



Ingénierie & systèmes automatisés

## COMPTE RENDU STAGE

*Année 2023-2024*

### SEMAINE 5

30/01/2023 – 03/01/2023

FONCTION	NOM
Étudiant stagiaire	Allan ESCOLANO
Représentante entreprise	Grégory MEUNIER
Tuteur en entreprise	Jean-François DANCKAERT
Tuteur professionnel	Jean-François DANCKAERT
Représentante Lycée	Patricia BUËR
Professeur chargé du suivi	David ROUMANET

## ***Sommaire / Summary***

# **Table des matières**

1.	Présentation de l'entreprise.....	3
2.	Mes missions.....	4
2.1.	Gestionnaire de licences. ....	4
2.1.1.	Présentation.....	4
2.1.2.	Modifications BDD.....	5
2.1.3	Changement graphiques.....	7
2.1.4.	Norme et format de la licence. ....	11
3.	Outils utilisés.....	15
4.	Conclusion.....	16
5.	Niko-Niko. ....	17
6.	Annexe codes.....	18
6.1.	Code ICONECT LICENCE MANAGER. ....	18

# 1. Présentation de l'entreprise.

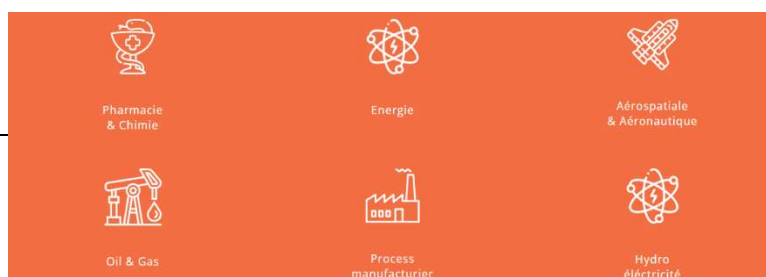


## **ICONE AUTOMATISATION**

**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.



## 2. 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
<b>Interface</b>	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
<b>Manque une page</b>	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
<b>Logos</b>	Aucun logos utiliser	Utiliser des logos
<b>Actions non clair</b>	Certain texte sur les boutons ne décrivait pas bien leur action	Utiliser des termes simple et efficace pour décrire le but principal du bouton
<b>Utilisation de comboBox pour client et intégrateur</b>	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
<b>Fonction générer licence</b>	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
<b>Code fonction</b>	Int	PK, Non null
<b>Nom</b>	Varchar(255)	Not null
<b>Date_update</b>	Datetime	Not null
<b>Description</b>	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
<b>Code_client</b>	Int	PK IDENTITY, Non null
<b>Nom</b>	Varchar(255)	Not null
<b>Date_update</b>	Datetime	Not null
<b>Description</b>	Varchar(max)	Nullable

Table Intégrateurs :

Nom	Type	Contraintes
<b>Code_integrateur</b>	Int	PK IDENTITY, Non null
<b>Nom</b>	Varchar(255)	Not null
<b>Date_update</b>	Datetime	Not null
<b>Description</b>	Varchar(max)	Nullable

Table Licences

Nom	Type	Contraintes
<b>Licence_ID</b>	Int	PK IDENTITY, Non null
<b>Nom</b>	Varchar(255)	Not null
<b>Code_client</b>	Int	FK REFERENCES Table Clients Column Code_client, Not null
<b>Code_integrateur</b>	Int	FK REFERENCES Table Integrateurs Column Code_integrateur, Not null
<b>Date_creation</b>	Datetime	Not null
<b>Date_expiration</b>	Datetime	Not null
<b>Nb_equipements</b>	Int	Not null
<b>Nb_variables</b>	Int	Not null

Table Liaison\_Licence\_Fonctions

Nom	Type	Contraintes
<b>Licence_ID</b>	Int	FK REFERENCES Table Licences Column Licence_ID, Not null
<b>Code fonction</b>	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 quasi totalité de l'interface (Approuvé par le chef le vendredi 3).

Voici la transformation :

### Page d'affichage des licences (Avant-Après) :

Gestionnaire\_Licences\_APP

Licences Integrateurs Clients Fonctions

ID_Licence	Date_creation	Date_expiration	Nb_equipements	Nb_variables	Checksum	Code_integrateur	Code_client	Code_fonction1	Code_fonction2	Code_fonction3
1	25/01/2023 14:05	25/01/2023	10	10	CS	7	1	1		
12	26/01/2023	10/10/3065	9000	9000	ts	3	4	3		
13	26/01/2023	30/05/2029	10	10	10	3	4	3		
14	26/01/2023	26/01/2023	150	150	tl	6	6		5	4
15	26/01/2023	26/01/2023	632	410	tl	3	7	3	5	6

Date de création: 03/02/2023, Date d'expiration: vendredi 3 février 2023, Nombre d'équipements: , Nombre de variables: , Checksum: , Code\_integrateur: , Code\_client: , Mettre à 0, Mettre fonctions à 0

Code fonction 1: , Code fonction 2: , Code fonction 3: , Code fonction 4: , Code fonction 5: , Code fonction 6: , Code fonction 7: , Code fonction 8: , Code fonction 9: , Code fonction 10: , Code fonction 11: , Code fonction 12: , Code fonction 13: , Code fonction 14: , Code fonction 15: , Code fonction 16: , Code fonction 17: , Code fonction 18: , Code fonction 19: , Code fonction 20: , Ajouter la licence

Sélectionner un integrateur: , Sélectionner un client: , Sélectionner une fonction:

Code_integrateur	Nom	Date_update	Descr
1	ICONE	26/01/2023	Entrepre
3	TagProduct	26/01/2023	Entrepre
4	FournisseurLicen...	26/01/2023	Fournis
5	Integrateur	26/01/2023	Ceci et
6	IN-TECH	26/01/2023	Fime c

Code_client	Nom	Date_update	Descr
1	testClient	25/01/2023 14:00	Test un
4	MonsieurToto	26/01/2023	Il veut
5	SNCF	26/01/2023	Fournis
6	BanqueDeFrance	26/01/2023	La ban
7	ICONE	26/01/2023	Entrepre

Code_fonction	Nom	Date_update	Descr
1	TestFonction	25/01/2023 14:02	Test un
3	Admin	26/01/2023	Donne
4	Accès variables	26/01/2023	Ceci d
5	Accès statistiques	26/01/2023	Ceci d
6	Accès sauvegard...	26/01/2023	Ceci d

ICONNECT LICENCE MANAGER

Licences Integrateurs Clients Fonctions A propos ILM V1.0.0

Licences :

ID	Nom	Nom de l'integrateur	Nom du client	Date de création	Date d'expiration	Nombre de variables	Nombre d'équipements
1	Administrateur	ICONE	ICONE	01/02/2023	01/02/2024	9999	9999
2	Niveau1	JF.DANCKAERT	ICONE	02/02/2023	02/02/2024	10	10
3	Niveau2	JF.DANCKAERT	SIEMENS	02/02/2023	02/02/2024	50	50
4	Niveau3	JF.DANCKAERT	SIEMENS	02/02/2023	02/02/2024	100	100
5	AdminMaintenance	ICONE	SIEMENS	02/02/2023	02/02/2024	75	45

Fonctions de la licence :

Code	Nom	Description
1	Paramétrage Clients	test
2	Paramétrage Sites	test
3	Paramétrage Ateliers	test
4	Paramétrage Equipements	test
5	Paramétrage Variables	test
6	Aide à la maintenance	test
7	Rapport	test
8	Notification	test

Création Modifications Générer la licence Supprimer la licence





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

Code_integrateur	Nom	Date_ajout	Description
1	ICONE	26/01/2023	Entreprise automatisée
3	TagProduct	26/01/2023	Entreprise pouvant fournir des licences
4	Technosau/Labex	26/01/2023	Fournisseur officiel de licences
5	Integrateur	26/01/2023	Ceci est un Intégrateur
6	R+TECH	26/01/2023	Faire d'entreprise
7	Beract	30/01/2023	On sait pas
8	Beract	30/01/2023	On sait pas
9	Beract	30/01/2023	On sait pas

Nom:   
 Date de mise à jour:   
 Description:   
 Bouton:

**AJOUTS**

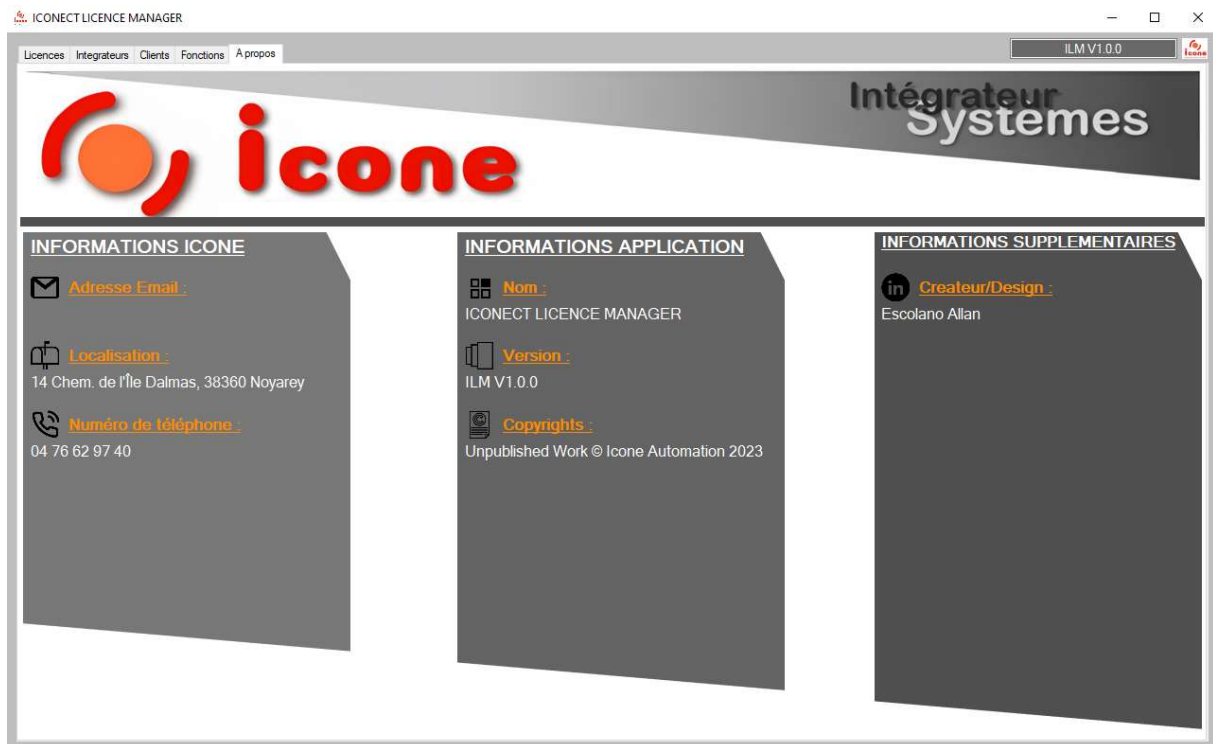
Données de l'intégrateur: Nom:  ICONE, Date de mise à jour:  26/01/2023, Description:  Entreprise automatisée  
 Modifications: Nom:  ICONE, Date de mise à jour:  03/02/2023, Description:  Entreprise automatisée  
 Boutons: , ,   
 Code Intégrateur:

**MODIFICATIONS**

Intégrateurs:   
 Boutons: , ,

Intégrateur au cours d'effacement: Nom:  ICONE, Date de mise à jour:  26/01/2023, Description:  Entreprise automatisée  
 Modifications: Nom:  ICONE, Date de mise à jour:  03/02/2023, Description:  Entreprise automatisée  
 Boutons: , ,

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 chaîne d'information crypter ainsi que la clé symétrique de décryptage.
- Chaque partie de la chaîne 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 IConnect®.

**Par exemple, la chaîne 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.

### 3. Outils utilisés.

Nom de l'outil	Fonction de l'outil
<b>Visual Studio 2019</b>	Espace de développement avec multiple langage de programmations
<b>VMWare Workstation 16</b>	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.
<b>Microsoft SQL Server</b>	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.

## 4. 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.


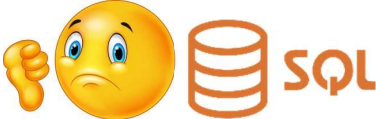



Se mettre ce point de vue permet vraiment de se mettre dans la peau d'un utilisateur et permet de créer des choses plaisantes visuel et en fonctionnalités aussi.

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).



## 5. Niko-Niko.

JOUR	RESSENTI	TACHE
Lundi		Jour de reunion (Stresse un peu quand même)
Mardi		Jour de la nouvelle BDD (SQL pas ouf)
Mercredi		Jour de la nouvelle interface (Choix artistiques mis en question)
Jeudi		Fonctionnement CRUD des Licences (Un CRUD classique)
Vendredi		Fonctions annexes/Page A propos (Validation chef + fin de stage)

## 6. 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;

/*
    TODO LIST

    -Faire CRUD pour CLIENTS
    -Faire CRUD pour INTEGRATEURS
    -Faire CRUD pour FONCTIONS
    -Faire page A propos
    -Rajouter logos

*/

/*
    FONCTIONS TERMINEES

    -CRUD LICENCES
    -Fonctionnement 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 d'affichage des Fonctions de la Licence
au datagridview après que les licence ont charger
dgv_R_Licences.ClearSelection();
dgv_R_Licences.SelectionChanged += SelectedLicenceChanged;
}

//Quand une licence est sélectionné, 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_FonctionsDisponibles();

//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_FonctionsDisponibles()
{
    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
derniere 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_Licence
Client.SelectedItem.ToString().Split('.').Count() - 1]);
            cmd.Parameters.Add("@Code_client", SqlDbType.Int).Value =
Convert.ToInt32(cb_U_LicenceIntegrateur.SelectedItem.ToString().Split('.')[cb_U_Li
cenceIntegrateur.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 paramètres licence : verifier que tous
les paramètres sont bien remplis. ⚠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('.')[0].Split('.')[1]).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('.')[0].Split('.')[1]).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_Licence
Client.SelectedItem.ToString().Split('.').Count() - 1]);
        cmd.Parameters.Add("@Code_client", SqlDbType.Int).Value =
Convert.ToInt32(cb_U_LicenceIntegrateur.SelectedItem.ToString().Split('.')[cb_U_Li
cenceIntegrateur.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 paramètres licence : verifier que tous les
paramètres 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 + " ΔFermeture du mode AjoutΔ");
                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 + ".IcôneLic";

    if (File.Exists(fileName))
    {
        MessageBox.Show("Nom de fichier déjà utilisé");
        return;
    }

    using (StreamWriter sw = File.CreateText(fileName))
    {
        sw.WriteLine(encryptedString);
        sw.WriteLine(key);
    }

    //pour débayer (voir si la chaîne reste la même après le
décryptage
    //MessageBox.Show(crypteur.DecryptString(key, encryptedString));

    MessageBox.Show("Licence généré avec succès");
}
}

//permet de récupérer le client/intégrateur à 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 Intégrateur get_Licence_Intégrateur(Licence licence)
{
    foreach (Intégrateur intégrateur in listeIntégrateurs)
    {
        if (intégrateur.Code_intégrateur == licence.Code_intégrateur)
        {
            return intégrateur;
        }
    }

    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
    }
}
```