**Application Echange de livre**

**Application d’échange de livre**

1. **Couche Models (Dossier Models)**
2. **Classes :**

* 1 ***User*** peut avoir plusieurs ***Books*** (OneToMany)
* 1 ***Book*** appartient à 1 ***User*** (ManyToOne)
* **BookExchange** :
  + On peut échanger un livre plusieurs fois
    - Plusieurs échanges (**BookExchange**) peuvent être effectués pour un **Book**
    - Un **Book** peut subir plusieurs échanges (**BookExchange)**
  + 1 **User** peut faire plusieurs échanges (BookExchange)
  + Un Echange Concerne un User

|  |  |  |
| --- | --- | --- |
| Classe User | Classe Book | Classe BookExchange  *« Classe contenant les données du livre échangé »* |
| Id : int | Id : int | Id : int |
| Name : string | ISBN : string | CreationDate : DateTime |
| Email : string | Title : string | Book : Book |
| Password : string | Author : string | BookId : int |
| TotalPoints : int | Price : double | OldOwner : User  *« L’ancien propriétaire du livre »* |
|  | EditionDate : DateTime | OldOwnerId : int |
|  | State : BookState *« enum »* |  |
|  | *enum* BookState :  {VERY\_GOOD, GOOD, MEDIOCRE, WORN} |  |
|  | Owner : User  *« Propriétaire du livre »* |  |
|  | OwnerId : int *« Champs optionnels »* |  |
|  | PointsValue : int  *Utiliser l’annotation [NotMapped]* |  |

1. **Context**

Créer le context (généré manuellement) + les DbSet (qui me généra la chaine de connexion dans app.config)

Modifier la chaine de connexion en ajoutant le nom de votre serveur et le nom de la base de données

* Activer les migrations (à exécuter qu'une seule fois dans le projet) enable-migrations
* Faire un add-migration nomMigration

*Remarque* : A chaque mise à jour des classes objet - add-migration nomMigration

* update-database

1. **Couche DAO (Dossier Repositories)**

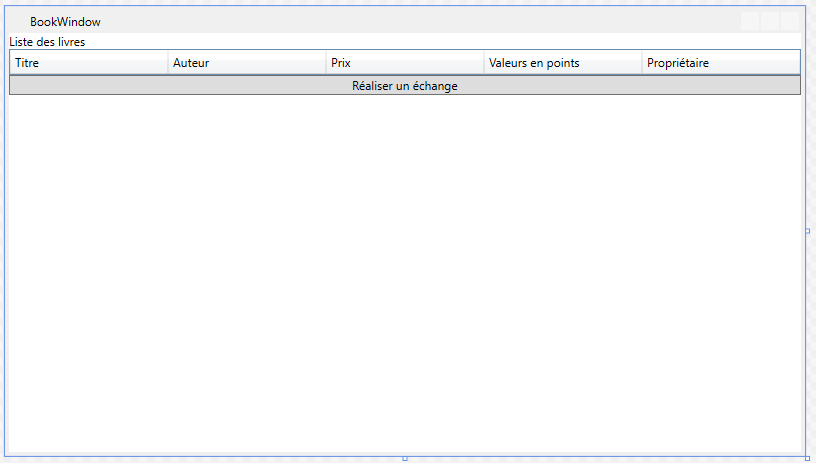
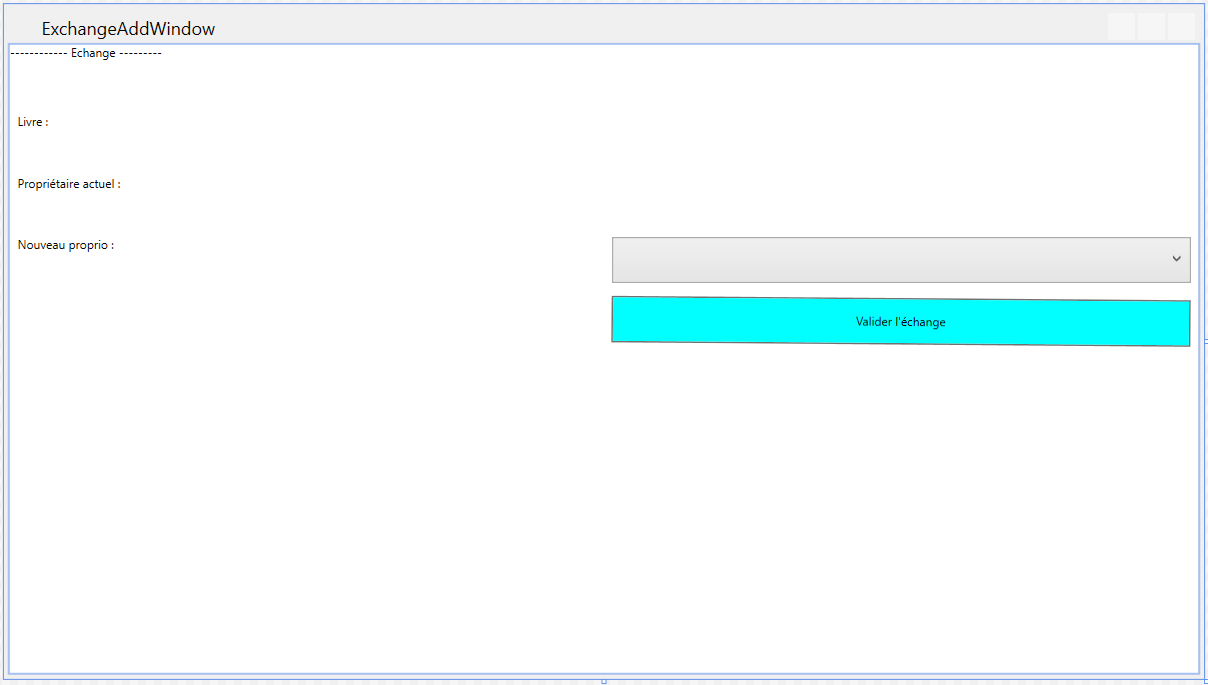
|  |  |
| --- | --- |
| IUserRepository | IBookRepository |
| FindAll()  *« Retourne la liste des Users »* | FindAll()  *« Retourne la liste des livres du User*  *»*  *« On inclut l'entité User (le propriétaire du livre) lors du chargement des books »* |
|  | SaveOrUpdate(BookExchange exc)  *« Permet de mettre à jour ou de sauvegarder le livre échangé »* |

1. **Couche Services (Dossier Services)**

|  |  |
| --- | --- |
| IBookService | IUserService |
| FindAll() | FindAll() |
| ValidateExchange(Book b, int newOwner)  Met à jour le nouveau propriétaire du livre   * b : livre échangé * newOwner : Nouveau propriétaire du livre |  |

1. **Couche Présentation**
2. **Liste de Fenêtre**

|  |  |
| --- | --- |
| **Liste de fenêtre** | **Description** |
| Fenêtre principale : **BookWindow.xaml** | Contient :   * La liste des livres afin de sélectionner le livre à échanger (Utiliser le contrôle *« DataGrid »)* * Un bouton permettant de faire l’échange |
| Fenêtre pour l’échange des livres **ExchangeAddWindow.xaml** | Contient :   * Le livre à échanger * Le propriétaire actuel * Le nouveau propriétaire à sélectionner dans un ComboBox * Un bouton pour valider l’échange |

* **Fenêtre BookWindow**
* **Fenêtre ExchangeAddWindow**

1. **BookWindow.xaml.cs**

|  |  |
| --- | --- |
| Méthodes | Descriptions |
| Window\_Loaded | Appel la méthode **LoadBooks()** pour charger les livres |
| LoadBooks | Appel la méthode **FindAll()** de BookService |
| btnExchange\_Click | On sélectionne un livre pour l’échange :   * Si on a sélectionné un livre on fait l’échange. * On affiche la fenêtre **ExchangeAddWindow** * Sinon on affiche un message « Vous devez sélectionner un livre » |

1. **ExchangeAddWindow**

|  |  |
| --- | --- |
| Méthodes | Descriptions |
| Window\_Loaded | * On récupère la fenêtre appelante afin d’appeler ces méthodes * On récupère le livre sélectionné * On met à jour le champs **TBBook** avec le titre du livre * On met à jour le champ du propriétaire avec le nom du propriétaire du livre * On récupère la liste des users qui n'ont pas ce livre dans un **ComboBox** * L’échange se fera avec le User sélectionné |
| BtnSave\_Click | * On récupère le nouveau propriétaire du livre sélectionné dans le **comboBox** * Appelle la méthode **ValidateExchange** de BookService afin de valider l’échange. Ceci mettra à jour le nouveau propriétaire du livre * On appelle la méthode **LoadBooks** pour mettrela liste des livres   à jours sur son interface graphique   * On ferme la fenêtre |
| Window\_Closing | On ferme la fenêtre |

1. **Erreurs gérées**

|  |  |
| --- | --- |
| Erreurs | Solutions (ex : Voir PDF EntityFramework page 27) |
| System.Data.Entity.Infrastructure.DbUpdateException  SqlException : La conversion d'un type de données datetime2 en type de données datetime a créé une valeur hors limites. L'instruction a été arrêtée.  Observation : Cela implique, comme lu, que mon data table a un type de DateTime2 et ma base de données un type DateTime; C'est faux. | Mettre à jour la base de données avec un DateTime2  Dans l'API Fluent d'Entity Framework 6, on peut mettre à jour toutes les propriétés DateTime à l'aide de cette ligne :  modelBuilder.Properties<DateTime>().Configure(c => c.HasColumnType("datetime2"));   * **Créer un dossier Conventions**   Créer une classe **DateTime2Convention** qui hérite de **Convention**  Dans la méthode DateTime2Convention, ajouter : this.Properties<DateTime>().Configure(c => c.HasColumnType("datetime2"));   * **Utilisation dans le contexte**   Dans la méthode **OnModelCreating**  modelBuilder.Conventions.Add(new DateTime2Convention()); |

* **Faire une migration** 
  + add-migration ModifModelDatime2
  + update-database

