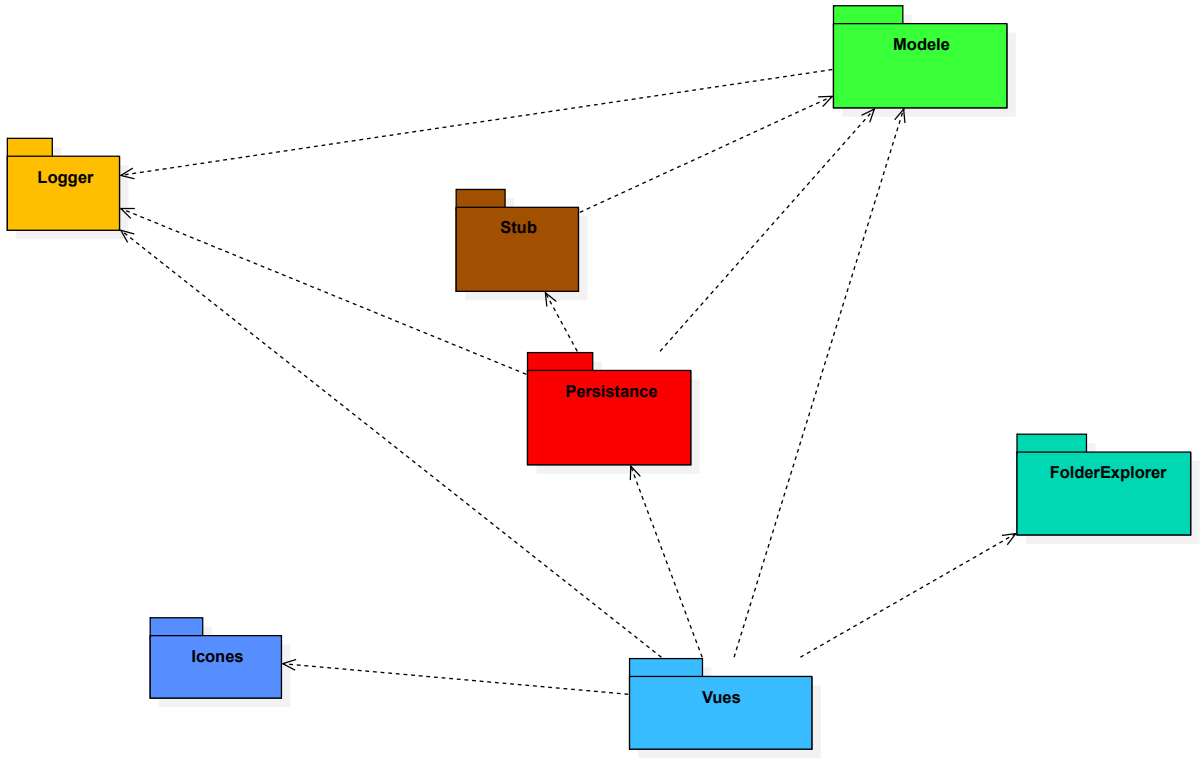
Documentation Projet Tutoré

-Persistance

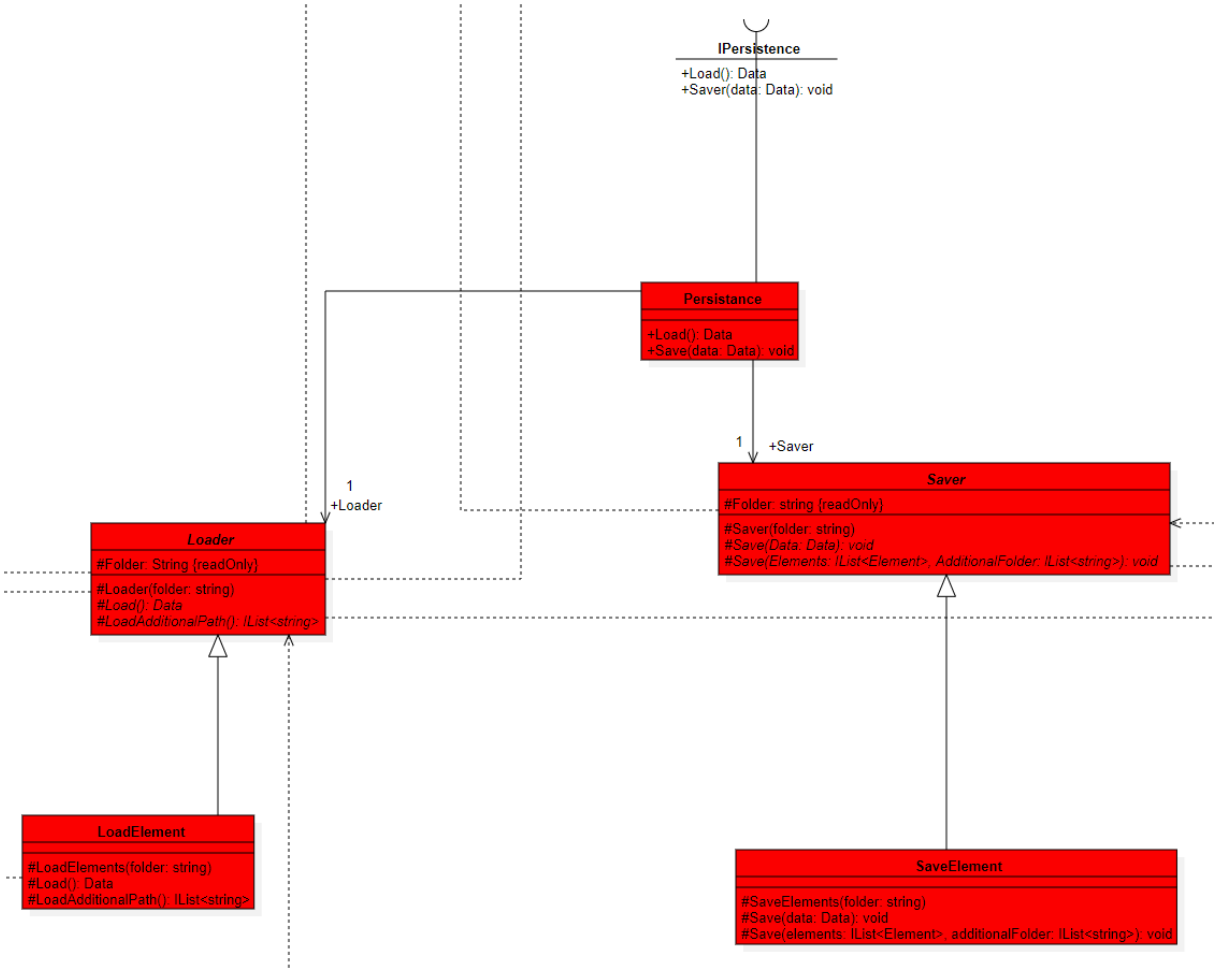
Diagramme de paquets :



Persistance dépend de Modele, Stub et Logger.

Persistance s'occupe de sauvegarder et charger les données de l'application. Elle dépend donc de Stub pour charger des jeux lorsqu'il n'y en a pas dans l'ordinateur (le professeur n'aura pas de jeu il faut bien qu'il puisse voir ce que propose l'appli). Elle dépend également de Modèle afin de créer des Elements.

Diagramme de classes :



La persistance de notre application est réalisée par le paquet Persistance. Ce paquet dépend de Modele pour pouvoir utiliser les types de Modele. Persistance implémente IPersistance qui se situe dans le Modele.

Persistance permet d'utiliser les classes LoadElement et SaveElement qui sont internal.

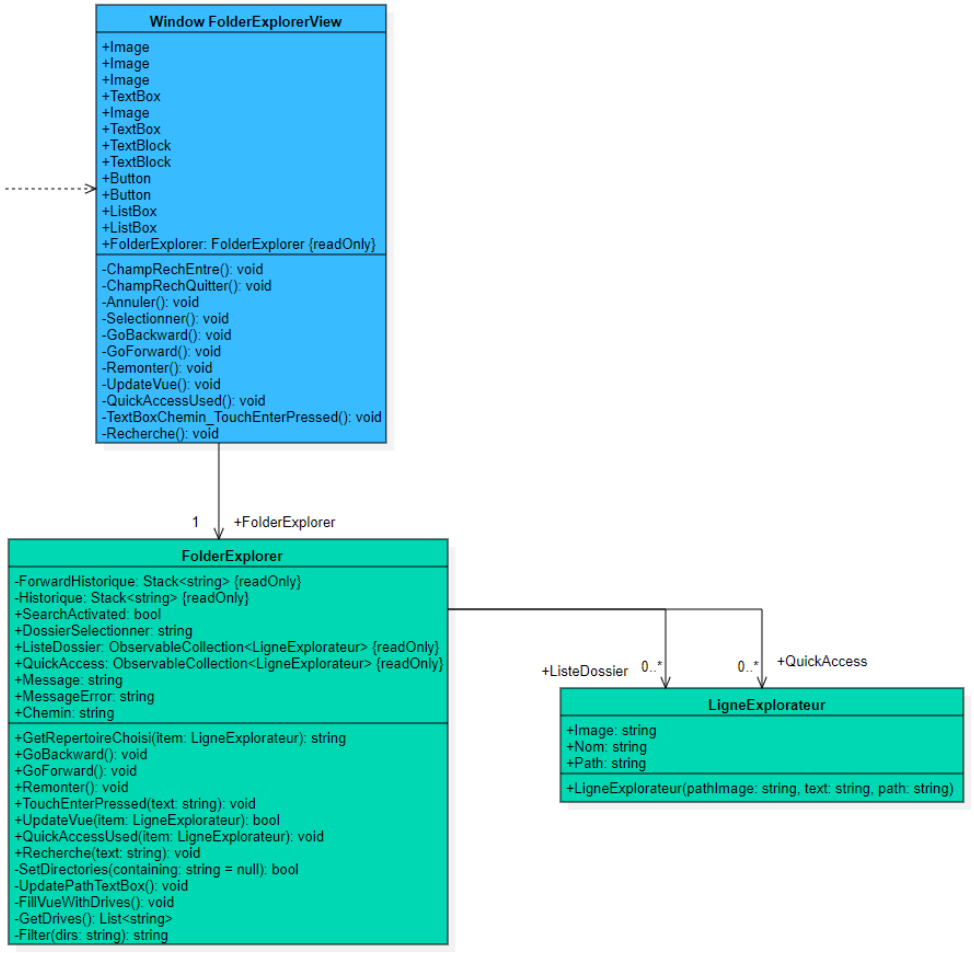
Lors de la sauvegarde, on sérialise la liste d'élément du Data contenu dans le Manager ainsi que les dossiers à afficher dans paramètre en fichier XML.

Lors du chargement, on récupère le contenu du fichier XML et on en extrait les informations importantes. Ensuite, on regarde si la sauvegarde contient tous les jeux qu'elle devrait contenir (exemple : un jeu a été désinstallé entre 2 lancements de l'appli). Si la sauvegarde n'est pas à jour on récupère les jeux présents sur l'ordinateur du client. Si aucun jeu n'est présent on charge le stub.

Enfin, chaque jeu reçoit ses informations, données par SearchInfo.

Chaque jeu récupère ses informations dans un thread séparé afin de ne pas gêner le déroulement du programme.

-Participation personnelles



Ces 3 classes implémentent la logique et la vue d'un explorateur de dossier

LigneExplorateur représente une ligne dans l'explorateur c'est-à-dire une image et un nom. Il contient aussi son chemin correspondant. De cette manière on retourne le chemin du LigneEplorateur sectionné dans la vue (fonction GetRepertoireChoisi()).

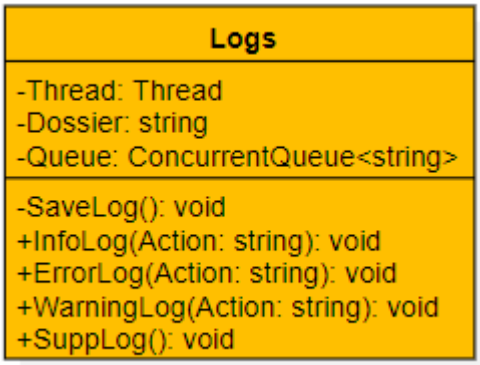
SetDirectories() rempli ListeDossier avec les répertoires a affiché. Pour cela il s'appuie sur l'historique. Lorsque SetDirectories() récupère un null donné par l'historique alors on affiche la racine (les disques ; équivalent à '/' sous Unix)

UpdateVue récupère en paramètre l'item cliquer depuis la vue et lance SetDirectories() en ayant préalablement mise à jour l'historique.

QuickAccess est rempli à l'initialisation avec les dossiers commun (Documents,Images…) et les disques de l'utilisateur. Lors d'un clic sur un des éléments de QuickAccess, la fonction QuickAccessUsed est lancer et lance SetDirectories() en ayant préalablement mise à jour l'historique.

FolderExplorerView correspond à l'affichage d'un explorateur de dossier pour cela il s'appuie sur FolderExplorer présenter plus haut.

Chaque évènement appel son équivalent dans le FolderExplorer.



La classe Logs sert à faire des Log de l'appli. Cette classe n'a pas de lien dans le diagramme de classe car elle pourrait être présente dans toutes les classes de l'appli. On laisse donc les développeurs l'utiliser quand bon leur semble.

Les threads permettent de gérer l'écriture dans le fichier sinon une erreur telle que l'ouverture d'un fichier déjà ouvert par un autre processus peut apparaitre.

Lorsqu'on ajoute une string dans la Queue on vérifie si le thread tourne ou pas et on relance le thread si jamais il était à l'arrêt. Ce qui permet d'éviter l'erreur.

InfoLog() ajoute une ligne concernant une info dans la Queue.

ErrorLog() ajoute une ligne concernant une erreur dans la Queue.

WarningLog() ajoute une ligne concernant une information importante dans la Queue.

SuppLog() regarde la date du premier log dans le fichier puis réinitialise le fichier si elle n'est pas égale à la date d'aujourd'hui (cela laisse au moins un jour pour voir le fichier et évite d'avoir un fichier contenant des milliers de lignes).

SaveLog() est exécuté par le thread et tant que la Queue n'est pas vide il écrit dans un fichier (Dossier) le contenu de la Queue.

Autres participations personnelles :

Dans la classe SearchInfo nous avons utilisé des requêtes internet pour chercher des informations sur des sites.

Dans les classes SteamSearcher, RiotSearcher, EpicSearcher et UplaySearcher nous avons utilisé les registres de Windows afin de pouvoir localiser les jeux.