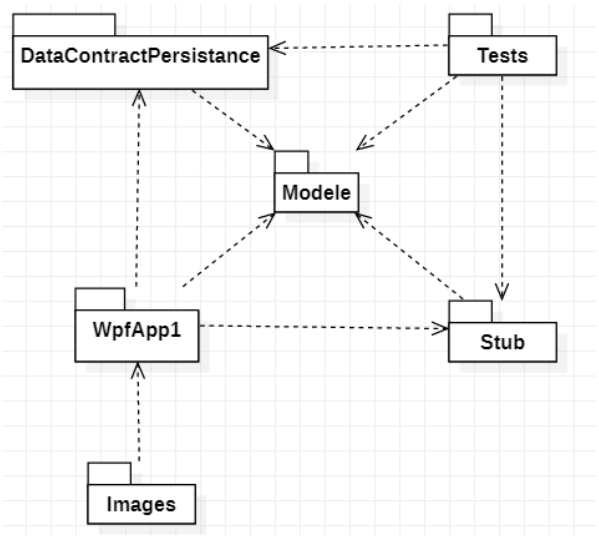
Documentation : Conception et Programmation Orientée Objet (C#, .NET)

Sommaire :

1. Introduction, description globale de la structure de l'application

Dans cette partie, la description globale du projet et de sa structure à travers le diagramme de classe (pages 3 et 4) et le diagramme de paquetage (présent sur cette page) vont permettre de comprendre le fonctionnement de l'application.

(La partie Persistance de l'application est plus détaillée dans DocumentationProjetTuteuréS2.pdf)



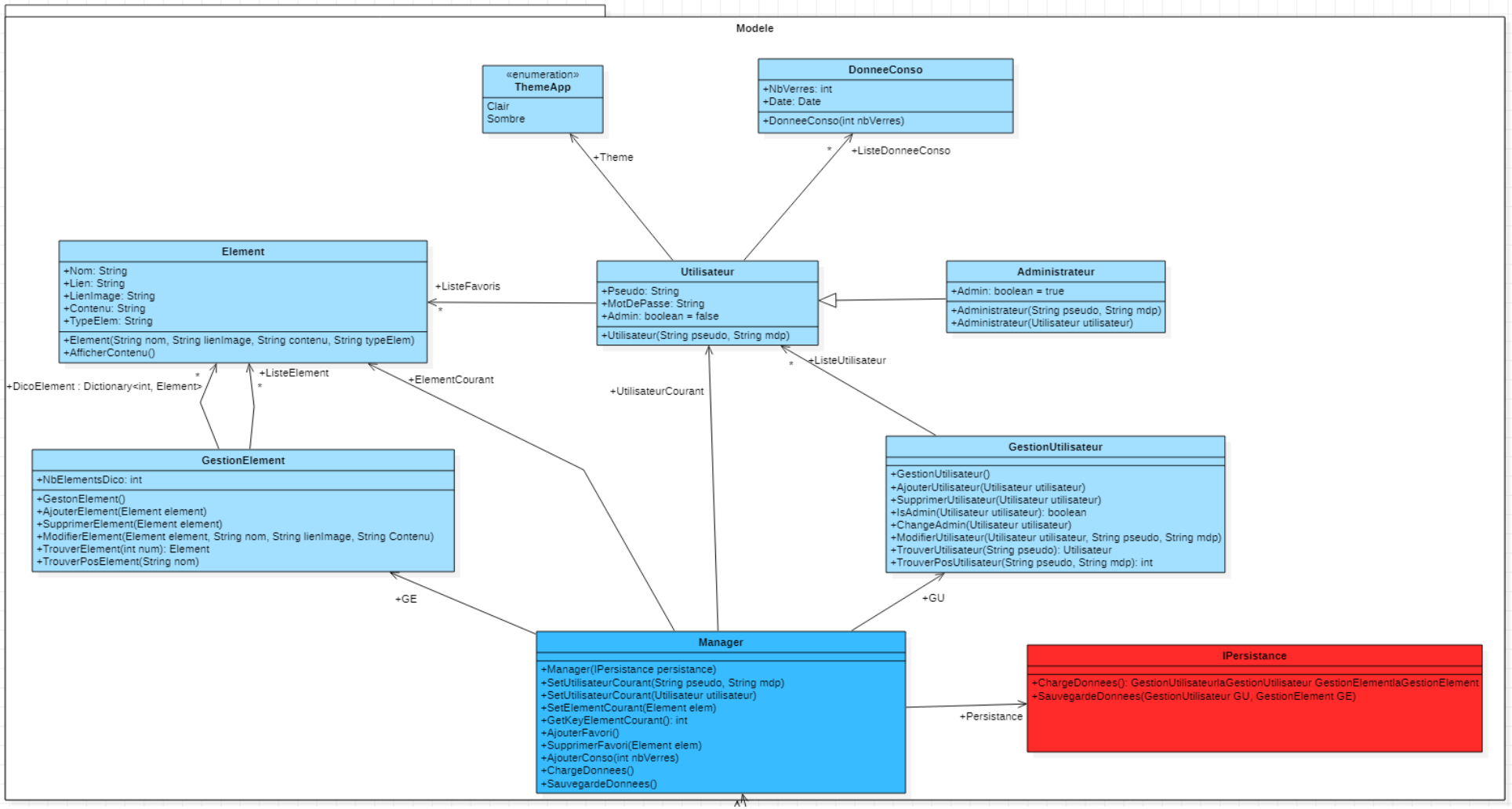
Le diagramme de paquetage ci-dessus montre que l'essentiel de l'application repose sur le modèle. En effet la partie « Vue » (WpfApp1) est dépendante du Modele et utilise ses classes.

Le package DataContractPersistance permet la sérialisation des classes du Modele (des précisions sont ajoutées dans la doc du projet tuteuré).

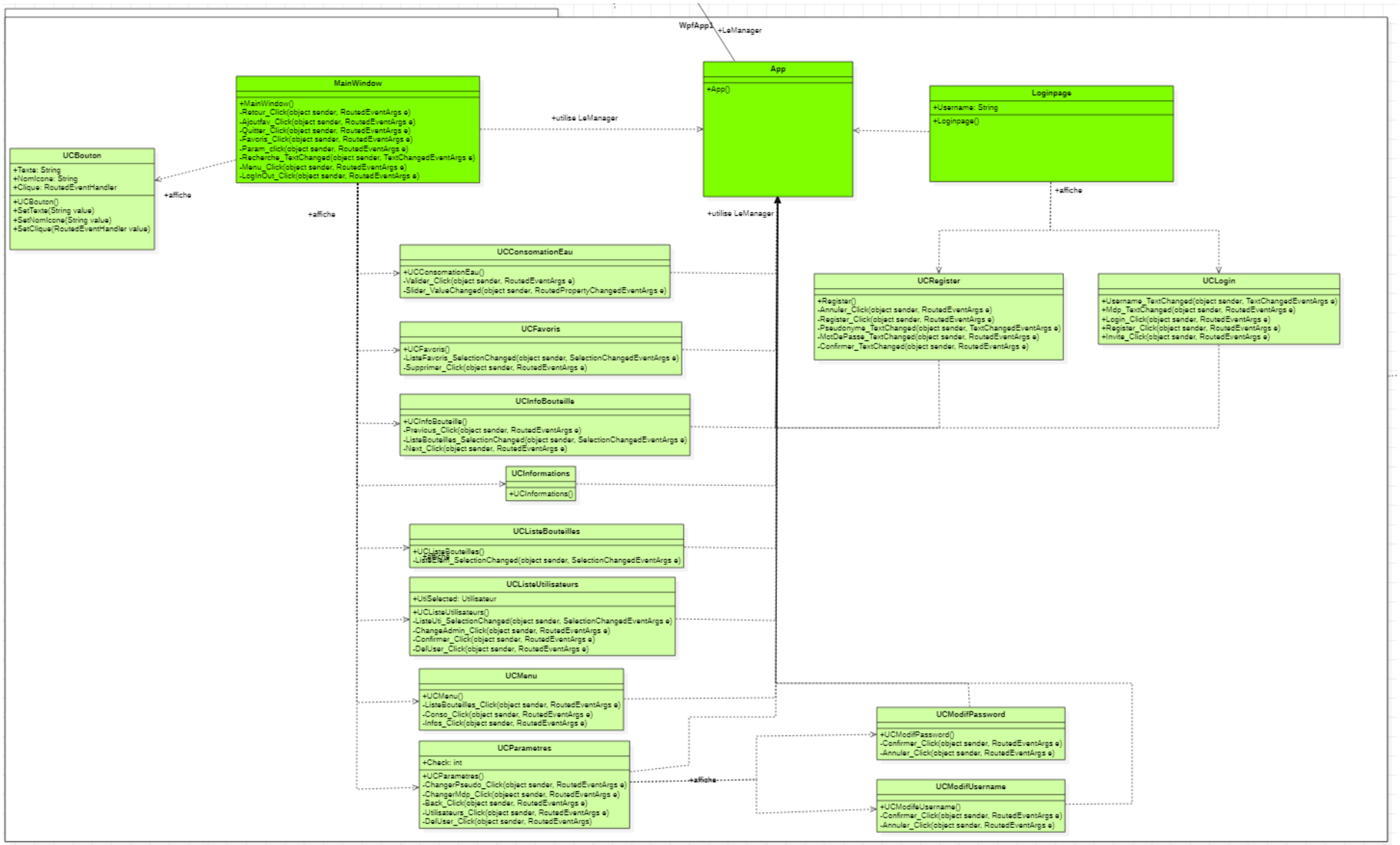
WpfApp1 est aussi dépendante du Stub pour permettre « l'injection » dans la Vue des premières données.

Ce diagramme reste simple et témoigne d'une architecture facile à comprendre, expliquée en détail juste après sur cette documentation.

1.1. Diagramme de classe : Package Model

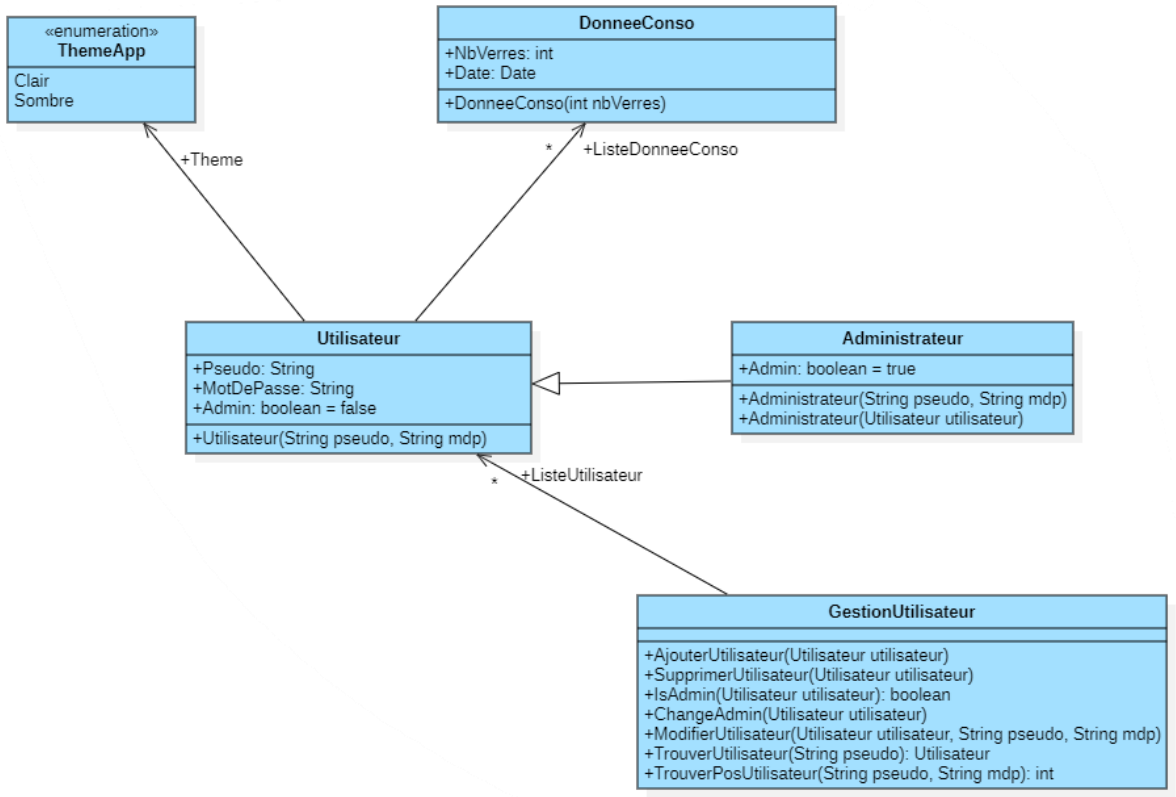


1.1. Diagramme de classe : Package WpfApp1



2. Description des classes utilisées par l'application

2.1. Côté gestion des utilisateurs



La classe Utilisateur permet un système de connexion dans l'application :

Elle est composée des propriétés Pseudo et MotDePasse (nécessaires à l'instanciation d'un nouvel Utilisateur) ainsi qu'Admin qui est un booléen « Set » false par défaut.

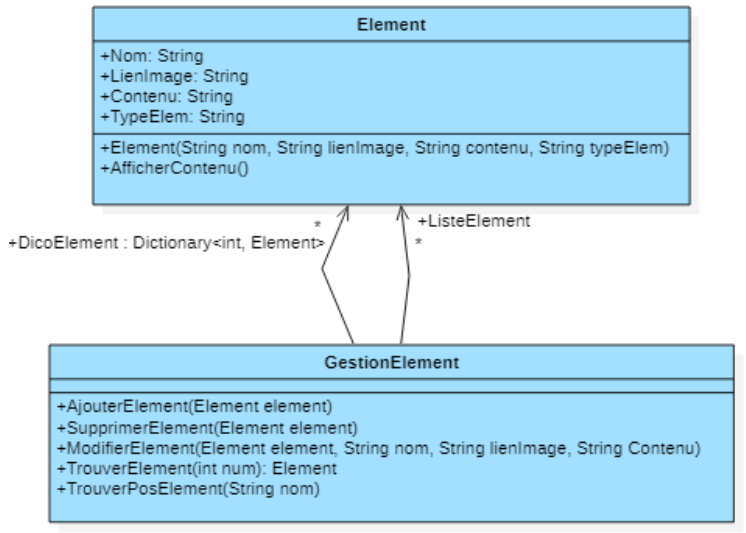
Elle contient aussi une liste d'objets de la classe DonneeConso, en lien avec la fonctionnalité « Consulter sa consommation d'eau ». Chaque DonneeConso a pour propriété une Date et un nombre de verres bus à cette date NbVerres.

Enfin, la propriété Theme de l'Utilisateur correspond à une enum (Clair par défaut) qui pourra être redéfinie en Sombre par la suite (fonctionnalité de modification des paramètres de l'application).

La classe Administrateur qui hérite d'Utilisateur permet quant à elle la création d'un Utilisateur dont Admin est égal true, lui permettant par la suite l'accès à des méthodes que la classe Utilisateur ne peut pas réaliser (cf. diagramme de cas d'utilisation de la doc. IHM).

La classe GestionUtilisateur « englobe » tout ceci grâce à une liste ListeUtilisateur pour permettre par la suite l'ajout, la suppression ou encore la modification d'un Utilisateur. Cette classe contient en fait l'ensemble des Utilisateurs inscrits à notre application.

2.2. Côté gestion des éléments



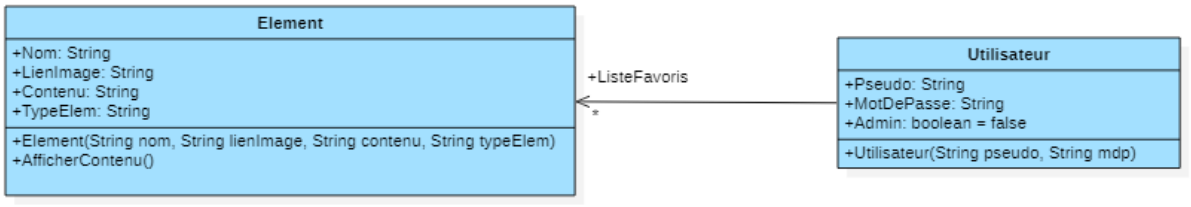
La classe Element correspond aux différentes informations qui seront affichées dans la vue comme, par exemple, dans le Master Detail.

Cette classe a pour propriétés Nom, LienImage, Contenu et TypeElem. Les noms des propriétés sont assez explicites, une précision sur TypeElem est nécessaire : cette propriété permet de classer les éléments en 2 catégories avec la classe GestionElement.

GestionElement contient un Dictionary<int, Element> DicoElement contenant tous les Elements de type « info » ainsi qu'une liste ListeElement contenant tous les Elements de type « bouteille ». Cette distinction de type permet de définir quels éléments serviront de « Contenu » à notre application (type « info ») et quels éléments serviront pour le MasterDetail (type « bouteille »).

GestionElement joue le même rôle que GestionUtilisateurs mais pour la classe Element avec des fonctions comme l'ajout, la suppression, la modification d'éléments présents dans la liste ou le dictionnaire.

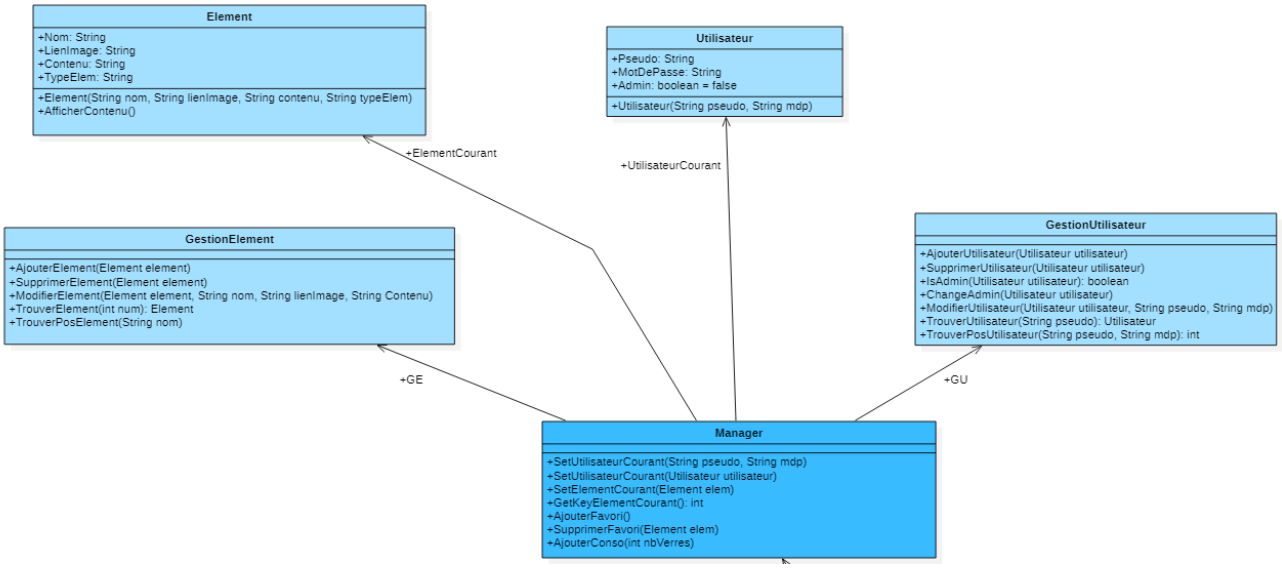
2.3. Interactions entre ces 2 types



La classe Utilisateur contient aussi une List<Element> ListeFavoris qui permettra à l'utilisateur de sélectionner ses pages favorites une fois dans l'application (fonctionnalité de favoris du diagramme de cas d'utilisation).

Pour permettre cette fonctionnalité ainsi que les autres fonctionnalités, il faut maintenant définir quel Utilisateur attribuer à quel Element. C'est pourquoi la classe Manager joue un rôle majeur dans le Modele (cf. page suivante).

2.4. Manager : son rôle



(Sur ce screenshot ne figurent pas les méthodes de sauvegarde et de chargments des données dans la classe Manager)

En effet la classe Manager est la classe la plus importante du Modele. A sa création, Manager instancie un objet de type GestionElement GE ainsi qu'un objet de type GestionUtilisateur GU.

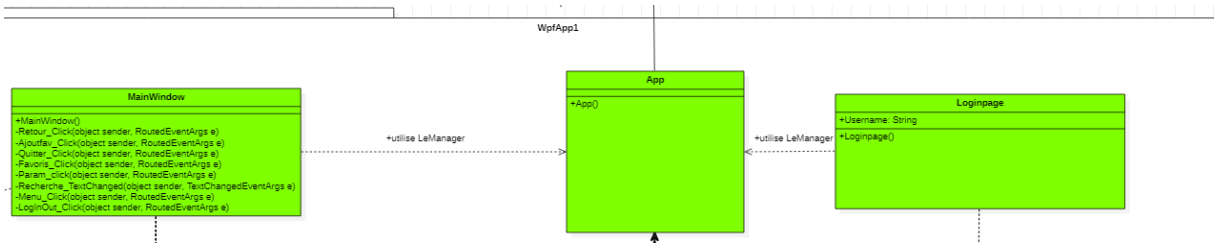
Ces deux éléments constituent, en quelque sorte, la base de données de l'application. Si une personne utilise l'application en se connectant à son compte existant, la propriété UtilisateurCourant est alors définie par cet utilisateur. Lorsqu'il se « balade » sur l'application, c'est la propriété ElementCourant qui est définie en fonction de la page courante ou de l'element du MasterDetail sur lequel il se trouve.

La définition de l'UtilisateurCourant et de l'ElementCourant rend alors possible l'utilisation des fonctions d'ajout/suppression de favoris ou encore l'enregistrement de la consommation quotidienne d'eau de l'utilisateur.

Le Manager est aussi la classe qui fait le lien entre Modele et Vue (WpfApp1) (cf. page suivante).

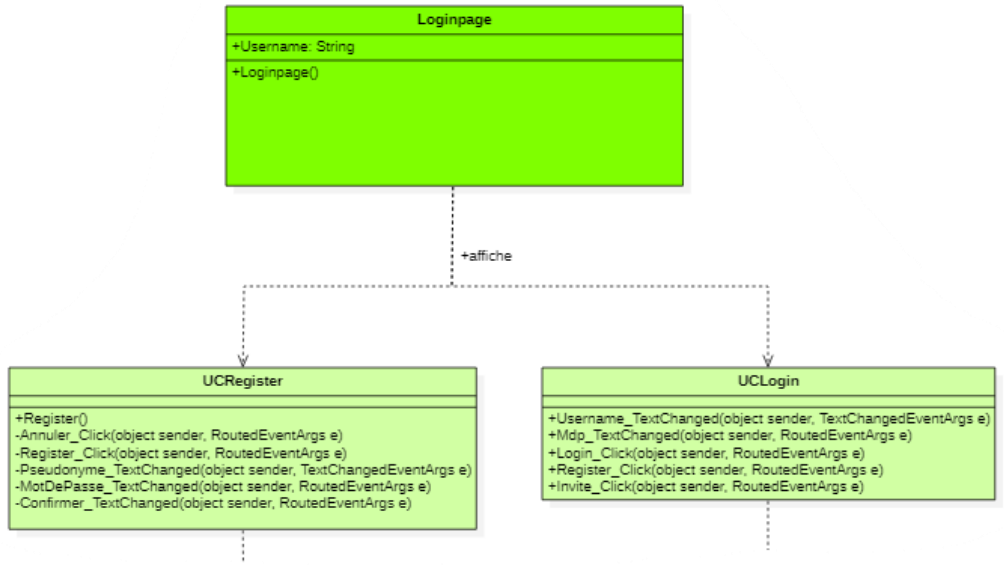
3.Description de la partie « Vue » de l'application

App.xaml étant le point de départ des déclarations de l'application, c'est dans son Code Behind que sera instancié un Manager : LeManager.



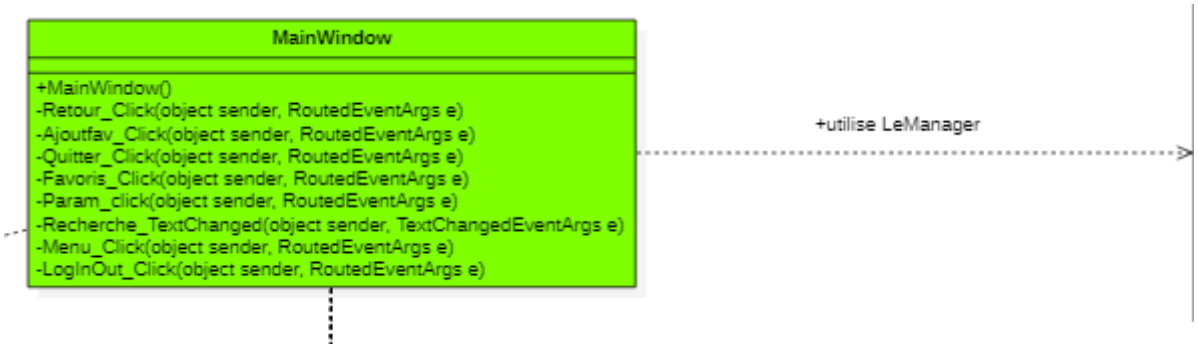
La vue est composée de trois fichier .xaml principaux : MainWindow.xaml, App.xaml et Loginpage.xaml. MainWindow et Loginpage utilisent le Manager instancié dans le Code Behind App.xaml.cs pour permettre la « transaction » de données à travers les fenêtres.

Dès l'ouverture de l'application, l'utilisateur arrive sur la fenêtre Loginpage.xaml.

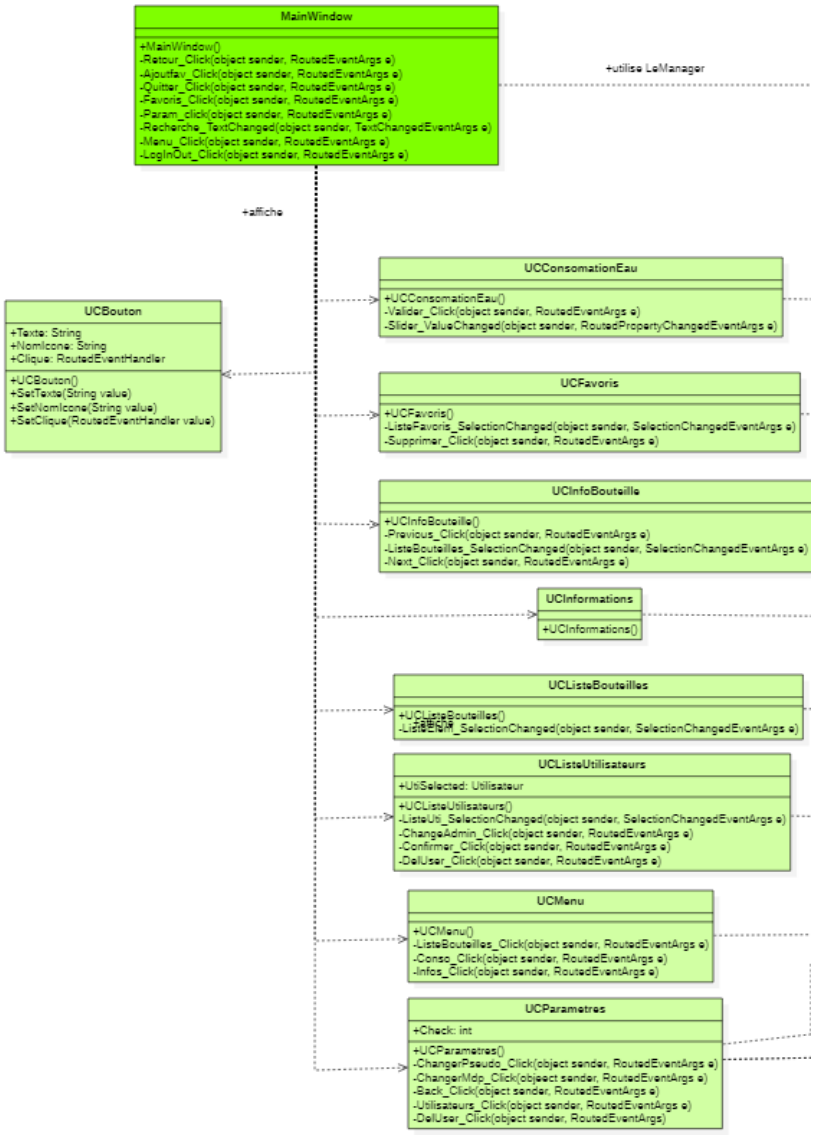


Le Content Control présent dans la fenêtre Loginpage afficher l'UserControl UCLogin par défaut. Si l'utilisateur ne possède pas de compte, il peut s'inscrire en cliquant sur un bouton changeant le Content Control actuel pour le User Control UCRegister.

Il est aussi possible d'accéder à l'application sans compte ou en se connectant directement



Une fois que l'utilisateur accède au contenu de l'application, il se situe sur la fenêtre MainWindow dont le contenu est composé en majorités de User Controls :



Les interactions de l'utilisateur avec l'environnement graphique de l'application sont la source de changement du contenu du Content Control, lui permettant ainsi de naviguer au sein de l'application.

4. Exemple de diagramme de séquence : fonction de suppression d'un utilisateur