RES508 - Qualité de développement Android - Projet noté

2025-2026

Jocelyn CARAMAN

15 janvier 2026

1 Objectifs

Créer une application Android *Mes jeux*. Elle permettra d'effectuer une recherche sur une base de données en ligne de jeux vidéos, d'afficher les résultats sous forme de liste et d'ouvrir une page détaillée pour chaque jeu. Une page dédiée regroupera la collection de jeux vidéos de l'utilisateur. Les informations sur les jeux seront récupérées par l'intermédiaire de l'API IGDB.com.

2 Modalités

— Travail à réaliser en **binôme**. Exceptionnellement, un **un seul trinôme** sera autorisé (car nous sommes un nombre impair) mais je serais moins tolérant sur la notation de ce groupe.

— Le projet sera abordé durant le cours du jeudi 15 janvier et sera à rendre au plus tard le **dimanche 15 février 2026** à **23h59**.

— Le code du projet sera à rendre par l'intermédiaire de GitHub Classroom, dans la classe nommée *IUTACY-BUT3-INFO-A - 2025-2026* , vous trouverez le devoir de groupe *S5A08 - Projet final*. Il vous faudra un compte GitHub. Le lien pour vous permettre de créer un groupe ou d'en rejoindre un existant sera sur Moodle. Merci de nommer votre groupe avec vos deux (ou trois) noms de famille séparer par un espace (exemple DUPONT-MARTIN EL HAIDI).

— La *version minimale* est tout ce qui est attendu au minimum pour le projet et permet d'avoir environ 13/20 si tout est fait correctement. Les *versions avancée* et *très avancée* regroupent des fonctionnalités supplémentaires permettant d'obtenir plus de points voire des points bonus. La grille de notation détaillée sera disponible sur Moodle.

3 Besoins

3.1 Version minimale (requis)

— Dans cette version minimale, l'application permet d'afficher, dès son ouverture, une **liste de jeux** à partir de l'**API** mais il n'y a pas de fonction de recherche intégrée : la requête est défini en dur dans le code. On peut afficher le **détail** d'un jeu. On peut également ajouter un jeu à notre bibliothèque mais cette information n'est pas persistée après redémarrage de l'app. La présence ou non d'un jeu dans la collection est représentée par une icône d'étoile. Une barre de navigation est présente en bas de l'écran et permet de naviguer entre les écrans liste/détails **ET** la bibliothèque

de jeux(2 entrées dans la barre de navigation donc).

— Vous devez écrire **2 tests unitaires** et **1 test d'instrumentation** :
  — un test unitaire sur le view model qui vérifie que l'appel de la méthode permettant de récupérer la liste de jeux depuis l'API renvoie bien le résultat attendu quand on utilise une implémentation mocké (une fausse implémentation avec des données en dur dans la classe) du repository.
  — un deuxième test unitaire sur le view model qui permet de vérifier l'ajout d'un jeu à la collection.
  — un test d'instrumentation qui vérifie qu'à l'ouverture de l'application, la liste de jeux est affichée en vérifiant le titre de la page. Puis, il vérifie la présence du bouton d'accès dans la barre de navigation vers l'écran de collection et que le titre correspondant est bien affiché. Ensuite, il clique dessus et vérifie la présence du titre pour l'écran de collection.

— Le code doit respecter les **recommandations d'architecture** vues en cours (architectures en couches).

— Le thème est **personnalisé**, l'interface est **soignée** et vous intégrerez **Material Design** dans le choix des composants.

3.1.1 Écran de recherche de jeux

— Implémentez un **écran** pour afficher la **liste de jeux** récupérée depuis l'API grâce à la bibliothèque Retrofit. L'affichage peut être sous forme de liste verticale ou en grille, dans les deux cas, on peut la faire défiler. Les informations à afficher pour chaque élément sont : l'**image** renvoyé par le champ *cover*, le **nom du jeu**, la **première date de sortie** et les **plateformes** sur lesquels le jeu est disponible (Nintendo Switch 2, PlayStation 5, PC...). Vous afficherez la plateforme sous forme d'icône.

— L'**icône "étoile"** sera également affichée pour chaque jeu. Elle sera remplie quand l'élément est présent dans la collection et vide sinon. Un appui dessus ajoute ou retire l'élément de la collection suivant s'il était déjà présent.

— Un **titre** est présent en haut dans une barre d'application (app bar). Une barre de navigation est présente en bas de l'écran avec 2 entrées : "Recherche" et "Bibliothèque". Une icône accompagne le label. L'entrée "Recherche" est en surbrillance car c'est l'écran affiché actuellement.

— Au clic sur un **élément de la liste**, on bascule sur la vue **détaillée** du jeu. Plus de détails dans la section 3.1.2.

3.1.2 Écran de détails d'un jeu

— Implémentez l'**écran de détails d'un jeu** qui affiche plus d'informations sur celui-ci. Voir le Tableau 1 pour connaître toutes ce qu'il y a à afficher. Vous êtes libre sur l'interface et l'agencement de cet écran hormis les points ci-dessous.

— Le **titre** dans l'app bar affiche maintenant le nom du jeu. La barre de navigation a toujours "Recherche" en surbrillance.

— Un **bouton** sous forme de flèche vers la gauche est visible dans la barre d'application permettant de retourner à l'écran précédent (écran de recherche).

— L'**étoile** est également présente sur cet écran. Elle sera remplie quand l'élément est présent dans la collection et vide sinon. Un appui dessus ajoute ou retire l'élément de la collection suivant s'il était déjà présent.

Nom	Description
identifiant	identifiant unique du jeu fourni par IGDB
nom	nom du jeux vidéo
image	une image du jeu, utiliser le champ **cover**
date de sortie	date de la première sortie du jeu dans le monde
genres	les genres du jeu en anglais (course, FPS, RPG...)
description	une courte description du jeu
plateformes	les supports sur lesquels le jeu est disponible, à afficher sous forme d'**icône**
éditeurs	publishers en anglais, la ou les sociétés qui vendent le jeu
développeurs	les sociétés qui développent le jeu (développeurs principaux seulement)

TABLE 1 – Informations à afficher pour un jeu

3.1.3 Écran de collection de jeux

— Implémentez l'**écran** pour afficher la **collection de jeux**. L'affichage peut être sous forme de liste verticale ou en grille, dans les deux cas, on peut la faire défiler. Les informations **sont toujours récupérées de l'API** grâce à Retrofit comme pour l'écran de recherche, à la différence que **seuls les jeux de votre collection** doivent être retournés en modifiant le body de la requête. Pour chaque élément, affichez les mêmes informations que sur l'écran de recherche (image, nom du jeu, date de première sortie et les plateformes sous forme d'icônes).

— Il y a également l'**étoile** pour retirer un élément de la collection qui fonctionne de la même façon que sur les autres écrans.

— Le **titre** dans l'app bar affiche le nom de l'écran. La barre de navigation est également présente et l'élément "Collection" apparaît sélectionnée.

— Comme sur l'écran de recherche, **taper** sur un élément de la collection ouvre les informations détails du jeux comme détaillé dans la section 3.1.2.

3.2 Version avancée (fonctionnalités supplémentaires pour avoir plus de points)

— Utiliser la bibliothèque **Jetpack Navigation with Compose** pour gérer la navigation entre les écrans de l'application.

— Afficher les **images** des jeux dans une **meilleure qualité**. Parcourez la documentation de IGDB.com pour trouver comment faire.

3.3 Version très avancée

— Utilisez une bibliothèque de **persistance de données DataStore** ou **Room** vues en annexe du cours, pour sauvegarder la collection de jeux. Cela vous permettra de garder en mémoire l'identifiant de chaque jeu de la collection que vous détenez afin de les retrouver après avoir quitté l'application.
Note : dans ce cas, pour repartir de zéro et supprimer votre collection actuelle, vous pouvez supprimer les données de l'application dans
Paramètres > *Applications* > *Le_nom_de_votre_app* > *Données* ou *Stockage* > *Supprimer les données*.

— Ajoutez une fonctionnalité de **recherche de jeux**. Sur l'écran de recherche, une **barre de recherche** est présente tout en haut de l'écran. Saisissez une requête puis validez pour afficher les résultats de la recherche en utilisant l'API IGDB. De plus, vous rajouterez un bouton pour **trier** les résultats par **nom** ou par **date de première sortie**. Enfin, vous rajouterez également un bouton pour **filtrer** les résultats par **plateforme** (consoles, PC, mobile...). L'utilisateur pourra sélectionner une ou plusieurs plateformes, dans ce cas, les résultats correspondront aux jeux ayant au moins une plateforme qui correspond aux filtres (OU logique). Si aucune plateforme n'est sélectionnée, les jeux ne sont pas filtré par plateforme.

4 Moyens

Un dépôt sera crée pour chaque groupe et contiendra le code de départ du projet pour éviter les erreurs de configuration.

Tous les groupes partageront un même identifiant client et chacun groupe se verra attribuer un **token d'accès** à l'API. Ce token sera valable environ 60 jours mais cela devrait être largement suffisant pour la réalisation de ce projet. Le **token** et le **clientId** seront affichés dans le **README** de votre dépôt.

La documentation de l'API se trouve ici : `https://api-docs.igdb.com/?shell`. Pensez bien à suivre la section #Requests pour comprendre comment utiliser les **clientId** et **accessToken** pour faire vos requêtes.

Vous ne devriez avoir besoin que de l'endpoint Game (`https://api.igdb.com/v4/games`) pour l'utilité du TP. Aidez-vous des références dans la documentation, notamment les sections :
— Fields pour récupérer seulement les champs qui nous intéressent ;
— Expander pour obtenir des informations supplémentaires normalement accessibles à un autre endpoint (par exemple cover.url pour obtenir lien url vers la couverture du jeu) ;
— Images pour afficher les images avec une meilleure qualité ;
— Filters pour ajouter des filtres à vos recherche.

Pour ceux qui ne feront n'implémenteront pas la recherche, voici le body d'une requête pour l'API que vous pourrez utiliser :

```
fields name; sort first_release_date desc;
where game_type = 0 & involved_companies.publisher = true &
involved_companies.company = (70,10100,17966); limit 30;
```

Cette requête affiche le nom des jeux de type *jeu principal* ayant comme éditeur Nintendo, Playstation ou Microsoft Xbox ; triés par date de première sortie du plus ancien au plus récent.

5 Livrable

Le code su projet sera rendu par l'intermédiaire de GitHub. Il ne sera pas possible de pousser des modifications après la date limite de rendu.

Il est attendu que le code :
— Soit clair et lisible ;
— Respecte au maximum les guides de style Kotlin (abordés dans la première partie du cours) ;
— Soit commenté quand nécessaire ;
— Ne crash pas (penser à la gestion des erreurs avec les blocks try/catch).
— Ne soit pas généré pas de l'**IA**.

⚠ **Attention** : Tout code lié à des fonctionnalités non mentionnées dans ce document **ne seront pas pris en compte pour la notation**. Vous devez vous contraindre aux besoins spécifiés dans ce document, ni plus, ni moins (pour avoir tous les points).