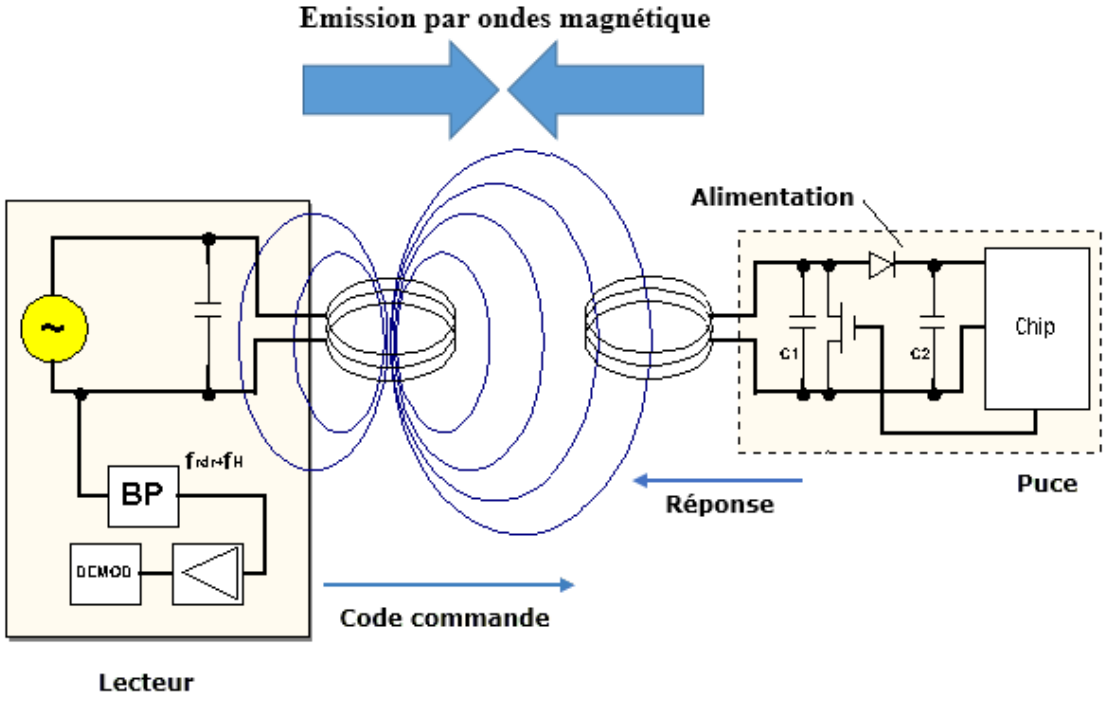
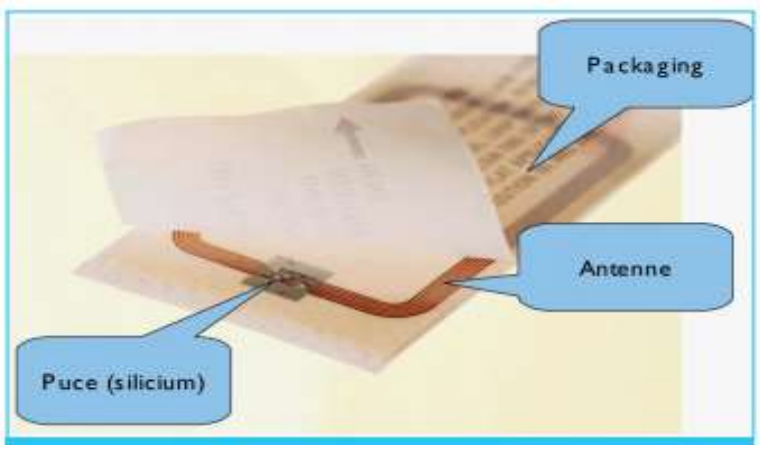
I) Présentation de la technologie RFID «Radio Frequency Identification » :

Le terme RFID désigne un système d'identification qui comprend une étiquette électronique (ou *tag*), pour mémoriser des informations, et un lecteur.

Le transfert d'information du composant électronique vers le lecteur s'effectue par radiofréquence



1) Principe du fonctionnement RFID :



Les étiquettes radiofréquences, également appelées tags, sont constituées de deux éléments :

- Une puce électronique qui contient des données.
- Une antenne qui transmet ces données par ondes radio.

Les deux éléments sont assemblés dans un « package».

La plupart des puces ne dispose pas de source d'énergie propre, et c'est le signal émis par le lecteur qui permet l'échange des données contenues dans la puce via l'antenne. En captant les fréquences, la puce se réveille et émet son numéro d'identification en retour.
Ces étiquettes radiofréquences sont appeler "tags passives"

Néanmoins, il existe des étiquettes radiofréquence alimentées par leur propre source d'allonger la distance de lecture d'énergie qui permettent d'allonger la distance de lecture, d'associer des capteurs de température aux étiquettes ou d'ajouter dans la puce des informations sur le produit

2) Les transmissions d'informations :

Cette dernière se fait selon divers fréquences, notamment en hautes fréquences abrévié HF (13,56 MHZ) et en ultra hautes fréquences UHF (850-950 MHZ)

- **La famille HF** se retrouve dans les cartes à puce sans contact et les mobiles. Cette famille HF a pour nom générique NFC.
- **La famille UHF** se retrouve dans les entrepôts pour produits ou pour permettre la réalisation d'inventaires dans des entrepôts ou les rayonnages de vêtements. Cette famille UHF est régulièrement appelée RFID par les industriels.

3) Lecture et/ou écriture :

Il existe plusieurs catégories d'étiquettes radiofréquences dont on peut citer :

- Les étiquettes "lecteur seule" ont un numéro d'identification gravé dès la fabrication de la puce, ce dernier peut être lu mais pas modifiable
- Les étiquettes "écriture une fois et lecture multiple" dont l'utilisateur peut enregistrer un seul numéro d'identification unique lors de la premier utilisation, après on pourra uniquement lire les informations sur la carte
- Les étiquettes "lecture réécriture" intégrant des pages de mémoire en plus du code unique, qui permet d'écrire et de modifier des nouvelles données.

4) Et en plus :

A ces fonctions citées précédemment sont ajoutées des fonctionnalités supplémentaires dont on peut citer :

- Gestion des collisions : un lecteur peut dialoguer avec plusieurs étiquettes présentes simultanément dans le champ de son antenne.
- L'alarme antivol : l'étiquette détecte la présence d'une ou plusieurs étiquettes dans le champ de l'antenne.
- Le cryptage des données : pour la sécurisation des informations

5) Pourquoi la technologie RFID :

- Capacité de mise à jour du contenue
- Vitesse de marquage
- Sécurité d'accès au contenue
- Grande durée de vie
- Grande souplesse de positionnement
- Une meilleure protection aux conditions environnementales

Dans la chaine d'approvisionnement des produits, elle permet :

- D'améliorer l'automatisation des opérations de réception et d'expédition
- De réduire les pertes et les vols
- De faciliter l'inventaire des produits

Dans les magasins, elle permet :

- De réduire les ruptures de stock
- De mieux contrôler les dates de péremption des produits
- De faciliter en cas de nécessité, les opérations de retrait des produits
- D'accélérer le passage en caisse

Cette technologie permet, d'autre part, un suivi unitaire des produits, et par conséquent, une traçabilité plus fine.

II) Présentation de l'ORM Entity Framework :

Définition :

Entity Framework est un mappeur relationnel objet (ORM). Fondamentalement, il génère des objets métiers et entités selon les tables de base de données et fournit le mécanisme pour :
-Effectuer les opérations CRUD (Create, Read, Update, Delete) de base.
-Gérer facilement les relations "1 à 1", "1 à plusieurs" et "plusieurs à plusieurs".
-Possibilité d'avoir des relations d'héritage entre des entités.
Et c'est un ensemble de technologies dans ADO.NET qui prennent en charge le développement d'applications logicielles orientées données. Les architectes et les développeurs d'applications orientées données sont confrontés à la nécessité d'atteindre deux objectifs très différents. Ils doivent modeler les entités, les relations et la logique des problèmes liés à l'activité de l'entreprise qu'ils résolvent, et ils doivent également travailler avec les moteurs de données utilisés pour stocker et récupérer les données. Les données peuvent être réparties entre plusieurs systèmes de stockage, chacun ayant ses propres protocoles ; même les applications qui fonctionnent avec un seul système de stockage doivent équilibrer les besoins du système de stockage par rapport aux besoins en matière d'écriture d'un code d'application efficace et facile à gérer.

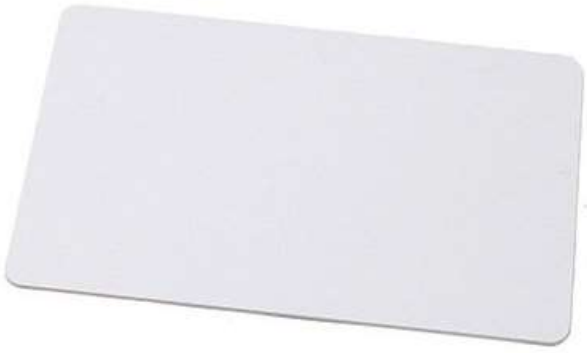
Entity Framework permet aux développeurs de travailler avec des données sous la forme de propriétés et d'objets spécifiques aux domaines, tels que des clients et des adresses de clients, sans qu'il soit nécessaire de se préoccuper des tables et des colonnes de base de données sous-jacentes dans lesquelles sont stockées ces données. Avec Entity Framework, les développeurs peuvent travailler à un niveau supérieur d'abstraction lorsqu'ils traitent les données, et peuvent créer et maintenir des applications orientées données avec moins de code que dans les applications traditionnelles. Étant donné qu'Entity Framework est un composant du .NET Framework, les applications Entity Framework peuvent s'exécuter sur tout ordinateur sur lequel .NET Framework 3.5 SP1 (ou une version ultérieure) est installé

Avantage :

Les avantages sont :
-Nous pouvons avoir toute logique d'accès aux données écrite dans des langages de niveau supérieurs.
-Le modèle conceptuel peut être représenté dans une meilleure façon en utilisant les relations entre entités.
-Le magasin de données sous-jacent peut être remplacé sans beaucoup de frais puisque toute logique d'accès aux données est présente à un niveau supérieur.

III) Analyse fonctionnelle

1) Caractéristique de la carte à puce (Mifare 1k) :



Cartes RFID 13,56Mhz Blanche
Normes ISO : 14443A
Epaisseur : 0,84mm.
Lecture/Ecriture
Puce : passive

La carte *MIFARE Classic 1k* offre 768 octets de stockage répartis sur 16 secteurs. Chaque secteur est composé de 3 blocs de 16 octets. Un bloc de sécurité supplémentaire vient protéger l'accès au secteur par deux clefs différentes (nommées A et B). Les secteurs peuvent être gérés via des opérations de lecture/d'écriture de données ou d'incrémentation/décrémentation de valeurs.

2) Caractéristique du lecteur/encodeur :

RS232 ou USB Interface

Comptabilité avec un système d'exploitation 32 bits

Fréquence 13,56 MHz RF exploitation

Protocoles ISO1443B, ISO15693 et ISO14443A

Dimension: 110 × 81 × 26 mm

Poids: 100g

3) Architecture logicielle :

a) IHM :

Visual studio 2012 avec c# comme language de programmation

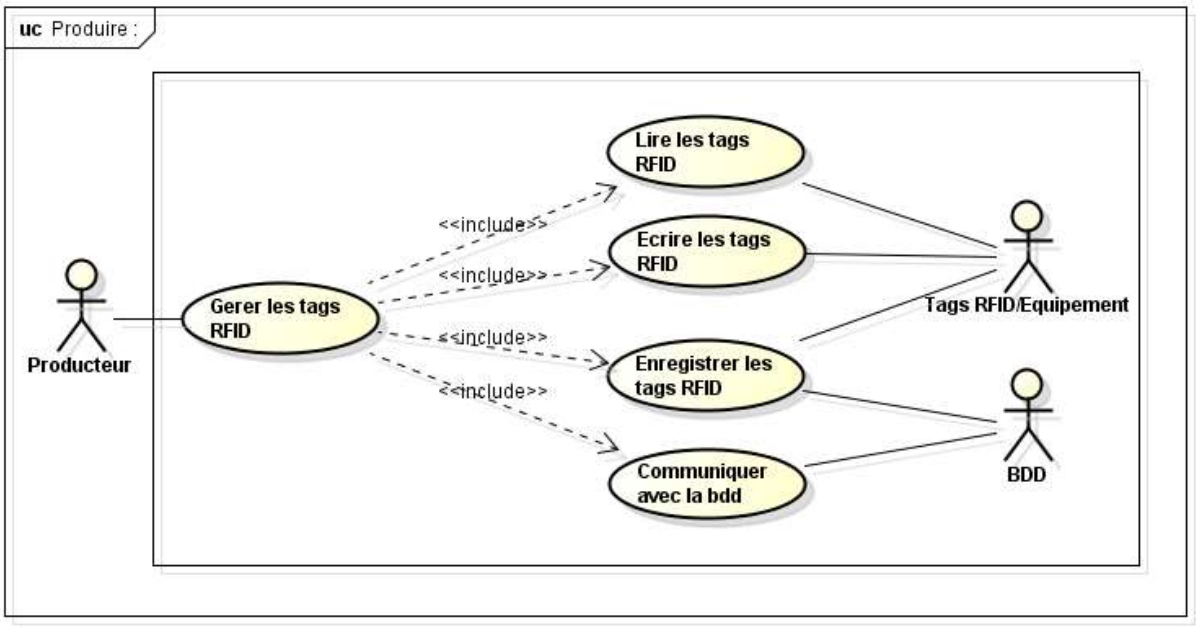
b) Base de données :

Avec l'ORM Entity Framework

IV) Prise en main du projet :

Use case :

Dans le cadre de notre projet, une puce est liée à une élingue, cette élingue appartient à un site qui appartient à une société qui souhaite pouvoir effectuer des contrôles. Au préalable les puces seront donc produites puis incorporées dans un équipement



Le producteur peut gérer les puces RFID, c'est-à-dire, lire et écrire sur des puces puis lier ces puces aux équipements, ou mettre à jours ses derniers.

Scénario :

Nominal :

1ère étape : Pour lire/écrire les tags, il faut connecter le lecteur au PC par port USB.

2ème étape : On doit se connecter au bon port ainsi qu'au bon baud, puis récupération de l'ID de la carte (après connexion).

3ème étape : On doit récupérer l'identifiant de la carte avant de lire et/ou écrire.

4ème étape : On peut écrire et/ou lire les tags RFID.

7ème étape : Les tags écrits sont également à enregistrés dans la base de données centrale.

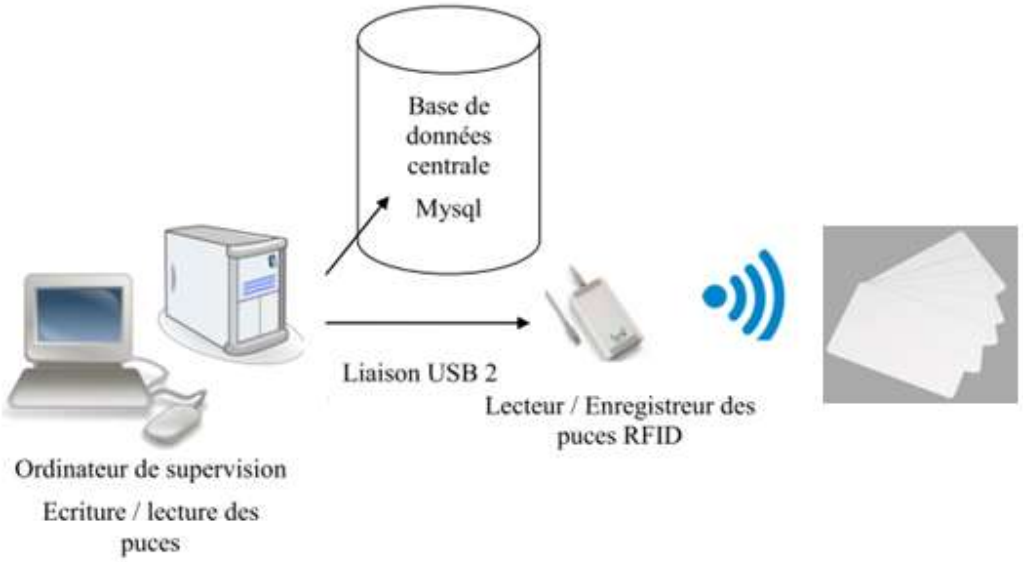
Alternatifs :

- Si la puce n'est pas présente sur la base de données, on pourra l'ajouter

D'exception :

- Affichage d'un message d'erreur si on n'arrive pas à se connecter au lecteur
- Affichage d'un message d'erreur s'il y a un problème de connexion avec la base de données
- Affichage d'une icône à côté de la zone de saisie dans l'application si les informations entrer ne sont pas complet

Architecture matérielle :



Un utilisateur va agir sur une puce grâce au lecteur et administrer celle-ci sur la base de données, la puce serra ensuite lier à une élingue.

Ce qui m'a été fournis :

- Un lecteur RFID.
- Une bibliothèque de liens dynamiques (dll).
- Une application de test.

Pour le projet :

Il m'a fallu me familiariser avec la technologie RFID, comprendre son fonctionnement puis travailler dessus.

Le lecteur nécessite un pilote pour le fonctionnement fournit avec l'application de test.

Pour pouvoir développer correctement il m'a fallu inclure une librairie à mon projet.

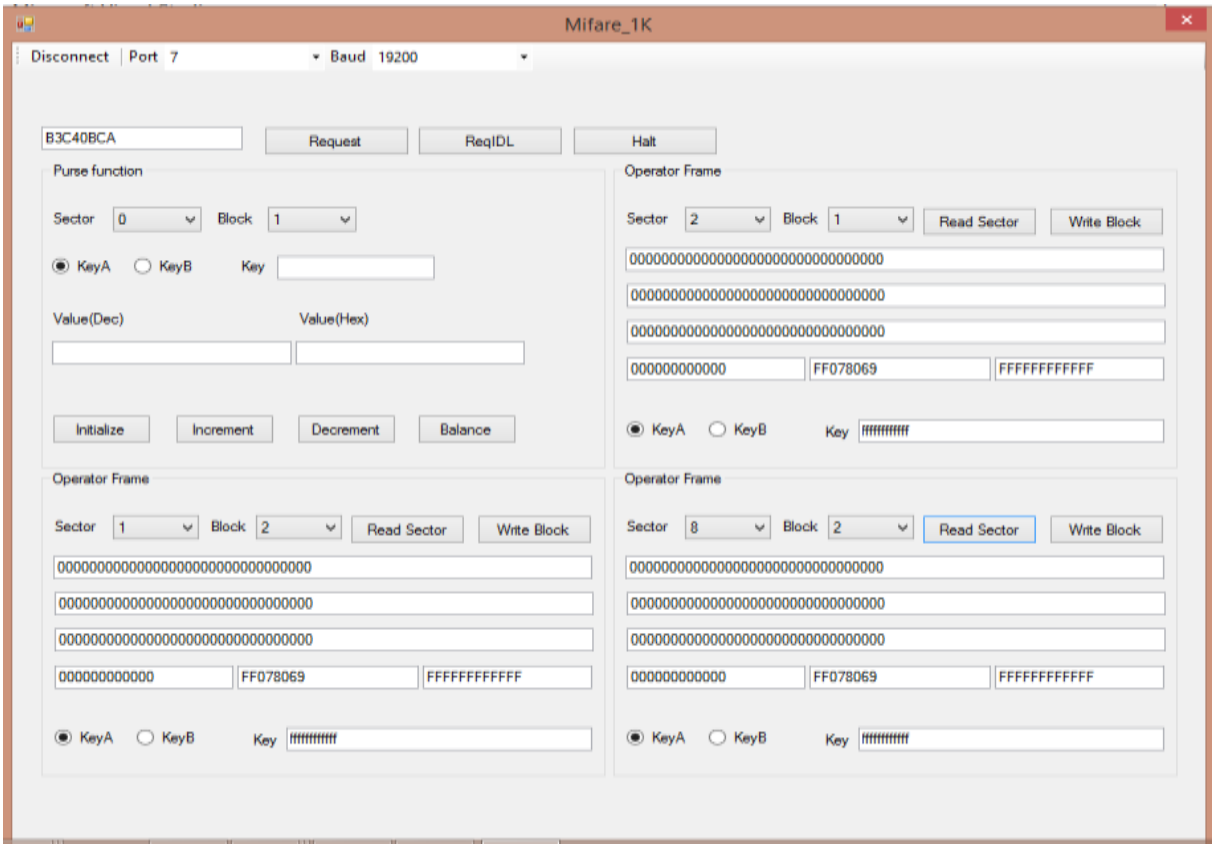
Conclusion :

Nous avons donc utilisé des puces sans contact que nous avons incorporé dans un équipement, l'utilisation de ses puces a été demandée dans le but de faciliter les contrôles ainsi que dans l'optique que celle-ci s'utilise beaucoup moins qu'une technologie avec contact.

V) Application :

Application de test fournis :

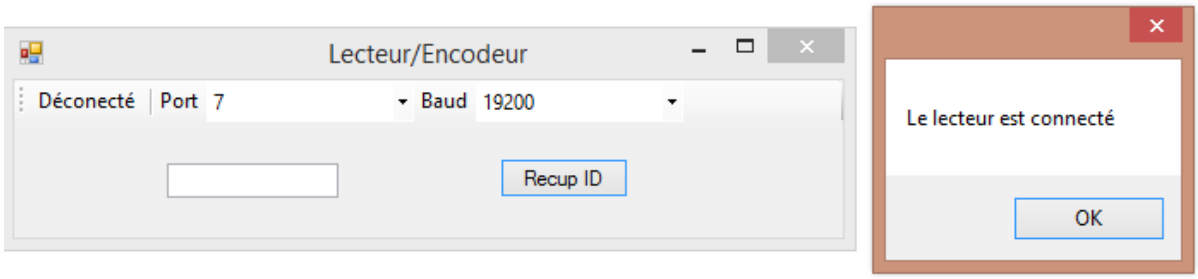
Qui m'a permis d'effectuer des tests sur les puces afin que de comprendre leurs mode de fonctionnement



Tests unitaires :

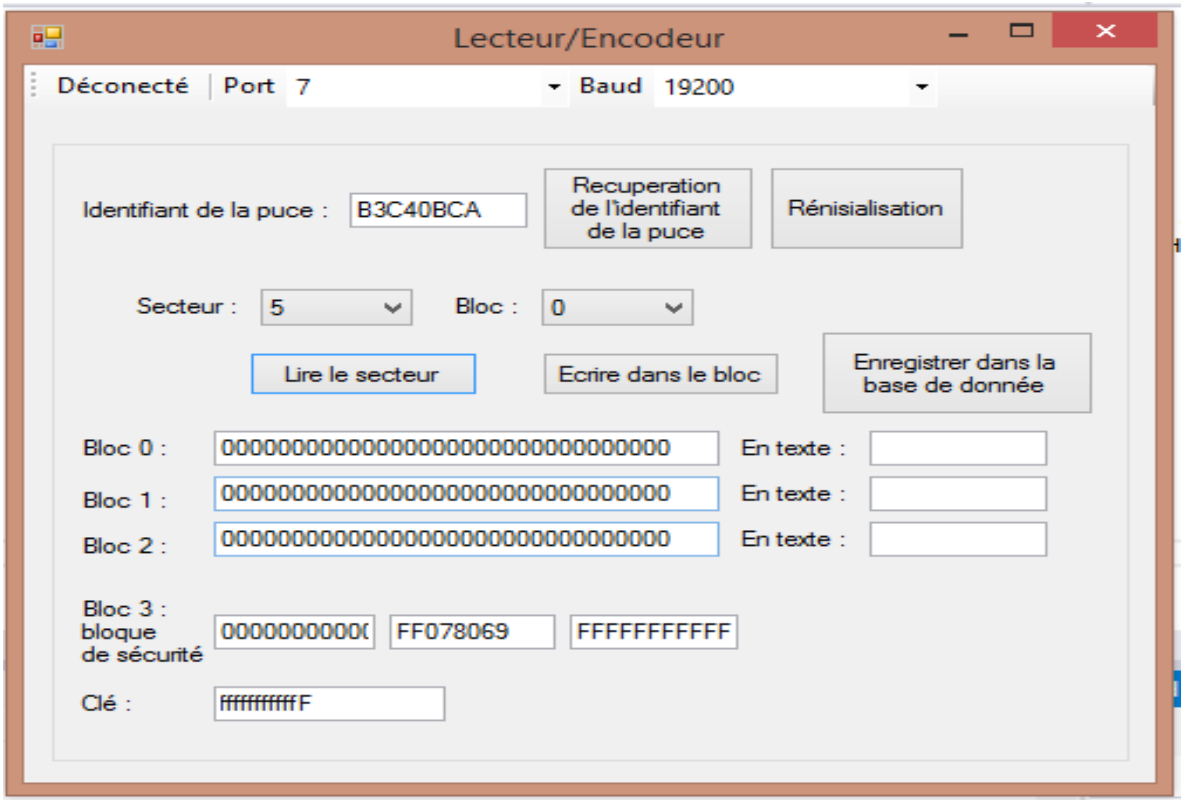
1er : Création d'une application de connexion au lecteur

Pour y arriver il m'a fallu utiliser la dll du lecteur



2eme : Amélioration en ajoutant l'écriture et la lecture d'une puce RFID

Pour y parvenir j'ai dû utiliser ainsi que de comprendre les fonctions de la dll, ainsi que les principes de conversion des caractères à enregistrer sur la puce



Pour réaliser un enregistrement dans une puce on ne peut pas enregistrer directement une chaine brute, il faut réaliser une conversion " caractère de la chaine en code ASCII hexadécimal des caractères décomposés ".

Dans le tableau ci-dessous on donne la conversion d'une chaine " toto " à enregistrer dans la puce :

Chaine	**t**	**o**	**t**	**o**
Code ASCII Hexadécimal de chaque caractère de la chaine	0x74	0x6F	0x74	0x6F
Caractères ASCII décomposés	0x7 0x4	0x6 0xF	0x7 0x4	0x6 0xF
Code ASCII des caractères décomposés en hexadécimal	0x37 0x34	0x36 0x46	0x37 0x34	0x36 0x46

Ecriture Puce

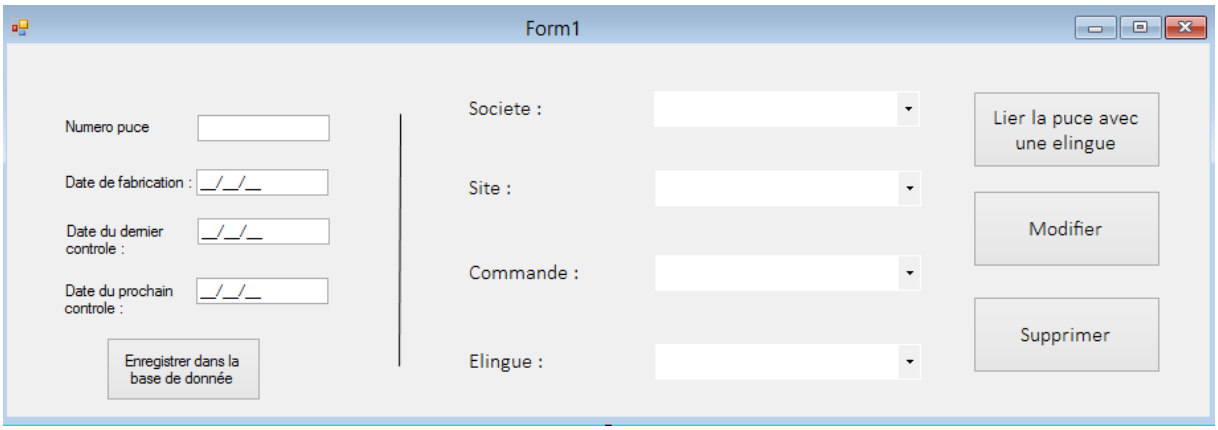
Lecture Puce

Chaque puce est divisée en 15 blocks de 00 à 15, chaque bloc peut contenir 1 octet donc 8 bits.

J'ai choisi de convertir les données à enregistrer en code ASCII hexadécimal car contrairement au code ASCII décimal et au code ASCII octal, il prend toujours 2 bits, donc moins de place.

3eme : BDD

Afin de pouvoir gérer la production, j'ai dû établir une connexion avec notre base de données centrale afin d'enregistrer et de consulter les données liées aux puces.



Notre base de données est constituée de XX tables (voir modèle ci-dessous)

----- mettre modele entity finalisé----

a) Communication avec la base de données :

```
controleelingue2015Entities bdd = new controleelingue2015Entities();
// c est la varibale qui sera assigné a ma table societe dans la base de données
var id_societe = (from c in bdd.societe
                  where c.RaisonSociale == comboBox_societe.Text
                  select c.Id).First();
```

Dans cet exemple, nous pouvons voir qu'une requête utilisant le modèle Entity s'exécute dans cet l'ordre suivant :

from → where → puis select

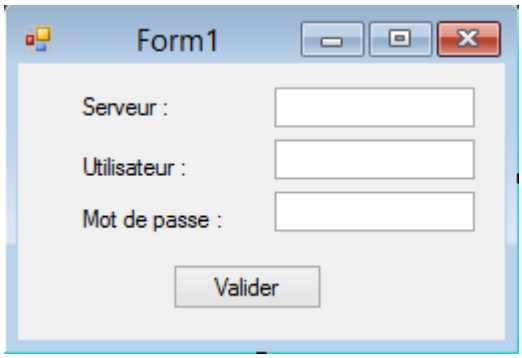
Dans ce cas, c'est pour vérifier si le nom de la société correspond bien au champ que ce dernier a inséré précédemment dans la comboBox et aux même données dans la table société puis sélectionne le premier ID.

b) Envoi de données aux tables :

```
// creation d'un nouveau objet qui sera instancier dans la classe
//-----classe----------------------------------------------constructeur de la classe-------
using (controleelingue2015Entities Controle_Helingue = new controleelingue2015Entities())
{
    // création d'une nouvelle table
    puce_rfid ma_puce_rfid = new puce_rfid
    {
        Numero_Puce = (tb_Numero_puce.Text),
        Date_Controle_Production = Convert.ToDateTime(mtb_date_fabrication.Text)
    };
    // ajout du groupe à l'entité
    Controle_Helingue.puce_rfid.Add(ma_puce_rfid);
    // sauvgrade de données dans la BDD
    Controle_Helingue.SaveChanges();
}
```

Pour envoyer des données vers une table avec Entity Framework

4eme : Changer la connexion de la base de données (connexion à distance à un autre serveur) :



Pour y arriver, il faut suivre tout d'abords les étapes du tutoriel "connexion à distance à la base de données" dans annexe.

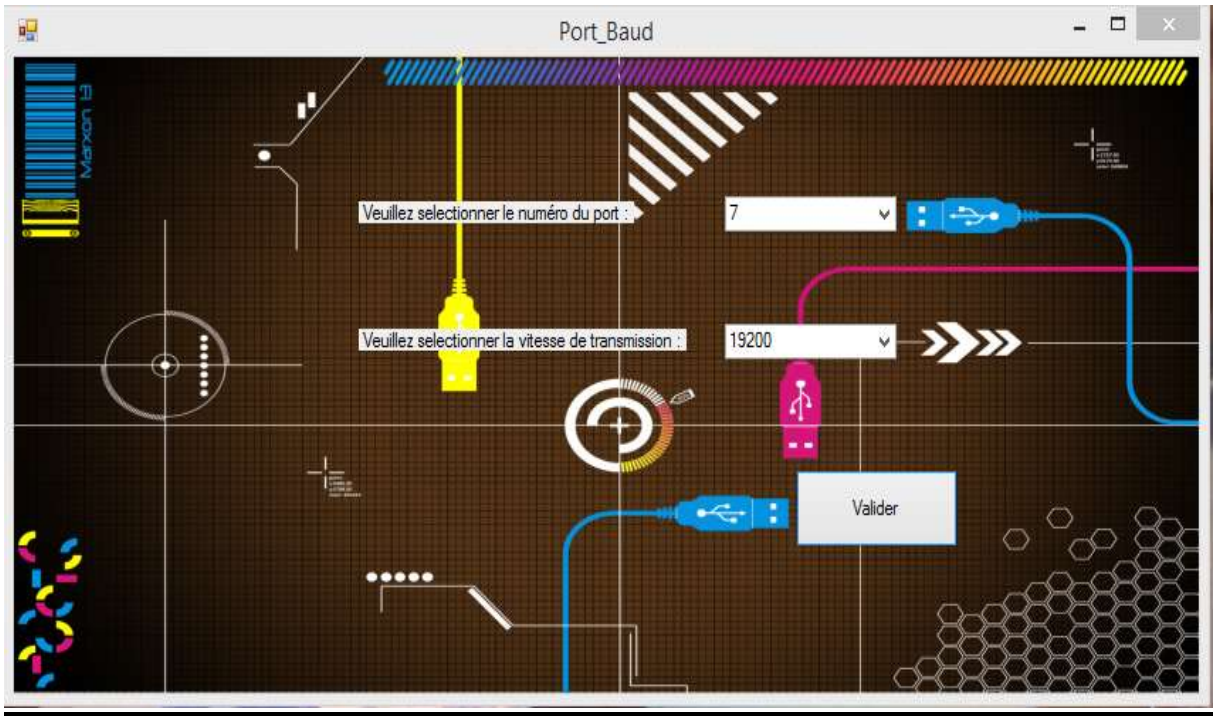
VI) Application réaliser :

Langage et logiciel de développement :

Afin de développer cette partie j'ai utilisé l'IDE (Integrated Development Environment) Visual Studio avec comme langage le C#.

Le projet développé en C# sur Visual Studio permet un développement managé par Microsoft le .NET qui offre la possibilité d'utiliser des objets crées par Microsoft et de développer des applications WinForm (application graphique).

Interface :



Fenêtre qui s'affiche pour se connecter au lecteur, on peut :

- Entrer le numéro du port
- Entrer la vitesse de transmission
- Une fois valider, vérification de la connexion à la base de données

Si la connexion est réussie :

Cette fenêtre nous permet de:

- Récupérer l’identifiant de la puce
- Lire une puce
- Récupérer les informations liées à la puce
- Ecrire sur la puce
- Enregistrer une puce dans la BDD
- Lier sur la BDD une puce à un client qui sera une société
- Mettre à jour les informations de la puce sur la BDD
- Supprimer----------
- Rafraîchir la connexion entre le lecteur et la puce
- Et changer la connexion à la base de données si on souhaite changer de serveur dans le menu options

Librairie utilisée :

Il m’a fallu utiliser la librairie "MasterRD.dll" afin de permettre la connexion ainsi qu’utiliser les différentes fonctions nécessaires au lecteur RFID (Explication de l’installation en annexe).

Il faut néanmoins comprendre leur fonctionnement c'est-à-dire leurs paramètres et les valeurs de retour

Gestions des erreurs :

Pour ma gestion des erreurs j’ai donc :

- Une fenêtre qui s’affiche quand y’a un problème de connexion au lecteur avec un bouton recommencer qui nous renvoie à notre fenêtre de connexion au lecteur afin de rectifier les informations rentrer et un bouton annuler pour quitter

- et une fenêtre qui s'affiche quand y'a un problème de connexion à la base de données avec un bouton recommencer pour ressayer une fois avoir vérifier la connexion au serveur et un bouton annuler pour quitter

- Un algorithme qui vérifie les informations liées à la puce sur la BDD
- Gestion des erreurs lors de l'enregistrement des informations sur la puce ou sur la base de données grâce au composant ErrorProvider

VII) Les problèmes rencontré

Avec la DLL :

Impossible d'installer la dll en référence sur visual studio que j'ai finis par mettre dans le dossier "debug" de mon application pour que sa marche

```
[DllImport("MasterRD.dll")]
      public static extern int rf_init_com(int port, int baud);
```

L'utilisation de "Dllimport" permet d'utiliser la fonction rf_init_com(int port, int baud) qui est en 32bits et non en 64 bits
Il faut donc créer une classe qui va récupérer toutes les fonction.

VIII) La démonstration pour le jour de la revue

- La communication avec le lecteur/enregistreur
- On lit/écrit dans une puce sous forme de chaîne de caractères brute
- Communication de l'application avec la base de données en utilisant l'ORM Entity Framework.
- Effectuer des opérations CRUD (Create, Read, Update, Delete)
- Changer la connexion string de la base de données

IX) Conclusion :

Ce projet a nécessité une certaine rigueur du faite que celui-ci soit divisé plusieurs parties indépendantes, puis rattachées les unes aux autres par la suite pour former un tout cohérent.

La découverte de la technologie RFID m'a permis de me rendre compte des problèmes récurrents des développeurs ainsi que d'apprécier plus encore la façon d'appréhender le code.

X) Annexe :

Changement connexion string :

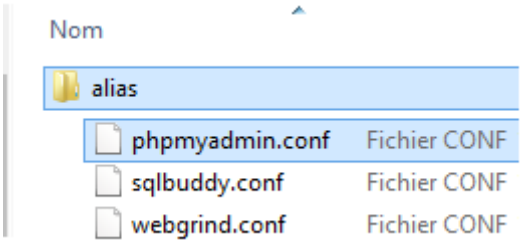
Message d'erreur lors du premier essaye :

Sans configuration, Wamp interdit tous les accès en provenance d'un autre poste que celui sur lequel il est installé.

Donc pour rendre phpmyadmin accessible depuis une machine extérieure, il faut éditer le fichier de configuration "phpmyadmin.conf". Pour modifier ce fichier veuillez suivre les étapes ci-dessous :

1- Dans le disque local sélectionner Wamp

2- Puis sélectionner alias pour pouvoir modifier de fichier de configuration "phpmyadmin.conf"



3- Une fois le fichier éditer, on doit remplacer :

# Les lignes :	# Par :
Order Deny,Allow Deny from all Allow from xxx.xxx.xxx.xxx	Order Allow,Deny Allow from all

Puis redémarrer Wamp.