Documentation technique - Projet Bibliothèque 3.0 4DOT

I. Introduction

Le présent document constitue la documentation technique du projet Bibliothèque 3.0, qui vise à développer une application de bibliothèque modernisée comprenant une API REST avec ASP .NET CORE, une application de bureau avec AvaloniaUI. Cette documentation fournit une vue d'ensemble des différents composants, fonctionnalités et technologies utilisées dans le projet, ainsi que des instructions pour le développement, le déploiement et les tests.

Le projet Bibliothèque 3.0 vise à moderniser une application de bibliothèque en utilisant les technologies ASP .NET CORE et AvaloniaUI. L'objectif est de développer une application conviviale permettant aux bibliothécaires de gérer les livres, les emprunteurs, les prêts, les retours, Cette documentation couvre l'ensemble du projet LibraryApp, y compris les classes, les espaces de noms et les fonctionnalités spécifiques.

II. Technologies utilisées

Pour réaliser notre projet nous avions utilisés les technologies suivantes :

- ASP .NET CORE : Framework de développement web utilisé pour créer l'API REST.

- AvaloniaUI : Bibliothèque de développement d'interfaces utilisateur multiplateformes utilisée pour créer l'application de bureau AvaloniaUI.

- SQLite : Système de gestion de base de données relationnelle utilisé pour stocker les informations des livres, des emprunteurs et des prêts.

- Langage de programmation : C#

- Framework d'interface utilisateur : AVALONIA

II.1. Aperçu de l'Architecture

Il nous as été demandé pour ce projet d’utiliser soit Modèle-Vue-Modèle (MVVM), soit Framework Avalonia

Modèle-Vue-Modèle (MVVM)

Le projet LibraryApp suit l'architecture MVVM, qui est un modèle de conception permettant de séparer clairement la logique métier (Modèle), la présentation (Vue) et la gestion des interactions (Vue-Modèle). Cette approche favorise la maintenabilité, la testabilité et la réutilisabilité du code.

Framework AVALONIA

AVALONIA est un Framework d'interface utilisateur multiplateforme basé sur .NET. Il est Utilisé dans le projet LibraryApp pour créer des interfaces utilisateur modernes et réactives. Dans notre projet

nous avons choisi d’utilisés le Framework Avalonia pour créer des interfaces utilisateurs modernes et réactives.

III. Structure générale

Le projet Bibliothèque 3.0 est organisé en plusieurs composants interconnectés :

API REST (ASP .NET CORE) **:** Cette partie du projet est responsable de la création d'une API REST qui permettra aux bibliothécaires de gérer les livres, les emprunteurs, les prêts, les retours, etc. L'API communiquera avec la base de données SQLite pour stocker et récupérer les données.

Application de bureau WPF **:** Cette partie du projet est responsable du développement d'une application de bureau avec WPF. L'application fournira une interface graphique conviviale permettant aux bibliothécaires de rechercher, modifier, ajouter et supprimer des livres, des emprunteurs, des prêts, etc. L'application communiquera uniquement avec l'API REST pour accéder aux données de la bibliothèque.

Application de bureau AvaloniaUI : Cette partie du projet est responsable du développement d'une application de bureau avec AvaloniaUI. L'application offrira une interface graphique moderne et réactive pour gérer les informations des livres, des emprunteurs, des prêts, des retours, etc. Elle communiquera également avec l'API REST pour accéder aux données de la bibliothèque.

Base de données SQLite **:** Une base de données SQLite sera utilisée pour stocker les informations sur les livres, les emprunteurs, les prêts, les retours, etc. La base de données sera accessible uniquement via l'API REST.

VI. Structure technique

Namespace`LibraryApp`

Le Namespace `LibraryApp` est le principal espace de noms du projet. Il contient la classe `Program`, qui représente le point d'entrée de l'application.

Namespace `LibraryApp.ViewModels`

Le Namespace `LibraryApp.ViewModels` contient les classes de ViewModel, qui fournissent la logique métier et la liaison de données entre la Vue et le Modèle.

Namespace `LibraryApp.Views`

L'espace de noms `LibraryApp.Views` contient les classes de Vue, qui définissent l'interface de l'application.

Class `ViewLocator`

La classe `ViewLocator` implémente l'interface `IDataTemplate` et est utilisée pour mapper les ViewModels aux Views correspondantes. Elle recherche dynamiquement les classes de Vue en utilisant le nom de la classe ViewModel et instancie la Vue correspondante.

Class`Program`

La classe `Program` représente le point d'entrée de l'application. Elle utilise la méthode `BuildAvaloniaApp` pour configurer et démarrer l'application Avalonia.

V. Développement de l'API REST

L'API REST sera développée en utilisant ASP .NET CORE et comprendra les fonctionnalités suivantes :

- Ajouter, modifier et supprimer un livre et des bonus que nous avons rajoutés 😜
- Rechercher un livre par titre, auteur, etc.
- Gérer les emprunteurs : ajouter, modifier, supprimer et rechercher des emprunteurs.
- Gérer les prêts : créer, supprimer, rechercher des prêts, marquer les retours, etc.

VI. Développement de l'application AvaloniaUI

Configuration d'Avalonia

Méthode `BuildAvaloniaApp`

L'application AvaloniaUI sera développée en utilisant la bibliothèque AvaloniaUI et offrira une interface graphique moderne et réactive pour gérer les informations de la bibliothèque(ps toutes les fonctionnalités ne sont pas opérationnelles). Les fonctionnalités incluront la gestion des livres, des emprunteurs, des prêts, des retours.

La méthode `BuildAvaloniaApp` configure l'application Avalonia en utilisant la classe `AppBuilder`. Elle détecte automatiquement la plateforme, active la journalisation et utilise le framework ReactiveUI.

Mappage Vue-Modèle

Class `ViewLocator`

La classe `ViewLocator` implémente l'interface `IDataTemplate` et est utilisée pour mapper les ViewModels aux Views correspondantes. Lorsqu'un ViewModel est associé à une Vue, la méthode `Build` instancie dynamiquement la Vue correspondante en utilisant la réflexion.

Remarques :

Pour déployer l'application, l'API REST a été déployée sur un serveur compatible ASP .NET CORE et AvaloniaUI ont été installées sur les postes de travail des bibliothécaires.

VII. Conclusion

Notre documentation technique du projet Bibliothèque 3.0 fournit une vue d'ensemble complète de l'application, de l'API REST à l'application de bureau WPF et AvaloniaUI, en passant par la base de données SQLite. Elle décrit les fonctionnalités principales de chaque composant, les technologies utilisées et les instructions pour le développement, le déploiement et les tests.

Bon usage 🙂