Documentation Technique du Projet Bibliothèque

1. Introduction

Le projet 'Bibliothèque' est une application Flutter permettant de gérer une bibliothèque numérique. L'application permet de gérer les auteurs, les livres, ainsi que les utilisateurs avec des fonctionnalités de gestion des livres et des rôles des utilisateurs. Ce document présente une vue d'ensemble des classes, des fonctionnalités et de l'architecture du projet.

2. Modèles

2.1 Auteur

Cette classe représente un auteur dans la bibliothèque. Elle contient des informations concernant l'identifiant de l'auteur, le nom de l'auteur, ainsi que la liste des livres écrits par l'auteur.

2.2 Livre

Cette classe représente un livre dans la bibliothèque. Elle contient des informations concernant l'identifiant du livre, le nom du livre, l'auteur du livre, ainsi que l'image de couverture du livre.

2.3 Utilisateur

Cette classe représente un utilisateur de l'application. Elle contient des informations personnelles telles que le nom, le prénom, le login, le mot de passe et le rôle de l'utilisateur.

3. Repository

3.1 AuteurDatabase

Cette classe est responsable des interactions avec la base de données concernant les auteurs. Elle permet d'ajouter, de récupérer, de mettre à jour et de supprimer des auteurs dans la base de données.

3.2 LivreDatabase

Cette classe est responsable des interactions avec la base de données concernant les livres. Elle permet d'ajouter, de récupérer, de mettre à jour et de supprimer des livres dans la base de données.

3.3 UserDatabase

Cette classe est responsable des interactions avec la base de données concernant les utilisateurs. Elle permet d'ajouter, de récupérer, de mettre à jour et de supprimer des utilisateurs dans la base de données. Elle inclut aussi une fonctionnalité pour vérifier les identifiants d'un utilisateur lors de la connexion.

4. ViewModels

4.1 AuteurViewModel

Le ViewModel pour les auteurs gère les données des auteurs entre la vue et le modèle. Il permet de charger la liste des auteurs, d'ajouter, de mettre à jour et de supprimer des auteurs.

4.2 LivreViewModel

Le ViewModel pour les livres gère les données des livres entre la vue et le modèle. Il permet de charger la liste des livres, d'ajouter, de mettre à jour et de supprimer des livres, tout en mettant à jour les données des auteurs associées.

4.3 UserViewModel

Le ViewModel pour les utilisateurs gère les données des utilisateurs entre la vue et le modèle. Il permet de charger la liste des utilisateurs, d'ajouter, de mettre à jour et de supprimer des utilisateurs, ainsi que de vérifier les identifiants lors de la connexion et de gérer la déconnexion.

5. Vues

5.1 Vues des Auteurs

Les vues des auteurs permettent à l'utilisateur d'ajouter, de modifier, de supprimer et de lister les auteurs. Cela inclut la possibilité d'afficher la liste des auteurs et d'interagir avec des actions spécifiques pour chaque auteur.

5.2 Vues des Livres

Les vues des livres permettent à l'utilisateur d'ajouter, de modifier, de supprimer et de lister les livres. Cela inclut la possibilité de lier chaque livre à un auteur et d'ajouter une couverture de livre.

5.3 Vues des Utilisateurs

Les vues des utilisateurs permettent d'ajouter, de modifier, de supprimer et de lister les utilisateurs. Cela inclut la gestion des rôles des utilisateurs ainsi que la possibilité de gérer leurs informations personnelles.

6. Fonctionnalités

Le projet offre les fonctionnalités suivantes :

- Gestion des auteurs : Ajout, modification, suppression, et affichage des auteurs.

- Gestion des livres : Ajout, modification, suppression, et affichage des livres avec l'attribution d'auteurs.

- Gestion des utilisateurs : Ajout, modification, suppression, et affichage des utilisateurs, avec gestion des rôles (admin, user, invité).

- Connexion et déconnexion des utilisateurs : Système de gestion des sessions d'utilisateurs avec un contrôle des accès.

7. Architecture

L'architecture du projet est basée sur le modèle MVVM (Model-View-ViewModel). Les données sont gérées par des classes Model, les interactions utilisateur sont gérées par des ViewModels, et les interfaces utilisateur sont définies dans les Views. La communication entre les différentes couches se fait via des notifications de changement d'état (notifyListeners).