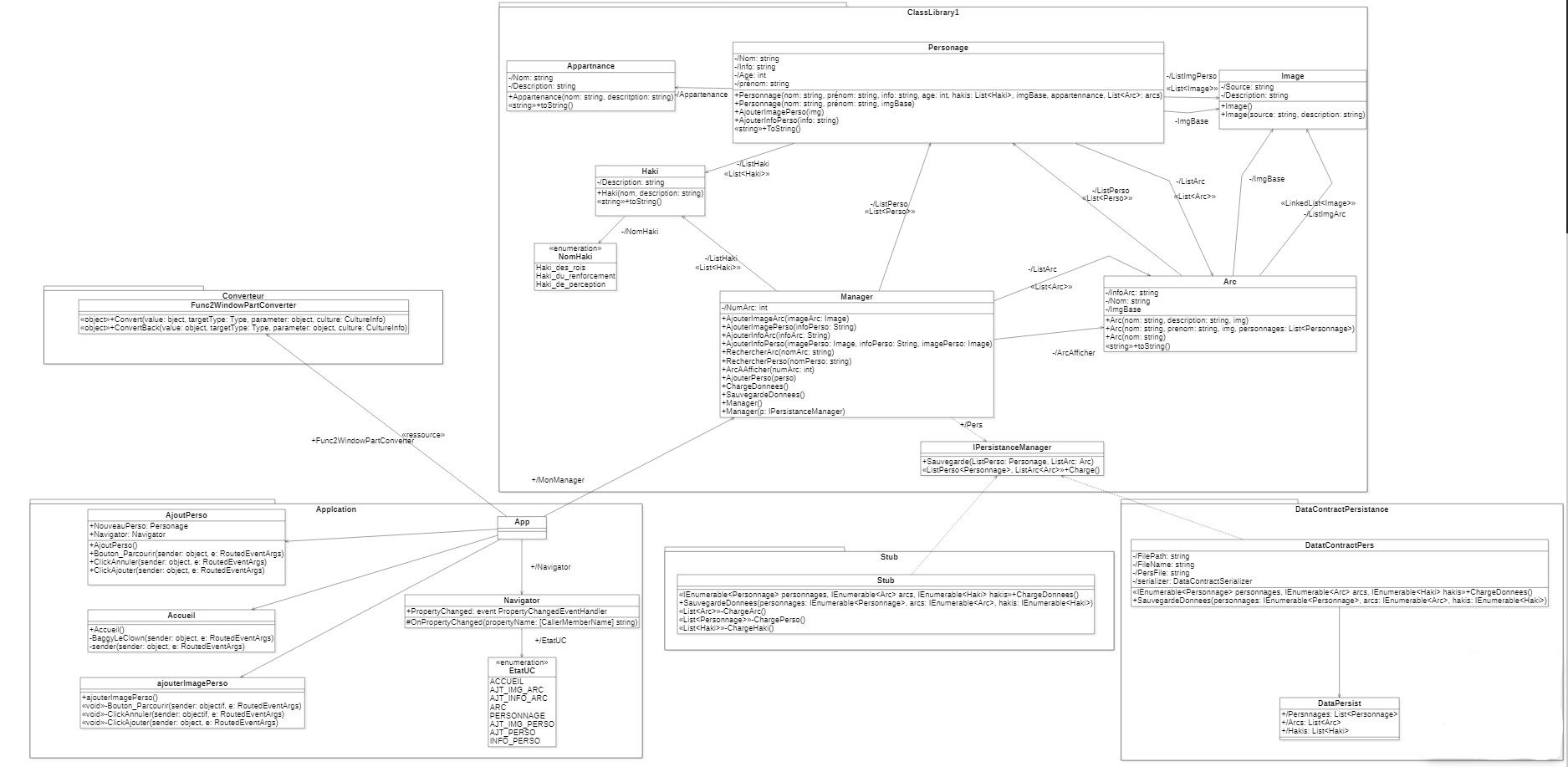
**Diagramme de classes**

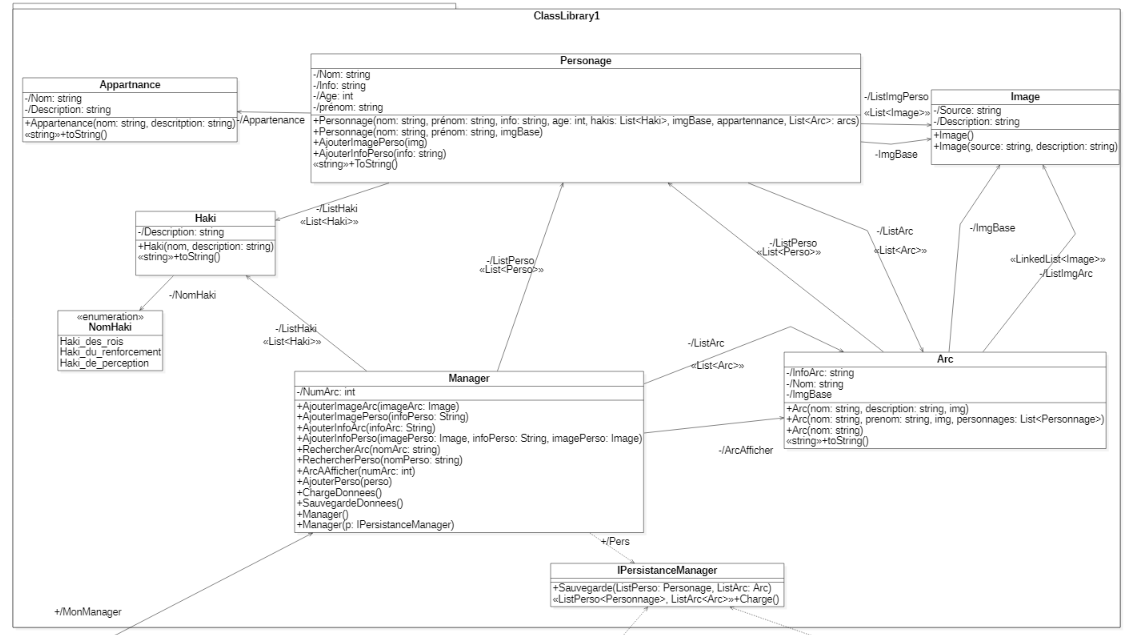


Voici donc notre diagramme de classe complet. Il comporte 5 paquets : Application, ClassLibrary1, Converteur, Stub et DataContractPersistance. Bien sûr dans ce diagramme de classe des éléments ne figure pas par soucis de place, nous n’arrivions pas à tout faire tenir donc on préféré enlever certaines vus ainsi que certaine méthode peu utile tel que les toString ou les constructeur que l’on a pas forcément toujours précisé.

Le paquet Converteur est relié à l’Application par les ressources de app.xaml. Il ne contient qu’un seul convertisseur Func2WindowPartConverter.cs pour le moment. Nous en rajouterons un pour convertir les NomHaki de la classe Haki qui sont issu de l’énumération NomHaki.

Le paquet ClassLibrary1 est relié à l’Application par le manager qui est déclaré en propriété dans App.xaml.cs.

Pour les paquets Stub et DataContractPersistance la même technique à été utilisé pour les relié à ClassLibrary, la class principale des deux paquet implémente l’interface IPeristanceManager. On précise que c’est deux paquets sont prévus pour le stockage des données. Le paquet stub ne peut que charger les données dans l’application. Tandis que le paquet DataContractPersistance lui permet de charger mais aussi de sauvegarder les données de l’application.

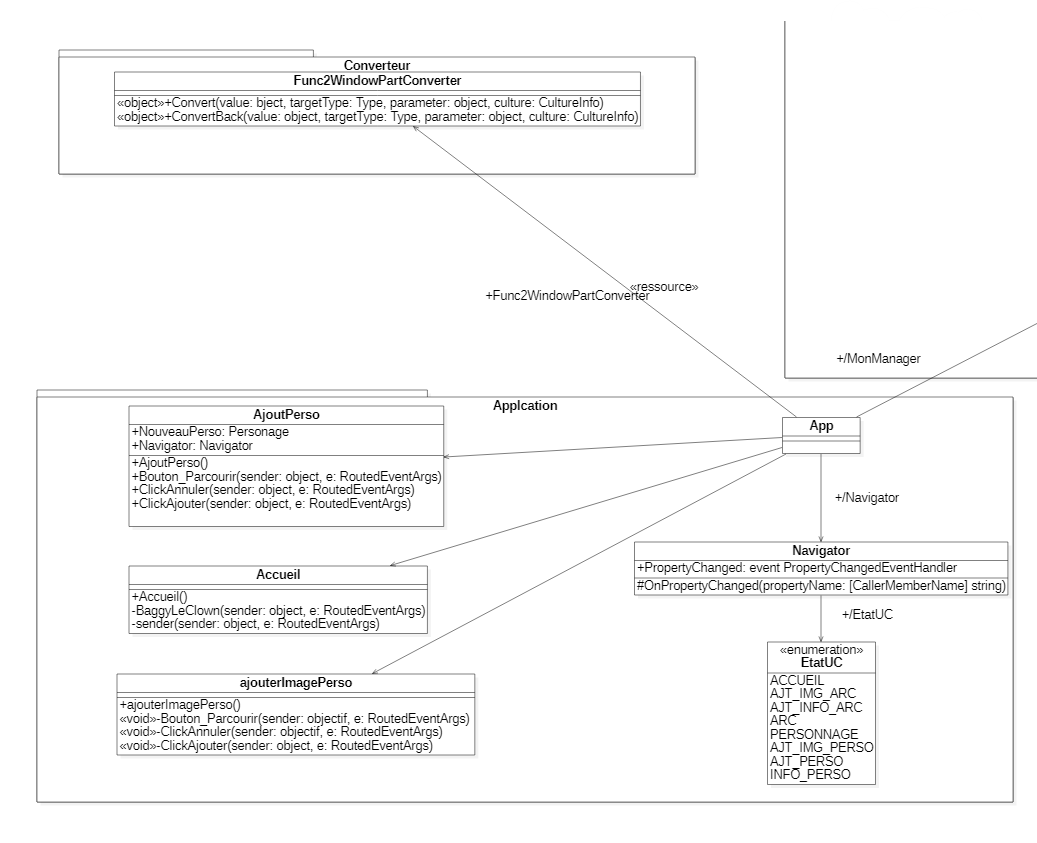


Maintenant rentrons un peu plus dans le détail de ClassLibrary :

Ce paquet contient :

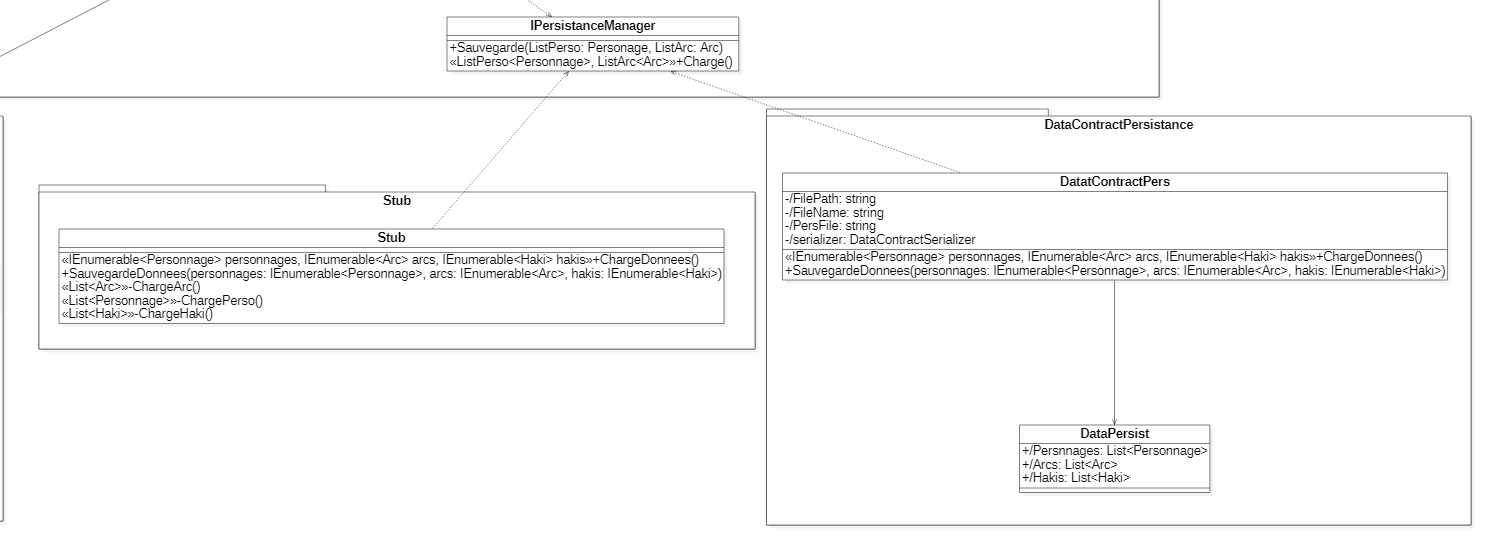
* 6 classes
* 1 énumération
* 1 interface

Toutes les classes sont là pour présenter les personnages et les arcs sauf le manager qui réunit toutes les classes. En effet le manager a 3 collections importantes, une pour les personnages, une pour les arcs et une pour regrouper les différents hakis. Toutes les collections de notre application sont des listes car nous n’avions pas besoin ni de dictionnaire, ni de collection complexe, de simple liste suffisait. Nous pouvons préciser que le Manager est relié au IPersistanceManager en l’implémentant mais aussi en ayant une propriété qui nous permet de changer de mode de sauvegarde ou de récupération. Nous avons également mis toutes nos propriétés en privé pour respecter l’encapsulation. Nous avons par choix pas mis l’interface « INotifyPropertyChanged » dans le diagramme car nous n’avions plus de place. Cependant nous précisons quand même que **le manager implémente INotifyPropertyChanged**.



Maintenant voyons plus le paquet Application, qui est appelé Projet dans le diagramme de paquetage. Il est lié à ClassLibrary avec la propriété MonManager. Il est ensuite relié à toutes les vues de notre application que nous n’avons pas pu représenter ici par soucis de place. Nous avons gardé celle qui représente le plus notre application :

* AjoutPerso, qui nous permet d’ajouter des personnages les boutons ont toujours les mêmes paramètres, le nouveau personnage est là pour pouvoir être ajouter et le navigateur pour naviguer sur une autre page.
* Accueil qui contient un des boutons pour chacun de nos arc, nous ne les avons donc pas tous représenté ici.
* ajoutImagePerso qui comprend des boutons qui ont tous les même paramètres



Et pour finir sur ce qui est plutôt de la partie 4, la persistance. Nous avons donc dans un premier temps fait le **Stub** pour pouvoir chargé les donnés qui sont rentré en dur dessus. Et dans un second temps le **DataContractPersistance** pour pouvoir avoir un fichier de chargement mais aussi de sauvegarde. La classe Stub permet de charger les données qu’elle contient. DataContract permet de charger et sauvegarder les données dans un fichier dont toutes les infos sont en propriété, le nom : FileName, le chemin : FilePath. PersFile est là pour pour combiner le nom et le chemin d’accès. Nous utilisons la classe DataPersist pour pouvoir chargé toutes les donnés dans le même fichier.