Rakel Ouakil
Date : 29/06/2025

Compte-rendu - Projet PMR Séquence 3

Introduction

Après avoir connecté l'application Android à une API REST lors de la séquence précédente, cette troisième phase du projet avait pour objectif d'ajouter une persistance locale des données à l'aide de SQLite (via l'ORM Room) et de mettre en place un mode offline. L'enjeu principal était de permettre à l'utilisateur de continuer à consulter et modifier ses listes et items même sans connexion Internet, tout en assurant la synchronisation des changements dès que le réseau redevient disponible.

1. Réalisation

Pour répondre à ce nouveau cahier des charges, j'ai commencé par intégrer la bibliothèque Room dans le projet. J'ai défini des entités pour les listes et les items, ainsi que des DAO pour gérer les opérations de lecture, d'insertion et de mise à jour dans la base locale. La base de données Room est initialisée dès le lancement de l'application.

Ensuite, j'ai adapté la logique de récupération des données : À chaque récupération de listes ou d'items depuis l'API, les données sont stockées dans la base locale Room. En cas d'absence de réseau, l'application charge automatiquement les données depuis la base locale, permettant à l'utilisateur de consulter ses listes et items même hors connexion.

Pour la gestion des actions utilisateur (cocher/décocher un item), j'ai mis en place un système de marquage :
- Lorsqu'un item est modifié hors ligne, il est marqué comme "à synchroniser".
- Dès que le réseau redevient disponible, une synchronisation automatique est lancée pour envoyer les changements au serveur.

Toutes les opérations de lecture/écriture en base et de communication réseau sont réalisées en coroutines, afin de ne jamais bloquer l'interface utilisateur.

2. Difficultés rencontrées et apprentissage

La principale difficulté a été de garantir la cohérence des données entre la base locale et le serveur, notamment lors de la synchronisation après une période offline. Il a fallu gérer les cas où des modifications locales pouvaient entrer en conflit avec des données récupérées du serveur. Pour cela, j'ai mis en place une logique de fusion : les changements locaux "à synchroniser" sont toujours prioritaires lors de la mise à jour du cache local après un retour du réseau.

Un autre défi a été la gestion de l'authentification en mode offline. J'ai choisi de permettre l'accès à l'application hors ligne uniquement si un token d'authentification valide avait déjà été obtenu lors d'une précédente connexion.

Enfin, la configuration de l'environnement de développement (notamment la compatibilité entre Room, kapt et le JDK 17 sous Windows) a nécessité de nombreux ajustements, notamment au niveau des arguments JVM et du chemin du JDK.

Conclusion et perspectives

Cette séquence m'a permis de franchir une étape importante dans le développement d'applications mobiles robustes, capables de fonctionner même en l'absence de réseau. J'ai pu approfondir mes connaissances sur la persistance locale avec Room, la gestion du mode offline, la synchronisation différée et la résolution de conflits de données.

Plusieurs axes d'amélioration sont envisageables :
- Permettre la création de listes et d'items en mode offline, avec synchronisation différée.
- Améliorer la gestion des conflits lors de la synchronisation (par exemple, en notifiant l'utilisateur en cas de conflit).
- Renforcer la sécurité du stockage local (chiffrement de la base ou du token).
- Optimiser l'ergonomie en affichant des indicateurs de synchronisation et des messages plus explicites en cas de perte de connexion.