Introduction

Ce cinquième travail pratique est plus conséquent que les précédents mais couvre des matières particulièrement importantes.

La première thématique est l'accès au réseau. A l'heure actuelle, il est difficile de penser à une application ne se connectant pas à internet. C'est pourquoi, il est très important de savoir envoyer des requêtes sur un service et de traiter les données en retour.

La deuxième thématique est l'architecture de l'application. Ce sujet est souvent mis de côté mais permet d'implémenter une application robuste, maintenable et testable. Il existe plusieurs architectures, chacune avec leurs avantages et inconvénients. L'architecture étudiée ici se nomme MVVM (Model-View-ViewModel) et peut être implémentée avec la collection de librairies Architecture Components.

Finalement, la dernière thématique est la persistance. Jusque-là, les applications qui ont été développées perdent leurs données une fois quittées. Il serait par exemple intéressant de sauver le plus haut score du jeu de la taupe. Ce travail pratique couvre justement la gestion d'une base de données locale ainsi que les préférences utilisateurs.

Objectifs

Voici une liste précise des objectifs à atteindre. Ces objectifs servent de guide sur la matière à savoir et permettent de cibler les recherches pendant l'étude du contenu du cours.

- Comprendre et implémenter l'architecture MVVM,
  - Utiliser les ViewModel et ViewModelProvider,
  - Utiliser un dépôt pour la logique (Repository),
- Comprendre et savoir utiliser le DataBinding
  - Activer DataBinding au sein d'un projet,
  - Ajouter des variables dans les fichiers d'interface,
  - Récupérer des interfaces avec DataBindingUtils,
- Utiliser LiveData et MutableLiveData,
- Implémenter des requêtes HTTP avec Volley,
- Convertir des données JSON en objet du code,
- Comprendre et implémenter les préférences utilisateurs
- Comprendre et utiliser Room pour stocker des données,
  - Utiliser des entités,
  - Utiliser des Data Access Object (DAO),
  - Utiliser des TypeConverter,

Travail pratique

Le travail pratique abordé ici se nomme « Trivia ». Il consiste en un jeu de questions-réponses proposant plusieurs catégories et des questions à choix multiples. Les questions sont téléchargées depuis une base de données en ligne qui propose sa propre API (https://opentdb.com/).

L'interface graphique comporte 5 écrans qui sont chacun composé des composants graphiques suivants :

- Écran d'accueil :
  - Liste personnalisée affichant les catégories de questions
    - Une carte cliquable avec un label pour le nom de la catégorie
- Écran de jeu :
  - Un label pour la catégorie de la question,

  - Un label pour la difficulté de la question,
  - Un label pour le texte de la question,
  - Quatre cartes cliquables avec les labels pour les réponses possibles,
- Écran du résultat :
  - Une image pour la médaille gagnée,
  - Un label pour le texte accompagnant la médaille,
  - Deux labels pour le score obtenu,
- Écran de profile (statistiques du joueur) :
  - Trois images pour afficher les trois médailles possibles,
  - Trois labels pour le nombre de médailles obtenues,
  - Huit labels pour d'autres statistiques,
- Écran d'historique :
  - Liste personnalisée affichant les questions jouées dans une carte,
    - Un label pour la catégorie,
    - Un label pour la question,
    - Une image pour l'état de réussite de la question.

Le jeu se déroule en 3 phases qui sont les suivantes :

1. Lorsque l'application est ouverte, un premier écran affiche une liste des catégories de questions qui sont directement téléchargées depuis internet (Figure 1) (https://opentdb.com/).
2. Après un clic sur une catégorie, l'application télécharge un certain nombre de questions liées à cette catégorie puis affiche un nouvel écran avec la première question (Figure 2). Lorsque le joueur clique sur une réponse, celle-ci devient rouge ou verte respectivement si la question a été répondue incorrectement (Figure 2) ou correctement (Figure 5). La bonne réponse est affichée en vert et la mauvaise en rouge. Un compte-à-rebours affiche la question répondue pendant un certain temps puis passe à la suivante automatiquement.
3. A la suite de la dernière question répondue, un écran de résultat est affiché et présente le score obtenu (Figure 3). Aucune action n'est possible sur cet écran, à l'exception de revenir sur la liste des catégories à l'aide du bouton de retour. Les seuils des différentes médailles ainsi que le nombre de questions par partie sont libres.

Deux écrans supplémentaires existent et sont accessibles depuis le menu de l'écran des catégories. Ceux-ci sont les statistiques (profile joueur) qui recensent plusieurs statistiques enregistrées au fil des parties (Figure 6) et un historique permettant de voir toutes les questions qui ont été jouées (correctes ou incorrectes) (Figures 4 et 5). Ces deux fonctionnalités nécessitent de sauver les données de manière persistante. Une base de données locale est utilisée pour sauvegarder toutes les questions jouées. Pour ceci, toutes les informations (attributs) qui définissent une question doivent être enregistrées pour que le visionnage d'une question jouée soit complet.

Afin d'enregistrer les statistiques du joueur, les préférences utilisateurs doivent être utilisées. Les statistiques sont les suivantes :

- Le nombre de médailles d'or gagnées,
- Le nombre de médailles d'argent gagnées,
- Le nombre de médailles de bronze gagnées,
- Le nombre de parties terminées,
- Le nombre de questions répondues (qr),
- Le nombre de questions répondues correctement (qc),
- Le pourcentage de réussite du joueur (qc / qr * 100),

Cette donnée est aussi disponible sous forme de codelab à l'adresse suivante (connexion à GitLab requise) : https://mobapp.pages.forge.hefr.ch/. De plus, elle contient une marche à suivre servant de guide pour la bonne réalisation du travail pratique car celui-ci peut sembler compliqué au premier abord.

> **REMARQUE**
>
> La base de données en ligne de questions comporte deux types de question : des questions à choix multiples (QCM) et des questions vraies ou fausses (V/F). Dans ce travail pratique, notamment l'interface fournie, il y a uniquement le support du type QCM. Il est donc important de bien paramétrer la requête pour uniquement récupérer le bon type de question.

Contraintes

Voici une liste de contraintes à respecter obligatoirement. Ces contraintes forcent à utiliser certains points précis de la thématique et permettent de bien la comprendre :

- L'interface graphique doit avoir tous les éléments cités dans la description du travail pratique,
- Le comportement de l'application doit être identique à celui décrit dans la description du travail pratique
- L'utilisation de la méthode "findViewById()" est interdite, uniquement le Data Binding est autorisé,
- Les composants "ViewModel" et "LiveData" doivent être utilisées (architecture MVVM),
- Les préférences partagées sont à utiliser pour les statistiques,
- Le composant Room est à utiliser pour gérer la base de données,

Conseils

Afin d'éviter des problèmes lors de la réalisation du travail pratique, voici quelques conseils permettant de prendre les bonnes directions :

- Il est conseillé de suivre la marche à suivre dans le codelab,
- Il est conseillé d'installer et de tester l'application fournie qui sert de référence,
- Il est conseillé de prendre connaissance des différentes ressources à disposition (layout, strings, colors, drawables) pour faciliter la suite du travail,
- Certains champs de la réponse JSON que renvoie le serveur contiennent des caractère "HTML" qu'il faut supprimer. La méthode "Html.fromHtml(<variable à décoder>).toString()" permet cela.
- La méthode "setBackgroundColor(<couleur>)" sur une View permet de changer la couleur de fond. Pour la supprimer, la couleur "Color.TRANSPARENT" peut être utilisée,
- La gestion de l'orientation paysage n'est pas gérée par les layouts et n'est pas à faire,
- La mise en pause et reprise du timer lorsque l'application est quittée (comme vu au TP02) n'a pas besoin d'être géré.

Ressources disponibles

Afin de permettre une concentration optimale sur les thématiques du travail pratique, les ressources suivantes sont à votre disposition sur Cyberlearn :

- APK de l'application terminée (.apk),
- Projet de départ du Trivia (navigation et interfaces complètes) (.zip).

Captures d'écran

Voici un exemple du résultat final qui peut servir d'inspiration pour le travail pratique. Il n'est pas nécessaire que l'interface ressemble exactement à ces captures d'écran : le plus important est que l'application soit fonctionnelle et qu'elle respecte la description et les contraintes du travail.

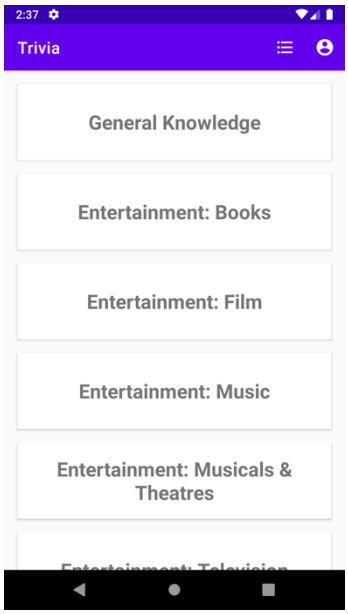

Figure 1 – Liste des catégories

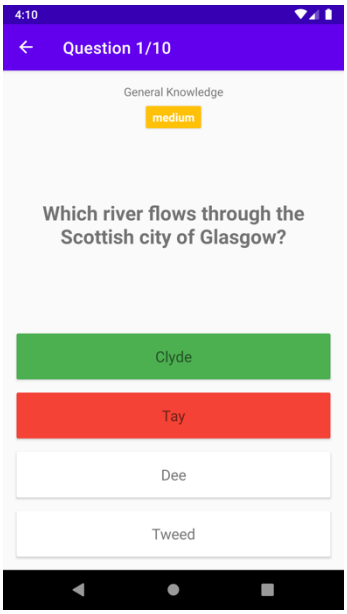

Figure 2 – Question fausse

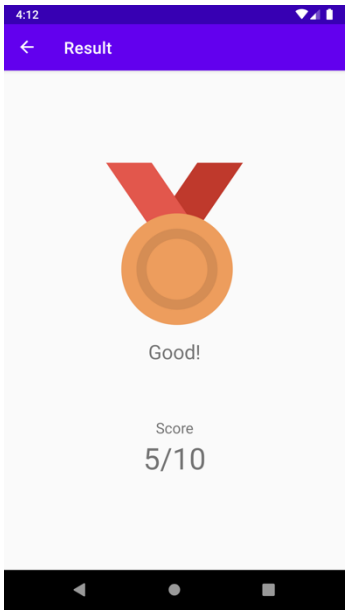

Figure 3 – Résultat

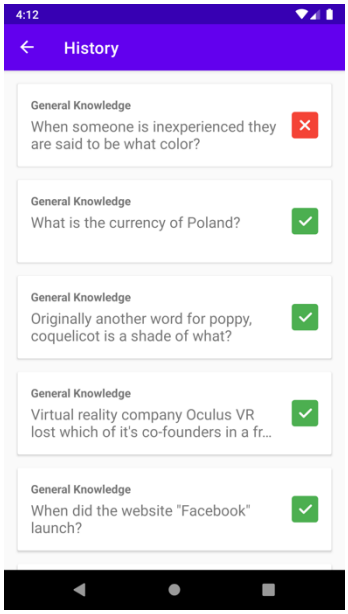

Figure 4 – Historique

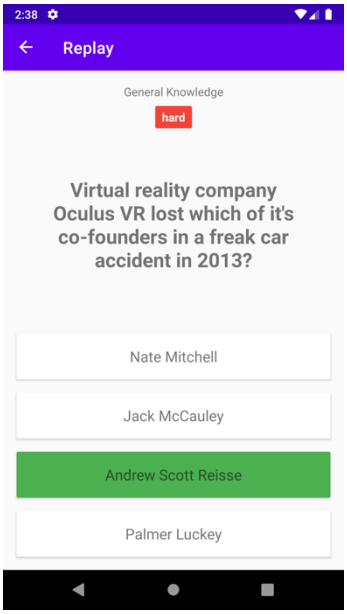

Figure 5 – Replay question

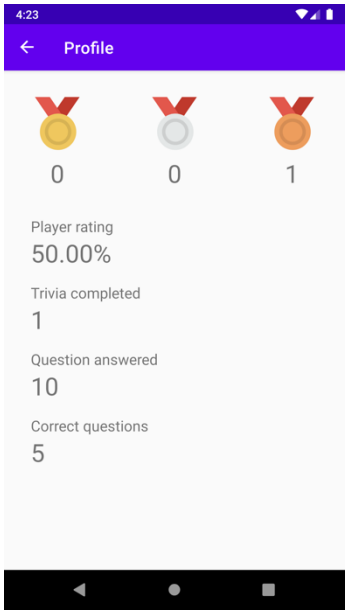

Figure 6 – Statistiques

Informations

Voici quelques liens utiles contenant dans un premier temps beaucoup d'informations concernant les différentes thématiques abordées. Il est conseillé d'accéder à ces liens dans l'ordre dans lequel ils sont listés afin de faciliter la compréhension de la matière.

- Envoyer des requêtes avec Volley (jusqu'à la section "Use newRequestQueue" comprise) : https://developer.android.com/training/volley/simple
- Tutoriel pour décoder du JSON : https://www.tutorialspoint.com/android/android_json_parser.htm

- Guide d'architecture d'application (complément à la théorie) : https://developer.android.com/jetpack/guide
- Guide sur comment faire du DataBinding (sections "Data object", "Binding Data" et "Event handling") : https://developer.android.com/topic/libraries/data-binding/expressions
- Guide sur les LiveData (jusqu'à la section "Transform LiveData") : https://developer.android.com/topic/libraries/architecture/livedata
- Guide sur les ViewModel (complément à la théorie) : https://developer.android.com/topic/libraries/architecture/viewmodel
- Guide pour l'utilisation de Room (complément à la théorie) : https://developer.android.com/training/data-storage/room

Rendu

Il est demandé de rendre sur la page Cyberlearn du cours un dossier compressé (.zip) nommé "tp05-<nom du groupe>" contenant les éléments suivants :

- Rapport concis (.pdf) avec les chapitres (nommé "tp05-report-<nom du groupe>")
  - Introduction
  - Problèmes rencontrés
  - Conclusion
- Code du projet (.zip) nettoyé (nommé "tp05-code-<nom du groupe>")
- Le fichier installable (.apk) de l'application (nommé "tp05-app-<nom du groupe>")

N'hésitez pas à décrire votre ressenti par rapport à ce travail pratique. Par exemple, si la quantité de travail, la durée, la complexité, l'intérêt qu'il sucite vous ont semble adaptés ou non et si possible expliquer pourquoi. Ceci permet d'adapter le travail pratique en cas de problème.

Le rendu du devoir est fixé au **Jeudi 16.12.21 à 23h59** et est à rendre par groupe. Une personne peut déposer le travail pour les deux membres du groupe.

Checklist

Voici une checklist pour éviter d'oublier certains aspects du travail demandé ce qui pourrait enlever des points lors de l'évaluation :

- ☐ Téléchargement des catégories et des questions depuis internet,
- ☐ Utilisation du DataBinding pour réagir aux interactions de l'utilisateur,
- ☐ Utilisation du DataBinding avec LiveData pour actualiser l'interface,
- ☐ Utilisation de l'architecture MVVM avec un ViewModel pour chaque écran,
- ☐ Implémentation de la logique du Trivia dans un Repository,
- ☐ Mise en place d'une base de données Room pour sauver un historique de questions jouées,
- ☐ Utilisation des préférences utilisateurs pour sauver les statistiques.