**Récapitulatif du Projet eCEP**

**État actuel de l'application**

L'application eCEP a été développée selon une architecture moderne et évolutive, en suivant les principes de Clean Architecture et du pattern MVVM (Model-View-ViewModel) avec GetX pour la gestion d'état. Nous avons mis en place toute l'infrastructure nécessaire pour l'intégration avec l'API, même si l'application fonctionne actuellement avec des données simulées.

**Fonctionnalités implémentées**

1. **Interface utilisateur complète** pour les différents rôles (élève, enseignant, parent, administrateur)
2. **Système d'authentification** avec gestion des rôles et des autorisations
3. **Module élève** avec accès aux cours, exercices, badges et suivi de progression
4. **Module enseignant** pour la gestion des classes et le suivi des élèves
5. **Module administrateur** pour la gestion des utilisateurs et du contenu
6. **Navigation intuitive** entre les différents espaces de l'application
7. **Mode hors ligne** avec possibilité de télécharger les cours

**Architecture technique**

L'application est structurée selon les principes de Clean Architecture :

* **Couche Présentation** : UI, contrôleurs, widgets personnalisés
* **Couche Domaine** : Entités, repositories (interfaces), use cases
* **Couche Data** : Implémentations des repositories, providers, modèles de données

La communication avec l'API est gérée par :

* Des classes API spécifiques pour chaque domaine (cours, exercices, utilisateurs)
* Un système centralisé de gestion des requêtes et des erreurs
* Des repositories qui transforment les données brutes en objets métier

**Difficultés rencontrées lors de l'intégration de l'API**

Malgré l'implémentation complète de l'infrastructure de connexion à l'API, nous avons rencontré certaines difficultés qui ont retardé l'intégration finale :

1. **Problèmes de synchronisation** entre le backend et le frontend
2. **Incompatibilités de format** dans certaines structures de données
3. **Gestion complexe des erreurs réseau** en environnement mobile
4. **Conflits de dépendances** dans la structure du projet Flutter

Pour maintenir le calendrier du projet, nous avons pris la décision d'utiliser temporairement des données simulées (mock data) qui reflètent fidèlement la structure des données attendues de l'API. Cette approche nous a permis de terminer le développement de l'interface utilisateur et de toute la logique métier sans être bloqués par les problèmes d'intégration.

**État de l'API REST**

L'API REST a été entièrement développée selon une architecture RESTful avec les fonctionnalités suivantes :

* **Endpoints complets** pour toutes les ressources nécessaires (utilisateurs, cours, exercices, badges)
* **Système d'authentification et d'autorisation** par JWT
* **Documentation Swagger** pour faciliter l'intégration future
* **Tests unitaires et d'intégration** pour garantir la fiabilité

**Prochaines étapes**

Pour finaliser l'intégration API, les actions suivantes seront nécessaires :

1. **Résolution des incompatibilités de format** entre le frontend et le backend
2. **Optimisation de la gestion des erreurs** pour une meilleure expérience utilisateur
3. **Mise en place d'une stratégie de cache** pour améliorer les performances
4. **Tests d'intégration exhaustifs** entre le mobile et l'API

**Conclusion**

Bien que l'application ne soit pas encore connectée à l'API en production, toute l'infrastructure technique est en place pour cette intégration. L'application est fonctionnelle avec des données simulées et offre déjà une expérience utilisateur complète et fidèle à ce que sera le produit final.

## Prérequis

Avant de pouvoir exécuter l'application, assurez-vous d'avoir installé les éléments suivants :

1. **Flutter SDK** (version 3.10.0 ou supérieure)
2. **Dart SDK** (version 3.0.0 ou supérieure)
3. **Android Studio** ou **Visual Studio Code** avec les extensions Flutter et Dart
4. **Git** pour cloner le dépôt
5. **JDK** (Java Development Kit) version 11 ou supérieure
6. Un **émulateur Android** ou un **appareil Android physique** connecté
7. *flutter pub get pour installer les dependances*