Programmation mobile et réalité augmentée :

TEA 01

Élèves : Adriana Gabriela IOCCHI SAETTONE

Bruno NASCIMENTO BRASILINO DE FREITAS

04/06/2025

I. Introduction

Dans cette première version d'une application mobile, nous avons réalisé un prototype (équivalent à un mock-up) permettant de stocker des données sans backend. Ce projet, développé en Kotlin sous Android Studio, a été géré avec Git pour le travail asynchrone en équipe.

II. Analyse

Pour la gestion de version, chaque développeur a d'abord travaillé sur son propre dépôt, puis nous avons migré les changements vers le dépôt principal une fois les fonctionnalités implémentées. Une branche distincte a été créée pour chaque Activity demandée, ce qui a facilité le suivi et la relecture du code. Concernant le stockage, nous avons utilisé SharedPreferences pour sauvegarder localement des chaînes de caractères. Les listes d'éléments étaient enregistrées sous forme de Strings avec « ; » comme délimiteur dans les fichiers XML.

Les clés des SharedPreferences prenaient la forme « list_{currentPseudo} » pour la liste de listes d'un utilisateur et « list_{currentList}_{currentPseudo} » pour une liste d'items liée à un pseudo. Cette solution permettait de manipuler rapidement des données sans base locale, mais montre ses limites dès que la complexité ou la volumétrie augmentent. Enfin, ce prototype a servi à mettre en place des bonnes pratiques de structuration de code (séparation des responsabilités entre Activities, gestion des ressources XML) et à formaliser une méthodologie Git (branches par fonctionnalité, commits réguliers, revues de code).

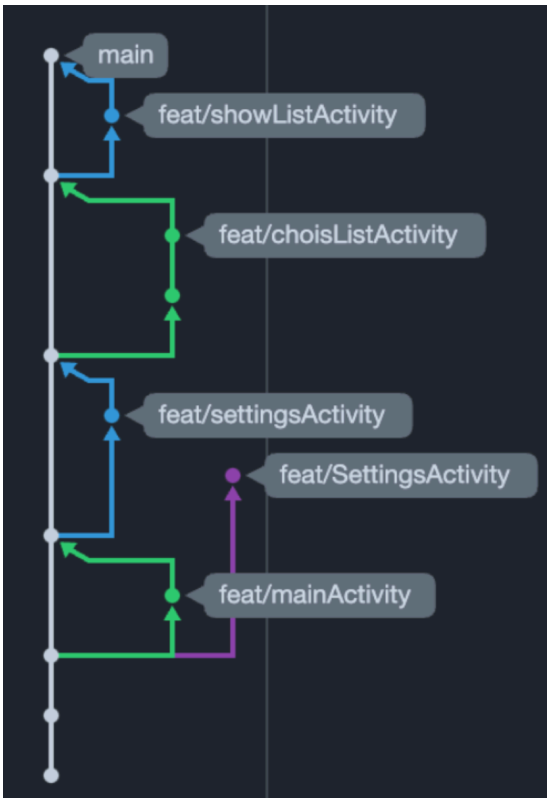

Figure 01: Représentation des branches

III. Conclusion

Cette version initiale est un bon exercice pour découvrir le développement Android et la collaboration via Git. SharedPreferences s'avère pratique pour un prototype simple, mais n'est pas une solution optimale pour gérer des relations complexes ou des requêtes avancées. En parallèle, ce projet a permis de définir des méthodes de travail collaboratif indispensables pour la suite.

La prochaine étape consiste à adopter une architecture MVVM afin de séparer clairement les couches UI (Activities/Fragments), logique métier (ViewModels) et données (Models/Repositories). Pour le stockage, l'intégration de Room offrira une gestion structurée des entités et des DAO, idéale pour un mode offline. En complément, la connexion à un backend distant (API REST ou GraphQL) via Retrofit ou Ktor permettra de stocker et synchroniser les données sur un serveur. Il faudra également prévoir une stratégie de cache et un mécanisme de synchronisation pour garantir une expérience fluide, que l'application soit en ligne ou non.

IV. Bibliographie

[01] *Kotlin Bootcamp for programmers*. (n.d.). https://www.udacity.com/course/kotlin-bootcamp-for-programmers--ud9011

[02] *OpenClassrooms*. (n.d.). OpenClassrooms. https://openclassrooms.com/fr/courses/5353106-initiez-vous-a-kotlin