|  |  |
| --- | --- |
| **Compte rendu STAGE**  ***Année 2023-2024*** | **SEMAINE 1**  ***03/01/2023 – 07/01/2023*** |

***Sommaire / Summary***

Table des matières

[1. Présentation de l’entreprise. 3](#_Toc122102105)

[2. Mes missions. 4](#_Toc122102106)

[2.1 Gestionnaire de licence. 4](#_Toc122102107)

[2.1.1 Analyse des ressources. 4](#_Toc122102108)

[2.1.2 Modification des ressources. 6](#_Toc122102109)

[2.1.3 Mise en œuvre du projet. 7](#_Toc122102110)

[2.2 Encrypteur de licences. 21](#_Toc122102111)

[2.2.1 Ressource supplémentaire. 22](#_Toc122102112)

[2.2.2 Mise œuvre du projet. 23](#_Toc122102113)

[2.3 Implémentation décryptage sur Wonderware/AVEVA. 31](#_Toc122102114)

[2.3.1 Ressources, contexte et technologies. 31](#_Toc122102115)

[2.3.2 Développement logiciel et découverte. 33](#_Toc122102116)

[3. Outils utilisés 37](#_Toc122102117)

[4. Conclusion. 38](#_Toc122102118)

[5. Niko-Niko. 39](#_Toc122102119)

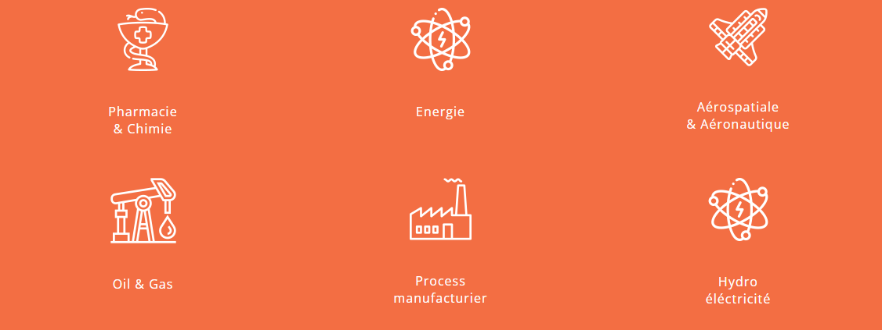
# Présentation de l’entreprise.



**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.



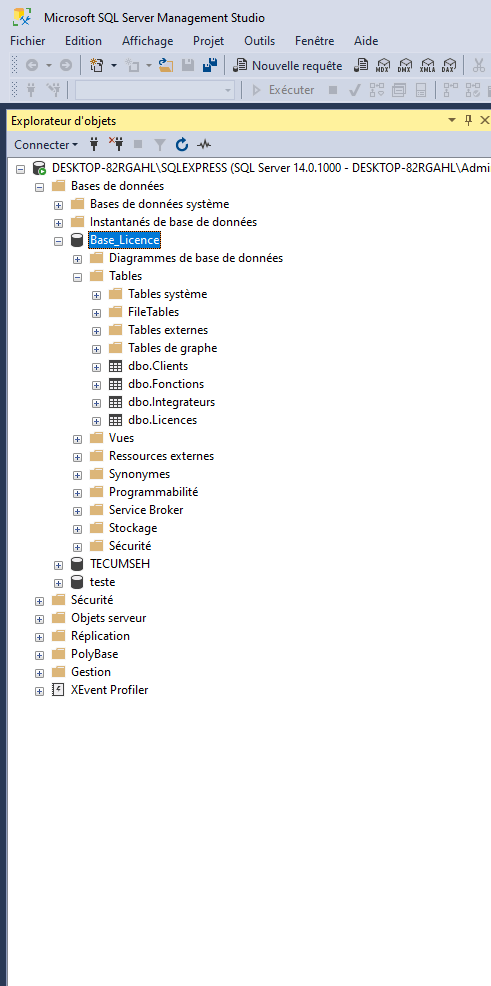
# Mes missions.

## Gestionnaire de licence.

Avec les informations de mon tuteur, je me suis vu affecter comme première mission : Développement d’une application graphique en C# exploitant une base de données SQL EXPRESS dans le but de gérer une base contenant des informations sur les licences d’un logiciel et permettant des en créer, les voir, les modifier et les supprimer.

Pour effectuer ma tâche, j’ai eu comme ressource la base de données avec jeu de test, une photocopie de la base de données avec ses tables et un schéma approximatif de l’interface attendu.

### Analyse des ressources.

Voici la base de données situé sur SQL Server Management Studio.

Une base de 4 tables contenants les informations nécessaires pour les licences (Fonction contient la fonction de la licence, le client est le détenteur de la licence et l’intégrateur est l’intégrateur de la licence)

Représentation de la photocopie fournie :

Table licences

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Date (2 digits) | Code intégrateur (3 digits) | Code Client (5 digits) | ID unique (6 digits) | Nb équipements (6 digits) | Nb variable (6 digits) | Date d’expiration (6 digits) | Checksum (2 digits |

Table intégrateur

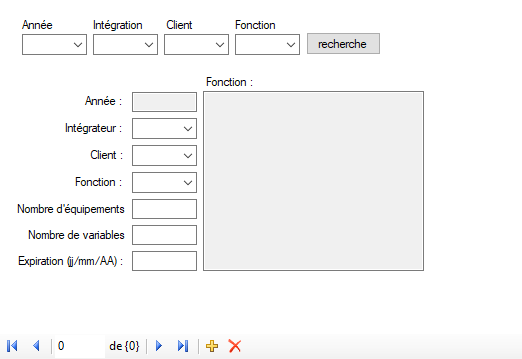
|  |  |  |  |
| --- | --- | --- | --- |
| Code intégrateur (3 digits) | Nom\_Integrateur (Varchar(250)) | Date de mise à jour (Datetime2) | Description (Varchar(250)) |

Table client

|  |  |  |  |
| --- | --- | --- | --- |
| Code Client (6 digits) | Nom\_Client (Varchar(250)) | Date de mise à jour (Datetime2) | Description (Varchar(250)) |

Table fonction

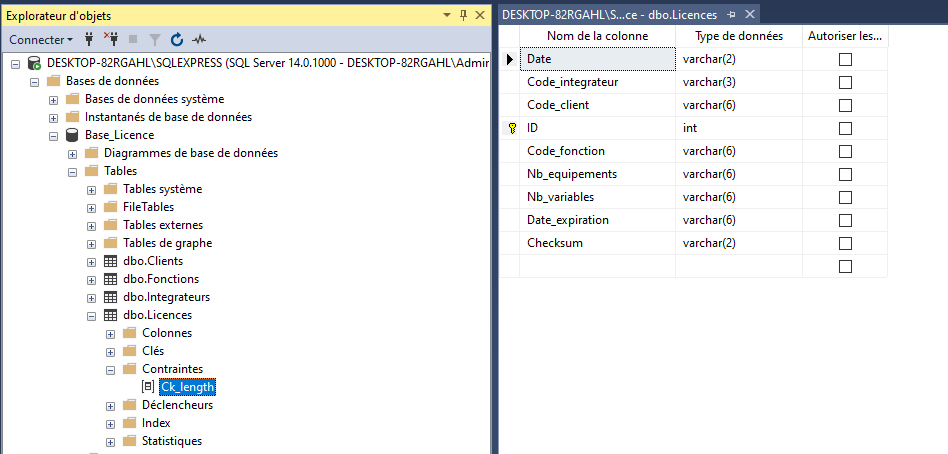
|  |  |  |  |
| --- | --- | --- | --- |
| Code Fonction (6 digits) | Nom\_Fonction (Varchar(250)) | Date de mise à jour (Datetime2) | Description (Varchar(250)) |



### Modification des ressources.

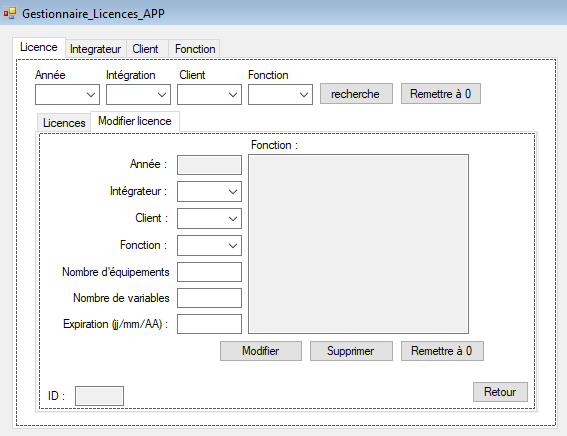
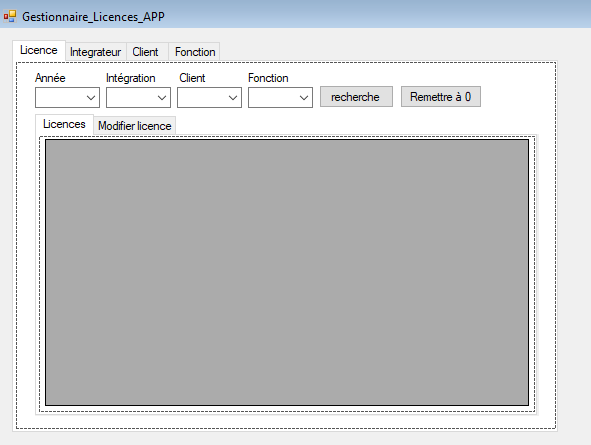
Suite a l’analyse des documents fournis, j’y ai repérer quelque potentiel problème et y ai proposé des solutions.

Premièrement, pour la base de données, j’ai proposé un changement sur les données.  
Tout d’abord un système d’augmentation incrémenter automatiquement pour l’ID des licences plutôt qu’un ID de 6 chiffres rentrer en dur. (Changement approuvé ✓)

Ensuite, pour assurer la bonne saisie des données dans l’application, j’ai proposé la mise en place d’une clé de contraintes pour les données de la licence forçant une saisie d’un format précis. (Changement approuvé ✓)

Par la suite, j’ai aussi proposé un changement niveau interface pour obtenir quelque chose de plus clair et versatile. (Changement approuvé ✓)

Vue et CRUD maintenant possible.

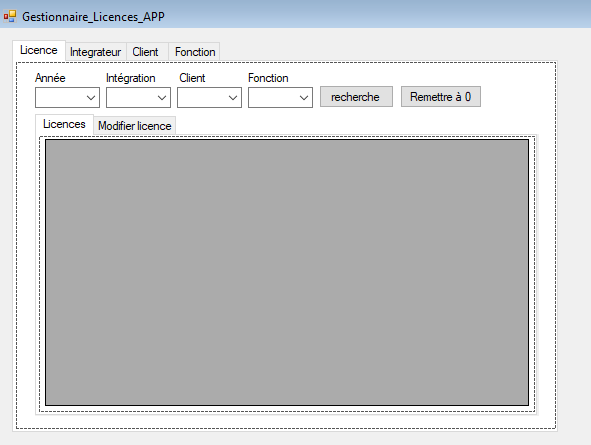


Après avoir organiser mes ressources pour permettre un début de projet plus concret, j’ai débuté sa création avec Visual Studio 2019 en C# WinForm.

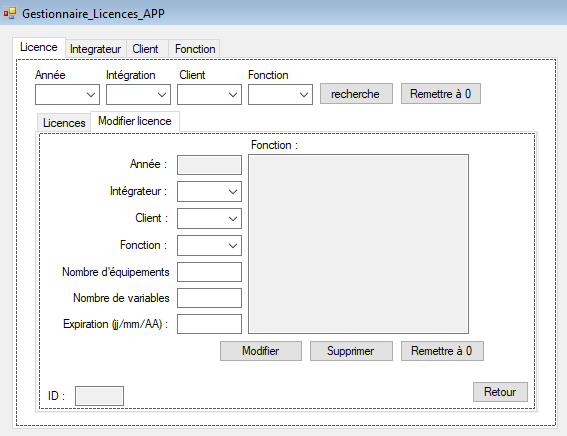
### Mise en œuvre du projet.

Pour bien commencé, j’ai tout d’abord fait l’interface graphique avec un style basic mais me permettant d’avoir une bonne base pour commencer le coté fonctionnement.

Cette première fenêtre permet d’afficher les licences sous forme d’un tableau d’argument avec l’objet DataGridView ainsi qu’un espace pour le filtrage des données du tableau.



Pour la deuxième fenêtre de notre interface, nous avons un CRUD basique avec l’affichage des valeurs de la licence, la description de la fonction et toute les informations nécessaires pour une CRUD précis.  
Nous avons aussi 3 boutons, un pour appliquer les modifications apporté a la licence, un pour supprimé la licence, et un qui permet de remettre les données de la licence a son état original avant la modification en cas d’erreur de saisie de l’utilisateur qui souhaiterai retrouve les informations originales de la licence



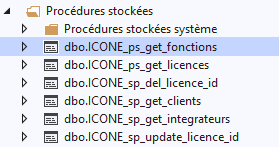
Après avoir mis en place cette interface, je me suis lancé dans son développement fonctionnel.

Pour commencer le codage de manière organisé, j’ai effectué une liste de fonction à faire dans un ordre me permettant de ne pas me perdre en cours de route.

* Récupérer les données dans les tables de la base de données avec une procédure stocker.
* Créer des objets me permettant d’instancier un objet contenant les informations des tables respectives aux objets (Exemple : Objet licence -> table licence).
* Stocker mes objets dans un tableau respectif à chaque objet (Exemple ; lesLicences() -> chaque objet licence).
* Alimenter mon DataGridView avec les données de ma requête.
* Avec un évènement de double clique sur une ligne, récupérer les données et les afficher dans la page de modification

Pour la suite des évènements tel que le filtrage, la suppression, la modification et les autres boutons, l’ordres de développement n’importait pas.

Procédures stockées :



Voici un exemple des ‘’ICONE\_ps\_get\_nomVariable’’ qui est le même pour tout les get a exception de la donnée récupérée :

CREATE PROCEDURE ICONE\_ps\_get\_fonctions

AS

SELECT \* FROM Fonctions

RETURN 0

ICONE\_sp\_del\_licence\_id :

CREATE PROCEDURE [ICONE\_sp\_del\_licence\_id]

@id int = 0

AS

DELETE FROM Licences

WHERE @id = Licences.ID

RETURN 0

ICONE\_sp\_update\_licence\_id :

CREATE PROCEDURE ICONE\_sp\_update\_licence\_id

@id int = 0,

@codeIntegrateur int = 0,

@codeClient int = 0,

@codeFonction int = 0,

@nbEquipements int = 0,

@nbVariables int = 0,

@dateExpiration int = 0

AS

UPDATE Licences

SET Code\_integrateur = @codeIntegrateur,

Code\_client = @codeClient,

Code\_fonction = @codeFonction,

Nb\_equipements = @nbEquipements,

Nb\_variables = @nbVariables,

Date\_expiration = @dateExpiration

WHERE ID = @id

RETURN 0

Les différents objets pour ces données.

Licence.cs :

class Licence

{

//Definition de nos arguments pour la classe

public int date;

public int codeIntegrateur;

public int codeClient;

public int ID;

public int codeFonction;

public int nbEquipement;

public int nbVariable;

public int dateExpiration;

public int checksum;

//Creation du constructeur permettant d'instancier une avec toutes ses valeurs nécessaires

public Licence(int date, int codeIntegrateur, int codeClient, int ID, int codeFonction, int nbEquipement, int nbVariable, int dateExpiration, int checksum)

{

this.date = date;

this.codeIntegrateur = codeIntegrateur;

this.codeClient = codeClient;

this.ID = ID;

this.codeFonction = codeFonction;

this.nbEquipement = nbEquipement;

this.nbVariable = nbVariable;

this.dateExpiration = dateExpiration;

this.checksum = checksum;

}

}

Client.cs :

class Client

{

//Definition de nos arguments pour la classe

public int codeClient;

public String nom;

public DateTime dateUpdate;

public String description;

//Creation du constructeur permettant d'instancier avec toutes ses valeurs nécessaires

public Client(int codeClient, String nom, DateTime dateUpdate, String description)

{

this.codeClient = codeClient;

this.nom = nom;

this.dateUpdate = dateUpdate;

this.description = description;

}

}

Fonction.cs :

class Fonction

{

//Definition de nos arguments pour la classe

public int codeFonction;

public String nom;

public DateTime dateUpdate;

public String description;

//Creation du constructeur permettant d'instancier avec toutes ses valeurs nécessaires

public Fonction(int codeFonction, String nom, DateTime dateUpdate, String description)

{

this.codeFonction = codeFonction;

this.nom = nom;

this.dateUpdate = dateUpdate;

this.description = description;

}

}

Integrateur.cs :

class Integrateur

{

//Definition de nos arguments pour la classe

public int codeIntegrateur;

public String nom;

public DateTime dateUpdate;

public String description;

//Creation du constructeur permettant d'instancier avec toutes ses valeurs nécessaires

public Integrateur(int codeIntegrateur, String nom, DateTime dateUpdate, String description)

{

this.codeIntegrateur = codeIntegrateur;

this.nom = nom;

this.dateUpdate = dateUpdate;

this.description = description;

}

}

Après avoir créé nos objets et avoir nos données, nous pouvons commencer l’alimentation de l’interface graphique.

Avant toutes fonctions :

SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;Initial Catalog=Base\_Licence;Integrated Security=True;");

//Creation d'une liste d'objet licence

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

//Creation d'une liste d'objet fonction

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

//Creation d'une liste d'objet integrateur

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

//Creation d'une liste d'objet client

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

Au chargement du Form1 :

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

{

//Suppressions des boutons du tabcontrol

licences.Appearance = TabAppearance.FlatButtons;

licences.ItemSize = new Size(0, 1);

licences.SizeMode = TabSizeMode.Fixed;

//Ouverture de la connexion

con.Open();

//Definition de la procédure stockée

SqlCommand cmd = new SqlCommand("Base\_Licence.dbo.ICONE\_ps\_get\_licences", con);

cmd.CommandType = CommandType.StoredProcedure;

//Alimentation de la DataGridView

SqlDataAdapter sd = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

sd.Fill(dt);

DGVLicences.DataSource = dt;

//Calcule du nombre d'année entre aujourd'hui et 2022

int thisYear = DateTime.Now.Year;

int nbrAnnee = thisYear - 2022;

//Alimentation du comboBox

CB\_Annee.Items.Add(2022);

for (int i = 1; i <= nbrAnnee; i++)

{

CB\_Annee.Items.Add(2022 + i);

}

SqlDataReader rdr = cmd.ExecuteReader();

//Utilisation d'un reader qui lis les lignes du résultat de la requete ligne par ligne

while(rdr.Read())

{

//récuperation des valeurs de la ligne puis convertion en int

int date = Convert.ToInt32(rdr["Date"]);

int codeIntegrateur = Convert.ToInt32(rdr["Code\_integrateur"]);

int codeClient = Convert.ToInt32(rdr["Code\_Client"]);

int ID = Convert.ToInt32(rdr["ID"]);

int codeFonction = Convert.ToInt32(rdr["Code\_fonction"]);

int nbEquipement = Convert.ToInt32(rdr["Nb\_equipements"]);

int nbVariable = Convert.ToInt32(rdr["Nb\_variables"]);

int dateExpiration = Convert.ToInt32(rdr["Date\_expiration"]);

int checksum = Convert.ToInt32(rdr["Checksum"]);

//Instanciation d'une licence

Licence uneLicence = new Licence(date, codeIntegrateur, codeClient, ID, codeFonction, nbEquipement, nbVariable, dateExpiration, checksum);

//Ajout de la licence dans la liste des licences

lesLicences.Add(uneLicence);

}

rdr.Close();

//recuperation des fonctions

cmd = new SqlCommand("Base\_Licence.dbo.ICONE\_ps\_get\_fonctions", con);

rdr = cmd.ExecuteReader();

while(rdr.Read())

{

int codeFonction = Convert.ToInt32(rdr["Code\_fonction"]);

String nom = Convert.ToString(rdr["Nom"]);

DateTime dateUpdate = Convert.ToDateTime(rdr["Date\_update"]);

String description = Convert.ToString(rdr["Description"]);

Fonction uneFonction = new Fonction(codeFonction, nom, dateUpdate, description);

lesFonctions.Add(uneFonction);

}

foreach(Fonction uneFonction in lesFonctions)

{

CB\_fonction.Items.Add(uneFonction.nom);

cbModifFonction.Items.Add(uneFonction.nom);

}

rdr.Close();

//recuperation des integrateurs

cmd = new SqlCommand("Base\_Licence.dbo.ICONE\_sp\_get\_integrateurs", con);

rdr = cmd.ExecuteReader();

while (rdr.Read())

{

int codeIntegrateur = Convert.ToInt32(rdr["Code\_integrateur"]);

String nom = Convert.ToString(rdr["Nom"]);

DateTime dateUpdate = Convert.ToDateTime(rdr["Date\_update"]);

String description = Convert.ToString(rdr["Description"]);

Integrateur unIntegrateur = new Integrateur(codeIntegrateur, nom, dateUpdate, description);

lesIntegrateurs.Add(unIntegrateur);

}

foreach (Integrateur unIntegrateur in lesIntegrateurs)

{

CB\_integration.Items.Add(unIntegrateur.nom);

cbModifIntegrateur.Items.Add(unIntegrateur.nom);

}

rdr.Close();

//recuperation des clients

cmd = new SqlCommand("Base\_Licence.dbo.ICONE\_sp\_get\_clients", con);

rdr = cmd.ExecuteReader();

while (rdr.Read())

{

int codeClient = Convert.ToInt32(rdr["Code\_client"]);

String nom = Convert.ToString(rdr["Nom"]);

DateTime dateUpdate = Convert.ToDateTime(rdr["Date\_update"]);

String description = Convert.ToString(rdr["Description"]);

Client unClient = new Client(codeClient, nom, dateUpdate, description);

lesClients.Add(unClient);

}

foreach (Client unClient in lesClients)

{

CB\_client.Items.Add(unClient.nom);

cbModifClient.Items.Add(unClient.nom);

}

rdr.Close();

con.Close();

}

Quand une case du DataGridView est double cliqué :

//Evenement appelé quand une cellule du DataGridView est double clické

public void cellDoubleClicked(Object sender, DataGridViewCellEventArgs args)

{

int row = args.RowIndex; //La ligne clické

int ID = Convert.ToInt32(DGVLicences.Rows[row].Cells[3].Value.ToString());

licences.SelectedIndex = 1;

fillLicenceTab(ID);

}

La fonction fillLicenceTab vu au-dessus qui permet de remplir la page de modification :

public void fillLicenceTab(int ID)

{

foreach (Licence uneLicence in lesLicences)

{

if (uneLicence.ID == ID) //Trouve la licence qui correspond a l'ID de celle clické

{

Licence licenceClicke = uneLicence;

tbModifID.Text = licenceClicke.ID.ToString();

tbModifAnnee.Text = "20" + licenceClicke.date.ToString();

tbModifNbrEquipements.Text = licenceClicke.nbEquipement.ToString();

tbModifNbrVariables.Text = licenceClicke.nbEquipement.ToString();

//Recuperation de la date (la requete ne recupere pas les 0 au debut donc il faut les rajouter

String date = licenceClicke.dateExpiration.ToString();

//Rajout des 0

while (date.Length < 6)

{

date = date.Insert(0, "0");

}

//Formatage de la date

date = date.Insert(2, "/");

date = date.Insert(5, "/");

tbModifDateExpiration.Text = date;

foreach (Fonction uneFonction in lesFonctions) //Recherche la fonction correspondante a celle du codeFonction de la licence clické

{

if (uneFonction.codeFonction == licenceClicke.codeFonction)

{

Fonction fonc = uneFonction;

cbModifFonction.SelectedIndex = cbModifFonction.FindString(fonc.nom);

tbModifShowFonction.Text = fonc.description;

}

}

foreach (Integrateur unIntegrateur in lesIntegrateurs) //Recherche l"integrateur correspondant a celui du codeIntegrateur de la licence clické

{

if (unIntegrateur.codeIntegrateur == licenceClicke.codeIntegrateur)

{

Integrateur integ = unIntegrateur;

cbModifIntegrateur.SelectedIndex = cbModifIntegrateur.FindString(integ.nom);

}

}

foreach (Client unClient in lesClients) //Recherche du client correspondant a celui du codeClient de la licence clické

{

if (unClient.codeClient == licenceClicke.codeClient)

{

Client cli = unClient;

cbModifClient.SelectedIndex = cbModifClient.FindString(cli.nom);

}

}

}

}

}

Le bouton retour de la page de modification :

public void retourBtnClicked(Object sender, EventArgs args)

{

licences.SelectedIndex = 0;

}

Le bouton remettre à 0 de la page de modification :

//remet les données de la page d'édition d'une licence en fonction de l'ID d'origine renseigner dans la page

public void resetLicenceEditPage(Object sender, EventArgs args)

{

int ID = Convert.ToInt32(tbModifID.Text);

fillLicenceTab(ID);

}

Fonction dans le bouton de suppression de la page modification :

public void delLicence(Object sender, EventArgs args)

{

con.Open();

SqlCommand cmd = new SqlCommand("Base\_Licence.dbo.ICONE\_sp\_del\_licence\_id", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@id", SqlDbType.Int);

cmd.Parameters["@id"].Value = Convert.ToInt32(tbModifID.Text);

Int32 rowsAffected = cmd.ExecuteNonQuery();

Console.WriteLine("RowsAffected: {0}", rowsAffected);

//Recharger le tableau des licences

cmd = new SqlCommand("Base\_Licence.dbo.ICONE\_ps\_get\_licences", con);

cmd.CommandType = CommandType.StoredProcedure;

//Alimentation de la DataGridView

SqlDataAdapter sd = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

sd.Fill(dt);

DGVLicences.DataSource = dt;

licences.SelectedIndex = 0;

}

Fonction du bouton recherche de la page des licences permettant un filtrage :

public void filterDataGridView(Object sender, EventArgs args)

{

resetDataGridView();

//filtrage avec l'année

int anneeFiltre = Convert.ToInt32(CB\_Annee.SelectedItem);

if (anneeFiltre != 0)

{

anneeFiltre = anneeFiltre - 2000;

foreach (DataGridViewRow row in DGVLicences.Rows)

{

if (Convert.ToInt32(row.Cells[0].Value) != anneeFiltre)

{

DGVLicences.Rows.RemoveAt(row.Index);

}

}

}

//filtrage avec l'intégrateur

if (CB\_integration.SelectedItem != null)

{

String integName = CB\_integration.SelectedItem.ToString();

foreach (Integrateur integ in lesIntegrateurs)

{

if (integ.nom == integName)

{

int integFiltre = integ.codeIntegrateur;

foreach (DataGridViewRow row in DGVLicences.Rows)

{

if (Convert.ToInt32(row.Cells[1].Value) != integFiltre)

{

DGVLicences.Rows.RemoveAt(row.Index);

}

}

}

}

}

//filtrage avec le client

if (CB\_client.SelectedItem != null)

{

String clientName = CB\_client.SelectedItem.ToString();

foreach (Client cli in lesClients)

{

if (cli.nom == clientName)

{

int cliFiltre = cli.codeClient;

foreach (DataGridViewRow row in DGVLicences.Rows)

{

if (Convert.ToInt32(row.Cells[2].Value) != cliFiltre)

{

DGVLicences.Rows.RemoveAt(row.Index);

}

}

}

}

}

//filtrage avec la fonction

if (CB\_fonction.SelectedItem != null)

{

String fonctionName = CB\_fonction.SelectedItem.ToString();

foreach (Fonction fonc in lesFonctions)

{

if (fonc.nom == fonctionName)

{

int foncFlitre = fonc.codeFonction;

foreach (DataGridViewRow row in DGVLicences.Rows)

{

if (Convert.ToInt32(row.Cells[4].Value) != foncFlitre)

{

DGVLicences.Rows.RemoveAt(row.Index);

}

}

}

}

}

}

La fonction du bouton remettre à 0 de la page des licences :

public void resetDataGridViewAndFilters(Object sender, EventArgs args)

{

con.Open();

//Recharger le tableau des licences

SqlCommand cmd = new SqlCommand("Base\_Licence.dbo.ICONE\_ps\_get\_licences", con);

cmd.CommandType = CommandType.StoredProcedure;

//Alimentation de la DataGridView

SqlDataAdapter sd = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

sd.Fill(dt);

DGVLicences.DataSource = dt;

CB\_Annee.SelectedItem = null;

CB\_client.SelectedItem = null;

CB\_fonction.SelectedItem = null;

CB\_integration.SelectedItem = null;

con.Close();

}

Fonction pour remettre le DataGridView à 0 :

public void resetDataGridView()

{

con.Open();

//Recharger le tableau des licences

SqlCommand cmd = new SqlCommand("Base\_Licence.dbo.ICONE\_ps\_get\_licences", con);

cmd.CommandType = CommandType.StoredProcedure;

//Alimentation de la DataGridView

SqlDataAdapter sd = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

sd.Fill(dt);

DGVLicences.DataSource = dt;

con.Close();

}

Fonction de modification d’une licence sur le bouton modifier (non terminé) :

public void updateLicence(Object sender, EventArgs args)

{

con.Open();

SqlCommand cmd = new SqlCommand("Base\_Licence.dbo.ICONE\_sp\_update\_licence\_id", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@id", SqlDbType.Int);

cmd.Parameters.Add("@codeIntegrateur", SqlDbType.Int);

cmd.Parameters.Add("@codeClient", SqlDbType.Int);

cmd.Parameters.Add("@codeFonction", SqlDbType.Int);

cmd.Parameters.Add("@nbEquipements", SqlDbType.Int);

cmd.Parameters.Add("@nbVariables", SqlDbType.Int);

cmd.Parameters.Add("@dateExpiration", SqlDbType.Int);

//parametre ID

int ID = Convert.ToInt32(tbModifID.Text);

cmd.Parameters["@id"].Value = ID;

MessageBox.Show("ID", ID.ToString());

//parametre codeIntegrateur

String integName = cbModifIntegrateur.SelectedItem.ToString();

foreach (Integrateur integ in lesIntegrateurs)

{

if (integ.nom == integName)

{

String integID = integ.codeIntegrateur.ToString();

while (integID.Length < 6)

{

integID = integID.Insert(0, "0");

}

MessageBox.Show("integID", integID);

cmd.Parameters["@codeIntegrateur"].Value = integID;

}

}

//parametre codeClient

String cliName = cbModifClient.SelectedItem.ToString();

foreach (Client cli in lesClients)

{

if (cli.nom == cliName)

{

String cliID = cli.codeClient.ToString();

while (cliID.Length < 6)

{

cliID = cliID.Insert(0, "0");

}

MessageBox.Show("cliID", cliID);

cmd.Parameters["@codeClient"].Value = cliID;

}

}

//parametre codeFonction

String foncName = cbModifFonction.SelectedItem.ToString();

foreach (Fonction fonc in lesFonctions)

{

if (fonc.nom == foncName)

{

String foncID = fonc.codeFonction.ToString();

while (foncID.Length < 6)

{

foncID = foncID.Insert(0, "0");

}

MessageBox.Show("foncID", foncID);

cmd.Parameters["@codeFonction"].Value = foncID;

}

}

//parametre nbEquipements

int nbEquipements = Convert.ToInt32(tbModifNbrEquipements.Text);

cmd.Parameters["@nbEquipements"].Value = nbEquipements;

MessageBox.Show("nbEquipements", nbEquipements.ToString());

//parametre nbVariables

int nbVariables = Convert.ToInt32(tbModifNbrVariables.Text);

cmd.Parameters["@nbVariables"].Value = nbVariables;

MessageBox.Show("nbVariables", nbVariables.ToString());

String dateExpiration = tbModifDateExpiration.Text;

dateExpiration = dateExpiration.Remove(2, 1);

dateExpiration = dateExpiration.Remove(4, 1);

int dateExpirationInt = Convert.ToInt32(dateExpiration);

cmd.Parameters["@dateExpiration"].Value = dateExpirationInt;

MessageBox.Show("dateExpiration", dateExpirationInt.ToString());

cmd.ExecuteNonQuery();

licences.SelectedIndex = 0;

resetDataGridView();

con.Close();

}

Après en être arrivé au fonctionnement de la modification pour le fonctionnement de l’application, a cause d’un changement de programme, le projet a de l’être mis de côté et de ce fait non terminer.

En perspective, une documentation du projet avait été commencer sur un document Word pour but de présenter le coté technique et explicatif de l’application.  
Celui-ci a du aussi être arrêter en cours à cause du changement de projet

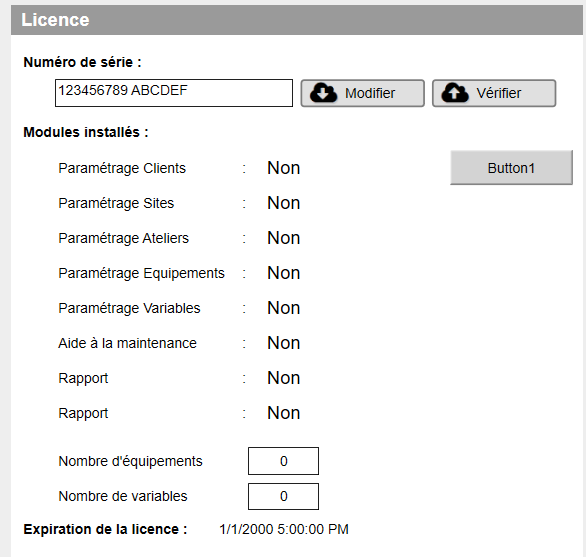
Ce projet m’a quand même servi pour le développement C#, un langage que je n’avais pas tant travailler que ça ainsi que l’utilisation d’une base SQL EXPRESS.

Il m’a donc été affecter par la suite un autre projet dans la même optique.

## Encrypteur de licences.

En suivant le contexte et les ressources du projet précédent, il m’a été demander de créer une autre application dans le même style permettant d’entrer les données de la licence, créer une chaine de caractères avec ces données, la crypter grâce a une clé générer par l’application, pour ensuite stocker la clé et la chaine crypté dans un fichier .txt .

### Ressource supplémentaire.

Une des ressources fournit en plus est un interface graphique représentant la finalité de l’application (l’espace où les informations de la licence seront retranscrits)

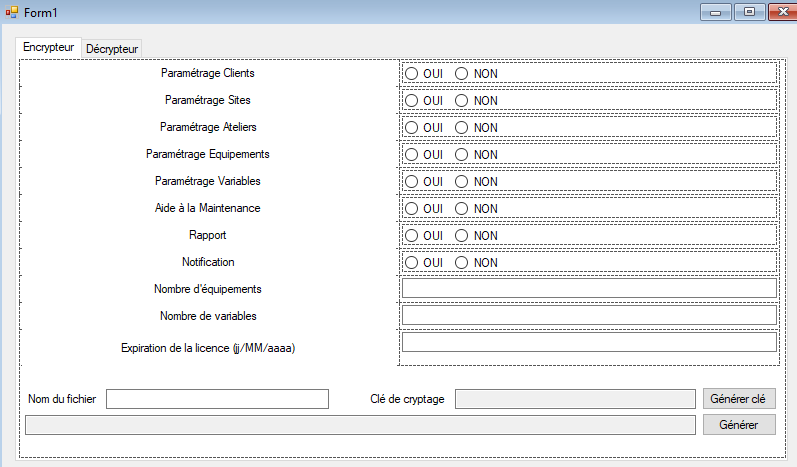
A partir de cette ressource, j’ai créé une application avec un interface permettant de correspondre aux besoins.

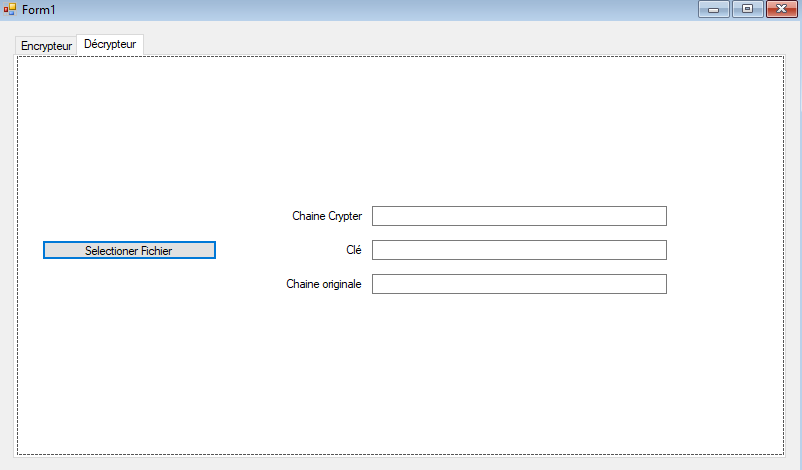
### Mise œuvre du projet.

Comme le projet précèdent, une application sous Visual Studio 2019 en C# WinForm m’a été demander.

Cette fois, n’ayant pas de base de données et des instructions clair, le projet à demander une organisation moins importante ce qui m’a permis de consacrer plus de temps à la finalisation de l’application.

Voici l’interface graphique de l’application :





La fonction suivante permet de générer la chaine de valeurs de la licence :

public void generateLicenceString()

{

String licenceStr = "";

//Parametrage clients ?

if (rbParamClientsOui.Checked == true)

{

licenceStr = licenceStr + "true;";

} else if (rbParamClientsNon.Checked == true)

{

licenceStr = licenceStr + "false;";

} else

{

licenceStr = licenceStr + "null;";

}

//Parametrage sites ?

if (rbParamSitesOui.Checked == true)

{

licenceStr = licenceStr + "true;";

}

else if (rbParamSitesNon.Checked == true)

{

licenceStr = licenceStr + "false;";

}

else

{

licenceStr = licenceStr + "null;";

}

//Parametrage ateliers ?

if (rbParamAteliersOui.Checked == true)

{

licenceStr = licenceStr + "true;";

}

else if (rbParamAteliersNon.Checked == true)

{

licenceStr = licenceStr + "false;";

}

else

{

licenceStr = licenceStr + "null;";

}

//Parametrage equipements ?

if (rbParamEquipementsOui.Checked == true)

{

licenceStr = licenceStr + "true;";

}

else if (rbParamEquipementsNon.Checked == true)

{

licenceStr = licenceStr + "false;";

}

else

{

licenceStr = licenceStr + "null;";

}

//Parametrage variables ?

if (rbParamVariablesOui.Checked == true)

{

licenceStr = licenceStr + "true;";

}

else if (rbParamVariablesNon.Checked == true)

{

licenceStr = licenceStr + "false;";

}

else

{

licenceStr = licenceStr + "null;";

}

//Aide à la maintenance ?

if (rbAideMaintenanceOui.Checked == true)

{

licenceStr = licenceStr + "true;";

}

else if (rbAideMaintenanceNon.Checked == true)

{

licenceStr = licenceStr + "false;";

}

else

{

licenceStr = licenceStr + "null;";

}

//Rapport ?

if (rbRapportOui.Checked == true)

{

licenceStr = licenceStr + "true;";

}

else if (rbRapportNon.Checked == true)

{

licenceStr = licenceStr + "false;";

}

else

{

licenceStr = licenceStr + "null;";

}

//Notification ?

if (rbNotifOui.Checked == true)

{

licenceStr = licenceStr + "true;";

}

else if (rbNotifNon.Checked == true)

{

licenceStr = licenceStr + "false;";

}

else

{

licenceStr = licenceStr + "null;";

}

//Nombres d'équipements ?

if (tbNbEquipements.Text == "")

{

licenceStr = licenceStr + "null;";

}

else

{

licenceStr = licenceStr + tbNbEquipements.Text + ";";

}

//Nombres de variables ?

if (tbNbVariables.Text == "")

{

licenceStr = licenceStr + "null;";

}

else

{

licenceStr = licenceStr + tbNbVariables.Text + ";";

}

String dateString = tbExpiration.Text;

String format = "dd/MM/yyyy";

CultureInfo provider = new CultureInfo("fr-FR");

try

{

DateTime result = DateTime.ParseExact(dateString, format, CultureInfo.InvariantCulture);

licenceStr = licenceStr + result.ToString().Remove(10, 9);

}

catch (FormatException)

{

licenceStr = licenceStr + "null";

}

tbLicenceStr.Text = licenceStr;

}

Fonction au chargement du formulaire :

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

{

generateLicenceString();

}

Fonction qui est dans chaque comboBox et textBox au moment d’un changement de valeur :

public void onChangeGenerateLicenceString(Object sender, EventArgs args)

{

generateLicenceString();

}

Cette fonction permet la génération d’une clé de cryptage selon un algorithme :

public void generateEncryptionKey (Object sender, EventArgs args)

{

using (Aes aesAlgorithm = Aes.Create())

{

aesAlgorithm.KeySize = 128;

aesAlgorithm.GenerateKey();

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

tbEncryptionKey.Text = keyBase64;

}

}

Cette fonction permet de vérifier que la clé a bien été générer avant de procéder au cryptage de la chaine :

public void checkGenerateConditions(Object sender, EventArgs agrs)

{

if (tbFileName.Text != "" && tbEncryptionKey.Text != "")

{

btnGenerateFile.Enabled = true;

} else

{

btnGenerateFile.Enabled = false;

}

}

Ensuite nous allons faire la classe Crypteur.cs permettant le cryptage et le décryptage de la de la chaine avec la clé de cryptage :

class Crypteur

{

public string EncryptString(string key, string plainText)

{

//definition de la taille de la clé

byte[] iv = new byte[16];

byte[] array;

using (Aes aes = Aes.Create())

{

//Passage de la clé et de sa taille au systeme AES

aes.Key = Encoding.UTF8.GetBytes(key);

aes.IV = iv;

//Création de l'encrypteur

ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV);

//Lécture et écriture de la chaine crypté

using (MemoryStream memoryStream = new MemoryStream())

{

using (CryptoStream cryptoStream = new CryptoStream((Stream)memoryStream, encryptor, CryptoStreamMode.Write))

{

using (StreamWriter streamWriter = new StreamWriter((Stream)cryptoStream))

{

streamWriter.Write(plainText);

}

array = memoryStream.ToArray();

}

}

}

return Convert.ToBase64String(array);

}

public string DecryptString(string key, string cipherText)

{

//definition de la taille de la clé

byte[] iv = new byte[16];

byte[] buffer = Convert.FromBase64String(cipherText);

using (Aes aes = Aes.Create())

{

//Passage de la clé et de sa taille au systeme AES

aes.Key = Encoding.UTF8.GetBytes(key);

aes.IV = iv;

//Création du décrypteur

ICryptoTransform decryptor = aes.CreateDecryptor(aes.Key, aes.IV);

//Lécture et écriture de la chaine décrypté

using (MemoryStream memoryStream = new MemoryStream(buffer))

{

using (CryptoStream cryptoStream = new CryptoStream((Stream)memoryStream, decryptor, CryptoStreamMode.Read))

{

using (StreamReader streamReader = new StreamReader((Stream)cryptoStream))

{

return streamReader.ReadToEnd();

}

}

}

}

}

}

Grace a la bibliothèque System.IO, nous pouvons créer des fichiers .txt, grâce à ceci nous allons laisser l’utilisateur choisir un chemin puis créer un fichier texte en .icone contenant la chaine crypter et la clé, voici la fonction :

public void generateEncryptedString(Object sender, EventArgs args)

{

Crypteur crypteur = new Crypteur();

String str = tbLicenceStr.Text;

String key = tbEncryptionKey.Text;

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

String decryptedString = crypteur.DecryptString(key, encryptedString);

pathToKeyFileDialog.ShowDialog();

String path = pathToKeyFileDialog.SelectedPath;

String name = tbFileName.Text;

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

if (File.Exists(fileName))

{

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

return;

}

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

{

sw.WriteLine(encryptedString);

sw.WriteLine(key);

}

rbParamClientsOui.Checked = false;

rbParamClientsNon.Checked = false;

rbParamSitesOui.Checked = false;

rbParamSitesNon.Checked = false;

rbParamAteliersOui.Checked = false;

rbParamAteliersNon.Checked = false;

rbParamEquipementsOui.Checked = false;

rbParamEquipementsNon.Checked = false;

rbParamVariablesOui.Checked = false;

rbParamVariablesNon.Checked = false;

rbAideMaintenanceOui.Checked = false;

rbAideMaintenanceNon.Checked = false;

rbRapportOui.Checked = false;

rbRapportNon.Checked = false;

rbNotifOui.Checked = false;

rbNotifNon.Checked = false;

tbNbEquipements.Text = "";

tbNbVariables.Text = "";

tbExpiration.Text = "";

tbEncryptionKey.Text = "";

tbFileName.Text = "";

onChangeGenerateLicenceString(null, null);

MessageBox.Show("Fichier creé avec succes");

}

Et pour finir, dans la page de décryptage, nous avons une fonction qui est celle-ci :

public void decryptFile(Object sender, EventArgs args)

{

openFileDialog.ShowDialog();

String file = openFileDialog.FileName;

if (file != null)

{

string[] lines = File.ReadAllLines(file);

Crypteur crypteur = new Crypteur();

String encryptedString = lines[0];

String key = lines[1];

String originalString = crypteur.DecryptString(key, encryptedString);

tbDecryptEncryptedString.Text = encryptedString;

tbDecryptKey.Text = key;

tbOriginalString.Text = originalString;

} else

{

MessageBox.Show("Erreur de chargement du ficher");

}

}

L’application est maintenant fonctionnelle, suite à cette tâche terminée, la suite logique des choses à été de réussir a implémenter le décryptage dans le projet concret.

## Implémentation décryptage sur Wonderware/AVEVA.

Pour cette tâche, il m’a été demander d’implémenter la fonction de décryptage du projet précèdent dans l’application concrète.

Cette application s’appel I-CONNECT et est développé sous WonderWare (Wonderware était une marque de logiciels industriels désormais détenue par Aveva et rebaptisée sous le nom d'AVEVA. Wonderware faisait partie d'Invensys plc, et Invensys plc a été acquis en janvier 2014 par Schneider Electric.).

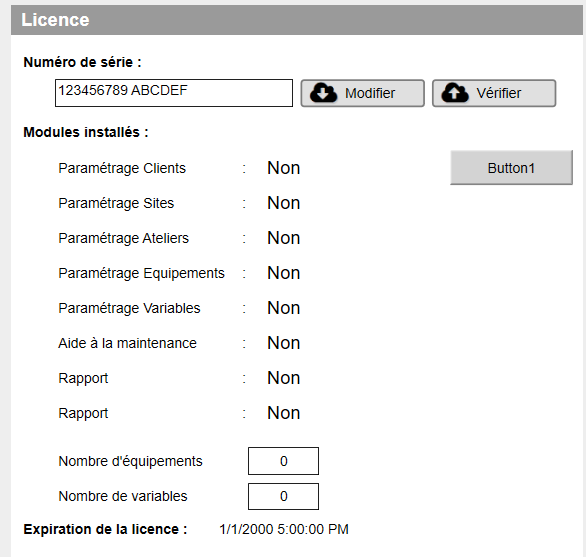
L’application utilise un langage de programmation très particulier, Le langage de script utilisé dans Wonderware Intouch s'appelle QuickScript qui a été modifier par Intouch ce qui nous donne un langage tout nouveau et une tache avec un niveau de challenge qui s’annonce intéressant.

### Ressources, contexte et technologies.

Les ressources pour ce projet on étaient les suivantes :

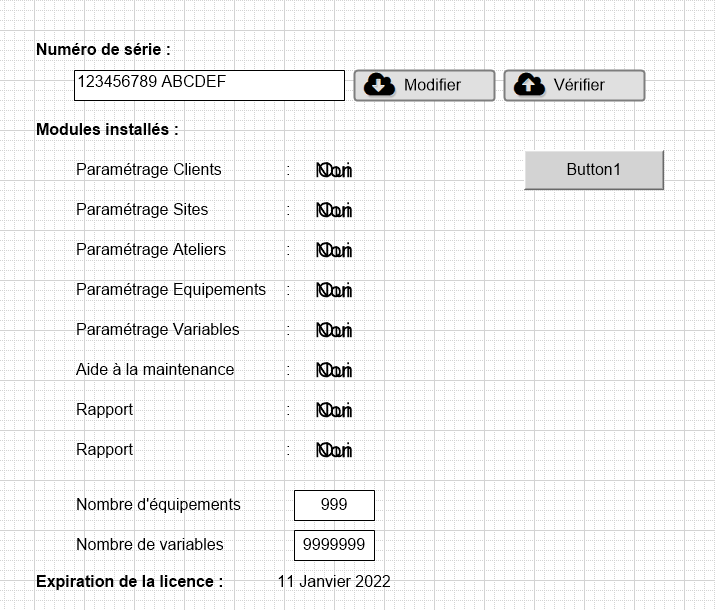
* L’application sur Wonderware
* Une interface de test
* Toute la logique des applications précédentes

Voici à quoi ressemble l’interface de l’application sur lequel j’ai travaillé :



Je devais à partir de mon fichier .icone retrouver les valeurs de la licence pour les rentrer dans l’applications permettant a l’utilisateur d’activer et vérifier sa licence.

Voici l’espace de test que j’avais pour pouvoir apprendre et m’entrainer sur cette nouvelle technologie :



Ma mission est simple, implémenter le décryptage dans le Button1 de cette page et mettre à jour les données de la page, mais ceci s’est annoncé plus compliquer que prévu.

### Développement logiciel et découverte.

Etant un logiciel assez couteux, j’y ai remarqué un manque crucial de documentation sur le web.  
Très peu d’exemple et d’explication m’on été présenter pour obtenir une base sur la logique de cette nouvelle technologie.

J’ai donc dû me fier au peu de manuel que j’ai trouvé sur le web (ces manuels faisant plusieurs centaines de page d’information, la recherche fu un vrai challenge de repérage d’informations).

De plus, l’environnement de développement ne contient ni débuggeur, ni point d’arrêt, sans compter les messages d’erreurs imprécis et assez aléatoire (des fois des messages d’erreurs de plusieurs dizaines de lignes, ou alors des messages d’erreurs faisant moins de 5 mots).

Malgré tout ces challenges, il a été intéressant de découvrir, apprendre et développé sous ce logiciel.

Voici le code affecter au Button1 :

'Dans le contexte de test avant l implémentation finale, nous utiliserons un chemin a choisir pour selectionné la licence

dim fileBrowser as System.Windows.Forms.OpenFileDialog;

fileBrowser = new System.Windows.Forms.OpenFileDialog();

fileBrowser.InitialDirectory = "c:\";

fileBrowser.Filter = "txt files (\*txt)|\*.txt|All files(\*.\*)|\*.\*";

fileBrowser.FilterIndex = 1;

fileBrowser.RestoreDirectory = true;

if fileBrowser.ShowDialog() == System.Windows.Forms.DialogResult.OK then

'definition de deux variables pour notre chaine crypter et notre clé symetrique de cryptage

dim encryptedString as System.String;;

dim key as string;

'lécture et affichage de chaque ligne du fichier et stockage des valeurs dans leurs variables respectables

dim lines = System.IO.File.ReadAllLines(fileBrowser.FileName);

encryptedString = lines[1];

key = lines[2];

LogMessage(encryptedString);

LogMessage(key);

'definition du buffer et de la taille en bytes

dim iv[16] as System.Byte;

dim buffer = System.Convert.FromBase64String(encryptedString);

dim Encoding as System.Text.Encoding;

'récuperation des bytes de la clé en UTF8

dim byteKey = System.Text.Encoding.UTF8.GetBytes(key);

'creation de notre AES

dim aes = System.Security.Cryptography.Aes.Create();

'définition de la clé et de la taille en bytes

aes.Key = byteKey;

aes.IV = iv;

'creation de notre décrypteur et assignation de la clé et de la taille

dim decryptor as System.Security.Cryptography.ICryptoTransform;

decryptor = aes.CreateDecryptor(aes.Key, aes.IV);

'dans un MemoryStream, nous passons le buffer (notre chaine crypté)

dim memoryStream = new System.IO.MemoryStream(buffer);

'avec notre MemoryStream, nous lui assignons le decrypteur et le cryptoStream fait le calcul de maniere read

dim cryptoStream = new System.Security.Cryptography.CryptoStream(memoryStream, decryptor, System.Security.Cryptography.CryptoStreamMode.Read);

'definition d un reader de Stream permettant de lire une objet de type stream

dim streamReader = new System.IO.StreamReader(cryptoStream);

'definition de deux variables pour la suite

dim originalValues as System.string;

dim string\_tab[11] as System.String;

'récupération de la chaine de pase avec notre streamReader et son affichage pour but de tests

originalValues = streamReader.ReadToEnd();

LogMessage(originalValues);

'séparation de la chaine à chaque chaine en parametre passée (dans ce cas ;) en tablau

string\_tab=originalValues.Split(";".ToCharArray());

'définition de toute nos valeurs correspondantes aux valeurs de la licence

dim parametrageClients as System.Boolean;

dim parametrageSites as System.Boolean;

dim parametrageAteliers as System.Boolean;

dim parametrageEquipements as System.Boolean;

dim parametrageVariables as System.Boolean;

dim aideMaintenance as System.Boolean;

dim rapport as System.Boolean;

dim notification as System.Boolean;

dim nombreEquipements as System.Int16;

dim nombreVariables as System.Int16;

dim dateExpiration as System.DateTime;

'association de nos valeurs avec leurs f=variables respectables

parametrageClients = System.Convert.ToBoolean(string\_tab[1]);

parametrageSites = System.Convert.ToBoolean(string\_tab[2]);

parametrageAteliers = System.Convert.ToBoolean(string\_tab[3]);

parametrageEquipements = System.Convert.ToBoolean(string\_tab[4]);

parametrageVariables = System.Convert.ToBoolean(string\_tab[5]);

aideMaintenance = System.Convert.ToBoolean(string\_tab[6]);

rapport = System.Convert.ToBoolean(string\_tab[7]);

notification = System.Convert.ToBoolean(string\_tab[8]);

nombreEquipements = System.Convert.ToInt16(string\_tab[9]);

nombreVariables = System.Convert.ToInt16(string\_tab[10]);

dateExpiration = System.Convert.ToDateTime(string\_tab[11]);

dateExpiration = dateExpiration.AddHours(23);

dateExpiration = dateExpiration.AddMinutes(59);

dateExpiration = dateExpiration.AddSeconds(59);

'Passage des données jusqu au nameSpace contentan les données de la licence

MyViewApp.Projet\_NS\_Licences.parametrageClients = parametrageClients;

MyViewApp.Projet\_NS\_Licences.parametrageSites = parametrageSites;

MyViewApp.Projet\_NS\_Licences.parametrageAteliers = parametrageAteliers;

MyViewApp.Projet\_NS\_Licences.parametrageEquipements = parametrageEquipements;

MyViewApp.Projet\_NS\_Licences.parametrageVariables = parametrageVariables;

MyViewApp.Projet\_NS\_Licences.aideMaintenance = aideMaintenance;

MyViewApp.Projet\_NS\_Licences.rapport = rapport;

MyViewApp.Projet\_NS\_Licences.notification = notification;

MyViewApp.Projet\_NS\_Licences.nbEquipements = nombreEquipements;

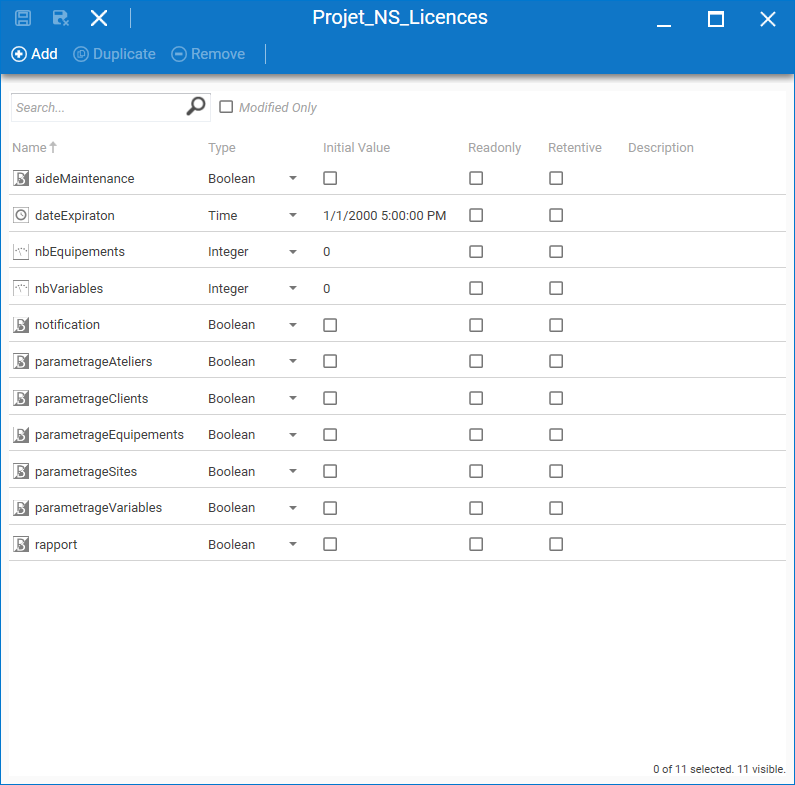
MyViewApp.Projet\_NS\_Licences.nbVariables = nombreVariables;

MyViewApp.Projet\_NS\_Licences.dateExpiraton = dateExpiration;

endif;

Et pour finir cette tâche, il manquait plus qu’à créer un ‘’NameSpace’’ (un espace de variables) me permettant de stocker et sauvegarder les données récupérer.

Voici le NameSpace utiliser :



Une fois ceci implémenter, mes taches de la semaine 1 se sont terminer.

# Outils utilisés

|  |  |
| --- | --- |
| Nom de l’outil | Fonction de l’outil |
| Visual Studio 2019 | Espace de développement avec multiple langage de programmations |
| SQL Server Management Studio 18 | Application logiciel de base de données |
| AVEVA Intouch HMI | AVEVA InTouch HMI est un logiciel de visualisation IHM qui permet aux clients de réaliser des projets avec excellence opérationnelle. |
| 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. |

# Conclusion.

Pour conclure cette première semaine, je l’ai trouvé vraiment intéressante malgré certains imprévus tel qu’un changement de projet en cours mais cela n’a pas enlever le coté éducatif de mes taches, j’ai pu approfondir mes compétences en C#, un langage que je maitriser légèrement, j’ai pu découvrir de nouvelle technologie comme SQL Server Management Studio pour les bases de données.  
Ensuite pour ce qui est de AVEVA/Wonderware, la tache a été un vrai challenge de A à Z, des syntaxes inconnues, une documentation casi inexistantes et un langage de programmation tourné a la sauce de InTouch.  
Tout ces éléments m’on permis de premièrement apprendre une nouvel technologie et un langage mais surtout d’apprendre à ma logique à s’adapter à un changement drastique dans les réflexions, les syntaxes, les mots clés etc.  
Réussir à s’adapter a quelque chose sans avoir d’information.  
C’est pour ca que j’ai trouver cette tache vraiment intéressante du fait que cela m’a demander une réflexion particulière pour la finalisation de la tâche.  
(Exemple : ‘’dim test as string’’ définit une variable chaine de caractère mais qui contient moins de méthode que ‘’dim test as System.String’’).

En finalité je trouve avoir été bien productif cette semaine et avoir réussi à me surpasser dans mes taches tout en apprenant dans une espace et un cadre d’entreprise professionnel.

# Niko-Niko.

|  |  |  |
| --- | --- | --- |
| JOUR | RESSENTI | TACHE |
| Lundi | Big Smile Emoticon With Thumbs Up Stock Illustration - Download Image Now -  Emoticon, Smiling, Thumbs Up - iStock | Gestionnaire de licence |
| Mardi | 15,256 Confused Emoji Stock Photos, Pictures & Royalty-Free Images - iStock | Gestionnaire de licence/Crypteur de licence (Changement de projet) |
| Mercredi | Hollow - Discord Emoji | AVEVA InTouch HMI (Découverte) |
| Jeudi | Amazed Emoticon Stock Illustration - Download Image Now - Shock, Emoticon,  Bad News - iStock | AVEVA InTouch HMI (Compréhension et apprentissage) |
| Vendredi |  | AVEVA InTouch HMI (Finalisation) |