



Ingénierie & systèmes automatisés

COMPTE RENDU STAGE

Année 2023-2024

SEMAINE 4

23/01/2023 – 27
/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.	Base de données I-CONNECT.....	4
2.2.	Gestionnaires_Licences.....	8
2.2.1.	Base de données.....	10
2.2.2.	Procédures stockées.....	12
2.2.3.	Interface graphique Licences.....	14
2.2.4.	Interface graphique Intégrateurs/Clients/Fonctions.....	16
2.2.5.	Fonctionnalités et utilisation.....	18
3.	Outils utilisés.....	22
4.	Conclusion.....	23
5.	Niko-Niko.....	24
6.	Annexe codes.....	25
6.1.	ICONE_sp_codes.....	25
6.1.1.	CRUD dbo.Variables :.....	25
6.1.2.	CRUD dbo.Historisable.....	42
6.1.3.	CRUD dbo.Alarme.....	48
6.1.4.	Create Equation.....	54
6.1.5.	GET_sp.....	58
6.2.	Gestionnaires_licences_sp.....	61
6.2.1.	CRUD Client/Fonctions/Integrateurs.....	61
6.2.2.	CRUD Licences.....	63
6.2.3.	Fonctionnalités.....	67

1. Présentation de l'entreprise.



ICÔNE 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. Base de données I-CONNECT

Pour commencer la semaine, j'ai continué les modifications demandées par mon tuteur à la base de données de son application.

Voici les principaux changements :

Création de la table Alarme (représente un niveau d'alarme pour une variable) :

Nom	Type	Contraintes
Variable_ID	Int	FK REFERENCES Table Variables Column ID, Non null
Type_alarme	Float	Value = 1 2 3 4, Non null
Seuil	Float	Non null
Priority	Int	Non null
Message	Nvarchar(max)	Nullable

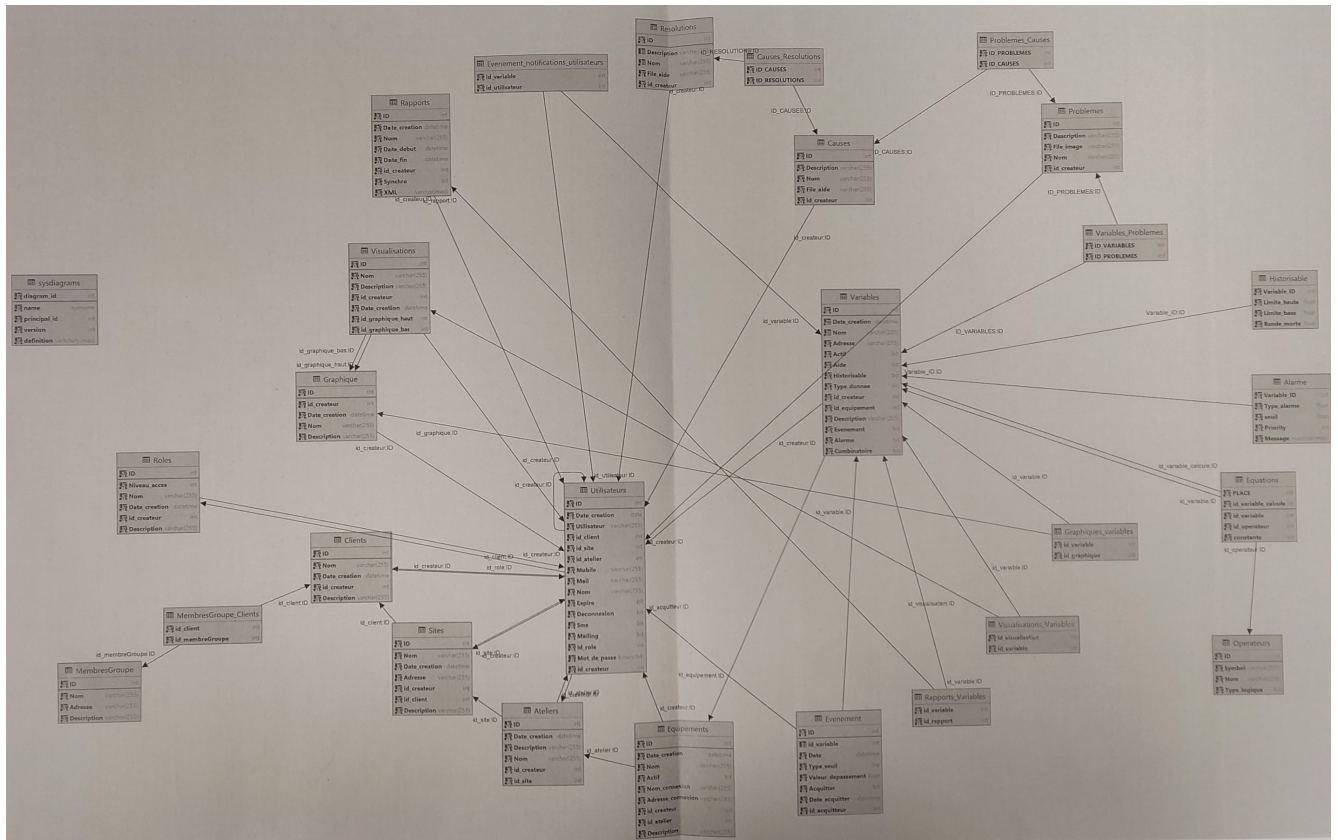
Création de la table Historisable (Représente les limites d'une variable) :

Nom	Type	Contraintes
Variable_ID	Int	FK REFERENCES table Variables Column ID, Non null
Limite_haute	Float	Non null
Limite_basse	Float	Non null
Bande_morte	float	Non null

Modifications de la table Variable : (Représente les variables, ajout de nouvelles options)

Nom	Type	Contraintes
ID	Int	PK, IDENTITY, Non null
Id_createur	Int	FK REFERENCES Table Utilisateurs Column Id_utilisateur, Non null
Id_equipement	Int	FK REFERENCES Table Equipements Column Id_equipement, Non null
Date_creation	Datetime	Non null
Nom	Varchar(255)	Non null
Adresse	Varchar(255)	Nullable
Actif	bit	Non null
Aide	Bit	Non null
Historisable	Bit	Non null
Type_donnee	Int	Non null
Description	Varchar(255)	Nullable
Evenement	Bit	Non null
Alarme	Bit	Non null
Combinatoire	bit	Non null

Suite aux rajouts et aux modifications, voici ce que donne le schéma de la base de données :



Après ces modifications faites, j'ai dû faire plusieurs procédure stocké ([Voir 6.1. ICONECT sp_codes](#) pour les codes des procédures) :

- CRUD sur dbo.Variables
- CRUD sur dbo.Historisable
- CRUD sur dbo.Alarme
- Create Equation
- Get_notification_email (renvoie l'email d'un utilisateur)
- Get_notification_sms (renvoie le numéro mobile d'un utilisateur)
- Get_variable_options (renvoie les options d'une variable)

Pour finir ce projet, j'ai, exporté la une sauvegarde de la base en un fichier .bak et l'ai stocké dans le fichier de stockage des sauvegardes de la base, ainsi que

mis à jour le fichier de documentation sur l'application au niveau des procédures stockées en suivant la norme.

Voici un exemple (ceci est fait pour chaque procédure) :

ICONE sp create variable

Paramètres :

Nom	Type	Contrainte d'entrée
Nom	Varchar (255)	Non null Pas vide
Id_utilisateur	Int	Non null
Id_equipement	Int	Non null
Description	Varchar (255)	
Adresse	Varchar (255)	
Actif	Bit	
Aide	Bit	
Mail	Bit	
Type_donnee	Int	
Evenement	Bit	Non null
Historisable	Bit	
Limite_haute	Float	
Limite_basse	Float	
Bande_morte	Float	
Evenement	Bit	
Type_seuil	Int	
Valeur_depassement	Int	
Alarme	Bit	
Alarme_type1	Bit	
Alarme_type2	Bit	
Alarme_type3	Bit	
Alarme_type4	Bit	
Seuil1	Float	
Seuil2	Float	
Seuil3	Float	
Seuil4	Float	
Priority1	Int	
Priority2	Int	
Priority3	Int	
Priority4	Int	

Message1	Nvarchar(max)	
Message2	Nvarchar(max)	
Message3	Nvarchar(max)	
Message4	Nvarchar(max)	
Combinatoire	Bit	
Liste_Libelles	Varchar(max)	
Liste_IDs	Varchar(max)	

Si l'utilisateur pointé par l'id fourni a le/les champs id_client et/ou id_site et/ou id_atelier qui n'est/sont pas null alors l'id_atelier de l'équipement fournie doit correspondre à l'utilisateur (appartenir au client/site/atelier de l'utilisateur).

Cette procédure stockée retourne un entier :

Valeurs	Signification
-1	Erreur non identifiée
0	Réussite de la procédure
1	L'utilisateur n'a pas le niveau d'accès nécessaire
2	L'id_client d'utilisateur n'est pas null et n'est pas égale à l'id_client du site auquel l'atelier de l'équipement appartient.
3	L'id_site de l'utilisateur n'est pas null et n'est pas égale à l'id_site de l'atelier de l'équipement fourni
4	L'id_atelier d'utilisateur n'est pas null et n'est pas égale à l'id_atelier de l'équipement fournie
10	Le nom est null ou vide
100	L'utilisateur n'existe pas

2.2. Gestionnaires Licences.

Dés mercredi après-midi, j'ai appris que le premier projet que j'avais commencer (le simulateur de licences) allait être présenté le lundi 30

Mon tuteur pensait que le projet était terminé, seulement, il avait été mis en pause/de coté a cause de changements de dernière minutes.

Ceci m'a donc laissé 2 jours et demi pour :

- Apporté les nouvelles modifications à la base de données
- Apporté les nouvelles contraintes graphiques et fonctionnel à l'application
- Refaire le document utilisateur sur l'application
- Terminé l'application/documentation pour Vendredi 27

Suite a toute ces modifications et ces changements, j'ai pris les décisions de :

1. Recommencer le projet à 0 pour pouvoir avoir les idées claires, et reprendre des bases solides pour accommoder les changements
2. Me mettre des deadlines pour mes taches afin de finir a temps le projet même si cela me demander de rester après mes horaires dans l'entreprise
3. Mettre en place un ordre de priorité dans les taches (Exemple 1. Fonctionnalité, 2. Adaptation, 3. Design etc...)

2.2.1. Base de données.

Voici la base de données reprise depuis 0 :

Dbp.Clients :

Nom	Type	Contraintes
Code_client	Int	PK, IDENTITY, Non null
Nom	Varchar(255)	Non null
Date_update	Datetime	Non null
Description	Varchar(255)	Nullables

Dbp.Fonctions :

Nom	Type	Contraintes
Code_fonction	Int	PK, IDENTITY, Non null
Nom	Varchar(255)	Non null
Date_update	Datetime	Non null
Description	Varchar(255)	Nullables

Dbp.Integrateurs

Nom	Type	Contraintes
Code_integrateur	Int	PK, IDENTITY, Non null
Nom	Varchar(255)	Non null
Date_update	Datetime	Non null
Description	Varchar(255)	Nullables

Dbp.Licences :

Nom	Type	Contraintes
ID_Licence	Int	PK, IDENTITY, Non null
Date_creation	Datetime	Non null
Date_expiration	Datetime	Non null
Nb_equipements	Int	Non null
Nb_variables	Int	Non null
Checksum	Varchar(2)	Non null

Code_integrateur	Int	FK REFERENCES Table Integrateurs Column Code_integrateur, Non null
Code_client	Int	FK REFERENCES Tables Clients Column Code_client, Non null
Code_fonction1	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction2	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction3	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction4	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction5	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction6	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction7	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction8	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction9	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction10	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction11	int	FK REFERENCES Table

		Fonctions Column Code_fonction, Non null
Code_fonction12	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction13	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction14	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction15	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction16	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction17	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction18	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction19	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null
Code_fonction20	int	FK REFERENCES Table Fonctions Column Code_fonction, Non null

2.2.2. Procédures stockées.

Après ceci j'ai fait des procédures stockées qui seront utilisées par l'application :

- CRUD sur dbo.Clients
- CRUD sur dbo.Fonctions

- CRUD sur dbo.Integrateurs
- CRUD sur dbo.Licences

Ces procédures sont simples et basiques et font les fonctions classiques d'un CRUD (Create, Read, Update, Delete)

[Voir 6.2. Gestionnaires_licences_sp](#) pour les codes

2.2.3. Interface graphique Licences.

Concernant le coté design, j'ai été plutôt libre sur la disposition et les choix, j'ai donc fait une interface dans un but pratique et simple d'utilisation :

Sélection de la table

Affichage de la table

Formulaire d'ajout

Fonction d'ajouts rapide
(explication par la suite)

Quand on double clique sur une licence dans le DataGridView :

Fonction d'ajouts rapide
(explication par la suite)

Gestionnaire_Licences_APP

Licences Integrateurs Clients Fonctions

Integrateurs :				Clients :				Fonctions :			
Code_integrateur	Nom	Date_update	Description	Code_client	Nom	Date_update	Description	Code_fonction	Nom	Date_update	Description
1	ICONE	26/01/2023	Entreprise	1	testClient	25/01/2023 14:00	Test un cli	1	TestFonction	25/01/2023 14:02	Test une f
3	TagProduct	26/01/2023	Entreprise	4	MonsieurToto	26/01/2023	Il veut adr	3	Admin	26/01/2023	Donne ac
4	FournisseurLicen...	26/01/2023	Fournisseu	5	SNCF	26/01/2023	Fournisseu	4	Accès variables	26/01/2023	Ceci donn
5	Integrateur	26/01/2023	Ceci est u	6	BanqueDeFrance	26/01/2023	La banque	5	Accès statistiques	26/01/2023	Ceci donn
6	IN-TECH	26/01/2023	Fime d'ent	7	ICONE	26/01/2023	Entreprise	6	Accès sauvegard...	26/01/2023	Ceci donn

< > < > < >

Données de la licence :

1	2	3	4	5
1. TestFonction	0. NULL	0. NULL	0. NULL	0. NULL
6	7	8	9	10
0. NULL	0. NULL	0. NULL	0. NULL	0. NULL
11	12	13	14	15
0. NULL	0. NULL	0. NULL	0. NULL	0. NULL
16	17	18	19	20
0. NULL	0. NULL	0. NULL	0. NULL	0. NULL
Checksum	Nbr d'équipements	Nbr de variables	Code integrateur	Code client
CS	10	10	1	1
Date d'expiration				
mercredi 25 janvier 2023				

ID

Date de création : 25/01/2023

Appliquer les modifications

Supprimer la licence

Retour

Modifications de la licence :

1	2	3	4	5
1	0	0	0	0
6	7	8	9	10
0	0	0	0	0
11	12	13	14	15
0	0	0	0	0
16	17	18	19	20
0	0	0	0	0
Checksum	Nbr d'équipements	Nbr de variables	Code integrateur	Code client
CS	10	10	1	1
Date d'expiration				
Mettre à 0 mercredi 25 janvier 2023 Remettre à l'origine				

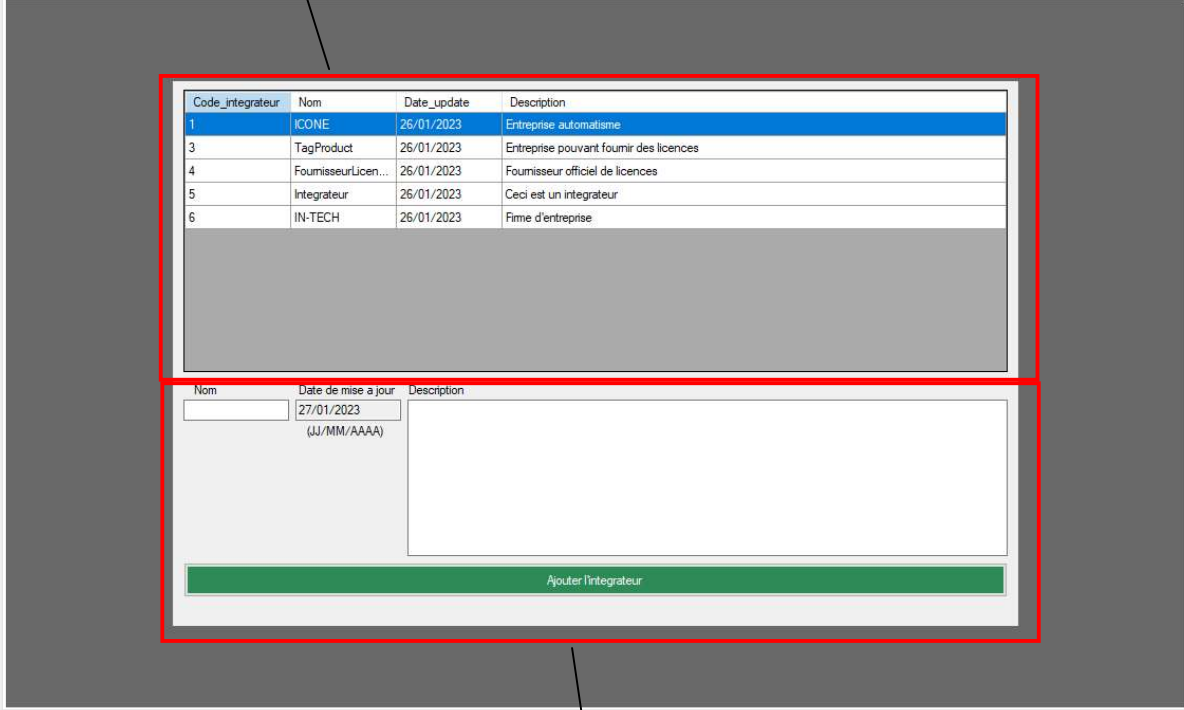
Données de la licence

Modifications

2.2.4. Interface graphique Intégrateurs/Clients/Fonctions.

Pour ces interfaces, ils sont tous les trois similaires.

Table intégrateurs



The screenshot shows a web application window titled 'Gestionnaire_Licences_APP'. It has four tabs: 'Licences', 'Integrateurs', 'Clients', and 'Fonctions'. The 'Integrateurs' tab is active. A red box highlights two main components: a table of existing integrators and a form for adding a new one.

Code_integrateur	Nom	Date_update	Description
1	ICONE	26/01/2023	Entreprise automatisée
3	TagProduct	26/01/2023	Entreprise pouvant fournir des licences
4	FournisseurLicen...	26/01/2023	Fournisseur officiel de licences
5	Integrateur	26/01/2023	Ceci est un integrateur
6	IN-TECH	26/01/2023	Fime d'entreprise

Below the table is a form for adding a new integrator. It contains three input fields: 'Nom' (empty), 'Date de mise à jour' (containing '27/01/2023' with a hint '(JJ/MM/AAAA)'), and 'Description' (empty). A green button labeled 'Ajouter l'integrateur' is at the bottom of the form.

Formulaire d'ajout

Si une donnée est double cliqué :

Donnée de
(Client/Intégrateur/Fonction

Modification de ses
données

The screenshot displays the 'Gestionnaire_Licences_APP' window with the 'Integrateurs' tab selected. The interface is divided into two main sections, each outlined with a red box. The left section, titled 'Données de l'intégrateur', contains form fields for 'Nom' (filled with 'ICONE'), 'Date de mise à jour' (filled with '26/01/2023'), and 'Description' (filled with 'Entreprise automatisme'). The right section, titled 'Modifications', contains identical form fields but with 'Date de mise à jour' filled with '27/01/2023'. Between these sections are three buttons: 'Appliquer modifications' (green), 'Supprimer l'intégrateur' (red), and 'Retour' (blue). At the bottom, there is a 'Code intégrateur' field with a small 'i' icon.

2.2.5. Fonctionnalités et utilisation.

(Chaque code de fonction peut se retrouver dans l'annexe des codes [Voir 6.2.3. Fonctionnalités](#))

Pour chaque interface, nous avons un cas d'utilisation similaire :

- Bouton vert = Ajout/Modification
- Bouton rouge = Suppression
- Bouton bleu = Retour en arrière
- Bouton marron = Fonctionnalité supplémentaire

Quand double cliqué, renvoie vers la page de modifications d'une licence

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 14:05	10	10	CS	1	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	11	6	6	6	5	4
15	26/01/2023	26/01/2023	632	410	11	3	7	3	5	6

Date de création: 27/01/2023 Date d'expiration: vendredi 27 janvier 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 integrateurs :

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	Firme c

Sélectionner un clients :

Code_client	Nom	Date_update	Descr
1	TestClient	25/01/2023 14:00	Test u
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

Sélectionner une fonctions :

Code_fonction	Nom	Date_update	Descr
1	TestFonction	25/01/2023 14:02	Test u
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

Ajoute une licence avec les données du formulaire

Met toutes les données à 0

Met les fonctions à 0

Quand un des trois DataGridView est cliqué, il remplit le formulaire de données de manière dynamique, par exemple, il suffit de double cliquer sur la ligne contenant le client ICONE pour mettre son ID dans la case, ainsi de suite pour l'intégrateur et les fonctions.

Gestionnaire_Licences_APP

Licences Integrateurs Clients Fonctions

Integrateurs :

Code_integrateur	Nom	Date_update	Descripti
1	ICONE	26/01/2023	Entreprise
3	TagProduct	26/01/2023	Entreprise
4	FournisseurLicen...	26/01/2023	Fournisseu
5	Integrateur	26/01/2023	Ceci est u
6	IN-TECH	26/01/2023	Firme d'ent

Clients :

Code_client	Nom	Date_update	Descripti
1	testClient	25/01/2023 14:00	Test un cli
4	MonsieurToto	26/01/2023	Il veut adr
5	SNCF	26/01/2023	Fournisseu
6	BanqueDeFrance	26/01/2023	La banque
7	ICONE	26/01/2023	Entreprise

Fonctions :

Code_fonction	Nom	Date_update	Descripti
1	TestFonction	25/01/2023 14:02	Test une f
3	Admin	26/01/2023	Donne aci
4	Accès variables	26/01/2023	Ceci donn
5	Accès statistiques	26/01/2023	Ceci donn
6	Accès sauvegard...	26/01/2023	Ceci donn

Données de la licence :

1	2	3	4	5
1. TestFonction	0. NULL	0. NULL	0. NULL	0. NULL
6	7	8	9	10
0. NULL	0. NULL	0. NULL	0. NULL	0. NULL
11	12	13	14	15
0. NULL	0. NULL	0. NULL	0. NULL	0. NULL
16	17	18	19	20
0. NULL	0. NULL	0. NULL	0. NULL	0. NULL
Checksum	Nbr d'équipements	Nbr de variables	Code integrateur	Code client
CS	10	10	1	1
Date d'expiration				
mercredi 25 janvier 2023				

Modifications de la licence :

1	2	3	4	5
1	0	0	0	0
6	7	8	9	10
0	0	0	0	0
11	12	13	14	15
0	0	0	0	0
16	17	18	19	20
0	0	0	0	0
Checksum	Nbr d'équipements	Nbr de variables	Code integrateur	Code client
CS	10	10	1	1
Date d'expiration				
mercredi 25 janvier 2023				

Appliquer les modifications

Supprimer la licence

Retour

Mettre à 0

Remettre à l'origine

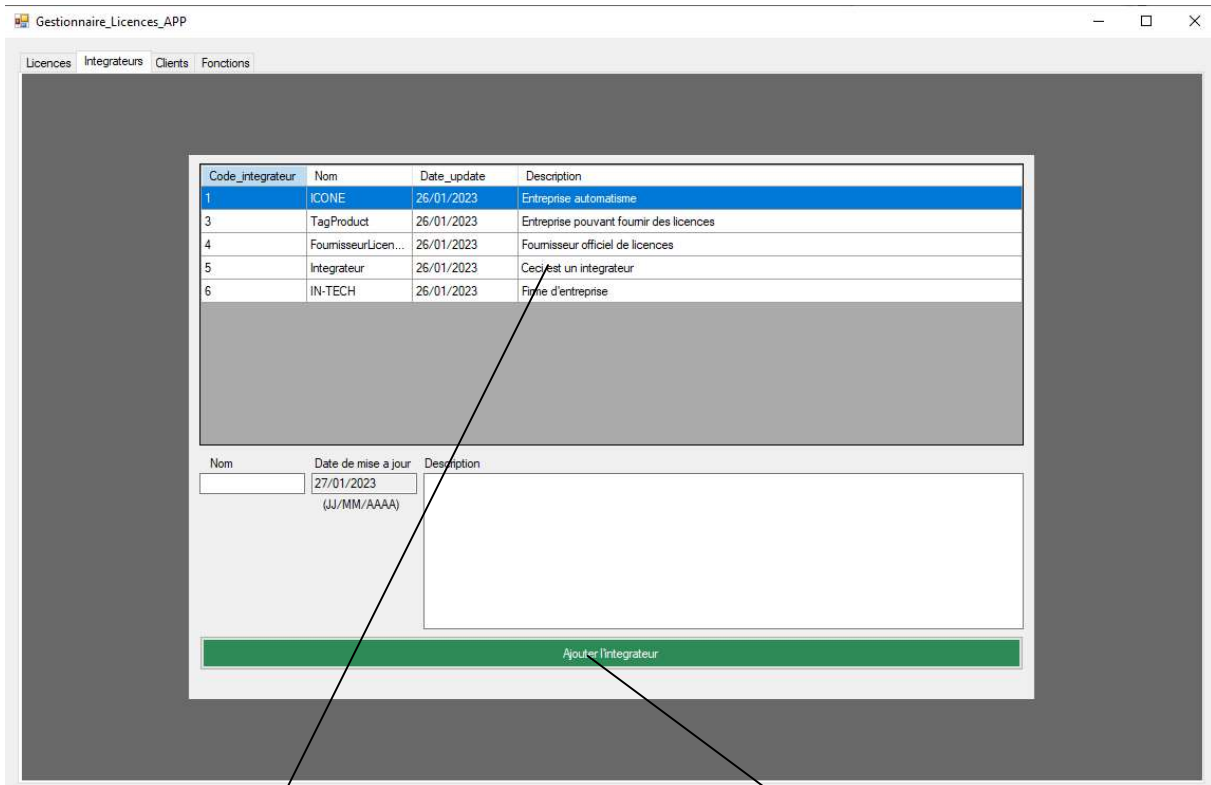
Appliquer les modifications apportées à la licence

Suppression de la licence

Renvoie à la page précédente

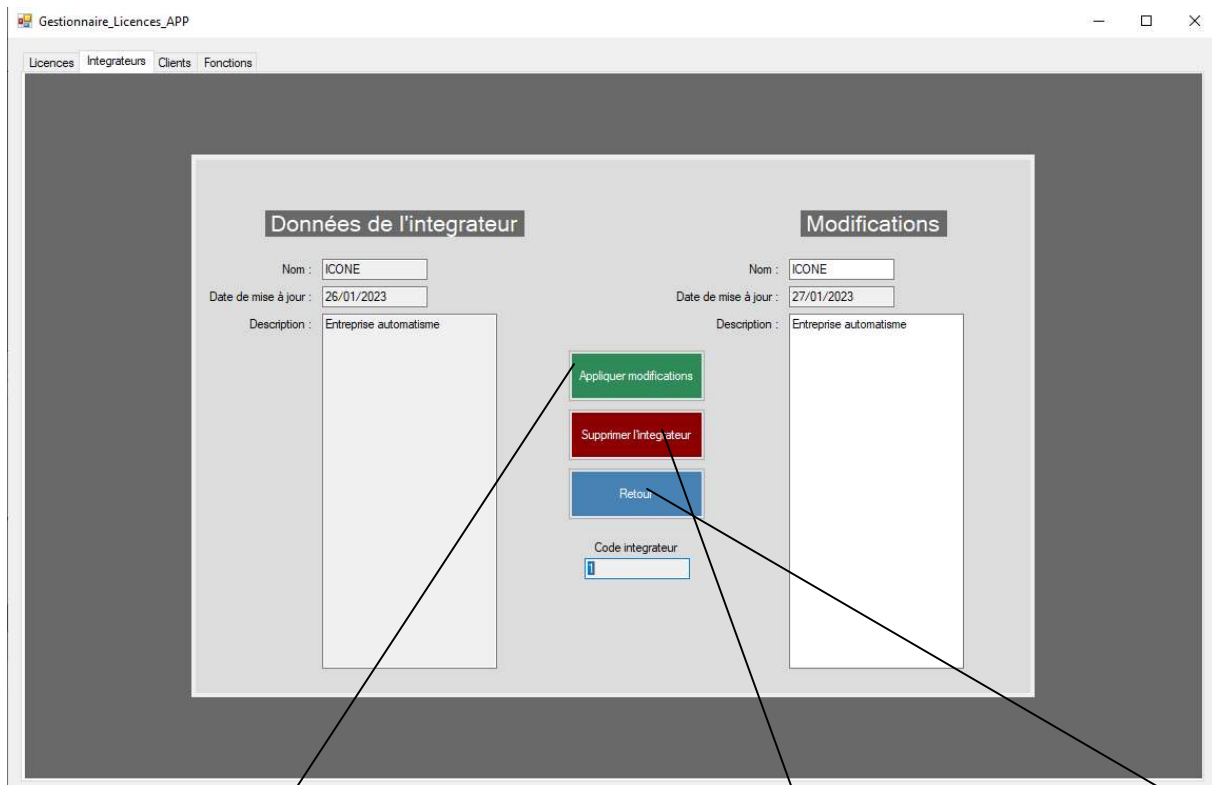
Met tout à 0

Remet les données à celle d'origine de la licence



Quand double cliqué, Renvoie vers la page de modification

Ajoute un intégrateur



Applique les modifications sur le Client/Intégrateur/Fonction

Supprime le Client/Intégrateur/Fonction

Renvoie vers la page précédente

Ce projet a donc été réalisé en 2 jours et demi et m'as demandé :

- 1500+ lignes de code
- Une répartition de mon temps très important
- Une rapidité d'exécution
- Une rapidité d'adaptation aux demandes
- Une certaine rigueur dans mon organisation et mon travail

Je passe donc Lundi 29 devant mon chef d'entreprise ainsi que des chargés d'affaires pour faire une présentation et une démonstration de mon application.

3. Outils utilisés.

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

4. Conclusion.






J'ai commencé cette semaine par une fin de tâche en SQL et ai été très rapidement confronter au plus gros challenge jusqu'à maintenant, qui a été la confection de l'application Gestionnaires_Licences depuis le début avec des nouveau changements en 2 jours et demi.

J'ai été mis à l'épreuve et ai pu tester mon habilité à réagir vite, organiser mon travail dans un temps répartie et mettre des deadlines dans mes tâches.

Malgré le stresse et la pression que j'avais, ceci à été à mon avis, l'expérience la plus intéressante jusque là dans le sens ou j'ai découvert les possibles problèmes qui peuvent se passer dans une entreprise et qui engendres des demandes rapides.

Je pense avoir été bien productif dans le temps répartie et j'espère, que l'application sera a la hauteur des demandes des spectateurs de la présentation

5. Niko-Niko.

JOUR	RESSENTI	TACHE
Lundi		Continuation modifications base I-CONNECT (Jour tranquille)
Mardi		Finition base I-CONNECT + Documentation de la base (Jour tranquille 2)
Mercredi		Mise en place de la reprise du projet Gestionnaires_Licences (Jour du changement de projet)
Jeudi		Création interfaces + Fonctions affichages et navigations (Jour de pression et de tryhard)
Vendredi		Création fonctionnalités CRUD + peaufinage et modification design (Jour des finitions et de la descente du stress)

6. Annexe codes.

6.1. ICONE sp codes

6.1.1. CRUD dbo.Variables :

Create :

```
USE [BASE_ICONECT]
GO
/***** Object: StoredProcedure [dbo].[ICONE_sp_create_variable]    Script Date:
27/01/2023 11:09:40 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          Escolano Allan
-- Description:      Crée une variable
-- Donnée d'entrée :
--      @Nom varchar(255),
--      @Adresse varchar(255),
--      @Actif bit,
--      @Aide bit,
--      @Type_donnee int,
--      @Description varchar(255),
--      @id_utilisateur int,
--      @id_equipement int,
--
--      --Option Historisable
--      @Historisable bit,
--      @limite_haute float,
--      @limite_basse float,
--      @Bande_morte float,
--
--      --Option Evenement
--      @Evenement bit,
--      @Type_seuil int,
--      @valeur_depassement int,
--
--      --Option Alarme
--      @Alarme bit,
--      @Alarme_type1 bit,
--      @Alarme_type2 bit,
--      @Alarme_type3 bit,
--      @Alarme_type4 bit,
--      @seuil1 float,
--      @seuil2 float,
--      @seuil3 float,
--      @seuil4 float,
--      @Priority1 int,
--      @Priority2 int,
--      @Priority3 int,
--      @Priority4 int,
```

```
--      @Message1 Nvarchar(max),
--      @Message2 Nvarchar(max),
--      @Message3 Nvarchar(max),
--      @Message4 Nvarchar(max),
--
--      --Option Combinatoire
--      @Combinatoire bit,
--      @liste_libelles varchar(max),
--      @liste_IDs varchar(max)
--
-- Code erreur :
--      1 => l'utilisateur n'a pas le niveau d'accès nécessaire
--      2 => l'utilisateur appartient à un client qui n'est pas celui pointé par
id_client du site auquel l'atelier de l'equipement appartient
--      3 => l'utilisateur appartient à un site qui n'est pas celui pointé par id_site
de l'atelier de l'equipement fourni.
--      4 => l'utilisateur appartient à un atelier qui n'est pas celui de
l'equipement.
--      100 => l'utilisateur fournie n'existe pas
--      101 => l'id equipement n'est pas présent en base

-- =====
ALTER PROCEDURE [dbo].[ICONE_sp_create_variable]
    @Nom varchar(255),
    @Adresse varchar(255),
    @Actif bit,
    @Aide bit,
    @Type_donnee int,
    @Description varchar(255),
    @id_utilisateur int,
    @id_equipement int,

    --Option Historisable
    @Historisable bit,
    @Limite_haute float,
    @Limite_basse float,
    @Bande_morte float,

    --Option Evenement
    @Evenement bit,
    @Type_seuil int,
    @valeur_depassement int,

    --Option Alarme
    @Alarme bit,
    @Alarme_type1 bit,
    @Alarme_type2 bit,
    @Alarme_type3 bit,
    @Alarme_type4 bit,
    @seuil1 float,
    @seuil2 float,
    @seuil3 float,
    @seuil4 float,
    @Priority1 int,
    @Priority2 int,
    @Priority3 int,
    @Priority4 int,
    @Message1 Nvarchar(max),
    @Message2 Nvarchar(max),
    @Message3 Nvarchar(max),
```

```

@Message4 Nvarchar(max),

--Option Combinatoire
@Combinatoire bit,
@liste_libelles varchar(max),
@liste_IDS varchar(max)
AS
BEGIN
    -- on vérifie que @Nom n'est pas null
    if(@Nom is null or LEN(@Nom) = 0)
    begin
        return 10;
    end

    -- on vérifie que @id_utilisateur est présent en BDD
    if (SELECT COUNT(*) FROM dbo.Utilisateurs WHERE ID = @id_utilisateur) = 0
    begin
        return 100;
    end

    -- on vérifie que @id_equipement est présent en BDD
    if @id_equipement is not null AND
        (SELECT COUNT(*) FROM dbo.Equipements WHERE ID = @id_equipement) = 0
    begin
        return 101;
    end

    -- on vérifie que l'utilisateur a un niveau d'accès nécessaire
    if (select Niveau_acces from roles where id = (
        select id_role from utilisateurs where id = @id_utilisateur
    )) < 6000
    begin
        return 1;
    end

    -- si l'id_atelier d'utilisateur existe,
    -- alors on vérifie que cet id est égale à l'id_atelier de l'équipement
    declare @id_atelier_equipement int;
    declare @id_atelier_utilisateur int;
    select @id_atelier_utilisateur = id_atelier from Utilisateurs where id =
@id_utilisateur;
    select @id_atelier_equipement = id_atelier from Equipements where id =
@id_equipement;
    if @id_atelier_utilisateur is not null AND @id_atelier_equipement !=
@id_atelier_utilisateur
    begin
        return 4;
    end

    -- si l'id_site d'utilisateur existe,
    -- alors on vérifie que cet id est égale à l'id_site de l'atelier de l'équipement
    declare @id_site_equipement int;
    declare @id_site_utilisateur int;
    select @id_site_utilisateur = id_site from Utilisateurs where id =
@id_utilisateur;
    select @id_site_equipement = id_site from Ateliers where id =
@id_atelier_equipement;
    if @id_site_utilisateur is not null AND @id_site_equipement !=
@id_site_utilisateur
    begin

```

```

        return 3;
    end

    -- si l'id_client d'utilisateur existe
    -- alors on vérifie que cet id est égale à l'id_client du site auquel l'atelier de
    l'équipement appartient
    declare @id_client_equipement int;
    declare @id_client_utilisateur int;
    select @id_client_utilisateur = id_client from Utilisateurs where id =
    @id_utilisateur;
    select @id_client_equipement = id_client from Sites where id =
    @id_site_equipement;
    if @id_client_utilisateur is not null AND @id_client_equipement !=
    @id_client_utilisateur
    begin
        return 2;
    end

    declare @Variable_ID int;
    select @Variable_ID = IDENT_CURRENT('Variables')-IDENT_INCR('Variables') + 2
    select @Variable_ID as variableID

    -- on démarre un try catch, normalement les erreurs ont déjà été attrapé, si une
    erreur a été omise, on retourne -1
    begin try
        INSERT INTO
            dbo.Variables(
                Date_creation, Nom, Adresse, Actif, Aide, Historisable,
                Type_donnee, id_createur, id_equipement, Description,
                Evenement, Alarme, Combinatoire)
        VALUES (
            GETDATE(), @Nom, @Adresse, @Actif, @Aide, @Historisable,
            @Type_donnee, @id_utilisateur, @id_equipement, @Description,
            @Evenement, @Alarme, @Combinatoire
        );

        --si l'option combinatoire
        if (@Combinatoire = 1)
        begin
            execute ICONE_sp_create_equation @Variable_ID, @id_utilisateur,
            @liste_libelles, @liste_IDs
        end

        --si l'option Alarme
        if (@Alarme = 1)
        begin
            if (@Alarme_type1 = 1)
            begin
                execute ICONE_sp_create_alarme @Variable_ID, 1, @seuil1,
            @Priority1, @Message1
            end

            if (@Alarme_type2 = 1)
            begin
                execute ICONE_sp_create_alarme @Variable_ID, 2, @seuil2,
            @Priority2, @Message2
            end

            if (@Alarme_type3 = 1)

```

```
begin
    execute ICONE_sp_create_alarmme @Variable_ID, 3, @seuil3,
@Priority3, @Message3
end

    if (@Alarme_type4 = 1)
begin
        execute ICONE_sp_create_alarmme @Variable_ID, 4, @seuil4,
@Priority4, @Message4
    end
end

--si l'option Evenement
if (@Evenement = 1)
begin
    execute ICONE_sp_create_evenement @Variable_ID, @Type_seuil,
@valeur_depassement
end

--si l'option Historisable
if (@Historisable = 1)
begin
    execute ICONE_sp_create_historisable @Variable_id, @Limite_haute,
@Limite_basse, @Bande_morte
end
    RETURN 0;
end try
begin catch
    return -1;
end catch
END
```


Read :

```
USE [BASE_ICONECT]
GO
/***** Object: StoredProcedure [dbo].[ICONE_sp_get_variables]    Script Date:
27/01/2023 11:10:29 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          ESCOLANO Allan
-- Description:      Renvoie la liste des variables
-- Donnée d'entrée :
-- @id_utilisateur int => id de l'utilisateur courant
-- Code erreur :
-- 1 => l'utilisateur n'a pas le droit d'accéder aux données
-- 100 => l'utilisateur fournie n'existe pas
-- Return :
-- ID int non null
-- Nom varchar(255) non null
-- Date création datetime non null
-- Description varchar (255)
-- Créateur varchar(255) non null
-- Client varchar(255) non null
-- Site varchar(255) non null
-- Atelier varchar (255) non null
-- Equipements varchar(255) non null
-- Actif bit non null
-- Aide bit non null
-- Historisable bit non null
-- Type de donnée int
-- Evenement bit
-- Limite_haute float
-- Limite_basse float
-- Limite_tres_haute float
-- Limite_tres_basse float
-- =====
ALTER PROCEDURE [dbo].[ICONE_sp_get_variables]
    @id_utilisateur int
AS
BEGIN
    -- Si l'utilisateur fournie n'existe pas dans la base de donnée
    if (Select count(*) from Utilisateurs where id = @id_utilisateur) = 0
    begin
        return 100;
    end

    -- Si l'utilisateur n'a pas le droit de voir les données
    if (
        Select Niveau_acces
        From Roles LEFT JOIN dbo.Utilisateurs U on Roles.ID = U.id_role
        WHERE U.ID = @id_utilisateur
    ) < 6000
    begin
        return 1;
    end
end
```

```
-- On s'occupe d'abord du cas où l'utilisateur fait partie du groupe
if (select id_client from Utilisateurs where id = @id_utilisateur) is null
begin
    Select v.ID, v.Nom, v.Description, v.Date_creation as 'Date création',
           u.Nom as 'Créateur', c.Nom as 'Client', s.Nom as 'Site', a.Nom as
'Atelier', e.Nom as 'Equipement',
           v.Actif, v.Aide, v.Historisable, v.Type_donnee as 'Type de donnée',
v.Evenement,
           v.Combinatoire, v.Alarme, e.id as 'id equipement',
           v.Adresse
    From Variables as v,
         Equipements as e,
         Ateliers as a,
         Utilisateurs as u,
         Clients as c,
         Sites as s
    Where v.id_equipement = e.id
          AND e.id_atelier = a.id
          AND a.id_createur = u.id
          AND c.id = s.id_client
          AND s.id = a.id_site;
end

-- on s'occupe ensuite du cas où l'utilisateur a un id_site null
-- donc il a le droit de voir tous ce qui est en rapport avec le client pointé par
id_client
else if (select id_site from Utilisateurs where id = @id_utilisateur) is null
begin
    Select v.ID, v.Nom, v.Description, v.Date_creation as 'Date création',
           u.Nom as 'Créateur', c.Nom as 'Client', s.Nom as 'Site', a.Nom as
'Atelier', e.Nom as 'Equipement',
           v.Actif, v.Aide, v.Historisable, v.Type_donnee as 'Type de donnée',
v.Evenement,
           v.Combinatoire, v.Alarme, e.id as 'id equipement',
           v.Adresse
    From Variables as v,
         Equipements as e,
         Ateliers as a,
         Utilisateurs as u,
         Clients as c,
         Sites as s
    Where v.id_equipement = e.id
          AND e.id_atelier = a.id
          AND a.id_createur = u.id
          AND c.id = s.id_client
          AND s.id = a.id_site
          AND c.id = u.id_client;
end

-- on s'occupe ensuite du cas où l'utilisateur a un id_atelier null
-- donc il a le droit de voir tous ce qui est en rapport avec le site pointé par
id_site
else if (select id_site from Utilisateurs where id = @id_utilisateur) is null
begin
    Select v.ID, v.Nom, v.Description, v.Date_creation as 'Date création',
           u.Nom as 'Créateur', c.Nom as 'Client', s.Nom as 'Site', a.Nom as
'Atelier', e.Nom as 'Equipement',
           v.Actif, v.Aide, v.Historisable, v.Type_donnee as 'Type de donnée',
v.Evenement,
           v.Combinatoire, v.Alarme, e.id as 'id equipement',
```

```
        v.Adresse
    From Variables as v,
        Equipements as e,
        Ateliers as a,
        Utilisateurs as u,
        Clients as c,
        Sites as s
    Where v.id_equipement = e.id
        AND e.id_atelier = a.id
        AND a.id_createur = u.id
        AND c.id = s.id_client
        AND s.id = a.id_site
        AND s.id = u.id_site;

end

-- On s'occupe finalement du cas où l'utilisateur a un id atelier
-- Dans ce cas il n'a le droit de voir que sont atelier
else
begin
    Select v.ID, v.Nom, v.Description, v.Date_creation as 'Date création',
        u.Nom as 'Créateur', c.Nom as 'Client', s.Nom as 'Site', a.Nom as
    'Atelier', e.Nom as 'Equipement',
        v.Actif, v.Aide, v.Historisable, v.Type_donnee as 'Type de donnée',
    v.Evenement,
        v.Combinatoire, v.Alarme, e.id as 'id equipement',
        v.Adresse
    From Variables as v,
        Equipements as e,
        Ateliers as a,
        Utilisateurs as u,
        Clients as c,
        Sites as s
    Where v.id_equipement = e.id
        AND e.id_atelier = a.id
        AND a.id_createur = u.id
        AND c.id = s.id_client
        AND s.id = a.id_site
        AND a.id = u.id_atelier;

end
END
```

Update :

```
USE [BASE_ICONECT]
GO
/***** Object: StoredProcedure [dbo].[ICONE_sp_update_variable]    Script Date:
27/01/2023 11:11:08 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          ESCOLANO Allan
-- Description:      Met a jour les données d'une variable et ses options respectives
-- Donnée d'entrée :
--      @ID int,
--      @Nom varchar(255),
--      @Adresse varchar(255),
--      @Actif bit,
--      @Aide bit,
--      @Type_donnee int,
--      @Description varchar(255),
--      @id_utilisateur int,
--      @id_equipement int,
--
--      --Option Historisable
--      @Historisable bit,
--      @Limite_haute float,
--      @Limite_basse float,
--      @Bande_morte float,
--
--      --Option Evenement
--      @Evenement bit,
--      @Type_seuil int,
--      @valeur_depassement int,
--
--      --Option Alarme
--      @Alarme bit,
--      @Alarme_type1 bit,
--      @Alarme_type2 bit,
--      @Alarme_type3 bit,
--      @Alarme_type4 bit,
--      @seuil1 float,
--      @seuil2 float,
--      @seuil3 float,
--      @seuil4 float,
--      @Priority1 int,
--      @Priority2 int,
--      @Priority3 int,
--      @Priority4 int,
--      @Message1 Nvarchar(max),
--      @Message2 Nvarchar(max),
--      @Message3 Nvarchar(max),
--      @Message4 Nvarchar(max),
--
--      --Option Combinatoire
--      @Combinatoire bit,
--      @liste_libelles varchar(max),
--      @liste_IDS varchar(max)
```

```
-- Code erreur :
--      1 => l'utilisateur n'a pas le niveau d'accès nécessaire
--      2 => l'utilisateur appartient à un client qui n'est pas celui pointé par
id_client du site auquel l'atelier de l'équipement appartient
--      3 => l'utilisateur appartient à un site qui n'est pas celui pointé par id_site
de l'atelier de l'équipement fourni.
--      4 => l'utilisateur appartient à un atelier qui n'est pas celui de
l'équipement.
--      5 => la variable n'existe pas
--      10 => @Nom est null ou vide
--      16 => @limite_basse est supérieur ou égale à @limite_haute
--      100 => l'utilisateur fournie n'existe pas
--      101 => l'id équipement n'est pas présent en base
-- =====
ALTER PROCEDURE [dbo].[ICONE_sp_update_variable]
    @ID int,
    @Nom varchar(255),
    @Adresse varchar(255),
    @Actif bit,
    @Aide bit,
    @Type_donnee int,
    @Description varchar(255),
    @id_utilisateur int,
    @id_equipement int,

    --Option Historisable
    @Historisable bit,
    @Limite_haute float,
    @Limite_basse float,
    @Bande_morte float,

    --Option Evenement
    @Evenement bit,
    @Type_seuil int,
    @valeur_depassement int,

    --Option Alarme
    @Alarme bit,
    @Alarme_type1 bit,
    @Alarme_type2 bit,
    @Alarme_type3 bit,
    @Alarme_type4 bit,
    @seuil1 float,
    @seuil2 float,
    @seuil3 float,
    @seuil4 float,
    @Priority1 int,
    @Priority2 int,
    @Priority3 int,
    @Priority4 int,
    @Message1 Nvarchar(max),
    @Message2 Nvarchar(max),
    @Message3 Nvarchar(max),
    @Message4 Nvarchar(max),

    --Option Combinatoire
    @Combinatoire bit,
    @liste_libelles varchar(max),
    @liste_IDS varchar(max)
```

```
BEGIN
-- on vérifie que @Nom n'est pas null
if(@Nom is null or LEN(@Nom) = 0)
begin
    return 10;
end

-- on vérifie que @id_utilisateur est présent en BDD
if (SELECT COUNT(*) FROM dbo.Utilisateurs WHERE ID = @id_utilisateur) = 0
begin
    return 100;
end

-- on vérifie que @id_equipement est présent en BDD
if @id_equipement is not null AND
(SELECT COUNT(*) FROM dbo.Equipements WHERE ID = @id_equipement) = 0
begin
    return 101;
end

-- on vérifie que l'utilisateur a un niveau d'accès nécessaire
if (select Niveau_acces from roles where id = (
    select id_role from utilisateurs where id = @id_utilisateur
)) < 6000
begin
    return 1;
end

-- si l'id_atelier d'utilisateur existe,
-- alors on vérifie que cet id est égale à l'id_atelier de l'équipement
declare @id_atelier_equipement int;
declare @id_atelier_utilisateur int;
select @id_atelier_utilisateur = id_atelier from Utilisateurs where id =
@id_utilisateur;
select @id_atelier_equipement = id_atelier from Equipements where id =
@id_equipement;
if @id_atelier_utilisateur is not null AND @id_atelier_equipement !=
@id_atelier_utilisateur
begin
    return 4;
end

-- si l'id_site d'utilisateur existe,
-- alors on vérifie que cet id est égale à l'id_site de l'atelier de l'équipement
declare @id_site_equipement int;
declare @id_site_utilisateur int;
select @id_site_utilisateur = id_site from Utilisateurs where id =
@id_utilisateur;
select @id_site_equipement = id_site from Ateliers where id =
@id_atelier_equipement;
if @id_site_utilisateur is not null AND @id_site_equipement !=
@id_site_utilisateur
begin
    return 3;
end

-- si l'id_client d'utilisateur existe
-- alors on vérifie que cet id est égale à l'id_client du site auquel l'atelier de
l'équipement appartient
declare @id_client_equipement int;
```

```

declare @id_client_utilisateur int;
select @id_client_utilisateur = id_client from Utilisateurs where id =
@id_utilisateur;
select @id_client_equipement = id_client from Sites where id =
@id_site_equipement;
if @id_client_utilisateur is not null AND @id_client_equipement !=
@id_client_utilisateur
begin
    return 2;
end

if (select COUNT(*) from Variables where ID = @ID) = 0
begin
    return 5;
end

declare @variable_historisable bit;
select @variable_historisable = Historisable from Variables where ID = @ID

declare @variable_alarmme bit;
select @variable_alarmme = Alarmme from Variables where ID = @ID

declare @variable_evenement bit;
select @variable_evenement = Evenement from Variables where ID = @ID

declare @variable_combinatoire bit;
select @variable_combinatoire = Combinatoire from Variables where ID = @ID

-- on démarre un try catch, normalement les erreurs ont déjà été attrapé, si une
erreur a été omise, on retourne -1
begin try
    UPDATE Variables
    SET Nom = @Nom, Adresse = @Adresse, Actif = @Actif, Aide = @Aide,
    Historisable = @Historisable, Type_donnee = @Type_donnee, id_equipement =
    @id_equipement, Description = @Description, Evenement = @Evenement, Alarmme = @Alarmme,
    Combinatoire = @Combinatoire
    WHERE ID = @ID

    --Si l'option Historisable a changer
    if (@Historisable != @variable_historisable)
    begin
        if (@Historisable != 1)
        begin
            EXECUTE ICONE_sp_delete_historisable @ID
        end

        if (@Historisable = 1)
        begin
            EXECUTE ICONE_sp_create_historisable @ID, @Limite_haute,
@Limite_basse, @Bande_morte
        end
    end

    --Si l'option Alarmme a changer
    if (@Alarmme != @variable_alarmme)
    begin
        if (@Alarmme != 1)
        begin

```



```

        EXECUTE ICONE_sp_delete_alarme @ID
    end

    if (@Alarme = 1)
    begin
        if (@Alarme_type1 = 1)
        begin
            execute ICONE_sp_create_alarme @ID, 1, @seuil1,
@Priority1, @Message1
        end

        if (@Alarme_type2 = 1)
        begin
            execute ICONE_sp_create_alarme @ID, 2, @seuil2,
@Priority2, @Message2
        end

        if (@Alarme_type3 = 1)
        begin
            execute ICONE_sp_create_alarme @ID, 3, @seuil3,
@Priority3, @Message3
        end

        if (@Alarme_type4 = 1)
        begin
            execute ICONE_sp_create_alarme @ID, 4, @seuil4,
@Priority4, @Message4
        end
    end
    end

    --Si l'option Combinatoire a changer
    if (@Combinatoire != @variable_combinatoire)
    begin
        if (@Combinatoire != 1)
        begin
            EXECUTE ICONE_sp_delete_equation @ID, @id_utilisateur
        end

        if (@Combinatoire = 1)
        begin
            EXECUTE ICONE_sp_create_equation @ID, @id_utilisateur,
@liste_libelles, @liste_IDS
        end
    end

    --Si l'option Evenement a changer
    /*if (@Evenement != @variable_evenement)
    begin
        if (@Evenement != 1)
        begin

        end

        if (@Evenement = 1)
        begin

        end
    end*/
end*/

```

```
                RETURN 0;  
            end try  
            begin catch  
                return -1;  
            end catch  
        END
```

Delete :

```
USE [BASE_ICONECT]
GO
/***** Object:  StoredProcedure [dbo].[ICONE_sp_delete_variable]    Script Date:
27/01/2023 11:11:41 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          ESCOLANO Allan
-- Create date: 2022-05-12
-- Description:      Supprime une variable
-- Donnée d'entrée :
--      @ID int non null,
--      @id_utilisateur int non null
-- Code erreur :
--      1 => l'utilisateur n'a pas le niveau d'accès nécessaire
--      2 => l'utilisateur n'a pas le niveau d'accès nécessaire pour supprimer un
équipement dont il n'est pas le créateur
--      3 => l'utilisateur essaye de supprimer une variable qui n'appartient pas à son
client
--      4 => l'utilisateur essaye de supprimer une variable qui n'appartient pas à son
site
--      5 => l'utilisateur essaye de supprimer une variable qui n'appartient pas à son
atelier
--      100 => l'utilisateur fournie n'existe pas
--      101 => la variable n'existe pas
-- =====
ALTER PROCEDURE [dbo].[ICONE_sp_delete_variable]
    @Id integer,
    @id_utilisateur int
AS
BEGIN
    -- si l'id utilisateur est null
    if (SELECT COUNT(*) FROM Utilisateurs WHERE Id = @id_utilisateur) = 0
    begin
        return 100;
    end

    -- on vérifie que @Id existe dans la BDD
    if (SELECT COUNT(*) FROM Variables WHERE Id = @ID) = 0
    begin
        return 101;
    end

    -- test sur le niveau d'accès
    declare @Niveau_acces int;

    select  @Niveau_acces = Niveau_acces from Roles where ID = (
        select id_role from Utilisateurs where id = @id_utilisateur
    );

    -- si l'utilisateur n'a pas les droits
    if @Niveau_acces < 6000
    begin
        return 1;
    end
end
```

```

-- si l'utilisateur ne possède pas le sites et n'a pas les droits
if @Niveau_acces < 7000 AND (
    select id_createur
    from Variables
    where id = @id
) != @id_utilisateur
begin
    return 2;
end

-- si l'utilisateur appartient à un client et qu'il essaye de supprimer une donnée
qui ne lui appartient pas
if (Select id_client from Utilisateurs where id = @id_utilisateur)
!=
(Select id_client from Sites where id = (select id_site from Ateliers where id
= (
    Select id_atelier from equipements where id = (
        select id_equipement from variables where id = @id
    )
)))
AND (select id_client from Utilisateurs where id = @id_utilisateur) is not
null
begin
    return 3;
end

-- si l'utilisateur appartient à un site et qu'il essaye de supprimer une donnée
qui ne lui appartient pas
if (Select id_site from Utilisateurs where id = @id_utilisateur)
!=
(Select id_site from Ateliers where id = (
    select id_atelier from equipements where id = (
        select id_equipement from variables where id = @id
    )
))
AND (select id_site from Utilisateurs where id = @id_utilisateur) is not null
begin
    return 4;
end

-- si l'utilisateur appartient à un client et qu'il essaye de supprimer une donnée
qui ne lui appartient pas
if (Select id_atelier from Utilisateurs where id = @id_utilisateur)
!=
(Select id_atelier from Equipements where id = (
    select id_equipement from variables where id = @id
))
AND (select id_atelier from Utilisateurs where id = @id_utilisateur) is not
null
begin
    return 5;
end

--on démarre un try catch, normalement les erreurs ont déjà été attrapé, si une
erreur a été omise, on retourne -1
begin try
    DELETE FROM Rapports_Variables WHERE id_variable = @id;
    DELETE FROM Visualisations_Variables WHERE id_variable = @id;
    DELETE FROM Evenement where id_variable = @id;

```

```
DELETE FROM Evenement_notifications_utilisateurs WHERE id_variable = @Id
DELETE FROM Variables_Problemes Where ID_VARIABLES = @id;
EXECUTE ICONE_sp_delete_equation @Id, @id_utilisateur
EXECUTE ICONE_sp_delete_alarme @Id
EXECUTE ICONE_sp_delete_historisable @Id
DELETE FROM Variables where id = @id;
RETURN 0;
end try
begin catch
return -1;
end catch
END
```

6.1.2. CRUD dbo.Historisable

Create :

```
USE [BASE_ICONECT]
GO
/***** Object:  StoredProcedure [dbo].[ICONE_sp_create_historisable]    Script Date:
27/01/2023 11:14:22 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      Allan ESCOLANO
-- Description:  Crée la ligne Historisable
-- Donnée d'entrée :
--      @ID int,
--      @Limite_haute float,
--      @Limite_base float,
--      @Bande_morte float,
-- Code erreur :
--      1 => Limite_haute ne peut pas etre NULL
--      2 => Limite_base ne peut pas etre NULL
--      3 => Bande_morte ne peut pas etre NULL
--      100 => Aucune variable ne contient l'ID renseigné
--      101 => La variable.historisable est a NULL
-- =====
ALTER PROCEDURE [dbo].[ICONE_sp_create_historisable]
    @ID int,
    @Limite_haute float,
    @Limite_base float,
    @Bande_morte float
AS
BEGIN
    -- on verifie si la variable existe
    if (SELECT COUNT(*) FROM dbo.Variables WHERE ID = @ID) = 0
    begin
        return 100;
    end

    -- on verifie que la variable.historisable est TRUE
    declare @variable_historisable bit;
    select @variable_historisable = historique FROM dbo.Variables WHERE ID = @ID
    if (@variable_historisable = 0 or @variable_historisable is null)
    begin
        return 101;
    end

    -- on verifie que Limite_haute n'est pas NULL
    if (@Limite_haute is null)
    begin
        return 1;
    end
end
```

```
-- on verifie que Limite_base n'est pas NULL
if (@Limite_base is null)
begin
    return 2;
end

-- on verifie que Bande_morte n'est pas NULL
if (@Bande_morte is null)
begin
    return 3;
end

begin try
    INSERT INTO dbo.Historisable (Variable_ID, Limite_haute, Limite_base,
Bande_morte)
    VALUES (@id, @Limite_haute, @Limite_base, @Bande_morte)
end try
begin catch
    return -1;
end catch

END
```

Read :

```
USE [BASE_ICONECT]
GO
/***** Object: StoredProcedure [dbo].[ICONE_sp_get_historisable]    Script Date:
27/01/2023 11:14:55 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          Allan ESCOLANO
-- Description:      Recupere la ligne historisable
-- Donnée d'entrée :
--      @ID int
-- Code erreur :
--      100 => Aucune variable ne contient l'ID renseigné
--      101 => La variable.historisable est a NULL
--      102 => La variable ne contient pas de ligne historisable
-- =====
ALTER PROCEDURE [dbo].[ICONE_sp_get_historisable]
    @ID int
AS
BEGIN

    -- on verifie si la variable existe
    if (SELECT COUNT(*) FROM dbo.Variables WHERE ID = @ID) = 0
    begin
        return 100;
    end

    -- on verifie que la variable.historisable est TRUE
    declare @variable_historisable bit;
    select @variable_historisable = historisable FROM dbo.Variables WHERE ID = @ID
    if (@variable_historisable = 0 or @variable_historisable is null)
    begin
        return 101;
    end

    -- on verifie si la il y a une ligne historisable pour la variable
    if (SELECT COUNT(*) FROM dbo.Historisable WHERE Variable_ID = @ID) = 0
    begin
        return 102;
    end

    begin try
        SELECT Variable_ID, Limite_haute, Limite_base, Bande_morte FROM
dbo.Historisable
        WHERE Variable_ID = @ID
    end try
    begin catch
        return -1;
    end catch

END
```


Update :

```
USE [BASE_ICONECT]
GO
/***** Object: StoredProcedure [dbo].[ICONE_sp_update_historisable]    Script Date:
27/01/2023 11:15:23 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          Allan ESCOLANO
-- Description:      Mettre a jour la ligne Historisable
-- Donnée d'entrée :
--      @ID int,
--      @Limite_haute float,
--      @Limite_base float,
--      @Bande_morte float,
-- Code erreur :
--      1 => Limite_haute ne peut pas etre NULL
--      2 => Limite_base ne peut pas etre NULL
--      3 => Bande_morte ne peut pas etre NULL
--      100 => Aucune variable ne contient l'ID renseigné
--      101 => La variable.historisable est a NULL
--      102 => La variable ne contient pas de ligne historisable
-- =====

ALTER PROCEDURE [dbo].[ICONE_sp_update_historisable]
    @ID int,
    @Limite_haute float,
    @Limite_base float,
    @Bande_morte float
AS
BEGIN

    -- on verifie si la variable existe
    if (SELECT COUNT(*) FROM dbo.Variables WHERE ID = @ID) = 0
    begin
        return 100;
    end

    -- on verifie que la variable.historisable est TRUE
    declare @variable_historisable bit;
    select @variable_historisable = historisable FROM dbo.Variables WHERE ID = @ID
    if (@variable_historisable = 0 or @variable_historisable is null)
    begin
        return 101;
    end

    -- on verifie si la variable existe
    if (SELECT COUNT(*) FROM dbo.Historisable WHERE Variable_ID = @ID) = 0
    begin
        return 102;
    end

    -- on verifie que Limite_haute n'est pas NULL
    if (@Limite_haute is null)
    begin
```

```
        return 1;
    end

    -- on verifie que Limite_base n'est pas NULL
    if (@Limite_base is null)
    begin
        return 2;
    end

    -- on verifie que Bande_morte n'est pas NULL
    if (@Bande_morte is null)
    begin
        return 3;
    end

    begin try
        UPDATE dbo.Historisable
        SET Limite_haute = @Limite_haute, Limite_base = @Limite_base, Bande_morte
= @Bande_morte
        WHERE Variable_ID = @ID
    end try
    begin catch
        return -1;
    end catch

END
```

Delete :

```
USE [BASE_ICONECT]
GO
/***** Object: StoredProcedure [dbo].[ICONE_sp_delete_historisable]    Script Date:
27/01/2023 11:15:53 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          ESCOLANO Allan
-- Description:      Supprime les données Historisable d'une variable
-- Donnée d'entrée :
--                  @ID_Variable int
-- Code erreur :
--                  1 => la variable n'existe pas
-- =====
ALTER PROCEDURE [dbo].[ICONE_sp_delete_historisable]

@ID_Variable int

AS
BEGIN

    if (SELECT COUNT(*) FROM dbo.Variables WHERE ID = @ID_Variable) = 0
    begin
        return 1;
    end

    begin try
        DELETE FROM dbo.Historisable
        WHERE Variable_ID = @ID_Variable
    end try
    begin catch
        return -1;
    end catch

END
```

6.1.3. CRUD dbo.Alarme

Create :

```
USE [BASE_ICONECT]
GO
/***** Object:  StoredProcedure [dbo].[ICONE_sp_create_alarme]    Script Date:
27/01/2023 11:16:23 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      Allan ESCOLANO
-- Description:  Crée la ligne Alarme
-- Donnée d'entrée :
--      @ID int,
--      @Type_alarme float,
--      @seuil float,
--      @Priority int,
--      @Message Nvarchar(max)
-- Code erreur :
--      1 => seuil ne peut pas etre NULL
--      2 => Priority ne peut pas etre NULL
--      100 => Aucune variable ne contient l'ID renseigné
--      101 => La variable.Alarme est a NULL
--      102 => La variable contient deja une instance de ce type d'alarme
--      103 => Le type d'alarme fournis n'est pas valide (1 = HIHI | 2 = HI | 3 = LO |
4 = LOL0)
-- =====

ALTER PROCEDURE [dbo].[ICONE_sp_create_alarme]
    @ID int,
    @Type_alarme int,
    @seuil float,
    @Priority int,
    @Message Nvarchar(max)
AS
begin
    -- on verifie si la variable existe
    if (SELECT COUNT(*) FROM dbo.Variables WHERE ID = @ID) = 0
    begin
        return 100;
    end

    -- on verifie que la variable.Alarme est TRUE
    declare @variable_alarme bit;
    select @variable_alarme = Alarme FROM dbo.Variables WHERE ID = @ID
    if (@variable_alarme = 0 or @variable_alarme is null)
    begin
        return 101;
    end
end
```

```
-- on verifie si il existe pas une ligne variable_ID + Alarme_type
if (SELECT COUNT(*) FROM dbo.Alarme WHERE Variable_ID = @ID and Type_alarme =
@Type_alarme) > 0
begin
    return 102;
end

if (@Type_alarme < 1 or @Type_alarme > 4)
begin
    return 103;
end

-- on verifie que seuil n'est pas NULL
if (@seuil is null)
begin
    return 1;
end

-- on verifie que Priority n'est pas NULL
if (@Priority is null)
begin
    return 2;
end

begin try
    INSERT INTO dbo.Alarme (Variable_ID, Type_alarme, seuil, Priority,
Message)
    VALUES (@ID, @Type_alarme, @seuil, @Priority, @Message)
end try
begin catch
    return -1;
end catch

end
```

Read :

```
USE [BASE_ICONECT]
GO
/***** Object:  StoredProcedure [dbo].[ICONE_sp_get_alarme]      Script Date:
27/01/2023 11:16:44 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          Allan ESCOLANO
-- Description:      Recupere les alarmes de la variables
-- Donnée d'entrée :
--      @ID int,
-- Code erreur :
--      100 => Aucune variable ne contient l'ID renseigné
--      101 => variable.Alarme est a NULL
--      102 => La variable contient aucune alarme
-- =====

ALTER PROCEDURE [dbo].[ICONE_sp_get_alarme]
    @ID int
AS
BEGIN

    -- on verifie si la variable existe
    if (SELECT COUNT(*) FROM dbo.Variables WHERE ID = @ID) = 0
    begin
        return 100;
    end

    -- on verifie que la variable.Alarme est TRUE
    declare @variable_alarm bit;
    select @variable_alarm = Alarme FROM dbo.Variables WHERE ID = @ID
    if (@variable_alarm = 0 or @variable_alarm is null)
    begin
        return 101;
    end

    -- on verifie si la variable contient des alarmes
    if (SELECT COUNT(*) FROM dbo.Alarme WHERE Variable_ID = @ID) = 0
    begin
        return 102;
    end

    begin try
        SELECT Variable_ID, Type_alarm, seuil, Priority, Message FROM dbo.Alarme
        WHERE Variable_ID = @ID
    end try
    begin catch
        return -1;
    end catch

END
```

Update :

```
USE [BASE_ICONECT]
GO
/***** Object: StoredProcedure [dbo].[ICONE_sp_update_alarme]    Script Date:
27/01/2023 11:17:10 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          Allan ESCOLANO
-- Description:      Met a jour une alarme de la variable
-- Donnée d'entrée :
--          @ID int,
--          @Target_Type_alarme float,
--          @seuil float,
--          @Priority int,
--          @Message Nvarchar(max)
-- Code erreur :
--      1 => seuil ne peut pas etre NULL
--      2 => Priority ne peut pas etre NULL
--      100 => Aucune variable ne contient l'ID renseigné
--      101 => La variable.Alarme est a NULL
--      103 => Le type d'alarme cible ne respecte pas le format (1 = HIHI | 2 = HI | 3
= LO | 4 = LOL0)
--      104 => La variable ne contient pas d'alarme de ce type a modifié
-- =====

ALTER PROCEDURE [dbo].[ICONE_sp_update_alarme]
    @ID int,
    @Target_Type_alarme float,
    @seuil float,
    @Priority int,
    @Message Nvarchar(max)
AS
begin
    -- on verifie si la variable existe
    if (SELECT COUNT(*) FROM dbo.Variables WHERE ID = @ID) = 0
    begin
        return 100;
    end

    -- on verifie que la variable.Alarme est TRUE
    declare @variable_alarme bit;
    select @variable_alarme = Alarme FROM dbo.Variables WHERE ID = @ID
    if (@variable_alarme = 0 or @variable_alarme is null)
    begin
        return 101;
    end

    -- on verifie le format du type d'alarme cible
    if (@Target_Type_alarme < 1 or @Target_Type_alarme > 4)
    begin
        return 103;
    end

    if (select count(*) from Alarme where Variable_ID = @ID AND Type_alarm =
@Target_Type_alarm) = 0
```

```
begin
    return 104;
end

-- on verifie que seuil n'est pas NULL
if (@seuil is null)
begin
    return 1;
end

-- on verifie que Priority n'est pas NULL
if (@Priority is null)
begin
    return 2;
end

begin try
    UPDATE dbo.Alarme
    SET seuil = @seuil, Priority = @Priority, Message = @Message
    WHERE Variable_ID = @ID
    AND Type_alarme = @Target_Type_alarme
end try
begin catch
    return -1;
end catch

end
```


Delete :

```
USE [BASE_ICONECT]
GO
/***** Object:  StoredProcedure [dbo].[ICONE_sp_delete_alarme]    Script Date:
27/01/2023 11:17:43 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      ESCOLANO Allan
-- Description:  Supprime les alarmes d'une variable
-- Donnée d'entrée :
--      @ID_Variable int
-- Code erreur :
--      1 => la variable n'existe pas
-- =====
ALTER PROCEDURE [dbo].[ICONE_sp_delete_alarme]

@ID_Variable int

AS
BEGIN

    if (SELECT COUNT(*) FROM dbo.Variables WHERE ID = @ID_Variable) = 0
    begin
        return 1;
    end

    begin try
        DELETE FROM dbo.Alarme
        WHERE Variable_ID = @ID_Variable
    end try
    begin catch
        return -1;
    end catch

END
```

6.1.4 Create Equation

```
USE [BASE_ICONECT]
GO
/***** Object: StoredProcedure [dbo].[ICONE_sp_create_equation]    Script Date:
27/01/2023 11:18:53 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          ESCOLANO Allan
-- Description:      Crée un équation
-- Donnée d'entrée :
--      @id_variable_calculé int,
--      @id_utilisateur int,
--      @list_libelles varchar(max), -- FORMAT : Variable|Operateur|Constante
--      @list_IDs varchar(max) -- FORMAT : id1|id2|id3
-- Code erreur :
--      1 => l'utilisateur essaye de modifier une équation lié à une variable qui
n'appartient ni à lui ni à son client
--      2 => la variable calculée fournie n'est pas une variable calculée
--      3 => l'utilisateur n'a pas le niveau d'accès nécessaire pour créer une
équation
--      4 => La variable calculé n'existe pas
--      5 => la variable calculé ne contient pas l'option Combinatoire
--      6 => la variable calculé contient déjà un calcul
--      10 => la liste des libelles est null
--      11 => la liste d'ids est null
--      100 => l'utilisateur fournie n'existe pas
-- =====
ALTER PROCEDURE [dbo].[ICONE_sp_create_equation]
    @id_variable_calculé int,
    @id_utilisateur int,
    @list_libelles varchar(max),
    @list_IDs varchar(max)
AS
BEGIN
    -- on vérifie que l'utilisateur existe
    if (select count(*) from Utilisateurs where id = @id_utilisateur) = 0
    begin
        return 100;
    end

    -- on vérifie que l'utilisateur est celui qui a créé la variable ou qu'elle
appartient au même client
    if (select id_createur from Variables where id = @id_variable_calculé) !=
    @id_utilisateur
        AND (select id_client from Utilisateurs where id = @id_utilisateur)
        =
        (select id_client from Utilisateurs where id = (
            select id_createur from Variables where id = @id_variable_calculé
        ))
    begin
```

```

        return 1;
    end

    -- on vérifie que la variable est une variable calculée
    if (select id_equipement from Variables where id = @id_variable_calcule) is not
null
    begin
        return 2;
    end

    -- on vérifie que l'utilisateur a le niveau d'accès nécessaire
    declare @niveau_acces int;
    select @niveau_acces = Niveau_acces from roles where id = (
        select id_role from Utilisateurs where id = @id_utilisateur
    );
    declare @createur int;
    select @createur = id_createur from Variables where id = @id_variable_calcule
    if (@niveau_acces < 7000 AND @createur != @id_utilisateur)
    OR
    ( @niveau_acces < 6000 AND @createur = @id_utilisateur)
    begin
        return 3;
    end

    --verification que la variable calculé existe
    if (select count(*) from dbo.Variables where ID = @id_variable_calcule) = 0
    begin
        return 4;
    end

    --verification que la variable a l'option combinatoire
    declare @combinatoire bit;
    select @combinatoire = Combinatoire from dbo.Variables where ID =
    @id_variable_calcule
    if (@combinatoire != 1)
    begin
        return 5;
    end

    --Verification qu'il n'existe pas déjà un calcul pour la variable
    if (select COUNT(*) from dbo.Equations where id_variable_calcule =
    @id_variable_calcule) > 0
    begin
        return 6;
    end

    --Verification que la liste n'est pas null
    if (@list_IDs is null)
    begin
        return 10;
    end

    --Verification que la liste n'est pas null
    if (@list_libelles is null)
    begin
        return 11;
    end
end

```

```
-- on démarre un try catch, normalement les erreurs ont déjà été attrapées, si une
erreur a été omise, on retourne -1
```

```
begin try
    --variables pour parcourir la liste de libelle
    DECLARE @pos INT
    DECLARE @len INT
    DECLARE @value varchar(8000)
    declare @posMot INT

    --variables pour parcourir la liste d'ids
    DECLARE @pos2 INT
    DECLARE @len2 INT
    DECLARE @value2 varchar(8000)
    declare @posMot2 INT

    --il faut que les listes finissent par un | sinon le dernier mot n'est
    pas pris en compte
    IF @list_libelles NOT LIKE '%| '
    BEGIN
        set @list_libelles = @list_libelles + '| '
    END

    IF @list_IDS NOT LIKE '%| '
    BEGIN
        set @list_IDS = @list_IDS + '| '
    END

    set @pos = 0
    set @len = 0
    set @posMot = 0
    --BOUCLE A TRAVERS LA LISTE DE LIBELLES
    WHILE CHARINDEX('|', @list_libelles, @pos+1)>0
    BEGIN
        set @len = CHARINDEX('|', @list_libelles, @pos+1) - @pos
        set @value = SUBSTRING(@list_libelles, @pos, @len)

        set @pos2 = 0
        set @len2 = 0
        set @posMot2 = 0
        --BOUCLE A TRAVERS LA LISTE D'IDS
        WHILE CHARINDEX('|', @list_IDS, @pos2+1)>0
        BEGIN
            set @len2 = CHARINDEX('|', @list_IDS, @pos2+1) -
@pos2

            set @value2 = SUBSTRING(@list_IDS, @pos2, @len2)

            --SI LE 1 MOT DE LA LISTE 1 CROISE LE MOT 1 DE LA
            LISTE 2

            if (@posMot = @posMot2)
            begin
                if (@value = 'Variable')
                begin
                    select @value as Libelle, @value2 as
VariableID, @posMot + 1 AS PLACE
                    INSERT INTO dbo.Equations (PLACE
, id_variable_calcule, id_variable, id_operateur, constante)
```

```

VALUES (@posMot +1
,@id_variable_calcule, @value2, NULL, NULL)
end
if (@value = 'Operateur')
begin
select @value as Libelle, @value2 as
OperateurID, @posMot + 1 AS PLACE
INSERT INTO dbo.Equations (PLACE
,id_variable_calcule, id_variable, id_operateur, constante)
VALUES (@posMot +1
,@id_variable_calcule, NULL, @value2, NULL)
end
if (@value = 'Constante')
begin
select @value as Libelle, @value2 as
ConstanteValue, @posMot + 1 AS PLACE
INSERT INTO dbo.Equations (PLACE
,id_variable_calcule, id_variable, id_operateur, constante)
VALUES (@posMot +1
,@id_variable_calcule, NULL, NULL, @value2)
end
end
set @pos2 = CHARINDEX('|', @list_IDS, @pos2+@len2)
+1
set @posMot2 += 1
END

set @pos = CHARINDEX('|', @list_libelles, @pos+@len) +1
set @posMot += 1
END
RETURN 0
end try
begin catch
print ERROR_MESSAGE();
return -1;
end catch
END

```

6.1.5 GET_sp

ICONE_sp_get_notification_email :

```
USE [BASE_ICONECT]
GO
/***** Object:  StoredProcedure [dbo].[ICONE_sp_get_notification_email]    Script
Date: 27/01/2023 11:19:57 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          ESCOLANO Allan
-- Description:      Renvoie les EMAIL des utilisateurs qui ont une notification pour
la variable
-- Donnée d'entrée :
--      @ID_Variable
-- Code erreur :
--      1 => La variable avec L'ID fournie n'existe pas
-- =====

ALTER PROCEDURE [dbo].[ICONE_sp_get_notification_email]

@ID_Variable int

AS
BEGIN

    --Verification que l'ID de la variable existe
    if (SELECT count(*) FROM dbo.Variables WHERE ID = @ID_Variable) = 0
    begin
        return 1;
    end

    begin try
        SELECT id_variable, id_utilisateur, Mail FROM
dbo.Evenement_notifications_utilisateurs, dbo.Utilisateurs
        WHERE id_variable = @ID_Variable
        AND id_utilisateur = ID
    end try
    begin catch
        return -1;
    end catch

END
```

ICONE_sp_get_notification_sms :

```
USE [BASE_ICONECT]
GO
/***** Object:  StoredProcedure [dbo].[ICONE_sp_get_notification_sms]    Script Date:
27/01/2023 11:20:31 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      ESCOLANO Allan
-- Description:  Recupere les numéro de telephone des utilisateurs avec une
notification pour la variable
-- Donnée d'entrée :
--      @ID_Variable
-- Code erreur :
--      1 => La variable avec L'ID fournie n'existe pas
-- =====

ALTER PROCEDURE [dbo].[ICONE_sp_get_notification_sms]

@ID_Variable int

AS
BEGIN

    --Verification que l'ID de la variable existe
    if (SELECT count(*) FROM dbo.Variables WHERE ID = @ID_Variable) = 0
    begin
        return 1;
    end

    begin try
        SELECT id_variable, id_utilisateur, Mobile FROM
    dbo.Evenement_notifications_utilisateurs, dbo.Utilisateurs
        WHERE id_variable = @ID_Variable
        AND id_utilisateur = ID
    end try
    begin catch
        return -1;
    end catch

END
```

ICONE_sp_get_variable_options :

```
USE [BASE_ICONECT]
GO
/***** Object:  StoredProcedure [dbo].[ICONE_sp_get_notification_sms]    Script Date:
27/01/2023 11:20:31 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      ESCOLANO Allan
-- Description:  Recupere les numéro de telephone des utilisateurs avec une
notification pour la variable
-- Donnée d'entrée :
--      @ID_Variable
-- Code erreur :
--      1 => La variable avec L'ID fournie n'existe pas
-- =====

ALTER PROCEDURE [dbo].[ICONE_sp_get_notification_sms]

@ID_Variable int

AS
BEGIN

    --Verification que l'ID de la variable existe
    if (SELECT count(*) FROM dbo.Variables WHERE ID = @ID_Variable) = 0
    begin
        return 1;
    end

    begin try
        SELECT id_variable, id_utilisateur, Mobile FROM
    dbo.Evenement_notifications_utilisateurs, dbo.Utilisateurs
        WHERE id_variable = @ID_Variable
        AND id_utilisateur = ID
    end try
    begin catch
        return -1;
    end catch

END
```


6.2. Gestionnaires licences sp.

6.2.1. CRUD Client/Fonctions/Integrateurs

Le CRUD suivant est le même pour les 3 tables, seulement le code visé est changer (sois Code_client/Code_fonction/Code_integrateur) :

Create :

```
USE [Base_NewLicences]
GO
/***** Object:  StoredProcedure [dbo].[create_Client]    Script Date: 27/01/2023
12:00:48 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[create_Client]
@Nom varchar(255),
@Date_update datetime2(7),
@Description varchar(255)
AS
BEGIN
    INSERT INTO dbo.Clients(Nom, Date_update, Description)
    VALUES (@Nom, @Date_update, @Description)
END
```

Read :

```
USE [Base_NewLicences]
GO
/***** Object:  StoredProcedure [dbo].[get_Clients]      Script Date: 27/01/2023
12:01:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[get_Clients]
AS
BEGIN
    SELECT * FROM dbo.Clients
END
```

Update :

```
USE [Base_NewLicences]
GO
/***** Object: StoredProcedure [dbo].[update_Client]      Script Date: 27/01/2023
12:02:11 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[update_Client]
@Code_client INT,
@Nom varchar(255),
@Date_update datetime2(7),
@Description varchar(255)

AS
BEGIN
    UPDATE dbo.Clients
    SET Nom = @Nom,
        Date_update = @Date_update,
        Description = @Description
    WHERE Code_client = @Code_client
END
```

Delete :

```
USE [Base_NewLicences]
GO
/***** Object: StoredProcedure [dbo].[delete_Client]      Script Date: 27/01/2023
12:03:24 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[delete_Client]

@Code_client int

AS
BEGIN
    DELETE FROM dbo.Clients
    WHERE Code_client = @Code_client
    declare @max int
    SELECT @max = MAX(Code_client) FROM dbo.Clients
    dbcc checkident('Clients', reseed, @max)
END
```

6.2.2. CRUD Licences

Create :

```
USE [Base_NewLicences]
GO
/***** Object: StoredProcedure [dbo].[create_Licence]    Script Date: 27/01/2023
12:05:11 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[create_Licence]

@Date_creation datetime2(7),
@Date_expiration datetime2(7),
@Nb_equipements int,
@Nb_variables int,
@Checksum varchar(2),
@Code_integrateur int,
@Code_client int,
@Code_fonction1 int,
@Code_fonction2 int,
@Code_fonction3 int,
@Code_fonction4 int,
@Code_fonction5 int,
@Code_fonction6 int,
@Code_fonction7 int,
@Code_fonction8 int,
@Code_fonction9 int,
@Code_fonction10 int,
@Code_fonction11 int,
@Code_fonction12 int,
@Code_fonction13 int,
@Code_fonction14 int,
@Code_fonction15 int,
@Code_fonction16 int,
@Code_fonction17 int,
@Code_fonction18 int,
@Code_fonction19 int,
@Code_fonction20 int

AS
BEGIN
    INSERT INTO dbo.Licences (Date_creation, Date_expiration, Nb_equipements,
Nb_variables, Checksum, Code_integrateur, Code_client,
    Code_fonction1, Code_fonction2, Code_fonction3, Code_fonction4, Code_fonction5,
Code_fonction6, Code_fonction7, Code_fonction8,
    Code_fonction9, Code_fonction10, Code_fonction11, Code_fonction12,
Code_fonction13, Code_fonction14, Code_fonction15, Code_fonction16,
    Code_fonction17, Code_fonction18, Code_fonction19, Code_fonction20)

    VALUES (@Date_creation, @Date_expiration, @Nb_equipements, @Nb_variables,
@Checksum, @Code_integrateur, @Code_client,
    @Code_fonction1, @Code_fonction2, @Code_fonction3, @Code_fonction4,
@Code_fonction5, @Code_fonction6, @Code_fonction7, @Code_fonction8,
```

```

        @Code_fonction9, @Code_fonction10, @Code_fonction11, @Code_fonction1,
@Code_fonction13, @Code_fonction14, @Code_fonction15, @Code_fonction16,
        @Code_fonction17, @Code_fonction18, @Code_fonction19, @Code_fonction20)
END

```

Read :

```

USE [Base_NewLicences]
GO
/***** Object:  StoredProcedure [dbo].[get_Licences]    Script Date: 27/01/2023
12:05:36 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[get_Licences]

AS
BEGIN
    SELECT * FROM dbo.Licences
END

```

Update :

```

USE [Base_NewLicences]
GO
/***** Object:  StoredProcedure [dbo].[update_Licence]  Script Date: 27/01/2023
12:05:52 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[update_Licence]

@ID int,
@Date_expiration datetime2(7),
@Nb_equipements int,
@Nb_variables int,
@Checksum varchar(2),
@Code_integrateur int,
@Code_client int,
@Code_fonction1 int,
@Code_fonction2 int,
@Code_fonction3 int,
@Code_fonction4 int,
@Code_fonction5 int,
@Code_fonction6 int,
@Code_fonction7 int,
@Code_fonction8 int,
@Code_fonction9 int,
@Code_fonction10 int,
@Code_fonction11 int,
@Code_fonction12 int,
@Code_fonction13 int,
@Code_fonction14 int,

```

```

@Code_fonction15 int,
@Code_fonction16 int,
@Code_fonction17 int,
@Code_fonction18 int,
@Code_fonction19 int,
@Code_fonction20 int

AS
BEGIN
    UPDATE dbo.Licences

    SET Date_expiration = @Date_expiration,
    Nb_equipements = @Nb_equipements,
    Nb_variables = @Nb_variables,
    Checksum = @Checksum,
    Code_integrateur = @Code_integrateur,
    Code_client = @Code_client,
    Code_fonction1 = @Code_fonction1,
    Code_fonction2 = @Code_fonction2,
    Code_fonction3 = @Code_fonction3,
    Code_fonction4 = @Code_fonction4,
    Code_fonction5 = @Code_fonction5,
    Code_fonction6 = @Code_fonction6,
    Code_fonction7 = @Code_fonction7,
    Code_fonction8 = @Code_fonction8,
    Code_fonction9 = @Code_fonction9,
    Code_fonction10 = @Code_fonction10,
    Code_fonction11 = @Code_fonction11,
    Code_fonction12 = @Code_fonction12,
    Code_fonction13 = @Code_fonction13,
    Code_fonction14 = @Code_fonction14,
    Code_fonction15 = @Code_fonction15,
    Code_fonction16 = @Code_fonction16,
    Code_fonction17 = @Code_fonction17,
    Code_fonction18 = @Code_fonction18,
    Code_fonction19 = @Code_fonction19,
    Code_fonction20 = @Code_fonction20

    WHERE ID_Licence = @ID

END

```

Delete :

```

USE [Base_NewLicences]
GO
/***** Object: StoredProcedure [dbo].[delete_Licence]    Script Date: 27/01/2023
12:06:12 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[delete_Licence]

@ID int

AS

```

```
BEGIN
DELETE FROM dbo.Licences
WHERE ID_Licence = @ID
declare @max int
SELECT @max = MAX(ID_Licence) FROM dbo.Licences
dbcc checkident('Licences', reseed, @max)
END
```

6.2.3. Fonctionnalités.

Le code suivant représente l'entièreté du code de l'application avec toutes les fonctionnalités :

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Globalization;

namespace Gestionnaire_Licences
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;Initial
Catalog=Base_NewLicences;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>();

            private void Form1_Load(object sender, EventArgs e)
            {
                //Suppressions des boutons du tabcontrol
                tc_Licences.Appearance = TabAppearance.FlatButtons;
                tc_Licences.ItemSize = new Size(0, 1);
                tc_Licences.SizeMode = TabSizeMode.Fixed;

                tc_Clients.Appearance = TabAppearance.FlatButtons;
                tc_Clients.ItemSize = new Size(0, 1);
                tc_Clients.SizeMode = TabSizeMode.Fixed;

                tc_Fonctions.Appearance = TabAppearance.FlatButtons;
                tc_Fonctions.ItemSize = new Size(0, 1);
                tc_Fonctions.SizeMode = TabSizeMode.Fixed;
```

```

tc_Integrateurs.Appearance = TabAppearance.FlatButtons;
tc_Integrateurs.ItemSize = new Size(0, 1);
tc_Integrateurs.SizeMode = TabSizeMode.Fixed;

fill_dgv_Licences();
fill_dgv_Integrateurs();
fill_dgv_Clients();
fill_dgv_Fonctions();
}

//Pour la date de creation d'une licence
private void Timer_DateCreation_Tick(object sender, EventArgs e)
{
    tb_DateCreation.Text = DateTime.Now.ToString("dd/MM/yyyy");
    tb_C_DateMiseAJourClient.Text = DateTime.Now.ToString("dd/MM/yyyy");
    tb_C_DateMiseAJourFonction.Text = DateTime.Now.ToString("dd/MM/yyyy");
    tb_C_DateMiseAJour_Integrateur.Text =
DateTime.Now.ToString("dd/MM/yyyy");
    tb_U_DateMiseAJourClient.Text = DateTime.Now.ToString("dd/MM/yyyy");
    tb_U_DateMiseAJourFonction.Text = DateTime.Now.ToString("dd/MM/yyyy");
    tb_U_DateMiseAJourInteg.Text = DateTime.Now.ToString("dd/MM/yyyy");
}

//Remet a jour (depuis la bdd) les listes, les datagridview et les classes
private void fill_dgv_Licences()
{
    //Definition de la procédure stockée
    SqlCommand cmd = new SqlCommand("Base_NewLicences.dbo.get_Licences",
con);

    cmd.CommandType = CommandType.StoredProcedure;

    //Alimentation de la DataGridView
    SqlDataAdapter sd = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    sd.Fill(dt);
    dgv_Licences.DataSource = dt;

    try
    {
        con.Open();
        SqlDataReader rdr = cmd.ExecuteReader();
        lesLicences.Clear();
        #region lecture des licences
        while (rdr.Read())
        {
            int ID = Convert.ToInt32(rdr[0]);
            DateTime Date_creation = Convert.ToDateTime(rdr[1]);
            DateTime Date_expiration = Convert.ToDateTime(rdr[2]);
            int Nb_equipements = Convert.ToInt32(rdr[3]);
            int Nb_variables = Convert.ToInt32(rdr[4]);
            string Checksum = Convert.ToString(rdr[5]);
            int Code_integrateur = Convert.ToInt32(rdr[6]);
            int Code_client = Convert.ToInt32(rdr[7]);
            int Code_fonction1 = 0;
            int Code_fonction2 = 0;

```



```
int Code_fonction3 = 0;
int Code_fonction4 = 0;
int Code_fonction5 = 0;
int Code_fonction6 = 0;
int Code_fonction7 = 0;
int Code_fonction8 = 0;
int Code_fonction9 = 0;
int Code_fonction10 = 0;
int Code_fonction11 = 0;
int Code_fonction12 = 0;
int Code_fonction13 = 0;
int Code_fonction14 = 0;
int Code_fonction15 = 0;
int Code_fonction16 = 0;
int Code_fonction17 = 0;
int Code_fonction18 = 0;
int Code_fonction19 = 0;
int Code_fonction20 = 0;
if (rdr[8].ToString() != "")
{
    Code_fonction1 = Convert.ToInt32(rdr[8]);
}
if (rdr[9].ToString() != "")
{
    Code_fonction2 = Convert.ToInt32(rdr[9]);
}
if (rdr[10].ToString() != "")
{
    Code_fonction3 = Convert.ToInt32(rdr[10]);
}
if (rdr[11].ToString() != "")
{
    Code_fonction4 = Convert.ToInt32(rdr[11]);
}
if (rdr[12].ToString() != "")
{
    Code_fonction5 = Convert.ToInt32(rdr[12]);
}
if (rdr[13].ToString() != "")
{
    Code_fonction6 = Convert.ToInt32(rdr[13]);
}
if (rdr[14].ToString() != "")
{
    Code_fonction7 = Convert.ToInt32(rdr[14]);
}
if (rdr[15].ToString() != "")
{
    Code_fonction8 = Convert.ToInt32(rdr[15]);
}
if (rdr[16].ToString() != "")
{
    Code_fonction9 = Convert.ToInt32(rdr[16]);
}
if (rdr[17].ToString() != "")
{
    Code_fonction10 = Convert.ToInt32(rdr[17]);
}
```

```

    }
    if (rdr[18].ToString() != "")
    {
        Code_fonction11 = Convert.ToInt32(rdr[18]);
    }
    if (rdr[19].ToString() != "")
    {
        Code_fonction12 = Convert.ToInt32(rdr[19]);
    }
    if (rdr[20].ToString() != "")
    {
        Code_fonction13 = Convert.ToInt32(rdr[20]);
    }
    if (rdr[21].ToString() != "")
    {
        Code_fonction14 = Convert.ToInt32(rdr[21]);
    }
    if (rdr[22].ToString() != "")
    {
        Code_fonction15 = Convert.ToInt32(rdr[22]);
    }
    if (rdr[23].ToString() != "")
    {
        Code_fonction16 = Convert.ToInt32(rdr[23]);
    }
    if (rdr[24].ToString() != "")
    {
        Code_fonction17 = Convert.ToInt32(rdr[24]);
    }
    if (rdr[25].ToString() != "")
    {
        Code_fonction18 = Convert.ToInt32(rdr[25]);
    }
    if (rdr[26].ToString() != "")
    {
        Code_fonction19 = Convert.ToInt32(rdr[26]);
    }
    if (rdr[27].ToString() != "")
    {
        Code_fonction20 = Convert.ToInt32(rdr[27]);
    }
    Licence newLicence = new Licence(ID, Date_creation,
Date_expiration, Nb_equipements, Nb_variables, Checksum, Code_integrateur,
Code_client
        , Code_fonction1, Code_fonction2, Code_fonction3,
Code_fonction4, Code_fonction5, Code_fonction6, Code_fonction7, Code_fonction8,
Code_fonction9, Code_fonction10
        , Code_fonction11, Code_fonction12, Code_fonction13,
Code_fonction14, Code_fonction15, Code_fonction16, Code_fonction17,
Code_fonction18, Code_fonction19, Code_fonction20);
    lesLicences.Add(newLicence);
}
#endregion
con.Close();
} catch (Exception err)
{

```

```

        MessageBox.Show("Erreur lecture/récupération des licences : " +
err.Message);
    }
}
private void fill_dgv_Integrateurs()
{
    //Definition de la procédure stockée
    SqlCommand cmd = new
SqlCommand("Base_NewLicences.dbo.get_Integrateurs", con);
    cmd.CommandType = CommandType.StoredProcedure;

    //Alimentation de la DataGridView
    SqlDataAdapter sd = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    sd.Fill(dt);
    dgv_Integrateurs.DataSource = dt;
    dgv_Integrateurs.Columns[3].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
    dgv_Licence_AllIntegrateurs.DataSource = dt;
    dgv_Licence_AllIntegrateurs2.DataSource = dt;

    try
    {
        con.Open();
        SqlDataReader rdr = cmd.ExecuteReader();
        lesIntegrateurs.Clear();
        #region lecture des integrateurs
        while (rdr.Read())
        {
            int ID = Convert.ToInt32(rdr[0]);
            string Nom = Convert.ToString(rdr[1]);
            DateTime Date_update = Convert.ToDateTime(rdr[2]);
            string Description = Convert.ToString(rdr[3]);
            Integrateur newIntegrateur = new Integrateur(ID, Nom,
Date_update, Description);
            lesIntegrateurs.Add(newIntegrateur);
        }
        #endregion
        con.Close();
    }
    catch (Exception err)
    {
        MessageBox.Show("Erreur lecture/récupération des Integrateurs : "
+ err.Message);
    }
}
private void fill_dgv_Clients()
{
    //Definition de la procédure stockée
    SqlCommand cmd = new SqlCommand("Base_NewLicences.dbo.get_Clients",
con);

    cmd.CommandType = CommandType.StoredProcedure;

    //Alimentation de la DataGridView
    SqlDataAdapter sd = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    sd.Fill(dt);

```

```

dgv_Clients.DataSource = dt;
dgv_Clients.Columns[3].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
dgv_Licence_AllClients.DataSource = dt;
dgv_Licence_AllClients2.DataSource = dt;

try
{
    con.Open();
    SqlDataReader rdr = cmd.ExecuteReader();
    lesClients.Clear();
    #region lecture des clients
    while (rdr.Read())
    {
        int ID = Convert.ToInt32(rdr[0]);
        string Nom = Convert.ToString(rdr[1]);
        DateTime Date_update = Convert.ToDateTime(rdr[2]);
        string Description = Convert.ToString(rdr[3]);
        Client newClient = new Client(ID, Nom, Date_update,
Description);
        lesClients.Add(newClient);
    }
    #endregion
    con.Close();
} catch (Exception err)
{
    MessageBox.Show("Erreur lecture/récupération des Clients : " +
err.Message);
}
}
private void fill_dgv_Fonctions()
{
    //Definition de la procédure stockée
    SqlCommand cmd = new SqlCommand("Base_NewLicences.dbo.get_Fonctions",
con);
    cmd.CommandType = CommandType.StoredProcedure;

    //Alimentation de la DataGridView
    SqlDataAdapter sd = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    sd.Fill(dt);
    dgv_Fonctions.DataSource = dt;
    dgv_Fonctions.Columns[3].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
    dgv_Licence_AllFonctions.DataSource = dt;
    dgv_Licence_AllFonctions2.DataSource = dt;

    try
    {
        con.Open();
        SqlDataReader rdr = cmd.ExecuteReader();
        lesFonctions.Clear();
        #region lecture des Fonctions
        while (rdr.Read())
        {
            int ID = Convert.ToInt32(rdr[0]);
            string Nom = Convert.ToString(rdr[1]);

```

```

        DateTime Date_update = Convert.ToDateTime(rdr[2]);
        string Description = Convert.ToString(rdr[3]);
        Fonction newFonction = new Fonction(ID, Nom, Date_update,
Description);
        lesFonctions.Add(newFonction);
    }
    #endregion
    con.Close();
} catch (Exception err)
{
    MessageBox.Show("Erreur lecture/récupération des Fonction : " +
err.Message);
}
}

private void retour(object sender, EventArgs e)
{
    if (sender == btn_RetourLicence)
    {
        tc_Licences.Height = 493;
        dgv_Licence_AllClients2.Visible = true;
        dgv_Licence_AllFonctions2.Visible = true;
        dgv_Licence_AllIntegrateurs2.Visible = true;
        lb_dgv_AllClients2.Visible = true;
        lb_dgv_AllFonctions2.Visible = true;
        lb_dgv_AllIntegrateurs2.Visible = true;
        tc_Licences.SelectedIndex = 0;
    }
    if (sender == btn_RetourInteg)
    {
        tc_Integrateurs.SelectedIndex = 0;
    }
    if (sender == btn_RetourClient)
    {
        tc_Clients.SelectedIndex = 0;
    }
    if (sender == btn_RetourFonction)
    {
        tc_Fonctions.SelectedIndex = 0;
    }
}

//Rajout d'une licence dans la base
private void create_Licence(object sender, EventArgs e)
{
    try
    {
        SqlCommand cmd = new
SqlCommand("Base_NewLicences.dbo.create_Licence", con);
        cmd.CommandType = CommandType.StoredProcedure;
        DateTime Date_creation = Convert.ToDateTime(tb_DateCreation.Text);
        DateTime Date_expiration =
Convert.ToDateTime(dtp_C_DateExpirationLicence.Value.ToString("dd/MM/yyyy"));
        int Nb_equipements = Convert.ToInt32(tb_NbEquipements.Text);
        int Nb_variables = Convert.ToInt32(tb_NbVariables.Text);
        string Checksum = Convert.ToString(tb_Checksum.Text);
    }
}

```

```

int Code_integrateur = Convert.ToInt32(tb_CodeIntegrateur.Text);
int Code_client = Convert.ToInt32(tb_CodeClient.Text);
char[] charsToTrim = { ' ' };

cmd.Parameters.Add("@Date_creation", SqlDbType.DateTime).Value =
Date_creation;
cmd.Parameters.Add("@Date_expiration", SqlDbType.DateTime).Value =
Date_expiration;
cmd.Parameters.Add("@Nb_equipements", SqlDbType.Int).Value =
Nb_equipements;
cmd.Parameters.Add("@Nb_variables", SqlDbType.Int).Value =
Nb_variables;
cmd.Parameters.Add("@Checksum", SqlDbType.VarChar).Value =
Checksum;
cmd.Parameters.Add("@Code_integrateur", SqlDbType.Int).Value =
Code_integrateur;
cmd.Parameters.Add("@Code_client", SqlDbType.Int).Value =
Code_client;

#region parametres code_fonctionX
if (tb_CodeFonction1.Text.Trim(charsToTrim) == "")
{
    cmd.Parameters.Add("@Code_fonction1", SqlDbType.Int).Value =
DBNull.Value;
}
else
{
    cmd.Parameters.Add("@Code_fonction1", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction1.Text.Trim(charsToTrim));
}

if (tb_CodeFonction2.Text.Trim(charsToTrim) == "")
{
    cmd.Parameters.Add("@Code_fonction2", SqlDbType.Int).Value =
DBNull.Value;
}
else
{
    cmd.Parameters.Add("@Code_fonction2", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction2.Text.Trim(charsToTrim));
}

if (tb_CodeFonction3.Text.Trim(charsToTrim) == "")
{
    cmd.Parameters.Add("@Code_fonction3", SqlDbType.Int).Value =
DBNull.Value;
}
else
{
    cmd.Parameters.Add("@Code_fonction3", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction3.Text.Trim(charsToTrim));
}

if (tb_CodeFonction4.Text.Trim(charsToTrim) == "")
{

```

```

        cmd.Parameters.Add("@Code_fonction4", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction4", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction4.Text.Trim(charsToTrim));
    }

    if (tb_CodeFonction5.Text.Trim(charsToTrim) == "")
    {
        cmd.Parameters.Add("@Code_fonction5", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction5", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction5.Text.Trim(charsToTrim));
    }

    if (tb_CodeFonction6.Text.Trim(charsToTrim) == "")
    {
        cmd.Parameters.Add("@Code_fonction6", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction6", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction6.Text.Trim(charsToTrim));
    }

    if (tb_CodeFonction7.Text.Trim(charsToTrim) == "")
    {
        cmd.Parameters.Add("@Code_fonction7", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction7", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction7.Text.Trim(charsToTrim));
    }

    if (tb_CodeFonction8.Text.Trim(charsToTrim) == "")
    {
        cmd.Parameters.Add("@Code_fonction8", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction8", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction8.Text.Trim(charsToTrim));
    }

    if (tb_CodeFonction9.Text.Trim(charsToTrim) == "")
    {
        cmd.Parameters.Add("@Code_fonction9", SqlDbType.Int).Value =
DBNull.Value;

```

```

    }
    else
    {
        cmd.Parameters.Add("@Code_fonction9", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction9.Text.Trim(charsToTrim));
    }

    if (tb_CodeFonction10.Text.Trim(charsToTrim) == "")
    {
        cmd.Parameters.Add("@Code_fonction10", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction10", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction10.Text.Trim(charsToTrim));
    }

    if (tb_CodeFonction11.Text.Trim(charsToTrim) == "")
    {
        cmd.Parameters.Add("@Code_fonction11", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction11", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction11.Text.Trim(charsToTrim));
    }

    if (tb_CodeFonction12.Text.Trim(charsToTrim) == "")
    {
        cmd.Parameters.Add("@Code_fonction12", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction12", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction12.Text.Trim(charsToTrim));
    }

    if (tb_CodeFonction13.Text.Trim(charsToTrim) == "")
    {
        cmd.Parameters.Add("@Code_fonction13", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction13", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction13.Text.Trim(charsToTrim));
    }

    if (tb_CodeFonction14.Text.Trim(charsToTrim) == "")
    {
        cmd.Parameters.Add("@Code_fonction14", SqlDbType.Int).Value =
DBNull.Value;
    }
    else

```



```

        {
            cmd.Parameters.Add("@Code_fonction14", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction14.Text.Trim(charsToTrim));
        }

        if (tb_CodeFonction15.Text.Trim(charsToTrim) == "")
        {
            cmd.Parameters.Add("@Code_fonction15", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction15", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction15.Text.Trim(charsToTrim));
        }

        if (tb_CodeFonction16.Text.Trim(charsToTrim) == "")
        {
            cmd.Parameters.Add("@Code_fonction16", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction16", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction16.Text.Trim(charsToTrim));
        }

        if (tb_CodeFonction17.Text.Trim(charsToTrim) == "")
        {
            cmd.Parameters.Add("@Code_fonction17", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction17", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction17.Text.Trim(charsToTrim));
        }

        if (tb_CodeFonction18.Text.Trim(charsToTrim) == "")
        {
            cmd.Parameters.Add("@Code_fonction18", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction18", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction18.Text.Trim(charsToTrim));
        }

        if (tb_CodeFonction19.Text.Trim(charsToTrim) == "")
        {
            cmd.Parameters.Add("@Code_fonction19", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {

```

```

        cmd.Parameters.Add("@Code_fonction19", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction19.Text.Trim(charsToTrim));
    }

    if (tb_CodeFonction20.Text.Trim(charsToTrim) == "")
    {
        cmd.Parameters.Add("@Code_fonction20", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction20", SqlDbType.Int).Value =
Convert.ToInt32(tb_CodeFonction20.Text.Trim(charsToTrim));
    }

    #endregion
    con.Open();
    cmd.ExecuteNonQuery();
    con.Close();
    fill_dgv_Licences();
} catch (Exception err)
{
    MessageBox.Show("Erreur create licence : " + err.Message);
}
}
private void Delete_Licence(object sender, EventArgs e)
{
    int ID = Convert.ToInt32(tb_R_IDLicence.Text);
    //Definition de la procédure stockée
    SqlCommand cmd = new SqlCommand("Base_NewLicences.dbo.delete_Licence",
con);

    cmd.CommandType = CommandType.StoredProcedure;

    cmd.Parameters.Add("@ID", SqlDbType.Int).Value = ID;

    con.Open();
    cmd.ExecuteNonQuery();
    con.Close();
    fill_dgv_Licences();
    tc_Licences.Height = 493;
    dgv_Licence_AllClients2.Visible = true;
    dgv_Licence_AllFonctions2.Visible = true;
    dgv_Licence_AllIntegrateurs2.Visible = true;
    lb_dgv_AllClients2.Visible = true;
    lb_dgv_AllFonctions2.Visible = true;
    lb_dgv_AllIntegrateurs2.Visible = true;
    tc_Licences.SelectedIndex = 0;
}
private void Update_Licence(object sender, EventArgs e)
{
    SqlCommand cmd = new SqlCommand("Base_NewLicences.dbo.update_Licence",
con);

    cmd.CommandType = CommandType.StoredProcedure;
    try
    {
        cmd.Parameters.Add("@ID", SqlDbType.Int).Value =
Convert.ToInt32(tb_R_IDLicence.Text);

```

```

        cmd.Parameters.Add("@Date_expiration", SqlDbType.DateTime).Value =
Convert.ToDateTime(dtp_U_DateExpirationLicence.Value.ToString("dd/MM/yyyy"));
        cmd.Parameters.Add("@Nb_equipements", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_NbrEquipements.Text);
        cmd.Parameters.Add("@Nb_variables", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_NbrVariables.Text);
        cmd.Parameters.Add("@Checksum", SqlDbType.VarChar).Value =
Convert.ToString(tb_C_Checksum.Text);
        cmd.Parameters.Add("@Code_integrateur", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeIntegrateur.Text);
        cmd.Parameters.Add("@Code_client", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeClient.Text);
        char[] charsToTrim = { ' ' };

        #region Parametres code fonction
        if (tb_C_CodeFonction1.Text.Trim(charsToTrim) == "" |
tb_C_CodeFonction1.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction1", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction1", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction1.Text.Trim(charsToTrim));
        }

        if (tb_C_CodeFonction2.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction2.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction2", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction2", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction2.Text.Trim(charsToTrim));
        }

        if (tb_C_CodeFonction3.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction3.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction3", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction3", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction3.Text.Trim(charsToTrim));
        }

        if (tb_C_CodeFonction4.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction4.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction4", SqlDbType.Int).Value =
DBNull.Value;

```

```

    }
    else
    {
        cmd.Parameters.Add("@Code_fonction4", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction4.Text.Trim(charsToTrim));
    }

    if (tb_C_CodeFonction5.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction5.Text.Trim(charsToTrim) == "0")
    {
        cmd.Parameters.Add("@Code_fonction5", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction5", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction5.Text.Trim(charsToTrim));
    }

    if (tb_C_CodeFonction6.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction6.Text.Trim(charsToTrim) == "0")
    {
        cmd.Parameters.Add("@Code_fonction6", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction6", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction6.Text.Trim(charsToTrim));
    }

    if (tb_C_CodeFonction7.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction7.Text.Trim(charsToTrim) == "0")
    {
        cmd.Parameters.Add("@Code_fonction7", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction7", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction7.Text.Trim(charsToTrim));
    }

    if (tb_C_CodeFonction8.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction8.Text.Trim(charsToTrim) == "0")
    {
        cmd.Parameters.Add("@Code_fonction8", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction8", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction8.Text.Trim(charsToTrim));
    }

    if (tb_C_CodeFonction9.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction9.Text.Trim(charsToTrim) == "0")

```

```

        {
            cmd.Parameters.Add("@Code_fonction9", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction9", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction9.Text.Trim(charsToTrim));
        }

        if (tb_C_CodeFonction10.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction10.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction10", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction10", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction10.Text.Trim(charsToTrim));
        }

        if (tb_C_CodeFonction11.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction11.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction11", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction11", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction11.Text.Trim(charsToTrim));
        }

        if (tb_C_CodeFonction12.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction12.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction12", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction12", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction12.Text.Trim(charsToTrim));
        }

        if (tb_C_CodeFonction13.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction13.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction13", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction13", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction13.Text.Trim(charsToTrim));
        }

```

```
        if (tb_C_CodeFonction14.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction14.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction14", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction14", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction14.Text.Trim(charsToTrim));
        }

        if (tb_C_CodeFonction15.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction15.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction15", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction15", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction15.Text.Trim(charsToTrim));
        }

        if (tb_C_CodeFonction16.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction16.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction16", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction16", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction16.Text.Trim(charsToTrim));
        }

        if (tb_C_CodeFonction17.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction17.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction17", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
            cmd.Parameters.Add("@Code_fonction17", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction17.Text.Trim(charsToTrim));
        }

        if (tb_C_CodeFonction18.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction18.Text.Trim(charsToTrim) == "0")
        {
            cmd.Parameters.Add("@Code_fonction18", SqlDbType.Int).Value =
DBNull.Value;
        }
        else
        {
```

```

        cmd.Parameters.Add("@Code_fonction18", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction18.Text.Trim(charsToTrim));
    }

    if (tb_C_CodeFonction19.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction19.Text.Trim(charsToTrim) == "0")
    {
        cmd.Parameters.Add("@Code_fonction19", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction19", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction19.Text.Trim(charsToTrim));
    }

    if (tb_C_CodeFonction20.Text.Trim(charsToTrim) == "" ||
tb_C_CodeFonction20.Text.Trim(charsToTrim) == "0")
    {
        cmd.Parameters.Add("@Code_fonction20", SqlDbType.Int).Value =
DBNull.Value;
    }
    else
    {
        cmd.Parameters.Add("@Code_fonction20", SqlDbType.Int).Value =
Convert.ToInt32(tb_C_CodeFonction20.Text.Trim(charsToTrim));
    }
}
#endregion

con.Open();
cmd.ExecuteNonQuery();
con.Close();
fill_dgv_Licences();
tc_Licences.SelectedIndex = 0;

}
catch (Exception err)
{
    MessageBox.Show("Erreur update licence : " + err.Message);
}
}

private void Dgv_Licences_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
{
    int ID =
Convert.ToInt32(dgv_Licences.Rows[e.RowIndex].Cells[0].Value);
    tc_Licences.SelectedIndex = 1;
    tc_Licences.Height = 580;
    dgv_Licence_AllClients2.Visible = false;
    dgv_Licence_AllFonctions2.Visible = false;
    dgv_Licence_AllIntegrateurs2.Visible = false;

    lb_dgv_AllClients2.Visible = false;
    lb_dgv_AllFonctions2.Visible = false;
    lb_dgv_AllIntegrateurs2.Visible = false;

    tb_R_IDLicence.Text = ID.ToString();

```

```

foreach (Licence licence in leslicences)
{
    if (licence.ID == ID)
    {
        #region alimentation des tb_CodeFonction
        tb_R_CodeFonction1.Text = licence.Code_fonction1.ToString() +
        ". " + getFonctionName(licence.Code_fonction1);
        tb_R_CodeFonction2.Text = licence.Code_fonction2.ToString() +
        ". " + getFonctionName(licence.Code_fonction2);
        tb_R_CodeFonction3.Text = licence.Code_fonction3.ToString() +
        ". " + getFonctionName(licence.Code_fonction3);
        tb_R_CodeFonction4.Text = licence.Code_fonction4.ToString() +
        ". " + getFonctionName(licence.Code_fonction4);
        tb_R_CodeFonction5.Text = licence.Code_fonction5.ToString() +
        ". " + getFonctionName(licence.Code_fonction5);
        tb_R_CodeFonction6.Text = licence.Code_fonction6.ToString() +
        ". " + getFonctionName(licence.Code_fonction6);
        tb_R_CodeFonction7.Text = licence.Code_fonction7.ToString() +
        ". " + getFonctionName(licence.Code_fonction7);
        tb_R_CodeFonction8.Text = licence.Code_fonction8.ToString() +
        ". " + getFonctionName(licence.Code_fonction8);
        tb_R_CodeFonction9.Text = licence.Code_fonction9.ToString() +
        ". " + getFonctionName(licence.Code_fonction9);
        tb_R_CodeFonction10.Text = licence.Code_fonction10.ToString()
+ ". " + getFonctionName(licence.Code_fonction10);
        tb_R_CodeFonction11.Text = licence.Code_fonction11.ToString()
+ ". " + getFonctionName(licence.Code_fonction11);
        tb_R_CodeFonction12.Text = licence.Code_fonction12.ToString()
+ ". " + getFonctionName(licence.Code_fonction12);
        tb_R_CodeFonction13.Text = licence.Code_fonction13.ToString()
+ ". " + getFonctionName(licence.Code_fonction13);
        tb_R_CodeFonction14.Text = licence.Code_fonction14.ToString()
+ ". " + getFonctionName(licence.Code_fonction14);
        tb_R_CodeFonction15.Text = licence.Code_fonction15.ToString()
+ ". " + getFonctionName(licence.Code_fonction15);
        tb_R_CodeFonction16.Text = licence.Code_fonction16.ToString()
+ ". " + getFonctionName(licence.Code_fonction16);
        tb_R_CodeFonction17.Text = licence.Code_fonction17.ToString()
+ ". " + getFonctionName(licence.Code_fonction17);
        tb_R_CodeFonction18.Text = licence.Code_fonction18.ToString()
+ ". " + getFonctionName(licence.Code_fonction18);
        tb_R_CodeFonction19.Text = licence.Code_fonction19.ToString()
+ ". " + getFonctionName(licence.Code_fonction19);
        tb_R_CodeFonction20.Text = licence.Code_fonction20.ToString()
+ ". " + getFonctionName(licence.Code_fonction20);

        tb_C_CodeFonction1.Text = licence.Code_fonction1.ToString();
        tb_C_CodeFonction2.Text = licence.Code_fonction2.ToString();
        tb_C_CodeFonction3.Text = licence.Code_fonction3.ToString();
        tb_C_CodeFonction4.Text = licence.Code_fonction4.ToString();
        tb_C_CodeFonction5.Text = licence.Code_fonction5.ToString();
        tb_C_CodeFonction6.Text = licence.Code_fonction6.ToString();
        tb_C_CodeFonction7.Text = licence.Code_fonction7.ToString();
        tb_C_CodeFonction8.Text = licence.Code_fonction8.ToString();
        tb_C_CodeFonction9.Text = licence.Code_fonction9.ToString();
        tb_C_CodeFonction10.Text = licence.Code_fonction10.ToString();
        tb_C_CodeFonction11.Text = licence.Code_fonction11.ToString();
    }
}

```



```

        tb_C_CodeFonction12.Text = licence.Code_fonction12.ToString();
        tb_C_CodeFonction13.Text = licence.Code_fonction13.ToString();
        tb_C_CodeFonction14.Text = licence.Code_fonction14.ToString();
        tb_C_CodeFonction15.Text = licence.Code_fonction15.ToString();
        tb_C_CodeFonction16.Text = licence.Code_fonction16.ToString();
        tb_C_CodeFonction17.Text = licence.Code_fonction17.ToString();
        tb_C_CodeFonction18.Text = licence.Code_fonction18.ToString();
        tb_C_CodeFonction19.Text = licence.Code_fonction19.ToString();
        tb_C_CodeFonction20.Text = licence.Code_fonction20.ToString();
        #endregion

        dtp_R_DateExpirationLicence.Value =
Convert.ToDateTime(licence.Date_expiration.ToString("dd/MM/yyyy"));
        tb_R_Licence_DateCreation.Text =
licence.Date_creation.ToString("dd/MM/yyyy");
        tb_R_NbrEquipements.Text = licence.Nb_equipements.ToString();
        tb_R_NbrVariables.Text = licence.Nb_variables.ToString();
        tb_R_CodeIntegrateur.Text =
licence.Code_integrateur.ToString();
        tb_R_CodeClient.Text = licence.Code_client.ToString();
        tb_R_Checksum.Text = licence.Checksum.ToString();

        tb_C_Checksum.Text = licence.Checksum.ToString();
        tb_C_NbrEquipements.Text = licence.Nb_equipements.ToString();
        tb_C_NbrVariables.Text = licence.Nb_variables.ToString();
        tb_C_CodeIntegrateur.Text =
licence.Code_integrateur.ToString();
        tb_C_CodeClient.Text = licence.Code_client.ToString();
        dtp_U_DateExpirationLicence.Value =
Convert.ToDateTime(licence.Date_expiration.ToString("dd/MM/yyyy"));
    }
}

private void create_Integrateur(object sender, EventArgs e)
{
    try
    {
        SqlCommand cmd = new
SqlCommand("Base_NewLicences.dbo.create_Integrateur", con);
        cmd.CommandType = CommandType.StoredProcedure;

        cmd.Parameters.Add("@Nom", SqlDbType.VarChar).Value =
Convert.ToString(tb_C_NomIntegrateur.Text);
        cmd.Parameters.Add("@Date_update", SqlDbType.DateTime).Value =
Convert.ToDateTime(tb_C_DateMiseAJour_Integrateur.Text);
        cmd.Parameters.Add("@Description", SqlDbType.VarChar).Value =
Convert.ToString(tb_C_DescriptionIntegrateur.Text);

        con.Open();
        cmd.ExecuteNonQuery();
        con.Close();
        fill_dgv_Integrateurs();
    }
    catch (Exception err)

```

```

        {
            MessageBox.Show("Erreur create integrateur : " + err.Message);
        }
    }
    private void delete_Integrateur(object sender, EventArgs e)
    {
        int Code_integrateur = Convert.ToInt32(tb_R_CodeInteg.Text);
        SqlCommand cmd = new
        SqlCommand("Base_NewLicences.dbo.delete_Integrateur", con);
        cmd.CommandType = CommandType.StoredProcedure;

        cmd.Parameters.Add("@Code_integrateur", SqlDbType.Int).Value =
        Code_integrateur;

        con.Open();
        cmd.ExecuteNonQuery();
        con.Close();
        fill_dgv_Integrateurs();
        tc_Integrateurs.SelectedIndex = 0;
    }
    private void update_Integrateur(object sender, EventArgs e)
    {
        SqlCommand cmd = new
        SqlCommand("Base_NewLicences.dbo.update_Integrateur", con);
        cmd.CommandType = CommandType.StoredProcedure;

        try
        {
            int Code_integrateur = Convert.ToInt32(tb_R_CodeInteg.Text);
            String Nom = Convert.ToString(tb_U_NomInteg.Text);
            String Description = Convert.ToString(tb_U_DescriptionInteg.Text);
            DateTime Date_update =
            Convert.ToDateTime(tb_U_DateMiseAJourInteg.Text);

            cmd.Parameters.Add("@Code_integrateur", SqlDbType.Int).Value =
            Code_integrateur;
            cmd.Parameters.Add("@Nom", SqlDbType.VarChar).Value = Nom;
            cmd.Parameters.Add("@Description", SqlDbType.VarChar).Value =
            Description;
            cmd.Parameters.Add("@Date_update", SqlDbType.DateTime).Value =
            Date_update;

            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
            fill_dgv_Integrateurs();
            tc_Integrateurs.SelectedIndex = 0;
        }
        catch (Exception err)
        {
            MessageBox.Show("Erreur update integrateur : " + err.Message);
        }
    }
    private void Dgv_Integrateurs_CellDoubleClick(object sender,
    DataGridViewCellEventArgs e)
    {

```

```

        int Code_integrateur =
Convert.ToInt32(dgv_Integrateurs.Rows[e.RowIndex].Cells[0].Value);
        foreach (Integrateur integrateur in lesIntegrateurs)
        {
            if (integrateur.codeIntegrateur == Code_integrateur)
            {
                tb_R_CodeInteg.Text = integrateur.codeIntegrateur.ToString();
                tb_R_NomInteg.Text = integrateur.nom.ToString();
                tb_R_DescriptionInteg.Text =
integrateur.description.ToString();
                tb_R_DateMiseAJourInteg.Text =
integrateur.dateUpdate.ToString("dd/MM/yyyy");

                tb_U_NomInteg.Text = integrateur.nom.ToString();
                tb_U_DescriptionInteg.Text =
integrateur.description.ToString();
            }
        }
        tc_Integrateurs.SelectedIndex = 1;
    }

    private void create_Client(object sender, EventArgs e)
    {
        try
        {
            SqlCommand cmd = new
SqlCommand("Base_NewLicences.dbo.create_Client", con);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.Add("@Nom", SqlDbType.VarChar).Value =
Convert.ToString(tb_C_NomClient.Text);
            cmd.Parameters.Add("@Date_update", SqlDbType.DateTime).Value =
Convert.ToDateTime(tb_C_DateMiseAJourClient.Text);
            cmd.Parameters.Add("@Description", SqlDbType.VarChar).Value =
Convert.ToString(tb_C_DescriptionClient.Text);

            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
            fill_dgv_Clients();
        }
        catch (Exception err)
        {
            MessageBox.Show("Erreur create Client : " + err.Message);
        }
    }

    private void delete_Client(object sender, EventArgs e)
    {
        int Code_client = Convert.ToInt32(tb_R_CodeCli.Text);
        SqlCommand cmd = new SqlCommand("Base_NewLicences.dbo.delete_Client",
con);

        cmd.CommandType = CommandType.StoredProcedure;

        cmd.Parameters.Add("@Code_client", SqlDbType.Int).Value = Code_client;

```

```

        con.Open();
        cmd.ExecuteNonQuery();
        con.Close();
        fill_dgv_Clients();
        tc_Clients.SelectedIndex = 0;
    }
    private void update_Client(object sender, EventArgs e)
    {
        SqlCommand cmd = new SqlCommand("Base_NewLicences.dbo.update_Client",
con);
        cmd.CommandType = CommandType.StoredProcedure;

        try
        {
            int Code_client = Convert.ToInt32(tb_R_CodeCli.Text);
            String Nom = Convert.ToString(tb_U_NomClient.Text);
            String Description =
Convert.ToString(tb_U_DescriptionClient.Text);
            DateTime Date_update =
Convert.ToDateTime(tb_U_DateMiseAJourClient.Text);

            cmd.Parameters.Add("@Code_Client", SqlDbType.Int).Value =
Code_client;
            cmd.Parameters.Add("@Nom", SqlDbType.VarChar).Value = Nom;
            cmd.Parameters.Add("@Description", SqlDbType.VarChar).Value =
Description;
            cmd.Parameters.Add("@Date_update", SqlDbType.DateTime).Value =
Date_update;

            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
            fill_dgv_Clients();
            tc_Clients.SelectedIndex = 0;
        }
        catch (Exception err)
        {
            MessageBox.Show("Erreur update intgrateur : " + err.Message);
        }
    }
    private void Dgv_Clients_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        int Code_client =
Convert.ToInt32(dgv_Clients.Rows[e.RowIndex].Cells[0].Value);
        foreach (Client client in lesClients)
        {
            if (client.codeClient == Code_client)
            {
                tb_R_CodeCli.Text = client.codeClient.ToString();
                tb_R_NomClient.Text = client.nom.ToString();
                tb_R_Description_Client.Text = client.description.ToString();
                tb_R_DateMiseAJourClient.Text =
client.dateUpdate.ToString("dd/MM/yyyy");

                tb_U_NomClient.Text = client.nom.ToString();
                tb_U_DescriptionClient.Text = client.description.ToString();
            }
        }
    }

```

```

    }
}
tc_Clients.SelectedIndex = 1;
}

private void create_Fonction(object sender, EventArgs e)
{
    try
    {
        SqlCommand cmd = new
SqlCommand("Base_NewLicences.dbo.create_Fonction", con);
        cmd.CommandType = CommandType.StoredProcedure;

        cmd.Parameters.Add("@Nom", SqlDbType.VarChar).Value =
Convert.ToString(tb_C_NomFonction.Text);
        cmd.Parameters.Add("@Date_update", SqlDbType.DateTime).Value =
Convert.ToDateTime(tb_C_DateMiseAJourFonction.Text);
        cmd.Parameters.Add("@Description", SqlDbType.VarChar).Value =
Convert.ToString(tb_C_DescriptionFonction.Text);

        con.Open();
        cmd.ExecuteNonQuery();
        con.Close();
        fill_dgv_Fonctions();
    }
    catch (Exception err)
    {
        MessageBox.Show("Erreur create Fonction : " + err.Message);
    }
}

private void delete_Fonction(object sender, EventArgs e)
{
    int Code_fonction = Convert.ToInt32(tb_R_CodeFonction.Text);
    SqlCommand cmd = new
SqlCommand("Base_NewLicences.dbo.delete_Fonction", con);
    cmd.CommandType = CommandType.StoredProcedure;

    cmd.Parameters.Add("@Code_Fonction", SqlDbType.Int).Value =
Code_fonction;

    con.Open();
    cmd.ExecuteNonQuery();
    con.Close();
    fill_dgv_Fonctions();
    tc_Fonctions.SelectedIndex = 0;
}

private void update_Fonction(object sender, EventArgs e)
{
    SqlCommand cmd = new
SqlCommand("Base_NewLicences.dbo.update_Fonction", con);
    cmd.CommandType = CommandType.StoredProcedure;

    try
    {
        int Code_fonction = Convert.ToInt32(tb_R_CodeFonction.Text);

```

```

        String Nom = Convert.ToString(tb_U_NomFonction.Text);
        String Description =
Convert.ToString(tb_U_DescriptionFonction.Text);
        DateTime Date_update =
Convert.ToDateTime(tb_U_DateMiseAJourFonction.Text);

        cmd.Parameters.Add("@Code_fonction", SqlDbType.Int).Value =
Code_fonction;
        cmd.Parameters.Add("@Nom", SqlDbType.VarChar).Value = Nom;
        cmd.Parameters.Add("@Description", SqlDbType.VarChar).Value =
Description;
        cmd.Parameters.Add("@Date_update", SqlDbType.DateTime).Value =
Date_update;

        con.Open();
        cmd.ExecuteNonQuery();
        con.Close();
        fill_dgv_Fonctions();
        tc_Fonctions.SelectedIndex = 0;
    }
    catch (Exception err)
    {
        MessageBox.Show("Erreur update Fonction : " + err.Message);
    }
}

private void Dgv_Fonctions_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
{
    int Code_fonction =
Convert.ToInt32(dgv_Fonctions.Rows[e.RowIndex].Cells[0].Value);
    foreach (Fonction fonction in lesFonctions)
    {
        if (fonction.codeFonction == Code_fonction)
        {
            tb_R_CodeFonction.Text = fonction.codeFonction.ToString();
            tb_R_NomFonction.Text = fonction.nom.ToString();
            tb_R_DescriptionFonction.Text =
fonction.description.ToString();
            tb_R_DateMiseAJourFonction.Text =
fonction.dateUpdate.ToString("dd/MM/yyyy");

            tb_U_NomFonction.Text = fonction.nom.ToString();
            tb_U_DescriptionFonction.Text =
fonction.description.ToString();
        }
    }
    tc_Fonctions.SelectedIndex = 1;
}

//Utilisation interactif des datagridview
private void Dgv_Licence_AllFonctions_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
{
    int Code_fonction =
Convert.ToInt32(dgv_Licence_AllFonctions.Rows[e.RowIndex].Cells[0].Value.ToString(
));

```

```
if (tb_C_CodeFonction1.Text == "0")
{
    tb_C_CodeFonction1.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction2.Text == "0")
{
    tb_C_CodeFonction2.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction3.Text == "0")
{
    tb_C_CodeFonction3.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction4.Text == "0")
{
    tb_C_CodeFonction4.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction5.Text == "0")
{
    tb_C_CodeFonction5.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction6.Text == "0")
{
    tb_C_CodeFonction6.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction7.Text == "0")
{
    tb_C_CodeFonction7.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction8.Text == "0")
{
    tb_C_CodeFonction8.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction9.Text == "0")
{
    tb_C_CodeFonction9.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction10.Text == "0")
{
    tb_C_CodeFonction10.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction11.Text == "0")
{
    tb_C_CodeFonction11.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction12.Text == "0")
{
    tb_C_CodeFonction12.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction13.Text == "0")
{
    tb_C_CodeFonction13.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction14.Text == "0")
{
    tb_C_CodeFonction14.Text = Code_fonction.ToString();
}
else if (tb_C_CodeFonction15.Text == "0")
```

```

        {
            tb_C_CodeFonction15.Text = Code_fonction.ToString();
        }
        else if (tb_C_CodeFonction16.Text == "0")
        {
            tb_C_CodeFonction16.Text = Code_fonction.ToString();
        }
        else if (tb_C_CodeFonction17.Text == "0")
        {
            tb_C_CodeFonction17.Text = Code_fonction.ToString();
        }
        else if (tb_C_CodeFonction18.Text == "0")
        {
            tb_C_CodeFonction18.Text = Code_fonction.ToString();
        }
        else if (tb_C_CodeFonction19.Text == "0")
        {
            tb_C_CodeFonction19.Text = Code_fonction.ToString();
        }
        else if (tb_C_CodeFonction20.Text == "0")
        {
            tb_C_CodeFonction20.Text = Code_fonction.ToString();
        }
    }
    private void Dgv_Licence_AllIntegrateurs_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        tb_C_CodeIntegrateur.Text =
dgv_Licence_AllIntegrateurs.Rows[e.RowIndex].Cells[0].Value.ToString();
    }
    private void Dgv_Licence_AllClients_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        tb_C_CodeClient.Text =
dgv_Licence_AllClients.Rows[e.RowIndex].Cells[0].Value.ToString();
    }

    private void Dgv_Licence_AllIntegrateurs2_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        tb_CodeIntegrateur.Text =
dgv_Licence_AllIntegrateurs2.Rows[e.RowIndex].Cells[0].Value.ToString();
    }
    private void Dgv_Licence_AllClients2_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        tb_CodeClient.Text =
dgv_Licence_AllClients2.Rows[e.RowIndex].Cells[0].Value.ToString();
    }
    private void Dgv_Licence_AllFonctions2_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
    {
        int Code_fonction =
Convert.ToInt32(dgv_Licence_AllFonctions2.Rows[e.RowIndex].Cells[0].Value.ToString
());
        if (tb_CodeFonction1.Text == "")
        {

```



```
        tb_CodeFonction1.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction2.Text == "")
    {
        tb_CodeFonction2.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction3.Text == "")
    {
        tb_CodeFonction3.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction4.Text == "")
    {
        tb_CodeFonction4.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction5.Text == "")
    {
        tb_CodeFonction5.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction6.Text == "")
    {
        tb_CodeFonction6.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction7.Text == "")
    {
        tb_CodeFonction7.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction8.Text == "")
    {
        tb_CodeFonction8.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction9.Text == "")
    {
        tb_CodeFonction9.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction10.Text == "")
    {
        tb_CodeFonction10.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction11.Text == "")
    {
        tb_CodeFonction11.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction12.Text == "")
    {
        tb_CodeFonction12.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction13.Text == "")
    {
        tb_CodeFonction13.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction14.Text == "")
    {
        tb_CodeFonction14.Text = Code_fonction.ToString();
    }
    else if (tb_CodeFonction15.Text == "")
    {
        tb_CodeFonction15.Text = Code_fonction.ToString();
    }
```

```

}
else if (tb_CodeFonction16.Text == "")
{
    tb_CodeFonction16.Text = Code_fonction.ToString();
}
else if (tb_CodeFonction17.Text == "")
{
    tb_CodeFonction17.Text = Code_fonction.ToString();
}
else if (tb_CodeFonction18.Text == "")
{
    tb_CodeFonction18.Text = Code_fonction.ToString();
}
else if (tb_CodeFonction19.Text == "")
{
    tb_CodeFonction19.Text = Code_fonction.ToString();
}
else if (tb_CodeFonction20.Text == "")
{
    tb_CodeFonction20.Text = Code_fonction.ToString();
}
}

```

//Fonctionnalités supplémentaires

private void ResetUpdateLicence(object sender, EventArgs e)

```

{
    int ID = Convert.ToInt32(tb_R_IDLicence.Text);
    foreach (Licence licence in lesLicences)
    {
        if (licence.ID == ID)
        {
            #region alimentation des tb_CodeFonction
            tb_C_CodeFonction1.Text = licence.Code_fonction1.ToString();
            tb_C_CodeFonction2.Text = licence.Code_fonction2.ToString();
            tb_C_CodeFonction3.Text = licence.Code_fonction3.ToString();
            tb_C_CodeFonction4.Text = licence.Code_fonction4.ToString();
            tb_C_CodeFonction5.Text = licence.Code_fonction5.ToString();
            tb_C_CodeFonction6.Text = licence.Code_fonction6.ToString();
            tb_C_CodeFonction7.Text = licence.Code_fonction7.ToString();
            tb_C_CodeFonction8.Text = licence.Code_fonction8.ToString();
            tb_C_CodeFonction9.Text = licence.Code_fonction9.ToString();
            tb_C_CodeFonction10.Text = licence.Code_fonction10.ToString();
            tb_C_CodeFonction11.Text = licence.Code_fonction11.ToString();
            tb_C_CodeFonction12.Text = licence.Code_fonction12.ToString();
            tb_C_CodeFonction13.Text = licence.Code_fonction13.ToString();
            tb_C_CodeFonction14.Text = licence.Code_fonction14.ToString();
            tb_C_CodeFonction15.Text = licence.Code_fonction15.ToString();
            tb_C_CodeFonction16.Text = licence.Code_fonction16.ToString();
            tb_C_CodeFonction17.Text = licence.Code_fonction17.ToString();
            tb_C_CodeFonction18.Text = licence.Code_fonction18.ToString();
            tb_C_CodeFonction19.Text = licence.Code_fonction19.ToString();
            tb_C_CodeFonction20.Text = licence.Code_fonction20.ToString();
            #endregion
            tb_C_Checksum.Text = licence.Checksum.ToString();
            tb_C_NbrEquipements.Text = licence.Nb_equipements.ToString();
            tb_C_NbrVariables.Text = licence.Nb_variables.ToString();
        }
    }
}

```

```

        tb_C_CodeIntegrateur.Text =
licence.Code_integrateur.ToString();
        tb_C_CodeClient.Text = licence.Code_client.ToString();
        dtp_U_DateExpirationLicence.Value =
Convert.ToDateTime(licence.Date_expiration.ToString("dd/MM/yyyy"));
    }
}
}
private void ResetUpdateLicenceTo0(object sender, EventArgs e)
{
    tb_C_CodeFonction1.Text = "0";
    tb_C_CodeFonction2.Text = "0";
    tb_C_CodeFonction3.Text = "0";
    tb_C_CodeFonction4.Text = "0";
    tb_C_CodeFonction5.Text = "0";
    tb_C_CodeFonction6.Text = "0";
    tb_C_CodeFonction7.Text = "0";
    tb_C_CodeFonction8.Text = "0";
    tb_C_CodeFonction9.Text = "0";
    tb_C_CodeFonction10.Text = "0";
    tb_C_CodeFonction11.Text = "0";
    tb_C_CodeFonction12.Text = "0";
    tb_C_CodeFonction13.Text = "0";
    tb_C_CodeFonction14.Text = "0";
    tb_C_CodeFonction15.Text = "0";
    tb_C_CodeFonction16.Text = "0";
    tb_C_CodeFonction17.Text = "0";
    tb_C_CodeFonction18.Text = "0";
    tb_C_CodeFonction19.Text = "0";
    tb_C_CodeFonction20.Text = "0";
    tb_C_CodeClient.Text = "0";
    tb_C_CodeIntegrateur.Text = "0";
    tb_C_NbrEquipements.Text = "0";
    tb_C_NbrVariables.Text = "0";
    tb_C_Checksum.Text = "0";
    dtp_U_DateExpirationLicence.Value = DateTime.Now;
}
private void ResetCreateLicence(object sender, EventArgs e)
{
    tb_NbEquipements.Text = "";
    tb_NbVariables.Text = "";
    tb_CodeIntegrateur.Text = "";
    tb_CodeClient.Text = "";

    tb_CodeFonction1.Text = "";
    tb_CodeFonction2.Text = "";
    tb_CodeFonction3.Text = "";
    tb_CodeFonction4.Text = "";
    tb_CodeFonction5.Text = "";
    tb_CodeFonction6.Text = "";
    tb_CodeFonction7.Text = "";
    tb_CodeFonction8.Text = "";
    tb_CodeFonction9.Text = "";
    tb_CodeFonction10.Text = "";
    tb_CodeFonction11.Text = "";
    tb_CodeFonction12.Text = "";
    tb_CodeFonction13.Text = "";

```

```

        tb_CodeFonction14.Text = "";
        tb_CodeFonction15.Text = "";
        tb_CodeFonction16.Text = "";
        tb_CodeFonction17.Text = "";
        tb_CodeFonction18.Text = "";
        tb_CodeFonction19.Text = "";
        tb_CodeFonction20.Text = "";

        dtp_C_DateExpirationLicence.Value = DateTime.Now;
    }
    private void Btn_ResetFonctionsTo0_Click(object sender, EventArgs e)
    {
        tb_CodeFonction1.Text = "";
        tb_CodeFonction2.Text = "";
        tb_CodeFonction3.Text = "";
        tb_CodeFonction4.Text = "";
        tb_CodeFonction5.Text = "";
        tb_CodeFonction6.Text = "";
        tb_CodeFonction7.Text = "";
        tb_CodeFonction8.Text = "";
        tb_CodeFonction9.Text = "";
        tb_CodeFonction10.Text = "";
        tb_CodeFonction11.Text = "";
        tb_CodeFonction12.Text = "";
        tb_CodeFonction13.Text = "";
        tb_CodeFonction14.Text = "";
        tb_CodeFonction15.Text = "";
        tb_CodeFonction16.Text = "";
        tb_CodeFonction17.Text = "";
        tb_CodeFonction18.Text = "";
        tb_CodeFonction19.Text = "";
        tb_CodeFonction20.Text = "";
    }

    private string getFonctionName(int CodeFonction)
    {
        foreach(Fonction fonction in lesFonctions)
        {
            if (fonction.codeFonction == CodeFonction) {
                return fonction.nom;
            }
        }
        return "NULL";
    }
}

```