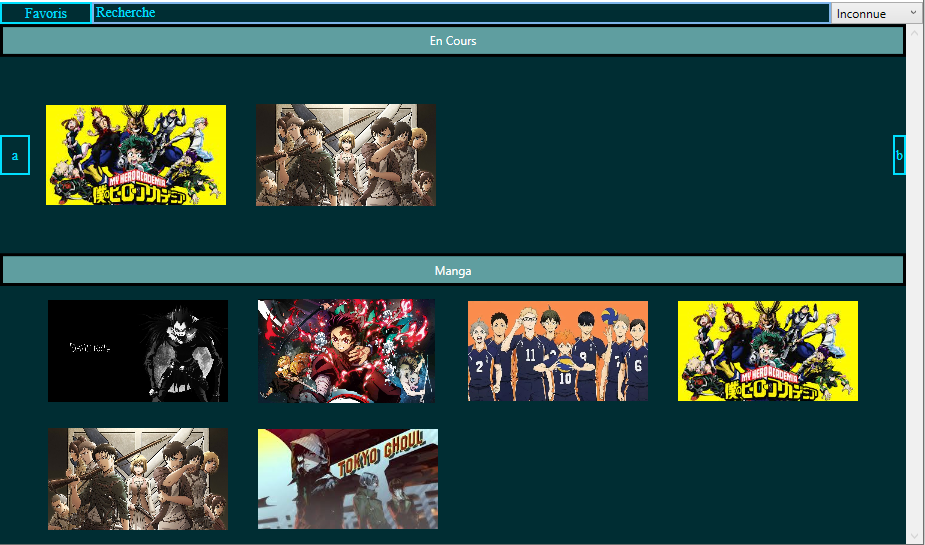
Documentations projet

« Watch Anime »

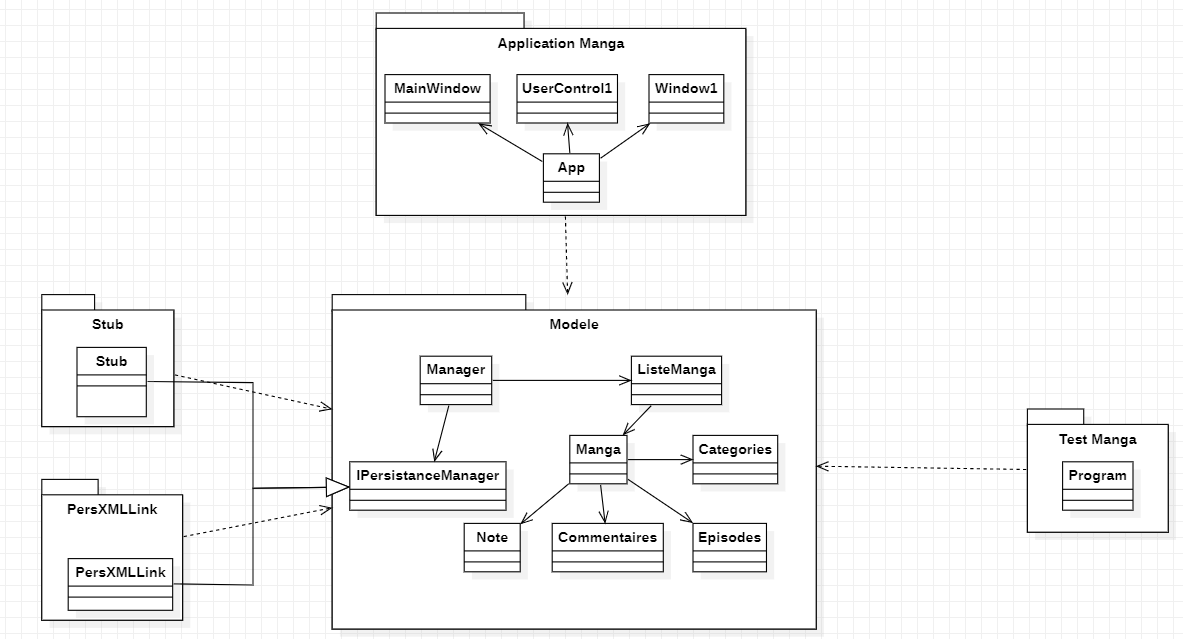


Sommaire

1. Diagramme de paquetage
2. Diagramme de classes
3. Diagramme de séquence
4. Description de l’architecture

Ruy Leo, Pimentel Corentin

I)Diagramme de paquetage



Le package Modèle regroupe la plus grande partie du code, la classe Manager gère la plupart des autres classes principales, c’est elle qui gère la navigation à travers l’application en appelant les différentes méthodes.

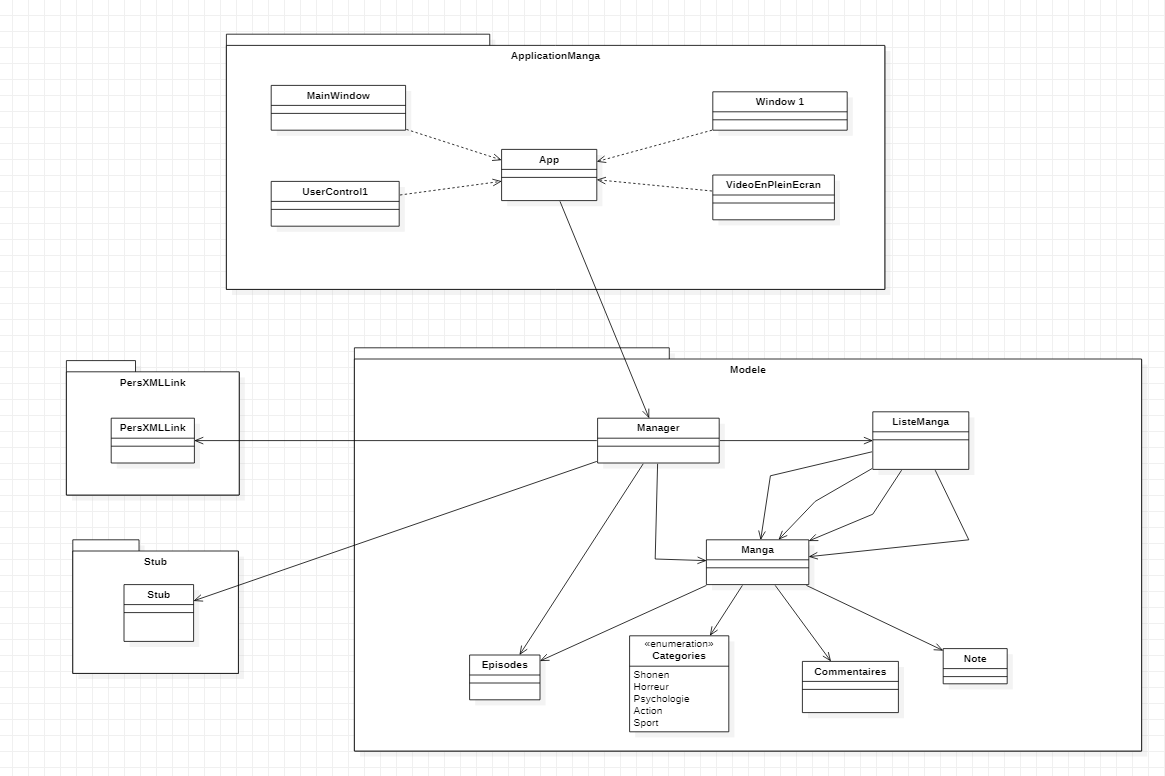
Le package Application Manga regroupe toutes la partie XML, c’est la classe App qui nous permet de faire le lien entre le code et le XML.

Le package Test Manga permet de faire les différents tests fonctionnels du code à partir de la classe Manager.

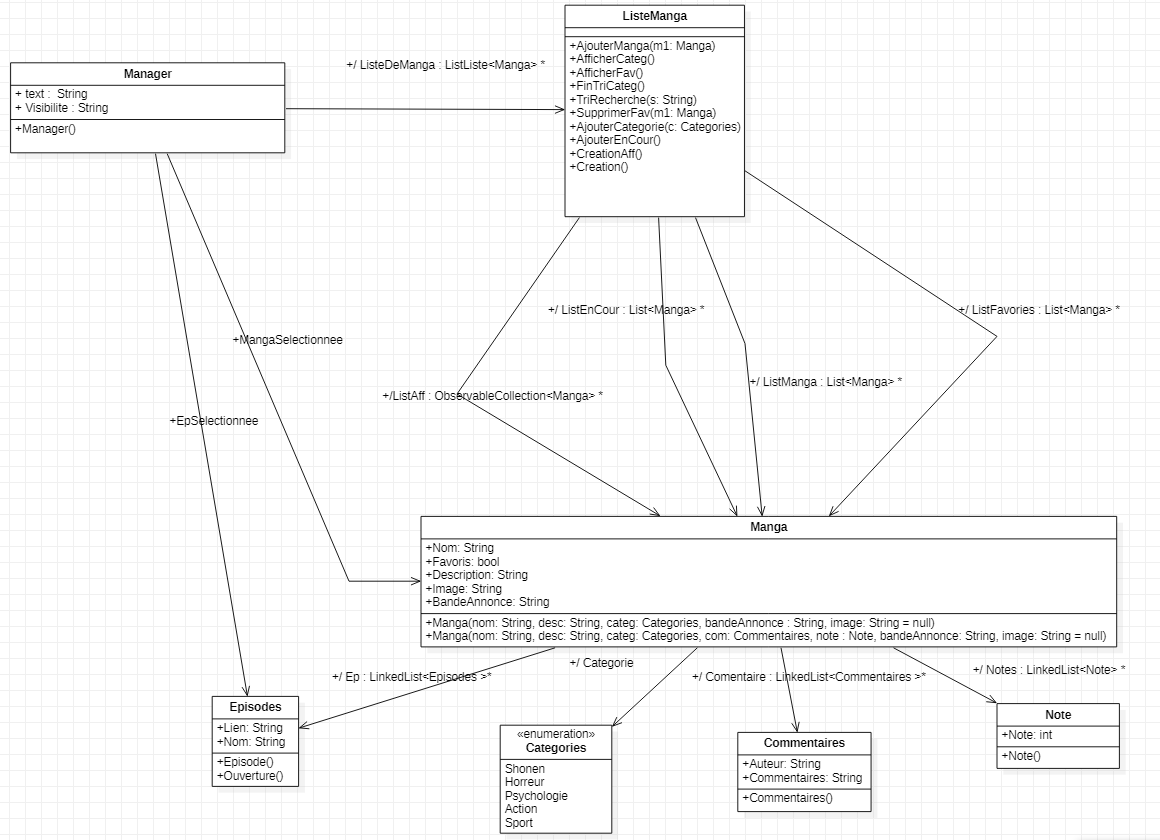
Les package Stub et PersXMLLink serve pour le chargement des données et PersXMLLink sert aussi pour la sauvegarde.

II)Diagramme de classes

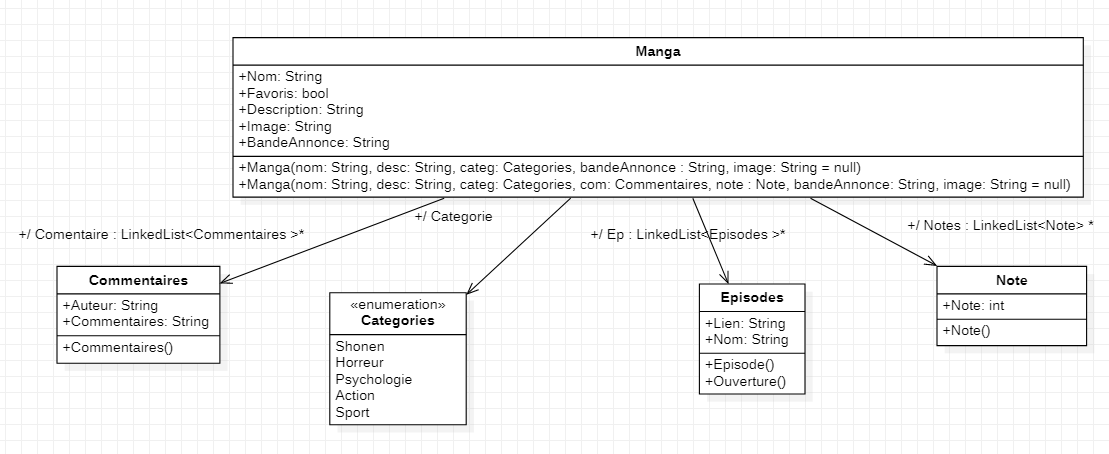
Diagramme complet simplifié :



Les classes de Modèles :



Manga :



Un Manga sera composé, d’une liste de Commentaires avec le texte du commentaire et son auteur, d’une Catégorie qui correspondra à un énum , d’une liste d’Episode avec le nom de l’épisode et le lien menant à cet épisode, et d’une liste de Note qui sera que des entiers.

ListeManga :

Une image contenant texte

Description générée automatiquement

La classe ListManga contient une liste de manga qui contiendra tous les mangas de l’application, une liste ListFavories et contient tous les mangas qui ont été cochés en favoris, ListEncour qui contiendra la liste des mangas qui ont au moins un épisode de regardé et listAff qui sera la liste qui sera tout le temps modifiée par les différentes méthodes de listManga.

CreationAff() qui copie les mangas de ListManga à listAff après avoir entièrement vidé listAff.

CreationFav() qui copie les mangas de ListFavories à listAff après avoir entièrement vidé listAff.

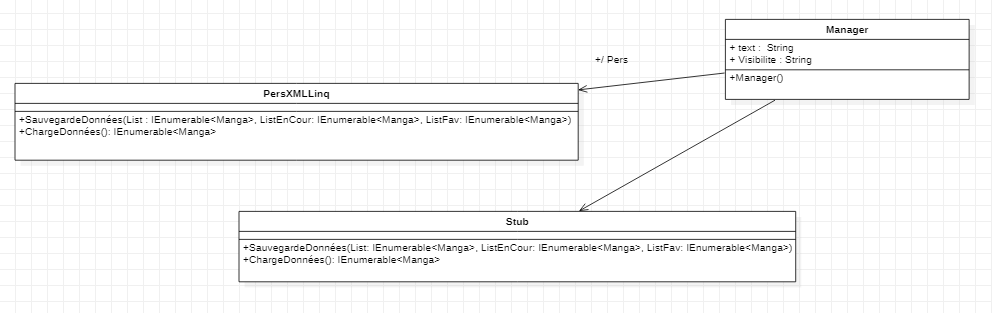
AjouterCategorieFav() qui copie les mangas de ListFavories à listAff après avoir entièrement vidé listAff.

AjouterFav() et SupprimerFav() servent respectivement à ajouter un favorie a la liste des favoris ou a en retirer un.

AfficherFav() sert à afficher les favoris lors des tests

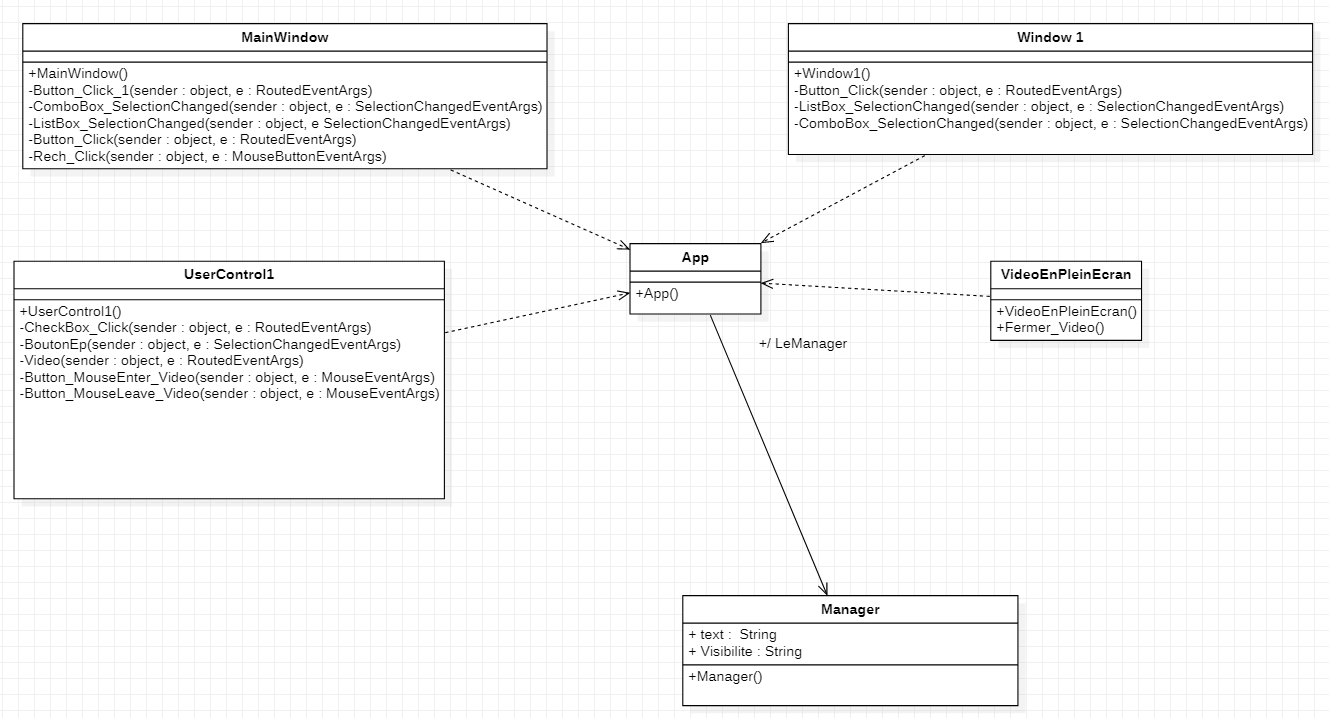
TriRecherche() est une méthode qui compare les nom de manga a un texte entré par l’utilisateur

Persistance :



Sur cette impression c’est la partie d’où viennent les données de l’application, ChargeDonnées() est la même fonction pour les deux classes PersXMLLing et Stub, elle permet de charger les données à une différence près que le stub sera utilisé que lors du premier lancement de l’application, ensuite PersXMLLing sauvegardera les données grâce à SauvegardeDonnées() et les rechargera lors de la prochaine utilisation avec son ChargeDonnées(). Le SauvegardeDonnées() de Stub ne saura pas utiliser puisque les données de stub seront écrit en « dur ».

Vue :



La partie vue, est constituée de la page principale qui est MainWindow c’est ici que listAff sera affiché,

Button\_Click\_1() sert à ouvrir proprement la page où tous nos favoris seront affichés

ComboBox\_SelectionChanged() nous permet de trier la listAff.

ListBox\_SelectionChanged() Affiche le UserControl1.

Button\_Click() servait pour des tests.

Rech\_Click() netoie le texte de la barre de recherche quand on clique dessus.

Il y a aussi le UserControl1 qui s’affiche sur le MainWindow avec les informations de mangaSelectionnee,

CheckBox\_Click() sert a ajouté le manga a notre liste de favoris.

BoutonEp() sert à ouvrir un épisode pour le visionner.

Video() qui permettra de faire apparaitre la bande annonce en plus gros.

Button\_MouseEnter\_Video() permet de lancer la bande annonce quand on passe sur l’image du manga

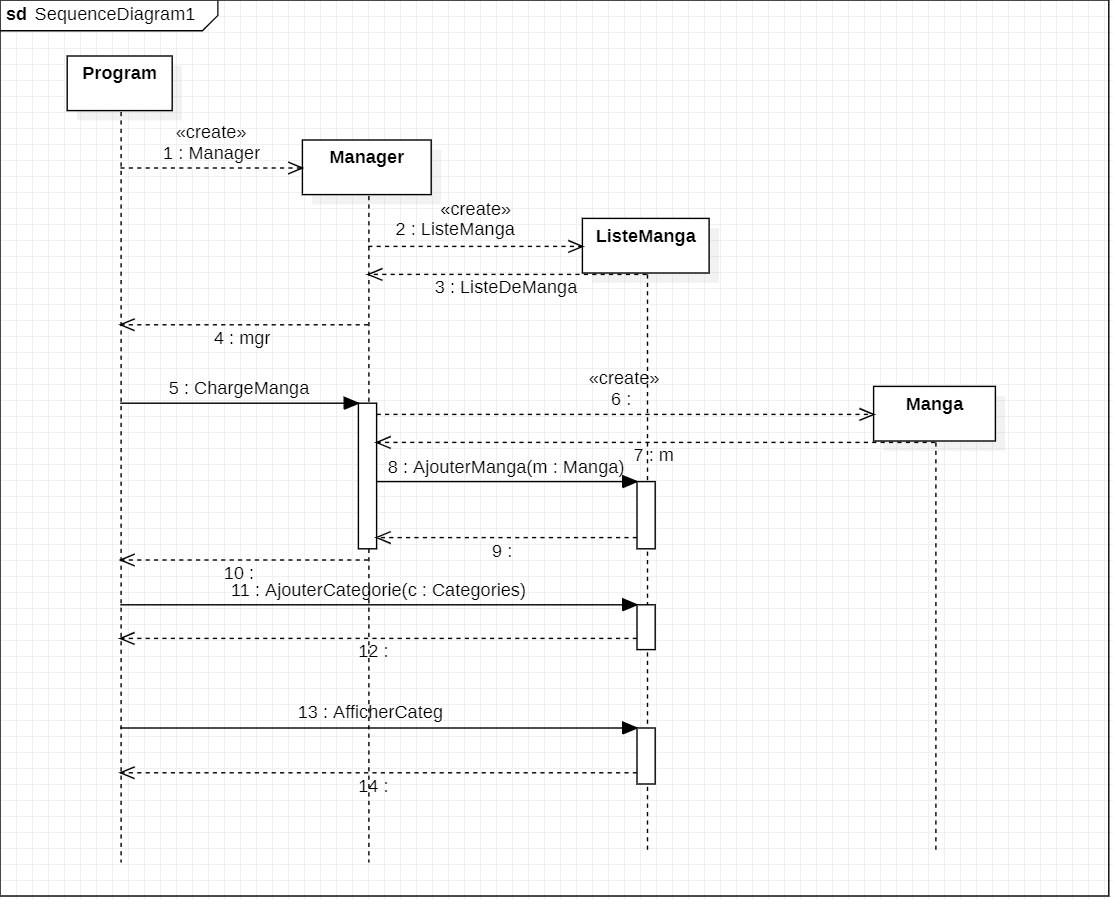
Button\_MouseLeave\_Video() permet d’arrêter la bande annonce quand on sort de l’image du manga

Sur Window1 il y a juste l’affiche des manga favoris avec la possibilité de les triés par catégorie

VideoEnPleinEcran est le usercontrol où sera afficher la bande annonce en gros.

Pour finir, le Manager qui est présent sur plusieurs impressions va contrôler tout ce qui communique du modèle à la vue, comme l’affichage de la ListManga, le Manager communique avec App qui lui-même communique avec les vues. Le Manager est raccordé au Stub et PersXMLLink qui seront respectivement le chargement de donné pour la première utilisation et la sauvegarde et le chargement des données pour les fois suivantes. ListManga est donc instancié dans Manager pour remplir la vue.

III)Diagramme de séquence



Ce diagramme de séquence représente le test fonctionnel qui nous permet d’afficher les Mangas contenu dans une certaines catégories défini dans la méthode AjouterCatégories.

Program est le fichier de test, on crée un manager avec une liste de manga en le remplissant par la suite de manga m crée juste avant, puis on tri la listAff grâce à la méthode AjouterCategorie qui aura comme argument c qui est la catégorie voulu dans le test puis on affiche cette liste dans la console.

IV)Description de l’architecture

Le paquetage Modèle regroupe la plus grande partie du code c’est elle qui est au centre l’application, elle permet la navigation à travers l’application en appelant les différentes méthodes.

Le paquetage Application Manga regroupe toute la partie XML, c’est la classe App qui nous permet de faire le lien entre le code et le XML. Elle dépend du paquetage Modèle.

Le paquetage Test Manga permet de faire les différents tests fonctionnels du code à partir de la classe Manager contenus dans le paquetage Modèle.

Le paquetage Stub permet de charger les données lors de la première utilisation, le paquetage PersXMLLinq prendra le relai pour les prochaines fois puisqu’il aura sauvegardé les données dans un fichier par rapport à Stub qui aura ses données directement dans le code.

Pour les paquetages, nous avons décidé de séparer le modèle de l'application pour rendre la réutilisation plus simple. En effet, si l'on souhaite utiliser uniquement le modèle dans une autre application cela nous permet de le réutiliser assez facilement.

On a aussi choisi séparer la persistance afin de la rendre interchangeable car il existe plusieurs façons d'effectuer la persistance et une solution peut être adaptée un jour puis ne plus l'être demain, c'est pourquoi de cette manière, on pourrait faire diffèrent paquetage de persistance et garder pour le client uniquement celle nécessaire à l'application.

Le test est aussi séparé pour que lorsque l'on transmet l'application à un client on puisse le supprimer afin que l’application soit moins lourde étant donné que le test n'est pas nécessaire au bon fonctionnement de l'application.

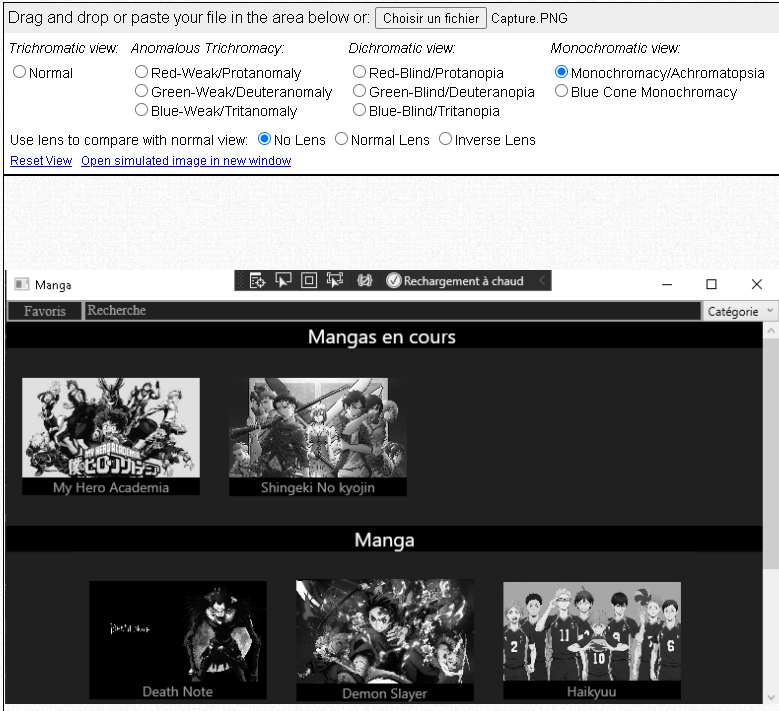
Nous avons décidé d’utiliser majoritairement des List car nous avons estimé que nous n’avions pas besoin de Collection plus complexe. De plus, les List contiennent une méthode par défaut qui permet à nos mangas d’être trié dans l’ordre alphabétique.

Nous avons aussi une Observable collection qui permet d’envoyer des notifications à chaque modification de la liste afin qu’elle se mette à jour dans l’affichage.

Nous avons utilisé des LinkedList car elles permettent d’ajouter des éléments plus facilement qu’une List simple.

V)Accessibilités

En matière d’accessibilité pour les daltoniens, le filtre qui rend la vision de l’application la plus désagréable est :



Mais ça reste tout de même plutôt visible grâce au bon contraste de l’écriture, ceci est dû principalement au choix de couleurs plutôt sombre ce qui apporte aussi un confort des yeux puisque les couleurs ne sont pas trop agressives, la police est aussi plutôt épurée pour faciliter la lecture.

Les éléments sont alignés dans la rubrique ou la recherche sera nécessaire (Manga)

Il n’y a effectivement pas besoins de beaucoup de clic (moins de 3) pour accéder une fiche de manga et même les épisodes. D’ailleurs pour faciliter le visionnage des prochains épisodes d’un manga il faudra juste cliquer sur les mangas en cours.