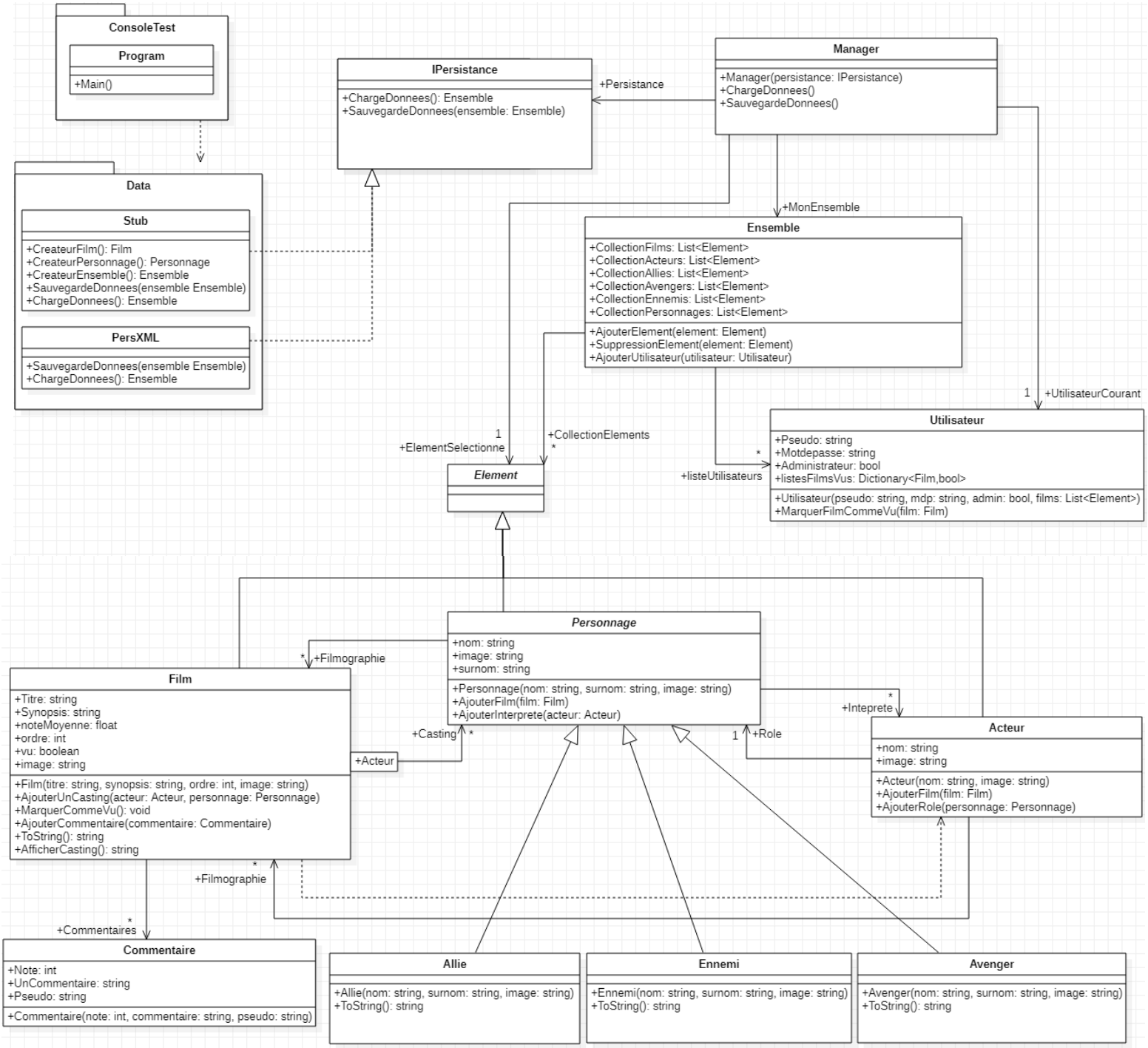
Conception et Programmation Orientées Objets (C#, .NET)
Documentation

Diagramme de classes



Dans la partie basse du diagramme, on a les classes qui désignent les objets qui seront affichés par l'application :

- Element : classe abstraite mère des trois principaux objets (Film, Personnage et Acteur)
- Film : classe qui représente un film. Elle dépend des deux autres principaux objets via sa collection Casting (un dictionnaire qui associe une valeur Personnage à une clé Acteur). Elle a aussi pour attribut une collection de Commentaires.
- Personnage : classe abstraite qui représente un personnage, classe mère des trois factions de personnages : Allie, Avenger et Ennemi. Elle dépend des deux autres principaux objets via une collection de Film (Filmographie, simple liste) et une collection d'Acteur (Interprete, simple liste).
- Acteur : classe qui représente un acteur. Elle dépend des deux autres principaux objets via une collection de Film (Filmographie, simple liste) et un attribut Personnage (Role).

Tous ces éléments sont stockés dans une classe Ensemble. Cette classe contient une collection d'Element, qui stocke tous les objets affichés par l'application, ainsi que de nombreuse propriétés calculées pour former des collections d'éléments triés par type (CollectionActeurs, CollectionAllies, etc.). Cette classe stocke aussi la liste d'utilisateurs.

Le modèle est ensuite relié aux vues via le Manager. Cette classe Manager a comme propriété un Ensemble MonEnsemble, où sont stockées toutes les données nécessaires au fonctionnement de l'application, un Element ElementSelectionne, qui désigne l'élément (Film, Acteur ou Personnage) le plus récemment sélectionné par l'utilisateur, et un Utilisateur UtilisateurCourant qui permet de mémoriser l'utilisateur actuellement connecté (si personne n'est connecté, cette propriété est nulle).

Le Manager a aussi une propriété Persistance, une instance de l'interface IPersistance entrée comme paramètre dans son constructeur. Ses deux autres méthodes permettent de charger puis de sauvegarder les données.

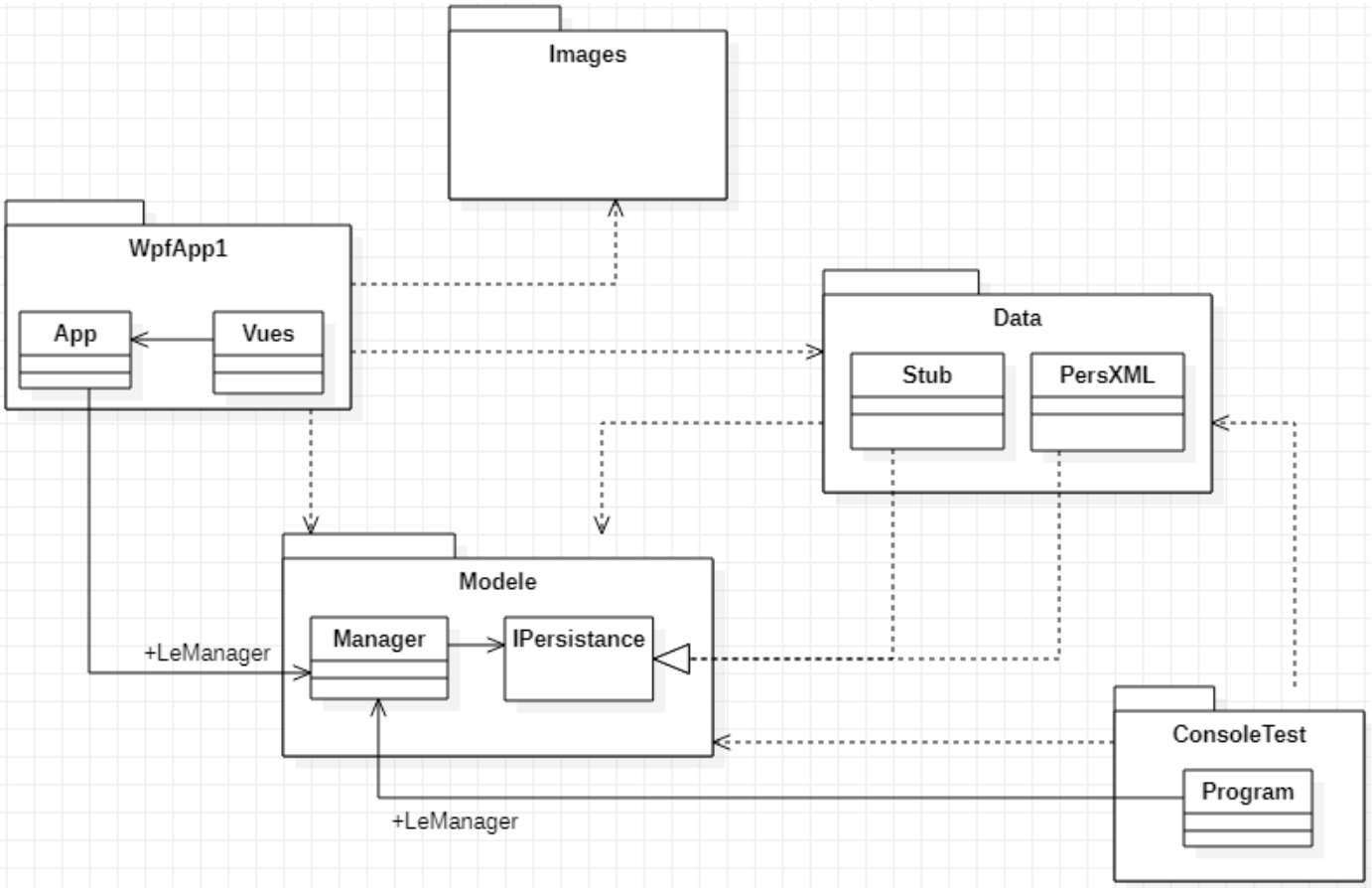
L'interface IPersistance a une méthode ChargeDonnees qui renvoie un Ensemble contenant les données de l'application. Son autre méthode, SauvegardeDonnees, prend en paramètre un Ensemble regroupant toutes les données et le sauvegarde.

Les classes Stub et PersXML du paquet Data implémentent cette interface et décrivent la manière dont sont chargées et sauvegardées les données.

Le paquet ConsoleTest vient utiliser ce Stub afin de charger des données directement dans un Ensemble pour pouvoir tester les fonctionnalités de l'application en dehors de vues.

Dans ce diagramme ne sont pas précisés les méthodes relevant des protocoles d'égalité, les constructeurs par défaut et certaines redéfinitions de la méthode ToString.

Diagramme de paquetage



L'application est composée de 5 projets représentés par des paquetages.

Le paquetage Modèle est une bibliothèque qui contient les différentes classes permettant de gérer les données. On y trouve le Manager qui est utilisé par le projet graphique (paquetage WpfApp1) pour afficher ces données.

Le projet graphique a donc une dépendance vers ce modèle, ainsi qu'envers le paquetage Images, une simple bibliothèque d'images. WpfApp1 dépend également du paquetage Data puisqu'il va utiliser une de ces classes afin de charger les données lors de l'instanciation du Manager dans sa classe App. Ce Manager sera ensuite réutilisé par les autres vues.

Le paquetage Data dépend du Modèle puisqu'il en utilise les classes et implémente son interface IPersistance.

Le paquet ConsoleTest, qui contient des tests des fonctionnalités de l'application, dépend de Modèle pour les classes et méthodes qu'il utilise (pour instancier le Manager et effectuer les tests) et de Data pour le chargement des données de test.