entre elles, vous développerez votre réponse et l'illustrerez, si demandé, par des extraits de code.

- 4.1 Lors du lancement de la tâche ponctuelle, comment pouvons-nous faire en sorte que la galerie soit rafraîchie ?
- 4.2 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.

Remarque : il n'est pas possible d'enregistrer des tâches périodiques avec une période de moins de 15 minutes, cela rend cette étape très longue et difficile à tester, prévoyez suffisamment de temps.

## Durée / Evaluation

- 4 périodes
- A rendre le lundi **9.12.2024 à 23h55** au plus tard.
- Pour rendre votre code, nous vous demandons de bien vouloir zipper votre projet Android Studio en veillant à bien supprimer les dossiers build (à la racine et dans app/) pour limiter la taille du rendu. Vous remettrez également un document **pdf** comportant les explications sur l'implémentation de votre solution ainsi que les réponses aux questions posées.
- Merci de rendre votre travail sur *CyberLearn* dans un zip unique. N'oubliez pas d'indiquer vos noms dans le code, sur vos réponses et de commenter vos solutions.