|  |  |
| --- | --- |
| **Compte rendu STAGE**  ***Année 2023-2024*** | **SEMAINE 3**  ***16/01/2023 – 20/01/2023*** |

***Sommaire / Summary***

Table des matières

[1. Présentation de l’entreprise. 3](#_Toc124512946)

[2. Mes missions. 4](#_Toc124512947)

[1.1 Simulateur\_CEOG. 4](#_Toc124512948)

[1.1.1 Documents et maquette. **Erreur ! Signet non défini.**](#_Toc124512949)

[1.1.2 Modifications graphiques. **Erreur ! Signet non défini.**](#_Toc124512950)

[1.1.3 Modifications code **Erreur ! Signet non défini.**](#_Toc124512951)

[3. Outils utilisés 19](#_Toc124512952)

[4. Conclusion. 20](#_Toc124512953)

[5. Niko-Niko. 21](#_Toc124512954)

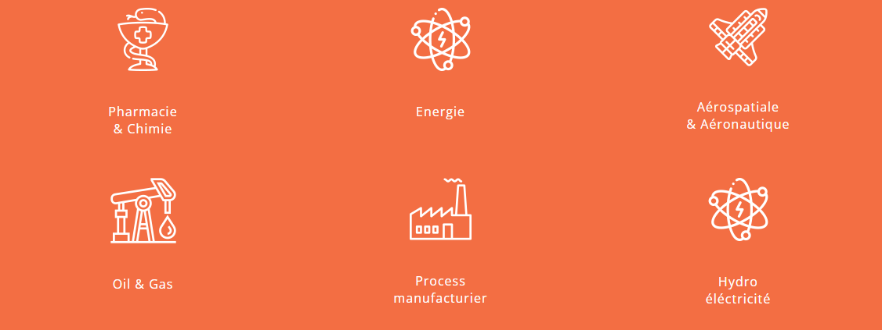
# Présentation de l’entreprise.



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

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

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



# Mes missions.

## Simulateur\_CEOG.

### Connexions Modbus RTU.

En continuité de la semaine dernière, après avoir fait fonctionner la connexion et la communication grâce au Protocol Modbus TCP, il m’a été demandé de faire la même chose pour les communications Modbus RTU.

Pour cela, l’utilisation de la bibliothèque EasyModbus permet de créer ces connexions et communications de la même manière que celle pour Modbus TCP ce qui a rendu la tâche beaucoup plus simple.

Le seul problème, c’est qu’à défaut de matériel, il est impossible de tester la connexion ainsi que la communication car les connexions RTU se font par câbles (Master/Slave).

Malgré ce problème, tout devrait fonctionné car le fonctionnement reste le même que pour la communication TCP

### Création de la classe.

La classe reste essentiellement la même sauf au niveau de la configuration de la liaison dans laquelle il faut renseigner les informations supplémentaires suivantes :

* UnitIdentifier
* SerialPort
* Baudrate
* Bits de parité
* Bits de stop

Ces informations sont renseignées dans le fichier de configuration sous ce format :

SerialPort|Baudrate|BitsDeParité|BitsDeStop|UnitIdentifier|Pooling|Port

Une fois ceci en tête, la classe reste la même que celle pour TCP avec quelques changements :

public class RTU\_Class

{

public string Nom;

public string Machine;

public string Protocol;

public string Configuration\_Liaison;

public string Mode;

public string Fichier\_Echange;

public byte ID;

public string SerialPort;

public int Pooling;

public int Baudrate;

public System.IO.Ports.Parity Parity;

public System.IO.Ports.StopBits Stopbits;

public int Port;

public ModbusClient Modbus\_Client;

public ModbusServer Modbus\_Server;

public RTU\_Form Equipement\_Form;

public string ConnexionStatus = "Disconnected";

public RTU\_Class(string Nom, string Machine, string Protocol, string Configuration\_Liaison, string Mode, string Fichier\_Echange)

{

this.Nom = Nom;

this.Machine = Machine;

this.Protocol = Protocol;

this.Configuration\_Liaison = Configuration\_Liaison;

this.Mode = Mode;

this.Fichier\_Echange = AppDomain.CurrentDomain.BaseDirectory + Fichier\_Echange;

this.ID = Convert.ToByte(Configuration\_Liaison.Split('|')[4]);

this.SerialPort = Configuration\_Liaison.Split('|')[0];

this.Pooling = Convert.ToInt32(Configuration\_Liaison.Split('|')[5]);

this.Baudrate = Convert.ToInt32(Configuration\_Liaison.Split('|')[1]);

this.Port = Convert.ToInt32(Configuration\_Liaison.Split('|')[6]);

string parity = Configuration\_Liaison.Split('|')[2];

switch (parity)

{

case "0":

this.Parity = System.IO.Ports.Parity.None;

break;

case "1":

this.Parity = System.IO.Ports.Parity.Odd;

break;

case "2":

this.Parity = System.IO.Ports.Parity.Even;

break;

case "3":

this.Parity = System.IO.Ports.Parity.Mark;

break;

case "4":

this.Parity = System.IO.Ports.Parity.Space;

break;

}

string stopbits = Configuration\_Liaison.Split('|')[3];

switch (stopbits)

{

case "0":

this.Stopbits = System.IO.Ports.StopBits.None;

break;

case "1":

this.Stopbits = System.IO.Ports.StopBits.One;

break;

case "2":

this.Stopbits = System.IO.Ports.StopBits.Two;

break;

case "3":

this.Stopbits = System.IO.Ports.StopBits.OnePointFive;

break;

}

}

}

Ensuite, la connexion, la communication et le fonctionnement de l’application reste la même de TCP à RTU donc sur ce plan la aucun changement n’est nécessaire.

### Protocol et bibliothèque Profinet.

Et pour finir, le dernier type de communication demander est Profinet.

PROFINET est un standard de communication ouvert pour l'automatisation industrielle. Il a été créé par PI (PROFIBUS & PROFINET International) - l'organisation des utilisateurs PROFIBUS qui compte plus de 1200 membres - et développé par Siemens, Phoenix Contact, Molex et d'autres constructeurs. De par son ouverture et l'utilisation d'un média de communication standard (Ethernet), PROFINET permet l’utilisation de toutes marques de matériel. La première version de ce standard a été publiée en août 2001. La version courante est la version V2.2. PROFINET est normalisé CEI 61158 et CEI 61784. (Source : <https://fr.wikipedia.org/wiki/Profinet>)

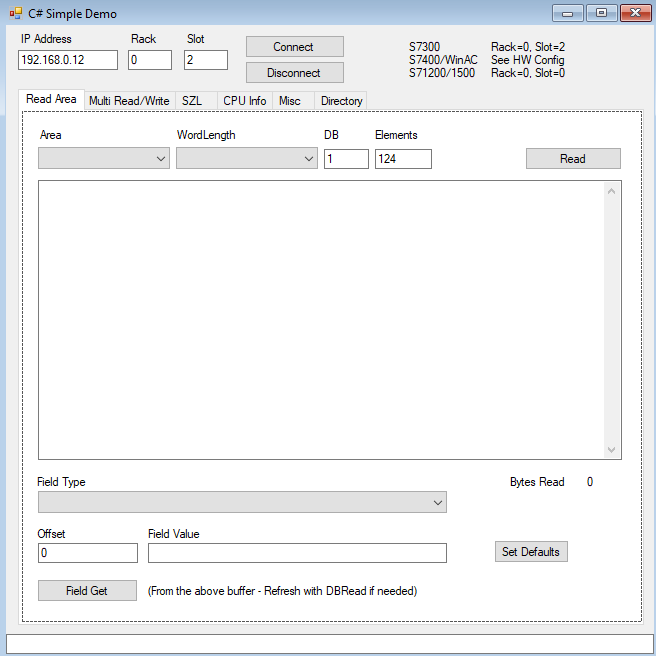
Pour ceci, l’implémentation d’une nouvelle bibliothèque est nécessaire.  
Après avoir fait des recherches, je suis tombé sur Sharp7, une adaptation de Snap7, une bibliothèque de communication par protocole S7Protocol (Protocole de communication développé par Siemens pour les automates), Pour C# (<https://snap7.sourceforge.net/sharp7.html>)

Pour C#, l’adaptation a été faite sous forme d’une classe Sharp7.cs contenant toutes les fonctions plutôt qu’une bibliothèque.

Avec ceci est fournis un fichier de documentation sur le fonctionnement ainsi que des exemples d’application fonctionnant avec la classe

### Implémentation Sharp7.

J’ai donc intégré la classe (6000+ lignes) et l’interface qui se trouve dans l’exemple dans mon application :

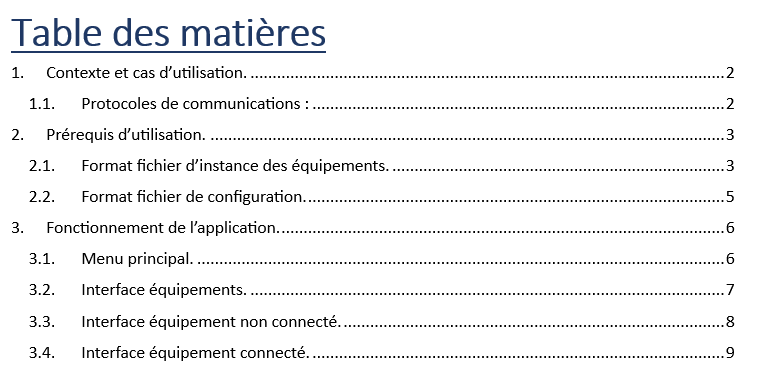


Seulement, a cause d’un manque de communication des attentes de l’utilisateur concernant les communications Profinet, il en a donc été dit plus judicieux de ne pas continuer le développement de cette partie avant plus d’informations.

### Documentation projet.

J’ai part la suite rédigé un fichier de documentation pour les utilisateurs de l’applications.  
Il m’a été demande d’expliquer le but de l’application, comment la faire marcher, décrire les possibles erreurs et solutions ainsi que les prérequis a son bon fonctionnement.

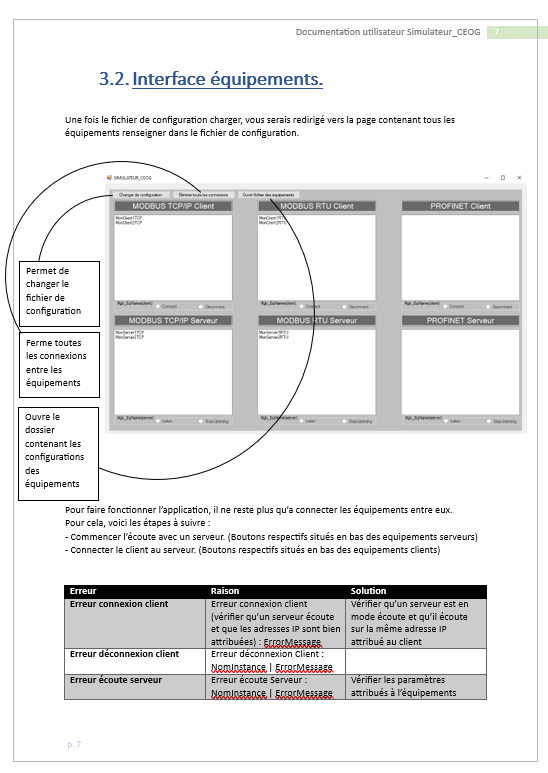
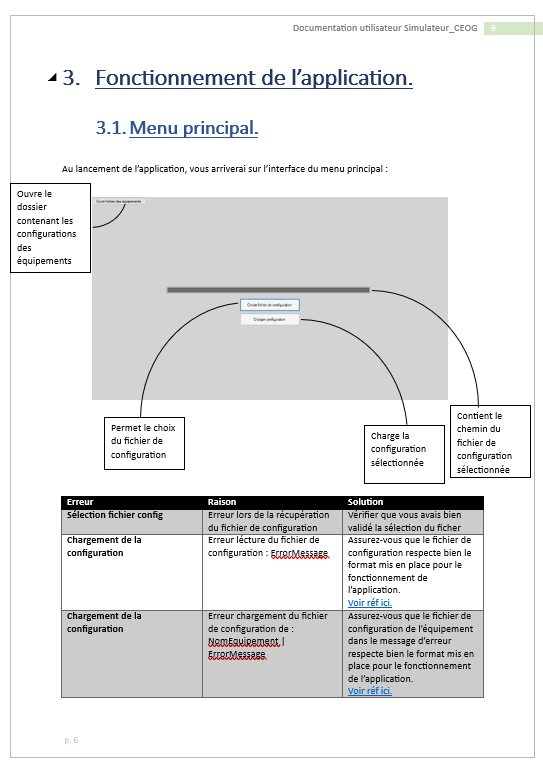
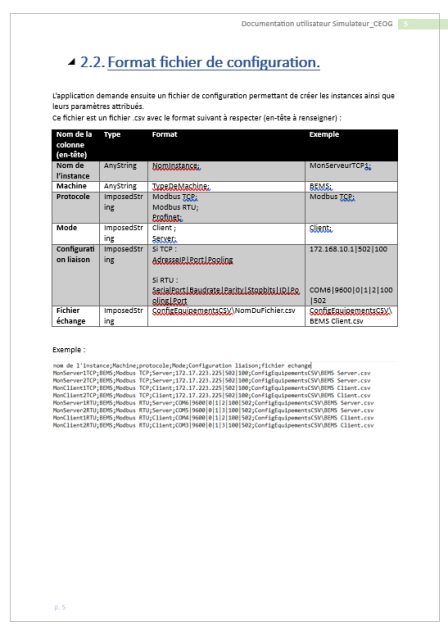
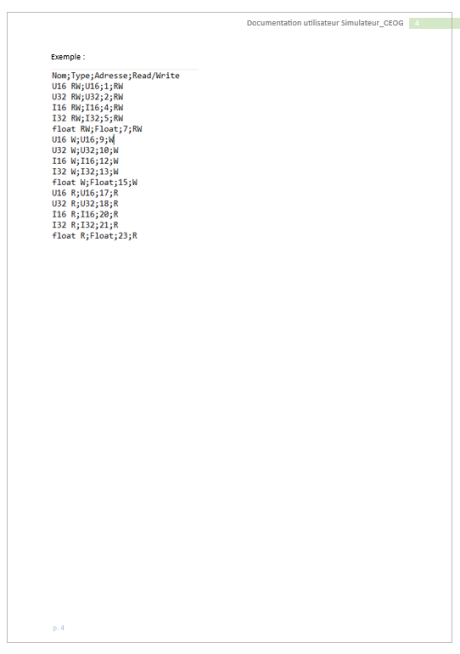
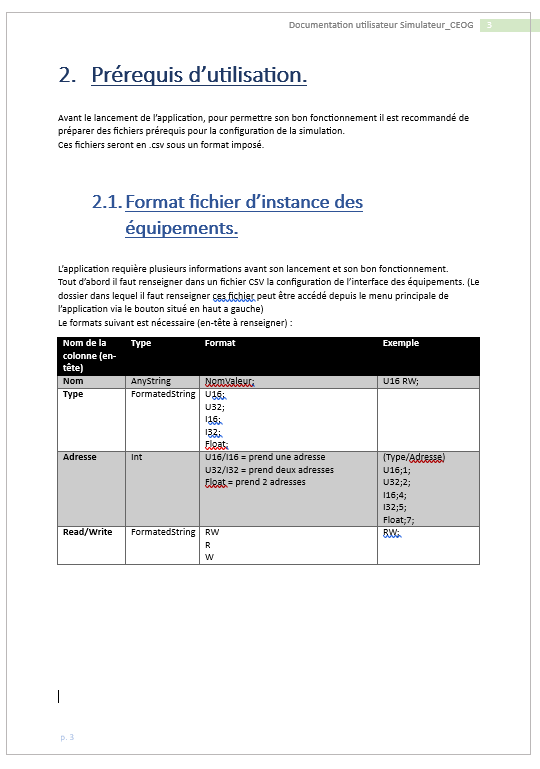
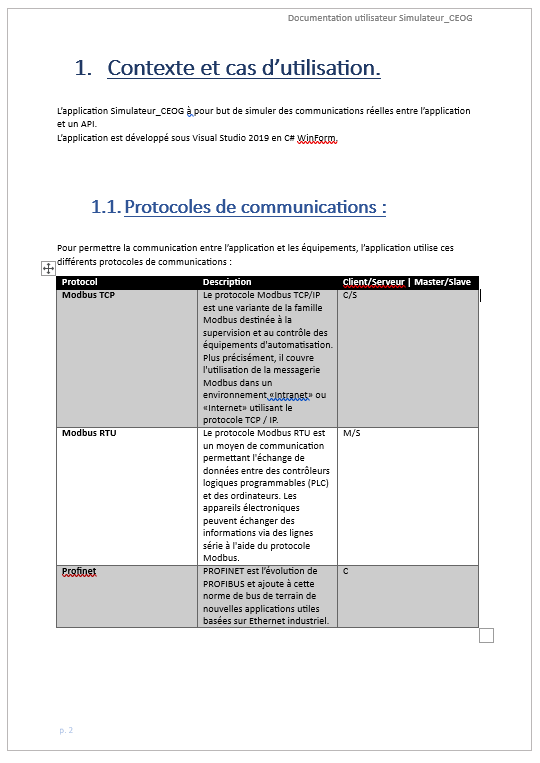
Voici le sommaire du document :

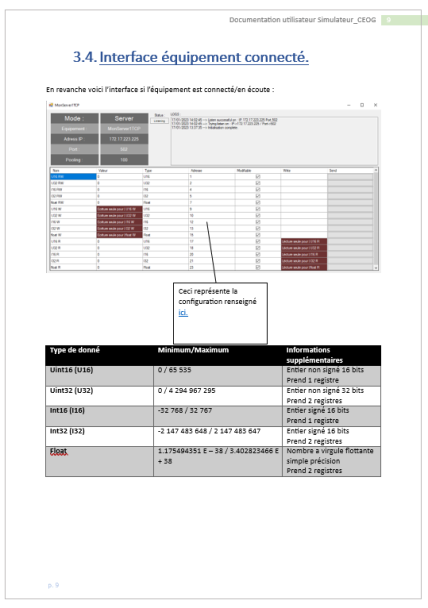
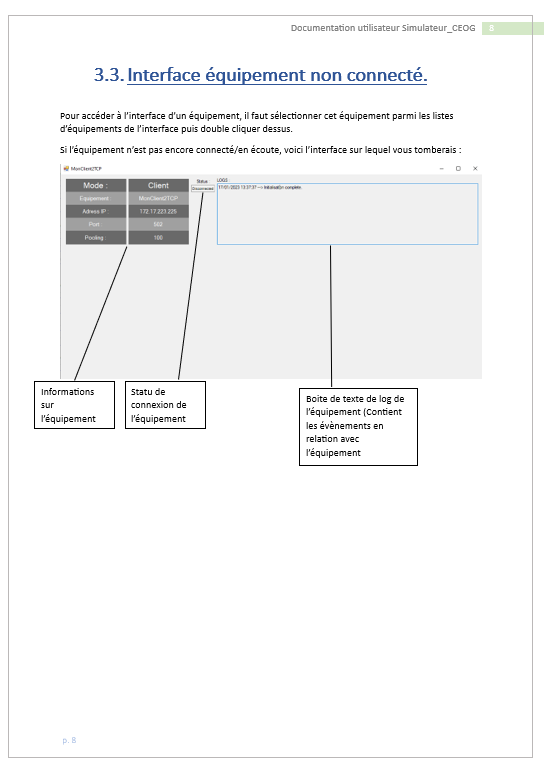


Le document explique :

* Comment remplir les fichiers des configurations des équipements CSV
* Comment remplir le fichier de configuration CSV
* Le fonctionnement des boutons et leurs fonctions
* Les erreurs possibles et leurs solutions
* Fonctionnement des communications et des protocoles

Voici comment se présente le document :





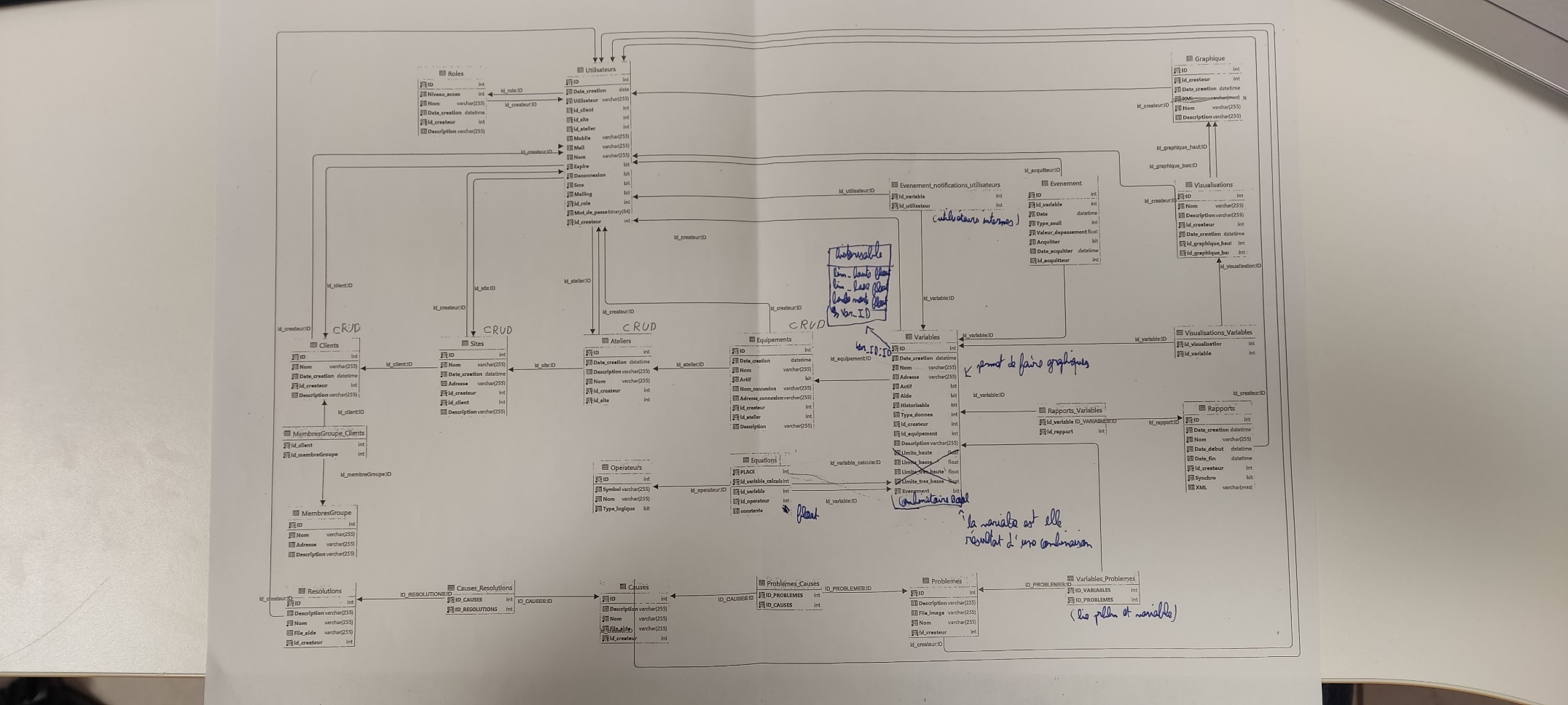
## Mise à jour de la BDD de I-CONNECT.

J’ai été affecter par la suite après le projet précèdent, au développement de l’application principale I-CONNECT.

Cette application est développée sous AVEVA et exploite une base de données sur SQL Server.

### Diagramme de la base

Voici le diagramme représentant la base de données :



Ce diagramme représente toutes les données exploitées par l’application.  
Son but est de pouvoir faire du ‘Monitoring’ sur les utilisateurs de l’application.

Il m’a été demander d’apport plusieurs modifications ainsi que de rajouter des tables et des procédures stocker selon le fonctionnement et la norme d’écriture de table et procédure imposé par mon tuteur.

### Normes de nommages et créations.

Pour créer les procédures stocker, il fallait suivre la norme de nommage suivante :

Dbo.ICONE\_sp\_action\_table

Exemples :

|  |  |  |
| --- | --- | --- |
| Action | Table | Résultat |
| GET | Test | Dbo.ICONE\_sp\_get\_test |
| UPDATE | Variable | Dbo.ICONE\_sp\_variable |
| DELETE | MrKoro | Dbo.ICONE\_sp\_MrKoro |

Ensuite, pour la norme de création des procédures stocker, il faut impérativement renseigner en haut de la procédure des informations de cette manière :

-- =============================================

-- Author : Prénom NOM

-- Description : description de la procédure

-- Donnée d'entrée :

-- @Parametre1,

-- @Parametre2,

-- @...

-- Code erreur :

-- 1 => Raison si la procédure retourne 1

-- 2 => Raison si la procédure retourne 2

-- 100 => Raison si la procédure retourne 100

-- ... => ...

-- =============================================

Cette norme permet à n’importe qui de reprendre la procédure stocker et la comprendre dans le plus court délai.

### Table historisable.

Il m’a été par la suite demander de créer une table ‘’Historisable’’ sous cette forme-là :

|  |  |  |  |
| --- | --- | --- | --- |
| Variable\_ID(clé étrangère) | Limite\_haute | Limite\_base | Bande\_Morte |
| Voir table variable | Float (NOT NULL) | Float (NOT NULL) | Float (NOT NULL) |

Puis sur cette table, j’ai dû faire un CREATE, GET et UPDATE (DELETE se fait en cascade sur variables)

Voici le CREATE :

USE [BASE\_ICONECT]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ICONE\_sp\_create\_historisable] Script Date: 18/01/2023 16:21:18 \*\*\*\*\*\*/

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 = historisable 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

-- 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

Ensuite le GET :

USE [BASE\_ICONECT]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ICONE\_sp\_get\_historisable] Script Date: 18/01/2023 16:22:33 \*\*\*\*\*\*/

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

Et pour finir pour la table historisable, le update :

USE [BASE\_ICONECT]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ICONE\_sp\_update\_historisable] Script Date: 18/01/2023 16:23:23 \*\*\*\*\*\*/

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

# Outils utilisés.

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

# Conclusion.

Cette semaine j’ai pu ‘conclure’ le projet Simulateur\_CEOG tout en apprenant une nouvelle bibliothèque ainsi que la fonction de plusieurs types de Protocol et équipements, j’ai pu mettre mes compétences et ma réflexion a l’épreuve et produire un développement efficace qui m’a rendu fier.

J’ai aussi pu approfondir mes compétences en SQL tout en utilisant MISCROSOFR SQL Server, sachant que SQL et les base de données ne sont pas mes points fort, c’est rafraichissant de sortir un peu de C# et approfondir une de mes lacunes.

J’ai aussi l’impression de mieux m’y prendre niveau organisation et préparation avant le développement pur et dur, a force de m’être planté plusieurs fois par manque de plan de travail, j’y ai appris à utiliser les ressources disponibles pour faire une plan et avoir des réflexions plus clair sur les taches et le projet en général.  
Ceci est particulièrement utile en entreprise car la plupart des projets sont partagé entre plusieurs personne et sont d’une taille bien plus grandes que les projets faits en cours.

# Niko-Niko.

|  |  |  |
| --- | --- | --- |
| JOUR | RESSENTI | TACHE |
| Lundi | Pin on Awesome Smiley | Mise en place des communications RTU |
| Mardi | Smiley face on computer : Virtual Learning Network | Découverte protocole Profinet + Sharp7 et implémentation. Documentation utilisateur de l’application |
| Mercredi | Premium Vector | Sad emoji looking to the side vector yellow face big eyes  raised eyebrows and sad mouthEmojiData.ai | AI AutoComplete for Emojis, Words and Hashtags | A  HelloWoofy.com Project - Emojidata.ai | Etude diagramme base de données I-CONNECT + début modifications de la DB |
| Jeudi | EmojiData.ai | AI AutoComplete for Emojis, Words and Hashtags | A  HelloWoofy.com Project - Emojidata.ai | Création de procédures stockées |
| Vendredi | EmojiData.ai | AI AutoComplete for Emojis, Words and Hashtags | A  HelloWoofy.com Project - Emojidata.ai | Création de procédures stockées (Tuteur malade) |