



# RAPPORT PERSONNEL DE PROJET DEBRAS NICOLAS

18.01.2021

---

<b>Rappel de ma partie</b>	<b>2</b>
<b>Le Réseau LoRaWAN</b>	<b>3</b>
Appairage de la passerelle LoRa et des capteurs véhicules	3
Appairage de la Passerelle LoRa	3
Appairage des capteurs véhicules	5
Configuration du serveur/capteur LoRaWAN pour renvoyer les données.	6
Configuration des capteur LoRaWAN	6
Configuration du serveur.	7
Développement du serveur d'application	8
<b>Application administrateur</b>	<b>9</b>
Utilisation d'un lecteur de badge RFID	10
Modbus TCP	11
Développement	11
Création d'un utilisateur :	12
La suppression/suspension d'un utilisateur	13
La réservation d'un parking	15
Connexion à la base de donnée	15
Développement	15
Test Unitaire	17
<b>Conclusion</b>	<b>18</b>

## I. Rappel de ma partie

Ma partie est de réaliser dans un premier temps l'installation et la configuration d'un réseau LoRA ainsi que le développement du serveur d'application.

Ce qui comprend :

- La configuration de la passerelle LoRa et du serveur LoRaWAN
- L'Appairage des capteurs véhicules et l'intégration dans le serveur
- La configuration du serveur/capteur LoRaWAN pour renvoyer les données.
- Développement du serveur d'application ce qui comprend le traitement des données reçu au format JSON et la mise en base de données

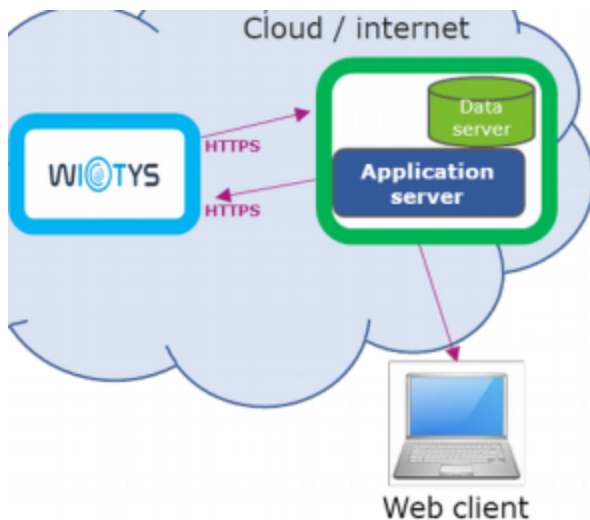
Dans un second temps, je dois me charger du développement d'une application pour gérer l'interface d'administration des usagers du parking en C#.

Ce qui comprend :

- L'attribution d'un badge
- La création d'utilisateur
- La suppression d'un utilisateur
- La suspension d'un utilisateur
- La réservation d'un parking

## II. Le Réseau LoRaWAN

Le système LoRaWAN que nous allons utiliser est produit par la société Wi6labs. Les capteurs sous les voitures sont des capteurs TBS-220. Les capteurs TBS-220 communiquent à l'aide du protocole Lora avec la passerelle. Wi6labs fournit le serveur Lora. Pour récupérer les infos, nous avons utilisé un serveur d'application qui demande les informations au serveur LoRa. Les informations récupérées seront ensuite envoyées à la BDD.



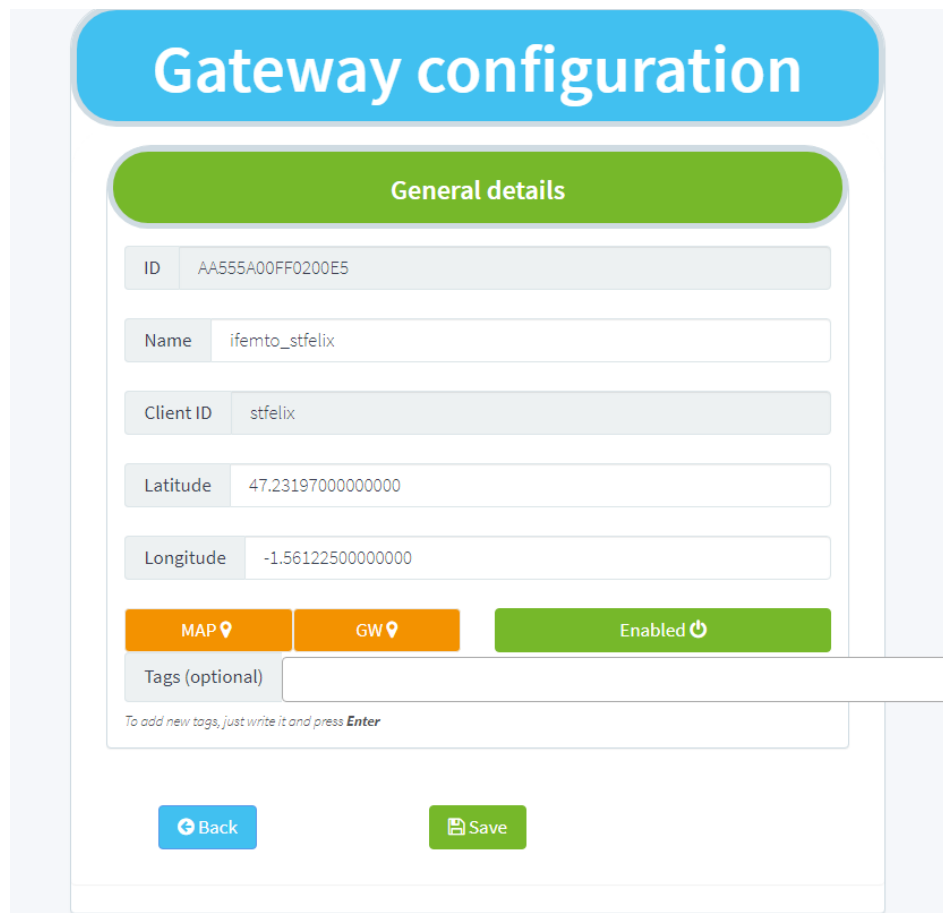
Un réseau LoRaWAN est constitué d'équipements sans-fils basse consommation qui communiquent avec des serveurs applicatifs au travers de passerelles. Le Lora est un protocole de télécommunication fonctionnant par radio (868 MHZ en Europe, 915MHZ en Amérique du Nord) basé sur la technique de modulation par étalement de spectre (voir Glossaire) de type CSS (Chirp Spread Spectrum), l'intérêt de ce fonctionnement, est de permettre une communication sûr de longue distance (10km+) et à bas coût en énergie. Le désavantage de ce moyen de communication est le débit qui n'est pas très élevé (moins de 50kbps) et son cycle de communication, qui est plus adapté à l'usage d'application qui permettent un délai dans les communications

### A. Appairage de la passerelle LoRa et des capteurs véhicules

#### a. Appairage de la Passerelle LoRa

L'appairage de la passerelle LoRa se fait via le serveur Lora fournit par l'entreprise Wi6labs. Pour faire ça, il faut rentrer les ID et le nom de la passerelle ainsi que le ID clients dans le menu prévu à cet effet. Ce menu se trouve via le site web de Wi6Labs qui permet la configuration du serveur LoRa.

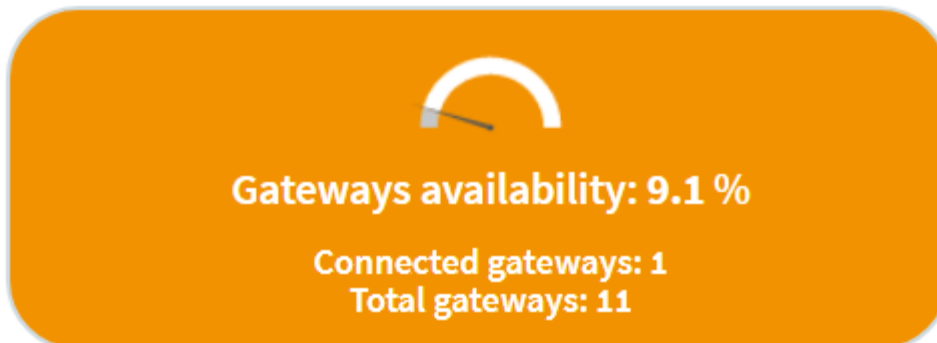
Dans mon cas :



The image shows a 'Gateway configuration' form with a blue header. Below the header is a green 'General details' section. It contains several input fields: 'ID' (AA555A00FF0200E5), 'Name' (ifemto\_stfelix), 'Client ID' (stfelix), 'Latitude' (47.23197000000000), and 'Longitude' (-1.56122500000000). There are three buttons: 'MAP' with a location pin icon, 'GW' with a location pin icon, and 'Enabled' with a power icon. Below these is a 'Tags (optional)' field with a placeholder text: 'To add new tags, just write it and press **Enter**'. At the bottom are 'Back' and 'Save' buttons.

General details	
ID	AA555A00FF0200E5
Name	ifemto_stfelix
Client ID	stfelix
Latitude	47.23197000000000
Longitude	-1.56122500000000
<div><span>MAP</span> <span>GW</span> <span>Enabled</span></div>	
Tags (optional) <small>To add new tags, just write it and press <b>Enter</b></small>	
<div><span>Back</span> <span>Save</span></div>	

Après la configuration, le statut (connecté ou pas) s'affiche sur la page principal du site web dédié à la configuration. Comme sur l'image en dessous.



## b. Appairage des capteurs véhicules

L'appairage des capteurs véhicules se fait quasiment de la même manière que la passerelle. On rentre les champs : clientID, Data Server list, ainsi que l'id du capteur.

Dans mon cas (l'un des deux capteur) :




**General details**

**Name**

**ClientID**

**Data Server list**

**Microsoft Azure IoT - Device ID (optional)** ⓘ  
☒ **Sensor Id (equal devEUI)**

	Tags	Name 	DevAddr	DevEUI	Enable	Actions
<input type="checkbox"/>		test_LoRa 2	00032356	70b3d53af0032356		   
<input type="checkbox"/>		test_LoRa 1	00032355	70b3d53af0032355		   
<div>First Previous 1 Next Last</div>						10 ▾

## B. Configuration du serveur/capteur LoRaWAN pour renvoyer les données.

### a. Configuration des capteur LoRaWAN

Pour modifier la configuration des capteurs, il faut envoyer une trame contenant certaines informations depuis le serveur LoRa vers les capteurs. Modifier la configuration des capteurs permet de régler plusieurs paramètres. Dans notre cas, j'avais besoin de modifier le temps entre chaque message d'un capteur vers le serveur LoRa.

Un message correspond à une trame où est stocké le statut du capteur.

La trame qui permet cette modification se construit via la documentation du capteur.

J'ai donc envoyé la trame correcte (vérifiée par notre professeur tuteur) au capteur malheureusement cela n'a eu aucun effet .

Après plusieurs recherches sur internet (demande de la documentation officiel au constructeur, prise de contact avec les revendeurs) qui n'ont pas été concluante. Avec notre professeur tuteur, on a donc conclu que le problème était matériel.

Un autre élément le confirme, avec la configuration dite "usine", le capteur doit envoyer un message au serveur toutes les 10 min. Après réinitialisation des capteurs et appairage des capteurs au serveur, les capteurs n'en voyaient pas de message.

## b. Configuration du serveur.

La configuration est quasiment pareille que pour la passerelle, ou un capteur. Il suffit de rentrer le type, le nom et l'id du serveur.




L'URL et le port sont donnée est définie automatiquement via la sélection du type.

**Parameters**

Type	HTTP
Name	test_lora
ServerID	server1
URL	http://82.231.146.148
Port	11300
Model	Default

☐ Send radio parameters

**AppEUI list**

70B3D53AF0032335	
70B3D53AF0032336	
	



## C. Développement du serveur d'application

La récupération des informations se fera de manière entièrement automatisée par un programme développé en JavaScript et exécuter par le logiciel Node

Pour pouvoir recevoir des informations via des requêtes http nous avons besoin d'un serveur en permanence à l'écoute.

La solution la plus efficace et facile à mettre en place est un simple serveur qui écoutera sur un port défini à l'avance et qui récupérera toutes les informations reçues avant d'en extraire les données pertinentes.

La création d'un serveur ce se fera grâce au framework pour node Express le résultat est le suivant :

```
var express = require('express'); // Web Framework
var app = express();
//var sql = require('mssql');

var sqlConfig = {
  user: 'TestLora',
  password: 'Password1',
  server: 'localhost',
  database: 'testlora'
};

app.use(express.json());

app.post('/', function(request, response){
  console.log(request.body);
  console.log("-----");

  /* var Id = request.body.DevAddr
  var Stat = request.body.data.status;
  console.log("Recu");
  if (request.body.data.frameType == 'status'){
    console.log(Id + ' ' + Stat);
  } */
  response.send("received");
});

app.listen(11300, '0.0.0.0' || 'localhost' || '10.16.2.219');
```

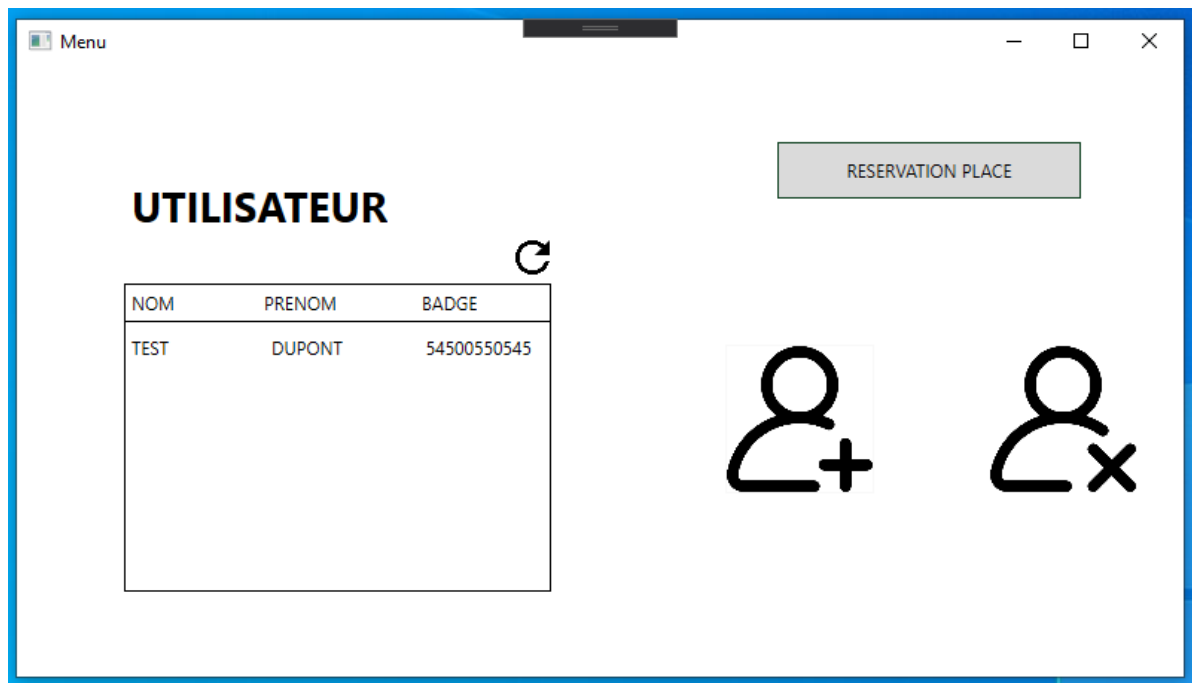
Du au problème rencontré ( détaillé plus haut dans la rapport), j'ai malheureusement pas pu continuer. En effet, comme le capteur n'envoie pas de trame. Je n'ai pas pu traiter la partie liée à la mise en base de données des informations reçues.

La partie mise en commentaire doit permettre d'afficher les trames liées au statut du capteur.

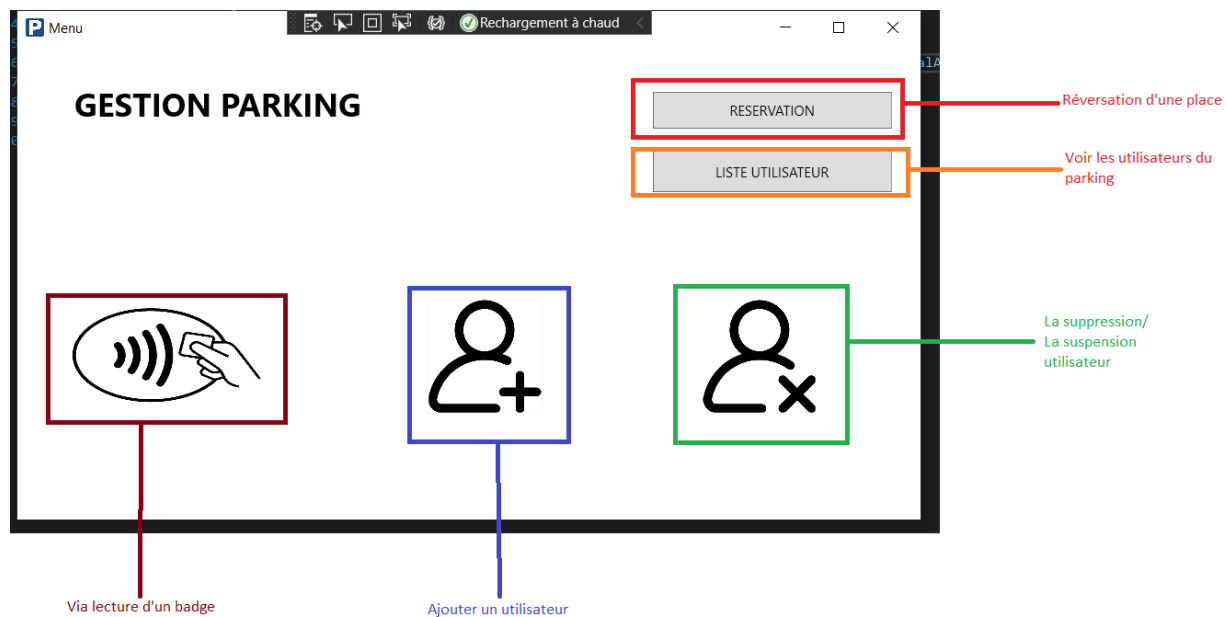
### III. Application administrateur

Pour accéder à chaque fonctionnalité, j'ai créé une interface simple d'utilisation. Le but de ce menu est qu'il doit être le plus intuitif possible.

Lors de la phase d'analyse, j'avais réalisé ce prototype :



Afin de rendre l'interface la plus simple possible, j'ai amélioré mon prototype. Chaque bouton correspond à une fonctionnalité du cahier des charges :



Exemple du code C# du bouton pour ajouter un utilisateur :

```

}
1 référence
private void add_Click(object sender, RoutedEventArgs e)
{
    Add_user add_user = new Add_user(connexion_sql);
    add_user.ShowDialog();
}
    
```

## A. Utilisation d'un lecteur de badge RFID

Afin de faciliter l'utilisation de l'application, j'ai décidé de rajouter la possibilité d'utiliser le lecteur Badge RFID XGCS850C201. Ce lecteur est le même que celui d'Alexandre. Nous avons à notre disponibilité un seul lecteur. La configuration de ce lecteur a été faite par Alexandre (voir détail dans son rapport).

L'utilisation du lecteur RFID est un plus. En effet, l'application marche complètement sans branchement et connexion du lecteur.



### a. Modbus TCP

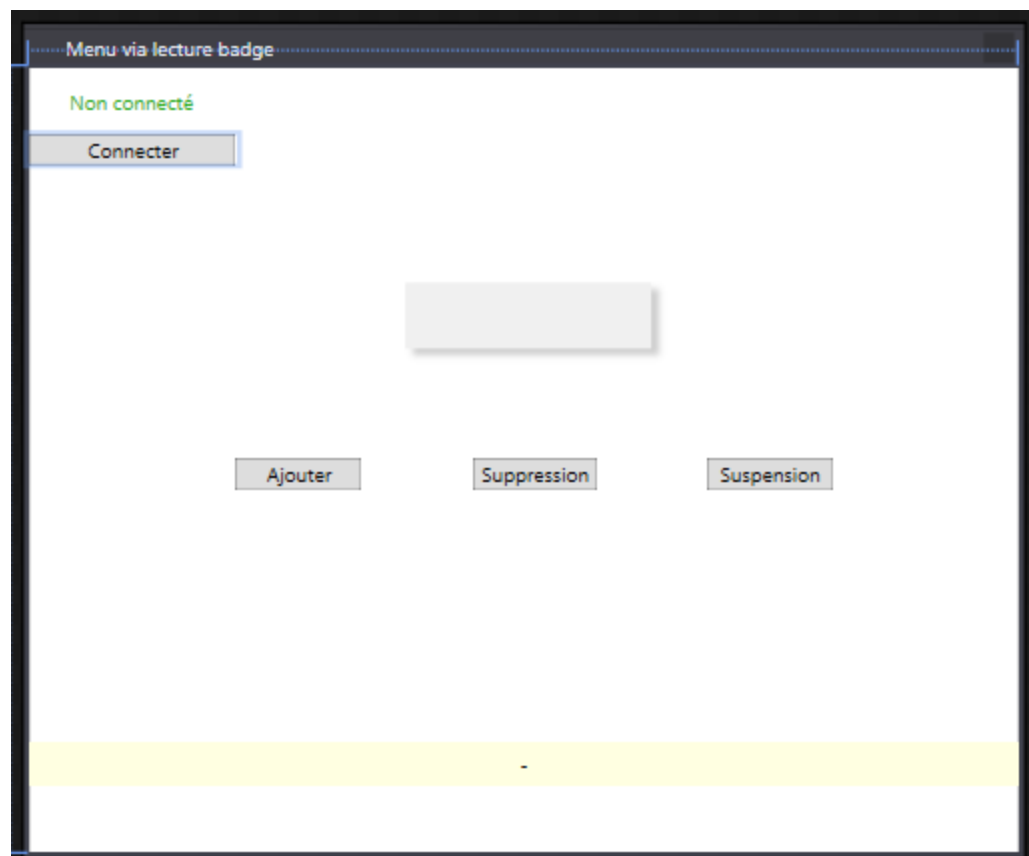
Le lecteur Badge RFID XGCS850C201 utilise comme protocole le modbus TCP afin de transmettre les informations du badge scanné. Il fonctionne sur le mode client-serveur. Seuls les clients sont actifs, le serveur est complètement passif. Ce sont les clients qui doivent lire et écrire dans le serveur Modbus (dans notre cas, le lecteur de Badge). Ce protocole utilise des trames contenant la fonction à traiter et la donnée.

Après avoir compris le fonctionnement du lecteur de badge, j'ai pu commencer à développer cette partie.

### b. Développement

Pour utiliser le protocole modbus TCP, j'ai dû utiliser une librairie particulière qui prend en charge le modbus. J'ai choisi de rester dans une interface la plus simple et la plus facile à utiliser.

Après le branchement du lecteur de badge, il suffit juste d'appuyer sur le bouton "Connecter" afin d'établir la connexion. Si une erreur se produit, une fenêtre apparaîtra alors.



Cette partie du code permet d'établir la connexion avec le capteur :

2 références

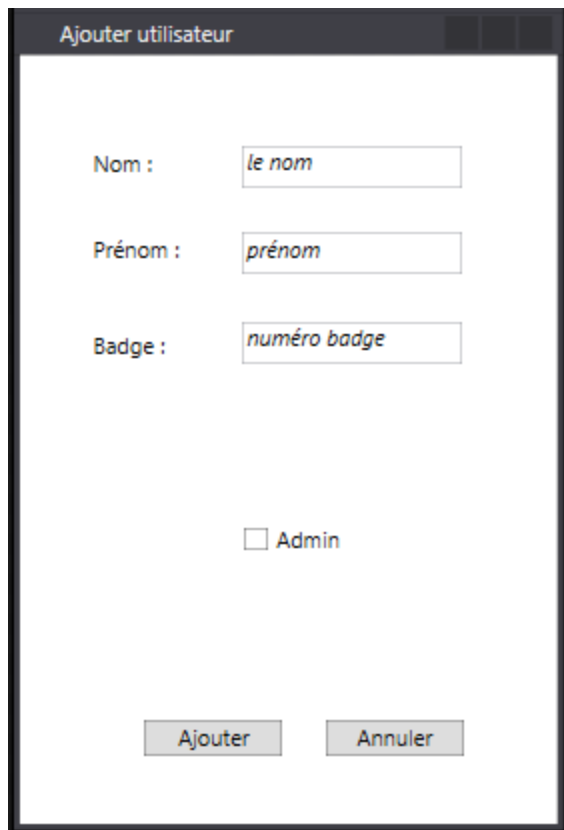
```
private bool connecter()
{
    try
    {
        // Create new modbus master and add event functions
        MBmaster = new Master(IP, 502);
        MBmaster.OnResponseData += new ModbusTCP.Master.ResponseData(MBmaster_OnResponseData);
        MBmaster.OnException += new ModbusTCP.Master.ExceptionData(MBmaster_OnException);
    }
    catch (SystemException error)
    {
        MessageBox.Show(error.Message);
    }

    return MBmaster.connected;
}
```

On crée un client (ici appelé Master) qui prend pour IP l'adresse du capteur ainsi que le port nécessaire à la connexion. Le client va demander la lecture d'un badge au capteur. L'information contenue dans le badge va être renvoyée à l'application.

## B. Création d'un utilisateur :

J'ai réalisé une interface qui permet de créer un utilisateur du parking. L'utilisateur du logiciel doit rentrer le nom, prénom ainsi que le badge. Pour le rôle de l'utilisateur du parking, il se choisit via un checkBox. Il est de base utilisateur "standard", mais si on coche la case, il obtient le rôle "administrateur".



Ajouter utilisateur

Nom :

Prénom :

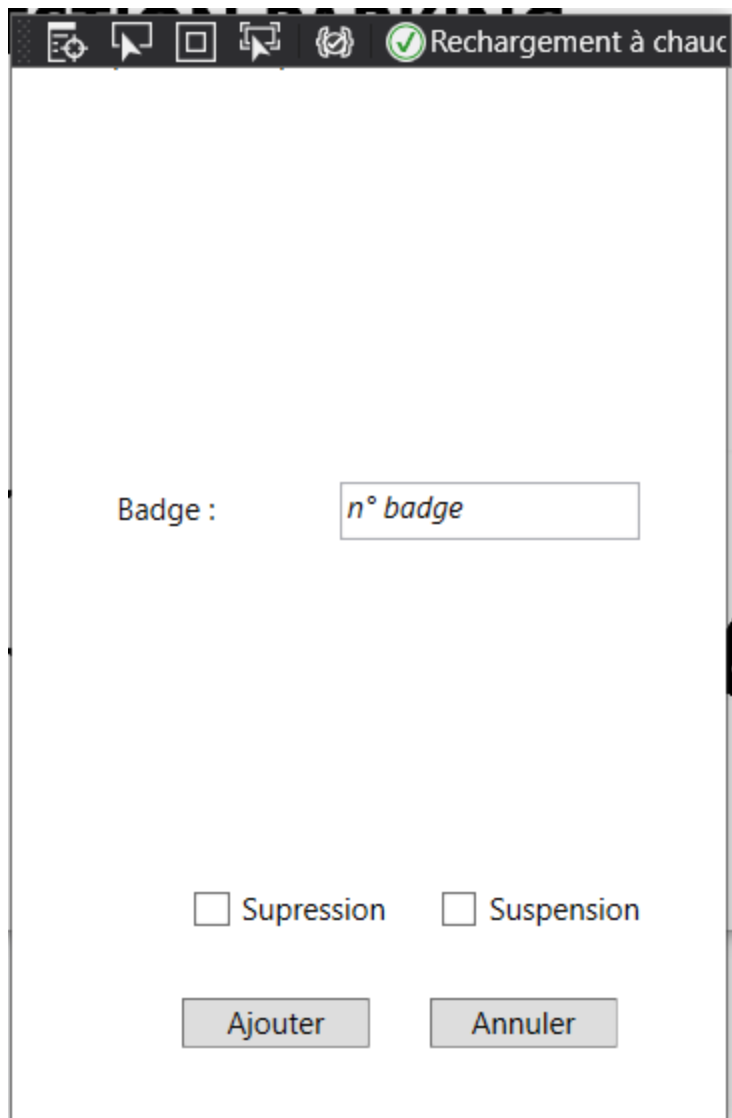
Badge :

☐ Admin

The image shows a web form titled "Ajouter utilisateur". It contains three text input fields: "Nom" with placeholder text "le nom", "Prénom" with placeholder text "prénom", and "Badge" with placeholder text "numéro badge". Below these fields is a checkbox labeled "Admin". At the bottom of the form are two buttons: "Ajouter" and "Annuler".

### C. La suppression/suspension d'un utilisateur

J'ai décidé de regrouper la suppression et la suspension dans le même menu. L'un supprime l'utilisateur de la base de données. L'autre change l'état de l'utilisateur. Le choix de supprimer ou suspendre un utilisateur se fait via un checkBox. L'utilisateur de l'interface a juste à rentrer le numéro du badge de l'utilisateur à suspendre ou supprimer.

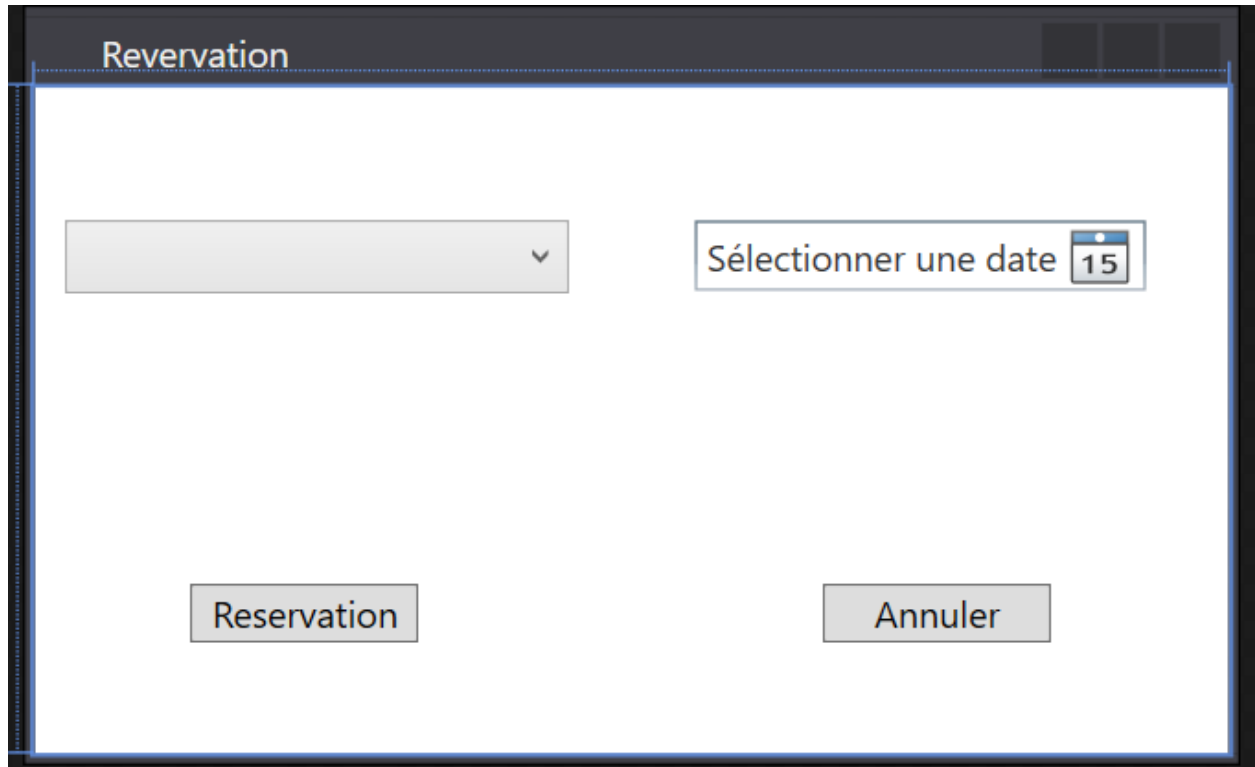


The image shows a screenshot of a software application window. The title bar at the top contains several icons (a gear, a magnifying glass, a square, a double arrow, a refresh icon, and a green checkmark) followed by the text "Rechargement à chauc". The main area of the window is white and contains the following elements:

- A label "Badge :" followed by a text input field containing the text "n° badge".
- Two checkboxes with labels: ☐ Supression and ☐ Suspension.
- Two buttons at the bottom: "Ajouter" and "Annuler".

## D. La réservation d'un parking

Pour faire une réservation, l'utilisateur de l'interface doit sélectionner un utilisateur via un ComboBox (liste de tous les utilisateurs dans la base de donnée) et une date.



The screenshot shows a window titled "Revervation" (note the typo). Inside the window, there is a light gray rectangular area containing two input fields. On the left is a ComboBox with a downward arrow icon. On the right is a date selection field with the text "Sélectionner une date" and a small calendar icon showing the number "15". Below these fields, there are two buttons: "Reservation" on the left and "Annuler" on the right.

## E. Connexion à la base de donnée

### a. Développement

L'application est connectée à la base de données du projet. Chaque fonctionnalité est liée à la base de données. J'ai opté pour faire une classe ou on retrouve chaque requête à la base de données sous forme d'une fonction différente.



Fonction Connexion :

```
1 référence
public Connexion()
{
    String connexionString = "SERVER=127.0.0.1; DATABASE=parking; UID=root; PASSWORD=";

    try
    {
        this.connexion = new MySqlConnection(connexionString);
        this.connexion.Open();
        list_badge = new List<string>();
        list_user = new List<string>();
        list_id = new List<string>();
    } catch (Exception e) {
        Console.WriteLine("Error: " + e);
        Console.WriteLine(e.StackTrace);
    }
}
```

Cette fonction permet de se connecter à la base de données.

Fonction add\_user :

```
public void add_user(string nom, string prenom, string badge, string role)
{
    try
    {
        string query = "INSERT INTO `utilisateur` (nom, prenom, badge, role, etat) VALUES (@nom, @prenom, @badge, @role, @etat)";

        MySqlCommand cmd = new MySqlCommand(query, connexion);

        cmd.Parameters.Add("@nom", MySqlDbType.String).Value = nom;
        cmd.Parameters.Add("@prenom", MySqlDbType.String).Value = prenom;
        cmd.Parameters.Add("@badge", MySqlDbType.String).Value = badge;
        cmd.Parameters.Add("@role", MySqlDbType.String).Value = role;
        cmd.Parameters.Add("@etat", MySqlDbType.VarChar).Value = 1;

        int test = cmd.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine("Error: " + e);
        Console.WriteLine(e.StackTrace);
    }
}
```

Cette fonction permet d'ajouter un utilisateur à la base de données. Cette fonction prend en paramètre le nom, prénom, badge et le rôle. Ces éléments sont nécessaires afin de remplir correctement la base de données.

Fonction delete\_user :

```
1 référence  
public void delete_user(string badge)  
{  
    try  
    {  
        string query = "DELETE FROM `utilisateur` WHERE `utilisateur`.`badge` = "+badge+"";  
        MySqlCommand cmd = new MySqlCommand(query, connexion);  
        int test = cmd.ExecuteNonQuery();  
    }  
    catch(Exception e)  
    {  
        System.Windows.Forms.MessageBox.Show("MUNERO DE BADGE PAS CORRECT");  
    }  
}
```

Cette fonction a pour but de supprimer un utilisateur en fonction de son badge. Cette fonction prend en paramètre le badge.

#### b. Test Unitaire

Cette classe est particulièrement sensible, j'ai donc décidé de réaliser des tests unitaires en particulier sur cette partie de mon projet.

J'ai malheureusement a ce jour pas eu le temps de finir ces tests unitaires.

## IV. Conclusion

J'ai beaucoup apprécié travailler sur ce projet. J'ai beaucoup appris et consolidé mes compétences. Ce projet était très intéressant car il permet de voir un panel de technologie très vaste. Par exemple, j'ai découvert l'existence et l'utilisation d'une raspberry industrielle.

Ce projet a aussi développé chez moi, des nouvelles compétences de gestion de projet. En effet, un travail de groupe comme celui-ci nécessite de bien s'organiser et de tout mettre en place pour que chaque personne soit dans les meilleures conditions de travail.