|  |  |
| --- | --- |
| **Compte rendu STAGE**  ***Année 2023-2024*** | **SEMAINE 4**  ***30/01/2023 – 03/01/2023*** |

***Sommaire / Summary***

Table des matières

[1. Présentation de l’entreprise. 3](#_Toc126325279)

[2. Mes missions. 4](#_Toc126325280)

[2.1. Gestionnaire de licences. 4](#_Toc126325281)

[3. Outils utilisés. 5](#_Toc126325282)

[4. Conclusion. 6](#_Toc126325283)

[5. Niko-Niko. 7](#_Toc126325284)

[6. Annexe codes. 8](#_Toc126325285)

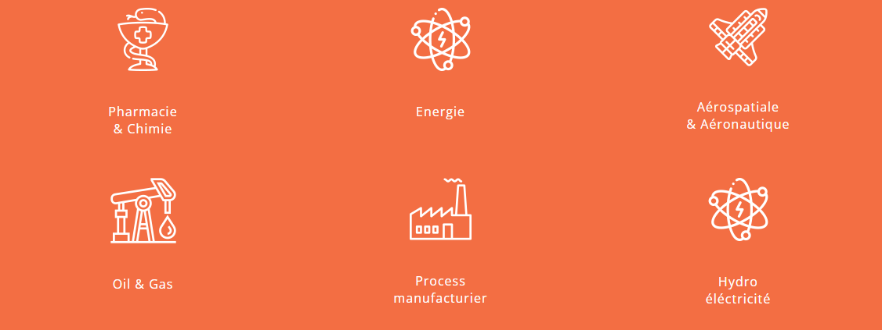
# Présentation de l’entreprise.



**ICONE AUTOMATISATION  
Société spécialisée dans la conception, l’intégration, la mise en service et la maintenance de système automatisés et informatisés pour les outils de production et les moyens d’essais.**

Créé en 1985, Icône Automation est spécialisée dans la conception, l’intégration, la mise en service et la maintenance de systèmes automatisés et informatisés pour les outils de production et les moyens d’essais. Icône Automation a développé depuis de nombreuses années son expertise dans la conception et la réalisation de contrôles commandes en milieu sensible.

Capable de prendre en compte les particularités des secteurs d’activités, Icone sait accompagner dans la réalisation des systèmes d’automatisation et d’informatisation de process industriels.



# Mes missions.

## 2.1. Gestionnaire de licences.

### 2.1.1. Présentation.

En suite de la semaine dernière, j’ai présenté mon application Gestionnaire\_Licences et en est ressortie plusieurs problèmes :

|  |  |  |
| --- | --- | --- |
| Problème | Cause | Résolution |
| Interface | Trop de chiffres, trop d’informations pour des actions simples, pas intuitif et pas assez orienter utilisateur | Simplifié l’interface en ne montrant que l’essentiel de tel à rendre n’importe quel utilisateur capable d’utiliser l’application |
| Manque une page | Manque la page « A propos » permettant de se renseigner sur l’application et l’entreprise ainsi que de renseigner les copyrights | Créer la page « A propos » avec les information nécessaires |
| Logos | Aucun logos utiliser | Utiliser des logos |
| Actions non clair | Certain texte sur les boutons ne décrivait pas bien leur action | Utiliser des termes simple et efficace pour décrire le but principal du bouton |
| Utilisation de comboBox pour client et intégrateur | La sélection du client/Intégrateur par fonction de remplissage de code n’est pas très intuitif | Utilisation de comboBox avec suggestion quand on écrit dedans pour le client/intégrateur |
| Fonction générer licence | Pour générer la licence, il faut ouvrir la seconde application que j’ai développé précédemment sauf qu’elle n’est pas en raccord avec le gestionnaire | Combiner les deux applications pour permettre une utilisation simple (inclure la fonction de génération de la licence et son cryptage dans le gestionnaire |

### 2.1.2. Modifications BDD

Suite aux remarques, j’ai décidé de reprendre la base de données pour changer le fonctionnement, cette fois, les fonctions sont liées grâce d’une nouvelle table aux licences.

Voici ce que donne les tables après mise à jour de la base :

Table Fonctions :

|  |  |  |
| --- | --- | --- |
| Nom | Type | Contraintes |
| Code fonction | Int | PK, Non null |
| Nom | Varchar(255) | Not null |
| Date\_update | Datetime | Not null |
| Description | Varchar(max) | Nullable |

(Le code fonction n’est pas identity (auto incrémenté) car chaque code correspond à une fonction bien précise dans ICONECT donc doit être saisie manuellement)

Table Clients :

|  |  |  |
| --- | --- | --- |
| Nom | Type | Contraintes |
| Code\_client | Int | PK IDENTITY, Non null |
| Nom | Varchar(255) | Not null |
| Date\_update | Datetime | Not null |
| Description | Varchar(max) | Nullable |

Table Intégrateurs :

|  |  |  |
| --- | --- | --- |
| Nom | Type | Contraintes |
| Code\_integrateur | Int | PK IDENTITY, Non null |
| Nom | Varchar(255) | Not null |
| Date\_update | Datetime | Not null |
| Description | Varchar(max) | Nullable |

Table Licences

|  |  |  |
| --- | --- | --- |
| Nom | Type | Contraintes |
| Licence\_ID | Int | PK IDENTITY, Non null |
| Nom | Varchar(255) | Not null |
| Code\_client | Int | FK REFERENCES Table Clients Column Code\_client, Not null |
| Code\_integrateur | Int | FK REFERENCES Table Integrateurs Column Code\_integrateur, Not null |
| Date\_creation | Datetime | Not null |
| Date\_expiration | Datetime | Not null |
| Nb\_equipements | Int | Not null |
| Nb\_variables | Int | Not null |

Table Liaison\_Licence\_Fonctions

|  |  |  |
| --- | --- | --- |
| Nom | Type | Contraintes |
| Licence\_ID | Int | FK REFERENCES Table Licences Column Licence\_ID, Not null |
| Code fonction | Int | FK REFERENCES Table Fonctions Column Code fonction, Not null |

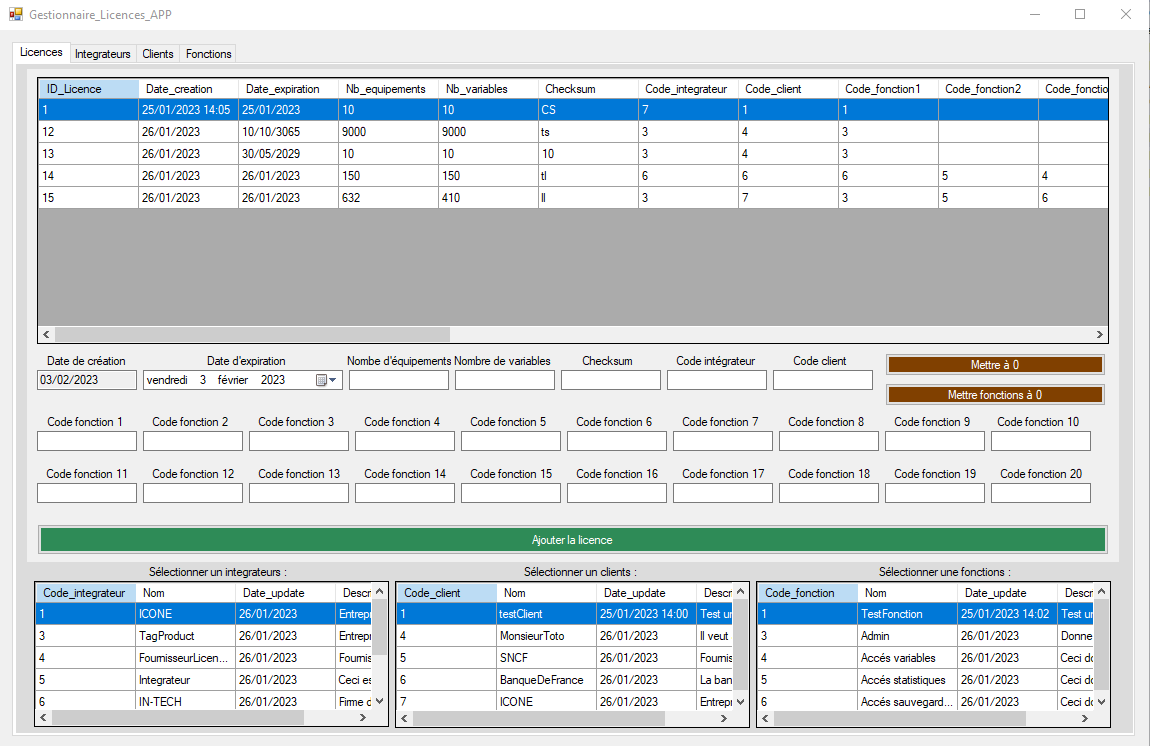
Ces modifications permettent d’inclure un nombre dynamique de fonction aux licences et ainsi limité le temps d’adaptation en cas de changements.

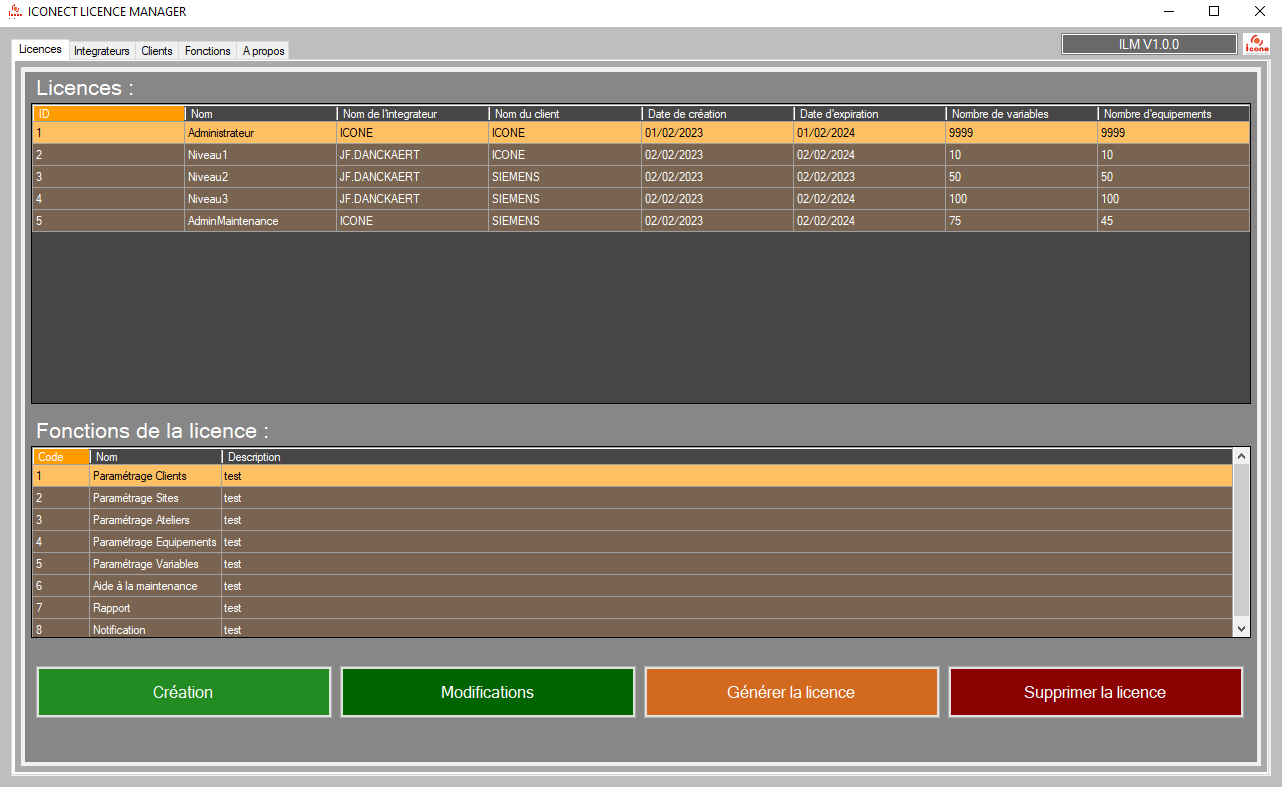
### 2.1.3 Changement graphiques.

Avec les remarques reçues sur les choix graphiques et fonctionnel, j’ai donc changé la casi entièreté de l’interface (Approuvé par le chef le vendredi 3).

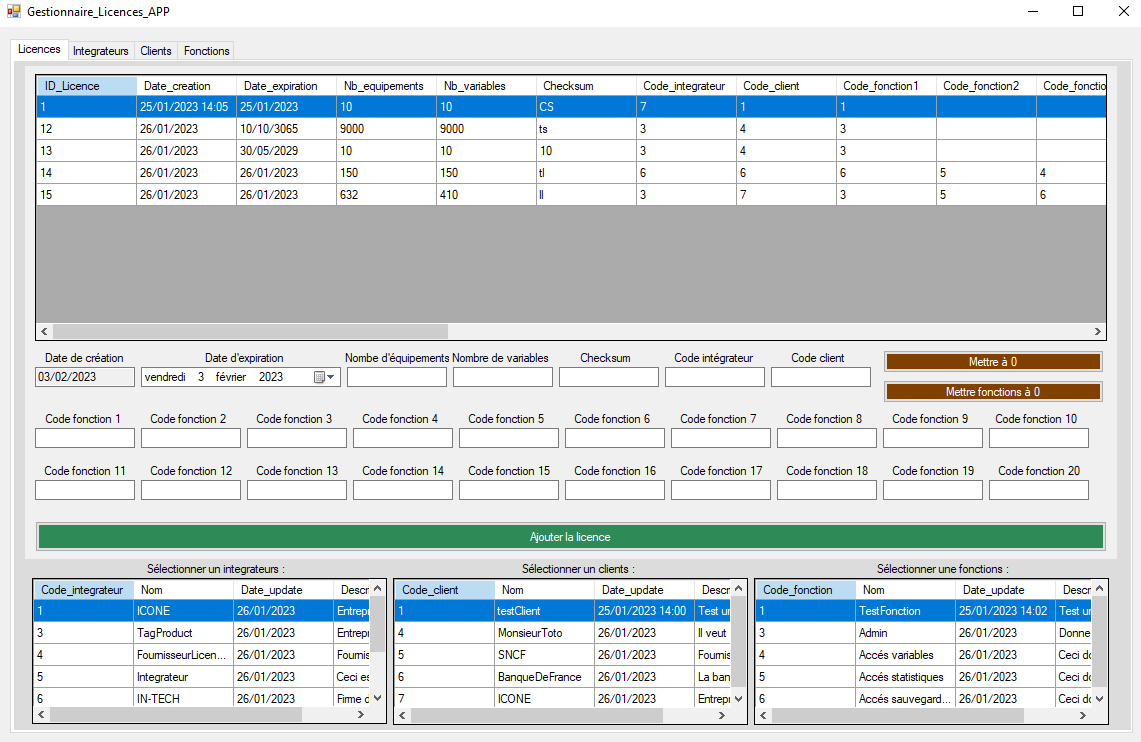
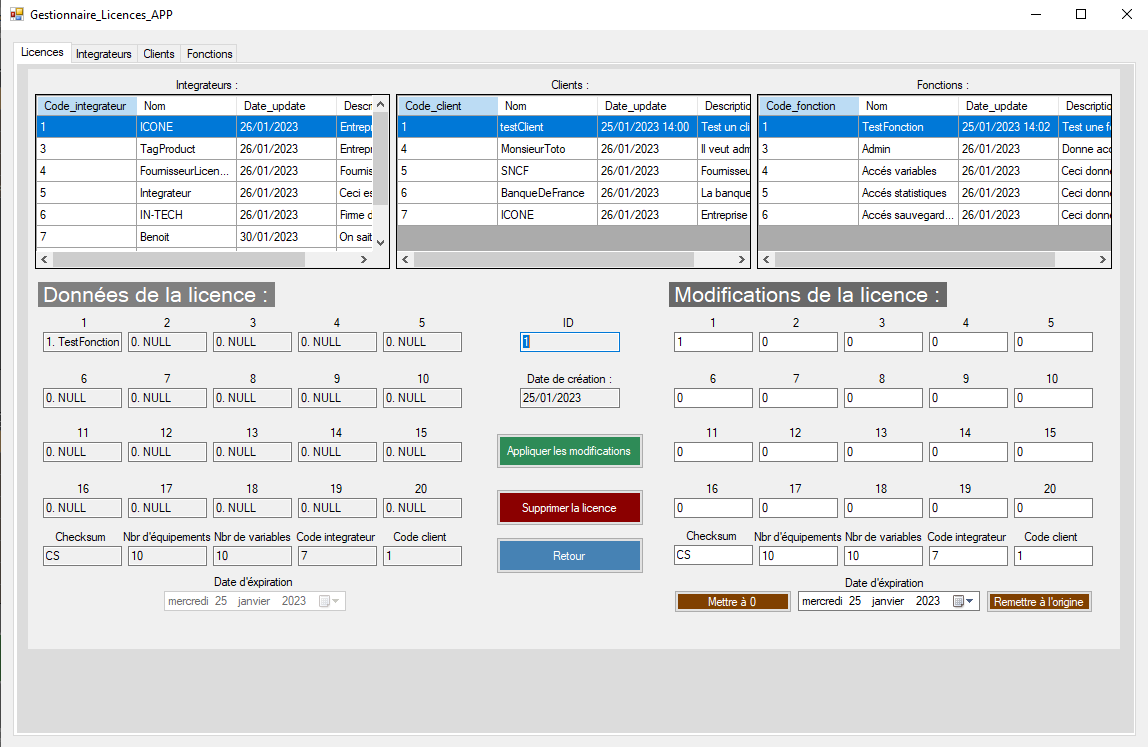
Voici la transformation :

Page d’affichage des licences (Avant-Après) :

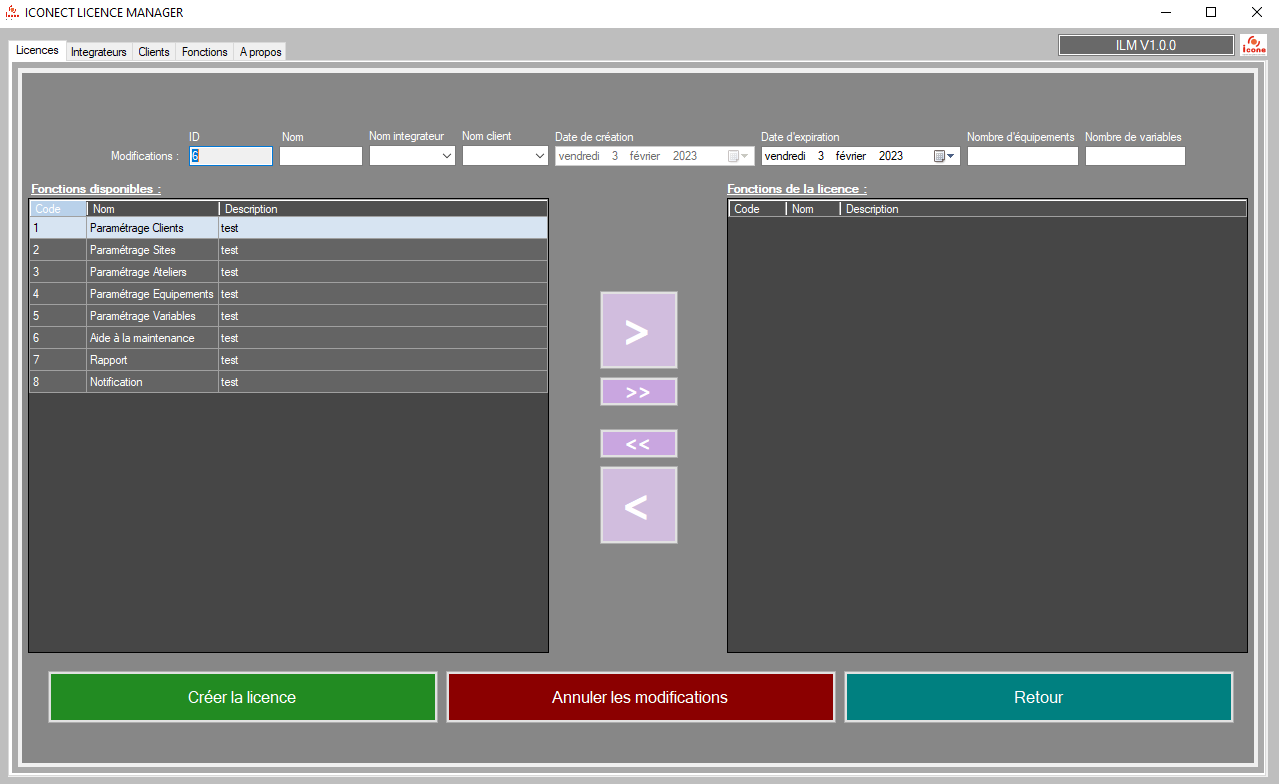
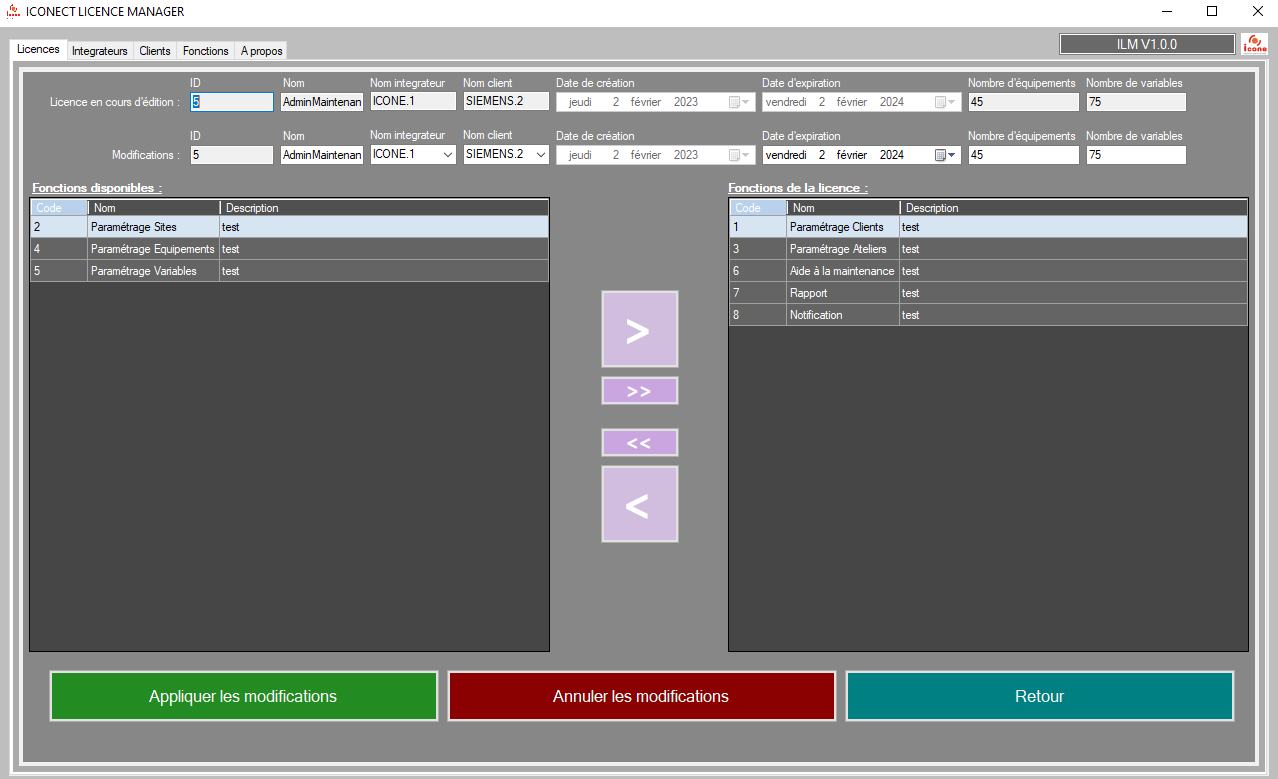




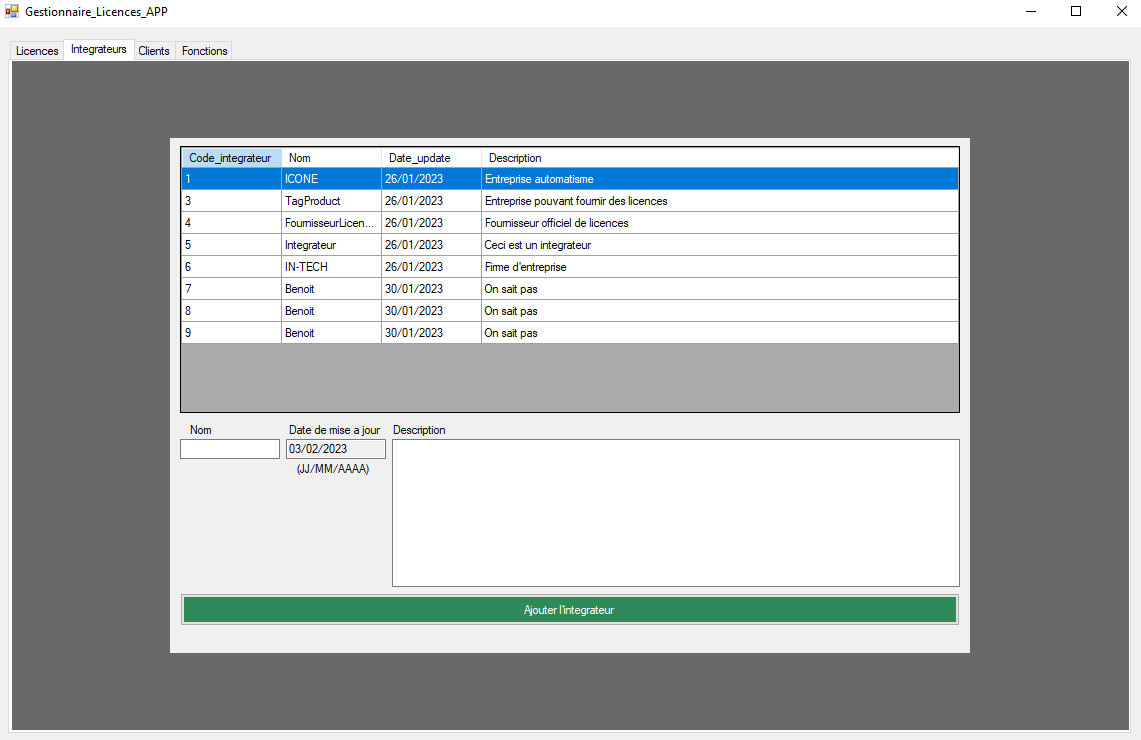
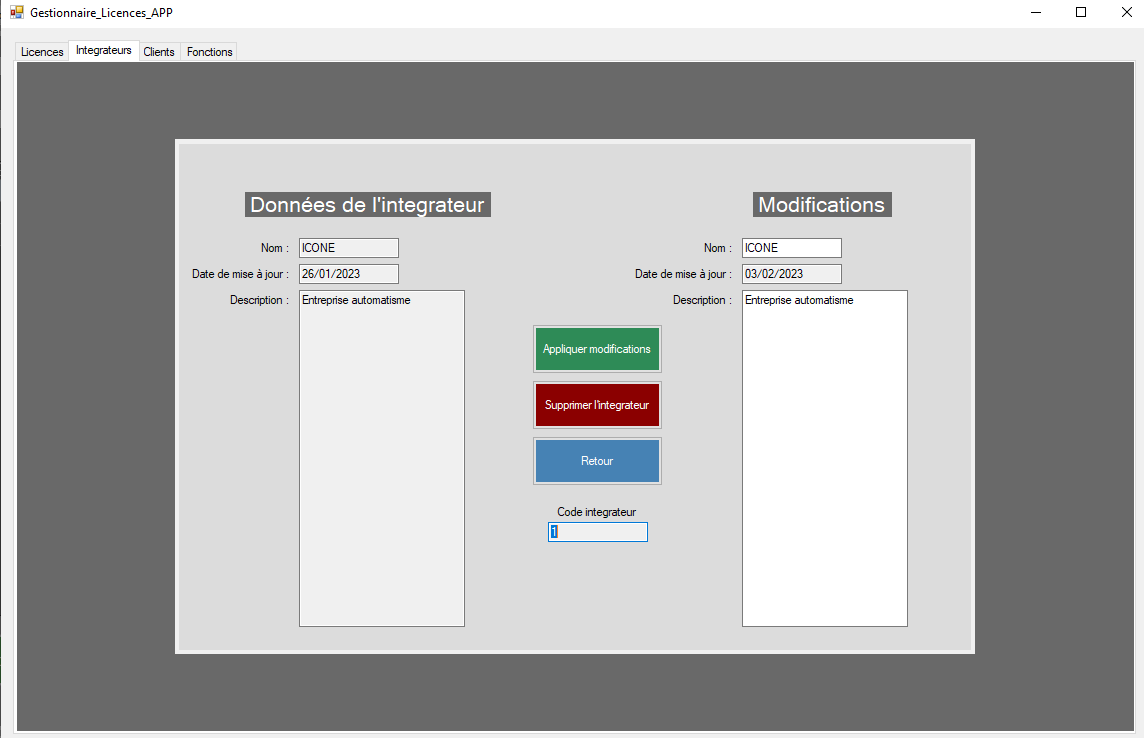
Page d’Ajouts/Modifications des licences (Avant-Après) :

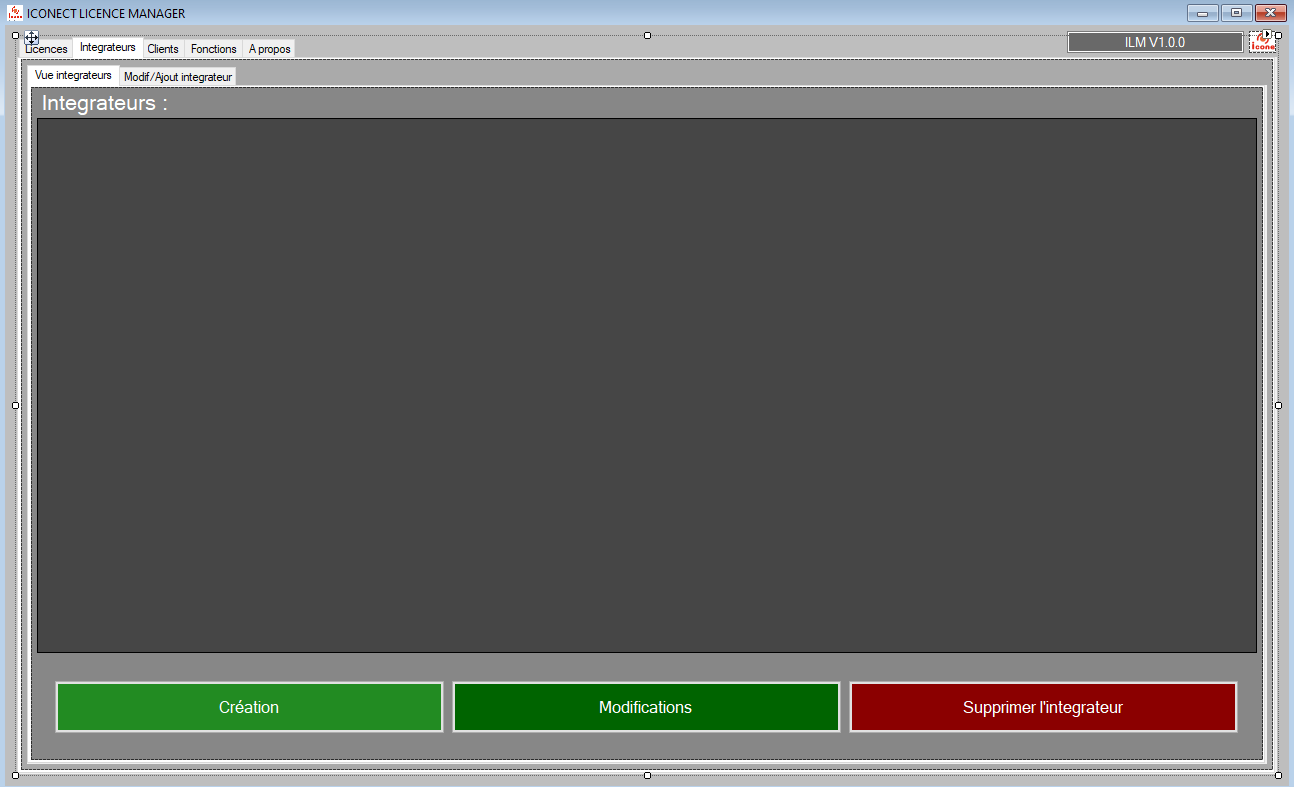
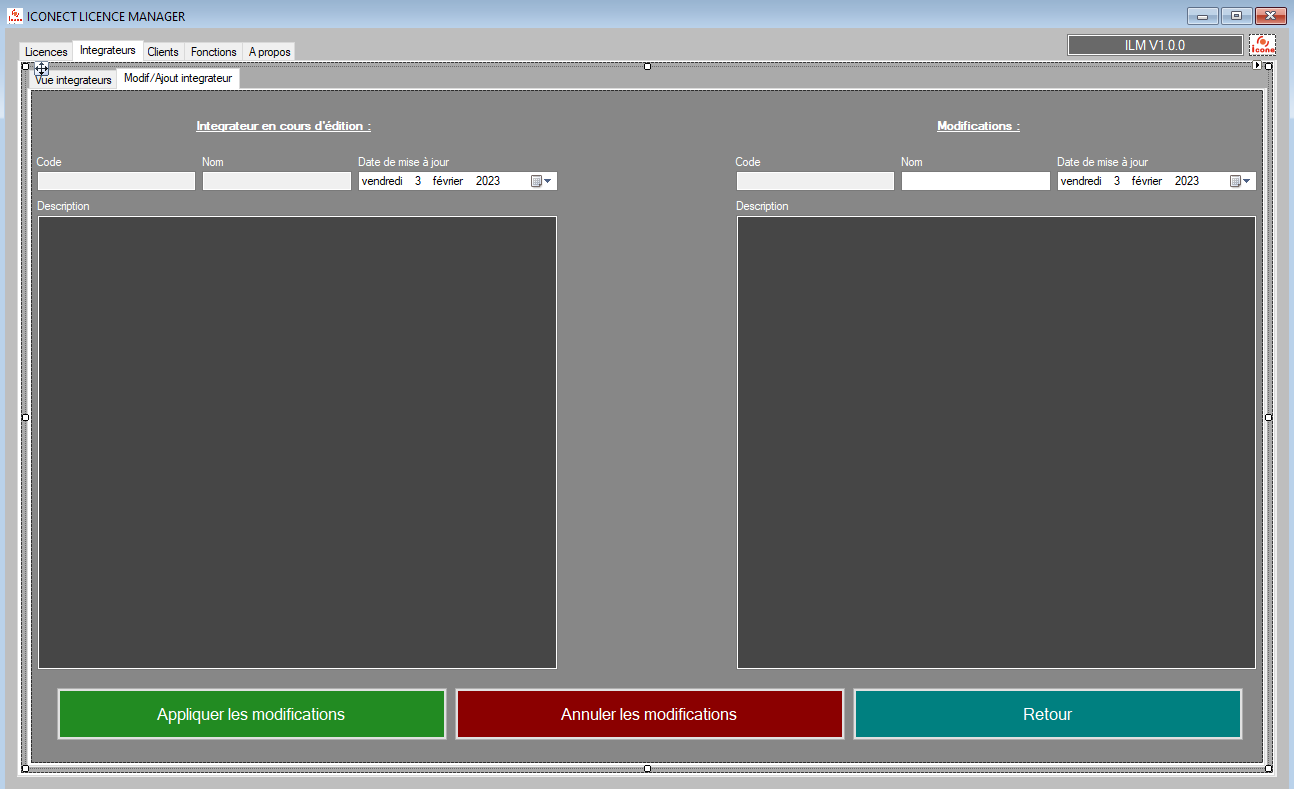
AJOUTS MODIFICATIONS

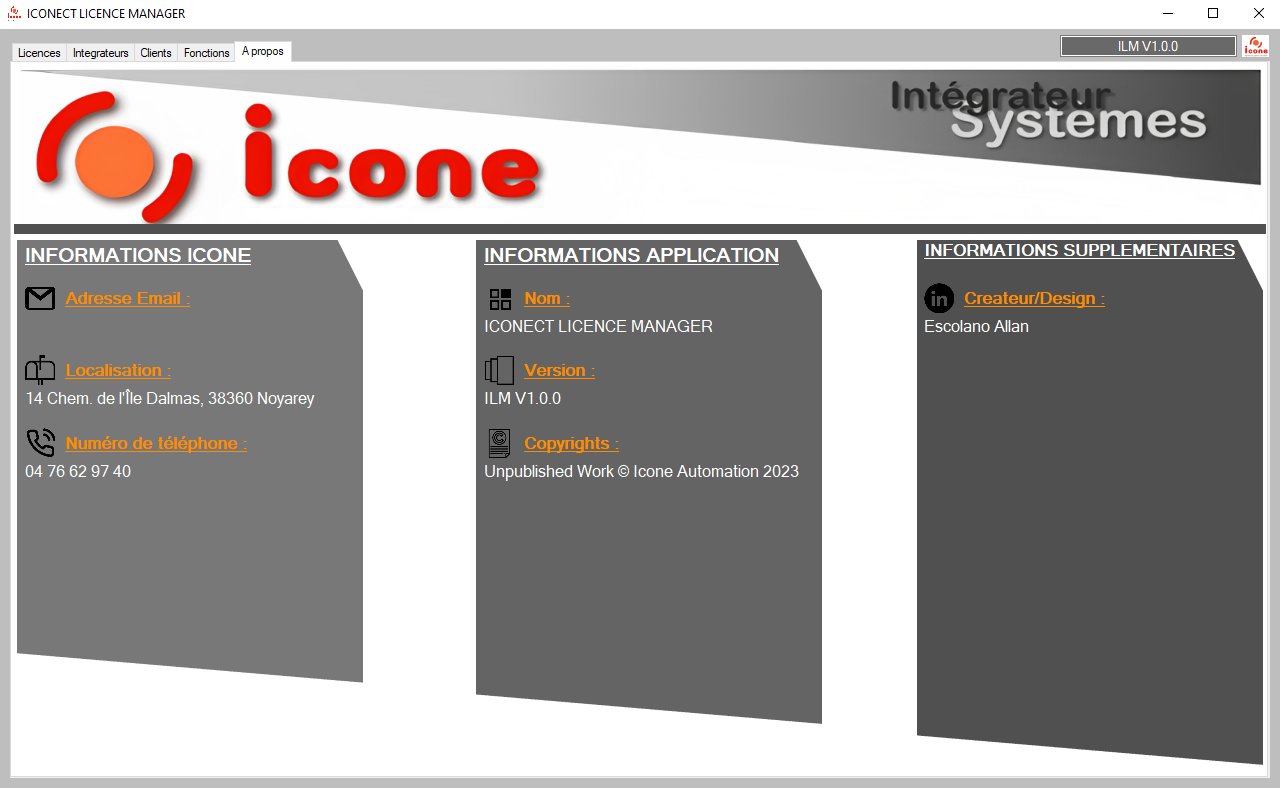
Page d’Ajouts/Modifications des clients/intégrateurs/fonctions (Avant-Après) :

AJOUTS MODIFICATIONS

Page A propos (rajouter dans la nouvelle version) :



Suite à tous ces changements graphiques, j’ai pu répondre à tous les changements demander lors de la réunion.

### 2.1.4. Norme et format de la licence.

Pour le format du fichier de licence, voici ce qui a été mis en place :

* Le fichier prendra comme nom : NomLicence.IconeLic
* La licence se présentera sous forme de fichier texte avec la chaine d’information crypter ainsi que la clé symétrique de décryptage.
* Chaque partie de la chaine de la licence à un rôle précis :

Chaque information est délimitée par « | »

|  |  |
| --- | --- |
| **Désignation** | **Explications** |
| ID | Numéro d’identification de la licence |
| Nom | Le nom de la licence |
| Nom Client | Le nom du client associé à la licence |
| Nom Intégrateur | Le nom de l’intégrateur |
| Date Expiration | La date d’expiration de la licence |
| Nombre d’équipement | Le nombre d’équipement associé à la licence |
| Nombre de variable | Le nombre de variables maximum autorisées par le système |
| Reserve 1 |  |
| Reserve 2 |  |
| Reserve 3 |  |
| Reserve 4 |  |
| Reserve 5 |  |
| Reserve 6 |  |
| Reserve 7 |  |
| Reserve 8 |  |
| Reserve 9 |  |
| Reserve 10 |  |
| Nombre de fonctions | Le nombre de fonctions notifiées dans la licence |
| Id de la fonction 1 | Identifiant de la fonction de la fonction N°1 |
| Id de la fonction 2 | Identifiant de la fonction de la fonction N°2 |
| … | .. |
| Id de la fonction n | Identifiant de la fonction de la fonction N°n |

* Les codes des fonctions correspondes aux fonctions suivantes :

|  |  |  |
| --- | --- | --- |
| **Identifiant de la fonction** | **Nom de la fonction** | **Explication** |
| 1 | Paramétrage Client | Fonction qui permet de Créer/Modifier/Supprimer des clients |
| 2 | Paramétrage Site | Fonction qui permet de Créer/Modifier/Supprimer des sites |
| 3 | Paramétrage Atelier | Fonction qui permet de Créer/Modifier/Supprimer des ateliers ou des lignes |
| 4 | Paramétrage Equipement | Fonction qui permet de Créer/Modifier/Supprimer des équipements |
| 5 | Paramétrage Variable | Fonction qui permet de Créer/Modifier/Supprimer des variables |
| 6 | Aide à la maintenance | Fonction qui permet de faire une aide au défaut sur une variable |
| 7 | Rapport | Fonction qui permet de générer les rapports dans l’application |
| 8 | Notification | Fonction qui permet de notifier un utilisateur par SMS ou mail si une variable dépasse un seuil. |

Ces codes fonctions correspondent aux fonctions disponibles dans IConect©.

***Par exemple, la chaine associée à la licence suivante :***

|  |  |
| --- | --- |
| **Désignation** | **Valeurs** |
| ID | 1 |
| Nom | Ma licence |
| Nom Client | Mon client |
| Nom Intégrateur | Mon Intégrateur |
| Date Expiration | 01/01/2023 |
| Nombre d’équipement | 100 |
| Nombre de variable | 100 |
| Reserve 1 | Non applicable |
| Reserve 2 | Non applicable |
| Reserve 3 | Non applicable |
| Reserve 4 | Non applicable |
| Reserve 5 | Non applicable |
| Reserve 6 | Non applicable |
| Reserve 7 | Non applicable |
| Reserve 8 | Non applicable |
| Reserve 9 | Non applicable |
| Reserve 10 | Non applicable |
| Nombre de fonctions | 3 |
| Id de la fonction 1 | 1 |
| Id de la fonction 2 | 2 |
| Id de la fonction 3 | 5 |

La licence Iconect© numéro 1 qui s’appelle « Ma Licence » pour le client « Mon Client » déployé par l’intégrateur « Mon intégrateur » qui expire le 01/01/2023 dont le logiciel peut accepter 100 équipements et 100 variables permet les fonctionnalités suivantes :

- Paramétrer les clients

- Paramétrer les sites

- Paramétrer les variables

La chaine contenue dans le fichier de licence sera donc la suivante :

1|Malicence|Monclient|MonIntégrateur|01/01/2023|100|100|Null|Null|Null|Null|Null|Null|Null|Null|Null|Null|3|1|2|5

Afin de maintenir un bon niveau de sécurité, j’ai utilisé la DLL System.Security.Cryptography.Algorithms.dll pour crypter la chaine. La chaine cryptée est cryptée grâce à une clé de 128bits générée aléatoirement pendant le remplissage des informations et sont ensuite stockées dans le fichier «MaLicence.IconeLic ».

Ce fichier de licence sera lu par le logiciel Iconect© pour permettre ou non les fonctionnalités associées.

# Outils utilisés.

|  |  |
| --- | --- |
| Nom de l’outil | Fonction de l’outil |
| Visual Studio 2019 | Espace de développement avec multiple langage de programmations |
| VMWare Workstation 16 | VMware Workstation est un outil de virtualisation de poste de travail, il sert à mettre en place un environnement de test pour développer de nouveaux logiciels, ou pour tester l'architecture complexe d’un système d’exploitation avant de l’installer réellement sur une machine physique. |
| Microsoft SQL Server | Microsoft SQL Server est un système de gestion de base de données en langage SQL incorporant entre autres un SGBDR développé et commercialisé par la société Microsoft. |

# Conclusion.

Cette semaine fu très compliqué et remplie de rebondissements, suite au remarques pendant la réunion j’ai vite réalisé que le projet n’était pas comme un petit projet pour des cours, mais bien un projet professionnel destiné à des utilisateurs et à une application finale.  
  
J’ai donc pu m’orienter plus coté front end et réfléchir à en tant qu’utilisateur, qu’est-ce que j’aimerais voir de l’application quand je l’utilise.  
Se mettre ce point de vue permet vraiment de se mettre dans la peau d’un utilisateur et permet de créer des choses plaisantes visuel et en fonctionnalités aussi.

Suite aux modifications, lors d’une courte réunion j’ai représenté la nouvelle version à mon chef qui a de suite approuvé et cité « Avec toi Allan, c’est le jour ou la nuit » ce que j’ai trouvé très drôle mais très intéressant quand remarque.

Tristement par manque de temps, l’application n’est pas finis (Coté CRUD intégrateurs/clients/fonctions) mais peut être assez rapidement terminé (car le CRUD est le même pour les trois).

# Niko-Niko.

|  |  |  |
| --- | --- | --- |
| JOUR | RESSENTI | TACHE |
| Lundi | Smiley Emoji Emoticon Fear Happiness - Sweating Png | Scared emoji, Emoticon,  Emoji images | Jour de reunion  (Stresse un peu quand même) |
| Mardi | 👎 Thumbs Down Emoji: Show Your Disapproval Quickly With Zero Fuss | 🏆  Emojiguide | Jour de la nouvelle BDD  (SQL pas ouf) |
| Mercredi | Man Artist Emoji (U+1F468, U+200D, U+1F3A8) | Jour de la nouvelle interface  (Choix artistiques mis en question) |
| Jeudi | Smiley engineer images vectorielles, Smiley engineer vecteurs libres de  droits | Depositphotos | Fonctionnement CRUD des Licences  (Un CRUD classique) |
| Vendredi | 🥳 Partying Face Emoji | Fonctions annexes/Page A propos  (Validation chef + fin de stage) |

# Annexe codes.

## 6.1. Code ICONECT LICENCE MANAGER.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

/\*

TODOLIST

-Faire CRUD pour CLIENTS

-Faire CRUD pour INTEGRATEURS

-Faire CRUD pour FONCTIONS

-Faire page A propos

-Rajouter logos

\*/

/\*

FONCTIONS TERMINEES

-CRUD LICENCES

-Foncitonnement du mode ajout/modification

-Fonction ANNULER modifications pour le mode ajout/modification

-Génération du fichier de licence

-Fonction retour pour le mode ajout/modifications

-Nom de l'application avec version dynamique (à changer dans app.config)

-Des fenetres de confirmation des actions

\*/

namespace ICONECT\_LICENCE\_MANAGER

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ToString());

//Creation d'une liste d'objet licence

List<Licence> listeLicences = new List<Licence>();

//Creation d'une liste d'objet fonction

List<Fonction> listeFonctions = new List<Fonction>();

//Creation d'une liste d'objet integrateur

List<Integrateur> listeIntegrateurs = new List<Integrateur>();

//Creation d'une liste d'objet client

List<Client> listeClients = new List<Client>();

private void Form1\_Load(object sender, EventArgs e)

{

tc\_Licences.Appearance = TabAppearance.FlatButtons;

tc\_Licences.ItemSize = new Size(0, 1);

tc\_Licences.SizeMode = TabSizeMode.Fixed;

tc\_Integrateurs.Appearance = TabAppearance.FlatButtons;

tc\_Integrateurs.ItemSize = new Size(0, 1);

tc\_Integrateurs.SizeMode = TabSizeMode.Fixed;

#region ajout des colonnes des datagridview

dgv\_R\_Licences.ColumnCount = 8;

dgv\_R\_Licences.Columns[0].Name = "ID";

dgv\_R\_Licences.Columns[1].Name = "Nom";

dgv\_R\_Licences.Columns[2].Name = "Nom de l'integrateur";

dgv\_R\_Licences.Columns[3].Name = "Nom du client";

dgv\_R\_Licences.Columns[4].Name = "Date de création";

dgv\_R\_Licences.Columns[5].Name = "Date d'expiration";

dgv\_R\_Licences.Columns[6].Name = "Nombre de variables";

dgv\_R\_Licences.Columns[7].Name = "Nombre d'equipements";

dgv\_R\_LicenceFonctions.ColumnCount = 3;

dgv\_R\_LicenceFonctions.Columns[0].Name = "Code";

dgv\_R\_LicenceFonctions.Columns[1].Name = "Nom";

dgv\_R\_LicenceFonctions.Columns[2].Name = "Description";

dgv\_U\_FonctionsDisponibles.ColumnCount = 3;

dgv\_U\_FonctionsDisponibles.Columns[0].Name = "Code";

dgv\_U\_FonctionsDisponibles.Columns[1].Name = "Nom";

dgv\_U\_FonctionsDisponibles.Columns[2].Name = "Description";

dgv\_U\_LicenceFonctions.ColumnCount = 3;

dgv\_U\_LicenceFonctions.Columns[0].Name = "Code";

dgv\_U\_LicenceFonctions.Columns[1].Name = "Nom";

dgv\_U\_LicenceFonctions.Columns[2].Name = "Description";

dgv\_R\_Integrateurs.ColumnCount = 3;

dgv\_R\_Integrateurs.Columns[0].Name = "Code";

dgv\_R\_Integrateurs.Columns[1].Name = "Nom";

dgv\_R\_Integrateurs.Columns[2].Name = "Description";

#endregion

adaptDataGridView();

tb\_AppVersion.Text = "ILM V" + ConfigurationManager.AppSettings["AppVersion"].ToString();

lb\_AdresseEmail.Text = ConfigurationManager.AppSettings["AdresseEmail"].ToString();

lb\_Localisation.Text = ConfigurationManager.AppSettings["Localisation"].ToString();

lb\_NumeroTel.Text = ConfigurationManager.AppSettings["NumeroTel"].ToString();

lb\_AppName.Text = ConfigurationManager.AppSettings["AppName"].ToString();

lb\_Copyrights.Text = ConfigurationManager.AppSettings["Copyrights"].ToString();

lb\_Version.Text = "ILM V" + ConfigurationManager.AppSettings["AppVersion"].ToString();

fill\_dgv\_Fonctions();

fill\_dgv\_Clients();

fill\_dgv\_Integrateurs();

fill\_dgv\_Licences();

#region alimentation des combobox

foreach (Client client in listeClients)

{

cb\_U\_LicenceClient.Items.Add(client.Nom + "." + client.Code\_client.ToString());

}

foreach (Integrateur integrateur in listeIntegrateurs)

{

cb\_U\_LicenceIntegrateur.Items.Add(integrateur.Nom + "." + integrateur.Code\_integrateur.ToString());

}

#endregion

}

//Remplie les liste et les datagridviews avec les données de la base

private void fill\_dgv\_Fonctions()

{

//Definition de la procédure stockée

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.get\_Fonctions", con);

cmd.CommandType = CommandType.StoredProcedure;

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

listeFonctions.Clear();

while (rdr.Read())

{

listeFonctions.Add(new Fonction(Convert.ToInt32(rdr[0]), Convert.ToString(rdr[1]), Convert.ToDateTime(rdr[2]), Convert.ToString(rdr[3])));

}

con.Close();

}

private void fill\_dgv\_Clients()

{

//Definition de la procédure stockée

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.get\_Clients", con);

cmd.CommandType = CommandType.StoredProcedure;

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

listeClients.Clear();

while (rdr.Read())

{

listeClients.Add(new Client(Convert.ToInt32(rdr[0]), Convert.ToString(rdr[1]), Convert.ToDateTime(rdr[2]), Convert.ToString(rdr[3])));

}

con.Close();

}

private void fill\_dgv\_Integrateurs()

{

//Definition de la procédure stockée

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.get\_Integrateurs", con);

cmd.CommandType = CommandType.StoredProcedure;

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

listeIntegrateurs.Clear();

while (rdr.Read())

{

listeIntegrateurs.Add(new Integrateur(Convert.ToInt32(rdr[0]), Convert.ToString(rdr[1]), Convert.ToDateTime(rdr[2]), Convert.ToString(rdr[3])));

}

con.Close();

}

private void fill\_dgv\_Licences()

{

dgv\_R\_Licences.SelectionChanged -= SelectedLicenceChanged;

dgv\_R\_Licences.Rows.Clear();

//Definition de la procédure stockée

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.get\_Licences", con);

cmd.CommandType = CommandType.StoredProcedure;

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

listeLicences.Clear();

while (rdr.Read())

{

listeLicences.Add(new Licence(Convert.ToInt32(rdr[0]), Convert.ToString(rdr[1]), Convert.ToInt32(rdr[2]), Convert.ToInt32(rdr[3]), Convert.ToDateTime(rdr[4]), Convert.ToDateTime(rdr[5]), Convert.ToInt32(rdr[6]), Convert.ToInt32(rdr[7])));

int rowIndex = dgv\_R\_Licences.Rows.Add();

dgv\_R\_Licences.Rows[rowIndex].Cells[0].Value = rdr[0];

dgv\_R\_Licences.Rows[rowIndex].Cells[1].Value = rdr[1];

foreach (Licence licence in listeLicences)

{

if (Convert.ToInt32(rdr[0]) == licence.Licence\_ID)

{

dgv\_R\_Licences.Rows[rowIndex].Cells[2].Value = get\_Licence\_Integrateur(licence).Nom;

dgv\_R\_Licences.Rows[rowIndex].Cells[3].Value = get\_Licence\_Client(licence).Nom;

}

}

dgv\_R\_Licences.Rows[rowIndex].Cells[4].Value = Convert.ToDateTime(rdr[4]).ToString("dd/MM/yyyy");

dgv\_R\_Licences.Rows[rowIndex].Cells[5].Value = Convert.ToDateTime(rdr[5]).ToString("dd/MM/yyyy");

dgv\_R\_Licences.Rows[rowIndex].Cells[6].Value = rdr[6];

dgv\_R\_Licences.Rows[rowIndex].Cells[7].Value = rdr[7];

}

con.Close();

//Permet d'assigner la fonction daffichage des Foncitons de la Licence au datagriview aprés que les licence ont charger

dgv\_R\_Licences.ClearSelection();

dgv\_R\_Licences.SelectionChanged += SelectedLicenceChanged;

}

//Quand une licence est séléctioné, afficher ses fonctions

private void fill\_dgv\_LicenceFonctions(int Licence\_ID)

{

dgv\_R\_LicenceFonctions.Rows.Clear();

//Definition de la procédure stockée

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.get\_LicenceFonctions", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@Licence\_ID", SqlDbType.Int).Value = Licence\_ID;

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

int rowIndex = dgv\_R\_LicenceFonctions.Rows.Add();

dgv\_R\_LicenceFonctions.Rows[rowIndex].Cells[0].Value = rdr[0];

dgv\_R\_LicenceFonctions.Rows[rowIndex].Cells[1].Value = rdr[1];

dgv\_R\_LicenceFonctions.Rows[rowIndex].Cells[2].Value = rdr[3];

}

con.Close();

}

private void SelectedLicenceChanged(object sender, EventArgs e)

{

if (dgv\_R\_Licences.SelectedRows.Count > 0)

{

int Licence\_ID = Convert.ToInt32(dgv\_R\_Licences.SelectedRows[0].Cells[0].Value);

fill\_dgv\_LicenceFonctions(Licence\_ID);

}

}

//Quand le mode Ajout est choisi

private void goToAjoutMode(object sender, EventArgs e)

{

#region cacher les champs

btn\_U\_Licence.Visible = false;

lb\_V1.Visible = false;

lb\_V2.Visible = false;

lb\_V3.Visible = false;

lb\_V4.Visible = false;

lb\_V5.Visible = false;

lb\_V6.Visible = false;

lb\_V7.Visible = false;

lb\_V8.Visible = false;

lb\_V9.Visible = false;

tb\_R\_LicenceID.Visible = false;

tb\_R\_LicenceIntegrateur.Visible = false;

tb\_R\_LicenceClient.Visible = false;

tb\_R\_LicenceIntegrateur.Visible = false;

tb\_R\_LicenceNbEquipements.Visible = false;

tb\_R\_LicenceNbVariables.Visible = false;

tb\_R\_LicenceNom.Visible = false;

dtp\_R\_LicenceDateCreation.Visible = false;

dtp\_R\_LicenceDateExpiration.Visible = false;

#endregion

fill\_dgv\_FonctionsDipsonibles();

//Definition de la procédure stockée

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.get\_nextLicenceID", con);

cmd.CommandType = CommandType.StoredProcedure;

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

tb\_U\_LicenceID.Text = rdr[0].ToString();

}

con.Close();

dtp\_U\_LicenceDateCreation.Value = DateTime.Now;

dtp\_U\_LicenceDateExpiration.Value = DateTime.Now;

adaptDataGridView();

tc\_Licences.SelectedIndex = 1;

}

private void fill\_dgv\_FonctionsDipsonibles()

{

dgv\_U\_FonctionsDisponibles.Rows.Clear();

//Definition de la procédure stockée

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.get\_Fonctions", con);

cmd.CommandType = CommandType.StoredProcedure;

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

int rowIndex = dgv\_U\_FonctionsDisponibles.Rows.Add();

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[0].Value = rdr[0];

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[1].Value = rdr[1];

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[2].Value = rdr[3];

}

con.Close();

}

private void ajouterLicence(object sender, EventArgs e)

{

DialogResult dr = MessageBox.Show("Vous êtes sur le point de créer la licence suivante : " + tb\_U\_LicenceID.Text + " " + tb\_U\_LicenceNom.Text + "\n\nconfirmez-vous cette action ?", "Création licence", MessageBoxButtons.YesNo);

if (dr == DialogResult.Yes)

{

#region Ajout de la licence

//Definition de la procédure stockée

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.create\_Licence", con);

cmd.CommandType = CommandType.StoredProcedure;

try

{

cmd.Parameters.Add("@Nom", SqlDbType.VarChar).Value = tb\_U\_LicenceNom.Text;

//Prend la chaine, la coupe au niveau des points et prend la derneriere valeur de la chaine (le code de l'integrateur/client)

cmd.Parameters.Add("@Code\_integrateur", SqlDbType.Int).Value = Convert.ToInt32(cb\_U\_LicenceClient.SelectedItem.ToString().Split('.')[cb\_U\_LicenceClient.SelectedItem.ToString().Split('.').Count() - 1]);

cmd.Parameters.Add("@Code\_client", SqlDbType.Int).Value = Convert.ToInt32(cb\_U\_LicenceIntegrateur.SelectedItem.ToString().Split('.')[cb\_U\_LicenceIntegrateur.SelectedItem.ToString().Split('.').Count() - 1]);

cmd.Parameters.Add("@Date\_creation", SqlDbType.DateTime).Value = dtp\_U\_LicenceDateCreation.Value.ToString("dd/MM/yyyy");

cmd.Parameters.Add("@Date\_expiration", SqlDbType.DateTime).Value = dtp\_U\_LicenceDateExpiration.Value.ToString("dd/MM/yyyy");

cmd.Parameters.Add("@Nb\_variables", SqlDbType.Int).Value = Convert.ToInt32(tb\_U\_LicenceNbVariables.Text);

cmd.Parameters.Add("@Nb\_equipements", SqlDbType.Int).Value = Convert.ToInt32(tb\_U\_LicenceNbEquipements.Text);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

catch (Exception err)

{

MessageBox.Show("Erreur paramêtres licence : verifier que tous les paramêtres sont bien remplies. ⚠Fermeture du mode Ajout⚠");

con.Close();

}

#endregion

#region Ajout des fonctions de la licence

foreach (DataGridViewRow row in dgv\_U\_LicenceFonctions.Rows)

{

SqlCommand cmd2 = new SqlCommand("Base\_ILM.dbo.create\_Liaison\_Licence\_Fonction", con);

cmd2.CommandType = CommandType.StoredProcedure;

try

{

int Licence\_ID = Convert.ToInt32(tb\_U\_LicenceID.Text);

int Code\_fonction = Convert.ToInt32(row.Cells[0].Value);

cmd2.Parameters.Add("@Licence\_ID", SqlDbType.Int).Value = Licence\_ID;

cmd2.Parameters.Add("@Code\_fonction", SqlDbType.Int).Value = Code\_fonction;

con.Open();

cmd2.ExecuteNonQuery();

con.Close();

}

catch (Exception err)

{

con.Close();

}

}

#endregion

fill\_dgv\_Licences();

dgv\_R\_LicenceFonctions.Rows.Clear();

dgv\_U\_FonctionsDisponibles.Rows.Clear();

dgv\_U\_LicenceFonctions.Rows.Clear();

adaptDataGridView();

tc\_Licences.SelectedIndex = 0;

#region vider les champs

tb\_U\_LicenceNom.Text = "";

cb\_U\_LicenceIntegrateur.SelectedItem = null;

cb\_U\_LicenceClient.SelectedItem = null;

dtp\_R\_LicenceDateExpiration.Value = DateTime.Now;

tb\_U\_LicenceNbEquipements.Text = "";

tb\_U\_LicenceNbVariables.Text = "";

cb\_U\_LicenceClient.Text = "";

cb\_U\_LicenceIntegrateur.Text = "";

#endregion

#region montrer les champs

btn\_U\_Licence.Visible = true;

lb\_V1.Visible = true;

lb\_V2.Visible = true;

lb\_V3.Visible = true;

lb\_V4.Visible = true;

lb\_V5.Visible = true;

lb\_V6.Visible = true;

lb\_V7.Visible = true;

lb\_V8.Visible = true;

lb\_V9.Visible = true;

tb\_R\_LicenceID.Visible = true;

tb\_R\_LicenceIntegrateur.Visible = true;

tb\_R\_LicenceClient.Visible = true;

tb\_R\_LicenceIntegrateur.Visible = true;

tb\_R\_LicenceNbEquipements.Visible = true;

tb\_R\_LicenceNbVariables.Visible = true;

tb\_R\_LicenceNom.Visible = true;

dtp\_R\_LicenceDateCreation.Visible = true;

dtp\_R\_LicenceDateExpiration.Visible = true;

btn\_C\_Licence.Visible = true;

btn\_U\_Licence.Visible = true;

#endregion

btn\_U\_Licence.Visible = true;

}

}

//Quand le mode Modifications est choisi

private void goToModificationsMode(object sender, EventArgs e)

{

if (dgv\_R\_Licences.SelectedRows.Count > 0)

{

int Licence\_ID = Convert.ToInt32(dgv\_R\_Licences.SelectedRows[0].Cells[0].Value);

foreach(Licence licence in listeLicences)

{

if (Licence\_ID == licence.Licence\_ID)

{

#region valeurs de la licence

tb\_R\_LicenceID.Text = licence.Licence\_ID.ToString();

tb\_R\_LicenceNom.Text = licence.Nom.ToString();

tb\_R\_LicenceIntegrateur.Text = get\_Licence\_Integrateur(licence).Nom + "." + licence.Code\_integrateur.ToString();

tb\_R\_LicenceClient.Text = get\_Licence\_Client(licence).Nom + "." + licence.Code\_client.ToString();

dtp\_R\_LicenceDateCreation.Value = licence.Date\_creation;

dtp\_R\_LicenceDateExpiration.Value = licence.Date\_expiration;

tb\_R\_LicenceNbEquipements.Text = licence.Nb\_equipements.ToString();

tb\_R\_LicenceNbVariables.Text = licence.Nb\_variables.ToString();

#endregion

#region valeurs modifiables de la licence

tb\_U\_LicenceID.Text = licence.Licence\_ID.ToString();

tb\_U\_LicenceNom.Text = licence.Nom.ToString();

foreach (string Code\_client in cb\_U\_LicenceClient.Items)

{

if (licence.Code\_client == Convert.ToInt32(Code\_client.ToString().Split('.')[Code\_client.ToString().Split('.').Count() - 1]))

{

cb\_U\_LicenceClient.SelectedItem = Code\_client;

}

}

foreach (string Code\_integrateur in cb\_U\_LicenceIntegrateur.Items)

{

if (licence.Code\_integrateur == Convert.ToInt32(Code\_integrateur.ToString().Split('.')[Code\_integrateur.ToString().Split('.').Count() - 1]))

{

cb\_U\_LicenceIntegrateur.SelectedItem = Code\_integrateur;

}

}

dtp\_U\_LicenceDateCreation.Value = licence.Date\_creation;

dtp\_U\_LicenceDateExpiration.Value = licence.Date\_expiration;

tb\_U\_LicenceNbEquipements.Text = licence.Nb\_equipements.ToString();

tb\_U\_LicenceNbVariables.Text = licence.Nb\_variables.ToString();

#endregion

}

}

btn\_C\_Licence.Visible = false;

fill\_dgv\_LicenceNoFonctions();

fill\_dgv\_LicenceFonctions();

adaptDataGridView();

tc\_Licences.SelectedIndex = 1;

}

}

private void fill\_dgv\_LicenceNoFonctions()

{

dgv\_U\_FonctionsDisponibles.Rows.Clear();

//Definition de la procédure stockée

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.get\_LicenceNoFonctions", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@Licence\_ID", SqlDbType.Int).Value = tb\_R\_LicenceID.Text.ToString();

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

int rowIndex = dgv\_U\_FonctionsDisponibles.Rows.Add();

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[0].Value = rdr[0];

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[1].Value = rdr[1];

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[2].Value = rdr[3];

}

con.Close();

}

private void fill\_dgv\_LicenceFonctions()

{

dgv\_U\_LicenceFonctions.Rows.Clear();

//Definition de la procédure stockée

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.get\_LicenceFonctions", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@Licence\_ID", SqlDbType.Int).Value = tb\_R\_LicenceID.Text.ToString();

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

int rowIndex = dgv\_U\_LicenceFonctions.Rows.Add();

dgv\_U\_LicenceFonctions.Rows[rowIndex].Cells[0].Value = rdr[0];

dgv\_U\_LicenceFonctions.Rows[rowIndex].Cells[1].Value = rdr[1];

dgv\_U\_LicenceFonctions.Rows[rowIndex].Cells[2].Value = rdr[3];

}

con.Close();

}

private void updateLicence(object sender, EventArgs e)

{

DialogResult dr = MessageBox.Show("Vous êtes sur le point de modifier la licence suivante : " + tb\_R\_LicenceID.Text + " " + tb\_R\_LicenceNom.Text + "\n\nconfirmez-vous cette action ?", "Modifications licence", MessageBoxButtons.YesNo);

bool updatePassed = false;

#region modification de la licence

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.update\_Licence", con);

cmd.CommandType = CommandType.StoredProcedure;

try

{

cmd.Parameters.Add("@Licence\_ID", SqlDbType.Int).Value = Convert.ToInt32(tb\_R\_LicenceID.Text);

cmd.Parameters.Add("@Nom", SqlDbType.VarChar).Value = tb\_U\_LicenceNom.Text;

cmd.Parameters.Add("@Code\_integrateur", SqlDbType.Int).Value = Convert.ToInt32(cb\_U\_LicenceClient.SelectedItem.ToString().Split('.')[cb\_U\_LicenceClient.SelectedItem.ToString().Split('.').Count() - 1]);

cmd.Parameters.Add("@Code\_client", SqlDbType.Int).Value = Convert.ToInt32(cb\_U\_LicenceIntegrateur.SelectedItem.ToString().Split('.')[cb\_U\_LicenceIntegrateur.SelectedItem.ToString().Split('.').Count() - 1]);

cmd.Parameters.Add("@Date\_expiration", SqlDbType.DateTime).Value = dtp\_U\_LicenceDateExpiration.Value.ToString("dd/MM/yyyy");

cmd.Parameters.Add("@Nb\_variables", SqlDbType.Int).Value = Convert.ToInt32(tb\_U\_LicenceNbVariables.Text);

cmd.Parameters.Add("@Nb\_equipements", SqlDbType.Int).Value = Convert.ToInt32(tb\_U\_LicenceNbEquipements.Text);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

updatePassed = true;

}

catch (Exception err)

{

MessageBox.Show("Erreur paramêtres licence : verifier que tous les paramêtres sont bien remplies. ⚠Fermeture du mode Modifications⚠");

con.Close();

}

#endregion

if (updatePassed == true)

{

#region suppression des anciennes fonction de la licence

SqlCommand cmd2 = new SqlCommand("Base\_ILM.dbo.delete\_LicenceFonctions", con);

cmd2.CommandType = CommandType.StoredProcedure;

cmd2.Parameters.Add("@Licence\_ID", SqlDbType.Int).Value = Convert.ToInt32(tb\_R\_LicenceID.Text);

con.Open();

cmd2.ExecuteNonQuery();

con.Close();

#endregion

#region Ajout des fonctions de la licence

foreach (DataGridViewRow row in dgv\_U\_LicenceFonctions.Rows)

{

SqlCommand cmd3 = new SqlCommand("Base\_ILM.dbo.create\_Liaison\_Licence\_Fonction", con);

cmd3.CommandType = CommandType.StoredProcedure;

try

{

int Licence\_ID = Convert.ToInt32(tb\_U\_LicenceID.Text);

int Code\_fonction = Convert.ToInt32(row.Cells[0].Value);

cmd3.Parameters.Add("@Licence\_ID", SqlDbType.Int).Value = Licence\_ID;

cmd3.Parameters.Add("@Code\_fonction", SqlDbType.Int).Value = Code\_fonction;

con.Open();

cmd3.ExecuteNonQuery();

con.Close();

}

catch (Exception err)

{

MessageBox.Show("Erreur création fonction pour la licence : " + err.Message + " ⚠Fermeture du mode Ajout⚠");

con.Close();

}

}

#endregion

}

fill\_dgv\_Licences();

dgv\_R\_LicenceFonctions.Rows.Clear();

dgv\_U\_FonctionsDisponibles.Rows.Clear();

dgv\_U\_LicenceFonctions.Rows.Clear();

adaptDataGridView();

tc\_Licences.SelectedIndex = 0;

btn\_C\_Licence.Visible = true;

#region vider les champs

tb\_U\_LicenceNom.Text = "";

cb\_U\_LicenceIntegrateur.SelectedItem = null;

cb\_U\_LicenceClient.SelectedItem = null;

dtp\_R\_LicenceDateExpiration.Value = DateTime.Now;

tb\_U\_LicenceNbEquipements.Text = "";

tb\_U\_LicenceNbVariables.Text = "";

cb\_U\_LicenceClient.Text = "";

cb\_U\_LicenceIntegrateur.Text = "";

#endregion

}

//Boutons de selection des fonctions

private void selectFonction(object sender, EventArgs e)

{

if (dgv\_U\_FonctionsDisponibles.SelectedRows.Count > 0)

{

//la fonction choisi

DataGridViewRow row = (DataGridViewRow)dgv\_U\_FonctionsDisponibles.SelectedRows[0];

//la fonction dans le nouveau datagridview

int rowIndex = dgv\_U\_LicenceFonctions.Rows.Add();

dgv\_U\_LicenceFonctions.Rows[rowIndex].Cells[0].Value = row.Cells[0].Value;

dgv\_U\_LicenceFonctions.Rows[rowIndex].Cells[1].Value = row.Cells[1].Value;

dgv\_U\_LicenceFonctions.Rows[rowIndex].Cells[2].Value = row.Cells[2].Value;

//supprimer la fonction dans l'ancien datagridview

dgv\_U\_FonctionsDisponibles.Rows.RemoveAt(row.Index);

}

}

private void deselectFonction(object sender, EventArgs e)

{

if (dgv\_U\_LicenceFonctions.SelectedRows.Count > 0)

{

//la fonction choisi

DataGridViewRow row = (DataGridViewRow)dgv\_U\_LicenceFonctions.SelectedRows[0];

//la fonction dans le nouveau datagridview

int rowIndex = dgv\_U\_FonctionsDisponibles.Rows.Add();

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[0].Value = row.Cells[0].Value;

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[1].Value = row.Cells[1].Value;

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[2].Value = row.Cells[2].Value;

//supprimer la fonction dans l'ancien datagridview

dgv\_U\_LicenceFonctions.Rows.RemoveAt(row.Index);

}

}

private void selectAllFonction(object sender, EventArgs e)

{

foreach (DataGridViewRow row in dgv\_U\_FonctionsDisponibles.Rows)

{

int rowIndex = dgv\_U\_LicenceFonctions.Rows.Add();

dgv\_U\_LicenceFonctions.Rows[rowIndex].Cells[0].Value = row.Cells[0].Value;

dgv\_U\_LicenceFonctions.Rows[rowIndex].Cells[1].Value = row.Cells[1].Value;

dgv\_U\_LicenceFonctions.Rows[rowIndex].Cells[2].Value = row.Cells[2].Value;

}

dgv\_U\_FonctionsDisponibles.Rows.Clear();

}

private void deselectAllFonction(object sender, EventArgs e)

{

foreach (DataGridViewRow row in dgv\_U\_LicenceFonctions.Rows)

{

int rowIndex = dgv\_U\_FonctionsDisponibles.Rows.Add();

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[0].Value = row.Cells[0].Value;

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[1].Value = row.Cells[1].Value;

dgv\_U\_FonctionsDisponibles.Rows[rowIndex].Cells[2].Value = row.Cells[2].Value;

}

dgv\_U\_LicenceFonctions.Rows.Clear();

}

//appuie sur le bouton retour

private void retour(object sender, EventArgs e)

{

#region montrer les champs

btn\_U\_Licence.Visible = true;

lb\_V1.Visible = true;

lb\_V2.Visible = true;

lb\_V3.Visible = true;

lb\_V4.Visible = true;

lb\_V5.Visible = true;

lb\_V6.Visible = true;

lb\_V7.Visible = true;

lb\_V8.Visible = true;

lb\_V9.Visible = true;

tb\_R\_LicenceID.Visible = true;

tb\_R\_LicenceIntegrateur.Visible = true;

tb\_R\_LicenceClient.Visible = true;

tb\_R\_LicenceIntegrateur.Visible = true;

tb\_R\_LicenceNbEquipements.Visible = true;

tb\_R\_LicenceNbVariables.Visible = true;

tb\_R\_LicenceNom.Visible = true;

dtp\_R\_LicenceDateCreation.Visible = true;

dtp\_R\_LicenceDateExpiration.Visible = true;

btn\_C\_Licence.Visible = true;

btn\_U\_Licence.Visible = true;

#endregion

#region vider les champs

tb\_U\_LicenceNom.Text = "";

cb\_U\_LicenceIntegrateur.SelectedItem = null;

cb\_U\_LicenceClient.SelectedItem = null;

dtp\_R\_LicenceDateExpiration.Value = DateTime.Now;

tb\_U\_LicenceNbEquipements.Text = "";

tb\_U\_LicenceNbVariables.Text = "";

cb\_U\_LicenceClient.Text = "";

cb\_U\_LicenceIntegrateur.Text = "";

#endregion

fill\_dgv\_Licences();

dgv\_R\_LicenceFonctions.Rows.Clear();

dgv\_U\_FonctionsDisponibles.Rows.Clear();

dgv\_U\_LicenceFonctions.Rows.Clear();

adaptDataGridView();

tc\_Licences.SelectedIndex = 0;

}

//Appuie sur le bouton supprimer

private void delete\_Licence(object sender, EventArgs e)

{

if (dgv\_R\_Licences.SelectedRows.Count > 0)

{

DialogResult dr = MessageBox.Show("Vous êtes sur le point de supprimer la licence suivante : " + dgv\_R\_Licences.SelectedRows[0].Cells[0].Value.ToString() + " " + dgv\_R\_Licences.SelectedRows[0].Cells[1].Value.ToString() + "\n\nconfirmez-vous cette action ?", "Suppression licence", MessageBoxButtons.YesNo);

if (dr == DialogResult.Yes)

{

SqlCommand cmd = new SqlCommand("Base\_ILM.dbo.delete\_Licence", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@Licence\_ID", SqlDbType.Int).Value = Convert.ToInt32(dgv\_R\_Licences.SelectedRows[0].Cells[0].Value);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

fill\_dgv\_Licences();

dgv\_R\_LicenceFonctions.Rows.Clear();

}

}

}

//Appuie sur le bouton annuler modifications

private void annuler\_Modifications(object sender, EventArgs e)

{

if (btn\_C\_Licence.Visible == true)

{

DialogResult dr = MessageBox.Show("Vous êtes sur le point d'annuler les modifications \nconfirmez-vous cette action ?", "Annuler modifications", MessageBoxButtons.YesNo);

if (dr == DialogResult.Yes)

{

fill\_dgv\_FonctionsDipsonibles();

dgv\_U\_LicenceFonctions.Rows.Clear();

adaptDataGridView();

tb\_U\_LicenceNom.Text = "";

cb\_U\_LicenceIntegrateur.SelectedItem = null;

cb\_U\_LicenceClient.SelectedItem = null;

dtp\_R\_LicenceDateExpiration.Value = DateTime.Now;

tb\_U\_LicenceNbEquipements.Text = "";

tb\_U\_LicenceNbVariables.Text = "";

}

}

if (btn\_U\_Licence.Visible == true)

{

DialogResult dr = MessageBox.Show("Vous êtes sur le point d'annuler les modifications \nconfirmez-vous cette action ?", "Annuler modifications", MessageBoxButtons.YesNo);

if (dr == DialogResult.Yes)

{

int Licence\_ID = Convert.ToInt32(tb\_R\_LicenceID.Text);

foreach (Licence licence in listeLicences)

{

if (Licence\_ID == licence.Licence\_ID)

{

#region valeurs de la licence

tb\_R\_LicenceID.Text = licence.Licence\_ID.ToString();

tb\_R\_LicenceNom.Text = licence.Nom.ToString();

tb\_R\_LicenceIntegrateur.Text = licence.Code\_client.ToString() + "." + get\_Licence\_Client(licence).Nom;

tb\_R\_LicenceClient.Text = licence.Code\_integrateur.ToString() + "." + get\_Licence\_Integrateur(licence).Nom;

dtp\_R\_LicenceDateCreation.Value = licence.Date\_creation;

dtp\_R\_LicenceDateExpiration.Value = licence.Date\_expiration;

tb\_R\_LicenceNbEquipements.Text = licence.Nb\_equipements.ToString();

tb\_R\_LicenceNbVariables.Text = licence.Nb\_variables.ToString();

#endregion

#region valeurs modifiables de la licence

tb\_U\_LicenceID.Text = licence.Licence\_ID.ToString();

tb\_U\_LicenceNom.Text = licence.Nom.ToString();

foreach (string Code\_client in cb\_U\_LicenceIntegrateur.Items)

{

if (licence.Code\_client == Convert.ToInt32(Code\_client.ToString().Split('.')[Code\_client.ToString().Split('.').Count() - 1]))

{

cb\_U\_LicenceIntegrateur.SelectedItem = Code\_client;

}

}

foreach (string Code\_integrateur in cb\_U\_LicenceClient.Items)

{

if (licence.Code\_integrateur == Convert.ToInt32(Code\_integrateur.ToString().Split('.')[Code\_integrateur.ToString().Split('.').Count() - 1]))

{

cb\_U\_LicenceClient.SelectedItem = Code\_integrateur;

}

}

dtp\_U\_LicenceDateCreation.Value = licence.Date\_creation;

dtp\_U\_LicenceDateExpiration.Value = licence.Date\_expiration;

tb\_U\_LicenceNbEquipements.Text = licence.Nb\_equipements.ToString();

tb\_U\_LicenceNbVariables.Text = licence.Nb\_variables.ToString();

#endregion

}

}

}

fill\_dgv\_LicenceNoFonctions();

fill\_dgv\_LicenceFonctions();

adaptDataGridView();

}

}

//Appuie sur le bouton de Génération de la licence

private void generateLicence(object sender, EventArgs e)

{

if (dgv\_R\_Licences.SelectedRows.Count > 0)

{

string LicenceString = "";

//Informations de la licence

LicenceString += dgv\_R\_Licences.SelectedRows[0].Cells[0].Value.ToString();

LicenceString += "|" + dgv\_R\_Licences.SelectedRows[0].Cells[1].Value.ToString();

LicenceString += "|" + dgv\_R\_Licences.SelectedRows[0].Cells[2].Value.ToString();

LicenceString += "|" + dgv\_R\_Licences.SelectedRows[0].Cells[3].Value.ToString();

LicenceString += "|" + Convert.ToDateTime(dgv\_R\_Licences.SelectedRows[0].Cells[5].Value).ToString("dd/MM/yyyy");

LicenceString += "|" + dgv\_R\_Licences.SelectedRows[0].Cells[6].Value.ToString();

LicenceString += "|" + dgv\_R\_Licences.SelectedRows[0].Cells[7].Value.ToString();

//10 emplacements de reserve pour de futurs informations

LicenceString += "|" + "null";

LicenceString += "|" + "null";

LicenceString += "|" + "null";

LicenceString += "|" + "null";

LicenceString += "|" + "null";

LicenceString += "|" + "null";

LicenceString += "|" + "null";

LicenceString += "|" + "null";

LicenceString += "|" + "null";

LicenceString += "|" + "null";

//Fonctions de la licence

LicenceString += "|" + dgv\_R\_LicenceFonctions.Rows.Count.ToString();

foreach (DataGridViewRow row in dgv\_R\_LicenceFonctions.Rows)

{

LicenceString += "|" + row.Cells[0].Value.ToString();

}

string key = "";

using (Aes aesAlgorithm = Aes.Create())

{

aesAlgorithm.KeySize = 128;

aesAlgorithm.GenerateKey();

string keyBase64 = Convert.ToBase64String(aesAlgorithm.Key);

key = keyBase64;

}

Crypteur crypteur = new Crypteur();

String encryptedString = crypteur.EncryptString(key, LicenceString);

selectLicenceFolder.ShowDialog();

String path = selectLicenceFolder.SelectedPath;

String name = dgv\_R\_Licences.SelectedRows[0].Cells[1].Value.ToString();

String fileName = path + "/" + name + ".IconeLic";

if (File.Exists(fileName))

{

MessageBox.Show("Nom de fichier déja utilisé");

return;

}

using (StreamWriter sw = File.CreateText(fileName))

{

sw.WriteLine(encryptedString);

sw.WriteLine(key);

}

//pour débugger (voir si la chaine reste la même apres le décryptage

//MessageBox.Show(crypteur.DecryptString(key, encryptedString));

MessageBox.Show("Licence généré avec succé");

}

}

//permet de récuperer le client/integrateur a partir de la licence

public Client get\_Licence\_Client(Licence licence)

{

foreach (Client client in listeClients)

{

if (client.Code\_client == licence.Code\_client)

{

return client;

}

}

return null;

}

public Integrateur get\_Licence\_Integrateur(Licence licence)

{

foreach (Integrateur integrateur in listeIntegrateurs)

{

if (integrateur.Code\_integrateur == licence.Code\_integrateur)

{

return integrateur;

}

}

return null;

}

//Style datagridviews

private void adaptDataGridView()

{

#region reset des style

dgv\_R\_Licences.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_Licences.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_Licences.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_Licences.Columns[3].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_Licences.Columns[4].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_Licences.Columns[5].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_Licences.Columns[6].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_Licences.Columns[7].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_LicenceFonctions.DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_R\_LicenceFonctions.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_LicenceFonctions.Columns[0].DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_R\_LicenceFonctions.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_LicenceFonctions.Columns[1].DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_R\_LicenceFonctions.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_U\_FonctionsDisponibles.DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_U\_FonctionsDisponibles.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_U\_FonctionsDisponibles.Columns[0].DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_U\_FonctionsDisponibles.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_U\_FonctionsDisponibles.Columns[1].DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_U\_FonctionsDisponibles.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_U\_LicenceFonctions.DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_U\_LicenceFonctions.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_U\_LicenceFonctions.Columns[0].DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_U\_LicenceFonctions.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_U\_LicenceFonctions.Columns[1].DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_U\_LicenceFonctions.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_Integrateurs.DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_R\_Integrateurs.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_Integrateurs.Columns[0].DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_R\_Integrateurs.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

dgv\_R\_Integrateurs.Columns[1].DefaultCellStyle.WrapMode = DataGridViewTriState.NotSet;

dgv\_R\_Integrateurs.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.None;

#endregion

#region Stylisation des datagridviews

dgv\_R\_Licences.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgv\_R\_Licences.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgv\_R\_Licences.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgv\_R\_Licences.Columns[3].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgv\_R\_Licences.Columns[4].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgv\_R\_Licences.Columns[5].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgv\_R\_Licences.Columns[6].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgv\_R\_Licences.Columns[7].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgv\_R\_LicenceFonctions.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_R\_LicenceFonctions.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;

dgv\_R\_LicenceFonctions.Columns[0].DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_R\_LicenceFonctions.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;

dgv\_R\_LicenceFonctions.Columns[1].DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_R\_LicenceFonctions.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgv\_U\_FonctionsDisponibles.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_U\_FonctionsDisponibles.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;

dgv\_U\_FonctionsDisponibles.Columns[0].DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_U\_FonctionsDisponibles.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;

dgv\_U\_FonctionsDisponibles.Columns[1].DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_U\_FonctionsDisponibles.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgv\_U\_LicenceFonctions.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_U\_LicenceFonctions.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;

dgv\_U\_LicenceFonctions.Columns[0].DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_U\_LicenceFonctions.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;

dgv\_U\_LicenceFonctions.Columns[1].DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_U\_LicenceFonctions.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dgv\_R\_Integrateurs.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_R\_Integrateurs.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;

dgv\_R\_Integrateurs.Columns[0].DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_R\_Integrateurs.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;

dgv\_R\_Integrateurs.Columns[1].DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dgv\_R\_Integrateurs.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

#endregion

}

}

}