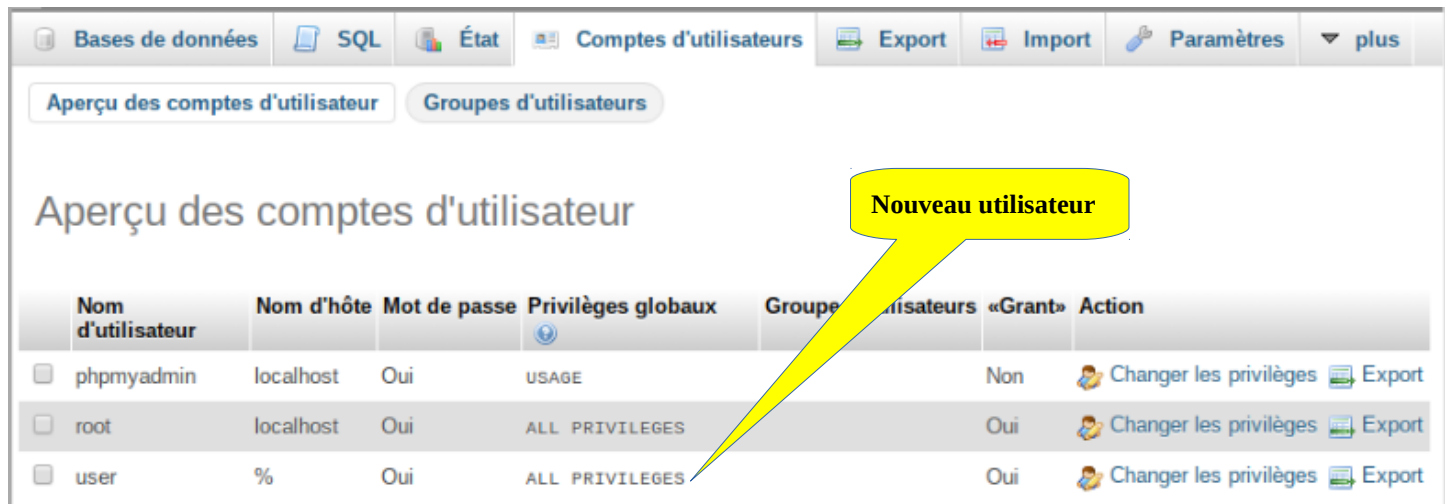


Création d'une application en C# pour administrer une base de donnée (installée sur Raspberry Pi)

La base de donnée MySQL étant installée sur une RaspberryPi et que l'application qui va être créée en C# sera utilisée pour l'administrer à distance sur un ordinateur sous Windows, il faut faire quelques modifications .

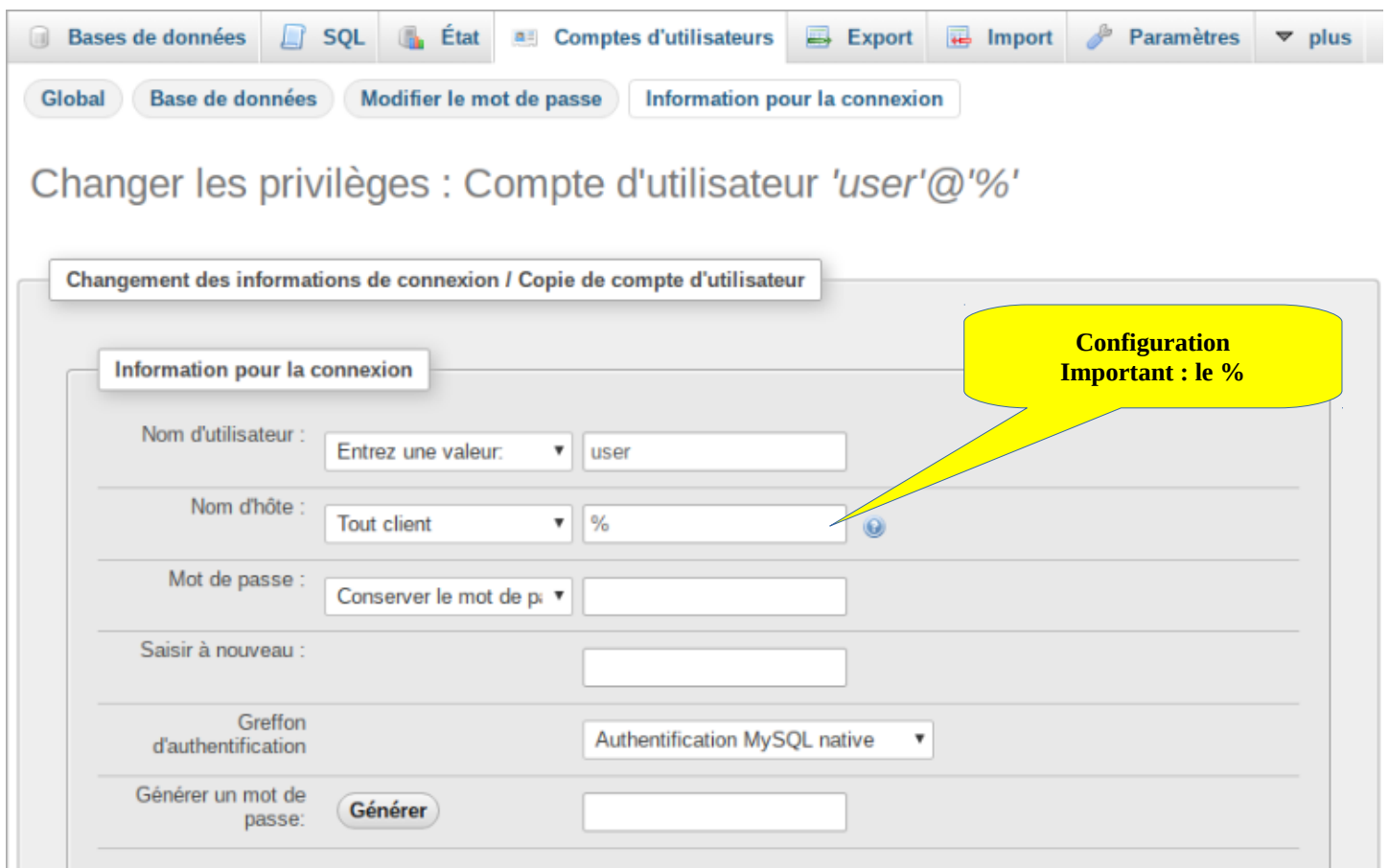
1 - Sur la Raspberry :

On va se connecter à **phpMyAdmin** afin de pouvoir créer un nouvel utilisateur destiné à gérer une base de donnée à distance.



The screenshot shows the 'Aperçu des comptes d'utilisateur' (User Accounts Overview) page in phpMyAdmin. The table lists three users: 'phpmyadmin', 'root', and 'user'. The 'user' entry is highlighted with a yellow callout bubble labeled 'Nouveau utilisateur'.

Nom d'utilisateur	Nom d'hôte	Mot de passe	Privileges globaux	Groupe d'utilisateurs	«Grant»	Action
<input type="checkbox"/> phpmyadmin	localhost	Oui	USAGE		Non	Changer les privileges Export
<input type="checkbox"/> root	localhost	Oui	ALL PRIVILEGES		Oui	Changer les privileges Export
<input type="checkbox"/> user	%	Oui	ALL PRIVILEGES		Oui	Changer les privileges Export



The screenshot shows the 'Changer les privileges : Compte d'utilisateur 'user'@'' (Change privileges: User account 'user'@') page in phpMyAdmin. The 'Nom d'hôte' (Host name) field is set to '%', which is highlighted by a yellow callout bubble labeled 'Configuration Important : le %'.

Global Base de données Modifier le mot de passe Information pour la connexion

Changer les privileges : Compte d'utilisateur 'user'@''

Changement des informations de connexion / Copie de compte d'utilisateur

Information pour la connexion

Nom d'utilisateur : Entrez une valeur: user

Nom d'hôte : Tout client %

Mot de passe : Conserver le mot de passe

Saisir à nouveau :

Greffon d'authentification : Authentification MySQL native

Générer un mot de passe: Générer

2 - Changer les privilèges : Compte d'utilisateur 'user'@'%'

Privileges globaux ☒ **Tout cocher**

Veillez noter que les noms de privileges sont exprimés en anglais.

<input checked="" type="checkbox"/> Données	<input checked="" type="checkbox"/> Structure	<input checked="" type="checkbox"/> Administration
<input checked="" type="checkbox"/> SELECT	<input checked="" type="checkbox"/> CREATE	<input checked="" type="checkbox"/> GRANT
<input checked="" type="checkbox"/> INSERT	<input checked="" type="checkbox"/> ALTER	<input checked="" type="checkbox"/> SUPER
<input checked="" type="checkbox"/> UPDATE	<input checked="" type="checkbox"/> INDEX	<input checked="" type="checkbox"/> PROCESS
<input checked="" type="checkbox"/> DELETE	<input checked="" type="checkbox"/> DROP	<input checked="" type="checkbox"/> RELOAD
<input checked="" type="checkbox"/> FILE	<input checked="" type="checkbox"/> CREATE TEMPORARY TABLES	<input checked="" type="checkbox"/> SHUTDOWN
	<input checked="" type="checkbox"/> SHOW VIEW	<input checked="" type="checkbox"/> SHOW DATABASES
	<input checked="" type="checkbox"/> CREATE ROUTINE	<input checked="" type="checkbox"/> LOCK TABLES
	<input checked="" type="checkbox"/> ALTER ROUTINE	<input checked="" type="checkbox"/> REFERENCES
	<input checked="" type="checkbox"/> EXECUTE	<input checked="" type="checkbox"/> REPLICATION CLIENT
	<input checked="" type="checkbox"/> CREATE VIEW	<input checked="" type="checkbox"/> REPLICATION SLAVE
	<input checked="" type="checkbox"/> EVENT	<input checked="" type="checkbox"/> CREATE USER
	<input checked="" type="checkbox"/> TRIGGER	

Ici on choisit les privilèges pour ce compte

Configurer MySQL pour accepter les connexions externes à la Raspberry Pi

Maintenant que les droits ont été donnés, nous allons devoir préciser à MySQL que nous souhaitons accepter des connexions externes à la Raspberry Pi.

Pour cela, nous allons éditer les fichiers de configuration de MySQL situé dans le dossier « */etc/mysql* » . Les fichiers à modifier sont *my.cnf* et *mariadb.cnf* .

Après avoir

```
<my.cnf>
Fichier  Édition  Recherche  Options  Aide
# The MariaDB configuration file
#
# The MariaDB/MySQL tools read configuration files in the following order:
# 1. "/etc/mysql/mariadb.cnf" (this file) to set global defaults,
# 2. "/etc/mysql/conf.d/*.cnf" to set global options.
# 3. "/etc/mysql/mariadb.conf.d/*.cnf" to set MariaDB-only options.
# 4. "~/.my.cnf" to set user-specific options.
#
# If the same option is defined multiple times, the last one will apply.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use
#
# This group is read both both by the
# use it for options that affect everything
#
[client-server]

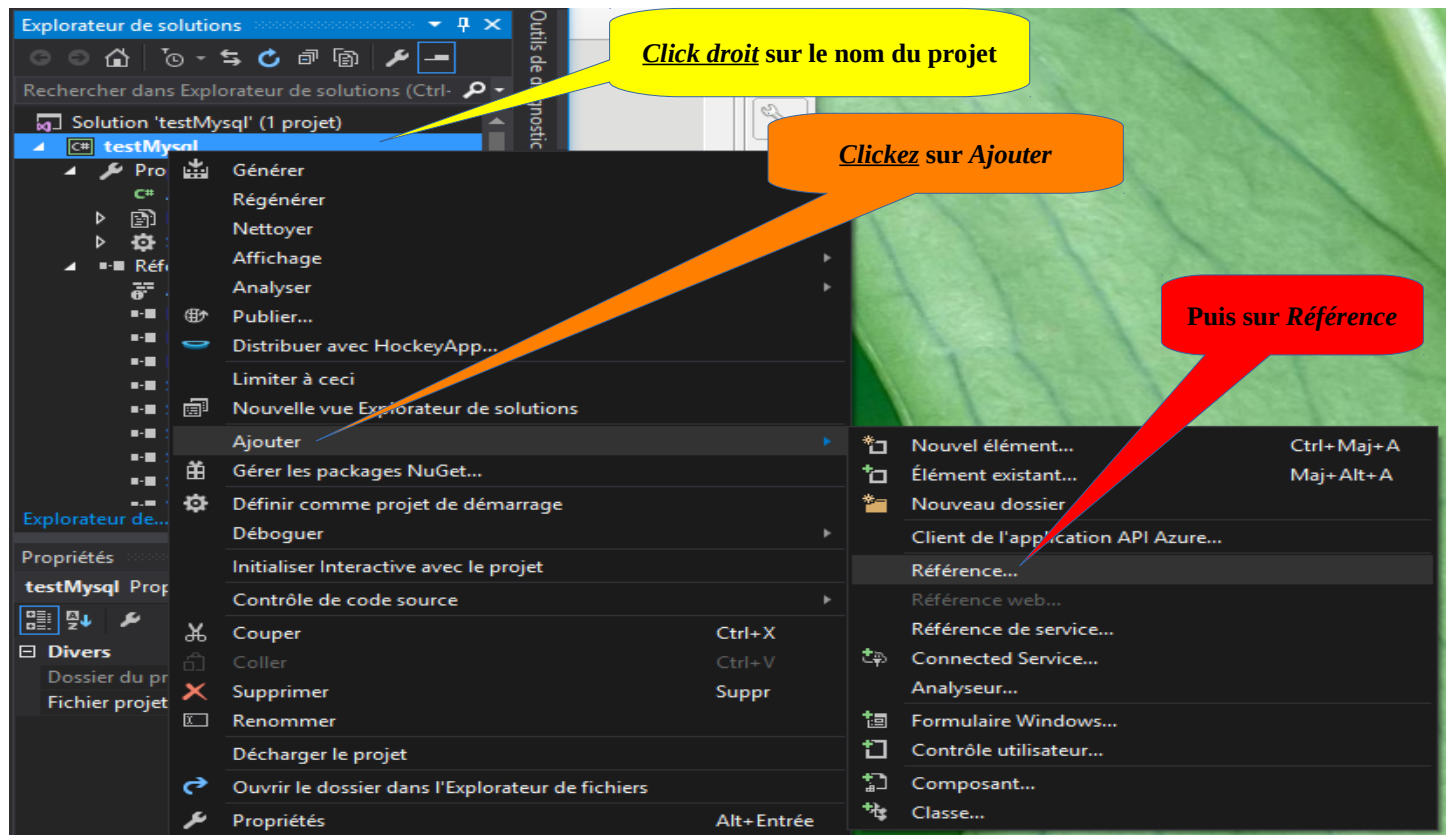
# Import all .cnf files from configuration directory
!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mariadb.conf.d/

bind-address = 0.0.0.0
```

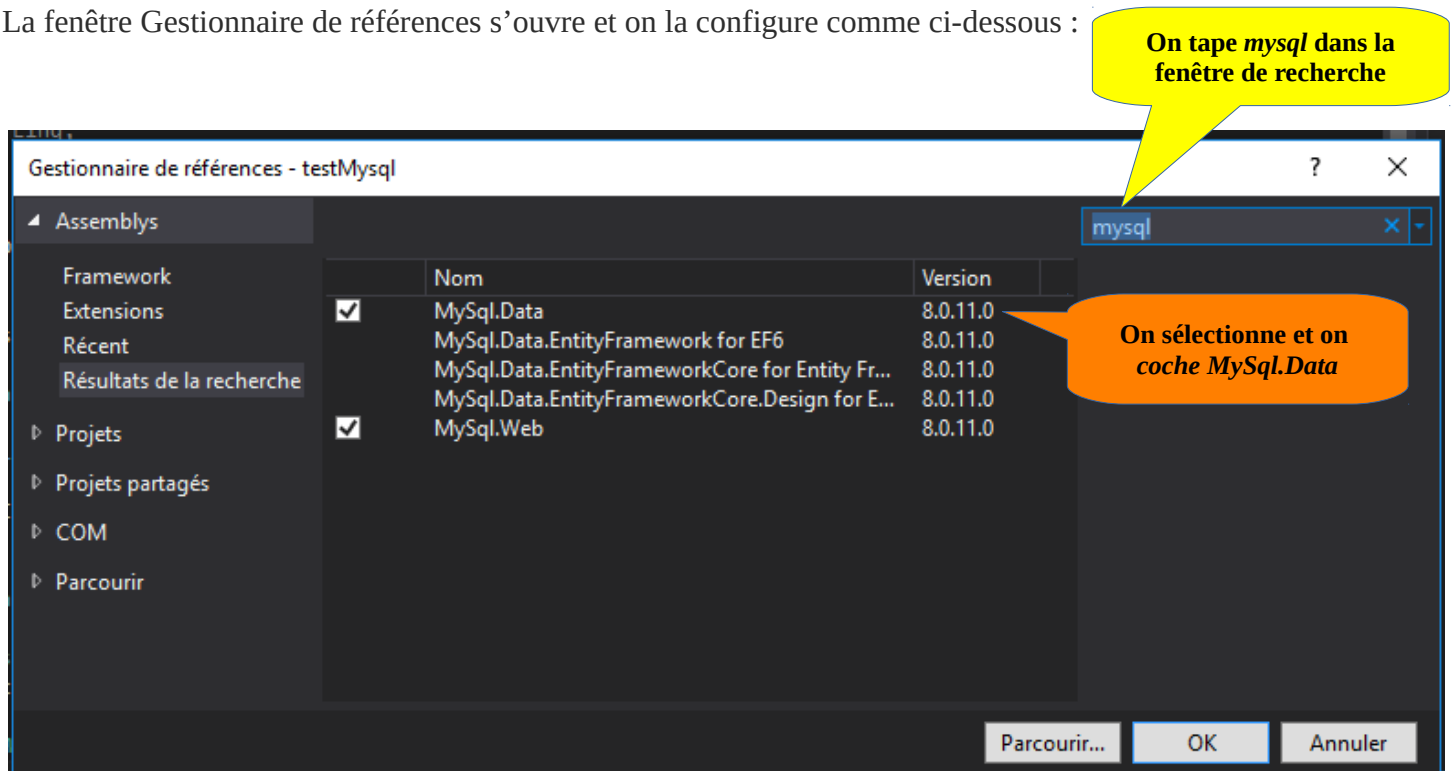
Ligne à rajouter dans les 2 fichiers !
Si bind-address = 127.0.0.1 existe, il faut la commenter
(#bind-address = 127.0.0.1)

appliquer toutes ces modifications sur la Raspberry Pi, on va maintenant se concentrer sur la réalisation de l'application en C# avec Visual Studio 2015 community. Tout d'abord, il faut installer une dll (**MySQL.data**) dans notre application. Pour ce faire il faut d'abord télécharger **Connector/Net 8.0.11** sur le site <https://dev.mysql.com/downloads/connector/net/8.0.html>. Installer le fichier téléchargé (**mysql-connector-net-8.0.11.msi**) sur votre ordinateur.

Après suivre les instructions suivantes en images :



La fenêtre Gestionnaire de références s'ouvre et on la configure comme ci-dessous :



Ci-dessous l'interface graphique de la gestion de la base de donnée :

The screenshot shows the 'Gestion MySQL' application window. It features several sections for managing a database:

- Ajouter**: A form with fields for Nom, Prénom, Status, NFC, and Véhicule. A blue circle with the number 4 is placed over the Status field.
- Recherche base de donnée Clients**: A search section with fields for Nom and Véhicule. A purple circle with the number 5 is placed over the Véhicule field. There are 'Chercher' buttons for each field.
- Se connecter**: A button with a red circle and the number 1 next to it.
- Recherche base de donnée enregistrement caméra**: A search section with fields for Nom, Immatriculation, and Date. A purple circle with the number 6 is placed over the Immatriculation field. There are 'Chercher' buttons for each field.
- Base de Données Clients**: A table with columns: ID, Nom, Prenom, Statut, Nfc, Vehicule, and Date. A yellow circle with the number 2 is placed over the table. To the right of the table are buttons for 'Affichage Clients' and 'Supprimer clients'.
- Enregistrements Caméra**: A table with columns: ID, Immatriculation, Nom, Date, Heure, and Autorisé. A green circle with the number 3 is placed over the table. To the right of the table are buttons for 'Affichage enregistrement' and 'Supprimer enregistrement'.
- Ajouter dans la base de donnée**: A button located below the 'Ajouter' form.

The screenshot shows the 'Modifier' application window, which is a form for updating database records. It contains the following fields:

- ID:
- Nom:
- Prenom:
- Status:
- Nfc:
- Vehicule:

A blue circle with the number 7 is placed to the right of the 'Prenom' field. At the bottom of the form is a 'Modifier' button.

Portion de programme pour la connection à la base de donnée avec la gestion du bouton 'Connection' - 'Déconnection' :

1

```
public partial class F_accueil : Form
{
    //Variable servant à créer la commande de connection au serveur:
    string connectionString = "Server=192.168.0.101;Port=3306;Database=BD_parking;Uid=user;Pwd=pass;SslMode=none";
    bool connecte = false; // Variable qui informe si la coonction est active ou pas (True ou Flase).
    MySqlConnection connection; // coonction = instance de la classe MySqlConnection.

    public F_accueil()
    {
        InitializeComponent();
    }

    // ***** Fonctions Connection - déconnection : *****
    private void button1_Click(object sender, EventArgs e)
    {
        // Association de l'instance 'connection' à la vraie connectionString:
        connection = new MySqlConnection(connectionString);

        if (button1.Text == "Se connecter")
        {
            try // Traitement si exceptions
            {
                //On vérifie si la connection est fermé:
                if (connection.State == ConnectionState.Closed)
                {
                    connection.Open(); // On ouvre la connection.
                    connecte = true; // On met à true car la connection est ouverte.
                    MessageBox.Show("Connecté !");
                    button1.Text = "Se deconnecter";
                }
            }
            catch (MySqlException co) // Si erreur, affichage de l'erreur.
            {
                MessageBox.Show(co.ToString());
                MessageBox.Show("Non Connecté !");
            }
        }
        else
        {
            // Pour fermer la connection au serveur:
            connection.Close();
            connecte = false; // On met à false car la connection est fermée.
            button1.Text = "Se connecter";
        }
    }
}
```

Section 'Base de Données Clients' :

2

Elle est constituée d'un **GroupBox**, d'un **ListView**, d'un **ContextMenuStrip** et de 2 **Button**. Un **Button** 'Affichage Clients' permet d'afficher les informations contenu dans la base de données TAB_clients dans le **ListView**. Le **Button** 'Supprimer clients' supprime un client sélectionné par la souris. **ContextMenuStrip** est un menu qui sera affiché avec un click sur le bouton droit de la souris . Il servira à modifier ou supprimer un client .

```

//***** Fonction affichage de la table TAB_clients : *****
private void button3_Click(object sender, EventArgs e)
{
    if (connecte)
    {
        listView1.Items.Clear(); // On efface les données affichées dans le contrôle listView1.
                                // (listView1 permet d'afficher les différents clients sous
                                // forme de tableau et colonnes.)
        // Commande SQL pour afficher tous les clients de TAB_clients:
        string query = "SELECT * FROM TAB_clients";
        MySqlCommand cmd = new MySqlCommand(query, connection);
        using (MySqlDataReader lecture = cmd.ExecuteReader())
        {
            while (lecture.Read()) //Boucle qui permet d'afficher tout les clients.
            {
                // On récupère les infos envoyé par la base de donnée:
                string ID = lecture["idClients"].ToString();
                string Nom = lecture["nomClients"].ToString();
                string Prenom = lecture["prenomClients"].ToString();
                string Statut = lecture["statutClients"].ToString();
                string Nfc = lecture["nfcClients"].ToString();
                string Vehicule = lecture["vehiculeClients"].ToString();
                string Date = lecture["date_creationClients"].ToString();
                //Affichage des clients dans la listView1:
                listView1.Items.Add(new ListViewItem(new[] { ID, Nom, Prenom, Statut, Nfc, Vehicule, Date }));
            }
        }
    }
    else
    {
        MessageBox.Show("Vous n'êtes pas connecté !");
    }
}

```

```

//***** Fonction supprimer une rangee de donnée de la table TAB_clients : *****
// Ici on utilise le click droit de la souris et l'outil "ContextMenuStrip".
private void supprimerToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (connecte)
    {
        if (listView1.SelectedItems.Count > 0) //count=13 si 13 lignes de donnée.
        {
            ListViewItem element = listView1.SelectedItems[0]; // 0 ou erreur.
            // recupère le contenu de la colonne 0 de la ligne élément ici l'idClients,
            // exemple: si.SubItems[2].text on récupérera le prénom du client :
            string Id = element.SubItems[0].Text;
            // Requête SQL pour effacer un client dans TAB_clients en donnant son idClients:
            string query = "DELETE FROM TAB_clients WHERE idClients=@id";
            MySqlCommand cmd = new MySqlCommand(query, connection);
            // On additionne L'ID dans la requête:
            cmd.Parameters.AddWithValue("@id", Id);
            cmd.ExecuteNonQuery();
            element.Remove();
            MessageBox.Show("Element supprimé !");
        }
    }
}

```



```

//***** 2ème Fonction supprimer une rangee de donnée de la table TAB_clients : *****
// Ici on utilise l'outil de base "button".
private void button2_Click_1(object sender, EventArgs e)
{
    if (connecte)
    {
        if (listView1.SelectedItems.Count > 0) //count=13 si 13 lignes de donnée.
        {
            ListViewItem element = listView1.SelectedItems[0]; // 0 ou erreur.
            // recupère le contenu de la colonne 0 de la ligne element ici l'idClients.
            string Id = element.SubItems[0].Text;
            string query = "DELETE FROM TAB_clients WHERE idClients=@id";
            MySqlCommand cmd = new MySqlCommand(query, connection);
            cmd.Parameters.AddWithValue("@id", Id);
            cmd.ExecuteNonQuery();
            element.Remove();
            MessageBox.Show("Element supprimé !");
        }
        else MessageBox.Show("Veuillez selectionner un clients !");
    }
}

//***** Fonction modifier clients dans la base de donnee : *****
// Utilisation du click droit souris puis click sur Modifier.
// Apparition d'une fenêtre 'Modifier' pour insérer les nouvelles données.
*/
private void modifierToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (listView1.SelectedItems.Count > 0) // Si on a une liste de client.
    {
        // On récupère les données de chaque colonne du client sélectionné :
        ListViewItem element = listView1.SelectedItems[0]; // 0 ou erreur.
        string ID = element.SubItems[0].Text;
        string nom = element.SubItems[1].Text;
        string prenom = element.SubItems[2].Text;
        string statut = element.SubItems[3].Text;
        string nfc = element.SubItems[4].Text;
        string vehicule = element.SubItems[5].Text;

        // Ci dessous on va modifier le Form nommé 'modifier' avec les données sélectionnées
        // dans 'listView' .
        using (modifier m = new modifier() )
        {
            m.Id = ID; // 'm.Id' correspond à la variable crée dans 'modifier.cs'
            m.Nom = nom; // Idem pour les données suivantes
            m.Prenom = prenom;
            m.Statut = statut;
            m.Nfc = nfc;
            m.Vehicule = vehicule;

            //m.ShowDialog(); ligne qui ouvre le Form 'modifier'.

            /* La condition suivante ouvre automatiquement la Fenêtre 'Modifier' même si la
            condition n'est pas remplie.
            Si la condition est remplie, c'est à dire si l'on a cliqué sur le bouton 'Modifier'
            de la Fenêtre 'Modifier' les valeurs du client id,nom etc... affichées dans les 'textBox'
            seront envoyées dans la Base de donnée pour la modifier.
            */
            if (m.ShowDialog() == DialogResult.Yes)
            {
                // Préparation de la requête SQL et envoi à la table 'TAB_clients :
                string query = "UPDATE TAB_clients SET statutClients=@statut, nfcClients=@nfc, vehiculeClients=@vehicule WHERE idClients=@id ";
                MySqlCommand cmd = new MySqlCommand(query, connection);
                cmd.Parameters.AddWithValue("@id", ID);
                cmd.Parameters.AddWithValue("@statut", m.Statut);
                cmd.Parameters.AddWithValue("@nfc", m.Nfc);
                cmd.Parameters.AddWithValue("@vehicule", m.Vehicule);
                cmd.ExecuteNonQuery();
                // mise à jour de la liste clients :
                element.SubItems[3].Text = m.Statut;
                element.SubItems[4].Text = m.Nfc;
                element.SubItems[5].Text = m.Vehicule;
                MessageBox.Show("Client modifié !");
            }
        }
    }
}
}
}

```

```

//***** Fonction affichage de la table TAB_camera : *****
private void button6_Click(object sender, EventArgs e)
{
    if (connecte)
    {
        listView2.Items.Clear();
        string query = "SELECT * FROM TAB_camera";
        MySqlCommand cmd = new MySqlCommand(query, connection);
        using (MySqlDataReader lecture = cmd.ExecuteReader())
        {
            while (lecture.Read())
            {
                string ID = lecture["idCamera"].ToString();
                string plaque = lecture["plaqueCamera"].ToString();
                string nom = lecture["nomCamera"].ToString();
                string date = lecture["dateCamera"].ToString();
                string heure = lecture["heureCamera"].ToString();
                string autorisation = lecture["autorisationCamera"].ToString();

                listView2.Items.Add(new ListViewItem(new[] { ID, plaque, nom, date, heure, autorisation }));
            }
        }
    }
    else MessageBox.Show("Vous n'êtes pas connecté !");
}

//***** Fonction supprimer une rangee de donnée de la table TAB_camera : *****
// Ici on utilise le click droit de la souris et l'outil "ContextMenuStrip (2)".
private void supprimerToolStripMenuItem1_Click(object sender, EventArgs e)
{
    if (connecte)
    {
        if (listView2.SelectedItems.Count > 0) //count=13 si 13 lignes de donnée.
        {
            ListViewItem element = listView2.SelectedItems[0]; // 0 ou erreur.
            // recupère le contenu de la colonne 0 de la ligne élément ici l'idClients.
            string Id = element.SubItems[0].Text;
            string query = "DELETE FROM TAB_camera WHERE idCamera=@id";
            MySqlCommand cmd = new MySqlCommand(query, connection);
            cmd.Parameters.AddWithValue("@id", Id);
            cmd.ExecuteNonQuery();
            element.Remove();
            MessageBox.Show("Element supprimé !");
        }
    }
}

```



```

//***** 2ème Fonction supprimer une rangee de donnée de la table TAB_camera : *****
// Ici on utilise l'outil de base "button".
private void button7_Click(object sender, EventArgs e)
{
    if (connecte)
    {
        if (listView2.SelectedItems.Count > 0) //count=13 si 13 lignes de donnée.
        {
            ListViewItem element = listView2.SelectedItems[0]; // 0 ou erreur.
            // recupère le contenu de la colonne 0 de la ligne element ici l'idClients.
            string Id = element.SubItems[0].Text;
            string query = "DELETE FROM TAB_clients WHERE idClients=@id";
            MySqlCommand cmd = new MySqlCommand(query, connection);
            cmd.Parameters.AddWithValue("@id", Id);
            cmd.ExecuteNonQuery();
            element.Remove();
            MessageBox.Show("Element supprimé !");
        }
        else MessageBox.Show("Veuillez selectionner un clients !");
    }
    else MessageBox.Show("Vous n'êtes pas connecté !");
}

```

4

```

//***** Fonction Ajouter : *****
private void button2_Click(object sender, EventArgs e)
{
    // On vérifie que tout les textBox sont remplis (1 à 5):
    if (textBox1.Text == "") MessageBox.Show("Entrez un nom !");

    else if (textBox2.Text == "") MessageBox.Show("Entrez un prénom !");

    else if (textBox3.Text == "") MessageBox.Show("Entrez le statut !");

    else if (textBox4.Text == "") MessageBox.Show("Entrez le code NFC !");

    else if (textBox5.Text == "") MessageBox.Show("Entrez l'immatriculation !");

    else
    {
        if (connecte) // Si on est connecté à la base de donnée.
        {
            try // Gestion des exceptions.
            {
                // Commande SQL pour ajouter un client dans la base de donnée:
                string query = "INSERT INTO TAB_clients(nomClients, prenomClients, statutClients, nfcClients, vehiculeClients) VALUES(@nom, @prenom, @statut, @nfc, @vehicule)";
                MySqlCommand cmd = new MySqlCommand(query, connection);
                cmd.Prepare();
                //On récupère les infos des textBox et on les ajoutent à la commande SQL 'query':
                cmd.Parameters.AddWithValue("@nom", textBox1.Text);
                cmd.Parameters.AddWithValue("@prenom", textBox2.Text);
                cmd.Parameters.AddWithValue("@statut", textBox3.Text);
                cmd.Parameters.AddWithValue("@nfc", textBox4.Text);
                cmd.Parameters.AddWithValue("@vehicule", textBox5.Text);
                cmd.ExecuteNonQuery(); // On envoie la requête SQL au serveur.
                cmd.Parameters.Clear();
                MessageBox.Show("Ajouté !!!");
            }
            catch (MySQL.Data.MySQLClient.MySQLException ex)
            {
                MessageBox.Show("Erreur ajouter" + ex.Message + ex.Number);
            }
        }
        else MessageBox.Show ("Vous n'êtes pas connecté !");
    }
}

```

***** Fonction chercher un client par son nom : *****

```
private void button4_Click(object sender, EventArgs e)
{
    if (connecte)
    {
        listView1.Items.Clear();
        string query = "SELECT * FROM TAB_clients WHERE nomClients = @nom";
        MySqlCommand cmd = new MySqlCommand(query, connection);
        cmd.Parameters.AddWithValue("@nom", textBox6.Text);
        using (MySqlDataReader lecture = cmd.ExecuteReader())
        {
            while (lecture.Read())
            {
                string ID = lecture["idClients"].ToString();
                string Nom = lecture["nomClients"].ToString();
                string Prenom = lecture["prenomClients"].ToString();
                string Statut = lecture["statutClients"].ToString();
                string Nfc = lecture["nfcClients"].ToString();
                string Vehicule = lecture["vehiculeClients"].ToString();
                string Date = lecture["date_creationClients"].ToString();

                listView1.Items.Add(new ListViewItem(new[] { ID, Nom, Prenom, Statut, Nfc, Vehicule, Date }));
            }
        }
    }
}
```

***** Fonction chercher un client par son immatriculation : *****

```
private void button5_Click(object sender, EventArgs e)
{
    listView1.Items.Clear();
    string query = "SELECT * FROM TAB_clients WHERE vehiculeClients = @vehicule";
    MySqlCommand cmd = new MySqlCommand(query, connection);
    cmd.Parameters.AddWithValue("@vehicule", textBox6.Text);
    using (MySqlDataReader lecture = cmd.ExecuteReader())
    {
        while (lecture.Read())
        {
            string ID = lecture["idClients"].ToString();
            string Nom = lecture["nomClients"].ToString();
            string Prenom = lecture["prenomClients"].ToString();
            string Statut = lecture["statutClients"].ToString();
            string Nfc = lecture["nfcClients"].ToString();
            string Vehicule = lecture["vehiculeClients"].ToString();
            string Date = lecture["date_creationClients"].ToString();

            listView1.Items.Add(new ListViewItem(new[] { ID, Nom, Prenom, Statut, Nfc, Vehicule, Date }));
        }
    }
}
```

```

//***** Fonction chercher un enregistrement par son nom dans TAB_camera: *****
private void button8_Click(object sender, EventArgs e)
{
    listView2.Items.Clear();
    string query = "SELECT * FROM TAB_camera WHERE nomCamera = @nom";
    MySqlCommand cmd = new MySqlCommand(query, connection);
    cmd.Parameters.AddWithValue("@nom", textBox8.Text);
    using (MySqlDataReader lecture = cmd.ExecuteReader())
    {
        while (lecture.Read())
        {
            string ID = lecture["idCamera"].ToString();
            string plaque = lecture["plaqueCamera"].ToString();
            string Nom = lecture["nomCamera"].ToString();
            string date = lecture["dateCamera"].ToString();
            string heure = lecture["heureCamera"].ToString();
            string autorisation = lecture["autorisationCamera"].ToString();

            listView2.Items.Add(new ListViewItem(new[] { ID, plaque, Nom, date, heure, autorisation }));
        }
    }
}

//***** Fonction chercher un enregistrement par son immatriculation dans TAB_camera: *****
private void button9_Click(object sender, EventArgs e)
{
    if (connecte)
    {
        listView2.Items.Clear();
        string query = "SELECT * FROM TAB_camera WHERE plaqueCamera = @plaque";
        MySqlCommand cmd = new MySqlCommand(query, connection);
        cmd.Parameters.AddWithValue("@plaque", textBox9.Text);
        using (MySqlDataReader lecture = cmd.ExecuteReader())
        {
            while (lecture.Read())
            {
                string ID = lecture["idCamera"].ToString();
                string plaque = lecture["plaqueCamera"].ToString();
                string Nom = lecture["nomCamera"].ToString();
                string date = lecture["dateCamera"].ToString();
                string heure = lecture["heureCamera"].ToString();
                string autorisation = lecture["autorisationCamera"].ToString();

                listView2.Items.Add(new ListViewItem(new[] { ID, plaque, Nom, date, heure, autorisation }));
            }
        }
    }
    else MessageBox.Show("Vous n'êtes pas connecté !");
}

```

```

//***** Fonction chercher un enregistrement entre 2 dates dans TAB_camera: *****
// Il faut ici modifier les propriétés Format=Custom et CustomFormat=yyyy-MM-dd
private void button10_Click(object sender, EventArgs e)
{
    if (connecte)
    {
        listView2.Items.Clear();
        string query = "SELECT * FROM TAB_camera WHERE dateCamera BETWEEN @date1 AND @date2";
        MySqlCommand cmd = new MySqlCommand(query, connection);
        cmd.Parameters.AddWithValue("@date1", dateTimePicker1.Text);
        cmd.Parameters.AddWithValue("@date2", dateTimePicker2.Text);

        using (MySqlDataReader lecture = cmd.ExecuteReader())
        {
            while (lecture.Read())
            {
                string ID = lecture["idCamera"].ToString();
                string plaque = lecture["plaqueCamera"].ToString();
                string Nom = lecture["nomCamera"].ToString();
                string date = lecture["dateCamera"].ToString();
                string heure = lecture["heureCamera"].ToString();
                string autorisation = lecture["autorisationCamera"].ToString();

                listView2.Items.Add(new ListViewItem(new[] { ID, plaque, Nom, date, heure, autorisation }));
            }
        }
    }
    else MessageBox.Show("Vous n'êtes pas connecté !");
}

```

***Pour rajouter une nouvelle fenêtre aller dans : **Projet** puis **Ajouter un formulaire Windows...** ***

```

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;

// Programme de la fenêtre 'Modifier' permettant d'insérer les nouvelles infos du client sélectionné
// dans listView1 de la fenêtre principale.

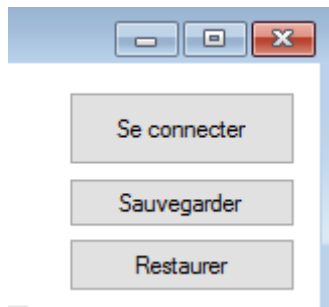
namespace testMysql
{
    public partial class modifier : Form
    {
        // Création de 6 variables string utilisées dans la fonction:
        // (private void modifierToolStripMenuItem_Click(object sender, EventArgs e)).
        public string Id { set { textBox7.Text = value; } }
        public string Nom { get { return textBox8.Text; } set { textBox8.Text = value; } }
        public string Prenom { get { return textBox9.Text; } set { textBox9.Text = value; } }
        public string Statut { get { return textBox10.Text; } set { textBox10.Text = value; } }
        public string Nfc { get { return textBox11.Text; } set { textBox11.Text = value; } }
        public string Vehicule { get { return textBox12.Text; } set { textBox12.Text = value; } }

        public modifier()
        {
            InitializeComponent();

            // Pour lancer la modification dans la base de donnée:
            private void button2_Click(object sender, EventArgs e)
            {
                DialogResult = DialogResult.Yes;
            }
        }
    }
}

```

Rajout de 2 boutons pour sauvegarder et restaurer la base de donnée :



```
//***** Sauvegarder la base de donnée : *****
private void button11_Click(object sender, EventArgs e)
{
    string constring = "server=192.168.0.101;user=user;pwd=pass;database=BD_parking;";
    string file = "D:\\backup.sql";
    using (MySqlConnection conn = new MySqlConnection(constring))
    {
        using (MySqlCommand cmd = new MySqlCommand())
        {
            using (MySqlBackup mb = new MySqlBackup(cmd))
            {
                cmd.Connection = conn;
                conn.Open();
                mb.ExportToFile(file);
                conn.Close();
            }
        }
    }
}

//***** Restaurer la base de donnée : *****
private void button12_Click(object sender, EventArgs e)
{
    string constring = "server=192.168.0.101;user=user;pwd=pass;database=BD_parking;";
    string file = "D:\\backup.sql";
    using (MySqlConnection conn = new MySqlConnection(constring))
    {
        using (MySqlCommand cmd = new MySqlCommand())
        {
            using (MySqlBackup mb = new MySqlBackup(cmd))
            {
                cmd.Connection = conn;
                conn.Open();
                mb.ImportFromFile(file);
                conn.Close();
            }
        }
    }
}
```

Ces deux fonctions utilisent la class **MySqlBackup**. Elle n'est pas installée par défaut dans Visual Studio. Pour l'installer, il faut la rajouter à notre projet en allant dans *l'explorateur de solutions* et faire un clic droit sur **Références** et choisir **Gérer les packages NuGet...** . Dans le Gestionnaire de package **NuGet**, chercher **MySqlBackup par Yasith Jayswardana** et installé le.

(Voir ci dessous une capture d'écran:)

