



GPI – Gestion de Parc Immobilier

Par Olga Fragnière. IDEC 2017

Contenu

Cahier des Charges	3
1. Introduction.....	3
2. Contexte de l'application.....	3
3. Fonctionnalités métiers	3
4. Evolutions	6
Spécifications techniques	7
1. Environnement.....	7
Outils du développement / langages de programmation :	7
2. Architecture de l'application	7
2.1. Vue d'ensemble sur le projet GPI.Presentation.....	8
2.2. Vue d'ensemble sur le projet GPI.Persistance.....	8
3. Conception de couches de base de l'application.....	8
3.1. Couche <i>Persistance</i> et la base de données.....	8
3.2. Architecture de la couche <i>Présentation</i>	10
3.3. Architecture de la couche <i>Métiers</i>	11
4. Description technique des fonctionnalités du logiciel	12
4.1. Classes ViewModel.....	12
4.2. Spécifications techniques du Module « Offre »	13
4.3. Spécifications techniques du Module « Demande »	15
4.4. Spécifications techniques du Module « Vente »	17
4.5. Spécifications techniques du Module « Agent »	19
4.6. Spécifications techniques du Module « Client »	21
Manuel Utilisateur	23
1. Onglet « Offres ».....	23
2. Onglet « Demandes »	23
3. Onglet « Vente »	24
4. Onglet « Agent »	25
5. Onglet « Client »	26

Cahier des Charges

1. Introduction

Pour entrer dans le monde immobilier et commencer ses affaires de manière rapide et efficace, une jeune agence immobilière a besoin d'un logiciel avec lequel elle peut démarrer dans les meilleurs délais. Comme on dit : « le plus difficile c'est commencer ». Ce logiciel possède toutes les fonctionnalités nécessaires pour simplifier l'étape de lancement de l'agence qui est la plus difficile.

2. Contexte de l'application

L'application « Gestion de Parc Immobilier » est une application permettant de gérer les besoins élémentaires d'une petite agence immobilière. Complet et convivial, ce logiciel de gestion d'achat/vente de biens immobiliers a pour but de simplifier le quotidien des courtiers, des administrateurs et des directeurs de l'agence. Grâce à cette application ils peuvent être opérationnels dans un temps record et efficace dans la gestion de leur parc immobilier.

1.1. Objectifs de « Gestion de Parc Immobilier » :

- ✓ Mettre en place une organisation efficace et réactive aux modifications
- ✓ Gagner du temps dans les tâches administratives
- ✓ Répondre aux besoins d'une clientèle exigeante dans les meilleurs délais
- ✓ Gérer et accéder à la base de données, trier les offres selon la région et type d'immobilier, c'est-à-dire donner des réponses aux clients sur la disponibilité des biens très rapidement
- ✓ Chercher des biens immobiliers selon des paramètres exigés
- ✓ Avoir la vue complète sur le fonctionnement de l'agence : les ventes, les offres, des demandes, les agents et clients
- ✓ Générer facilement les rapports de gestion et calculer automatiquement les commissions de courtage de l'agence et des agents
- ✓ Entrer les données et suivre les propriétaires, saisir des mandats, générer les ventes, gérer le suivi des clients et des courtiers

3. Fonctionnalités métiers

L'application « GPI » est un logiciel intégré regroupant cinq modules de gestion, qui font partie intégrante du logiciel : « Offre », « Demande », « Vente », « Agent », « Client ». Chaque module permet d'ajouter, supprimer, modifier et voir les détails d'une instance (CRUD : Create, Read, Update, Delete), ainsi qu'imprimer le rapport sur cette instance. Chaque fois quand un changement est fait, les listes et les instances se mettent à jour.

3.1. Description des modules du programme

Module « Offre »

Depuis ce module nous pouvons voir toutes les offres de l'agence et leurs détails. Pour faciliter l'accès aux données, et la recherche, en plus de CRUD il y a les fonctionnalités suivantes :

Recherche :

- ✓ chercher des offres selon une région, où se trouve le bien ;
- ✓ chercher des offres selon le type de bien demandé ;
- ✓ chercher toutes les demandes des clients correspondantes à l'offre choisie, les modifier et supprimer.

Accès aux données :

- ✓ il est possible de voir toutes les détails sur le vendeur de l'offre choisie, les imprimer, modifier, supprimer ainsi que voir les offres et les demandes du vendeur.

Impression :

- ✓ dans ce module il y a la possibilité d'imprimer les détails d'une offre : adresse, type de bien, prix, surface, nombre de pièces et étages, étage, actualité de l'offre et le nom du vendeur.

Module « Demande »

Ce module permet de traiter les demandes des clients, qui s'adressent à une agence. En plus de CRUD, j'ai réalisé les fonctionnalités suivantes :

Recherche :

- ✓ recherche des offres appropriées selon les paramètres qui sont indiqués par un client afin de faciliter la recherche des offres et adapter la proposition à la demande ;
- ✓ chercher les demandes par nom du client correspondant.

Accès aux données :

- ✓ nous pouvons voir les informations sur le client, qui offre le bien correspondant à la recherche et les imprimer.

Impression :

- ✓ quand la recherche est faite, nous pouvons imprimer les résultats. Le rapport va comporter la liste des offres demandées avec les détails nécessaires.

Contraintes :

- ✓ comme la recherche est effectuée pour la demande d'un client donné, le résultat va uniquement comporter les offres actuelles.

Module « Vente »

Dans ce module il s'agit de la gérance des ventes de l'agence. Pour faciliter cela et assurer l'accès rapide aux données, en plus de fonctionnalités CRUD, il y a les fonctionnalités suivantes :

Recherche :

- ✓ chercher les ventes en fonction de la période de la vente (de telle date à telle date) ;
- ✓ chercher les ventes par noms des acteurs de la vente correspondante (Vendeur ou Acheteur).

Accès aux données :

- ✓ voir les informations sur les acteurs de la vente : Vendeur, Acheteur, leurs Offres et demandes, ainsi que l'Agent. Il est possible d'imprimer et modifier les résultats.

Impression :

- ✓ il est possible d'imprimer le rapport avec les données d'offres vendues, les montants de la commission de l'agence et de l'agent et les noms des acteurs de la vente.

Module « Agent »

Le but de ce module est de gérer les cadres d'une agence. En plus de fonctionnalités CRUD, il y a les fonctionnalités suivantes :

Recherche :

- ✓ possibilité de chercher un agent par son nom ou prénom.

Accès aux données :

- ✓ possibilité de voir la liste des ventes effectuées par un agent choisi;
- ✓ voir les commissions sur chaque vente effectuée et la commission totale d'un agent;
- ✓ voir toutes les détails sur chaque vente effectuée et les imprimer.

Impression :

- ✓ il est possible d'imprimer le rapport sur chaque agent avec les détails personnelles (adresse, téléphone etc.) ainsi que les détails de ses ventes et la commission gagnée.

Contraintes :

- ✓ vu que un agent fait partie des ventes, Il n'est pas possible de le supprimer définitivement. Par contre, il devient inactif et il ne figure plus dans aucune liste des agents actifs dans le programme. Cependant, s'il avait effectué une vente, il va rester enregistré comme Agent ayant fait la vente.

Module « Client »

Ce module est conçu pour effectuer la gérance des clients de l'agence. En plus de fonctionnalités CRUD, il y a les fonctionnalités suivantes :

Recherche :

- ✓ possibilité de chercher un client par son nom ou prénom.

Accès aux données :

- ✓ comme le client peut être vendeur et / ou acheteur du bien, dans ce module nous pouvons voir deux listes des ventes – Celles, dont il fait partie comme acheteur et celles, dont il fait partie comme vendeur ;
- ✓ voir toutes les offres d'un client choisi, les imprimer et supprimer ;
- ✓ voir les demandes d'un client choisi, les imprimer, supprimer et exécuter la recherche des offres appropriées;
- ✓ voir toutes les détails sur chaque vente et les imprimer.

Impression :

- ✓ il est possible d'imprimer le rapport sur chaque client avec les détails personnelles (adresse, téléphone etc.) ainsi que les détails de ses biens vendus et biens achetés et la liste des offres du client.

Contraintes :

- ✓ vu que un client fait partie des ventes et des offres, il n'est pas possible de le supprimer définitivement. Par contre, nous le rendons inactif et il ne figure plus dans aucune liste des clients actifs dans le programme. Cependant, s'il avait effectué une vente ou proposé une offre, il va rester enregistré comme faisant partie de celles-ci.

4. Evolutions

Les taches « à faire » pour ce logiciel sont comme la suite :

- ✓ Ajouter la possibilité de voir les demandes et ventes du client depuis le module « Clients » et les modifier
- ✓ Développer la fonctionnalité d'envoyer la liste des offres recherchées directement à l'adresse e-mails des clients
- ✓ Ajouter les photos aux offres pour avoir tout réuni au même endroit
- ✓ Ajouter une entité « Baux » pour pouvoir gérer les baux des clients
- ✓ Gérer les locations

Spécifications techniques

L'application « Gestion de Parc Immobilier » est une application basée sur le système de gestion de base de données. Elle utilise une interface graphique normalisée de type Windows. Les données sont gérées selon les principes usuels des bases de données relationnelles, et du langage de requêtes universel SQL.

1. Environnement

L'application fonctionne sous SO Windows.

Outils du développement / langages de programmation :

- ✓ Visual C#
- ✓ Linq
- ✓ "Windows Presentation Foundation (WPF)"
- ✓ Mvvm Light
- ✓ "ADO.NET Entity Framework (EF)"
- ✓ Base de données SQL-Server.

2. Architecture de l'application

Afin de faciliter la maintenance et de rendre plus flexible le développement, l'architecture de l'application est structurée en trois couches :

- ✓ **Présentation**, réalisée en WPF et comportant toutes les fonctions visuelles et l'interface utilisateur ;
- ✓ **Métiers**, où se trouve l'implémentation logique du programme qui communique entre les données et la présentation visuelle ;
- ✓ **Persistance**, la couche réalisée avec "ADO.NET Entity Framework (EF)", qui gère l'accès à la base de données et permet d'utiliser les données en forme d'entités.

Afin de réaliser cette architecture, j'ai utilisé le pattern de programmation « Mvvm » - Model, View, ViewModel :

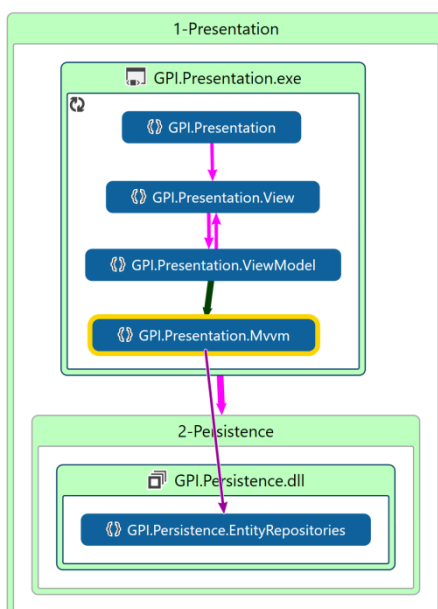


Diagramme 1. L'architecture générale

Comme ce diagramme le montre il y a deux projets principaux :

1. « **Présentation** », qui se charge des couches *Présentation* et *Métiers*
2. « **Persistance** », où sont les données du programme représentées en Entities.

Le projet GPI.Presentation comporte deux espaces des noms principaux GPI.Presentation.View avec les interfaces graphiques et GPI.Presentation.ViewModel où est concentré la logique métier du programme. La

connexion à la base de données se trouve dans une classe de base de l'espace des noms `GPI.Presentation.Mvvm`. La bibliothèque des classes `GPI.Persistance` comporte un espace des noms `GPI.Persistance.EntityRepositories` où se trouve le modèle des données `EntityFramework`.

2.1. Vue d'ensemble sur le projet `GPI.Presentation`

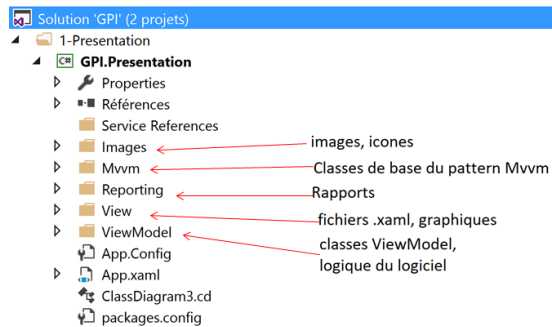


Diagramme 4. Vue d'ensemble du Projet `GPI.Presentation`.

Comme le montre le diagramme, les deux fichiers de base sont le fichier `View`, où se trouvent toutes les fenêtres graphiques et contrôles utilisateur pour assurer l'interactivité ; et le fichier `ViewModel`, qui connecte l'interface avec les données.

2.2. Vue d'ensemble sur le projet `GPI.Persistance`

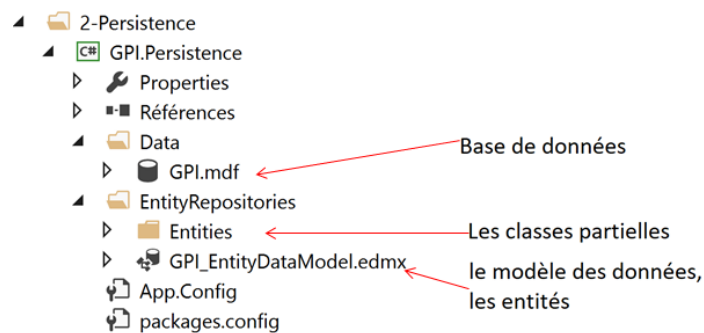


Diagramme 5. Vue d'ensemble du Projet `GPI.Persistance`.

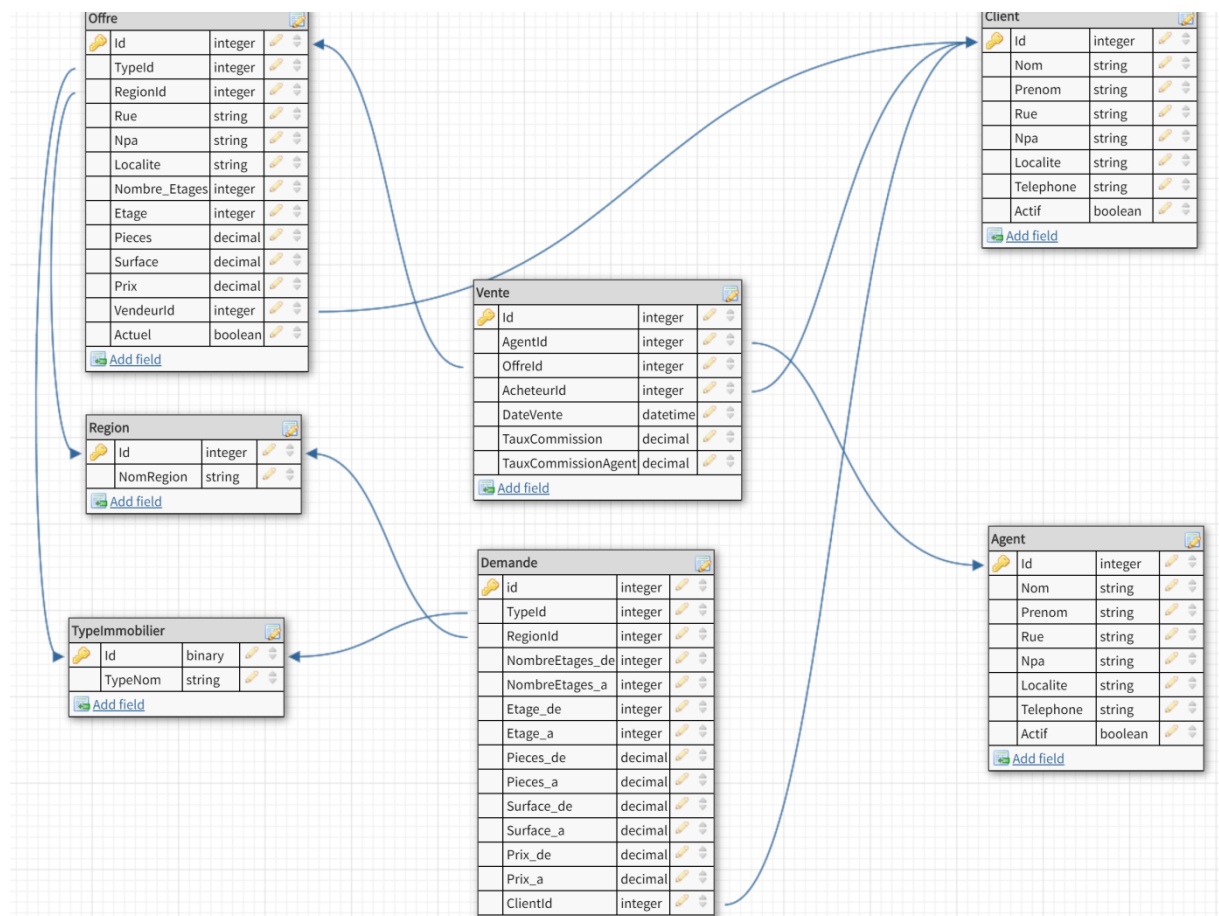
Comme le montre le diagramme, dans le projet `GPI.Persistance` se trouvent les données et le modèle des données.

3. Conception de couches de base de l'application

3.1. Couche *Persistance* et la base de données

La base de données du logiciel est une BD SQL Server relationnelle, qui contient 7 tables. Le modèle logique de celle-ci est comme la suite :

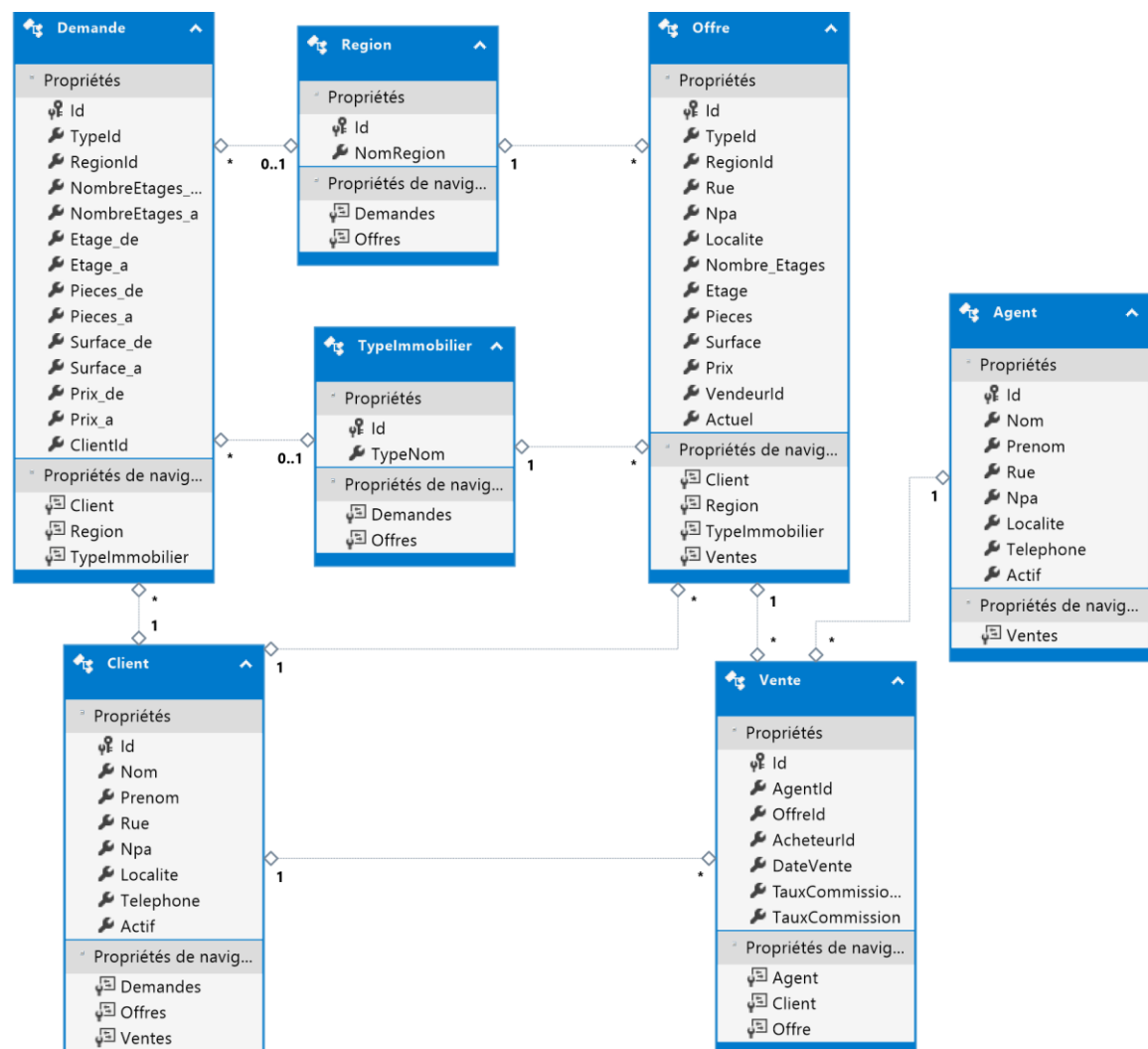
Diagramme 2. Base de données. Modèle Logique



Comme indiqué sur ce diagramme, toutes les relations de la base sont du type « un à plusieurs ». Ce schéma a défini cinq unités principales : « Offre », « Demande », « Vente », « Agent » et « Client », qui vont nous servir comme les modules du programme.

Ce modèle était tourné en chemin d'objets .edmx Entity Framework avec 7 entités, dont 5 entités principales :

Diagramme 3. Chemin d'objets



3.2. Architecture de la couche *Présentation*

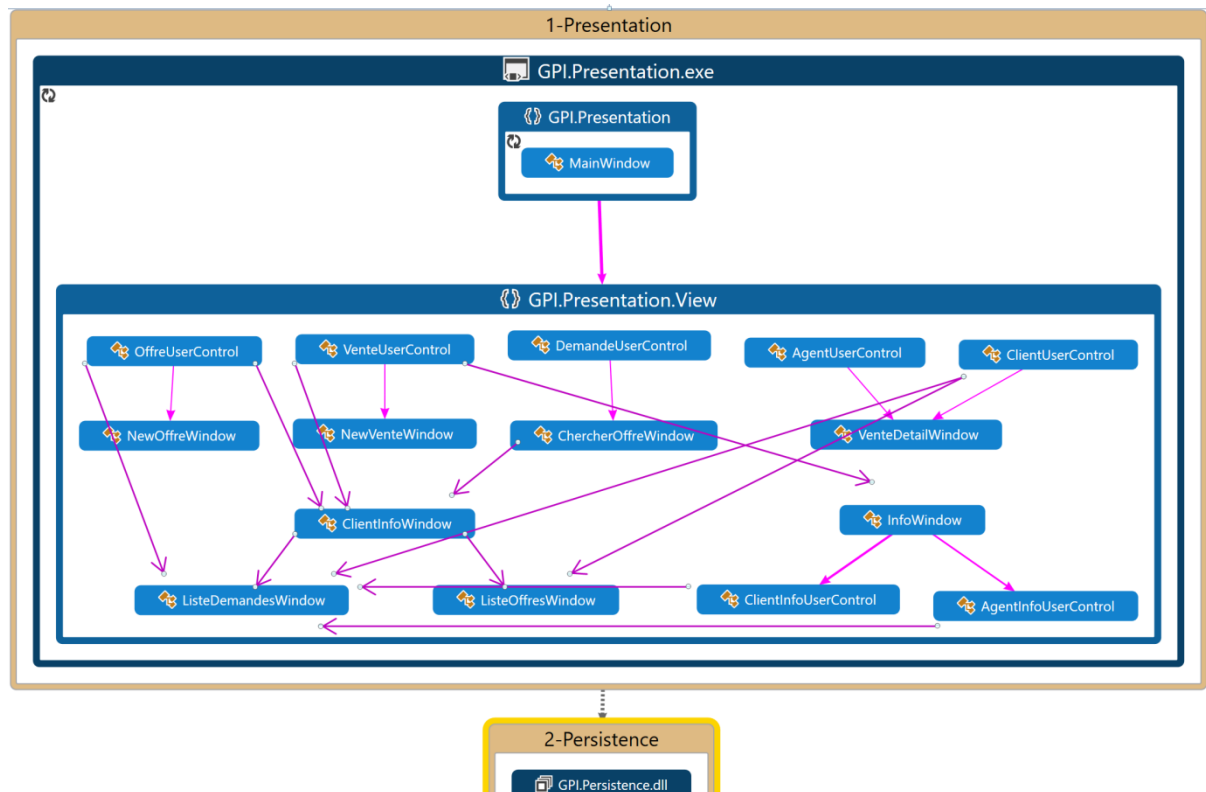
La présentation des données est organisée selon le principe des « onglets ». Dans la fenêtre principale il y a 5 onglets, qui correspondent aux modules du logiciel :



La couche Présentation est complètement présentée dans l'espace des noms GPI.Presentation.View.

Les onglets sont représentés par les contrôles utilisateurs (e.g. OffreUserControl), qui changent en fonction du click de l'utilisateur sur le Menu. Les contrôles appellent les fenêtres selon les demandes de l'utilisateur (e.g. pour créer une nouvelle offre, l'utilisateur clique un bouton sur OffreUserControl et appelle NewOffreWindow). Le diagramme suivant montre quelles fenêtres correspondent à quel contrôle utilisateur pour assurer l'utilisation de l'application la plus pratique.

Diagramme 4. Schéma des contrôles utilisateurs et fenêtres correspondantes de la couche Présentation.



3.3. Architecture de la couche Métiers

Le couche Métiers est complètement présenté dans l'espace des noms GPI.Presentation.ViewModel. Les classes de cet espace de noms servent comme DataContext aux contrôles utilisateurs correspondants.

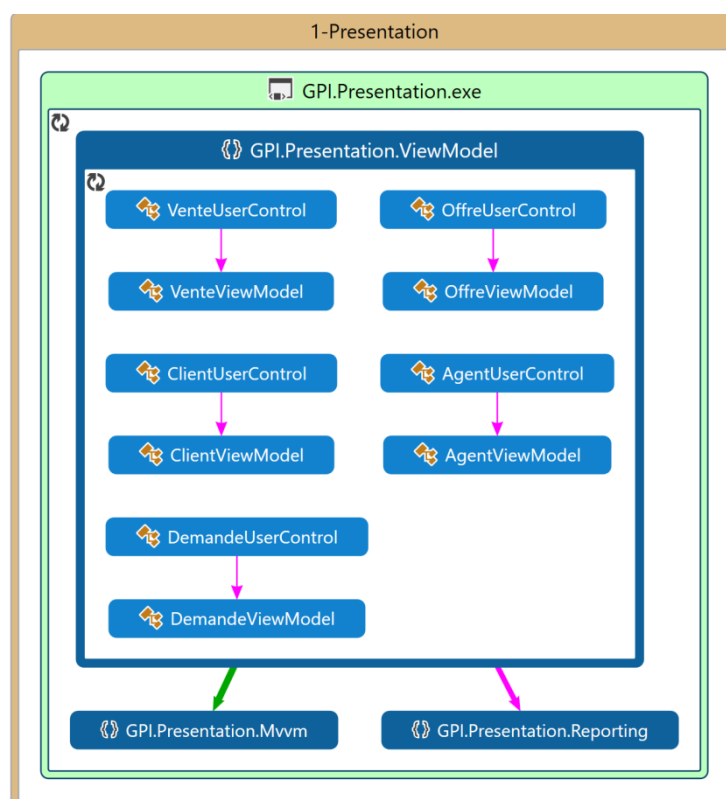


Diagramme 5. Schéma des relations de base entre View et ViewModel.

Comme le montre ce diagramme, les classes ViewModel servent comme contextes des données aux contrôles correspondants. Pour une meilleure compréhension, seuls les contrôles principaux sont inclus dans le diagramme.

4. Description technique des fonctionnalités du logiciel

4.1. Classes ViewModel

Toutes les classes ViewModel héritent de la classe ViewModelBase, qui est dans l'espace des noms GPI.Presentation.Mvvm. La connexion à la base de données est réalisé avec le pattern de la programmation *Singleton* et se trouve dans ViewModelBase, donc il n'y a qu'une seule connexion pour toutes les classes.

Toutes les classes ViewModel implémentent l'interface « ICrud », donc toutes les classes sont obligées d'avoir les méthodes pour créer, modifier, supprimer et imprimer une instance.

Diagramme 6. Schéma des classes ViewModel avec l'interface

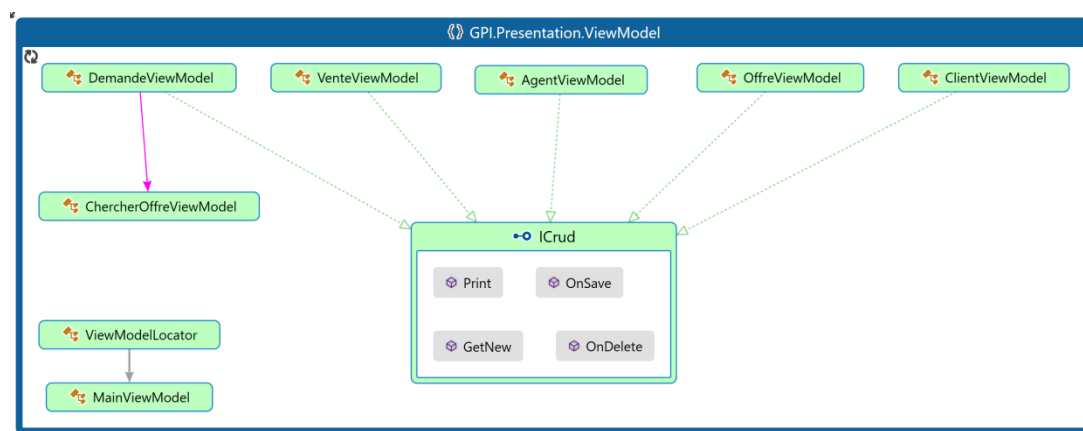
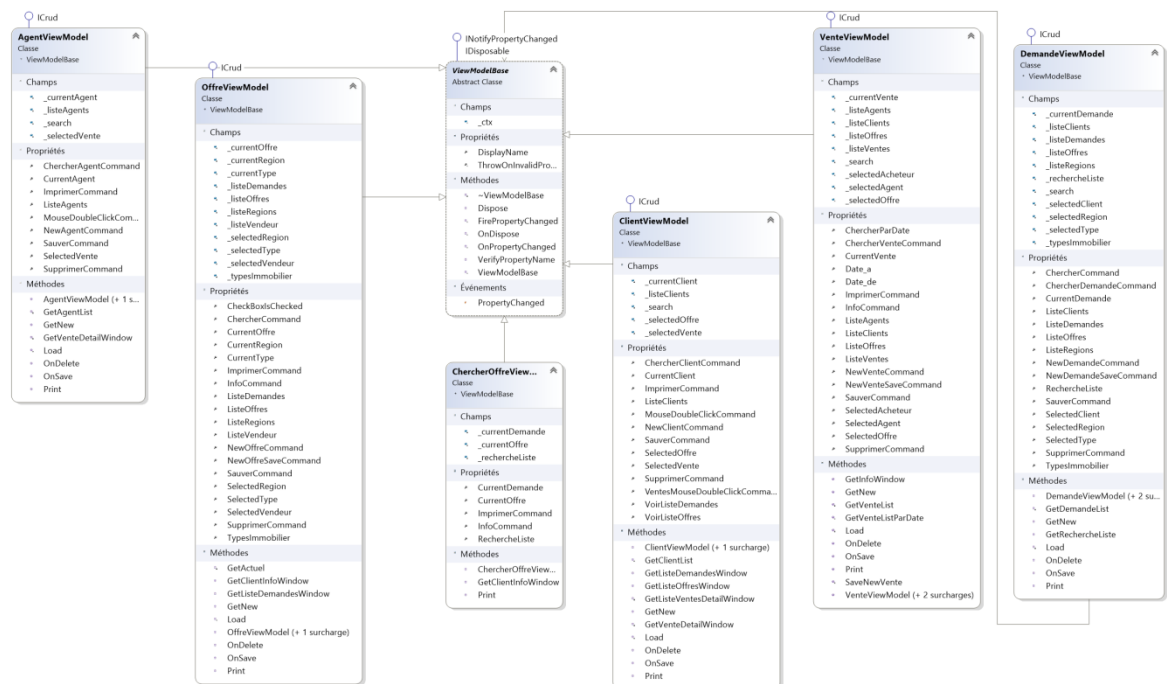


Diagramme 7. Diagramme de classes UML



Comme indiqué dans le diagramme UML, toutes les classes ont les méthodes exigées par l'interface ICrud : GetNew – pour créer une nouvelle instance, OnDelete – pour

supprimer une instance, OnSave – pour enregistrer les modifications et Print – pour imprimer le rapport sur chaque instance.

Ces méthodes et d'autres correspondent aux commandes, qui sont appelées par l'utilisateur depuis l'interface graphique. Presque toutes les actions de l'utilisateur sont réalisées dans les ViewModels en utilisant les commandes *RelayCommand*, qui sont fournies par le package NuGet *MvvmLight*. Seulement dans le Code Behind de MainWindow je réalise le changement des onglets par les événements de click sur les boutons.

En plus de la connexion à la base de données, dans la Classe ViewModelBase il existe une méthode *FirePropertyChanged*, qui permet de mettre à jour les listes des instances et des instances elles-mêmes, après que des changements aient été effectués.

4.2. Spécifications techniques du Module « Offre »

Quand l'application s'ouvre, l'onglet « Offre » est affiché.

Gestion de Parc Immobilier

✓ Offre Demande Vente Agent Client

Choisissez la région :

Choisissez le bien :

Appartement 3.5 pcs
Maison 7.5 pcs
Maison 5.0 pcs
Appartement 3.0 pcs
Appartement 4.5 pcs
Duplex 3.5 pcs
Appartement 3.5 pcs
Duplex 5.0 pcs
Maison 6.5 pcs
Appartement 2.5 pcs
Maison 3.5 pcs
Appartement 2.5 pcs
Appartement 2.5 pcs
Appartement 5.0 pcs
Duplex 6.5 pcs
Maison 8.0 pcs
Attique 3.5 pcs
Appartement 2.5 pcs
Appartement 5.0 pcs
Duplex 5.5 pcs
Appartement 0.0 pcs

Type de bien : Surface : Nbr de pieces :

Etage : Nbr d'étages : Prix:

Rue : NPA :

Localité : Région : Actuel :

Vendeur :

Gestion de Parc Immobilier - 2017. Par Olga Fragnière

Le nom d'une offre est composé du nom du Type de bien+ Nbr de pièces + « pcs ».

Depuis la recherche d'une offre il est possible de choisir une région et/ou un type d'immobilier dans les menus déroulants. Ces actions vont trier et diminuer la liste selon la région et/ou le type choisis.

Cette fonctionnalités est réalisé dans les propriétés Current Region et CurrentType. La recherche est effectuée par une requête LINQ. Chaque fois je vérifie, si l'utilisateur a choisi ou laisser vide une région ou un type de bien, car le résultat doit être différent si un des critères de la recherche est rempli ou non :

```
public Region CurrentRegion
{
    get { return _currentRegion; }
    set {
        _currentRegion = value;
        if (CurrentType == null)
        {
            ListOffres = new
ObservableCollection<Offre>(_ctx.Offres.Where(o => o.Region.Id == CurrentRegion.Id));
        }
        else
        {
            ListOffres = new
ObservableCollection<Offre>(_ctx.Offres.Where(o => o.Region.Id == CurrentRegion.Id &&
o.TypeImmobilier.Id == CurrentType.Id));
        }
        FirePropertyChanged("Offres");
        FirePropertyChanged();
    }
}

public TypeImmobilier CurrentType
{
    get { return _currentType; }
    set {
        _currentType = value;
        if (CurrentRegion == null)
        {
            ListOffres = new
ObservableCollection<Offre>(_ctx.Offres.Where(o => o.TypeImmobilier.Id ==
CurrentType.Id));
        }
        else
        {
            ListOffres = new
ObservableCollection<Offre>(_ctx.Offres.Where(o => o.Region.Id == CurrentRegion.Id &&
o.TypeImmobilier.Id == CurrentType.Id));
        }
        FirePropertyChanged("Offres");
        FirePropertyChanged();
    }
}
```

Depuis ce module il est possible de voir toutes les demandes des clients, qui correspondent à une offre choisie en cliquant sur le bouton de la recherche. Comme presque tous les champs de la demande peuvent être *Null*, il faut vérifier ceci et si c'est vrai, toutes les offres correspondent au critère de la recherche. Si l'utilisateur n'a pas choisi une offre, il n'y a rien à afficher, donc une fenêtre d'avertissement va s'afficher. La liste des demandes recherchées, quant à elle, va apparaitre dans une nouvelle fenêtre.

```

public void GetListeDemandesWindow()
{
    if (CurrentOffre.Id != 0)
    {
        ListeDemandes = new
ObservableCollection<Demande>(_ctx.Demandes.Where
(d => ((d.Prix_de == null ? CurrentOffre.Prix >= 0 :
d.Prix_de >= CurrentOffre.Prix) && (d.Prix_a == null ? CurrentOffre.Prix >= 0 :
CurrentOffre.Prix <= d.Prix_a)) &&
(d.RegionId == null ? CurrentOffre.RegionId > 0 :
CurrentOffre.RegionId == d.RegionId) &&
(d.TypeId == null ? CurrentOffre.TypeId > 0 :
d.TypeId == CurrentOffre.TypeId) &&
((d.Etage_de == null ? CurrentOffre.Etage >= 0 :
CurrentOffre.Etage >= d.Etage_de) && (d.Etage_a == null ? CurrentOffre.Etage >= 0 :
CurrentOffre.Etage <= d.Etage_a)) &&
((d.NombreEtages_de == null ?
CurrentOffre.Nombre_Etages >= 0 : CurrentOffre.Nombre_Etages >= d.NombreEtages_de) &&
d.NombreEtages_a == null ? CurrentOffre.Nombre_Etages >= 0 :
CurrentOffre.Nombre_Etages <= d.NombreEtages_a) &&
((d.Surface_de == null ? CurrentOffre.Surface > 0 :
CurrentOffre.Surface >= d.Surface_de) && d.Surface_a == null ? CurrentOffre.Surface >
0 : CurrentOffre.Surface <= d.Surface_a) &&
((d.Pieces_de == null ? CurrentOffre.Pieces > 0 :
CurrentOffre.Pieces >= d.Pieces_de) && (d.Pieces_a == null ? CurrentOffre.Pieces > 0 :
CurrentOffre.Pieces <= d.Pieces_a)))));
        ListeDemandesWindow win = new
ListeDemandesWindow(ListeDemandes);
        win.Owner = Application.Current.MainWindow;
        win.ShowDialog();
    }
    else
    {
        MessageBox.Show("D'abord choisissez une offre");
    }
}

```

4.3. Spécifications techniques du Module « Demande »

Le module « Demande » c'est le deuxième onglet de la fenêtre principale. Ici l'utilisateur peut poser sa demande et rechercher les offres correspondantes.

Il n'est pas obligatoire de remplir tous les champs, si un ou plusieurs paramètres n'ont pas d'importance pour le client, la recherche sera effectuée quand même. Les paramètres de la recherche peuvent être les suivants :

- **Région** – choisir dans la liste déroulante
- **Type de bien** - choisir dans la liste déroulante
- **Prix de...à** – insérer avec le clavier
- **Pièces de...à** – insérer avec le clavier
- **Surface de...à** – insérer avec le clavier
- **Etage de...à** – insérer avec le clavier

L'algorithme de la recherche est le contraire de celui de la recherche des demandes par une offre donnée. Nous devons également vérifier, que le paramètre de la recherche ne soit pas Null. De plus, les offres recherchées doivent être actuelles :

```

public void GetRechercheListe()
{
    try
    {
        if (CurrentDemande.Id != 0)
        {
            RechercheListe = new
ObservableCollection<Offre>(_ctx.Offres.Where
(o => ((CurrentDemande.Prix_de == null ?
o.Prix > 0 : o.Prix >= CurrentDemande.Prix_de) && (CurrentDemande.Prix_a == null ?
o.Prix > 0 : o.Prix <= CurrentDemande.Prix_a)) &&
(CurrentDemande.RegionId == null ? o.RegionId
> 0 : o.RegionId == CurrentDemande.RegionId) &&
(CurrentDemande.TypeId == null ? o.TypeId > 0
: o.TypeId == CurrentDemande.TypeId) &&
((CurrentDemande.Etage_de == null ? o.Etage >=
0 : o.Etage >= CurrentDemande.Etage_de) && (CurrentDemande.Etage_a == null ? o.Etage
>= 0 : o.Etage <= CurrentDemande.Etage_a)) &&
((CurrentDemande.NombreEtages_de == null ?
o.Nombre_Etages >= 0 : o.Nombre_Etages >= CurrentDemande.NombreEtages_de) &&
CurrentDemande.NombreEtages_a == null ? o.Nombre_Etages >= 0 : o.Nombre_Etages <=
CurrentDemande.NombreEtages_a) &&
((CurrentDemande.Surface_de == null ?
o.Surface > 0 : o.Surface >= CurrentDemande.Surface_de) && CurrentDemande.Surface_a ==
null ? o.Surface > 0 : o.Surface <= CurrentDemande.Surface_a) &&
((CurrentDemande.Pieces_de == null ? o.Pieces
> 0 : o.Pieces >= CurrentDemande.Pieces_de) && (CurrentDemande.Pieces_a == null ?
o.Pieces > 0 : o.Pieces <= CurrentDemande.Pieces_a && o.Actuel == 1)))));

            var vm = new ChercherOffreViewModel(RechercheListe);
            ChercherOffreWindow win = new
ChercherOffreWindow(RechercheListe);
            win.DataContext = vm;
            win.Owner = Application.Current.MainWindow;
            win.ShowDialog();
        }
        else
            MessageBox.Show("D'abord choisissez une demande");
    }
    catch
    {
        MessageBox.Show("D'abord choisissez une demande");
    }
}

```

Pour chercher les offres il faut cliquer sur le bouton de la recherche. Les résultats vont apparaitre dans une nouvelle fenêtre.

Il est aussi possible de chercher une demande par le nom du client. Cette fonctionnalité est effectuée avec une commande paramétrée (voir 4.5. Spécifications *techniques du Module « Agent »*).

4.4. Spécifications techniques du Module « Vente »

Le nom de la vente est composé du nom de l'acheteur « / » le nom du vendeur. Dans ce module il y a une barre de recherche afin de faciliter la recherche d'une vente. Il suffit d'entrer un nom ou quelques lettres du nom d'un acteur de la vente (acheteur ou vendeur) et la liste des ventes va se mettre à jour en fonction de la recherche. Cette fonctionnalité est effectuée avec une commande paramétrée (voir 4.5. Spécifications *techniques du Module « Agent »*).

Quand la nouvelle vente est créée, l'offre vendue devient inactuelle, donc changer sa valeur en 0. En revanche, quand la vente est supprimée, l'offre correspondante devient actuelle, donc 1.

```
private void SaveNewVente()
{
    if (CurrentVente.Id == 0)
    {
        _ctx.Ventes.Add(CurrentVente);
        SelectedOffre.Actuel = 0;
        FirePropertyChanged("SelectedOffre");
    }
    try
    {
        _ctx.SaveChanges();
        MessageBox.Show("La nouvelle vente a été enregistrée");
    }
}
```

```

    }
    catch (Exception)
    {
        MessageBox.Show("Erreur lors de l'enregistrement de la
vente ");
    }

    FirePropertyChanged("CurrentVente");
    FirePropertyChanged("ListeVentes");
    Load();
}

```

En création d'une nouvelle vente nous devons remplir le taux de commission de courtage de l'agence et de l'agent en pourcentage. La commission de l'agence est basée sur un pourcentage du prix du bien, alors que la commission de l'agent est en fonction du pourcentage de la commission de l'agence.

```

public decimal ComissionAgence
{
    get
    { return OffrePrix * TauxCommission / 100; }
    set{}
}
public decimal ComissionAgent
{
    get { return ComissionAgence * TauxCommissionAgent / 100; }
    set { }
}

```

4.5. Spécifications techniques du Module « Agent »

Dans ce module il est possible de voir toutes les ventes de l'agent choisi avec la commission gagnée sur chaque vente et la commission totale. En cliquant deux fois sur la vente, l'utilisateur va appeler la fenêtre avec tous les détails de la vente.

Calcul de la commission totale :

```
public decimal ComissionTotale
{
    get
    {
        decimal total = 0;
        foreach (Vente v in Ventes)
        {
            total += v.ComissionAgent;
        }
        return total;
    }
    set { }
}
```

Dans ce module il y a une barre de recherche afin de faciliter la recherche d'un agent. Il suffit d'entrer un nom ou quelques lettres du nom de l'agent et la liste des agents va se mettre à jour en fonction de la recherche. Cette fonctionnalité est effectuée avec une commande paramétrée.

Nous établissons le paramètre de la recherche dans le code XAML. Le paramètre c'est le texte entré par l'utilisateur :

```
<TextBox Width="175" Background="FloralWhite" x:Name="ChercheTextBox"
Margin="2,2,0,2">
</TextBox>
```

```

        <Button Height="25" Width="25" Margin="0,2,2,2"
Background="FloralWhite" Command="{Binding ChercherAgentCommand}"
CommandParameter="{Binding ElementName=ChercheTextBox, Path=Text}">
        <Image Source="/Images/searchIcon2.png"></Image>
    </Button>

```

Dans le AgentViewModel nous avons la commande paramétrée avec le « string » :

```

private RelayCommand<String> _search;

public RelayCommand<string> ChercherAgentCommand
{
    get { return _search; }
    set { _search = value; }
}

```

Le constructeur :

```

    ChercherAgentCommand = new RelayCommand<string>(GetAgentList);

```

La méthode de la recherche :

```

private void GetAgentList(string str)
{
    if (str != "")
        ListeAgents = new
ObservableCollection<Agent>(_ctx.Agents.Where(v => (v.Nom.Contains(str.ToLower()) ||
v.Prenom.Contains(str.ToLower())) && v.Actif == 1));
    else
        Load();
}

```

Vu qu'un agent fait partie des ventes, il n'est pas possible de le supprimer définitivement depuis la base de données. Par contre, en cliquant sur le bouton « Supprimer » nous le rendons inactif et il ne figure plus dans aucune liste des agents actifs du programme. Donc, s'il avait effectué une vente, il va rester toujours enregistré comme ayant effectué celles-ci (voir 4.6. *Spécifications techniques du Module « Client »*).

4.6. Spécifications techniques du Module « Client »

Gestion de Parc Immobilier

Offre Demande Vente Agent **✓ Client**

Chercher le Client :

Demandes **Offres**

Raymond Gassman
Pierre Luthi
Anthony Bustamente
Vanessa Coquard

Nom : **Prenom:**

Rue : **NPA :**

Localité : **Téléphone :**

Biens achetés :

Nom du bien	Prix de bien	Date de Vente	
Duplex 0.0 pcs	0.00	18.06.2017	
Appartement 4.5 pcs	495000.00	23.07.2016	

Biens Vendus :

Nom du bien	Prix de bien	Date de Vente	
Appartement 3.0 pcs	315000.00	24.05.2016	

Gestion de Parc Immobilier - 2017. Par Olga Fragnière

Dans ce module il y a une barre de recherche afin de faciliter la recherche d'un client. Il suffit d'entrer un nom ou quelques lettres du nom de client et la liste des clients va se mettre à jour en fonction de la recherche. Cette fonctionnalité est effectuée avec une commande paramétrée (voir 4.5. Spécifications techniques du Module « Agent »).

En cliquant deux fois sur la vente, l'utilisateur va appeler la fenêtre avec tous les détails de la vente.

En cliquant sur le bouton « Supprimer », l'utilisateur va recevoir le message d'avertissement, car en supprimant un client vous allez également supprimer toutes ses demandes. Vu que un client fait partie des ventes et des offres, il n'est pas possible de le supprimer définitivement depuis la base de données. En revanche, en cliquant sur le bouton « Supprimer » nous le rendons inactif et il ne figure plus dans aucune liste des clients actifs dans le programme. Donc, s'il avait effectué une vente ou proposé une offre, il va rester enregistré comme le faisant partie de celles-ci. Ceci est fait pour sécuriser la suppression de données importantes ainsi que pour garder l'intégrité et la cohérence de la base de données.

La méthode de la suppression :

```

public void OnDelete()
{
    System.Windows.MessageBoxResult result = MessageBox.Show("Le
demandes du Client seront supprimées. \n Voulez-vous supprimer ce Client?",
"Question", MessageBoxButton.YesNo, MessageBoxImage.Warning);

    if (result == MessageBoxResult.Yes)
    {
        try
        {
            CurrentClient.Actif = 0;
            foreach (Demande d in _ctx.Demandes)
            {
                if (d.ClientId == CurrentClient.Id)
                { _ctx.Demandes.Remove(d); }
            }

            _ctx.SaveChanges();
        }
        catch (Exception e)
        {
            MessageBox.Show("Impossible de supprimer!" + e);
        }
    }
    FirePropertyChanged("ListeClients");
    FirePropertyChanged("ListeDemandes");
    Load();
}

```

Comme le Client peut être le vendeur et l'acheteur, mais dans le tableau vente, le client figure seulement comme un acheteur, j'ai ajouté une collection des ventes, dont le client fait partie comme vendeur. Pour l'afficher dans la DataGrid il fallait créer cette collection dans la classe partielle Client (non pas dans le ViewModel) :

```

private ICollection<Offre> _listeVentes;
public ICollection<Offre> ListeVentes
{
    get{

        _listeVentes = new HashSet<Offre>();
        foreach (Offre o in Offres)
        {
            foreach (Vente v in o.Ventes)
            {
                if (v.Offre.VendeurId == Id)
                { _listeVentes.Add(o); }
            }
        }
        return _listeVentes;
    }
    set { }
}

```

Manuel Utilisateur

Bienvenu dans GPI – Gestion de Parc Immobilier. L'application va s'ouvrir directement sur onglet « Offres »


1. Onglet « Offres »

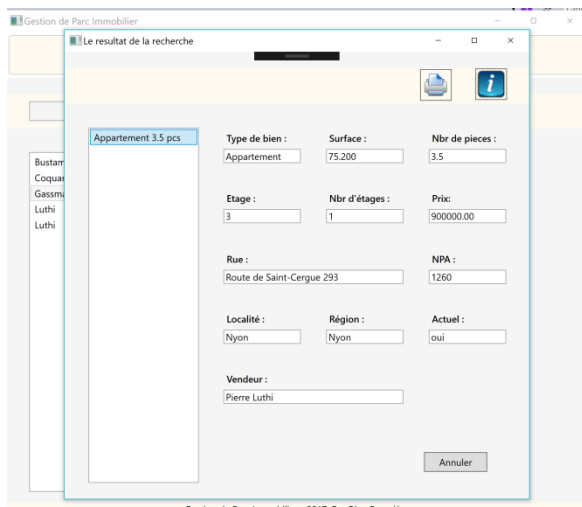
- Pour imprimer les détails : choisissez l'offre dans la liste et appuyez sur le bouton « Imprimer »
- Pour voir les détails sur le vendeur de l'offre : choisissez l'offre dans la liste et appuyez sur le bouton « Information »

2. Onglet « Demandes »



Utilisation Generale:

- Pour chercher la demande d'un client : entrer le nom du Client ou quelques lettres du nom dans le bar de recherche et appuyez sur le bouton « Chercher »
- Pour supprimer : choisissez la demande dans la liste et appuyez sur le bouton « Supprimer »
- Pour ajouter une nouvelle offre appuyez sur le bouton « Ajouter »
- Sauvegardez en appuyant sur le bouton « Sauvegarder »
- Pour chercher les offres correspondantes à la demande , sauvegarder la demande et appuyez

sur le bouton « Chercher » 




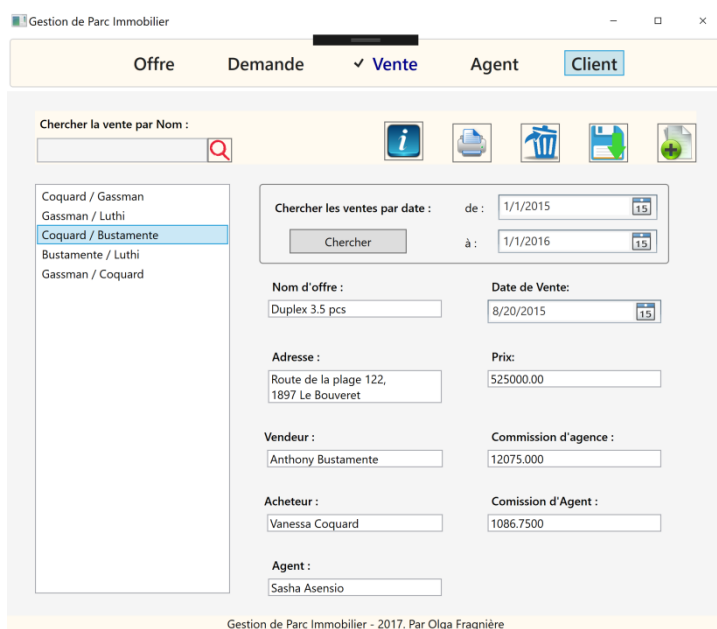
Vous allez voir la fenêtre contenant les résultats de la recherche :

- Choisissez une offre dans la liste à gauche pour voir les détails
- Pour imprimer les résultats de la recherche : choisissez l'offre dans la liste et appuyez sur le bouton « Imprimer » 
- Pour voir les détails sur le vendeur de l'offre : choisissez l'offre dans la liste et appuyez sur le bouton « Information » 


3. Onglet « Vente »


Utilisation Generale:


- Pour chercher la vente: entrer le nom de l'acheteur ou de vendeur, ou quelques lettres du nom dans la barre de recherche et appuyez sur le bouton « Chercher » 




• Pour chercher la vente en fonction de la date de vente : choisissez la date de départ et la date de la fin dans les calendriers, appuyez sur le bouton « Chercher »


• Pour voir les détails sur le vendeur, acheteur et agent: choisissez la vente dans la liste et appuyez sur le bouton « Information » 

• Pour imprimer détails de la vente : choisissez la vente dans la liste et appuyez sur le bouton « Imprimer » 

• Pour supprimer : choisissez la vente dans la liste et appuyez sur le bouton « Supprimer » 

• Pour ajouter une nouvelle vente

appuyez sur le bouton « Ajouter » 

- Sauvegardez en appuyant sur le bouton « Sauvegarder » 

Nouvelle Vente

Offre : Appartement 2.5 pcs Surface : 42.000 Nbr de pieces : 2.5

Prix : 305000.00 Commission % : 0 Commission d'agent % : 0

Adresse : Chemin de la Source 3, 1854 Laysin

Vendeur : Pierre Luthi Date de Vente : 6/25/2017

Acheteur : Agent :

Annuler Enregistrer

Gestion de Parc Immobilier - 2017. Par Olga Fragnière

Ajout d'une nouvelle Vente :

- Choisissez l'offre à vendre dans la liste déroulante. Les informations sur cette offre vont se remplir automatiquement.
- Remplissez le reste des champs et appuyez sur « Enregistrer » pour enregistrer la vente.

4. Onglet « Agent »

Agent

Chercher l'Agent :

Sasha Asensio
Margo Dutti
Nicolas Ratier
Natacha Rogner

Nom : Ratier Prenom : Nicolas

Rue : Avenue de Beaulieu 19 NPA : 1014

Localité : Lausanne Téléphone : +41794556798

Ventes effectuées :

Bien vendu	Prix de bien	Commission
Appartement 4.5 pcs	495000.00	1237.5000
Maison 6.5 pcs	1050000.00	3570.0000

4807.5000

Gestion de Parc Immobilier - 2017. Par Olga Fragnière

Utilisation générale :

- Choisissez un agent dans la liste à gauche pour voir les détails
- Pour chercher l'agent: entrer le nom de l'agent ou quelques lettres du nom dans le bar de recherche et appuyez sur le bouton « Chercher »
- Pour voir les détails de la vente de l'agent : appuyez deux fois sur la vente de l'agent
- Pour supprimer : choisissez l'agent dans la liste et appuyez sur le bouton « Supprimer »
- Pour ajouter un nouvel agent appuyez sur le bouton « Ajouter »
- Sauvegardez en appuyant sur le

bouton « Sauvegarder »

- Pour imprimer détails de l'agent: choisissez l'agent dans la liste et appuyez sur le bouton « Imprimer »

5. Onglet « Client »

Gestion de Parc Immobilier

Offre Demande Vente Agent **Client**

Chercher le Client :

Demandes Offres

Raymond Gassman
Pierre Luthi
Anthony Bustamente
Vanessa Coquard

Nom : Bustamente Prénom : Anthony

Rue : Rue du Valentin 45 NPA : 1014

Localité : Lausanne Téléphone : +41783779087

Biens achetés :

Nom du bien	Prix de bien	Date de Vente
Maison 6.5 pcs	1050000.00	15.06.2017

Biens Vendus :

Nom du bien	Prix de bien	Date de Vente
Duplex 3.5 pcs	525000.00	20.08.2015

Gestion de Parc Immobilier - 2017. Par Olga Fragnière

Utilisation générale :

- Pour chercher le client: entrer le nom du client ou quelques lettres du nom dans le bar de recherche et appuyez sur le bouton « Chercher »

- Voir les demandes du client : choisissez un client et appuyer sur le bouton « Demandes »

- Voir les offres du client : choisissez un client et appuyer sur le bouton « Offres »

- Pour voir les détails des achats et des ventes du client : appuyez deux fois sur la vente ou achat du client

- Pour supprimer : choisissez un client dans la liste et appuyez sur le

bouton « Supprimer »

- Pour ajouter un nouvel client appuyez sur le bouton « Ajouter »
- Sauvegardez en appuyant sur le bouton « Sauvegarder »
- Pour imprimer détails du client: choisissez l'agent dans la liste et appuyez sur le bouton «

Imprimer »