Chevassus Noé

Bonhomme Paul

G5

Projet GameLib

Description écrite de l'architecture

Nous allons décrire notre application en plusieurs étapes, nous allons tout d'abord parler de son architecture puis des patrons de conceptions et enfin des différentes dépendances entre les projets de l'application.

Pour commencer, notre application se divise en quatre gros projets distincts a qui s'ajoutes des projets de tests ainsi qu'un projet de déploiement de l'application.

Les gros projets sont respectivement « GameLib_Projet » qui contient tous les éléments visuels. En effet, le projet contient différents user-control qui sont répartis dans l'espace de notre fenêtre principale. Il contient aussi une classe Navigator qui permet une navigation entre nos différents user-control afin de fluidifier la navigation. Nous avons aussi ajouté des convertisseurs pour la visibilité de certains éléments qui dépendent du statut de connexion de l'utilisateur de l'application.

Il y a ensuite le projet « Modele » qui contient les différentes classes qui sont à la base de l'application. Il y a les classes JeuVidéo, UtilisateurConnecté, Administrateur ainsi que des énumérations.

Il y a le projet Managment qui est le gestionnaire de notre application. Il contient une classe Manager qui contient la plupart de notre nos fonctionnalités telles que la création et suppression d'un jeu. La création d'un compte et la connexion à un compte, etc… Ce projet contient aussi une classe avec des méthodes statiques de tris sur la liste qui sera affiché dans l'application. De plus, il y a une classe qui contrôle la création d'un utilisateur et d'un jeu vidéo.

Il y a enfin le projet Persistance, ce projet est assez complet car il contient nos données virtuelles dans notre Stub, ainsi que notre persistance dans un fichier XML. Cette hiérarchie démarre avec une interface « IDataManager » que nous avons implémenté à notre stub ainsi qu'à notre BddDataManager, classe qui effectue la persistance. Nous avons décidé d'avoir une persistance qui n'impacte pas l'écriture de nos différentes classes dites hermétique, on ne retrouve aucune trace de l'utilisation des dataContract, DataMember dans nos classes dites POCO (Modele). On utilise des objets DTO (Data transfert Object) pour persister les données. A chaque sérialisation les objets de types POCO seront transformés en DTO à l'aide des méthodes ToDtos et ToDTO et inversement lors de la désérialisation les méthodes ToPOCOs et ToPOCO pour charger les données.

Les différents tests vérifient que les méthodes des classes et les interactions entre les différentes classes soient fonctionnels et ne présentent aucun bug.

Description patrons de conceptions

Nous utilisons un patron de conception. C'est est un singleton, il est utilisé dans notre classe Navigator dans le projet GameLib. Il fait en sorte qu'une seule instance du Navigator soit instanciée dans notre code-behind. Il permet d'avoir accès à cette même instance à l'aide de la méthode GetInstance() ; cela nous permet de contrôler qu'un seul Navigator soit instancié afin d'avoir une navigation totalement fonctionnelle.

Description des dépendances

Enfin, nous allons parler des dépendances entre nos différents projets.

La vue GameLib_Projet est liée à la persistance pour charger un Stub ou des données serialisées, elle est liée au Modèle pour instancier des objets auxiliaires lors de la création de jeux ou utilisateurs ainsi qu'un Manager qui va permettre d'utiliser toutes les fonctionnalités prévues pour notre application.

Managment possède des dépendances envers Modele car il contient des listes de jeux et d'utilisateurs, envers Persistance car il contient une propriété de type IdataManager qui lui permet de charger les données reçues dans les différentes listes de manager.

Modele n'a aucune référence vers d'autres projet car il est la racine de l'application et ne dépend de rien pour exister.

La persistance a une dépendance vers Modele car il instancie dans la classe BddDataManager des objets de type POCO qui sont des objets du Modele lors du chargement des données à partir du fichier XML ; et dans le Stub des objets de type JeuVidéo, Utilisateurs et Administrateurs.

Les Test ont des dépendances envers tous les projets excepté la vue afin de tester toutes nos fonctionnalités.

Description des ajouts personnels

Enfin, nous pensons que nos ajouts personnels sont la possibilité de se connecter à l'application ce qui donne des droits supplémentaires à l'utilisateur qui se serait connecté tels que l'ajout en favori d'un jeu à sa liste de jeux.

Aussi nos fonctionnalités de tris de la liste de jeux et de recherche de jeux par un nom complet ou non.