📚 Gestion de Bibliothèque Numérique

Introduction :

Le projet **Gestion de Bibliothèque Numérique** est une application mobile développée avec **Flutter**. Elle permet de gérer une bibliothèque numérique en suivant l'architecture **MVVM** (Model-View-ViewModel). Cette application facilite la gestion des auteurs, des livres et des utilisateurs grâce à des fonctionnalités CRUD (Créer, Lire, Mettre à jour, Supprimer). Elle prend également en charge la gestion des rôles utilisateurs (administrateur et utilisateur) afin de restreindre l'accès aux fonctionnalités en fonction du rôle de l'utilisateur. 🔐

Architecture MVVM

L'architecture MVVM sépare les différentes responsabilités de l'application en trois composants principaux :

1. Modèle (Model)

Le modèle représente les entités de l'application sous forme de classes de données. Ces entités incluent les **auteurs**, les **livres** et les **utilisateurs**, qui sont des éléments clés de l'application.

Fichiers principaux :

- `Auteur.dart`
- `Livre.dart`
- `User.dart`

2. Repository

Le repository est responsable des interactions avec la base de données SQLite via la bibliothèque **Sqflite**. Chaque entité dispose de son propre fichier de gestion des opérations de persistance (ajout, modification, suppression, lecture).

Fichiers principaux :

- `Database.dart` : Configuration de la base de données et création des tables.
- `AuteurDatabase.dart` : Gestion des auteurs dans la base de données.
- `LivreDatabase.dart` : Gestion des livres dans la base de données.
- `UserDatabase.dart` : Gestion des utilisateurs dans la base de données.

3. ViewModel

Les ViewModels servent d'intermédiaires entre les modèles de données et les vues. Ils contiennent la logique métier et utilisent **ChangeNotifier** pour notifier les vues des changements d'état. Chaque entité a son propre ViewModel pour gérer la logique spécifique.

Fichiers principaux :

- `AuteurViewModel.dart`
- `LivreViewModel.dart`
- `UserViewModel.dart`

4. Vue (View)

Les vues sont responsables de l'interface utilisateur. Elles récupèrent les données des ViewModels et affichent ces données à l'utilisateur. Elles permettent également de transmettre les actions utilisateur vers les ViewModels pour traitement.

Fichiers principaux :

- `AjouterAuteurView.dart` : Vue pour ajouter un auteur.
- `ModifierAuteurView.dart` : Vue pour modifier un auteur.
- `AuteurListView.dart` : Vue pour afficher la liste des auteurs.
- `AjouterLivreView.dart` : Vue pour ajouter un livre.
- `LoginView.dart` : Vue pour la page de connexion.
- `HomeScreen.dart` : Vue pour l'écran principal de l'application.

Fonctionnalités

1. Gestion des Auteurs ✍️

- Ajouter, modifier, supprimer et afficher des auteurs.
- Afficher les livres associés à un auteur.

2. Gestion des Livres 📖

- Ajouter, modifier, supprimer et afficher des livres.
- Associer chaque livre à un auteur.
- Charger des jaquettes de livres depuis des fichiers locaux.

3. Gestion des Utilisateurs 👤

- Création d'utilisateurs avec des rôles (admin ou utilisateur).
- Authentification avec hachage des mots de passe via **BCrypt**.

4. Gestion des Rôles 🔑

- Les administrateurs peuvent gérer l'ensemble des utilisateurs.
- Les utilisateurs de type "user" ont un accès limité aux fonctionnalités.

Technologies Utilisées

- **Flutter** : Framework principal utilisé pour le développement de l'interface utilisateur.
- **Sqflite** : Bibliothèque SQLite utilisée pour la persistance des données.
- **Provider** : Gestion de l'état de l'application à l'aide de **ChangeNotifier**.
- **BCrypt** : Utilisé pour le hachage sécurisé des mots de passe des utilisateurs.

Conclusion

Ce projet implémente une architecture **MVVM** robuste, garantissant une séparation claire des responsabilités. Grâce à l'utilisation des meilleures pratiques de développement Flutter, l'application est flexible et évolutive, offrant ainsi une base solide pour l'ajout de nouvelles fonctionnalités à l'avenir. 🚀