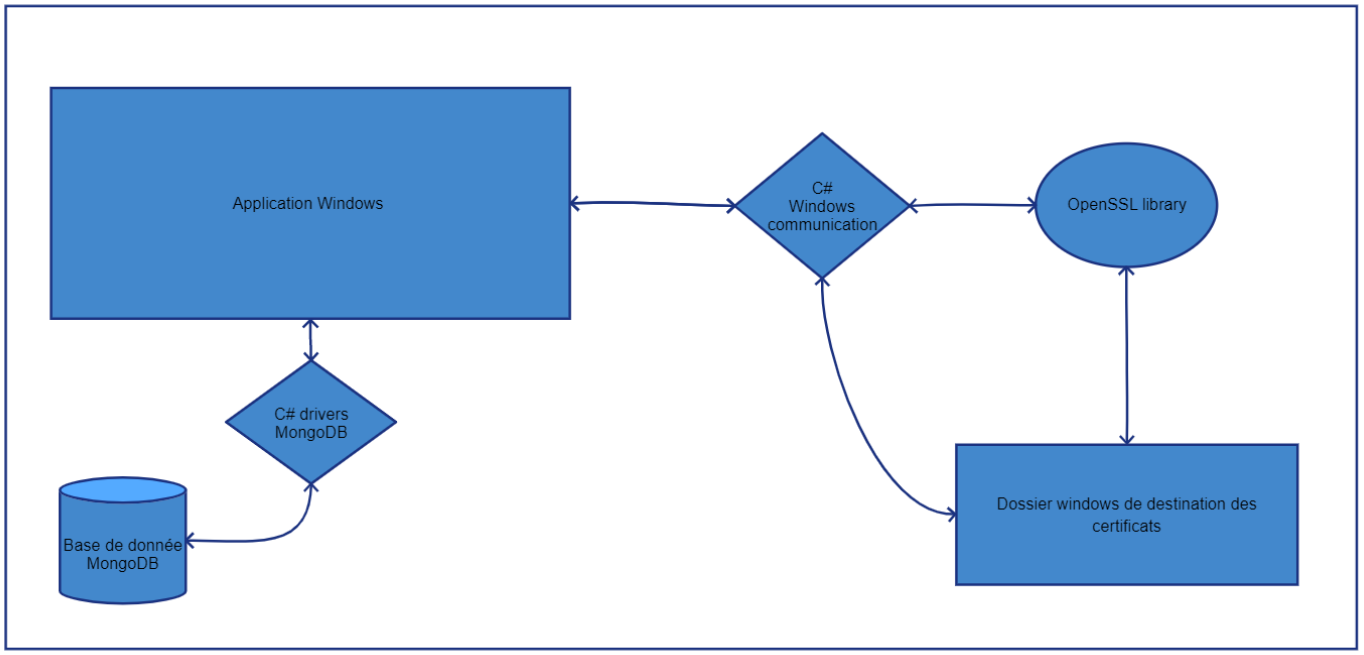
Maitrise technique

Présentation : Cette maitrise technique a pour but d'expliquer en détail la conception ainsi que les choix de conceptions du projet CertSSL.

1) Schéma logique de l'application.

CertSSL est une application Windows dont le but est de généré des certificats SSL. L'application s'articule autour d'un système de profils stockés en base de données contenant les informations nécessaires à la création de certificats. On utilisera les Library incluses avec Visual studio afin de communiquer avec Windows (pour la création, déplacement des fichiers). On communiquera avec OpenSSL afin de pouvoir signer les certificats. L'application fera office de client-serveur à l'aide de la Library mongoDB.Drivers permettant l'envoie et la réception de donnée ainsi que le traitement de celle-ci.



2) Technique

Notre application est développée en C# à l'aide du Framework .NET. Afin de faciliter l'installation, nous mettrons en place un setup à l'aide de Microsoft Setup Installer qui permettra automatiquement d'installer .NET si il n'est pas présent sur a machine ainsi que d'installer l'application facilement (installation, création de raccourcis, des logo etc…).

J'utilise donc un stack C#/.NET MongoDB :

MongoDB = Base de données nosql

C# = langage de programmation objet

.Net = Framework Windows

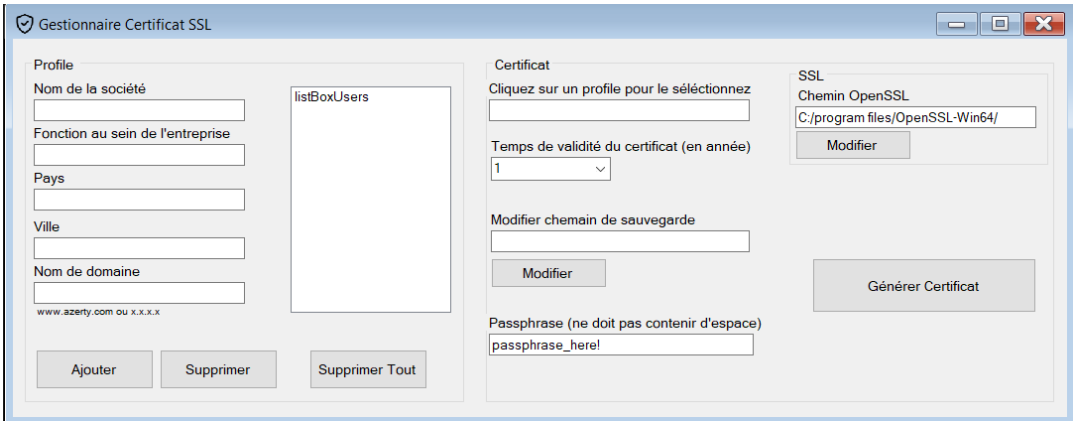
Je détaille par la suite comment s'articule ces différentes technologies et leurs utilisations à travers ce projet.

2.1) Frontend

Le Frontend est créer grâce au Framework .NET via la fonction [Design] permettant de créer des objets visuels et ce qui permet donc d'interagir avec. On peut définir des fonctions qui se déclenche en interagissant avec (exemple ici d'un bouton cliquable)

```
1 référence
private void button5_Click(object sender, EventArgs e)
{
```

On utilise directement l'outil de design intégrer à l'IDE afin de créer le design.



On retrouve donc un design séparer en 2 parties : Profil et certificat.

Profil : lorsque l'on sélectionne un profil, le champ correspondant à celui-ci doivent se remplir automatiquement. On peut également le supprimer ou créer un nouveau profil.

Certificats : lorsqu'un profil est sélectionné, il se charger dans la partie certificat. Une combobox permet de choisir une date de validité (6 mois, 1,2,5 ans).

Les champs de cheminement ne sont éditables que via le bouton modifié qui ouvre un explorateur de fichier et permet à l'utilisateur de choisir les dossiers cibles.

```
DialogResult result = folderBrowserDialog1.ShowDialog();
if (result == DialogResult.OK)
{
    textBoxSSL.Text = folderBrowserDialog1.SelectedPath;
    Environment.SpecialFolder root = folderBrowserDialog1.RootFolder;
}
```

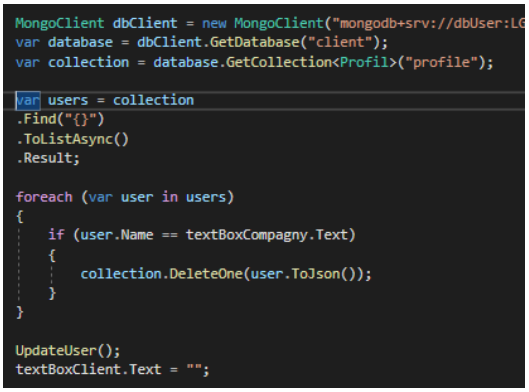
2.2) *Backend*

2.2.1) Communication MongoDB

-Ajouter : On peut ajouter un profil, pour cela on vérifie d'abord s'il n'existe pas déjà en base (car pour ne pas les confondre, on oblige un nom unique par profil). S'il existe on retourne un message d'erreur, sinon on le créer et on l'ajoute à la base de données.

```
if (test() == true)//verifie que le profile n'existe pas deja
{
    labelDebug.ForeColor = System.Drawing.Color.Red;
    labelDebug.Text = "Ce profile existe deja";
}
else
{
    MongoClient dbClient = new MongoClient("mongodb+srv://dbUser:LGNYQijN2ZSmiFel@cluster0.hqef6.mongodb.net/client?retryWrites=true&w=majority");
    var database = dbClient.GetDatabase("client");
    var collection = database.GetCollection<Profil>("profile");
    var entity = new Profil { Name = textBoxCompagny.Text, Function = textBoxFunction.Text, Country = textBoxCountry.Text, StateLetter = CompagnyStateLe
    collection.InsertOne(entity); //Insert le nouveau profile en base
    UpdateUser();
    labelDebug.ForeColor = System.Drawing.Color.Green;
    labelDebug.Text = "Profile ajouter avec succes";
}
```

-Supprimer : On récupère la base de données complète et on vérifie chaque profil afin du supprimer celui correspondant à celui sélectionné (il est plus rapide de faire ca que de directement trouvé le bon en base car les server atlas sont peu performant).



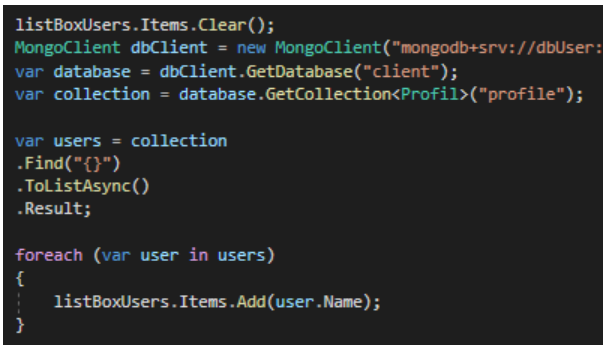
```
MongoClient dbClient = new MongoClient("mongodb+srv://dbUser:LG
var database = dbClient.GetDatabase("client");
var collection = database.GetCollection<Profil>("profile");

var users = collection
.Find("{}")
.ToListAsync()
.Result;

foreach (var user in users)
{
    if (user.Name == textBoxCompagny.Text)
    {
        collection.DeleteOne(user.ToJson());
    }
}

UpdateUser();
textBoxClient.Text = "";
```

-Mettre à jour la liste : On utilise la fonction UpdateUser qui permet de réactualiser la liste des profils depuis la base de données et met à jour la liste des profils en fonction de la base, ainsi les données visible en base et par l'utilisateur sont constamment les mêmes.



```
listBoxUsers.Items.Clear();
MongoClient dbClient = new MongoClient("mongodb+srv://dbUser:
var database = dbClient.GetDatabase("client");
var collection = database.GetCollection<Profil>("profile");

var users = collection
.Find("{}")
.ToListAsync()
.Result;

foreach (var user in users)
{
    listBoxUsers.Items.Add(user.Name);
}
```

2.2.2) L'architecture

Voici l'architecture de la base de données.

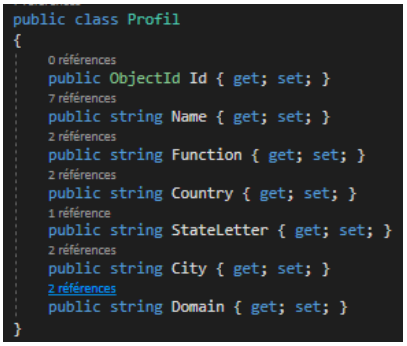
Ces modèles appelés schéma seront appelé dès qu'il faudra créer un utilisateur.

Profil:

CompanyName = string ;
CompanyFunction = string ;
CompanyCounty = string;
CompanyCity= string;
CompagnyDomain = string;

2.2.3) MongoDB

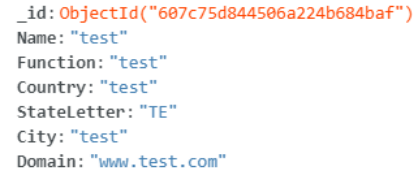
Ce service nosql permet d'héberger les données sous format json. Des schémas sont prédéfinis ce qui permet ensuite de récupérer les données grâce à mongoDB.Driver.



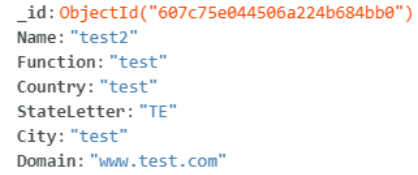
```
public class Profil
{
    0 références
    public ObjectId Id { get; set; }
    7 références
    public string Name { get; set; }
    2 références
    public string Function { get; set; }
    2 références
    public string Country { get; set; }
    1 référence
    public string StateLetter { get; set; }
    2 références
    public string City { get; set; }
    2 références
    public string Domain { get; set; }
}
```

Exemple de stockage en base de données :

```
_id: ObjectId("607b539dbd6b95046653c5a9")
Name: "rfit"
Function: "dev"
Country: "france"
StateLetter: "FR"
City: "Valence"
Domain: "www.rfit-tech.com"
```



```
_id: ObjectId("607c75d844506a224b684baf")
Name: "test"
Function: "test"
Country: "test"
StateLetter: "TE"
City: "test"
Domain: "www.test.com"
```



```
_id: ObjectId("607c75e044506a224b684bb0")
Name: "test2"
Function: "test"
Country: "test"
StateLetter: "TE"
City: "test"
Domain: "www.test.com"
```

2.2.4) MongoDB

Afin de généré les certificat on utilise un appel en ligne de commande au library de OpenSSL qui doit etre préinstaller sur la machine. On commence par crerr i, nouveau processus accedant au ressources de OpenSSL via commande line (car OpenSSL est développé avec des library en C++ imcompatible avec C#, on execute donc le rpogramme via un cli en interne de notre programme C# pour vouvoir communiquer avec).

```
string SSL = textBoxSSL.Text;
if (File.Exists(SSL + "/bin/openssl.exe") == true) //verifie que OpenSSL soit accesible
{
    //démarre un processus hérité a la racine d'Openssl
    System.Diagnostics.ProcessStartInfo procStartInfo =
    new System.Diagnostics.ProcessStartInfo("cmd", "/K ");
    procStartInfo.UseShellExecute = false;
    procStartInfo.WorkingDirectory = SSL + "/bin";
    procStartInfo.CreateNoWindow = true;

    System.Diagnostics.Process proc = new System.Diagnostics.Process();
    proc.StartInfo = procStartInfo;

    proc.StartInfo.RedirectStandardOutput = true;
    proc.StartInfo.RedirectStandardInput = true;
    proc.Start();
```

On récupere tous les champs « profils » qui seront utiles dans une variable au format attendu par OpenSSL

```
string stateletter = textBoxCountry.Text.Substring(0, 2);
stateletter = stateletter.ToUpper();
string subj = "/C=" + stateletter + "/ST=" + textBoxCountry.Text + "/L=" + textBoxCity.Text + "/O=" + text
```

On envoies les commades avec les arguments attendu par OpenSSL afin de crer les certificat, et les signé avec les datas renseignié via l'application et via le profil séléctionné.

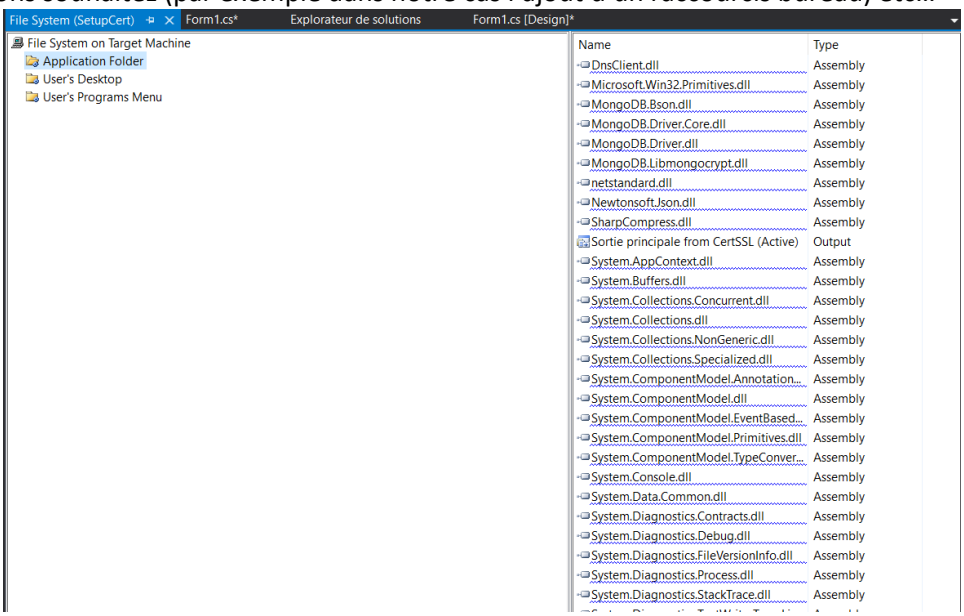
```
//Permet de généré les différants certificat / clée
proc.StandardInput.WriteLine("openssl genrsa -des3 -passout pass:" + textBoxPass.Text + " -out ca.key 2048 ");
proc.StandardInput.WriteLine("openssl req -new -x509 -days " + days + " -key ca.key -passin pass:" + textBoxPass.Text + " -out ca.crt
proc.StandardInput.WriteLine("openssl genrsa -out server.key 2048");
proc.StandardInput.WriteLine("openssl req -new -out server.csr -key server.key -subj \"" + subj + "\"");
proc.StandardInput.WriteLine("openssl x509 -req -passin pass:" + textBoxPass.Text + " -in server.csr -CA ca.crt -CAkey ca.key -CAcreat
proc.StandardInput.WriteLine("openssl genrsa -out client.key 2048");
proc.StandardInput.WriteLine("openssl req -new -out client.csr -key client.key -subj " + subj);
proc.StandardInput.WriteLine("openssl x509 -req -passin pass:" + textBoxPass.Text + " -in client.csr -CA ca.crt -CAkey ca.key -CAcreat
proc.Close(); //Ferme le processus
```

Les certificats sont créé, on à ensuite qu'à les récupéré avec leurs clés et les déplacé dans le dossier attendu en les triant par profils, et par dates.

```
//crer un répertoire à l'endroit séléctionné préalablement
string path = textBoxFolder.Text + "/" + textBoxCompagny.Text + "/" + DateTime.Now.T
Directory.CreateDirectory(path);
//Déplace les certificats dans le répertoire
File.Move(SSL + "/bin/ca.crt", path + "/" + "ca.crt");
File.Move(SSL + "/bin/ca.key", path + "/" + "ca.key");
File.Move(SSL + "/bin/ca.srl", path + "/" + "ca.srl");
File.Move(SSL + "/bin/client.crt", path + "/" + "client.crt");
File.Move(SSL + "/bin/client.key", path + "/" + "client.key");
File.Move(SSL + "/bin/client.csr", path + "/" + "client.csr");
File.Move(SSL + "/bin/server.crt", path + "/" + "server.crt");
File.Move(SSL + "/bin/server.key", path + "/" + "server.key");
File.Move(SSL + "/bin/server.csr", path + "/" + "server.csr");
```

3) Setup

Le Setup à été développé à l'aide de Microsoft Setup Installer. On créer d'abord une realease du projet, et le plugin permet de faire le liens entre les fichier nécéssaire à l'installation de cette realease, il faut ensuite inclure les library et développé les options souhaitez (par exemple dans notre cas l'ajout d'un raccourcis bureau) etc…



4) Test

Les tests se sont révélé etre concluant. L'ajout d'un nombre allant jusqu'à 5000 profil ne pose pas de problèmes.

Le Setup fonctionne correctement sous windows 7, 8, 8.1 et 10.

Une connexion internet instable peut poser problème et faire planter l'application car celle si communique avec la base de donnée. En effet les PC de RFI T etant cablé et le réseaux étant stable, la problèmatique d'un réseau instable ne se pose pas réelement de plus l'application est prévu pour etre constament connecté à internet pour pouvoir échanger les profils entre les diffférants techniciens.