DESCRIPTION ARCHITECTURE SPOTIWISH

Classes :

Dans notre Application nous avons 8 classes principales :

- **Artiste**
  Classe contenant un string **Nom** correspondant au nom de l'artiste.

- **Image**
  Classe contenant un string **Chemin** correspondant au chemin de l'image.

- **Vidéo**
  Classe contenant un string **Lien** correspondant au lien de la vidéo.

- **Titre**
  Classe contenant un **Id** (unique), un string **Nom**, un string **NomAlbum**, une Video **LienVideo**, un Artiste **NomArtiste**, une Image **CheminImage**, un string Bio et un booléen **IsFavoris** , cette classe implémente les interfaces INotifyPropertyChanged et IEquatable**.**

- **CollectionTitre**
  Classe contenant une ReadOnlyObservableCollection **ListeTitre** implémentant l'interface INotifyPropertyChanged et qui appelle OnPropertyChanged à chaque changement.

- **Favoris**
  Classe héritant de **CollectionTitre,** contient les Titres qui seront Favoris.

- **Manager**
  Classe contenant deux ReadOnlyObservableCollection une de Titre **ListeTitre** et une autre de CollectionTitre **ListeCollection,** ainsi qu'une List ListeTitreSupp contenant les Titres qui vont être supprimé, cette classe a aussi un Titre **SelectedTitre** correspondant au Titre sélectionné par l'utilisateur dans le Master, cette classe implémente l'interface INotifyPropertyChanged.

- **Stub**
  Classe permettant une simulation de la persistance qui implémente IPersistanceManager un interface que l'on a créé permettant de faire de la persistance.

Interfaces :

Nous avons implémenter une interface :

- **IPersistanceManager**
  IPersistanceManager nous permet de ‘gérer la Persistance

Converters :

Nous avons implémenté 2 converters :

- **String2ImageConverter**
  String2ImageConverter nous permet de convertir un string (Le Chemin de la Classe Image CheminImage du Titre) en un chemin absolu vers l’Image.
- **String2VideoConverter**
  String2VideoConverter nous permet de convertir un string (Le Lien de la Classe Video LienVideo du Titre) en un chemin absolu vers l’Image.

Persistance :

Nous avons implémenté 2 persistance :

- **Stub**
  Stub est une simulation de la persistance.
- **XMLPersistance**
  XMLPersistance est une persistance qui permet d'enregistrer et de charger les Titres du Manager à partir d’un fichier XML.

UserControls :

Nous avons 4 userControl dans notre Application :

- **UserControl1**
  UserControl1 permet d’afficher les informations d’un Titre dans la ListBox du Master
- **UCDelete**
  UCDelete permet d’afficher les informations d’un Titre dans la ListBox du Master quand l’on est en mode suppression
- **UserControl3**
  UserControl3 permet d’afficher les éléments (TextBlock, TextBox, Image, Valider / Annuler) lors de l’ajout d’un Titre
- **UserControlModif**
  UserControlModif permet d’afficher les éléments (TextBlock, TextBox, Image, Valider / Annuler) lors de la modification d’un Titre

Fenêtres :

Nous avons 3 fenêtre dans notre Application :

- **MainWindow**
  MainWindow est la fenêtre principale de l’Application, elle est affiché lors de son lancement, elle contient 3 boutons Ajouter(+), Supprimer(Poubelle) et

Afficher Favoris(Coeur) en haut à gauche, en dessous de ses trois boutons nous retrouvons le Master de l'Application, c'est une ListBox contenant tous les Titres du Manager affiché sous forme de UserControl, nous pouvons pour chacun de ses Titres soit les Modifier, soit les Ajouter au Favoris (Attention il faut bien cliquer sur le Titre avec de cliquer sur les Boutons).
Sur la partie de droite de la fenêtre nous retrouvons le Titre de l'Application (SpotiWish) ainsi que le Detail de l'Application contenant une Vidéo ainsi que trois Boutons Previous, Play/Pause et Next, permettant de changer le Titre Sélectionné, ou de mettre en lecture ou en pause la vidéo, nous avons aussi l'image du Titre ainsi que ses informations en dessous de l'Image.

- **WindowAdd**
  WindowAdd est la fenêtre qui apparaît lorsque l'on clique sur le premier bouton + (Ajouter) en haut à droite de la MainWindow, cette fenêtre contient un Titre en haut (Ajout Titre) et un UserControl (UserControl3) il nous propose de remplir les informations à propos d'un nouveau Titre, ajouter une Image ou une Vidéo à partir de nos fichiers et de Valider ou Annuler la création du Titre
- **WindowModif**
  WindowModif est la fenêtre qui apparaît lorsque l'on clique sur le Bouton Modifier (Crayon) de UserControl1 (Attention il faut cliquer sur le Titre que l'on veut Modifier avant de cliquer sur le bouton Modifier), cette fenêtre nous propose de Modifier notre Titre en écrivant dans des TextBlock qui sont déjà remplie avec les valeurs du Titre Sélectionné.

Collection avancées :

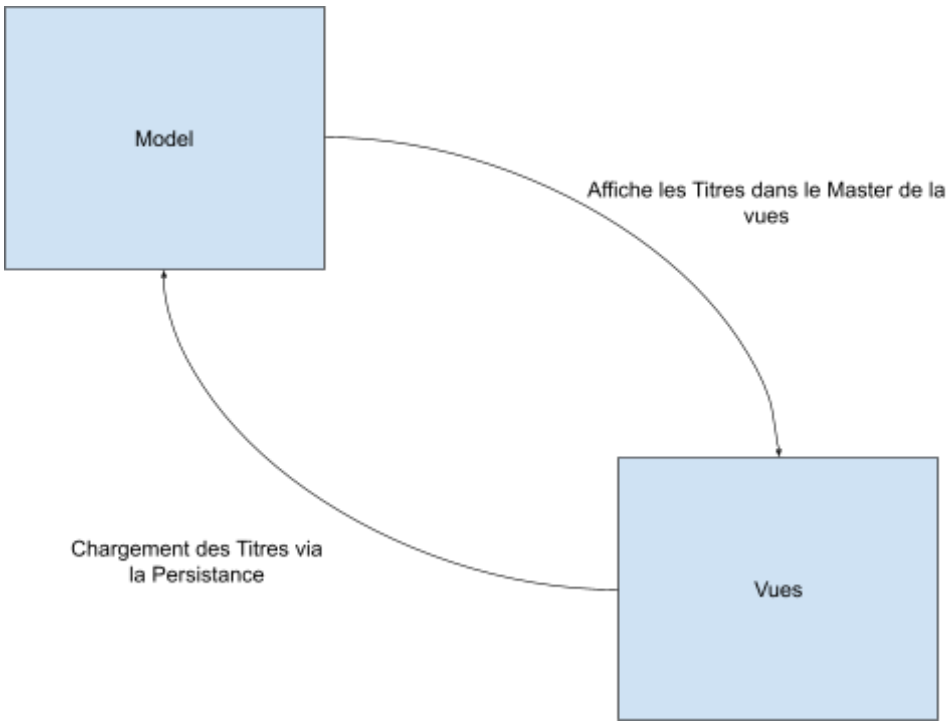
Nous n'avons pas vu l'utilité de faire une collection avancées.

Dépendances :

- Tous les paquets de notre Application dépendent de ClassOfSpotiWish, en effet ClassOfSpotiWish contient les classes de l'Application (Manager, Titre, …) sans ses classes notre Application ne pourrait pas fonctionner.
- Le dernier paquet Test contenant les Tests Unitaires et les Tests Consoles dépend de StubLib, en effet lors de nos Tests Unitaires de Manager nous utilisons le Constructeur de Manager avec un Stub.

Patrons de conceptions

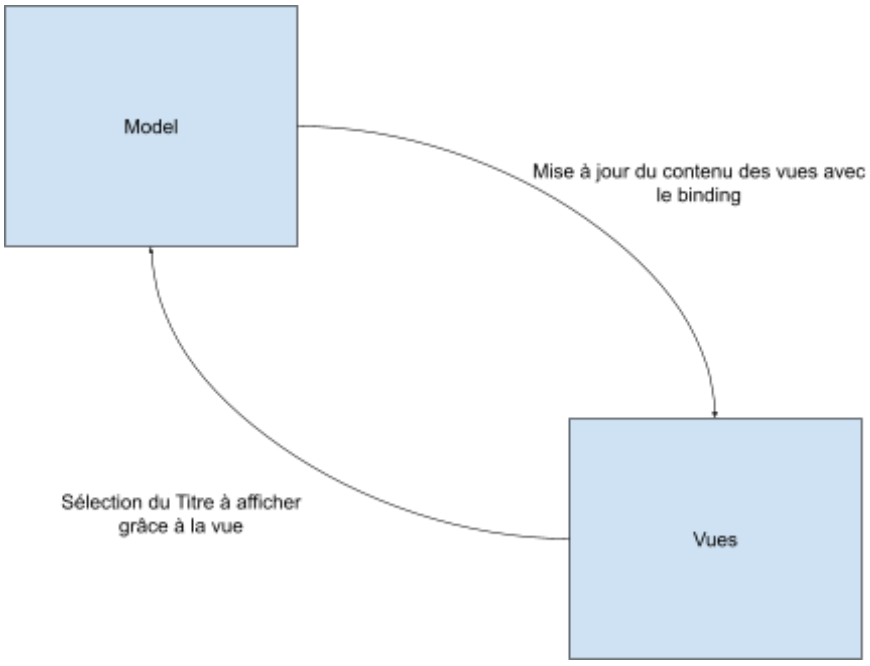
Patron de conception mettant en avant le chargement du Master



Lors du chargement de l'application, MainWindow appelle la méthode Window_Loaded qui charge les Titres dans le Manager, via la méthode LoadTitres du Manager.

Le Model quant à lui renvoie la ListeTitre du Manager remplie avec la persistance à la vue qui l'affiche dans une ListBox ayant comme ItemsSource ListeTitre.

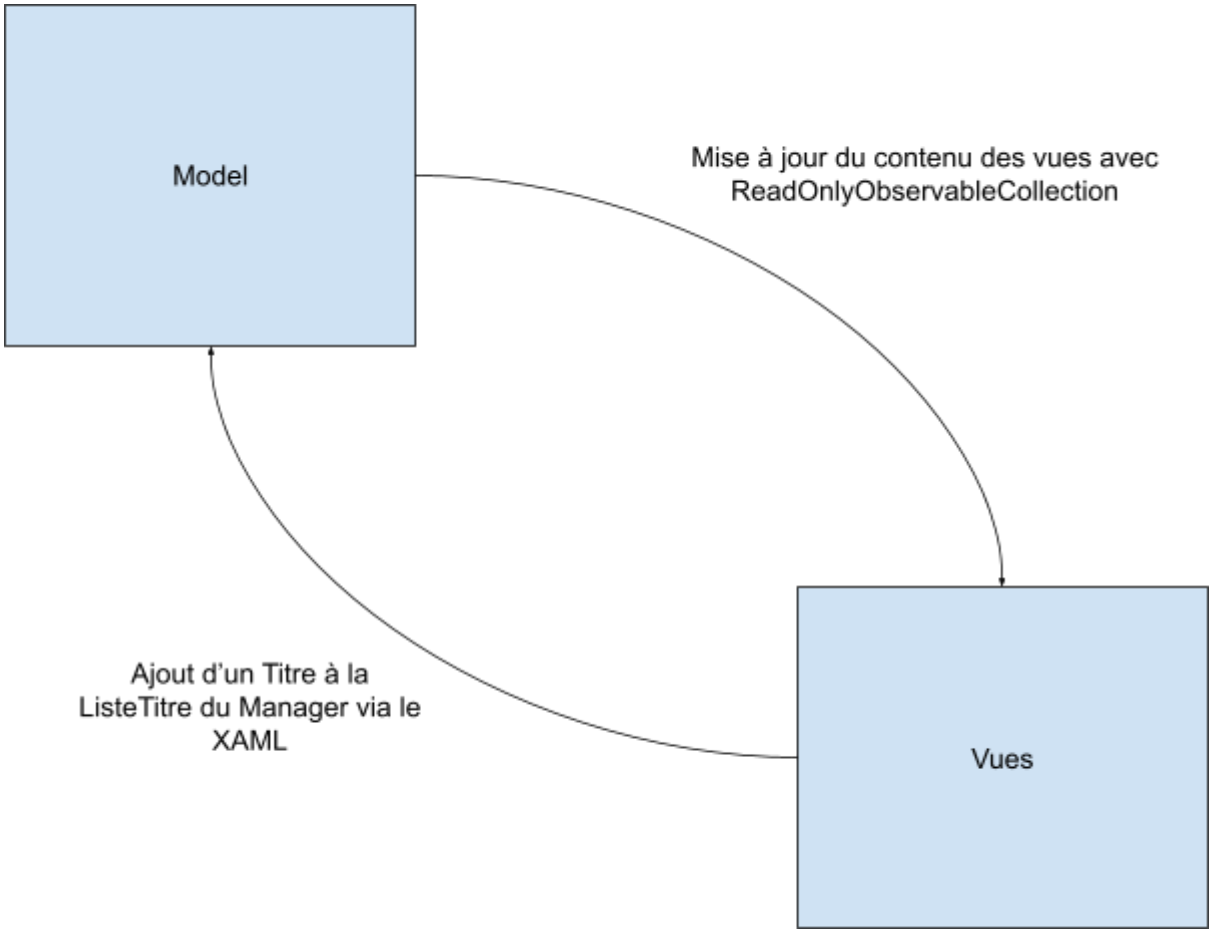
Patron de conception mettant en avant le binding



Nous avons décidé d'ajouter une variable SelectedTitre de Type Titre à notre Manager, celle-ci permettra d'afficher dans le détail de la vue les informations du titre que l'on a sélectionné.

Dans la partie Details de la vue nous avons bind les Informations à SelectedTitre, le Titre sélectionné par l'utilisateur, il y a plusieur moyen de changer SelectedTitre, soit en utilisant les Boutons Previous et Next en dessous de la Vidéo, qui change SelectedTitre au titre suivant/ précédent dans la ListeTitre, ou en cliquant avec la souris dans la ListeBox sur le Titre que l'on veut affiché.

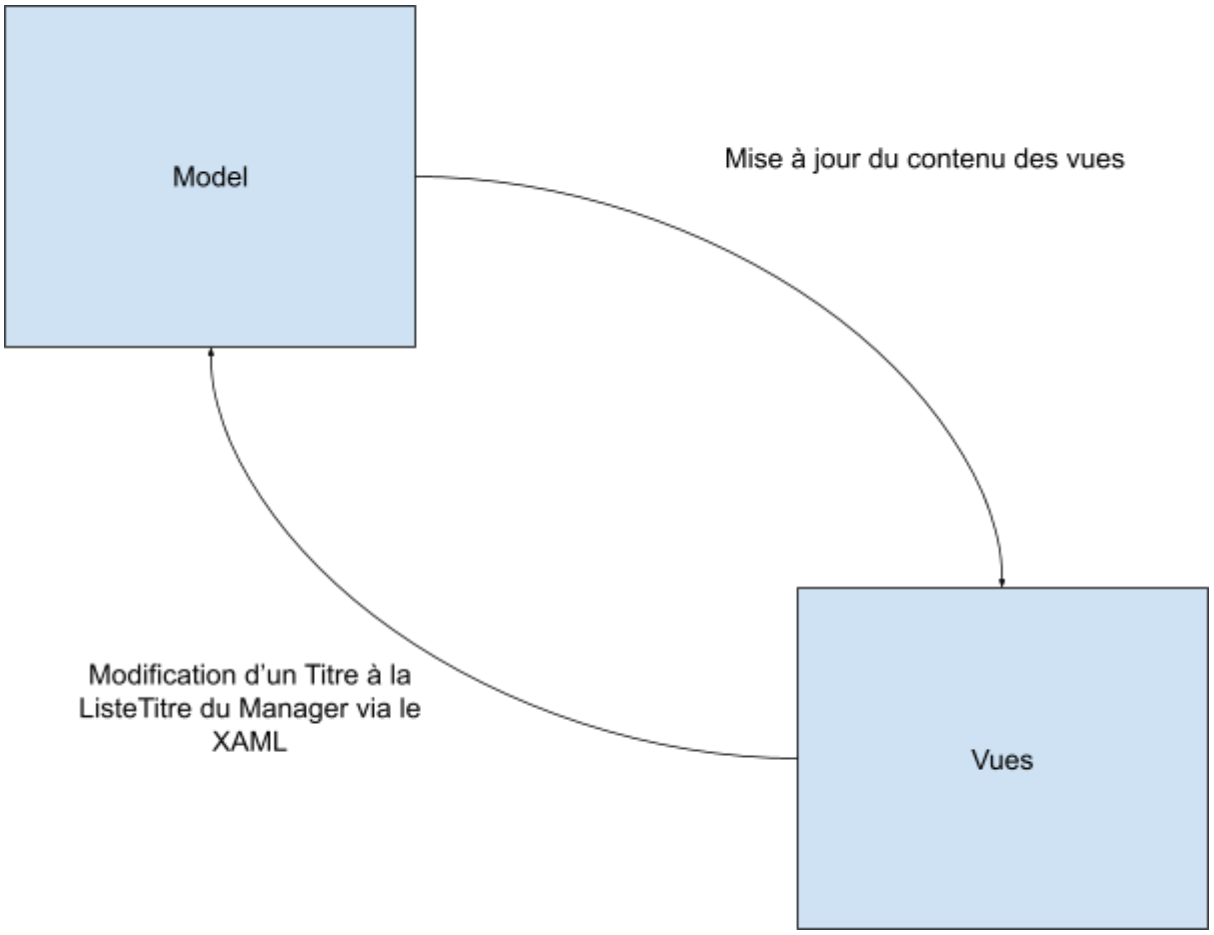
Patron de conception mettant en avant l'ajout d'un Titre



Nous avons choisi de faire des Listes de Titres en ReadOnlyObservableCollection car ses collections implémentent INotifyPropertyChanged et OnPropertyChanged, cela nous permet d'actualiser la vues lors de nos ajouts ou suppressions de Titre du Manager, pour garder le Master à jour et ainsi avoir un Application réactive.

Pour ce qui est des Vues nous avons décidé d'implémenter un Bouton Ajouter sous forme de "+" en haut à gauche de l'application. Lors du clique de ce bouton, une fenêtre apparaît demandant le Nom du Titre, le Nom de l'album, le nom de l'Artiste du Titre, le Lien de la vidéo mp4 installé sur le l'ordinateur (Un bouton pour parcourir les dossier de l'utilisateur est disponible), la Biographie de la Vidéo, l'Id de la vidéo (Celui-ci est unique) et finalement le chemin vers l'image du Titre (Un bouton permettant de parcourir les fichiers de l'Utilisateur est aussi disponible).

Patron de conception mettant en avant l'ajout d'un Titre



Nous avons décidé d’implémenter INotifyPropertyChanged à notre classe Titre, ainsi toutes les propriétées modifiables de la classe Titre ont OnPropertyChanged dans leurs setter, cela nous permet de changer les propriété du Titre et à la vue de se mettre à jour automatiquement sans que l’on soit obligé de la recharger.