Universite Clermont Auvergne

SAE 2.01

Création d’une application

Anthony RICHARD | Jade VAN_BRABANDT
06/06/2023

Anthony RICHARD
Jade VAN_BRABANDT

IUT Clermont-Auvergne
1ère année
BUT Informatique

Table des matières

DESCRIPTION DES PROJETS ET DES CLASSES

L'application est composée de 6 projets. Le patron de conception est une façade. Dans ce patron de conception, une interface simplifiée permet de dissimuler la complexité du l'application pour la rendre plus agréable à utiliser. Nous avons dans ce projet un nombre important de classes et de projets liés entre eux par beaucoup de dépendances. Heureusement pour nous, l'interface du .NET MAUI permet de faire du multiplateforme sans changer de code, rendant ainsi le développement d'une application plus rapide et plus simple. Les spécificités des différentes plateformes sont gérées par le Framework, ce qui rend les choses plus simples vu que l'on doit seulement se concentrer sur la logique métier. De plus, l'utilisation de ce patron permet de proposer une interface graphique (grâce au XAML) simple à utiliser pour l'utilisateur, et simple à implémenter sur toutes les plateformes à la fois pour le développeur.

Projet Stim

Le projet Stim est la partie MAUI de l'application. C'est dans cette partie que nous avons les vues qui permettent d'avoir une interface graphique (.xaml), ainsi que les fonctions (fichiers .xaml.cs) qui permettent de naviguer et d'interagir avec éléments graphiques (les boutons par exemple).

Projet Model

Le projet Model contient les classes de nos objets. Nous avons des objets de type Game pour un jeu, User pour un utilisateur et Review pour un commentaire sur un jeu. Chaque classe contient des attributs, des propriétés et des méthodes. Également, nous avons le projet Test, qui contient les tests unitaires de l'application. Chaque script est lié à une classe dont il teste un maximum de méthodes, pour garantir une couverture maximale du code.

Classe Game

Cette classe représente un jeu, qui est caractérisé par un nom, une description, une année de sortie, une image, des étiquettes, une note moyenne, un lien vers une boutique en ligne et une liste de commentaires. Elle contient également un constructeur ainsi que des méthodes permettant de modifier ses attributs et des méthodes permettant d'ajouter ou de supprimer un commentaire sur le jeu.

Classe User

Cette classe représente un utilisateur, qui est caractérisé par un nom d'utilisateur, une biographie, une adresse mail, un mot de passe, une photo et une liste de jeu favoris. Elle contient également des méthodes permettant de s'identifier lors de l'ajout d'un commentaire sur un jeu et de supprimer un de ses propres commentaires.

Classe Review

Cette classe représente un commentaire laissé sur un jeu, qui est caractérisé par une référence à son auteur, une note et un message. Les méthodes de cette classe permettent de modifier ses attributs.

Classe Manager

La classe Manager contient les listes d'utilisateurs et de jeux qui sont utilisés dans l'application. Elle contient également une référence au jeu sélectionné par l'utilisateur et une référence à l'utilisateur qui s'est authentifié. Enfin, elle contient une persistance, qui permet de charger et sauvegarder les données de l'application. Ses méthodes permettent d'ajouter ou de retirer des utilisateurs et des jeux des listes, ainsi que de filtrer des jeux avec une barre de recherche.

Interface IPersistance

Cette interface permet de faire le lien entre les différentes méthodes de sauvegarde et de chargement utilisés dans l'application (persistance et stub). Ces méthodes de sauvegarde et de chargement sont toutes abstraites car elles sont redéfinies dans les classes héritées. L'héritage permet ici une plus grande simplicité, en permettant par exemple de rapidement changer le mode de persistance de l'application (stub ou vraie persistance).

Projet AppConsole

Le projet AppConsole est une application console qui nous permet d'effectuer des tests fonctionnels sans passer par une interface graphique. On peut ainsi rapidement tester l'efficacité du code, de l'implémentation des classes et des méthodes.

Projet Stub

Le projet Stub est un projet contenant une unique classe, qui nous a permis de tester notre binding, en nous permettant de remplir notre liste master avec des données écrites en dur (directement dans le code C#). Il nous a également permis d'initialiser notre fichier de sauvegarde avec les données du Stub. Ce projet a une unique classe Stub héritée de l'interface IPersistance et qui contient les méthodes de sauvegarde et de chargement.

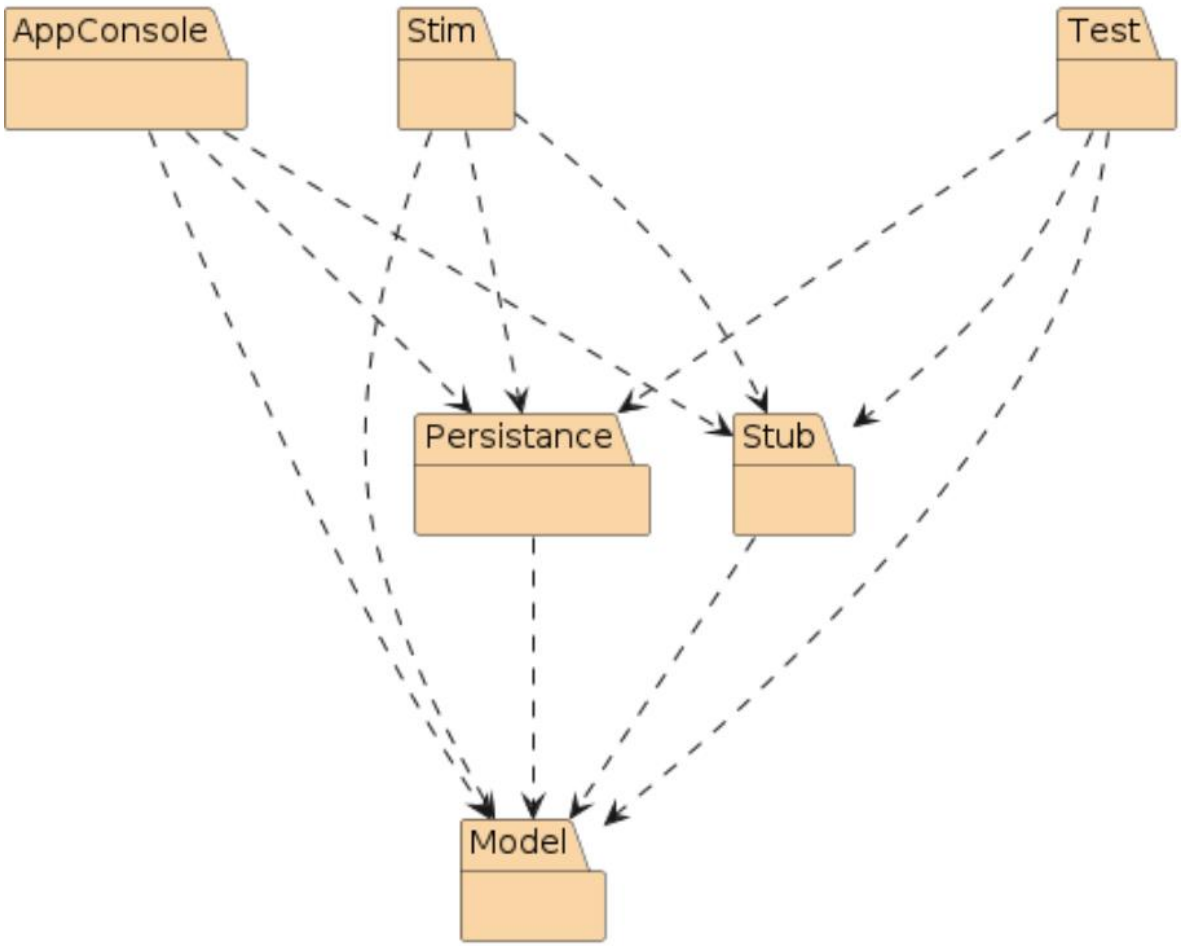
Projet Persistance

Le projet Persistance, c'est le projet qui permet de gérer la sauvegarde de nos données dans un fichier xml. Ce projet contient des fonctions de sauvegarde et de chargement de jeux et d'utilisateurs. Tout comme le stub, il est composé d'une seule classe Persistance, héritée de l'interface IPersistance, contenant les méthodes de sauvegarde et de chargement.

Projet Test

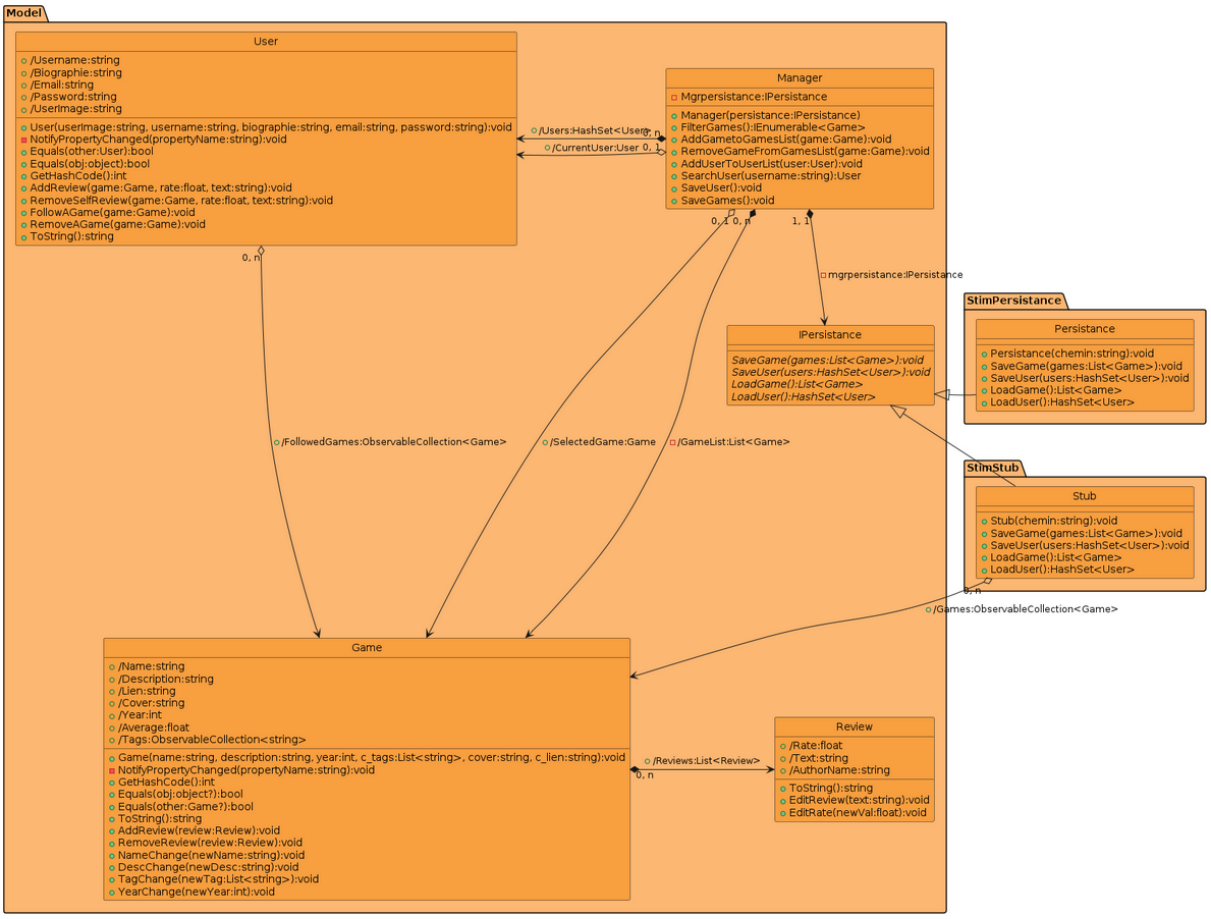
Le projet Test est un projet contenant les tests unitaires de l'application. Pour garantir une qualité optimale, nous nous sommes fixés pour objectif d'avoir un minimum de 80% de notre code (seulement les modèles et le manager) couvert par des tests. Pour chaque classe de notre code, nous avons une classe de test associée.

DIAGRAMME DE PAQUETAGE



Sur ce diagramme de paquetage, on voit les dépendances qui existent entre nos différents projets. Les blocs représentent les projets, et les flèches les dépendances. Comme on peut le remarquer, le projet Model, contenant les classes d'objets, est un projet central car tous les autres projets en dépendant. Cela s'explique par le fait que nous créons des objets dans tous les projets.

DIAGRAMME DE CLASSE



Sur ce diagramme de classe, on voit le contenu de chaque classe et de chaque namespace du projet, ainsi que les liens entre les différentes classes. Les rectangles extérieurs (couleur plus claire) représentent les namespaces, et contiennent une ou plusieurs classes, représentées par les rectangles intérieurs (couleur plus foncée). Dans les classes, les attributs et les propriétés sont dans le deuxième rectangle (sous le nom de la classe) et les méthodes sont dans le dernier carré. Le symbole qui précède les noms indique si la méthode ou l'attribut est publique (rond vert), privé (carré rouge), et l'italique signifie que la méthode est abstraite. Quant aux flèches, elles montrent les liens d'héritage, de composition et d'agrégation entre les classes. L'héritage est symbolisé par une flèche dont la pointe est vide, la composition par une flèche dont le bout est rempli et l'agrégation est symbolisée par une flèche dont le bout est vide. On voit également les cardinalités, 0, 1 ou n (n voulant dire que le nombre d'objet peut être infini).