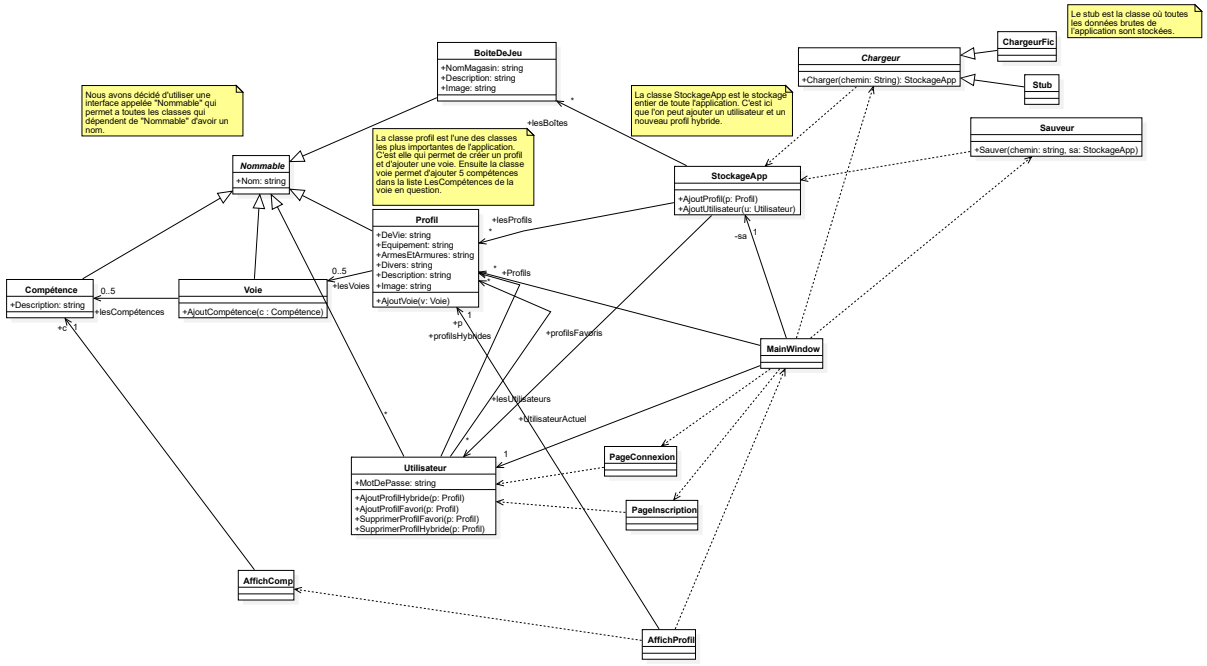
Diagramme de classe :



L'une des classes les plus importantes de notre application est la classe **Profil**. C'est cette dernière qui permet de créer un profil. Elle contient toutes les informations d'un profil du jeu. Elle dépend de la classe **Voie**. La classe **Profil** contient une liste **lesVoies** qui contient 5 voies. Ces 5 voies doivent être différentes. Chaque voie possède une liste **lesCompétences** de 5 compétences qui viens de la classe **Compétence**. Ces 5 compétences doivent être différentes.

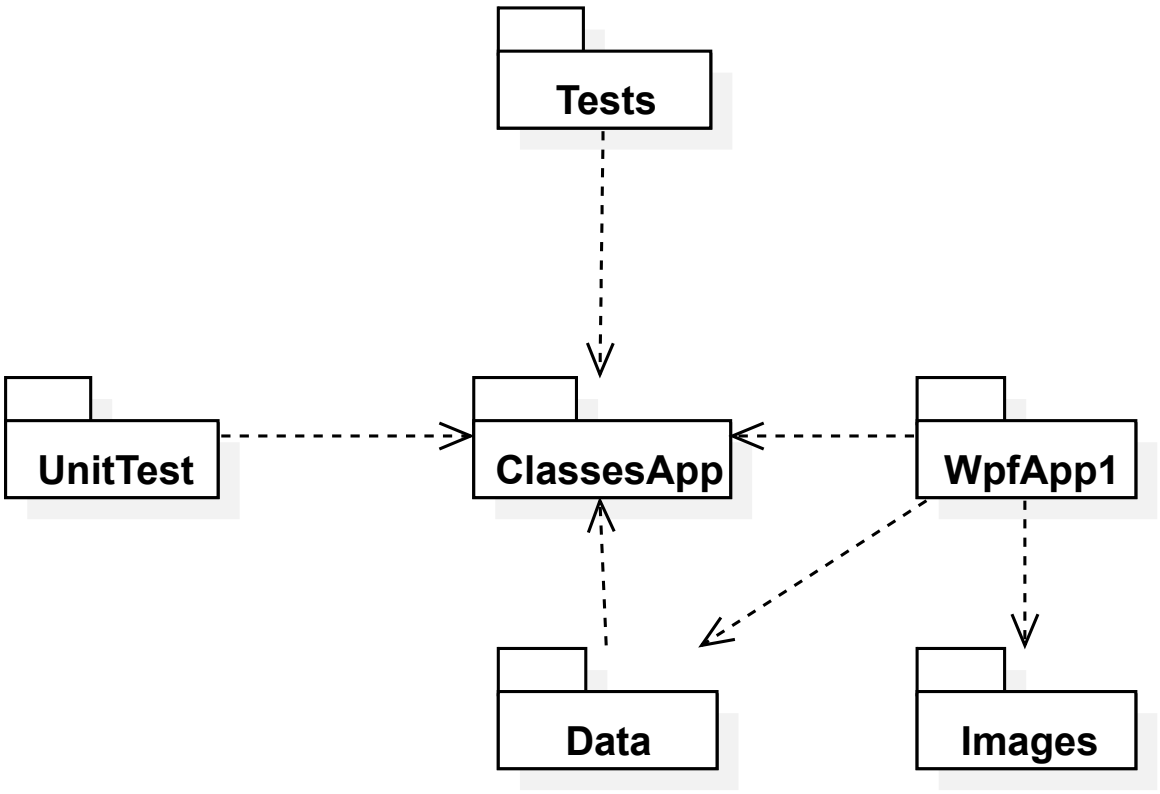
L'utilisation de l'interface **Nommable** évite une répétition du code car les classes : **BoiteDeJeu**, **Profil**, **Compétence**, **Voie** et **Utilisateur** possèdent un nom.

La classe **Utilisateur** permet à une personne utilisant l'application de se créer un compte. Une personne, pour créer un compte doit avoir un nom et un mot de passe. Avoir un compte permet de mettre des profils en favoris et de créer des profils hybride. Les profils favoris sont stockés dans la liste **profilsFavoris** dans la classe **Utilisateur**. Les profils hybrides sont stockés dans la liste **profilsHybrides** dans la classe Utilisateur. Mettre ces listes dans la classe Utilisateur permet que ces listes soient privées. Seul l'utilisateur pourra accéder et modifier à ces listes.

La classe **BoiteDeJeu** permet de créer une boite de jeu. Nous avons décider de créer une classe car si une nouvelle boite de jeu est publiée, il suffira de créer une boite de jeu dans le stub plutôt que dans le code XML.

La classe **Sauveur** permet d'enregistrer les données de création d'un nouvel utilisateur ou l'ajout d'un profil hybride et ajout d'un profil en favori. Ces données sont enregistrées dans un fichier binaire.

Diagramme de paquetage :



Le package **Tests** contient tous les tests fonctionnels de l'application.

Le package **UnitTest** contient tous les tests unitaires de l'application.

Le package **ClassesApp** contient toutes les classes de la conception et de la programmation orientées objets.

La package **WpfApp1** contient toutes les classes de la vue, c'est-à-dire toutes les pages de l'application.

La package **Images** contient toutes les images de l'application.

Le package **Data** contient toutes les données de l'application.

Toutes les données de l'application sont stockées dans le package **Data** ce qui permet de mettre toutes les données au même endroit. Aussi, c'est dans ce package que les différentes classes permettant d'interagir avec les fichiers. De plus, le fait d'avoir séparé les données du code permet d'éviter tous conflit lors d'ajout de données. Le package **ClassesApp** est indépendant. Si le package **Data** est supprimé, le package **ClassesApp** fonctionne quand même.

Nous avons décidé de faire de même avec les images. Nous avons créé une bibliothèque d'images indépendantes de la vue. De plus cela nous permet de mettre nos images en tant que ressources et non en tant que contenu.

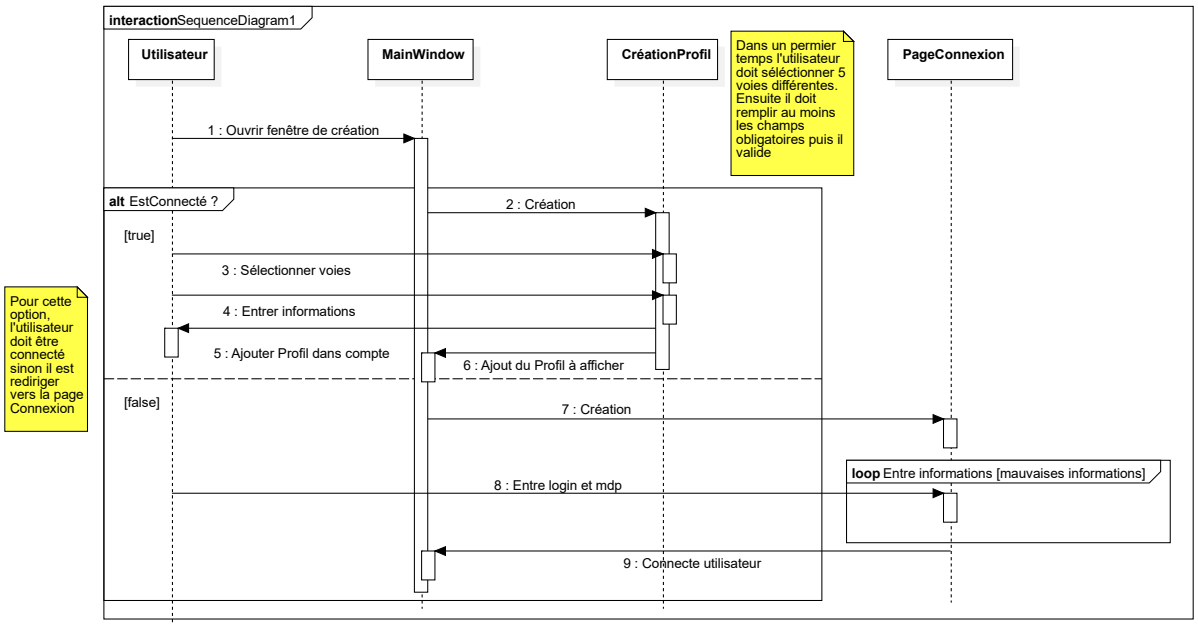


Diagramme de séquence :

Le diagramme de séquence décrit la création d'un profil hybride pour un utilisateur. Tous d'abord, on test si l'utilisateur est connecté. S'il n'est pas connecté, la page **Connexion** apparaît, et il doit se connecter. Sil se trompe dans son code ou son identifiant il peut recommencer. S'il n'a pas de compte, il doit quitter la page **Connexion** et cliquer sur le bouton **Inscription**. Ensuite, il peut cliquer sur le bouton **Création hybride**, ce qui fera s'ouvrir la page **CréationProfil**. En haut de la page, il y a la liste de 14 profils de base. Il pourra cliquer sur l'un d'eux pour pouvoir afficher ses voies. Il devra ensuite cocher 5 voies différentes. Il devra cliquer sur le bouton **Valider les voies**. Ensuite il devra remplir les champs obligatoires. Ils sont représentés avec une * à côté. S'il le veut, il peut remplir les informations complémentaires **Divers** et **Description du nouveau profil**. Pour créer son profil, il devra juste cliquer sur le bouton **Valider**. Il retrouvera son profil dans le master sur la **MainWindow**.