Documentation Technique de l'Application E-Learning

Ce document présente l'architecture complète, les modèles de données, les flux d'API et la structure de l'application mobile Flutter E-Learning, conçue pour interagir avec un backend Django REST Framework (DRF).

1. Feuille de Route du Projet (Roadmap)

Cette section détaille les étapes du développement, avec un statut clair sur la progression du projet. Les étapes marquées d'un ✓ sont complétées ou en cours d'intégration.

PHASE 1 — Backend (Django)

Étape	Objectif	Statut	Remarques
1 : Authentification & Gestion utilisateurs	Modèle User personnalisé, Inscription, Connexion (JWT), Champs level et intérêts.		Jeton d'authentification simulé ou réel (selon la version de l'API) est géré dans le AuthProvider.
2 : Gestion des cours (Courses)	Modèles : Catégories, Cours, Modules, Leçons (vidéo, texte, quiz). API : Liste/filtre/recherche, Détails du cours, Suivi de progression.		Le flux d'API /api/courses/ est fonctionnel, géré par le CourseProvider.
3 : Tests & Évaluations	Tests de niveau, Quiz par cours, Résultats + classement.		
4 : Parcours personnalisé	Algorithme simple de recommandations basé sur : intérêts, niveau et historique.		
5 : Forum de discussion	Sujets, messages, réponses. API sécurisée.		
6 : Sécurisation + Optimisation	Permissions DRF, Pagination, Validation des données.		Pagination visible dans le parsing du CourseProvider (data['results']).

PHASE 2 — Frontend (Flutter)

Étape	Objectif	Statut	Remarques
7 : Structure & Auth	Projet Flutter initialisé, Login/Register, Stockage token, Navigation (Home/Cours/Profil).		L'AuthProvider gère le flux de navigation vers le DashboardPage.
8 : Interface Apprenant	Page des cours, Vue détaillée d'un cours, Lancement vidéos, lecture PDF, Suivi progression visible.		DashboardPage affiche la liste des cours avec progression.
9 : Tests / Quiz	Interface quiz, Affichage résultats.		

10 : Recommandations	**Page « Pour vous » personnalisée.**	**Dépend de l'Étape 4 (Backend).**
11 : Forum	**Liste des discussions, Réponses, Création de sujets.**	**Dépend de l'Étape 5 (Backend).**

2. Architecture Technique et Gestion d'État

L'application utilise une architecture Flutter standard avec le modèle Provider pour la gestion d'état centralisée.

Composant	Rôle	Fichiers Clés
AuthProvider	**Gère l'état d'authentification (isAuthenticated), le jeton utilisateur et les informations de l'utilisateur (User model).**	**lib/providers/auth_provider.dart**
CourseProvider	**Gère l'état de chargement et le catalogue des cours. Inclus une logique de simulation de données pour le mode débogage en cas d'échec de l'API.**	**lib/providers/course_provider.dart**
Modèles (Models)	**Structures de données Dart (User, Course, Lesson) pour le parsing JSON.**	**lib/models/...**
Pages UI	**Utilise context.watch() ou Provider.of() pour réagir aux changements d'état des Providers.**	**lib/screens/...**

3. Modèles de Données Clés

Les structures de données sont définies pour correspondre aux réponses JSON de l'API.

Modèle User

Représente l'utilisateur connecté.

Champ	Type	Description
id	**int**	**Identifiant unique.**
username	**String**	**Nom d'utilisateur.**
niveauInitial	**String?**	**Niveau linguistique.**

Modèle Course

Représente un cours disponible.

Champ	Type	Description

id	**int**	**Identifiant du cours.**
title	**String**	**Titre du cours.**
niveauCible	**String**	**Niveau visé par ce cours.**
progress	**double**	**Pourcentage de progression (0.0 à 1.0).**

4. Flux d'API et Robustesse

4.1 Flux de Connexion (Login)

La méthode login de l'AuthProvider est le point de départ :

1. **Requête : POST vers http://10.0.2.2:8000/api/login/.**
2. **Gestion des Erreurs : En cas d'échec, une exception est levée et propagée au LoginForm pour un affichage utilisateur clair.**
3. **Correction Critique : Le retrait des délais asynchrones après une connexion réussie a permis de résoudre l'erreur setState() called after dispose() en assurant que l'état n'est mis à jour que lorsque le widget est monté.**

4.2 Flux de Chargement des Cours

Géré par la fonction fetchCourses(authToken) dans le CourseProvider.

1. **Endpoint : GET http://10.0.2.2:8000/api/courses/**
2. **Sécurité : L'en-tête Authorization: Token <AuthToken> est utilisé pour chaque requête API sécurisée.**
3. **Gestion des Données : Le parsing des cours utilise la clé results de la réponse JSON pour s'adapter à la pagination de Django REST Framework.**
4. **Affichage de l'Erreur/Simulation :**
    - **Le CourseProvider enregistre les erreurs API dans l'attribut errorMessage.**
    - **Le DashboardPage affiche ce message en grand et un bouton "Réessayer" si une erreur est détectée ou si des données de simulation ont été chargées (uniquement en mode débogage).**