



# PROJET PERSONNEL

SGBD – Juin 2016

[Résumé](#)

Projet personnel réalisé en C# avec MSSQL Server et LinQ

Michaël Defraene  
michael.defraene@condorcet.be

## Table des matières

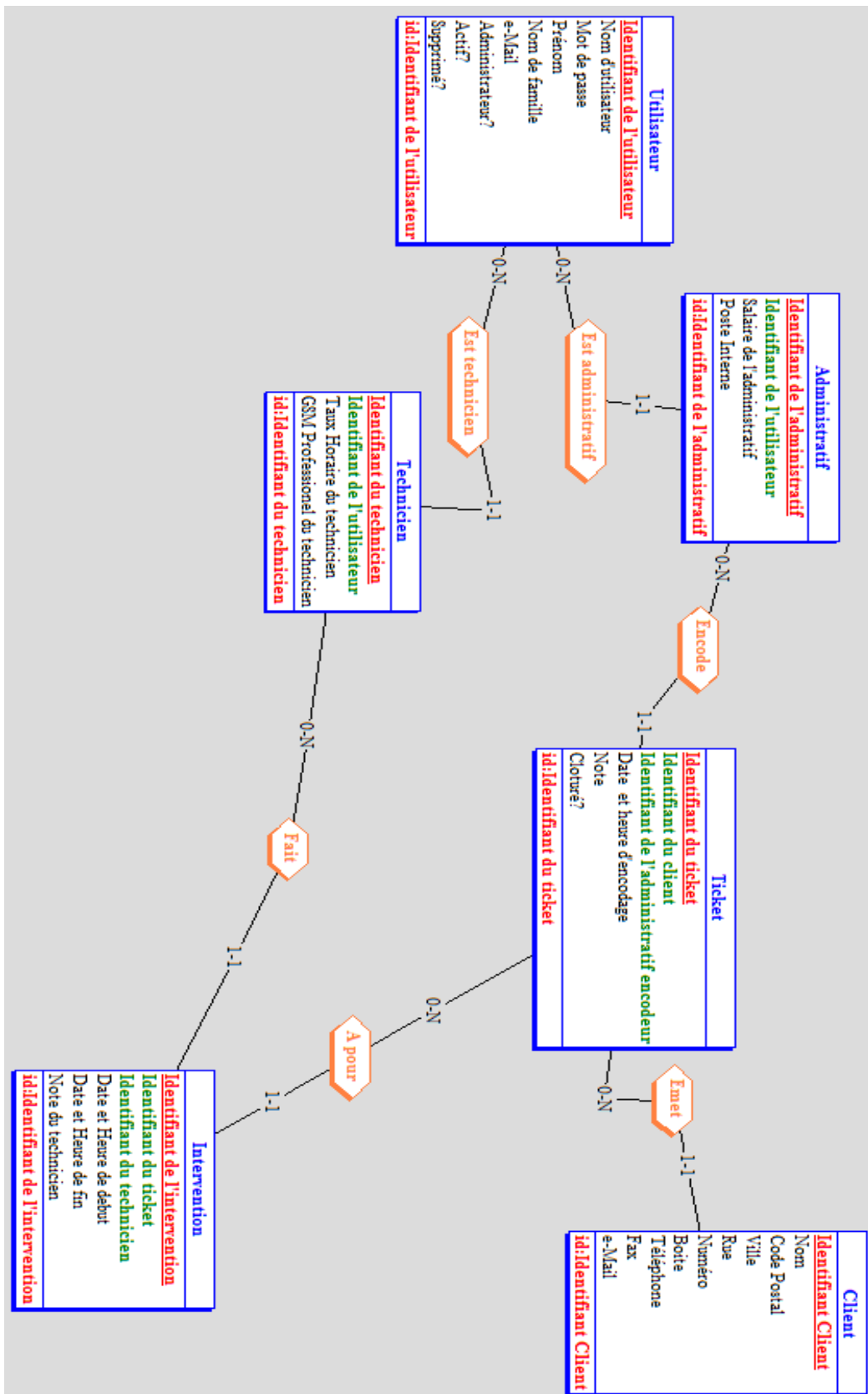
1. Enoncé. ....	3
2. Schéma conceptuel.....	4
3. Diagramme des Use-Cases.....	5
4. Diagramme de classe.....	6
5. Ecrans + TOE.....	7
5.1. Ecran « WPF_Login.xaml ».....	7
5.1.1. Screenshot de l'écran.....	7
5.1.2. Utilité de l'écran.....	7
5.1.3. TOE.....	7
5.2. Ecran « MainWindow.xaml ».....	8
5.2.1. Screenshot de l'écran.....	8
5.2.2. Utilité de l'écran.....	8
5.2.3. TOE.....	8
5.3. UserControl « UC_Administrative.xaml ».....	9
5.3.1. Screenshot de l'UC.....	9
5.3.2. Utilité de l'UC.....	9
5.3.3. TOE.....	9
5.4. UserControl « UC_Technical.xaml ».....	11
5.4.1. Screenshot de l'UC.....	11
5.4.2. Utilité de l'UC.....	11
5.4.3. TOE.....	11
5.5. Ecran « WPF_AddRMA.xaml ».....	12
5.5.1. Screenshot de l'écran.....	12
5.5.2. Utilité de l'écran.....	12
5.5.3. TOE.....	12
5.6. Ecran « WPF_ModifRMA.xaml ».....	14
5.6.1. Screenshot de l'écran.....	14
5.6.2. Utilité de l'écran.....	14
5.6.3. TOE.....	14
5.7. Ecran « WPF_SearchRMA.xaml ».....	16
5.7.1. Screenshot de l'écran.....	16
5.7.2. Utilité de l'écran.....	16
5.7.3. TOE.....	16
5.8. Ecran « WPF_AddInterv.xaml ».....	18
5.8.1. Screenshot de l'écran.....	18
5.8.2. Utilité de l'écran.....	18
5.8.3. TOE.....	18
5.9. Ecran « WPF_ModifInterv.xaml ».....	20

5.9.1.	Screenshot de l'écran.....	20
5.9.2.	Utilité de l'écran. ....	20
5.9.3.	TOE.....	20
5.10.	Ecran « WPF_MySettings.xaml ».....	22
5.10.1.	Screenshot de l'écran.....	22
5.10.2.	Utilité de l'écran. ....	22
5.10.3.	TOE.....	22
5.11.	Ecran « WPF_SearchClient.xaml ».....	23
5.11.1.	Screenshot de l'écran.....	23
5.11.2.	Utilité de l'écran. ....	23
5.11.3.	TOE.....	23
5.12.	Ecran « WPF_AddClient.xaml ».....	24
5.12.1.	Screenshot de l'écran.....	24
5.12.2.	Utilité de l'écran. ....	24
5.12.3.	TOE.....	24
5.13.	Ecran « WPF_ModifClient.xaml ».....	25
5.13.1.	Screenshot de l'écran.....	25
5.13.2.	Utilité de l'écran. ....	25
5.13.3.	TOE.....	25
6.	Nouveauté.....	26
6.1.	La DataGridView gérée par une DataTable pour éviter les liens directs avec la DB.....	26
6.2.	Le timer permettant de recharger une DataTable toutes les x secondes. ....	27
6.3.	Les UserControls permettant d'avoir le même écran principal mais avec des composants différents selon le type d'utilisateur.....	28

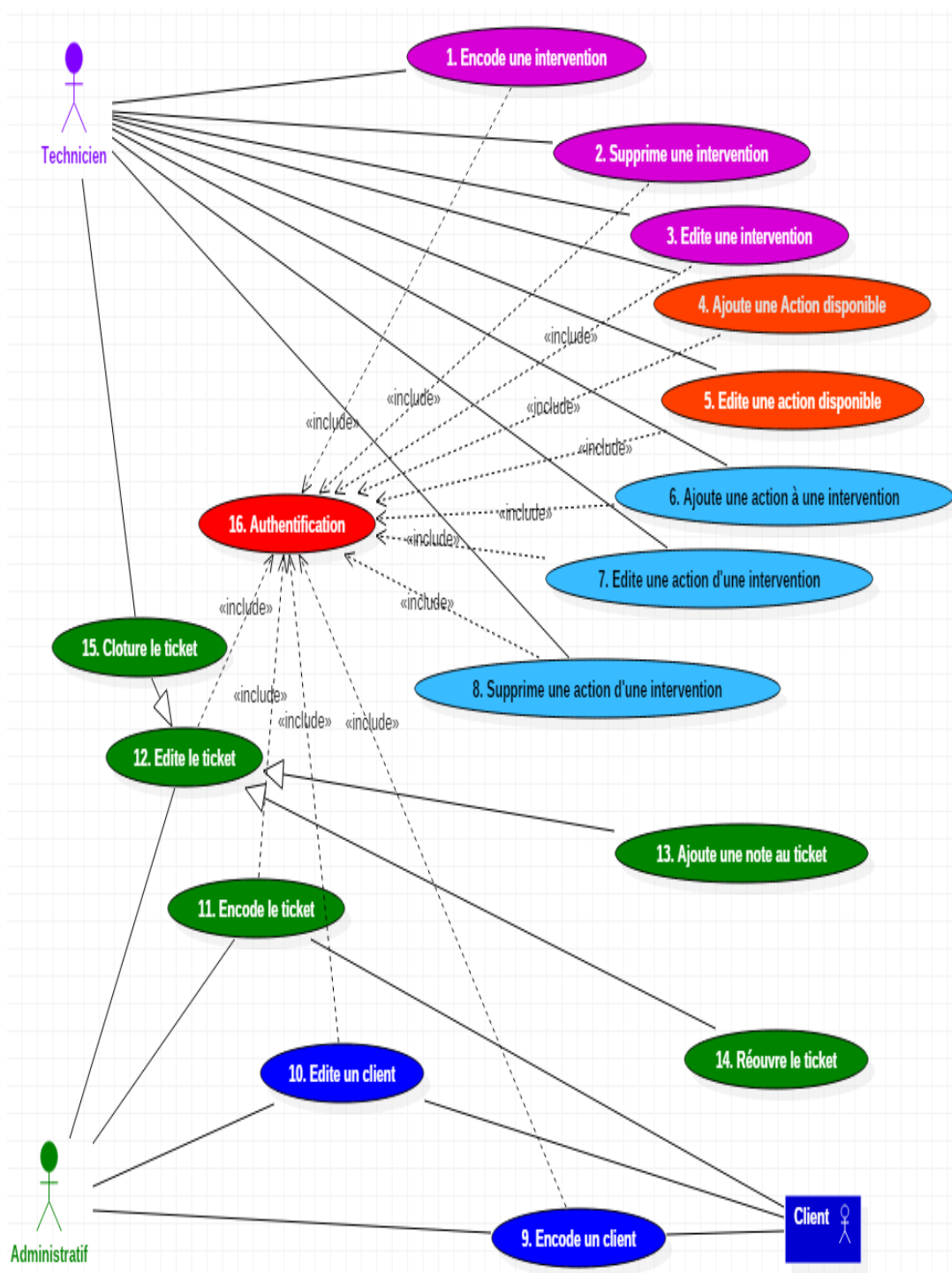
## 1. Enoncé.

Mon projet est une application de gestion des tickets (demande faites par un client) d'un service informatique. L'administratif pourra gérer les différents clients, ainsi qu'encoder les nouveaux tickets et ouvrir ou fermer les tickets déjà existants. Le technicien pourra encoder ses interventions et le temps que cela lui aura pris.

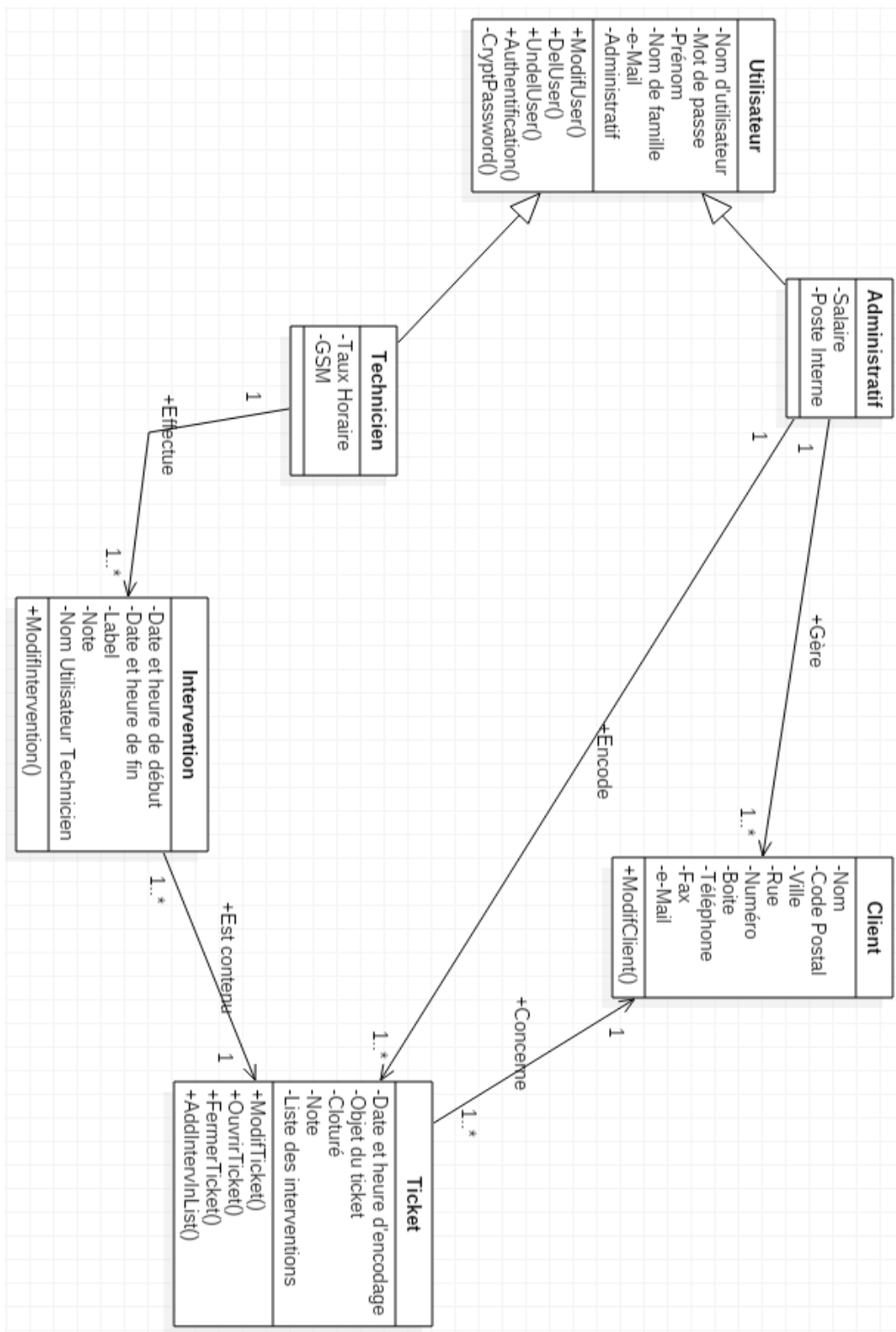
## 2. Schéma conceptuel.



### 3. Diagramme des Use-Cases.



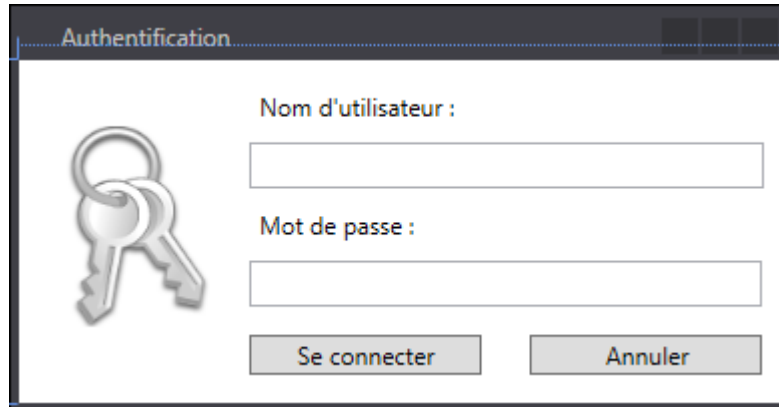
#### 4. Diagramme de classe.



## 5. Ecrans + TOE.

### 5.1. Ecran « WPF\_Login.xaml ».

#### 5.1.1. Screenshot de l'écran.



#### 5.1.2. Utilité de l'écran.

Cet écran permet à l'utilisateur de s'authentifier dans l'application.

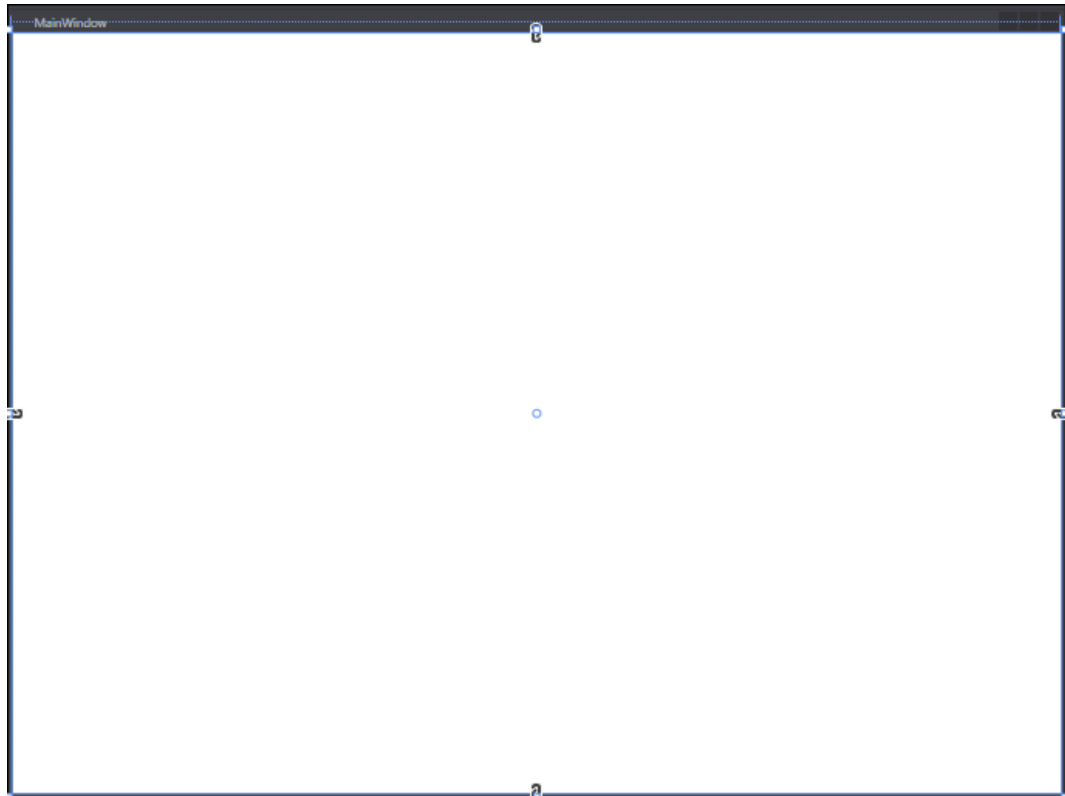
#### 5.1.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Récupérer le nom d'utilisateur entré par l'utilisateur.	TXT_Username	/
Vérifier que le nom d'utilisateur et le mot de passe sont bien rempli pour pouvoir activer le bouton « Se connecter ».	TXT_Username	TextChanged
Récupérer le mot de passe entré par l'utilisateur.	PWD_Password	/
Vérifier que le nom d'utilisateur et le mot de passe sont bien rempli pour pouvoir activer le bouton « Se connecter ».	PWD_Password	PasswordChanged
Authentifier l'utilisateur et fermer la fenêtre si l'authentification s'est bien passée.	BTN_Connection	Click
Fermer la fenêtre et donner les valeurs négatives pour que le main ferme le programme.	BTN_Cancel	Click
Initialise les composants et remet à zéro les différents champs.	WPF_Login	Load



## 5.2. Ecran « MainWindow.xaml ».

### 5.2.1. Screenshot de l'écran.



### 5.2.2. Utilité de l'écran.

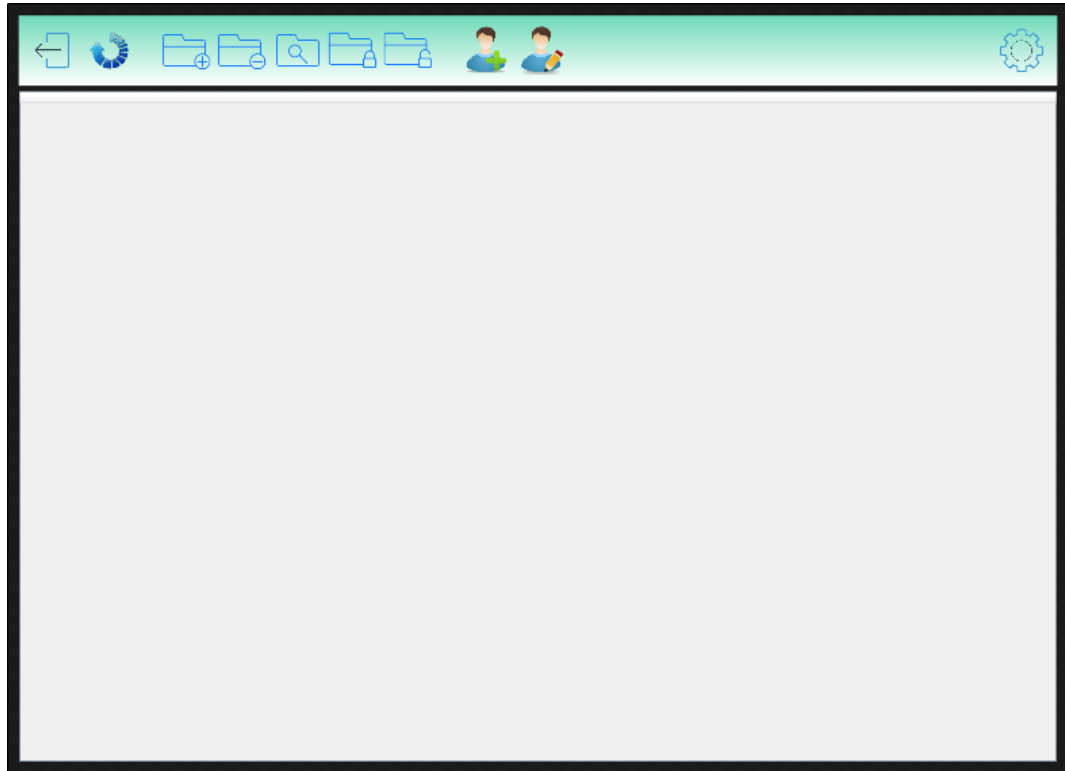
C'est l'écran principal du programme. Il est aussi celui qui accueillera les différents UserControls.

### 5.2.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
<ul style="list-style-type: none"> <li>• Prépare et affiche la fenêtre de login.</li> <li>• Vérifie que le login s'est bien passé, si cela n'est pas le cas, ferme l'application.</li> <li>• Si le login s'est bien passé prépare les différents UserControls de la Grid et les ajoute aux enfants de la Grid.</li> <li>• Affiche le bon UserControl par rapport au type d'utilisateur qui s'est logué.</li> </ul>	MainWindow	Load

### 5.3. UserControl « UC\_Administrative.xaml ».

#### 5.3.1. Screenshot de l'UC.



#### 5.3.2. Utilité de l'UC.

Cet UC est l'écran de base qu'aura l'administratif, il comporte son menu général et la grille contenant les tickets ouverts.

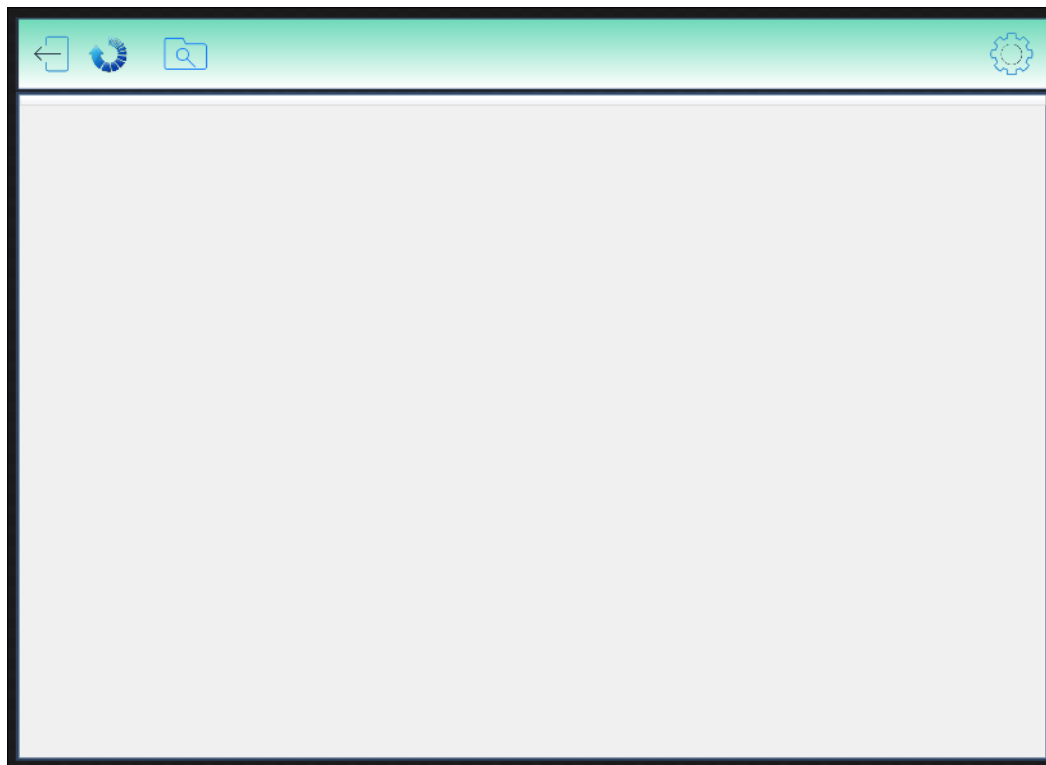
#### 5.3.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Permet de quitter l'application.	BTN_Exit	Click
Redémarre l'application totalement.	BTN_Restart	Click
Prépare et ouvre la fenêtre d'ajout de ticket.	BTN_RMAdd	Click
Demande la confirmation de suppression du ticket sélectionner. (Celui-ci ne doit pas contenir d'interventions). Si on répond oui, il le supprime, si non, il ne fait rien.	BTN_RMARemove	Click
Ouvre la fenêtre de recherche de ticket.	BTN_RMASearch	Click
Ouvre la fenêtre de fermeture du ticket	BTN_RMAClose	Click

Ouvre la fenêtre de réouverture du ticket.	BTN_RMAREopen	Click
Ouvre la fenêtre de mes paramètres.	BTN_MySettings	Click
Ouvre la fenêtre d'ajout d'un client.	BTN_ClientAdd	Click
Ouvre la fenêtre de modification d'un client.	BTN_ClientMod	Click
Affiche la liste des tickets ouverts.	DGV_RMAList	/
Ouvre le ticket sélectionner pour modification.	DGV_RMAList	MouseDoubleClick
<ul style="list-style-type: none"> <li>• Remet à zéro tous les composants</li> <li>• Remplis la DataGrid avec les tickets ouverts.</li> </ul>	UC_Administrative	Load

## 5.4. UserControl « UC\_Technical.xaml ».

### 5.4.1. Screenshot de l'UC.



### 5.4.2. Utilité de l'UC.

Cet UC est l'écran de base qu'aura le technicien, il comporte son menu général et la grille contenant les tickets ouverts.

### 5.4.3. TOE.


<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Permet de quitter l'application.	BTN_Exit	Click
Redémarre l'application totalement.	BTN_Restart	Click
Ouvre la fenêtre de recherche de ticket.	BTN_RMASearch	Click
Ouvre la fenêtre de mes paramètres.	BTN_MySettings	Click
Affiche la liste des tickets ouverts.	DGV_RMAList	/
Ouvre le ticket sélectionner ajout d'une intervention	DGV_RMAList	MouseDoubleClick
<ul style="list-style-type: none"> <li>• Remet à zéro tous les composants</li> <li>• Remplis la DataGrid avec les tickets ouverts.</li> </ul>	UC_Technical	Load

## 5.5. Ecran « WPF AddRMA.xaml ».

### 5.5.1. Screenshot de l'écran.

Formulaire d'ajout de ticket

Informations client :

Nom :  

Code Postal :  Ville :  Rue :

Num Maison :  Boîte :  Tel :  Fax :

e-Mail :

Objet :

Note :

Annuler Remise à Zéro Encoder

### 5.5.2. Utilité de l'écran.

Cet écran permet l'encodage d'un nouveau ticket et si nécessaire offre la possibilité d'ajouter un nouveau client.

### 5.5.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Affiche le nom du client.	TXT_ClientName	/
Affiche l'écran de sélection du client.	TXT_ClientName	MouseDoubleClick
Affiche le code postal du client.	TXT_ClientZipCode	/
Affiche le nom de la ville du client.	TXT_ClientCity	/
Affiche le nom de la rue du client.	TXT_ClientStreet	/
Affiche le numéro de rue du client.	TXT_ClientStreetNB	/
Affiche la boîte de rue du client.	TXT_ClientStreetBox	/
Affiche le numéro de téléphone du client	TXT_ClientTel	/
Affiche le numéro de fax du client.	TXT_ClientFax	/
Affiche l'adresse e-mail du client.	TXT_ClientMail	/
Affiche l'objet du ticket énoncé par le client.	TXT_RMABject	/
Affiche la note que l'administratif va rentrer (souvent des commentaires du client).	TXT_RMANote	/
Ouvre la fenêtre d'ajout d'un nouveau client.	BTN_AddClient	Click

Encode le ticket dans la base de données et ferme la fenêtre.	BTN_Accept	Click
Remet tout le formulaire à zéro.	BTN_RAZ	Click
Remet tout le formulaire à zéro et ferme la fenêtre.	BTN_Cancel	Click
Remet le formulaire à zéro.	WPF_AddRMA.xaml	Load

## 5.6. Ecran « WPF ModifRMA.xaml ».

### 5.6.1. Screenshot de l'écran.

Formulaire de modification du ticket

Informations client :

Nom :

Code Postal :  Ville :  Rue :

Num Maison :  Boîte :  Tel :  Fax :

e-Mail :

Objet :

Note :

### 5.6.2. Utilité de l'écran.

Cet écran permet la modification d'un ticket existant. Si on est administratif on peut changer la note et visionner les interventions. Si on est technicien on peut gérer les interventions du ticket.

### 5.6.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Affiche le nom du client.	TXT_ClientName	/
Affiche le code postal du client.	TXT_ClientZipCode	/
Affiche le nom de la ville du client.	TXT_ClientCity	/
Affiche le nom de la rue du client.	TXT_ClientStreet	/
Affiche le numéro de rue du client.	TXT_ClientStreetNB	/
Affiche la boîte de rue du client.	TXT_ClientStreetBox	/
Affiche le numéro de téléphone du client	TXT_ClientTel	/
Affiche le numéro de fax du client.	TXT_ClientFax	/
Affiche l'adresse e-mail du client.	TXT_ClientMail	/

Affiche l'objet du ticket énoncé par le client.	TXT_RMABject	/
Affiche la note que l'administratif va rentrer (souvent des commentaires du client).	TXT_RMANote	/
Affiche la liste des interventions du ticket.	DGV_InterventionList	/
Ouvre la fenêtre de modification d'intervention.	DGV_InterventionList	MouseDoubleClick
Ouvre la fenêtre d'ajout d'intervention	BTN_AddInterv	Click
Demande une confirmation de suppression de l'intervention sélectionnée. Si oui, il la supprime, sinon il ne fait rien	BTN_RemoveInterv	Click
Encode le ticket dans la base de données et ferme la fenêtre.	BTN_Accept	Click
Remet tout le formulaire à zéro et ferme la fenêtre.	BTN_Cancel	Click
Remet le formulaire à zéro. Et charge la liste des interventions.	WPF_ModifRMA.xaml	Load



## 5.7. Ecran « WPF SearchRMA.xaml ».

### 5.7.1. Screenshot de l'écran.

### 5.7.2. Utilité de l'écran.

Cet écran permet la recherche de ticket selon le nom du client ou le numéro de ticket. Il permet la recherche dans les tickets fermés.

### 5.7.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Ouvre le champ de recherche par le numéro de ticket, ferme le champ de recherche par le nom de client et remet à zéro ce dernier	RDB_NumRMA	Checked
Récupère le texte du numéro de ticket recherché.	TXT_NumRMA	/
Affiche les résultats de la recherche dans la DataGrid.	TXT_NumRMA	TextChanged
Ouvre le champ de recherche par le nom de client, ferme le champ de recherche par le numéro de ticket et remet à zéro ce dernier.	RDB_NomClient	Checked
Récupère le texte du nom de client recherché.	TXT_NomClient	/
Affiche les résultats de la recherche dans la DataGrid	TXT_NomClient	TextChanged

S'il est coché fait la recherche dans tous les tickets (Fermés et ouverts). S'il n'est pas coché fait la recherche dans les tickets ouverts.	CHK_ClosedRMA	Checked
S'il est coché fait la recherche dans tous les tickets (Fermés et ouverts). S'il n'est pas coché fait la recherche dans les tickets ouverts.	CHK_ClosedRMA	Unchecked
Récupère le résultat de la recherche	DGV_Search	/
Sélectionne et renvoie le client sélectionné.	DGV_Search	MouseDoubleClick
Remet à zéro le formulaire et ferme ce dernier.	BTN_Close	Click
Remet les champs à vide et affiche la liste des tickets ouverts dans la DataGrid	WPF_SearchRMA	Load

## 5.8. Ecran « WPF AddInterv.xaml ».

### 5.8.1. Screenshot de l'écran.

### 5.8.2. Utilité de l'écran.

Cet écran permet l'ajout d'intervention sur le ticket sélectionné

### 5.8.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Récupère le numéro du ticket en cours.	TXT_RMANumber	/
Sélectionner la date de début de l'intervention	DTP_DateBeg	/
Récupère l'heure de début de l'intervention	TXT_HeureBeg	/
Vérifie que l'heure est un entier compris entre 0 et 23	TXT_HeureBeg	LostFocus
Récupère les minutes de l'heure de début de l'intervention.	TXT_MinuteBeg	/
Vérifie que les minutes est un entier compris entre 0 et 59	TXT_MinuteBeg	LostFocus
Sélectionner la date de fin de l'intervention	DTP_DateFin	/
Récupère l'heure de fin de l'intervention	TXT_HeureFin	/
Vérifie que l'heure est un entier compris entre 0 et 23	TXT_HeureFin	LostFocus
Récupère les minutes de l'heure de fin de l'intervention.	TXT_MinuteFin	/
Vérifie que les minutes est un entier compris entre 0 et 59	TXT_MinuteFin	LostFocus
Récupère l'objet de l'intervention.	TXT_Object	/
Récupère la note du technicien.	TXT_Note	/
Encode l'intervention dans la base de données et ferme la fenêtre.	BTN_Accept	Click
Remet à zéro le formulaire	BTN_RAZ	Click

Remet à zéro le formulaire et ferme la fenêtre	BTN_Cancel	Click
Remet le formulaire à zéro	WPF_AddInterv	Load

## 5.9. Ecran « WPF ModifInterv.xaml ».

### 5.9.1. Screenshot de l'écran.

Formulaire de modification d'intervention

Numéro du ticket :

Date de début :   Heure de début :  h

Date de fin :   Heure de fin :  h

Objet :

Note :

### 5.9.2. Utilité de l'écran.

Cet écran permet la modification d'une intervention donnée sur un ticket donné.

### 5.9.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Récupère le numéro du ticket en cours.	TXT_RMNumber	/
Sélectionner la date de début de l'intervention	DTP_DateBeg	/
Récupère l'heure de début de l'intervention	TXT_HeureBeg	/
Vérifie que l'heure est un entier compris entre 0 et 23	TXT_HeureBeg	LostFocus
Récupère les minutes de l'heure de début de l'intervention.	TXT_MinuteBeg	/
Vérifie que les minutes est un entier compris entre 0 et 59	TXT_MinuteBeg	LostFocus
Sélectionner la date de fin de l'intervention	DTP_DateFin	/
Récupère l'heure de fin de l'intervention	TXT_HeureFin	/
Vérifie que l'heure est un entier compris entre 0 et 23	TXT_HeureFin	LostFocus
Récupère les minutes de l'heure de fin de l'intervention.	TXT_MinuteFin	/
Vérifie que les minutes est un entier compris entre 0 et 59	TXT_MinuteFin	LostFocus
Récupère l'objet de l'intervention.	TXT_Object	/
Récupère la note du technicien.	TXT_Note	/
Encode les modifications de l'intervention dans la base de données et ferme la fenêtre.	BTN_Accept	Click

Remet à zéro le formulaire et ferme la fenêtre	BTN_Cancel	Click
Remet à zéro le formulaire	WPF_ModifInter	Load

## 5.10. Ecran « WPF MySettings.xaml ».

### 5.10.1. Screenshot de l'écran.

### 5.10.2. Utilité de l'écran.

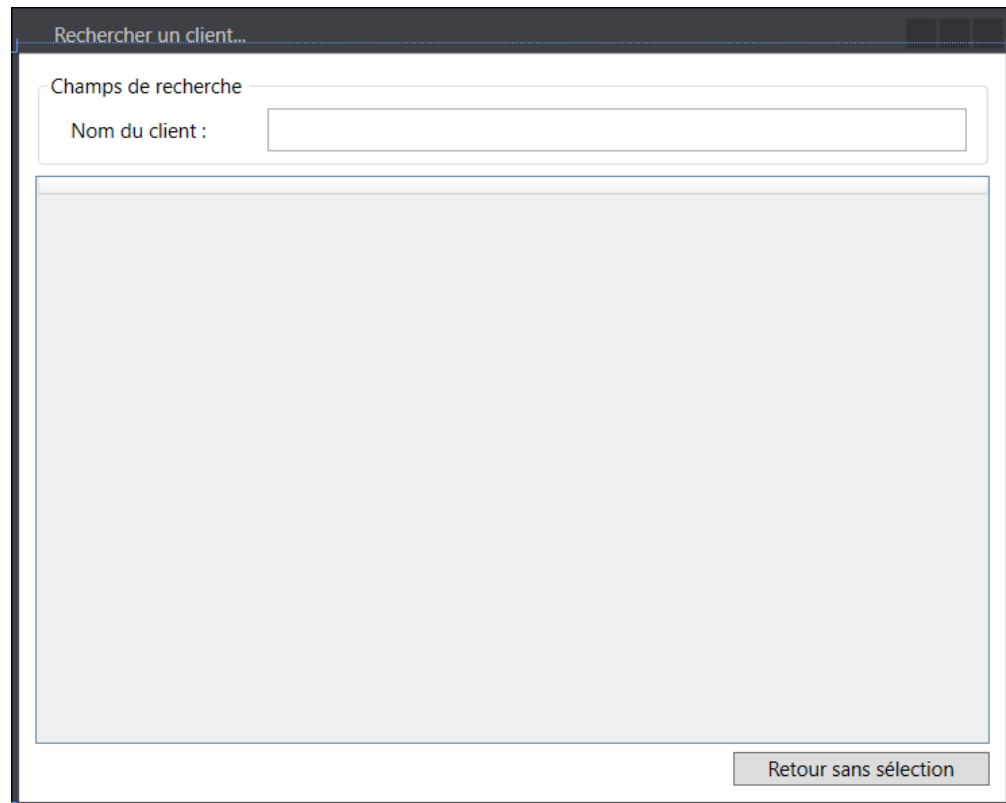
Cet écran permet à l'utilisateur de modifier son mot de passe.

### 5.10.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Récupère le mot de passe actuel entré par l'utilisateur	PWD_OldPWD	/
Vérifie que le mot de passe est le même que celui de l'objet utilisateur	PWD_OldPWD	TextChanged
Récupère le nouveau mot de passe entré	PWD_NewPWD	/
Vérifie que les 2 mots de passes sont les mêmes	PWD_NewPWD	TextChanged
Récupère le nouveau mot de passe entré en second	PWD_NewPWD1	/
Vérifie que les 2 mots de passes sont les mêmes	PWD_NewPWD1	TextChanged
Modifie le mot de passe de l'objet et l'encode dans la base de données et ferme la fenêtre	BTN_Accept	Click
Remet le formulaire à zéro et ferme la fenêtre	BTN_Cancel	Click
Remet le formulaire à zéro	WPF_MySettings	Load

## 5.11. Ecran « WPF SearchClient.xaml ».

### 5.11.1. Screenshot de l'écran.



### 5.11.2. Utilité de l'écran.

Cet écran permet la recherche de client suivant le nom du client.

### 5.11.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Récupère le texte de recherche sur le nom entré par l'utilisateur	TXT_ClientName	/
Affine la recherche et met à jour la datagrid	TXT_ClientName	TextChanged
Récupère les résultats de la recherche	DGV_Client	/
Transmet le client et ferme la fenêtre	DGV_Client	MouseDoubleClick
Remet à zéro le formulaire et le ferme sans choisir de client	BTN_Close	Click
Met à zéro le formulaire et charge la datagrid avec tous les clients	WPF_SearchClient	Load



## 5.12. Ecran « WPF AddClient.xaml »

### 5.12.1. Screenshot de l'écran.

Formulaire d'ajout de client...

Nom :

Code Postal :  Ville :  Rue :  Num. :  Boite :

Téléphone :  Fax :  e-Mail :

Annuler Remise à Zéro Encoder

### 5.12.2. Utilité de l'écran.

Cet écran permet l'encodage des nouveaux clients.

### 5.12.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Récupère le nom du client	TXT_Name	/
Vérifie que le nom du client n'existe pas déjà dans la base de données et prévient s'il est déjà existant	TXT_Name	LostFocus
Récupère le code postal du client	TXT_ZipCode	/
Vérifie que le code postal est de type entier.	TXT_ZipCode	TextChanged
Récupère la ville du client	TXT_City	/
Récupère la rue du client	TXT_Street	/
Récupère le numéro de maison du client	TXT_StreetNB	/
Vérifie que le numéro de maison est de type entier.	TXT_StreetNB	TextChanged
Récupère le numéro de boîte de la maison du client	TXT_StreetBox	/
Récupère le numéro de téléphone du client	TXT_Telephone	/
Récupère le numéro de fax du client	TXT_Fax	/
Récupère l'email du client	TXT_eMail	/
Encode le client dans la base de données	BTN_Accept	Click
Remet le formulaire à zéro	BTN_RAZ	Click
Remet le formulaire à zéro et le referme	BTN_Cancel	Click
Remet à zéro le formulaire	WPF_AddClient	Load

### 5.13. Ecran « WPF ModifClient.xaml ».

#### 5.13.1. Screenshot de l'écran.

Formulaire de modification de client...

Nom :

Code Postal :  Ville :  Rue :  Num. :  Boite :

Téléphone :  Fax :  e-Mail :

Annuler Remise à Zéro Encoder

#### 5.13.2. Utilité de l'écran.

Cet écran permet la modification d'un client déjà existant.

#### 5.13.3. TOE.

<u>Tâche</u>	<u>Objet</u>	<u>Evènement</u>
Ouvre le formulaire de recherche de client	TXT_Name	MouseDoubleClick
Affiche le nom du client sélectionné	TXT_Name	/
Récupère le code postal du client	TXT_ZipCode	/
Récupère la ville du client	TXT_City	/
Récupère la rue du client	TXT_Street	/
Récupère le numéro de maison du client	TXT_StreetNB	/
Récupère le numéro de boîte de la maison du client	TXT_StreetBox	/
Récupère le numéro de téléphone du client	TXT_Telephone	/
Récupère le numéro de fax du client	TXT_Fax	/
Récupère l'email du client	TXT_eMail	/
Encode les modifications du client dans la base de données	BTN_Accept	Click
Remet le formulaire à zéro	BTN_RAZ	Click
Remet le formulaire à zéro et le referme	BTN_Cancel	Click
Remet le formulaire à zéro	WPF_ModifClient	Load

## 6. Nouveauté.

### 6.1. La DataGridView gérée par une DataTable pour éviter les liens directs avec la DB.

La DataGridView peut être gérée par plusieurs méthodes. La première est celle du DataBindind, elle concerne en une liaison directe avec la base de données (par une vue ou une table). Celle que j'utilise est le fait que je charge les données depuis la base de données dans une DataTable et puis je charge la DataGridView avec la DataTable. Cela me permet de sélectionner la ligne et la modifier plus facilement sans directement impacter la base de données.

Un petit exemple de code (UC\_Administrative – Chargement de la DataTable et de la DataGridView) :

```
// Préparation de la DataTable
dtDGV = new DataTable();
dtDGV.Columns.Add("Numéro");
dtDGV.Columns.Add("Date d'encodage");
dtDGV.Columns.Add("Nom du client");
dtDGV.Columns.Add("Sujet");
this.DGV_RMAList.ItemsSource = dtDGV.AsDataView();

// Préparation de la DataTable
this.dt = new DataTable();
this.dt = Class.C_Database.RetrieveOpenedTickets(); // Récupération depuis la DB
```

```
private void FillDataGridView()
{
    // Définition des variables locales
    object[] rowArray = new object[4];
    DataRow dr;

    // Vide la table
    dtDGV.Rows.Clear();

    // Remplis la table
    foreach (DataRow r in dt.Rows)
    {
        // Création de la nouvelle ligne
        dr = dtDGV.NewRow();

        // Préparation des champs
        rowArray[0] = r.ItemArray[0];
        rowArray[1] = r.ItemArray[1];
        rowArray[2] = r.ItemArray[2];
        rowArray[3] = r.ItemArray[3];

        // Ajout des champs à la nouvelle ligne
        dr.ItemArray = rowArray;

        // Ajout de la nouvelle ligne à la DataTable
        dtDGV.Rows.Add(dr);
    }
}
```

## 6.2. Le timer permettant de recharger une DataTable toutes les x secondes.

Le timer permet d'effectuer une tâche toutes les x millisecondes. Dans mon cas il est utilisé pour recharger les DataGridView nécessitant une mise à jour permanente.

Un petit exemple de code (UC\_Administrative – Le timer permettant de recharger toutes les millisecondes la datatable et la datagrid)

```
DispatcherTimer timer;
```

```
// Création du timer
timer = new DispatcherTimer(); // Instanciation du timer
timer.Tick += new EventHandler(timer_Tick); // Abonnement à la méthode pour le Tick
timer.Interval = new TimeSpan(0, 1, 0); // Définition du temps d'attente entre les Ticks
timer.Start();
```

```
private void timer_Tick(object sender, EventArgs e)
{
    try
    {
        // Récupération des tickets
        this.dt = Class.C_Database.RetrieveOpenedTickets();

        // Remplissage de la DataGridView
        FillDataGridView();
    }
    catch (Exception ex)
    {
        // Affichage du message d'erreur
        MessageBox.Show("Le programme a rencontré une ou plusieurs erreur(s) : \n\n" + ex.Message);
    }
}
```

### 6.3. Les UserControls permettant d'avoir le même écran principal mais avec des composants différents selon le type d'utilisateur.

Les UserControls me permettent d'avoir une seule fenêtre et plusieurs contrôles qui diffèrent selon le type d'utilisateur. Les deux UserControls sont quasiment les mêmes à quelques différences près dans le menu.

```

Class.C_Administrative administrative = null;
Class.C_Technical technical = null;

0 references
public MainWindow()
{
    // Initialise les composants
    InitializeComponent();

    // Préparation de la fenêtre de Login
    XAML.WPF_Login wpf_login = new XAML.WPF_Login();
    wpf_login.ShowDialog();

    // Vérification si le login s'est bien passé
    if (wpf_login.LoginState)
    {
        // Vérification du type d'utilisateur
        if (wpf_login.UserType == "Technical")
        {
            // Instancie un technicien
            technical = Class.C_Database.SelectTechnical(wpf_login.Username);

            // Instancie le UserControl du technicien et l'affiche
            UserControls.UC_Technical uc_tech = new UserControls.UC_Technical(technical);
            GRD_uc.Children.Clear();
            GRD_uc.Children.Add(uc_tech);
        }
        else
        {
            // Instancie un administratif
            administrative = Class.C_Database.SelectAdministrative(wpf_login.Username);

            // Instancie le UserControl de l'administratif et l'affiche
            UserControls.UC_Administrative uc_adm = new UserControls.UC_Administrative(administrative);
            GRD_uc.Children.Clear();
            GRD_uc.Children.Add(uc_adm);
        }
    }
    else
    {
        // Ferme l'application
        Application.Current.Shutdown();
    }
}

```