

# RAPPORT DE PROJET : GESTION ACCES

---

HAMOUTEN Yassine, ET-TOUIL Abderrahmane, BAUDELET Maxence

## Table des matières

INTRODUCTION .....	4
REPARTITION DES TACHES .....	5
PLANNING PREVISIONNEL.....	6
SEMAINIER .....	7
BASE DE DONNEES .....	10
ANALYSE UML .....	12
1. Diagramme de déploiement.....	12
2. Diagramme de cas d'utilisation .....	13
3. Diagramme d'activité.....	15
4. Application BECK DK40 (HAMOUTEN Yassine) .....	16
Diagramme de classe .....	16
Diagramme de séquence .....	17
APPLICATION EMBARQUÉE BECK DK40 (YASSINE HAMOUTEN) .....	18
1. Présentation du module BECK DK40 .....	19
Le BIOS (Basic Input Output System) .....	21
Le système d'exploitation RTOS (Real Time Operating System) .....	22
Le port série.....	22
Le port Ethernet .....	23
2. Configuration du module BECK DK40 .....	25
3. Le matériel utilisé .....	28
Lecteur RFID .....	28
Serrure électrique .....	28
4. Environnement de travail .....	29
5. Environnement de développement.....	29
6. Tests de mise en œuvre.....	32
Test simple d'application sur le module BECK DK40 .....	32
Test des Entrées / Sorties du module BECK DK40 .....	32
Test du lecteur RFID .....	33
Test de connexion au serveur .....	35
7. Recettes .....	37

Module d'identification RFID .....	37
Module Entrées / Sortie .....	38
Module Client .....	39
Application complète .....	40
8. Tests unitaires et tests d'intégrations .....	42
9. Problèmes rencontrés / solutions .....	46
10. Synthèse .....	47
Base de données - Client lourd et serveur TCP (Maxence Baudalet) .....	48
1. Création de la base de données .....	49
Présentation MYSQL .....	50
PHPMYADMIN .....	50
Présentation du serveur WAMP .....	53
2. Application Utilisateur .....	56
Qu'est-ce qu'une couche DAL ? .....	57
Classe DAL Base .....	59
Comment modifier la base de données à partir de l'application ? .....	62
3. Serveur TCP en C# .....	64
4. Test unitaire .....	66
5. Démonstration de l'application .....	69
6. Conclusion .....	71
Client lourd (Abderrahmane Et-touil) .....	72
1. Outils de développement .....	73
EasyPHP .....	73
Visual studio .....	73
CSHARP(C#) .....	74
2. Gestion d'administrateur .....	75
3. Gestion des utilisateurs .....	77
4. Accès aux couches de données .....	81
Présentation des classes DAL .....	81
5. Objet de transfert de donnée .....	84
Présentation des classes DTO .....	84

6. Problématique .....	85
7. Tests .....	88
DOSSIER DE MAINTENANCE.....	94
INSTALLATION .....	95
NOTICE D'UTILISATION .....	96
CONCLUSION.....	97
ANNEXES .....	98
1. Code source de l'application embarquée (Yassine Hamouten) .....	98
Le main : .....	98
La classe rs232 :.....	100
La classe rfidgrove125 :.....	100
La classe pio :.....	101
La classe serrure :.....	102
La classe client :.....	102
La classe becdk40 : .....	103

## INTRODUCTION

Le but du projet consiste à développer une solution pour gérer les accès libres des étudiants, apprentis ainsi que le personnel, aux salles de foyer et de TP. Il faut pouvoir contrôler les accès à ces salles. L'accès se fera avec un badge RFID.

Les horaires d'accès, la création, modification, mise à jour et la suppression des utilisateurs seront modifiable par un administrateur via un client lourd. Un historique des accès sera enregistré en base de données et consultable par l'administrateur.

Le personnel (enseignant, entretien) pourra également accéder aux salles mais ils n'auront aucunes restrictions.

On impose pour l'accès à la base de données de passer par une couche logicielle type DAL (Data Access Layer).

Les serrures des salles seront pilotées par un calculateur embarqué, le module BECK DK40. Le calculateur embarqué passera par un serveur en C# pour accéder à la base de données.

La solution est guidée par souci d'économie, les moyens qui seront alloués pour la mise en place de ce service seront faibles.

On ne demande pas de gérer dans cette étude les dysfonctionnements possibles sur le système embarqué (panne de la gâche, panne du lecteur RFID, etc..) ou sur l'accès au réseau.

## REPARTITION DES TACHES

### Etudiant 1 HAMOUTEN Yassine

- Prise en main BECK
- Mise en œuvre E/S TOR avec API spécifique
- Mise en œuvre lecteur RFID
- Mise en œuvre d'un client C++ pour accès base de données
- Application embarquée de contrôle d'accès

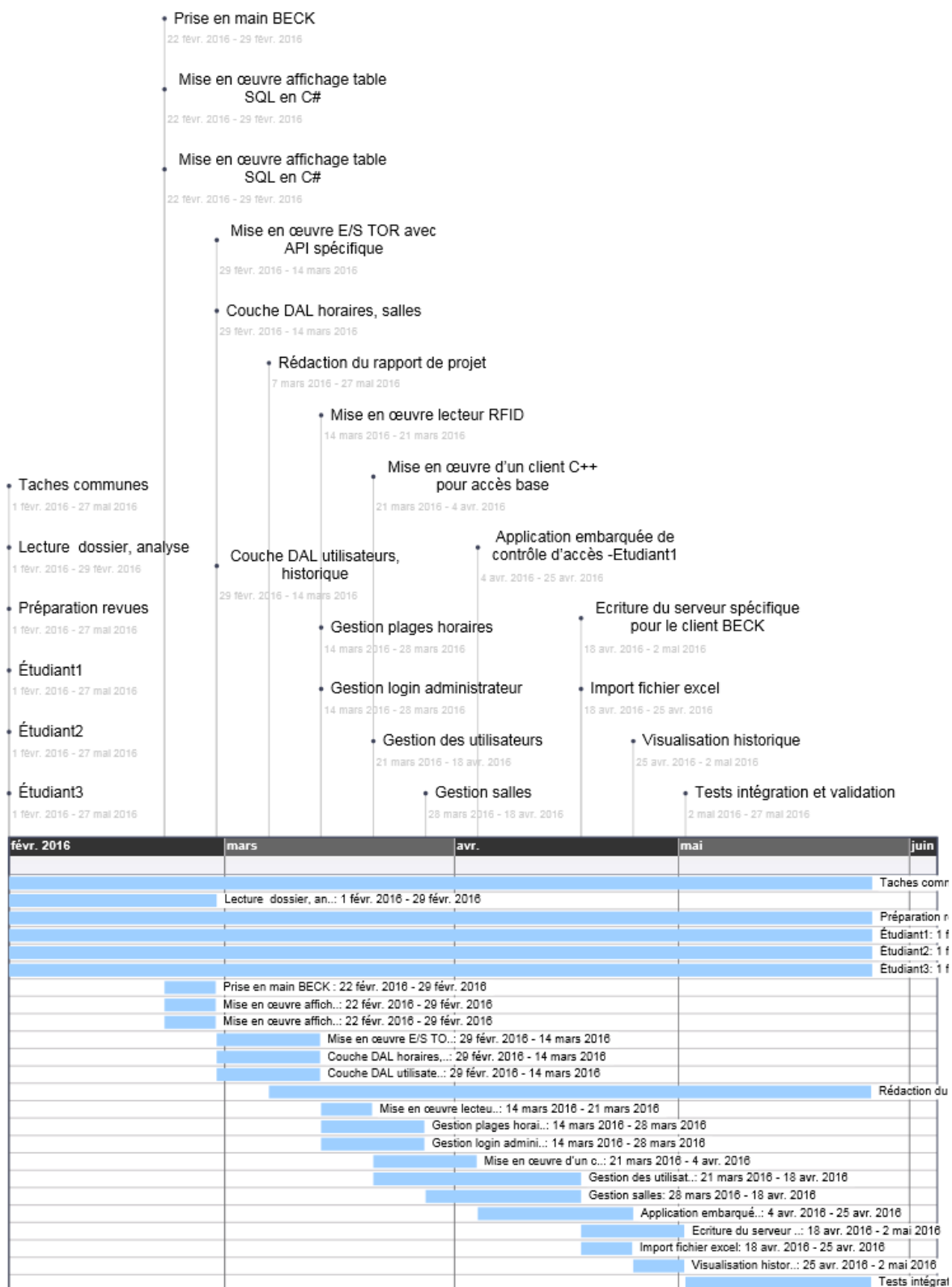
### Etudiant 2 BAUDELET Maxence

- Mise en œuvre affichage table SQL en C#
- Couche DAL horaires, salles
- Gestion plages horaires
- Gestion salles
- Ecriture du serveur spécifique pour le client BECK
- Installation, paramétrage des serveurs sur machine physique

### Etudiant 3 ET-TOUIL Abderrahmane

- Mise en œuvre affichage table SQL en C#
- Couche DAL utilisateurs, historique
- Gestion login administrateur
- Gestion des utilisateurs
- Import fichier Excel
- Visualisation historique

## PLANNING PREVISIONNEL



## SEMAINIER

Semaine 1 du 4 au 8 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>• Etude du projet</li> <li>• Mise en œuvre de la BECK</li> <li>• Installation chaîne de développement</li> <li>• Configuration de la BECK</li> <li>• Mise en œuvre d'une machine virtuelle</li> </ul>
Maxence Baudalet	<ul style="list-style-type: none"> <li>• Etude du projet</li> <li>• Elaboration de la base de données</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>• Etude du projet</li> <li>• Gestion login administrateur</li> <li>• Installation d'un connecteur MYSQL</li> </ul>

Semaine 2 du 11 au 14 janvier 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>• Recherche de librairies pour BECK DK40</li> <li>• BTS Blanc</li> </ul>
Maxence Baudalet	<ul style="list-style-type: none"> <li>• Début du code gestion d'accès</li> <li>• BTS Blanc</li> <li>• Avancement sur la partie gestion</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>• Avancement gestion login administrateur</li> <li>• BTS Blanc</li> <li>• Installation de WAMP</li> <li>• Ajout des DLL dans la librairie</li> </ul>

Semaine 3 du 18 au 21 janvier 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>• Librairie BECK DK40 fonctionnelle trouvée en pièce jointe forum</li> <li>• Test de la librairie</li> <li>• Création de la classe beckdk40 pour initialiser la beck + le port COM</li> </ul>
Maxence Baudalet	<ul style="list-style-type: none"> <li>• Début développement des classes DTO</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>• Remise au propre du code</li> <li>• Création classe DAO</li> </ul>



Semaine 4 du 25 au 28 janvier 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>Finalisation de la classe becdk40</li> </ul>
Maxence Baudalet	<ul style="list-style-type: none"> <li>Couche DAL</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>Couche DAL historique</li> </ul>

Semaine 5 du 1 au 4 février 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>Création d'un client de test pour communiquer avec le serveur</li> <li>Le client arriver à se connecter à un serveur de test que j'ai fait en C#</li> </ul>
Maxence Baudalet	<ul style="list-style-type: none"> <li>Création d'un serveur TCP en C#</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>Avancement couche DAL</li> </ul>

Semaine 6 du 22 au 25 février 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>Le client arrive à communiquer avec le serveur</li> <li>Communication entre le lecteur RFID et l'application fonctionne</li> <li>Test des E/S avec le relais</li> <li>Mise au propre des classes</li> </ul>
Maxence Baudalet	<ul style="list-style-type: none"> <li>DAL</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>DAL</li> </ul>

Semaine 7 du 29 février au 3 mars 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>Mise au propre des classes</li> </ul>
Maxence Baudalet	<ul style="list-style-type: none"> <li>Séparation couche DAL et DTO</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>Finalisation DAL</li> </ul>

Semaine 8 du 7 au 10 mars 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>Ma partie est terminée</li> <li>Développement application web en PHP avec le framework CodeIgniter</li> </ul>
Maxence Baudalet	<ul style="list-style-type: none"> <li>Gestion salles</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>Amélioration DAL</li> </ul>

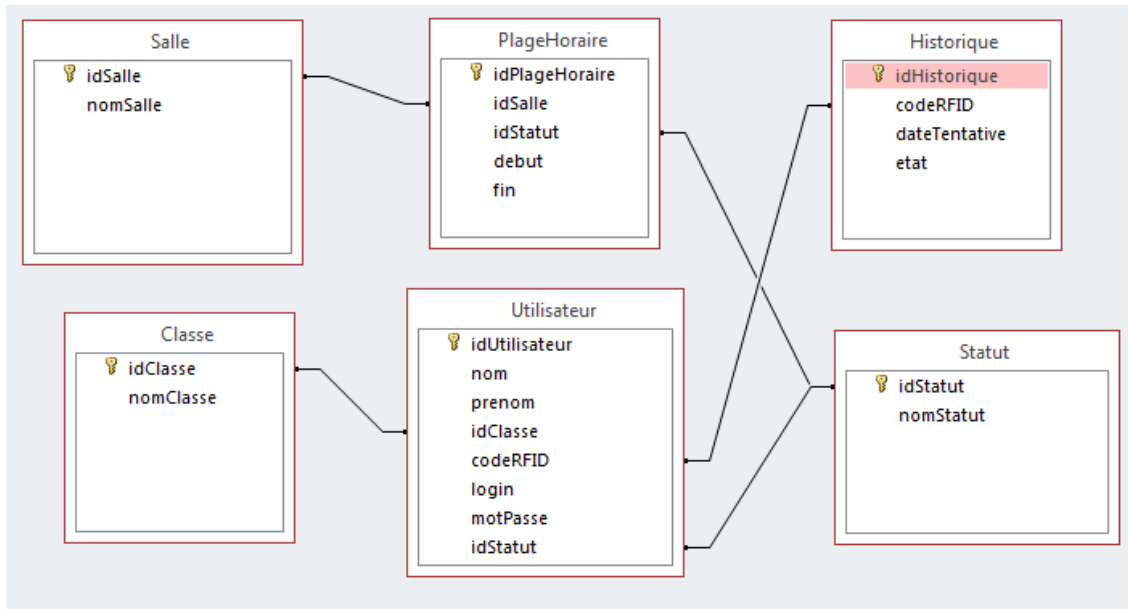
Semaine 9 du 14 au 17 mars 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>• DEV application web</li> </ul>
Maxence Baudelet	<ul style="list-style-type: none"> <li>• Gestion salles</li> <li>• Préparation de revue</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>• Préparation de revue</li> </ul>

Semaine 10 du 21 au 26 mars 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>• DEV application web</li> <li>• Finalisation de la partie historique (application web)</li> </ul>
Maxence Baudelet	<ul style="list-style-type: none"> <li>• Gestion salles</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>• Gestion historique</li> </ul>

Semaine 11 du 28 mars au 2 avril 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>• DEV application web</li> <li>• Partie gestion utilisateurs (application web)</li> </ul>
Maxence Baudelet	<ul style="list-style-type: none"> <li>• CRUD</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>• Rapport de projet</li> </ul>

Semaine 12 du 18 au 23 avril 2016	
Yassine Hamouten	<ul style="list-style-type: none"> <li>• Rapport de projet</li> </ul>
Maxence Baudelet	<ul style="list-style-type: none"> <li>• Rapport de projet</li> </ul>
Abderrahmane Et-touil	<ul style="list-style-type: none"> <li>• Rapport de projet</li> </ul>

## BASE DE DONNEES

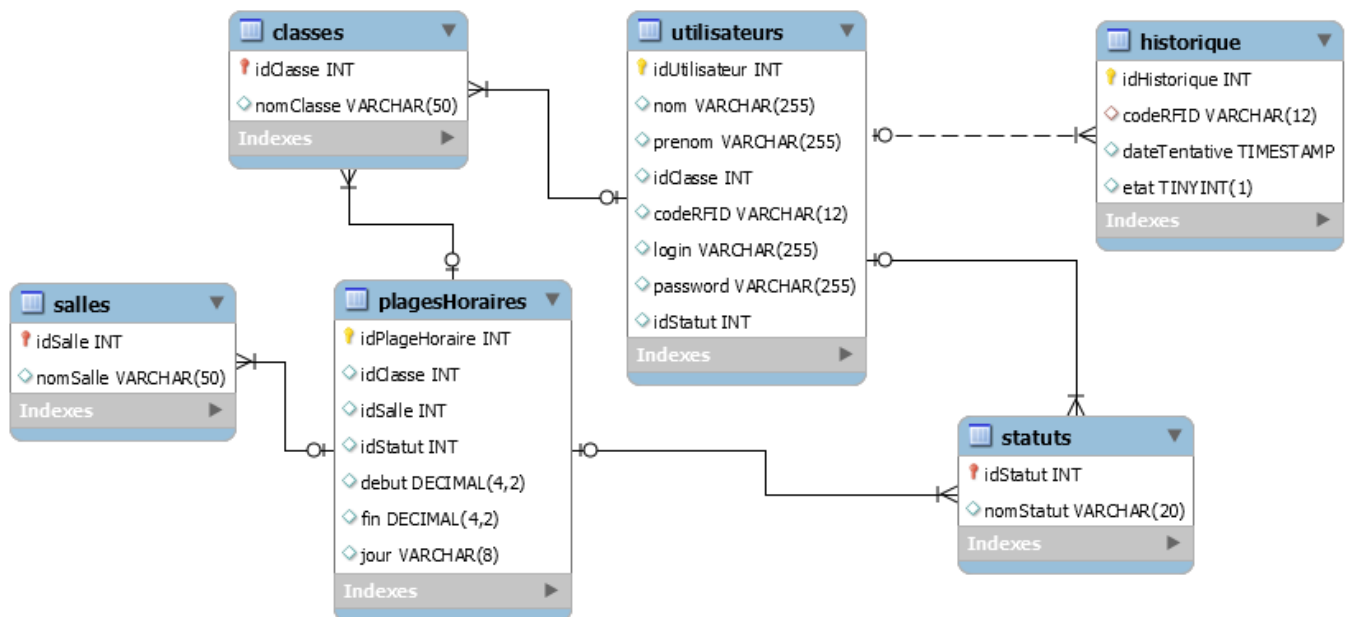


La base de données nous permet de stocker toutes les informations nécessaires à la réussite de ce projet comme par exemple la liste des utilisateurs. MySQL nous a été imposé comme SGBD (Système de Gestion de Base de Données).

La base de données comporte 6 tables :

- Salle : La liste des noms des salles gérées par le système.
- PlageHoraire : La liste de tous les créneaux horaires pour accéder aux salles avec pour attributs le début et la fin.
- Historique : liste toutes les tentatives d'accès aux salles qu'elles soient réussies ou échouées, avec la date de la tentative et l'état (réussite ou échec).
- Statut : Statut des utilisateurs (étudiant ou personnel).
- Utilisateurs : liste des utilisateurs avec pour attributs le nom, le prénom, l'idClasse, l'idStatut, le login et le mot de passe. En sachant que le login et le mot de passe ne servira que pour l'administrateur qui va lui permettre de se connecter sur le client. L'idClasse ne sera utilisé que pour les étudiants.
- Classes : la liste des classes des étudiants.

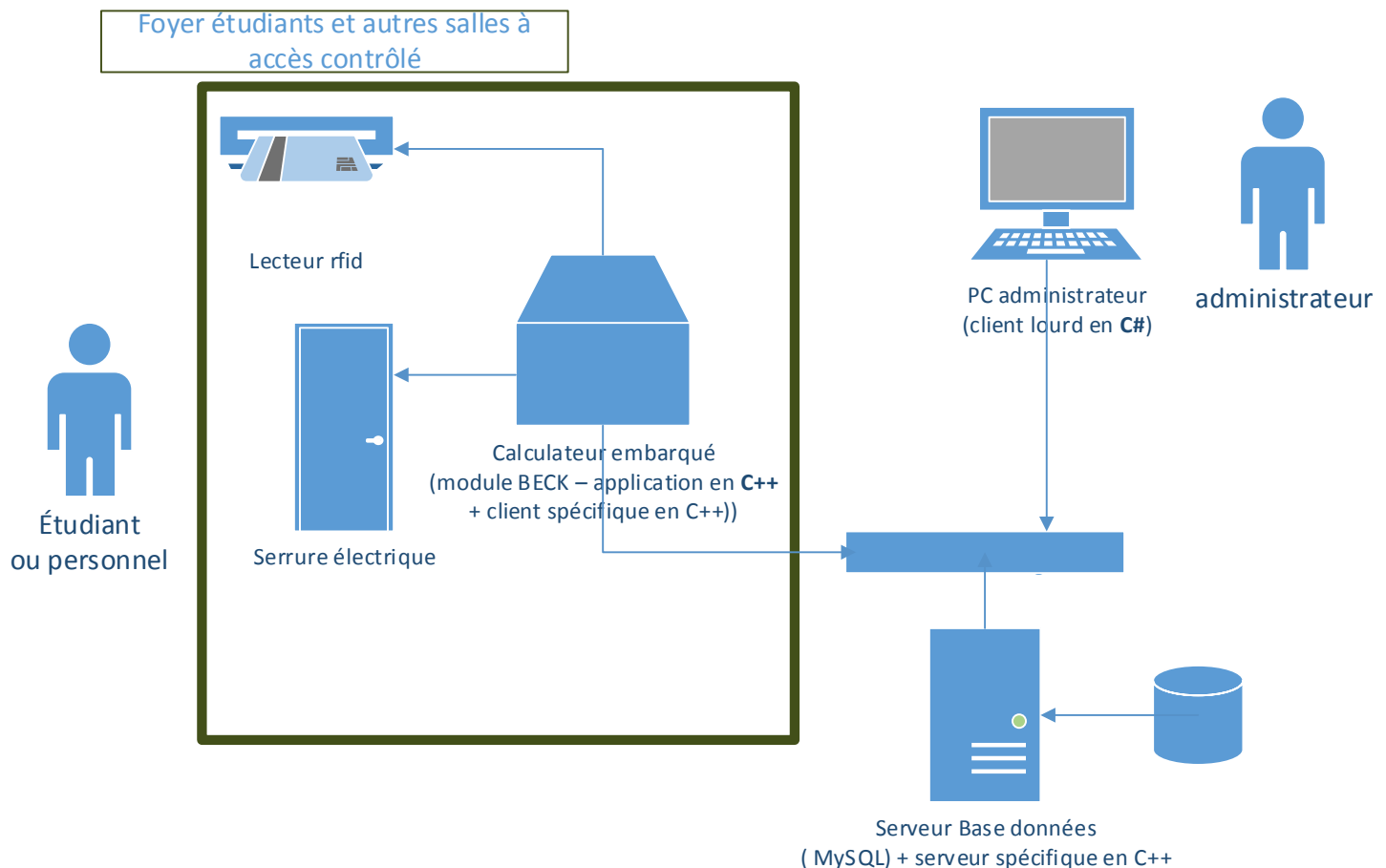
Nous avons légèrement modifié la base de données :



On peut voir que dans la table plagesHoraires nous avons ajouté le champ jour pour permettre une flexibilité niveau créneaux horaires.

## ANALYSE UML

### 1. Diagramme de déploiement



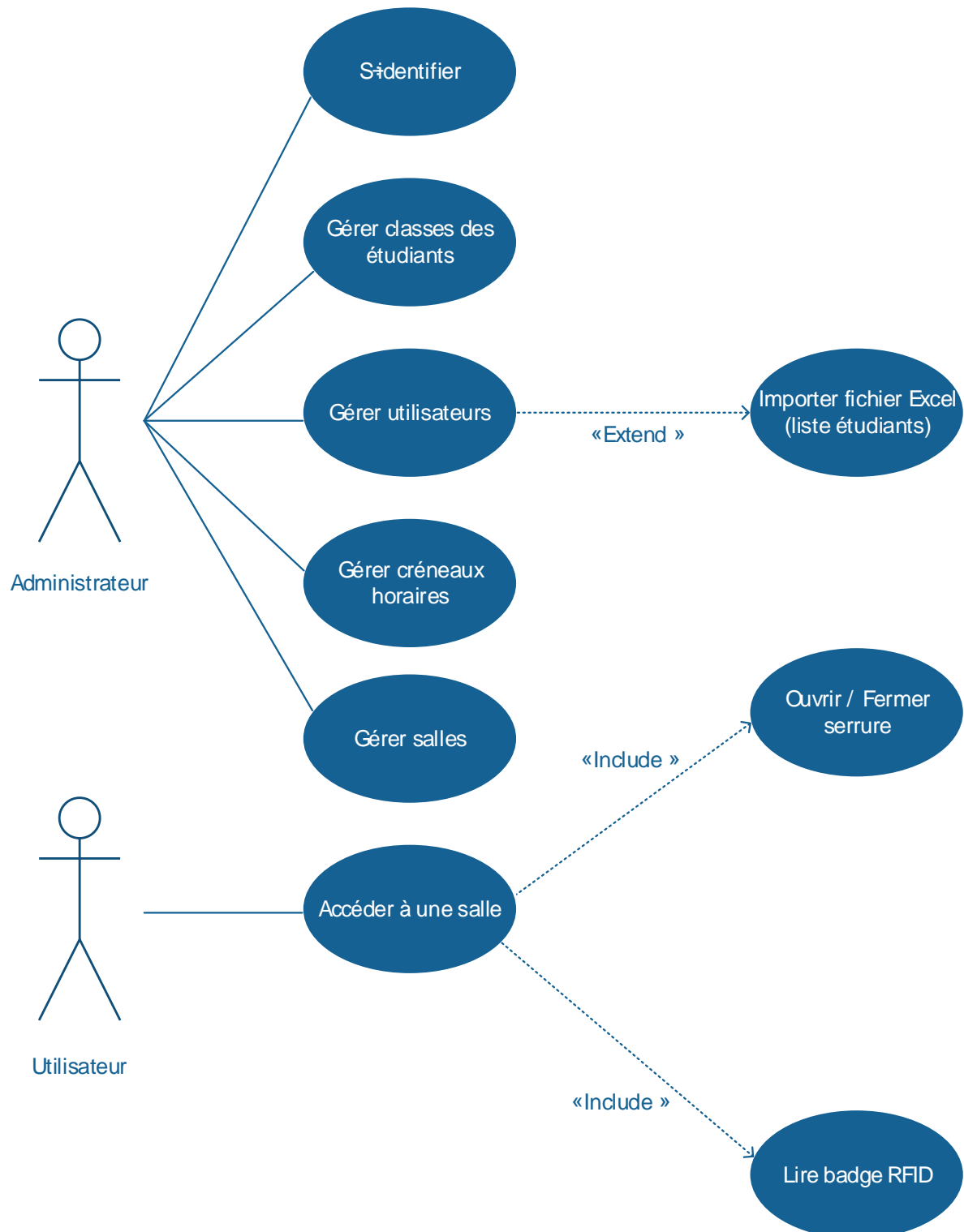
La technologie RFID est une méthode pour mémoriser et récupérer des données à distance en utilisant des marqueurs appelés « radio-étiquettes ». Ces radio-étiquettes sont des petits objets qui peuvent être incorporés ou collés, ici on va s'en servir sur un badge (ou carte) RFID. Quand le badge va passer devant le lecteur, celui-ci va envoyer un signal à l'étiquette. L'étiquette retourne alors son identification numérique (Code RFID).

Pour que l'utilisateur puisse accéder à l'une des salles, il va passer sa carte devant le lecteur, le calculateur embarqué (BECK DK40) va récupérer le code RFID et va envoyer ce code au serveur.

Le serveur va ensuite interroger la base de données (Base de données MySQL). Ainsi on va pouvoir vérifier que l'utilisateur correspondant à ce code RFID existe réellement, et que l'utilisateur essaie d'accéder à la salle dans la bonne plage horaire par rapport à son statut (Etudiant, Personnel).

Un administrateur va pouvoir se connecter sur le client lourd (Application C#) pour gérer les utilisateurs pouvant accéder aux salles et visualiser l'historique.

## 2. Diagramme de cas d'utilisation



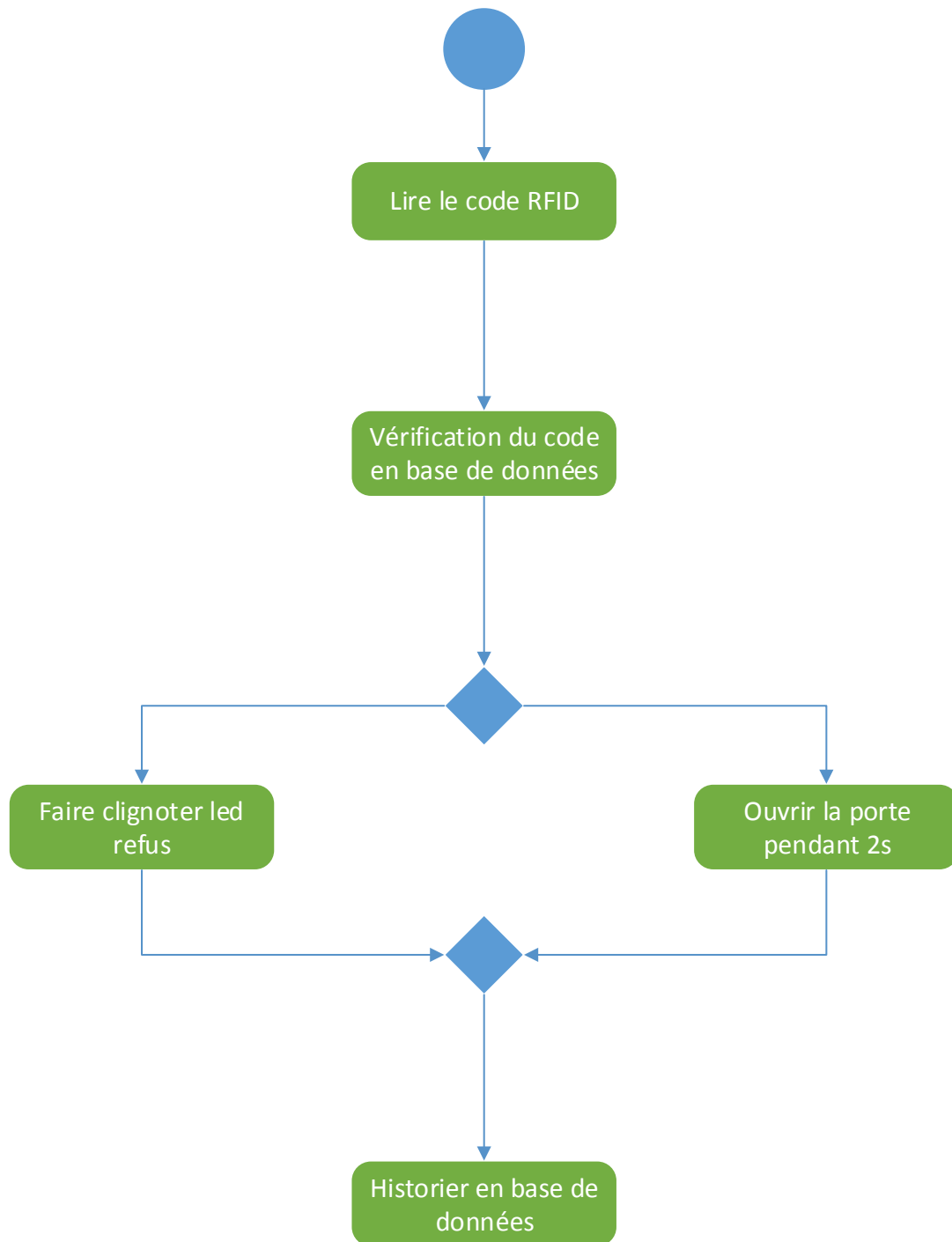
L'administrateur a la possibilité d'effectuer plusieurs actions depuis le client. Il est le seul à pouvoir s'identifier à l'application grâce à des identifiants stockés en base de données. Il peut effectuer des opérations CRUD (Create, Read, Update, Delete) sur les utilisateurs (étudiants et personnels du lycée) ainsi que sur les plages horaires en fonction du statut de la personne.

Pour finir, il peut consulter l'historique des salles. L'historique est constitué de toutes les tentatives d'accès aux salles (échouées ou réussies).

Un étudiant doit passer sa carte RFID devant le lecteur, si le créneau horaire le permet, il peut accéder à la salle.

Le personnel de l'établissement n'a pas de restrictions, lorsqu'il passe sa carte RFID devant le lecteur, la porte s'ouvre automatiquement.

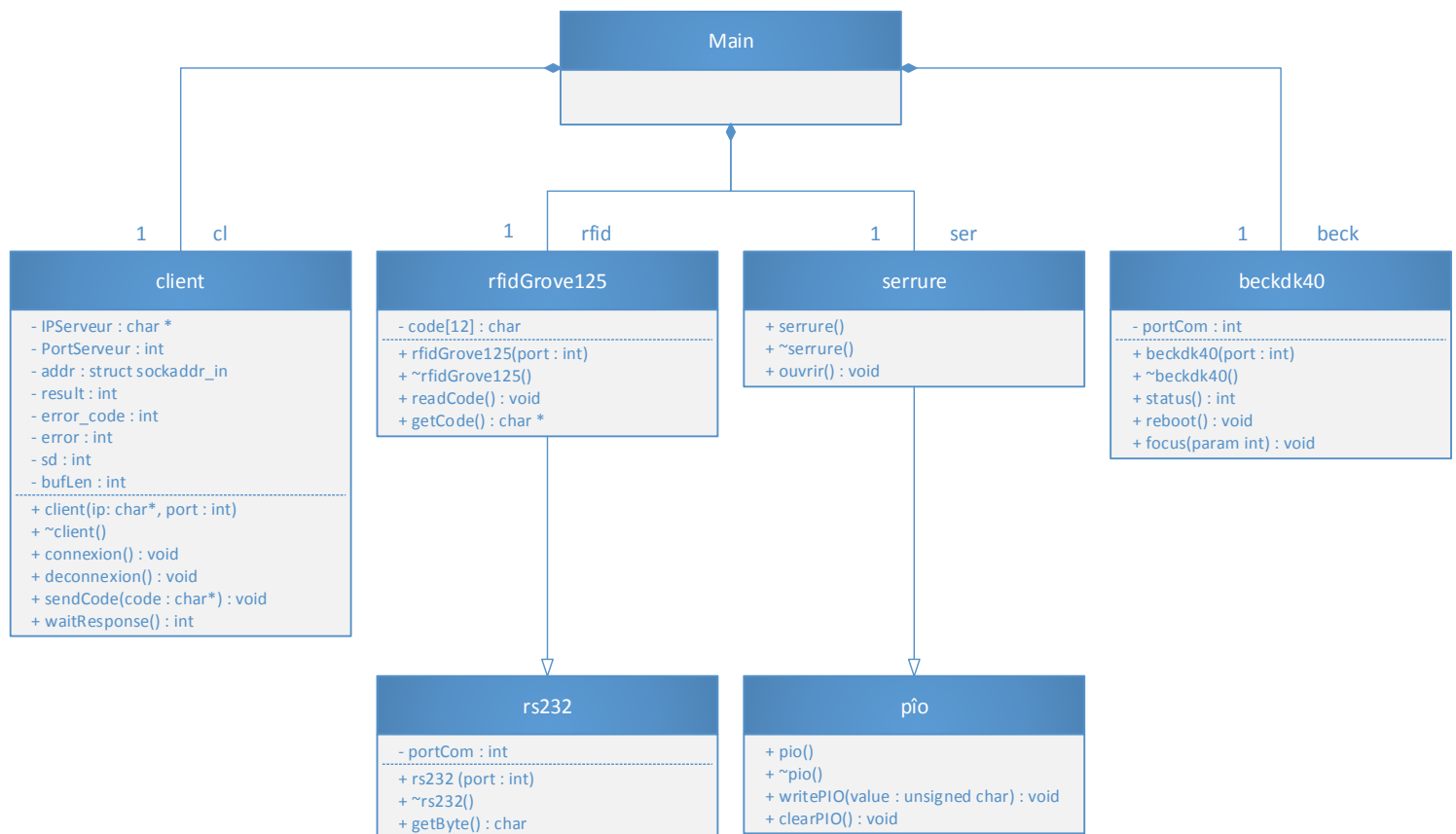
### 3. Diagramme d'activité



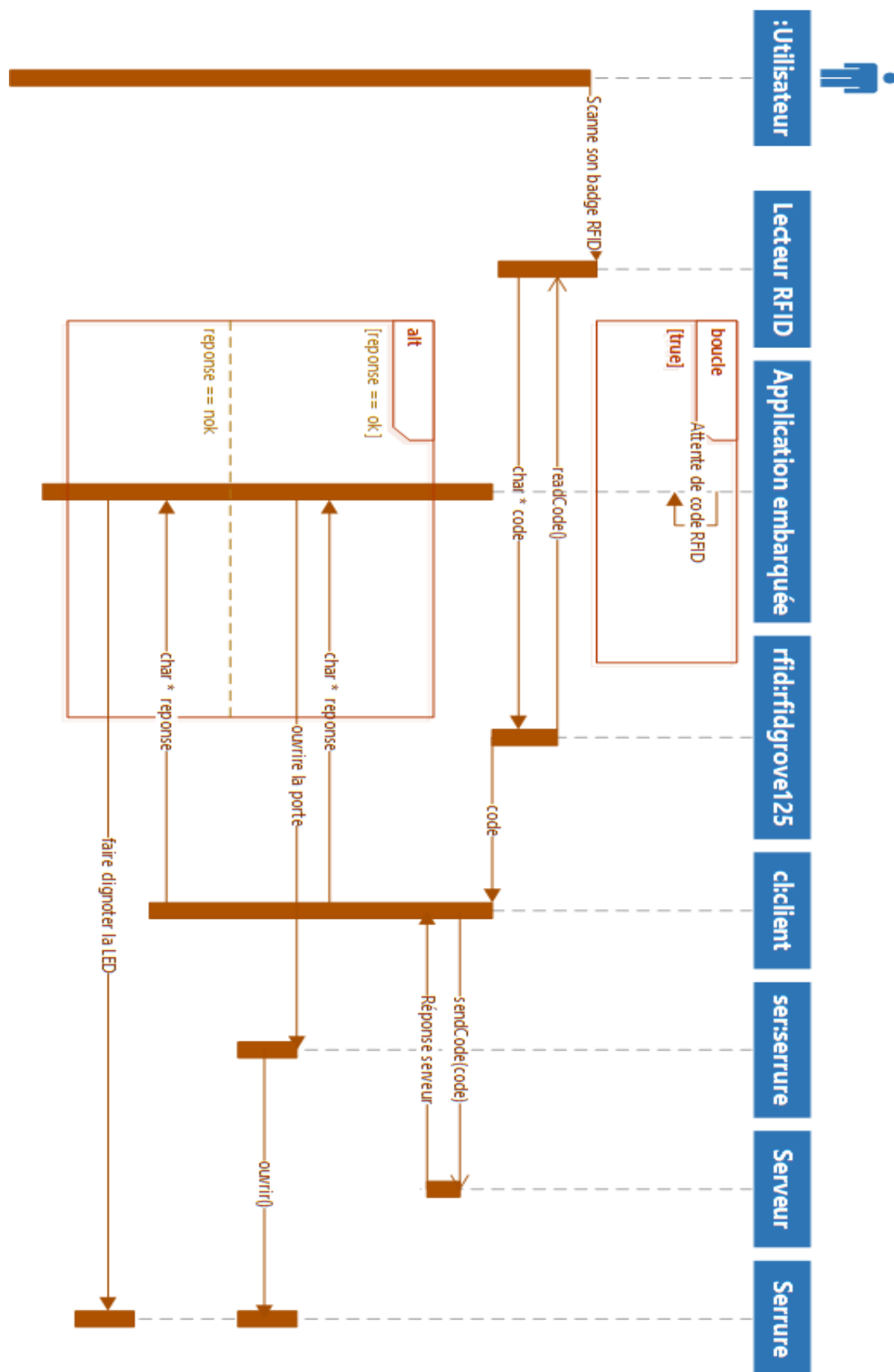


## 4. Application BECK DK40 (HAMOUTEN Yassine)

### Diagramme de classe



## Diagramme de séquence



APPLICATION EMBARQUÉE BECK DK40  
(YASSINE HAMOUTEN)

## 1. Présentation du module BECK DK40

Le DK40 est un produit de la société allemande BECK IPC. BECK est un membre du groupe Festo GmbH qui travaille notamment dans l'automatisation industriel. Il est spécialisé dans la conception de systèmes embarqués.

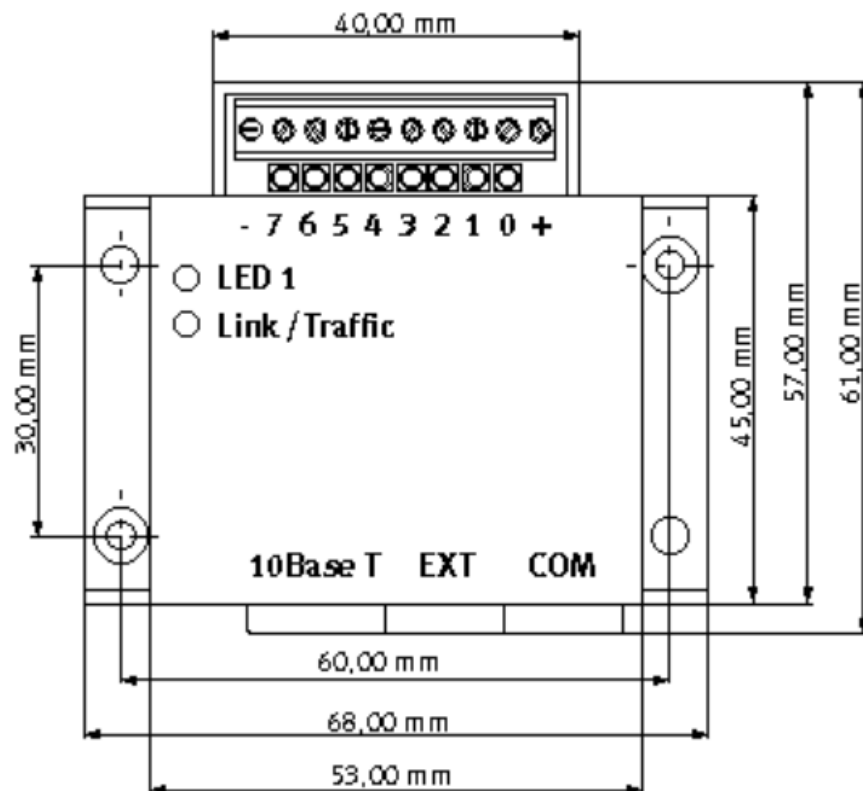
Définissons exactement les termes DK40 et SC12 que nous utiliserons.

Le SC12 est le cœur du système. C'est le microcontrôleur qui va permettre de gérer les périphériques que sont la serrure électrique, le lecteur RFID ainsi que la LED. Ce microcontrôleur intègre un microprocesseur 16 bits du type 80186 cadencé à 20 Mhz, 512 Ko de mémoire Flash, 512 Ko de RAM, des possibilités de communication par RS232 et Ethernet 10 Base-T.



Mais on ne peut exploiter le SC12 qu'en l'intégrant dans un module qui relie directement les pattes du microcontrôleur aux connectiques normales utilisées. Ce module s'appelle le DK40. C'est un boîtier qui intègre 8 Entrées/Sorties avec leurs LEDs de contrôle, 2 ports séries (COM et EXT), 1 port Ethernet 10 Base-T, 1 LED programmable et 1 LED de trafic Ethernet.

Les dimensions :



## Le BIOS (Basic Input Output System)

Il existe différentes versions de BIOS pour le module BECK, selon l'utilisation que l'on souhaite en faire.

Voici les versions disponibles :

	Tiny	Small	Medium	Large
RTOS-Kernel	X	X	X	X
Serial	X	X	X	X
RTOS-Filesystem	X	X	X	X
Ext Disk			X	X
XMODEM-Protocol	X	X	X	X
TCPIP-Ethernetdriver		X	X	X
Ethernet Packet-Interface	X	X	X	X
TCP-IP		X	X	X
I2C	X	X	X	X
Hardware API	X	X	X	X
CFG server		X	X	X
Webserver				X
FTP server			X	X
Telnet Server			X	X
PPP only with Medium & Large				
PPP client & server				

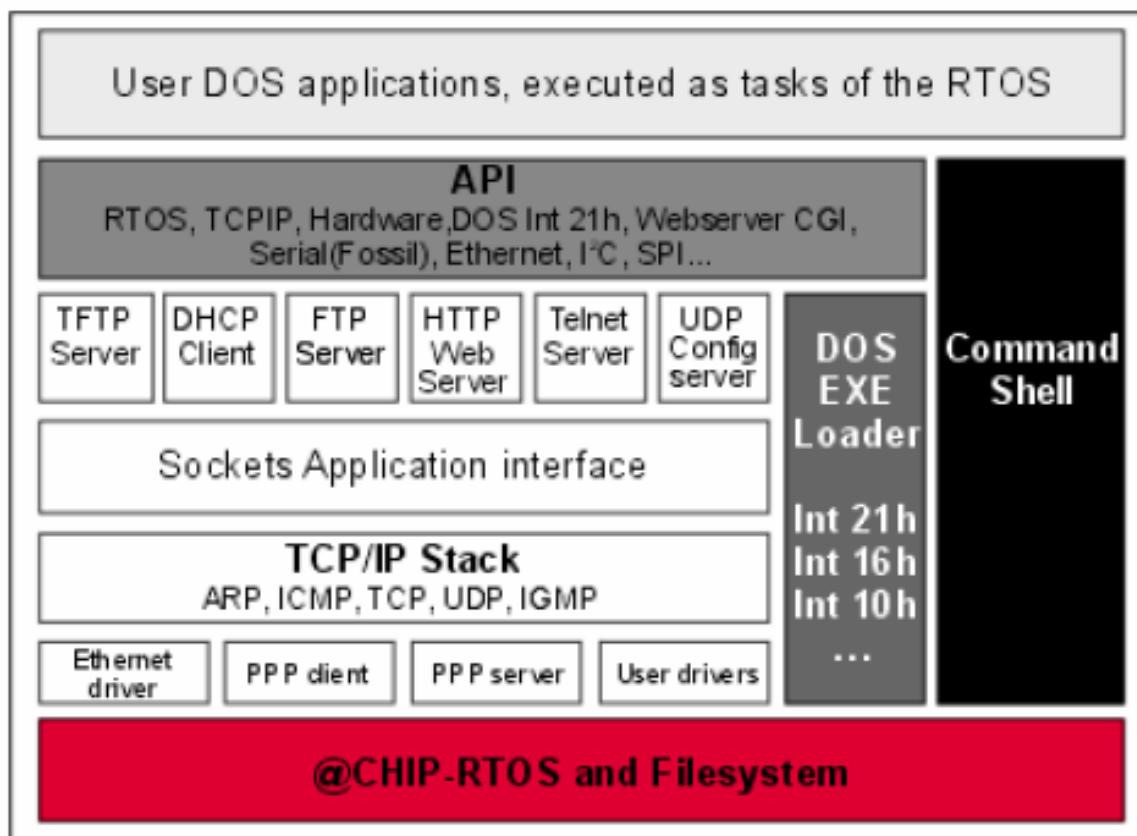
La Ram et la mémoire Flash disponible selon la version du BIOS :

	Available RAM(kBytes)	Available Flash memory(kBytes)
Tiny	453	392
Small	356	287
Medium	351	260
Medium_PPP	337	209
Large	326	237
Large_PPP	304	185

## Le système d'exploitation RTOS (Real Time Operating System)

L'avantage du SC12 réside dans son noyau temps réel multitâche orchestré par son système d'exploitation RTOS. Celui-ci est propre à l'IPC@CHIP qui comporte une API (Application Programmer Interface) complète autorisant à un accès aisé à toutes ses fonctionnalités mais aussi à un accès simplifié aux services TCP/IP.

La figure ci-dessous représente l'architecture du RTOS :



## Le port série

L'interface série est une interface asynchrone, ce qui signifie que le signal de cette interface n'est pas synchronisé avec celui d'un bus quelconque. C'est à dire que les bits des données sont envoyés les uns après les autres. Un caractère est composé d'un ensemble de bit. C'est généralement une matrice de 8x8 bits codé par une valeur. Cette valeur est comprise entre 0 et 255 et elle est stockée sur 1 octets c'est à dire 8 bits. Chaque caractère est délimité par un signal de début qui est un bit à 0 et par signal de fin standard qui peut correspondre à un ou deux bits de fin, cela permet d'indiquer que

le caractère a été envoyé. L'interface asynchrone est orienté caractère, c'est à dire que l'on doit utiliser les signaux de début et de fin pour identifier un caractère. L'inconvénient de ce processus c'est qu'il augmente la durée des transferts de presque 25 %. En effet pour chaque ligne de 8 bits il faut au minimum 2 bits.

Le terme "série" vient du fait que les bits sont envoyés les uns après les autres sur un seul fil pour l'émission et un autre fil pour la réception, comme pour le téléphone. Il existe de nombreuses cartes d'extension permettant d'avoir plusieurs ports séries ou port parallèle.

## Le port Ethernet

Ethernet (aussi connu sous le nom de norme IEEE 802.3) est une technologie de réseau local basé sur le principe suivant :

*Toutes les machines du réseau Ethernet sont connectées à une même ligne de communication, constituée de câbles cylindriques.*

On distingue différentes variantes de technologies Ethernet suivant le diamètre des câbles utilisés:

- 10Base-2: Le câble utilisé est un câble coaxial de faible diamètre
- 10Base-5: Le câble utilisé est un câble coaxial de gros diamètre
- 10Base-T: Le câble utilisé est une paire torsadée, le débit atteint est d'environ 10Mbps
- 100Base-TX: Comme 10Base-T mais avec une vitesse de transmission beaucoup plus importante (100Mbps)

Technologie	Type de câble	Vitesse	Portée
10 Base-2	Câble coaxial de faible diamètre	10 Mb/s	185m
10 Base-5	Câble coaxial de gros diamètre (environ 1cm)	10 Mb/s	500m
10 Base-T	double paire torsadée	10 Mb/s	100m
100 Base-TX	double paire torsadée	100 Mb/s	100m
1000 Base-SX	fibre optique	1000 Mb/s	500m

Ethernet est une technologie de réseau très utilisée car le coût d'un tel réseau n'est pas très élevé. Tous les ordinateurs d'un réseau Ethernet sont reliés à une même ligne de transmission, et la communication se fait à l'aide d'un protocole appelé CSMA/CD (Carrier Sense Multiple Access with Collision Detect ce qui signifie qu'il s'agit d'un



protocole d'accès multiple avec surveillance de porteuse (Carrier Sense) et détection de collision).

Avec ce protocole toute machine est autorisée à émettre sur la ligne à n'importe quel moment et sans notion de priorité entre les machines. Cette communication se fait de façon simple:

- Chaque machine vérifie qu'il n'y a aucune communication sur la ligne avant d'émettre
- Si deux machines émettent simultanément, alors il y a collision (c'est-à-dire que plusieurs trames de données se trouvent sur la ligne au même moment)
- Les deux machines interrompent leur communication et attendent un délai aléatoire, puis la première ayant passé ce délai peut alors ré émettre.

Ce principe est basé sur plusieurs contraintes:

- Les paquets de données doivent avoir une taille maximale
- Il doit y avoir un temps d'attente entre deux transmissions

Le temps d'attente varie selon la fréquence des collisions:

- Après la première collision une machine attend une unité de temps
- Après la seconde collision la machine attend deux unités de temps
- Après la troisième collision la machine attend quatre unités de temps
- ... avec un petit temps supplémentaire aléatoire.

## 2. Configuration du module BECK DK40

Il n'y avait pas d'installation de système d'exploitation à faire. En effet le système étant déjà installé j'ai juste eu à configurer l'adresse IP du module.

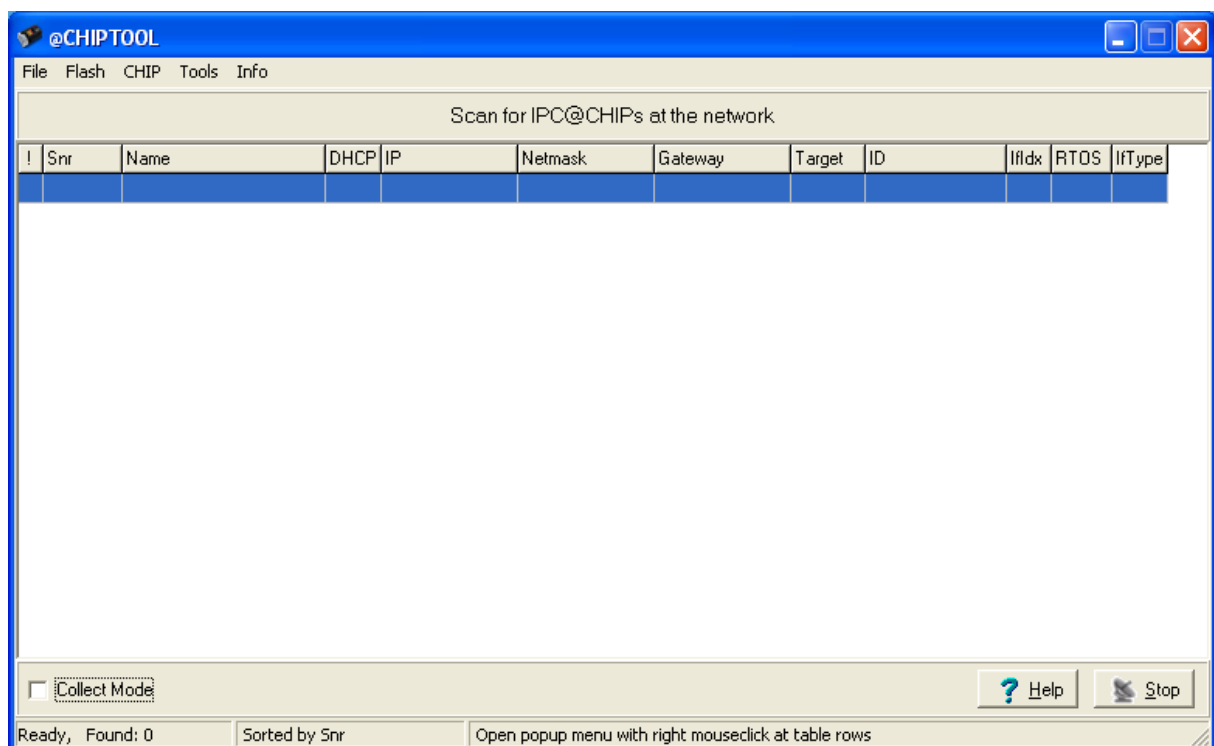
Pour pouvoir modifier la configuration réseau, deux solutions s'offraient à moi.

Soit utiliser la connexion RS232 pour accéder via un HyperTerminal à la console.

Soit utiliser l'application fournie par le constructeur qui s'appelle Chiptool.

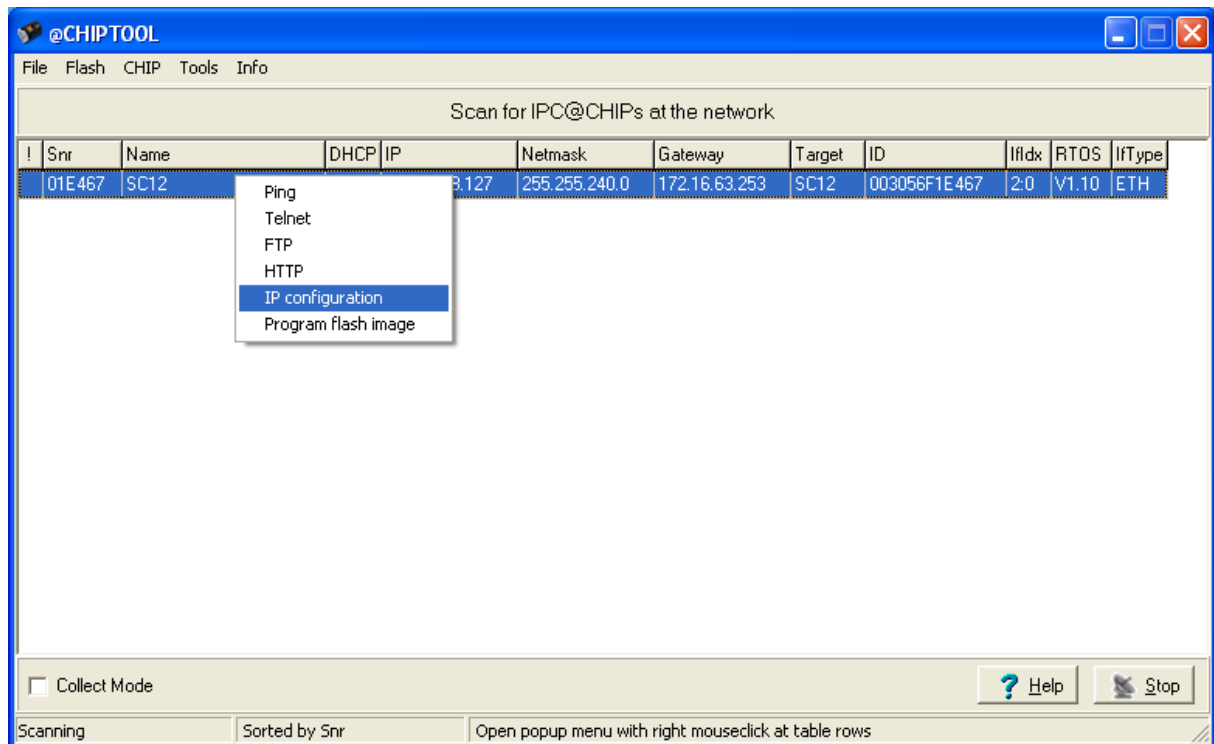
J'ai opté pour la seconde solution. En effet cela était beaucoup plus simple pour moi puisque tous les ordinateurs n'étaient pas équipés d'un port série.

Chiptool se présente sous cette forme :

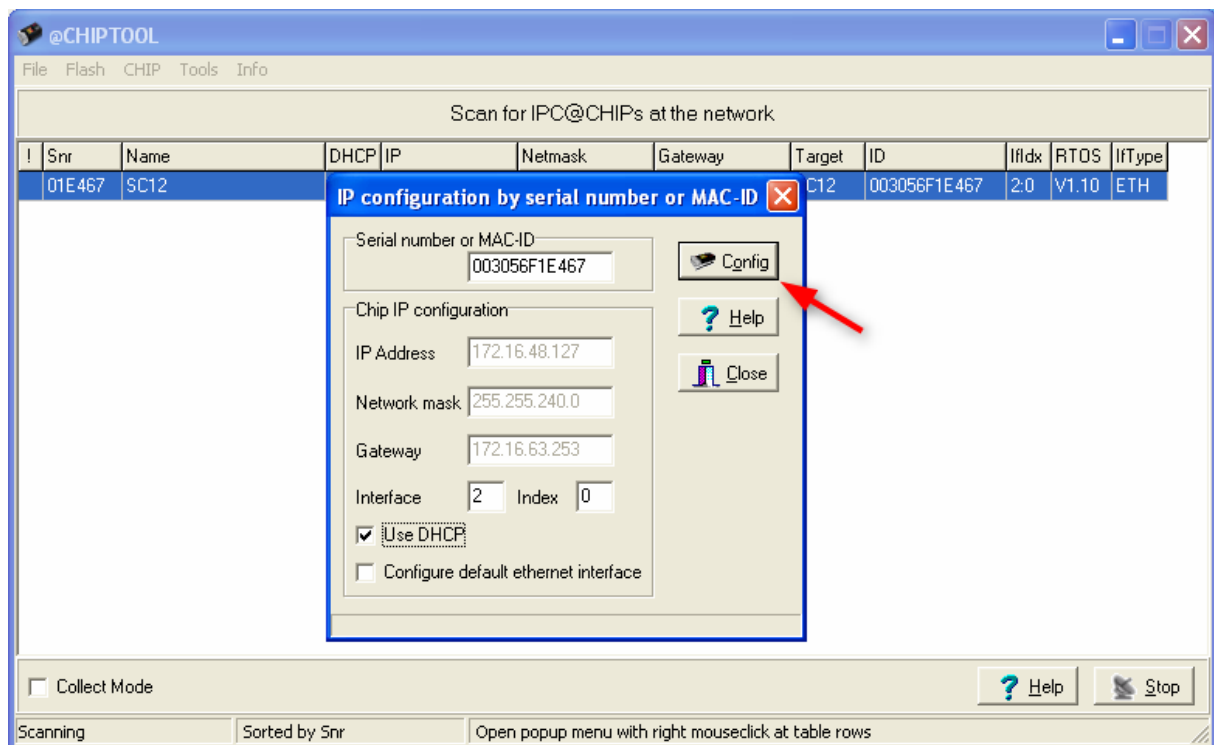


J'ai préféré utiliser une adresse IP dynamique (DHCP).

Cela évite de reconfigurer l'adresse IP à chaque changement de salle. En effet, les salles ne disposent pas des mêmes adresses IP, cela aurait été une perte de temps de toujours reconfigurer le module BECK DK40.



Il suffit de cocher la case Use DHCP et puis cliquer sur config.



On peut vérifier que la modification a bien été prise en compte en se connectant via un terminal sur le module DK40.

```
@CHIPTOOL Terminal (172.16.48.127)
File Edit View Connection

Username: tel
Password: ***
User logged in

A:\>ipcfg

Device = NE2000
Type = ETH
Idx = 2
IP = 172.16.48.127
Netmask = 255.255.240.0
DHCP = 1
Mac = 00 30 56 F1 E4 67

Device = LOOPBACK
Type = LPK
Idx = 1
IP = 127.0.0.1
Netmask = 255.255.255.255

Gateway = 172.16.63.253

A:\>

Connected via Telnet to 172.16.48.127:Telnet.
```

Ici, on voit bien que le DHCP est activé.

J'aurais aussi pu directement modifier le fichier chip.ini qui se trouve à la racine du ftp. C'est un fichier de configuration du module BECK DK40, il permet entre autre de configurer l'adresse IP, la vitesse de transfert sur le port série, d'activer les différents services intégrés au module (serveur web, serveur FTP, Telnet...).

### 3. Le matériel utilisé

#### Lecteur RFID

La communication avec l'utilisateur se fera par le biais d'un lecteur RFID. Ce lecteur RFID est placé à l'entrée des salles. L'utilisateur devra scanner son badge RFID pour pouvoir entrer dans la salle, à condition que l'utilisateur soit autorisé à y entrer. Le code RFID est composé de 12 caractères.

Les différents codes sont stockés en base de données.

RFID (Radio Frequency Identification) est une technologie d'identification automatique qui utilise le rayonnement radiofréquence pour identifier les objets porteurs d'étiquettes lorsqu'ils passent à proximité d'un interrogateur.

J'ai eu à ma disposition un lecteur RFID Grove – 125KHz.



Voici les caractéristiques de ce lecteur RFID :

- Alimentation : 5V
- Fréquence : 125KHz
- Distance de détection : 7cm max
- Sortie UART : 9600 bauds, 8 bits de données et 1 bit de stop

#### Serrure électrique

La gestion de l'ouverture de la porte est faite via une serrure électrique.

Cette serrure nécessite 12 V et 500mA pour fonctionner.

## 4. Environnement de travail

Durant ces 5 mois de développement, j'ai eu à ma disposition un module BECK DK40, un lecteur RFID, un badge RFID ainsi qu'une serrure électrique.

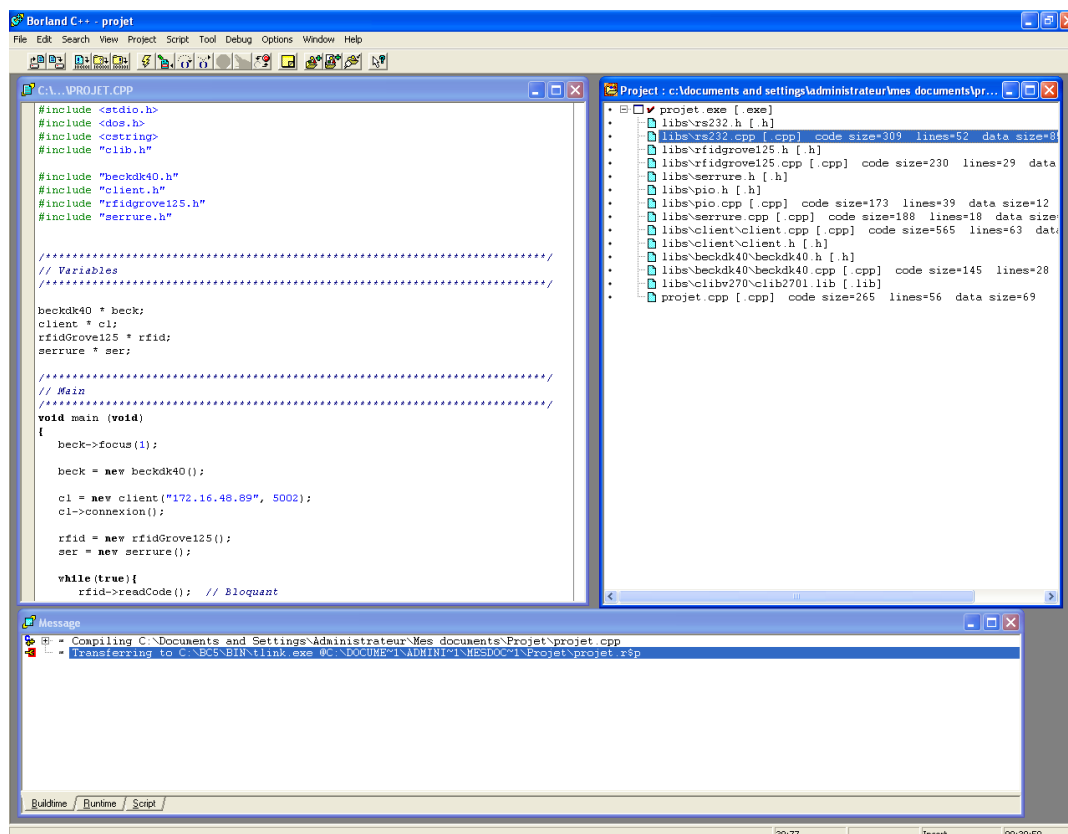
J'ai décidé de travailler sur une machine virtuelle puisque nous n'avions pas de salle attribué pour notre projet, donc pas de poste fixe. Etant donné que les machines sont freezer, donc toutes les applications installées sont effacées une fois la machine redémarrée.

Ce qui était une grosse perte de temps de toujours devoir tout réinstaller.

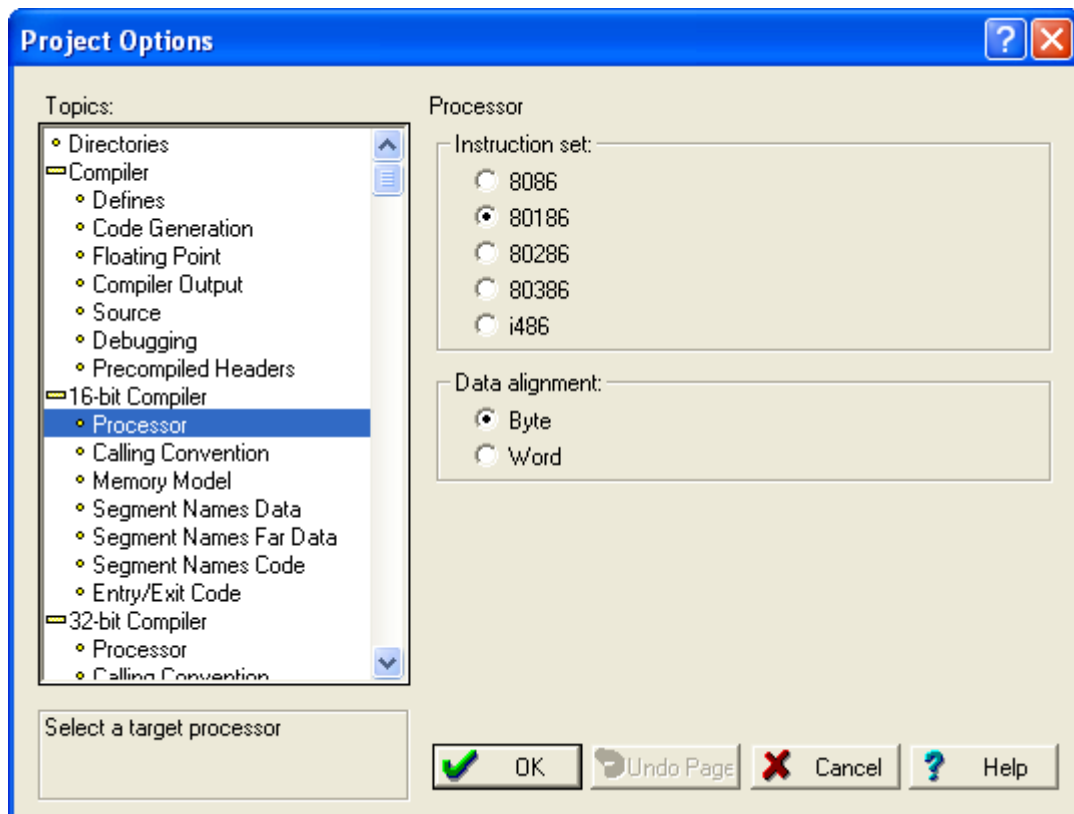
De plus, les machines des salles utilisées Windows 7, j'étais dans l'obligation d'utiliser une machine virtuelle car les applications de développement ne fonctionnent que sous Windows XP.

## 5. Environnement de développement

Le module BECK DK40 utilise un processeur 80186, qui est en soit très ancien. L'un des seuls éditeurs qui propose de compiler pour ce genre de processeur est Borland C++. Borland C++ est un outil de développement pour les systèmes MS-DOS.

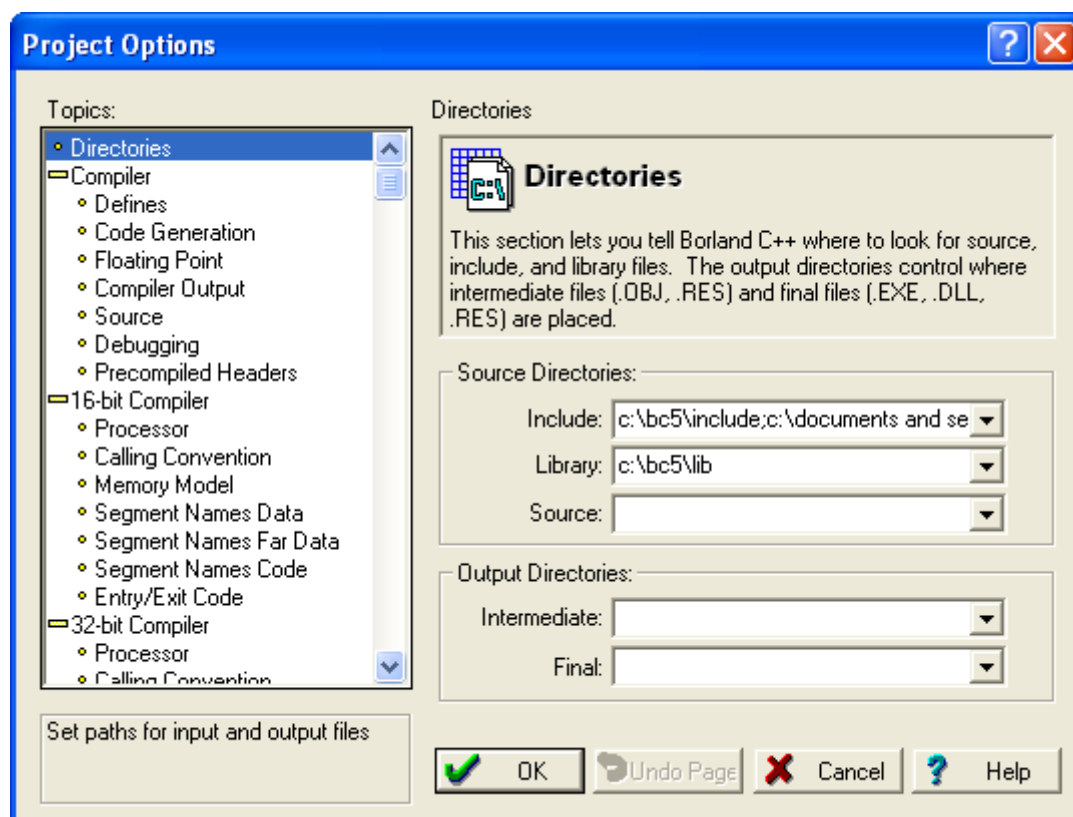


Avant de pouvoir coder j'ai tout d'abord configuré Borland C++ de façon à ce qu'il compile pour un processeur 80186.



Ensuite j'ai inclus les différents dossiers où se trouvent les bibliothèques : le dossier où se trouve la bibliothèque de Borland C++, le dossier où se trouvent les différentes classes ainsi que le dossier où se trouvent les bibliothèques de la BECK DK40.

On inclut aussi directement la bibliothèque de Borland C++.



Après avoir configuré le projet, j'ai pu commencer à développer l'application embarqué de contrôle d'accès.



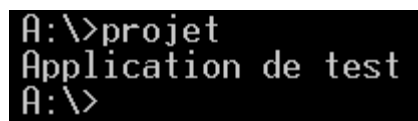
## 6. Tests de mise en œuvre

### Test simple d'application sur le module BECK DK40

Cette application va juste permettre d'écrire un message dans la console.

```
void main (void)
{
    printf("Application de test");
    exit(0);
}
```

Résultat :



```
A:\>projet
Application de test
A:\>
```

### Test des Entrées / Sorties du module BECK DK40

Pour ouvrir la porte il faut utiliser les E/S du module BECK DK40.

```
void pio::writePIO(unsigned char value){
    //0x10 -> PIN5
    //0x20 -> PIN6
    //0x40 -> PIN7
    //0x80 -> PIN8

    // ecrit les donnees
    hal_write_bus(0x600, value, 0xFFFF, 0x0000);

    if (value&0x80){
        //exemple: 0x10 & 0x80 -> 1 0000 & 1000 0000 -> pin n4 (5eme led)
allumee
        hal_write_pio(13, 1);
    }else{
        hal_write_pio(13, 0);
    }
}

void serrure::ouvrir(){
    printf("\nOuverture de la porte\n");
    this->clearPIO();
    this->writePIO(0xC0);
    RTX_Sleep_Time(2000);
    this->clearPIO();
    printf("Fermeture de la porte\n");
}
```

On voit que le témoin lumineux de la sortie s'est allumé



## Test du lecteur RFID

Pour pouvoir utiliser le lecteur RFID, j'ai dû utiliser la liaison RS232. Pour cela j'ai utilisé des fonctions spécifiques à la librairie cLib.

Tout d'abord il faut initialiser le port COM :

```
rs232::rs232(int port)
{
    // Initialisation du port COM

    //Init Fossil
    this->portCom = port;
    if(fossil_init(this->portCom) != 6484) {
        printf("Erreur Init fossil!\n");
    }else{
        printf("Pas d'erreur Init fossil!\n");
    }

    // Purge des buffers
    fossil_purge_output(this->portCom);
    fossil_purge_input(this->portCom);

    // Definition des parametres pour le portCom
    fossil_setbaud (this->portCom , 9600L, 0, 8, 1);
    fossil_set_flowcontrol(this->portCom,0);

    printf("RS232 OK!\r\n");
}
```

Lecture d'un caractère sur le port COM :

```
char rs232::getBytes() {  
    // Lit un caractère sur le port COM  
    return fossil_getbyte_wait(this->portCom);  
}
```

Pour lire la totalité du code RFID :

```
void rfidGrove125::readCode() {  
    printf("\n\nEn attente du code RFID\n");  
  
    int i = 0;  
    char c;  
    while(i<12){  
        c = this->getBytes();  
        if(c != 0x02 && c != 0x03){  
            code[i] = c;  
            i++;  
        }  
    }  
  
    printf("Code RFID lu : %s\n",code);  
}
```

## Test de connexion au serveur

Pour pouvoir ouvrir la porte d'une salle, il faut tout d'abord savoir si la personne peut y accéder. Pour cela il faut interroger la base de données via un serveur qui va faire office de passerelle entre la base de données et le système embarqué.

J'ai créé un client pour pouvoir communiquer avec le serveur.

Durant la phase de test j'ai aussi créé un serveur de test.

Pour communiquer avec le serveur il faut ouvrir un socket :

```
client::client(char * ip, int port){
    this->IPServeur = ip ;
    this->PortServeur = port ;
    this->bufLen = 1024;
    this->addr.sin_family = PF_INET ;
    this->addr.sin_addr.s_addr = 0 ;
    // Convertit entier depuis l'ordre des octets de l'hôte vers celui du
réseau
    this->addr.sin_port = htons (this->PortServeur) ;
    // Convertit l'adresse ip en binaire
    inet_addr (this->IPServeur, &addr.sin_addr.s_addr);

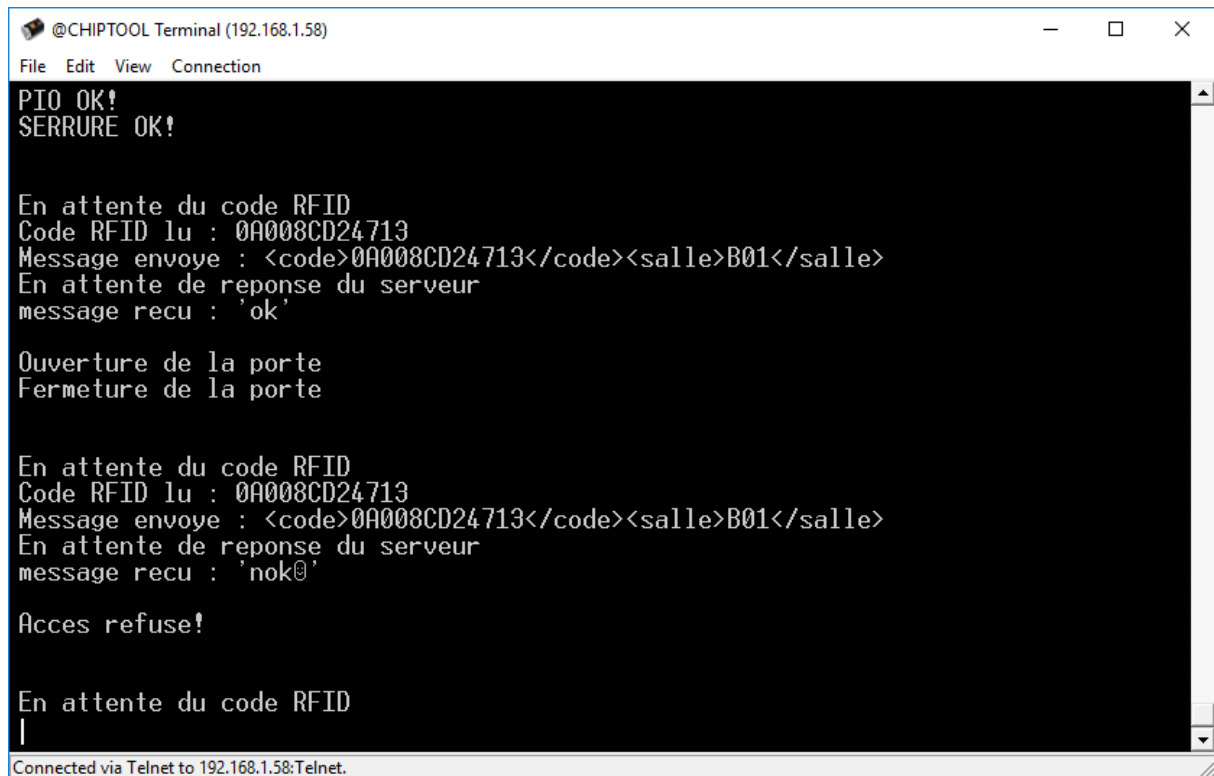
    this->sd = opensocket(1, 0);
    if(this->sd!=-1){
        printf("\nSocket ouvert!\n");
    }
}
```

Ensuite on peut se connecter au serveur :

```
void client::connexion(){
    result = connect( sd, (const struct sockaddr far *)&addr, &error_code
);
    if(result == 0){
        //Connexion reussie
        this->connecte = true;
    }else{
        //Connexion echouee
        this->connecte = false;
    }
}
```

Ces étapes effectuées, on peut envoyer le code RFID ainsi que le nom de la salle d'où l'utilisateur souhaite accéder.

On peut voir que le client arrive à communiquer avec le serveur :



```
@CHIPTOOL Terminal (192.168.1.58)
File Edit View Connection

PIO OK!
SERRURE OK!

En attente du code RFID
Code RFID lu : 0A008CD24713
Message envoye : <code>0A008CD24713</code><salle>B01</salle>
En attente de reponse du serveur
message reçu : 'ok'

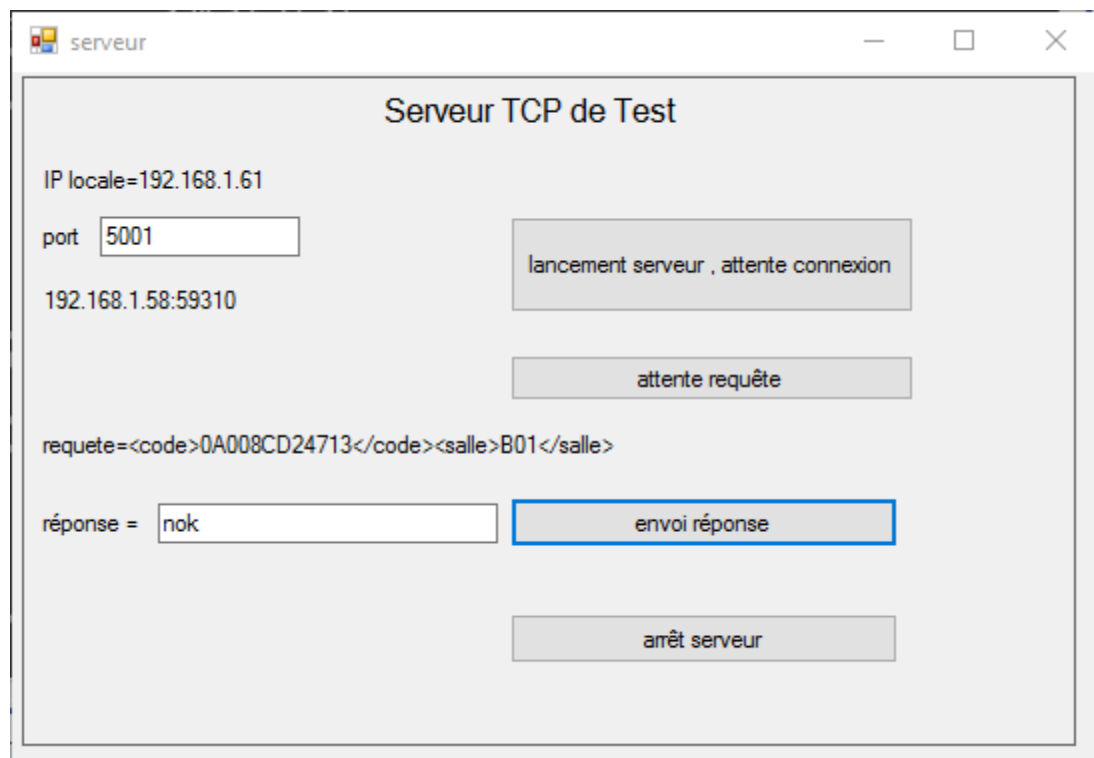
Ouverture de la porte
Fermeture de la porte

En attente du code RFID
Code RFID lu : 0A008CD24713
Message envoye : <code>0A008CD24713</code><salle>B01</salle>
En attente de reponse du serveur
message reçu : 'nok'

Acces refuse!

En attente du code RFID
|

Connected via Telnet to 192.168.1.58:Telnet.
```



serveur

### Serveur TCP de Test

IP locale=192.168.1.61

port

192.168.1.58:59310

requete=<code>0A008CD24713</code><salle>B01</salle>

réponse =

## 7. Recettes

### Module d'identification RFID

<b>Test fonctionnel : [BDK40-1] Test RFID</b>				
<b>Objectif :</b>		Vérification de la lecture du badge RFID		
<b>Éléments à tester :</b>		Application embarqué BECK DK40		
<b>Pré requis :</b>		<ul style="list-style-type: none"> <li>– Module BECK DK40 allumé</li> <li>– Module BECK DK40 connecté au réseau</li> <li>– Le serveur est lancé</li> <li>– Un lecteur RFID connecté au module BECK DK40</li> <li>– Un utilisateur avec un badge RFID</li> </ul>		
<b>Initialisation :</b>		– On lance l'application		
<b>Scénario :</b>				
<b>ID</b>	<b>Démarche</b>	<b>Données</b>	<b>Comportement attendu</b>	<b>Validation</b>
1	L'utilisateur scan son badge RFID	Un code RFID correspondant à l'utilisateur	Le code est lu par le module BECK DK40 (vérification à faire via le Terminal)	
<b>Rapport de test</b> <b>Testé par :</b> <b>Le :</b>				
<b>Fonctionnalité :</b>		<b>Conformité :</b>		<b>Ergonomie :</b>
<input type="checkbox"/> Excellente <input type="checkbox"/> Bonne <input type="checkbox"/> Moyenne <input type="checkbox"/> Faible		<input type="checkbox"/> Excellente <input type="checkbox"/> Bonne <input type="checkbox"/> Moyenne <input type="checkbox"/> Faible		<input type="checkbox"/> Excellente <input type="checkbox"/> Bonne <input type="checkbox"/> Moyenne <input type="checkbox"/> Faible
<b>Commentaire :</b>			<b>Approbation :</b>	
Fiches d'anomalies émises :				

## Module Entrées / Sortie

<b>Test fonctionnel : [BDK40-2] Test des Entrées / Sorties</b>				
<b>Objectif :</b>		Vérification de la mise en œuvre des E/S		
<b>Éléments à tester :</b>		Application embarqué BECK DK40		
<b>Pré requis :</b>		<ul style="list-style-type: none"> <li>– Module BECK DK40 allumé</li> <li>– Module BECK DK40 connecté au réseau</li> <li>– Le serveur est lancé</li> <li>– Un lecteur RFID connecté au module BECK DK40</li> <li>– Un utilisateur avec un badge RFID</li> </ul>		
<b>Initialisation :</b>		On lance l'application, on scanne un badge RFID		
<b>Scénario :</b>				
<b>ID</b>	<b>Démarche</b>	<b>Données</b>	<b>Comportement attendu</b>	<b>Validation</b>
1	Le serveur répond à la demande avec comme réponse « OK »		Sur le module BECK DK40, le voyant au-dessus des Entrées / Sorties s'allume	
<b>Rapport de test</b> <b>Testé par :</b> <b>Le :</b>				
<b>Fonctionnalité :</b>		<b>Conformité :</b>		<b>Ergonomie :</b>
<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente		
<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne		
<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne		
<input type="checkbox"/> Faible	<input type="checkbox"/> Faible	<input type="checkbox"/> Faible		
<b>Commentaire :</b>			<b>Approbation :</b>	
Fiches d'anomalies émises :				

## Module Client

<b>Test fonctionnel : [BDK40-3] Test du client spécifique</b>				
<b>Objectif :</b>		Vérification de la communication entre le client et le serveur		
<b>Eléments à tester :</b>		Application embarqué BECK DK40		
<b>Pré requis :</b>		<ul style="list-style-type: none"> <li>– Module BECK DK40 allumé</li> <li>– Module BECK DK40 connecté au réseau</li> <li>– Le serveur est lancé</li> <li>– Un lecteur RFID connecté au module BECK DK40</li> <li>– Un utilisateur avec un badge RFID</li> </ul>		
<b>Initialisation :</b>		On lance l'application		
<b>Scénario :</b>				
<b>ID</b>	<b>Démarche</b>	<b>Données</b>	<b>Comportement attendu</b>	<b>Validation</b>
1	L'utilisateur scanne son badge RFID, le code RFID est envoyé au serveur	Un code RFID correspondant à l'utilisateur	Le serveur répond « ok » ou « nok » (vérification à faire via le Terminal ou avec WIRESHARK)	
<b>Rapport de test</b>		<b>Testé par :</b>		<b>Le :</b>
<b>Fonctionnalité :</b>		<b>Conformité :</b>		<b>Ergonomie :</b>
<input type="checkbox"/> Excellente <input type="checkbox"/> Bonne <input type="checkbox"/> Moyenne <input type="checkbox"/> Faible		<input type="checkbox"/> Excellente <input type="checkbox"/> Bonne <input type="checkbox"/> Moyenne <input type="checkbox"/> Faible		<input type="checkbox"/> Excellente <input type="checkbox"/> Bonne <input type="checkbox"/> Moyenne <input type="checkbox"/> Faible
<b>Commentaire :</b>			<b>Approbation :</b>	
Fiches d'anomalies émises :				



## Application complète

Test fonctionnel : [BDK40-4] Test de l'application embarqué finale				
Objectif :		La porte s'ouvre après qu'un utilisateur scanne son badge RFID si celui-ci y a accès sinon une LED clignote pour signaler que l'utilisateur n'a pas le droit d'accéder à la salle		
Eléments à tester :		Application embarqué BECK DK40		
Pré requis :		<ul style="list-style-type: none"><li>— Module BECK DK40 allumé</li><li>— Module BECK DK40 connecté au réseau</li><li>— Le serveur est lancé</li><li>— Un lecteur RFID connecté au module BECK DK40</li><li>— La serrure électrique est reliée au module BECK DK40</li><li>— Un utilisateur avec un badge RFID</li></ul>		
Initialisation :		On lance l'application		
Scénario :				
ID	Démarche	Données	Comportement attendu	Validation
1	L'utilisateur scanne son badge RFID	Un code RFID correspondant à l'utilisateur	Le code RFID est lu (vérification à faire via le Terminal)	
2	Le code est envoyé au serveur via le client intégré à l'application	Le code RFID précédemment scanné	Le code RFID est envoyé au serveur (vérification à faire via le Terminal)	
3	Le serveur répond à la demande du client	« ok » ou « nok »	Une réponse est reçue de la part du serveur (vérification à faire via le Terminal)	
4	Suivant la réponse du serveur la porte s'ouvre ou non		La porte s'ouvre si la réponse du serveur est « ok » Elle reste fermée et une LED clignote si la réponse du serveur est « nok »	
Rapport de test		Testé par :		Le :

Fonctionnalité :	Conformité :	Ergonomie :
<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente
<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne
<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne
<input type="checkbox"/> Faible	<input type="checkbox"/> Faible	<input type="checkbox"/> Faible
Commentaire :		Approbation :
Fiches d'anomalies émises :		

## 8. Tests unitaires et tests d'intégrations

### Test rs232

#### Méthode de test getByte()

```
char rs232::getByte() {
    // Lit un caractere sur le port COM
    return fossil_getbyte_wait(this->portCom);
}
```

### Résultats attendus

#### Récupérer le code RFID

#### Validation

```
void rfidGrove125::readCode() {
    printf("\n\nEn attente du code RFID\n");

    int i = 0;
    char c;
    while(i<12){
        c = this->getByte();
        if(c != 0x02 && c != 0x03){
            code[i] = c;
            i++;
        }
    }

    printf("Code RFID lu : %s\n",code);
}
```

### Résultat dans la console

```
En attente du code RFID
Code RFID lu : 0A008CD24713
```

**Test client****Méthode de test connexion()**

```

void client::connexion(){
    result = connect( sd, (const struct sockaddr far *)&addr, &error_code
);
    if(result == 0){
        //Connexion reussie
        this->connecte = true;
    }else{
        //Connexion echouee
        this->connecte = false;
    }
}

```

**Résultats attendus**

On arrive à se connecter au serveur distant

**Validation**

```

void main (void)
{
    beckdk40 * beck;
    client * cl;
    rfidGrovel25 * rfid;
    serrure * ser;
    char * ip;
    int port;
    char * salle = "B01";
    int portCOM = 0;

    FILE * fichier;
    char str[20];
    fichier = fopen ("config.ini","r");
    if (fichier!=NULL){
        int i=0;
        while(fgets (str, sizeof(str), fichier)!=NULL){
            if(i==0){
                strcpy(ip, str);
                ip[strlen(ip)-1] = 0;
            }else{
                port = atoi(str);
            }
            i++;
        }
        fclose (fichier);
    }else{
        printf("impossible d'ouvrir le fichier de config!");
        exit(0);
    }

    beck = new beckdk40(portCOM);
    beck->focus(1);

    printf("ip : %s port : %d", ip, port);
    cl = new client(ip, port);
    cl->connexion();
}

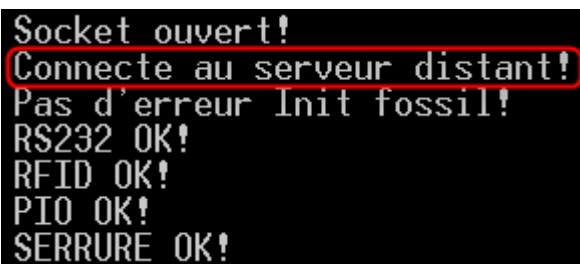
```

```
if(cl->connecte){
    printf("Connecte au serveur distant!\n");
    rfid = new rfidGrovel25(portCOM);
    ser = new serrure();

    while(true){
        rfid->readCode(); // Bloquant
        char * code = rfid->getCode();
        char message[100];
        strcpy(message, "<code>");
        strcat(message, code);
        strcat(message, "</code><salle>");
        strcat(message, salle);
        strcat(message, "</salle>");
        cl->sendCode(message);
        int rep = cl->waitResponse();
        if(rep == 1){
            ser->ouvrir();
        }else if(rep == 2){
            printf("\nPas de reponse!\n");
        }else {
            printf("\nAcces refuse!\n");
        }
    }
}else{
    printf("Impossible de se connecter au serveur!\n");
    printf("Verifier que le serveur est bien lance\n");
    printf("Verifier le fichier de configuration(config.ini)");
}

beck->focus(0);
exit(0);
}
```

### Résultat dans la console



```
Socket ouvert!
Connecte au serveur distant!
Pas d'erreur Init fossil!
RS232 OK!
RFID OK!
PIO OK!
SERRURE OK!
```

**Test serrure****Méthode de test ouvrir()**

```
void serrure::ouvrir(){  
    printf("\nOuverture de la porte\n");  
    this->clearPIO();  
    this->writePIO(0xC0);  
    RTX_Sleep_Time(2000);  
    this->clearPIO();  
    printf("Fermeture de la porte\n");  
}
```

**Résultats attendus**

On arrive à utiliser une sortie du module BECK DK40 pour pouvoir ouvrir la porte

**Validation**

Vérification visuelle du voyant (LED) directement sur le module BECK DK40

**Résultat**

## 9. Problèmes rencontrés / solutions

La première difficulté que j'ai rencontrée a été de trouver la librairie (cLib) du module BECK DK40 pour Borland C++. Cela a été très compliqué étant donné que la dernière version de cette librairie ne contenait pas la librairie compilée pour Borland C++ mais uniquement pour Paradigm (prix de la licence pour Paradigm 800€ sur le site du constructeur BECK). De ce fait j'ai passé beaucoup de temps à chercher une librairie compatible pour Borland C++. Au bout de plusieurs heures de recherche, j'ai finalement trouvé la version cLibv2701, qui est compatible avec Borland C++, en pièce jointe sur un forum.

Le second problème que j'ai rencontré a été de fournir assez de puissance pour pouvoir utiliser la serrure électrique. En effet, la serrure nécessite 500mA pour pouvoir fonctionner alors que le module BECK DK40 peut délivrer maximum 230mA en sortie. Pour résoudre ce problème j'ai utilisé un relais électrique qui va permettre de délivrer assez de mA pour alimenter la serrure.

## 10. Synthèse

L'application embarquée demandée respecte le cahier des charges.

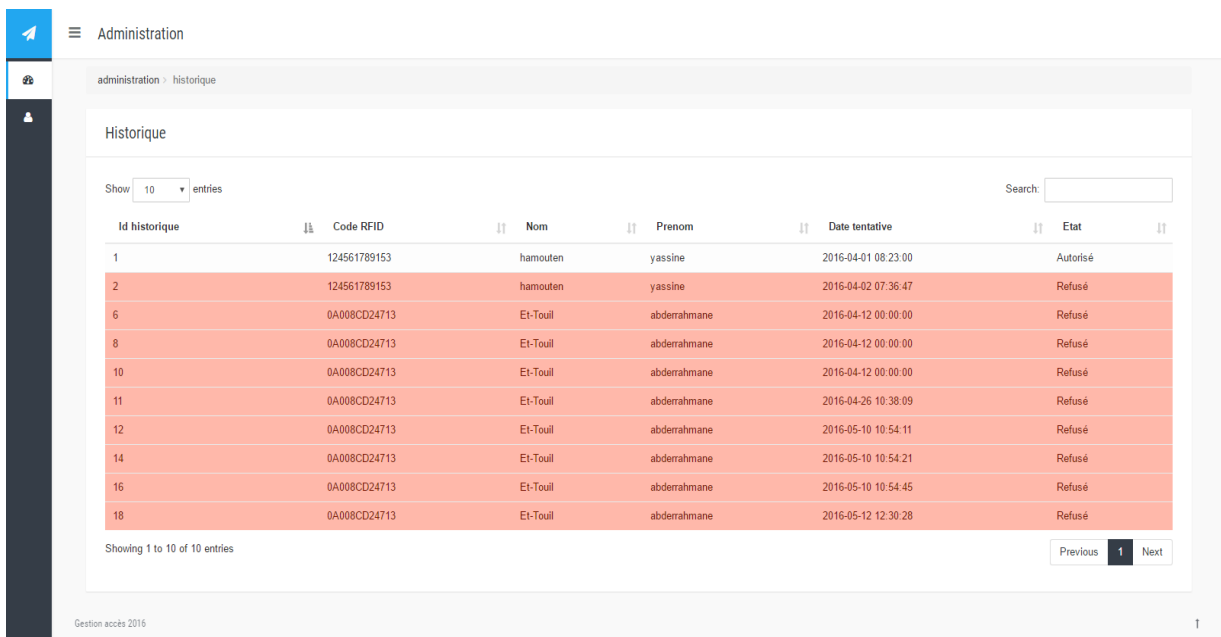
J'ai ajouté des fonctionnalités comme par exemple le fait de pouvoir modifier l'adresse IP et le port du serveur, sur lequel l'application souhaite accéder, grâce au fichier config.ini.

J'ai aussi ajouté un timeout pour la connexion au serveur ainsi qu'un timeout quand on attend une réponse du serveur. Cela permet de ne pas attendre indéfiniment une réponse du serveur si celui-ci n'est pas lancé.

Les modifications apportées n'étaient pas comprises dans le cahier des charges.

Ayant terminé assez rapidement l'application embarquée (environ 2 mois), j'ai commencé à développer une application web pour pouvoir interagir avec la base de données (gestion utilisateurs, gestion plages horaire ainsi que la visualisation de l'historique).

J'ai utilisé le langage PHP avec le Framework CodeIgniter pour développer cette application.



Administration

administration > historique

Historique

Show 10 entries

Search:

Id historique	Code RFID	Nom	Prenom	Date tentative	Etat
1	124561789153	hamouten	yassine	2016-04-01 08:23:00	Autorisé
2	124561789153	hamouten	yassine	2016-04-02 07:36:47	Refusé
6	0A008CD24713	Et-Touil	abderrahmane	2016-04-12 00:00:00	Refusé
8	0A008CD24713	Et-Touil	abderrahmane	2016-04-12 00:00:00	Refusé
10	0A008CD24713	Et-Touil	abderrahmane	2016-04-12 00:00:00	Refusé
11	0A008CD24713	Et-Touil	abderrahmane	2016-04-26 10:38:09	Refusé
12	0A008CD24713	Et-Touil	abderrahmane	2016-05-10 10:54:11	Refusé
14	0A008CD24713	Et-Touil	abderrahmane	2016-05-10 10:54:21	Refusé
16	0A008CD24713	Et-Touil	abderrahmane	2016-05-10 10:54:45	Refusé
18	0A008CD24713	Et-Touil	abderrahmane	2016-05-12 12:30:28	Refusé

Showing 1 to 10 of 10 entries

Previous 1 Next

Gestion accès 2016



# BASE DE DONNÉES - CLIENT LOURD ET SERVEUR TCP (MAXENCE BAUDELET)

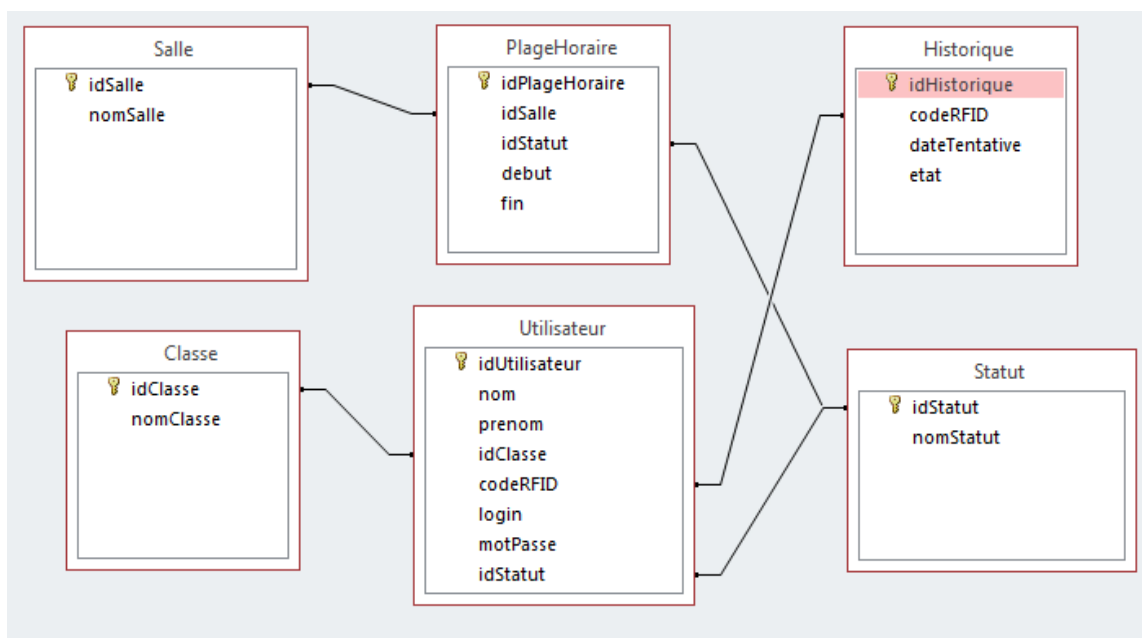
Pour commencer voici la description de l'ensemble des tâches que j'ai dû réaliser durant la période de projet.

Etudiant 2	Description des tâches individuelles	Objectif ou travail attendu	Critère de réussite
	Mise en œuvre affichage table SQL en C#	Petite application de mise en œuvre	L'application permet de lister une table SQL
	Couche DAL horaires, salles	Développement couche DAL en C#	Une couche DAL est clairement identifiée, un programme de test permet les opérations sur la base
	Gestion plages horaires	Développement module de gestion plages horaires en C#	La base de données est correctement mise à jour après les opérations CRUD
	Gestion salles	Développement module de gestion salles en C#	La base de données est correctement mise à jour après les opérations CRUD
	Ecriture du serveur spécifique pour le client BECK	Ecrire le serveur spécifique en C++	Le serveur récupère bien les requêtes d'un client et renvoi les données de la base, ou met à jour la base. On voit la trame avec WIRESHARK.
	Installation, paramétrage des serveurs sur machine physique.	Préparer serveur et l'intégrer dans le réseau du Lycée.	Une machine est correctement configurée avec les serveurs déployés. Elle est accessible depuis le LAN du Lycée.

## 1. Création de la base de données

J'ai été chargé de créer la base de données pour l'ensemble de notre projet selon le modèle physique de données suivant :

Le modèle physique des données est une représentation des entités d'une base de données relationnelle.



Cependant, comme vu en partie commune, nous nous sommes rendu compte que nous devons améliorer et modifier certains points donc ce MPD n'est plus à jour, il est juste là pour présenter la base de départ.

J'ai créé la base de données à partir de PHPMYADMIN car cela était imposé de travailler avec une base de données en MYSQL.

### Présentation MYSQL



MySQL est un serveur de [bases de données relationnelles SQL](#) développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. Il est multithread et multiutilisateur.

Les bases de données y sont accessibles avec une multitude de langage.

La première version de MySQL est apparue le 23 mai 1995. Il a d'abord été créé pour un usage personnel à partir de [mSQL](#) en s'appuyant sur le langage de bas niveau [ISAM](#) qu'ils trouvaient trop lent et trop rigide.

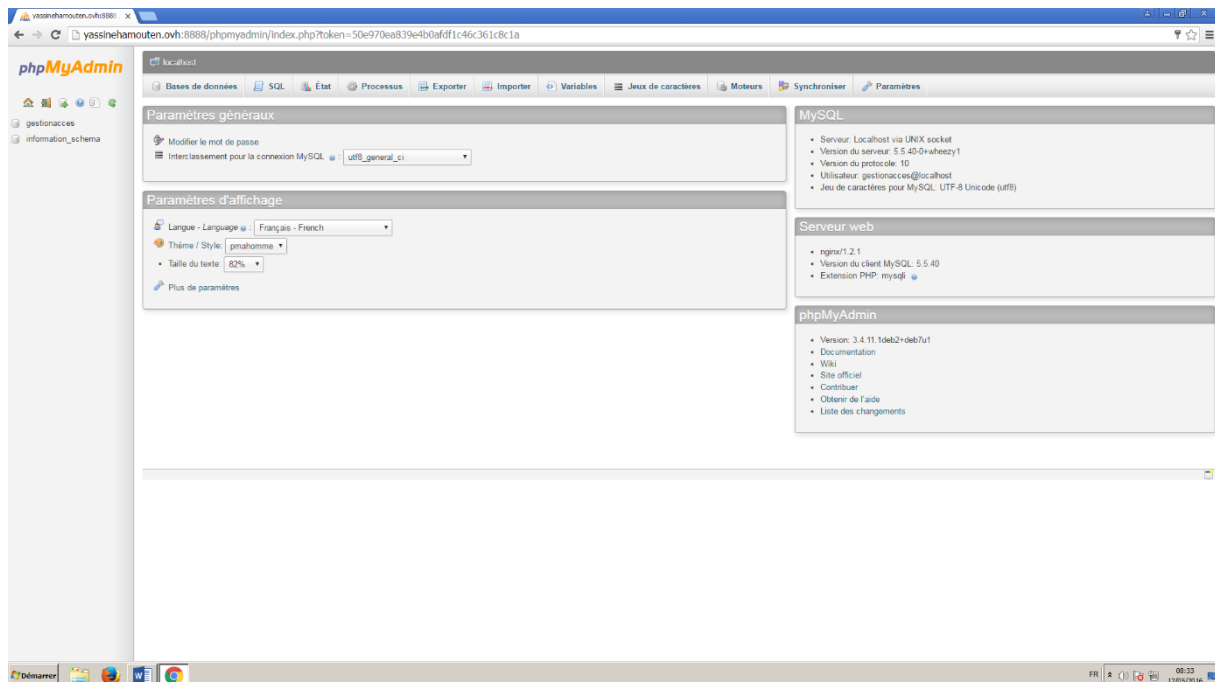
### PHPMYADMIN



PHPMYADMIN est un logiciel gratuit écrit en PHP, conçu pour gérer l'administration d'une base de données MySQL.

Cette interface pratique permet d'exécuter, très facilement et sans grandes connaissances en base de données des requêtes comme la création de table, l'insertion d'une précédente base de données.

Voici l'interface d'utilisation de PHPMYADMIN :



J'ai donc d'abord du créer la base de données « gestion d'accès » et y ajouter l'ensemble des tables : « Salle, Plage Horaire, Historique, Classe, Utilisateur et Statut » ainsi que chacun de leur attribut.



Voici l'interface pour ajouter une table :

Nom de la table:

Structure

Colonne

Type: INT

Taille/Valeurs\*

Défaut: Aucune

Interclassement

Attributs

Null

Index

AUTO\_INCREMENT

Commentaires

Type MIME

Transformation

Options de transformation

On peut ensuite y ajouter le nombre de colonnes qu'il nous faut pour l'ensemble des attributs

Sauvegarder Ou Ajouter 1 colonne(s) Exécuter

Voici la structure de notre base de données :

Table	Action	Lignes	Type	Interclassement	Taille	Perte
classe	Afficher Structure Rechercher Insérer Vider Supprimer	5	InnoDB	latin1_swedish_ci	16,0 Kio	-
historique	Afficher Structure Rechercher Insérer Vider Supprimer	17	InnoDB	latin1_swedish_ci	16,0 Kio	-
plagehoraire	Afficher Structure Rechercher Insérer Vider Supprimer	2	InnoDB	latin1_swedish_ci	16,0 Kio	-
salles	Afficher Structure Rechercher Insérer Vider Supprimer	2	InnoDB	latin1_swedish_ci	16,0 Kio	-
statut	Afficher Structure Rechercher Insérer Vider Supprimer	0	InnoDB	latin1_swedish_ci	16,0 Kio	-
utilisateurs	Afficher Structure Rechercher Insérer Vider Supprimer	8	InnoDB	latin1_swedish_ci	16,0 Kio	-
6 tables	Somme	34	InnoDB	latin1_swedish_ci	96,0 Kio	0 o

Tout cocher / Tout décocher Pour la sélection :

Nous avons donc ensuite créé plusieurs utilisateurs pour la phase de tests futurs ainsi que plusieurs classes, car le projet s'étend sur plusieurs salles de classes, la salle de travail et le foyer et pour finir plusieurs plages horaires pour gérer l'accès

Voici un exemple pour la table « Utilisateur » :

←T→		idUtilisateur	nom	prenom	idClasse	codeRFID	login	motPasse	idStatut
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer		1	Baudelet	Maxence	0	123456789123	admin	admin	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer		2	hamouten	yassine	1	124561789153	y.hamouten	yassine	3
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer		3	Et-Touil	abderrahmane	1	0A008CD24713	a.et-touil	test	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer		4	test	test	1	123456789102	t.test	test	1
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer		5	test2	test2	1	123456789103	t.test2	test2	1
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer		6	test3	test3	1	123456789104	t.test3	test3	1
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer		7	test4	test4	1	123456789105	t.test4	test4	1
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer		8	Chatouf	souli	1	147258369789	s.chatouf	test5	1

↑ Tout cocher / Tout décocher Pour la sélection : Modifier Effacer Exporter  
 Afficher :  ligne(s) à partir de la ligne n°  en mode  et répéter les en-têtes à chaque groupe de

La base de données a été créée au départ en localhost à partir d'un WAMPP, il y a deux choses qui expliquent cela :

- Mes camarades n'avaient pour le moment pas besoin d'accéder à la base de données, de ce fait, l'avoir en localhost pour réaliser mes quelques petits tests était amplement suffisant.
- Deuxièmement, aucun serveur ni ordinateur n'étaient disponibles pour y placer mon serveur sans être assuré qu'ils n'y aurait aucun problème (coupure etc.). Nous avons donc opté pour cette solution afin de ne pas être gêné au niveau des tests.

La difficulté dans l'élaboration de la base de données résidait dans le fait de choisir le bon type pour chaque attribut de la table.

## Présentation du serveur WAMP



Tout d'abord, l'acronyme WAMP signifie: Windows Apache MySQL PHP.

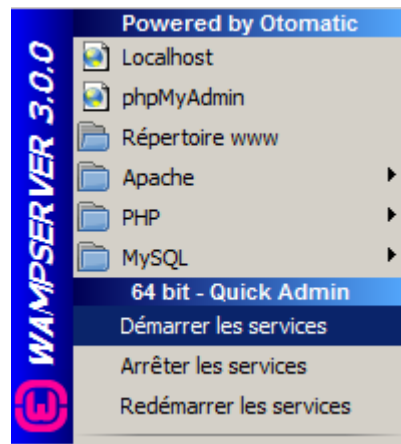
WampServer est une plate-forme de développement Web sous Windows pour des applications Web dynamiques à l'aide du serveur Apache2, du langage de scripts PHP et d'une base de données MySQL. Il possède également PHPMyAdmin pour gérer plus facilement vos bases de données.

J'aurais pu aussi utiliser un LAMP, qui est basé sur le même principe mais sous linux, je n'ai tout simplement pas utilisé de machine virtuelle.

Le lycée ayant des problèmes pour fournir un serveur , nous avons donc décidé d'utiliser l'un des serveurs personnels que nous avons car devoir réinstaller à chaque début de séance le WAMP puis importer la base de données que j'avais enregistré au préalable et au final, pour l'ensemble des séances, le temps perdu était considérable.

## - Utilisation du WAMP

### - Démarrage



Une fois le serveur démarré, nous nous rendons sur PHPMYADMIN pour gérer notre base de données.

### - Comment importer une base de données ?



On sélectionne ensuite le fichier dans notre disque dur :

Parcourir :  Aucun fichier choisi (Taille maximum: 1 024Mio)

On l'exécute ensuite et la base de données est lancée. Il faut au préalable créer une base de données pour l'importation, sans cela, on retourne un échec comme ici :

## Erreur

Requête SQL:

```
--  
-- Base de données : `gestion_accès`  
--
```

```
-----
```

```
--  
-- Structure de la table `classe`  
--
```



```
CREATE TABLE IF NOT EXISTS `classe` (
```

```
.....  
  `idClasse` INT( 11 ) NOT NULL AUTO_INCREMENT ,  
  `nomClasse` TEXT NOT NULL ,  
  PRIMARY KEY ( `idClasse` )  
) ENGINE = INNODB DEFAULT CHARSET = latin1 AUTO_INCREMENT = 1;
```

MySQL a répondu: 🐞

#1046 - No database selected



## 2. Application Utilisateur

Une fois la base de données terminée, je me suis donc orienté vers ma partie du travail sur le client lourd, car en effet le client lourd est divisé en deux parties, l'une pour l'étudiant 2 et l'autre pour l'étudiant 3. Le travail à réaliser y est sensiblement identique, avec des tables et attributs différents.

### - Le logiciel de programmation utilisé



Le langage de programmation devant être le C#, nous avons donc tout naturellement opté pour le logiciel de développement Windows.

Le seul problème notable est que le logiciel ne possède pas les connecteurs MYSQL pour la base de données, nous verrons plus tard comment nous y avons remédié.

Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles. Basic, Visual, Visual C# utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages.

### - Pourquoi un client lourd et non un client léger pour l'application utilisateur ?

#### Caractéristique d'un client lourd :

Un client lourd est un logiciel qui propose des fonctionnalités complexes avec un traitement autonome. La notion de client s'entend dans une architecture client-serveur. Contrairement au client léger, le client lourd ne dépend du serveur que pour l'échange des données dont il prend généralement en charge l'intégralité du traitement.

#### Caractéristique d'un client léger :

L'usage veut qu'une application en client léger n'impose à l'utilisateur que d'avoir un navigateur Web.

La logique qui prévaut au déploiement de clients légers est une logique essentiellement économique. Il s'agit de réduire le coût total de possession et de gestion. Il ne faut toutefois pas oublier une certaine dimension écologique dans le cas du recyclage de vieux ordinateurs en clients légers matériels.

Le fait de choisir un client lourd en lieu et place d'un client léger est purement spécifié comme obligation dans le cahier des charges.

Les conditions à respecter dans le cahier des charges pour la réalisation de l'application utilisateur mise à part l'utilisation d'un client lourd :

- Langage de programmation : C#
- Couche DAL

### **- Comment modifier la base de données à partir d'un client lourd ?**

Ce qu'on l'on cherche à faire, c'est modifier la base de données sans écrire en brut dessus comme on peut le faire sur PHPMYADMIN.

Nous avons donc utilisé une couche DAL comme imposé dans le projet.

### **Qu'est-ce qu'une couche DAL ?**

Tout d'abord, DAL est l'acronyme de « Data Access Layer » ou en français « couche d'accès aux données ».

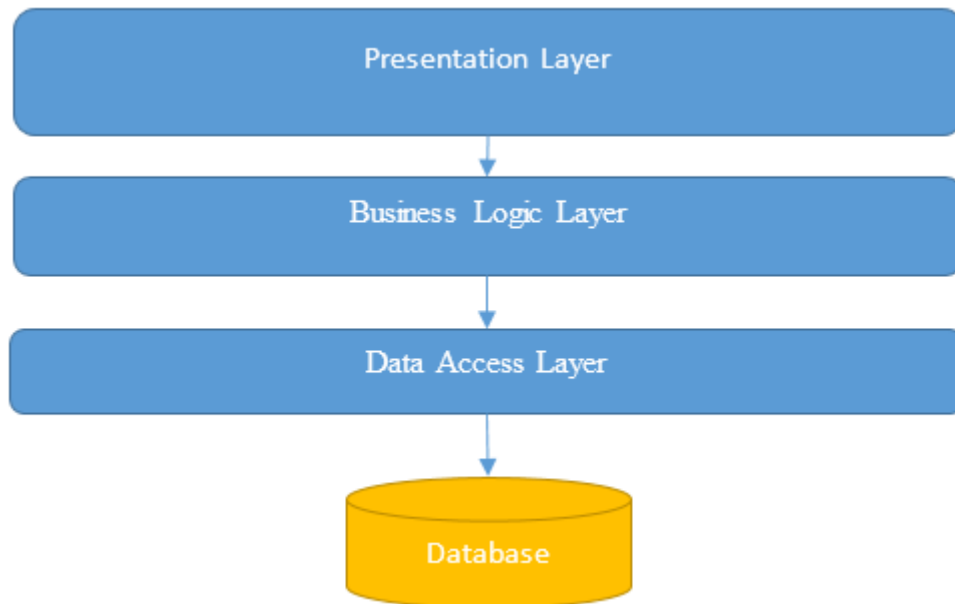
C'est le composant logiciel réalisant l'interface et la séparation entre une base de données et les composants de haut niveau les exploitant.

Basiquement, la DAL va donc mettre à disposition des méthodes permettant d'ajouter, mettre à jour, lire, supprimer un enregistrement, et ce quel que soit le support de stockage des données.

### **- Comment fonctionne les différentes couches ?**

- les couches les plus connues sont : l'interface homme-machine (IHM), les processus métiers (Business Logic Layer, BLL), l'accès aux données (Data Access Layer, DAL) ;
- chaque couche regroupe les composants (objets, méthodes) partageant les mêmes fonctionnalités, les mêmes rôles. Ainsi, en respectant cette architecture, on ne devrait jamais voir une seule requête SQL dans une page Web ;
- l'implémentation d'une couche métier indépendante de l'interface permet de créer facilement une couche de services destinés à l'IHM ou d'autres logiciels ;
- chaque couche ne peut communiquer qu'avec celle qui lui est immédiatement voisine ;
- la modularité et le faible couplage de l'approche multicouche sont une formidable contrepartie à l'apparente complexité de son implémentation.

### - Schéma de l'architecture des couches



### - L'avantage du développement sur différentes couches

Il y a plusieurs avantages à utiliser cette technique :

- La maintenance des données est indépendante du support physique de stockage
- La maintenance des traitements est simplifiée
- La gestion des traitements depuis la couche de présentation est facilitée
- Le travail en équipe est optimisé
- La migration d'un environnement graphique à un autre est relativement simple

### - Comment les informations sont-elles envoyées ?

#### - Utilisation de DTO

« DTO » qui signifie « Data Transfert Object », est un objet qui définit comment les données seront envoyées à la base de données

Tout d'abord, les DTO sont conçus pour se déplacer entre des couches de l'application. Donc, ils n'ont pas leur place dans la DAL. C'est donc grâce à ces derniers que les couches communiquent entre-elle.

L'utilisation de DTO est plus adapté car c'est le conteneur le plus léger et le plus adapté.

- Exemple : DTOPlagehoraire

```
public class DTOPlageHoraire
{
    public int idPlageHoraire { get; set; }
    public int idSalle { get; set; }
    public int idStatut { get; set; }
    public float debut { get; set; }
    public float fin { get; set; }
    public string jour { get; set; }
}
```

La présence des « getter » et des « setter » s'explique car nous voulons soit récupérer une information à partir de la base de données, soit ajouter dans la base de données

## Classe DAL Base

Voici le code de la classe DALBase :

```
namespace GestionAcces
{
    public class DALBase
    {
        public MySqlConnection basededonnee;
        //string database = "foyer";
        //string userDB = "admin";
        //string passwordDB = "admin";

        protected string connection = "server=37.187.118.104;user id=admin; password=admin; database=gestionacces; convertzerodatetime=True; allowzerodatetime=True";

        public DALBase()
        {
            //this.basededonnee = new MySqlConnection("Database=" + database + ";server=127.0.0.1;user id=" + userDB + ";pwd=" + passwordDB + "");
            this.basededonnee = new MySqlConnection(connection);
            try
            {
                //this.basededonnee.Open(); // Enlever car erreur lors de la compilation
            }
            catch (MySqlException ex)
            {
            }
        }

        ~DALBase()
        {
            //this.basededonnee.Close();
        }
    }
}
```

Le rôle de cette classe est d'établir la connexion à la base de données.

On peut voir que la chaîne de connexion soulignée en noir était la première que j'utilisais mais que j'ai changée car je trouve la première plus adaptée.

La présence de « convertzerodatetime » et de « allowzerodatetime » s'explique car sans elle, il y a fréquemment des erreurs au niveau du chiffrement des données.

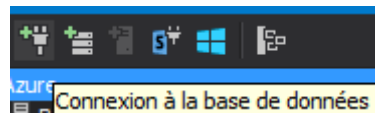
### - Comment se connecter directement à la base de données à partir de Visual Studio ?

En attendant d'avoir notre serveur personnel actif, nous avons utilisé l'outil de connexion à la base de données que nous offre Visual Studio pour effectuer divers tests.

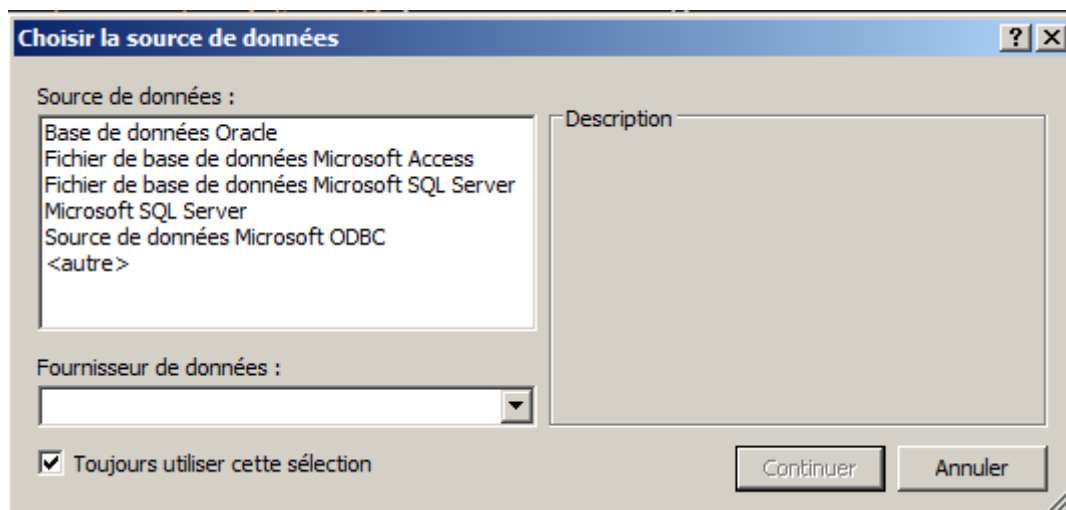
Pour se connecter, il faut tout d'abord se rendre dans l'explorateur de serveurs :



Ensuite, sélectionner l'outil « Connexion à la base de données »



On arrive donc ensuite sur la fenêtre où l'on choisit la source des données, c'est-à-dire qu'elle genre de base de données nous utilisons :

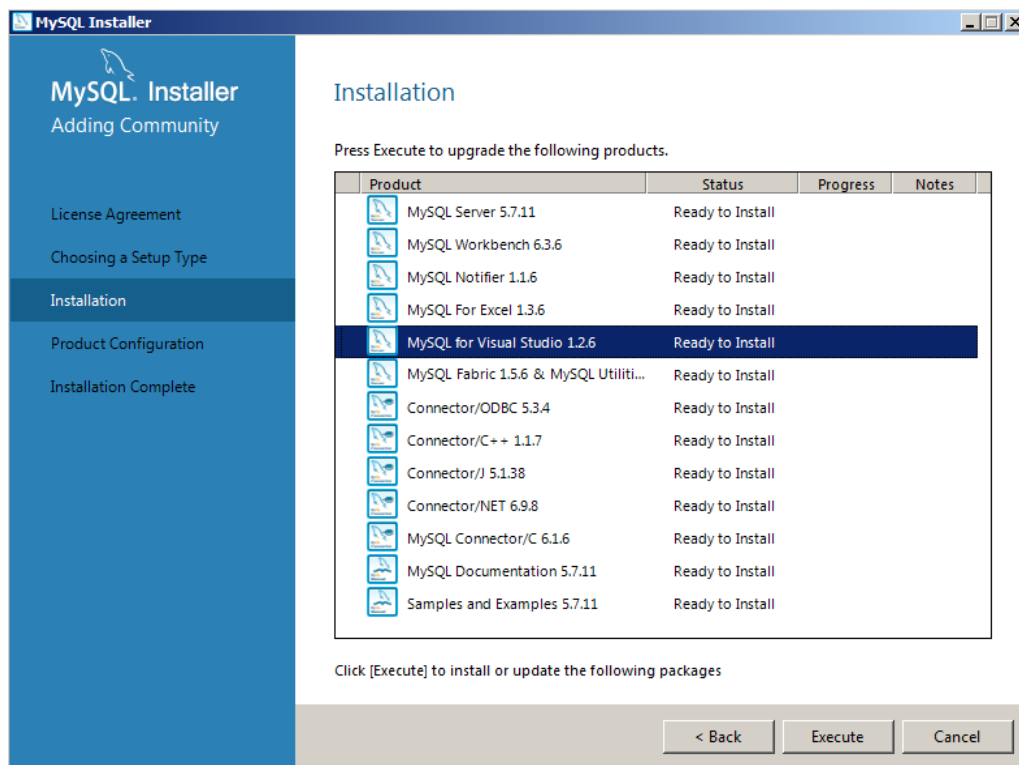


On peut donc voir que Visual Studio ne reconnaît pas la base de données MySQL, pour y remédier, nous devons donc passer par l'utilisation d'un connecteur.

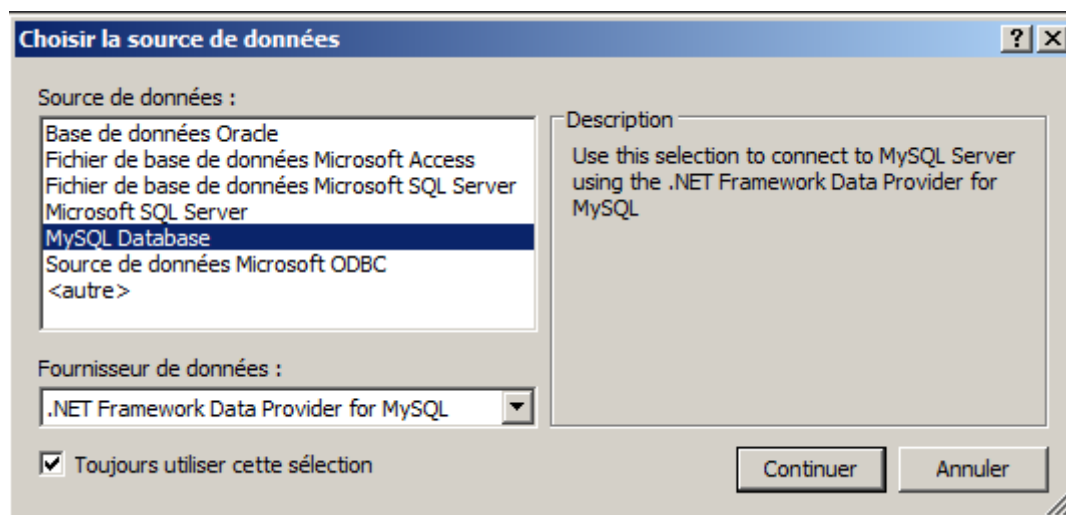
### - Connecteur MySQL

Le connecteur est utilisé pour que le logiciel Visual Studio puisse afficher la source de données : MySQL

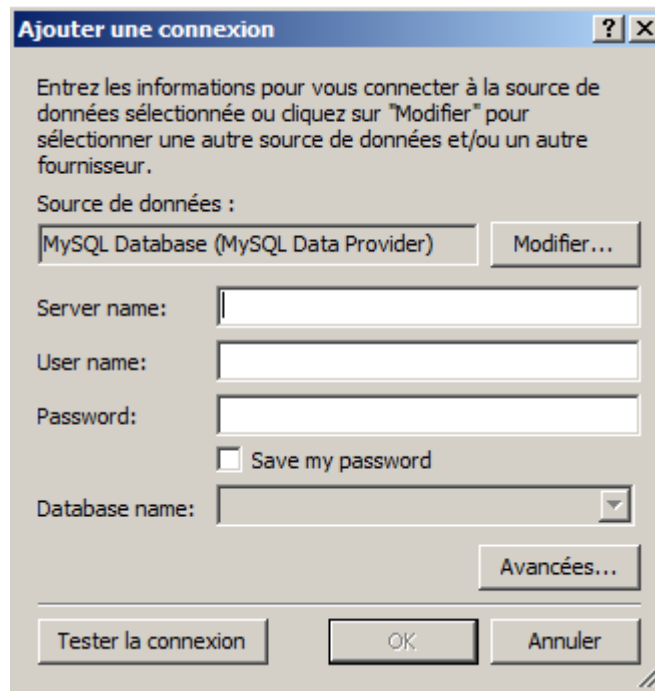
On sélectionne donc la version pour visual studio :



Une fois téléchargé, on relance l'application, on refait le même processus que précédemment et on peut se rendre compte que désormais, la source de données MYSQL apparait.



On rentre donc ensuite tous les paramètres du serveur afin de s'y connecter :



Nous voici donc désormais connecter à notre base de données à partir de Visual Studio

### Comment modifier la base de données à partir de l'application ?

Comme vu précédemment, la DAL met à disposition des méthodes pour ajouter, modifier, supprimer ou encore lire des informations dans la base de données.

Ces méthodes sont appelées les CRUD.

CRUD est l'acronyme de « CREATE READ UPDATE DELETE ».

Comme vu dans le cahier des charges, je suis responsable de la gestion des plages horaires ainsi que de la gestion des salles. J'ai donc créé un DTO et un DAO pour chacun des cas.

Le principe du pattern DAO « Data Access Layer » est de créer une couche gérant le stockage des données, logiquement nommée couche de données. Il s'agit là des opérations classiques de stockage : la création, la lecture, la modification et la suppression.

### - Exemple : La classe DAO Plage horaires

Voici quelques exemples de la classe DAO plage horaire au niveau des opérations de types CRUD :

- La méthode AddPlagesHoraires :

```
public void AddPlageHoraires(string plagehoraire, float debut, float fin, string jour)
{
    DTOPlageHoraire DTOPlageHoraire = null;
    DTOPlageHoraire = new DTOPlageHoraire();
    MySqlCommand cmd = new MySqlCommand("INSERT INTO plagehoraire (`idClasse`, `idSalle`, 'idStatut',
cmd.Parameters.AddWithValue("@debut", debut);
cmd.Parameters.AddWithValue("@fin", fin );
cmd.Parameters.AddWithValue("@jour", jour);
    MySqlDataAdapter da = new MySqlDataAdapter(cmd);
    //da.Fill(DTOPlageHoraire);
}
```

- La méthode DeletePlagesHoraires :

```
public void DeletePlageHoraires(string plagehoraire, float debut, float fin, string jour)
{
    DTOPlageHoraire DTOPlageHoraire = null;
    DTOPlageHoraire = new DTOPlageHoraire();
    MySqlCommand cmd = new MySqlCommand("DELETE FROM plagehoraire (`idSalle`, `idClasse`, 'idStatut',
cmd.Parameters.Remove("@debut");
cmd.Parameters.Remove("@fin");
cmd.Parameters.Remove("@jour");
    MySqlDataAdapter da = new MySqlDataAdapter(cmd);
    //da.Fill(DTOPlageHoraire);
}
```

Au niveau de la récupération de données (Read) tout cela se passe dans le datagridview comme expliqué juste après.

Concernant l'Update, je n'y ai pas encore travaillé pour le moment, c'est la partie manquante de mon application.



### 3. Serveur TCP en C#

A la base, le serveur devait être développé en C++ mais après discussion avec le professeur, il a décidé de modifier le cahier des charges. Nous avons donc développé le serveur en langage C#.

Le but étant de recevoir la requête du client. C'est une requête d'accès à l'une des salles. Nous recevons donc un code RFID et nous devons interroger la base de données sur deux choses :

- Est-ce que l'utilisateur a accès à cette salle et si oui, la plage horaire convient-t-elle ?

Il ne faut pas oublier aussi que nous devons historier chaque demande d'accès.

Tout d'abord, voici l'initialisation des paramètres :

```
public partial class Form1 : Form
{
    byte[] requete = new byte[1000];
    byte[] reponse;
    int port;
    TcpListener serv;
    TcpClient client;
    NetworkStream str;
    public Form1()
    {
        InitializeComponent();
    }
}
```

Nous utilisons TcpListener pour le serveur et TcpClient pour le client

Voici ensuite le code qui permet de créer le serveur, de le démarrer.

Nous voyons aussi la variable client qui attend la connexion et le flux d'échange opérationnel

```
while (true)
{
    try
    {
        serv = new TcpListener(5002); // création serveur
        serv.Start(); // démarrage
        client = serv.AcceptTcpClient(); // attente connexion
        str = client.GetStream(); // flux pour les échanges
    }
}
```

Ensuite, on lit la requête entrante du client et on ouvre la connexion (flux d'échange) avec ce dernier ainsi qu'avec la base de données.

```
try
{
    byte[] requete = new byte[1000];
    str.Read(requete, 0, 1000); // lire la requête (bloquant) donc on attend la requête
    string code = System.Text.Encoding.ASCII.GetString(requete);
    MessageBox.Show(code);

    string connStr = "server=37.187.118.104;user=gestionaccs;database=gestionaccs;port=3306;password=gestionaccs;";
    MySqlConnection conn = new MySqlConnection(connStr);
```

Nous devons ensuite voir ce qu'il arrive, que l'accès soit accordé ou non, historiser l'accès dans la base de données. Deux champs seront remplis :

- le Code RFID de la personne qui a essayé d'entrer dans la salle
- L'Etat de la requête, 1 pour accorder et 0 pour refusé

Pour que l'accès soit autorisé, il faut que leur de début soit inférieur à l'heure actuelle et que l'heure de fin soit inférieur à l'heure actuelle.

```
try
{
    conn.Open();

    string sql = "INSERT INTO gestionaccs.historique (`codeRFID`, `etat`) VALUES ('" + code + "', '1');";
    MySqlCommand cmd = new MySqlCommand(sql, conn);
    MySqlDataReader rdr = cmd.ExecuteReader();
```

On cherche ensuite à vérifier dans la base de données si l'utilisateur en question peut accéder à la salle

On récupère les informations à partir du code RFID

```
string sql2 = "SELECT * FROM plagehoraire, classe, utilisateurs WHERE plagehoraire.idClasse = classe.idClasse AND classe.idClasse = utilisateurs.idClasse AND codeRFID = '" + code + "'";
MySqlCommand cmd2 = new MySqlCommand(sql2, conn);
MySqlDataReader reader = cmd2.ExecuteReader();
```

Voici la suite de la requête :

```
= classe.idClasse AND classe.idClasse = utilisateurs.idClasse AND codeRFID = '" + code + "'";
```

Concernant la réponse que le serveur doit envoyer au client pour savoir si l'accès est autorisé ou non, nous avons décidé de nous répondre 'Ok' si c'est autorisé ou 'NOK' si c'est refusé

```
try
{
    reponse = System.Text.Encoding.ASCII.GetBytes("ok");
    str.Write(reponse, 0, reponse.Length); // envoi de la réponse
}
```

## 4. Test unitaire

### Tentative d'historisation de la base de données après une connexion

Test Historisation de la base de données
Méthode de test
1 – On lance le serveur et on établit la connexion avec le client (lecteur RFID). 2 – On passe le badge RFID sur le lecteur et on envoie l'information au serveur 3 – On historise la requête dans la base de données avec le code RFID et l'état de la requête
Résultats attendus :
La base de données a correctement été mise à jour
Validation
Résultat

Voici le résultat de toutes nos tentatives dans la base de données

← T →	idHistorique	codeRFID	dateTentative	etat
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	1	124561789153	2016-04-01 08:23:00	1
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	2	124561789153	2016-04-02 07:36:47	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	3	132152454544	2016-04-12 00:00:00	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	4	132152454544	2016-04-12 00:00:00	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	5	132152454544	2016-04-12 00:00:00	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	6	0A008CD24713	2016-04-12 00:00:00	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	8	0A008CD24713	2016-04-12 00:00:00	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	10	0A008CD24713	2016-04-12 00:00:00	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	11	0A008CD24713	2016-04-26 10:38:09	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	12	0A008CD24713	2016-05-10 10:54:11	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	14	0A008CD24713	2016-05-10 10:54:21	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	16	0A008CD24713	2016-05-10 10:54:45	0
<input type="checkbox"/> Modifier  Éditer en place  Copier  Effacer	18	0A008CD24713	2016-05-12 12:30:28	0

☐ Tout cocher / ☐ Tout décocher Pour la sélection : Modifier Effacer Exporter

## Vérification de la trame reçue par le serveur à partir de Wireshark

### - Présentation de Wireshark

Wireshark est un [analyseur de paquets libre](#) utilisé dans le dépannage et l'analyse de [réseaux informatiques](#), le développement de [protocoles](#), l'éducation et la rétro-ingénierie

Wireshark utilise la [bibliothèque logicielle GTK+](#) pour l'implémentation de son [interface utilisateur](#) et [pcap](#) pour la capture des [paquets](#)

On nous a demandé d'utiliser ce logiciel dans le cahier des charges pour voir si mon serveur recevait bien les requêtes d'un client, qui signifie la demande d'accès à une salle et qu'il y avait bien une mise à jour de l'historique dans la base de données.

```
5141 470.5245570(172.16.32.191) 172.16.32.24 TCP 1090 55194 > rfe [PSH, ACK] Seq=1025 Ack=1 Win=4344 Len=1024 TSval=694700 TSecr=1293641
```

Voici ci-dessus la trame que l'on reçoit lorsque l'on passe le badge RFID sur le lecteur.

Ensuite, on reçoit continuellement des trames qui signifient que la connexion est faite entre le serveur et le client.

La commande a tapé dans WIRESHARK est :

ipaddr.source== avec ici l'adresse du client.

### Ajouter une plage horaire à partir de l'application

Test Historisation de la base de données
Méthode de test
1 – On se rend dans l'ajout de plages horaires 2 – On n'y rentre les informations que l'on a choisies 3 – On vérifie dans la base de données si l'ajout à été effectué
Résultats attendus :
La base de données à correctement était mise à jour
Validation
Résultat

### Démonstration avec l'application :

On remplit tout d'abord les informations dans les textbox dédiée :

Ajouter une plage horaire

Heure de début      Heure de fin

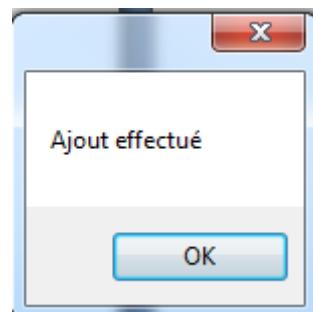
8.00      10.30

Jour

jeudi

Ajouter à la base de données

Une fois que j'ai cliqué sur le bouton Ajouter à la base de données, voici le message de confirmation



On consulte la base de données et on voit que la plage horaire a été ajoutée.

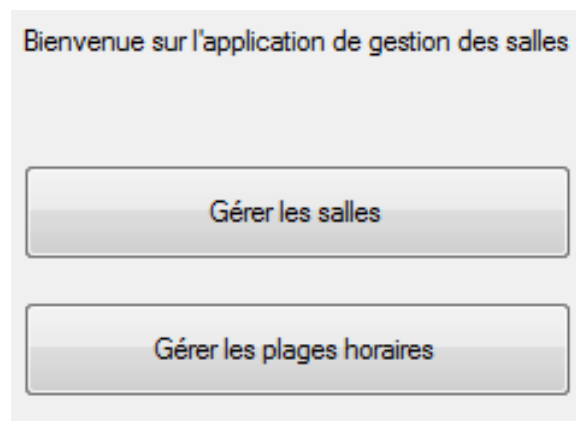
				idPlageHoraire	idClasse	idSalle	idStatut	debut	fin	jour
				1	0	0	0	8.00	10.30	jeudi

## 5. Démonstration de l'application

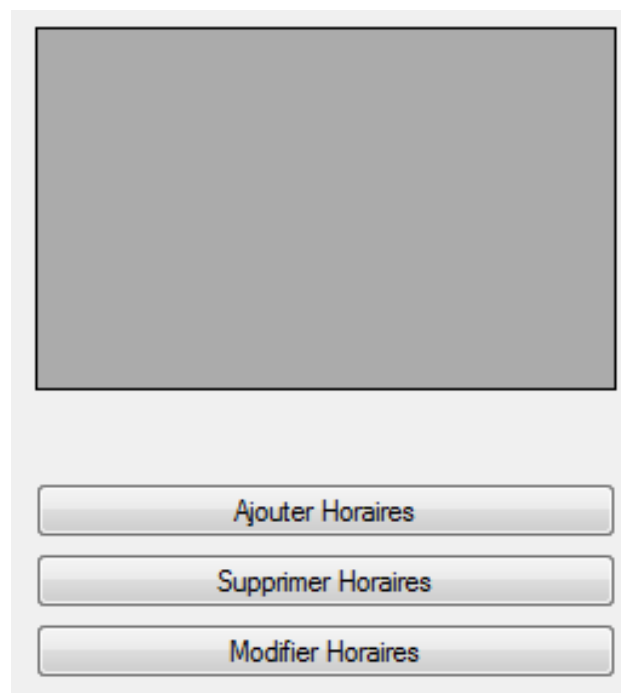
Voici l'interface d'utilisation de mon application pour le moment, elle sera améliorée une fois que le développement sera terminé :

La gestion de la connexion n'est pas mon travail donc je pars du principe que la personne est connectée en mode admin

Voici l'accueil après la connexion en mode admin :



On dispose de deux choix, les deux ont des interfaces identiques. Comme dit précédemment, on veut faire des opérations de types CRUD donc on possède un bouton pour chaque opérations. Le Read se fait dans le data grid view qui affichera l'ensemble des plages horaires disponibles dans la base de données.



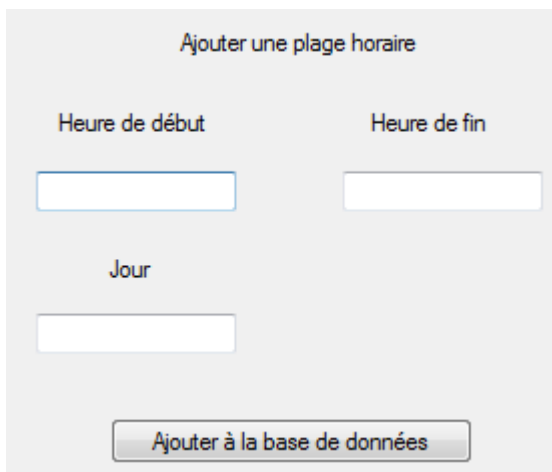
Voici le code qui permet d'afficher les informations de la base de données dans le datagridview.

```
public void getData()
{
    //connexion a la base de données
    string config =
"server=37.187.118.104;userid=gestionacces;database=gestionacces;pwd=gestionacces;Con
vert Zero Datetime=True";
    MySqlConnection connexion = new MySqlConnection(config);
    connexion.Open ();
    MySqlDataAdapter adaptateur = new MySqlDataAdapter("SELECT * FROM
plagehoraire", connexion);
    DataSet data = new DataSet ();
    adaptateur.Fill(data);
    dataGridView1.DataSource = data.Tables[0];
}
```

Et enfin pour ajouter une plage horaire, chaque texte écrit dans le text box correspondante sera ajouté à la base de données.

S'il y a une erreur, il y a un message qui s'affiche avec le message « Echec, Veuillez réessayer » à partir d'un messagebox.

MessageBox.Show(« Echec, veuillez réessayer ») ;



## 6. Conclusion

Après 4 mois de projet, l'application est bientôt terminée et mon serveur est prêt à l'utilisation même si plusieurs améliorations peuvent y être ajoutées.

J'ai pris beaucoup de retard sur les couches DAL, n'ayant pas les connaissances nécessaires pour mener à bien ce projet j'ai dû m'aider d'un cours d'internet, cependant, l'utilisation des procédures stockées m'a beaucoup dérangé donc je suis revenu finalement sur une utilisation plus simple des couches DAL en écrivant directement les requêtes MYSQL même si les procédures stockées sont plus facile et moins contraignantes d'utilisation.

Je pense que l'application sera terminée d'ici la date de revue de projet finale.

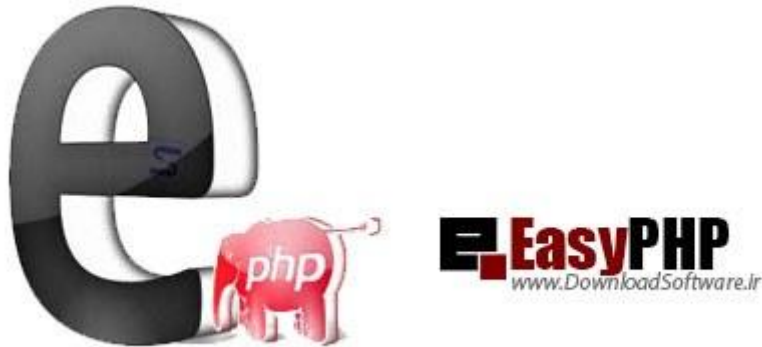
Personnellement, ce projet m'a appris beaucoup de choses dans le développement d'une application et toutes les contraintes qui y sont accordés notamment la notion de temps et des dates dans la remise de projet. L'aspect professionnel m'a permis de me mettre dans les conditions du monde du travail et d'engranger un peu d'expérience



CLIENT LOURD  
(ABDERRAHMANE ET-TOUIL)

## 1. Outils de développement

### EasyPHP



Comme son nom l'indique, EasyPHP permet de simplifier le travail sur les fichiers au format PHP. En effet, ce format nécessite d'être interprété, ce qu'un simple navigateur Web ne peut pas faire. EasyPHP vous offre donc la possibilité de travailler dans un environnement serveur complet. L'application comprend un serveur Apache, une base de données MySQL, la dernière version de PHP, et tous les outils nécessaires pour travailler sur le code PHP et créer votre site.

### Visual studio



Microsoft Visual Studio est une suite de logiciels de développement pour Windows conçue par Microsoft. La dernière version s'appelle Visual Studio 2015. Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C# utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications web ASP et de services web XML grâce à Visual Web Developer.

## CSHARP(C#)

C# est un langage de programmation orienté objet, fortement typé, dérivé de C et C++, ressemblant au langage Java. Il est utilisé pour développer des applications web, ainsi que des applications de bureau, des services web, des commandes, des widgets ou des bibliothèques de classes. En C# une application est un lot de classes où une des classes comporte une méthode *Main*, comme cela se fait en Java.

C# est destiné à développer sur la plateforme .NET, une pile technologique créée par Microsoft pour succéder à COM.

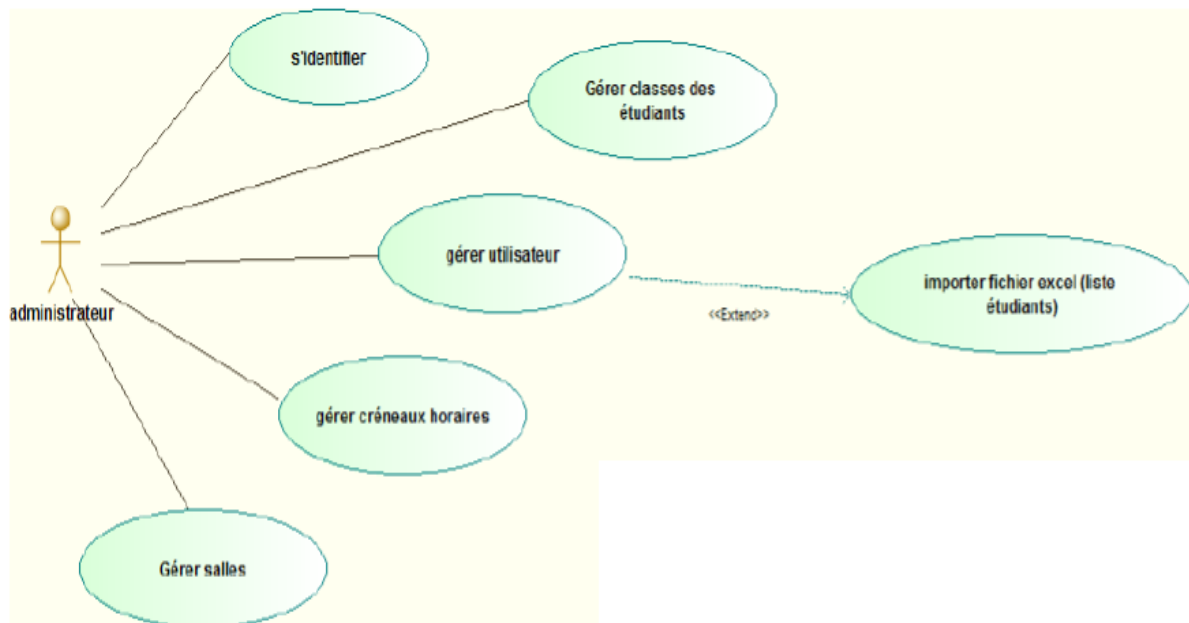
Les exécutables en C# sont subdivisés en assemblies, en namespaces, en classes et en membres de classe. Un assembly est la forme compilée, qui peut être un programme ou une bibliothèque de classes. Un assembly contient le code exécutable en MSIL, ainsi que les symboles. Le code MSIL est traduit en langage machine au moment de l'exécution par la fonction *just-in-time* de la plateforme .NET.

## 2. Gestion d'administrateur

Dans cette partie on abordera:

- La gestion des droits d'accès à l'IHM d'administration des utilisateurs (droits administrateur)
- La gestion des utilisateurs : création, modification, suppression

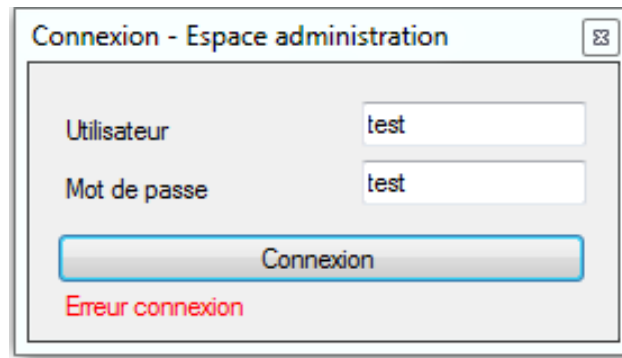
## Authentification de l'administrateur



L'écran d'accueil invite l'administrateur à saisir son login et mot de passe

Image d'un écran de connexion intitulé "Connexion - Espace administration". Le formulaire contient deux champs de saisie : "Utilisateur" et "Mot de passe". En dessous de ces champs se trouve un bouton "Connexion".

Si le couple Utilisateur/Mot de passe n'existe pas en base, le message d'erreur suivant s'affiche :



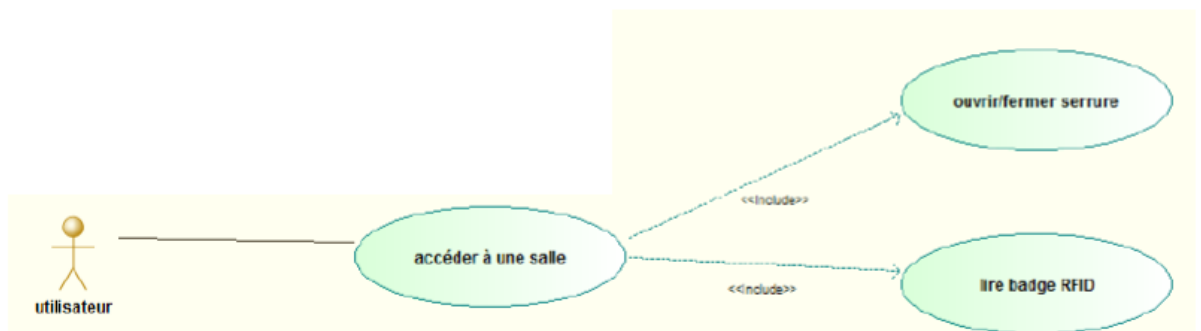
Si le couple utilisateur/ Mot de passe existe en base de données, alors l'administrateur est redirigé vers l'écran de gestion des utilisateurs.

Le code gestionnaire de l'authentification administrateur

```
private void BT_Connexion_Click(object sender, EventArgs e)
{
    DataTable getTable = dalUsr.getUserbyLog(TB_Utilisateur.Text, TB_MDP.Text);

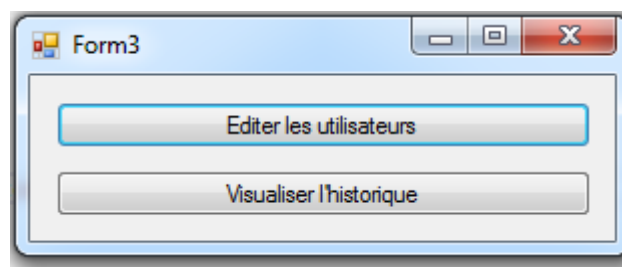
    if (getTable != null)
    {
        Form3 Form3 = new Form3();
        Form3.Show();
        this.Hide();
    }
    else
    {
        msg_erreur.Text = "Erreur connexion";
    }
}
```

### 3. Gestion des utilisateurs



L'écran de gestion des utilisateurs ci-dessous propose deux fonctionnalités

- 1- La fonctionnalité « éditer les utilisateurs » permettant la création, la modification et la suppression des utilisateurs
- 2- La fonctionnalité « Visualiser l'historique » permettant la traçabilité des accès utilisateurs



### a- La fonctionnalité « Editer les utilisateurs »

Le clic sur le bouton « éditer les utilisateurs » donne accès à la liste des utilisateurs habilités

The screenshot shows a web application window titled "Gestion des utilisateurs". Inside, there is a section labeled "Import utilisateurs" containing a table with the following columns: idUtilisateur, nom, prenom, idClasse, codeRFID, login, motPasse, and idStatut. The table lists 10 users, with the first user (id 1, nom Baudalet, prenom Maxence) highlighted in blue. To the right of each row is a red 'X' icon. Below the table is a large grey rectangular area. At the bottom of the window, there is a blue button labeled "Enregistrer les modifications" and a smaller button labeled "Retour menu principal".

	idUtilisateur	nom	prenom	idClasse	codeRFID	login	motPasse	idStatut
▶	1	Baudalet	Maxence	0	123456789123	admin	admin	0
	2	hamouten	yassine	1	124561789153	y.hamouten	yassine	3
	3	Et-Touil	abderrahmane	1	0A008CD24713	a.et-touil	test	1
	4	test222	test	1	123456789102	t.test	test	1
	5	test2	test2	1	123456789103	t.test2	test2	1
	6	test3	test3	1	123456789104	t.test3	test3	1
	7	test4	test4	1	123456789105	t.test4	test4	1
	8	Chatouf	souli	1	147258369789	s.chatouf	test5	1
	9	test2	test	1	121454547487	schatouf	test	1
	10	Chevalier	Kevin	2	012211545457	k.chevalier	test	1
*								

Sur cet écran, l'administrateur a la possibilité d'exécuter toutes les actions du CRUD :

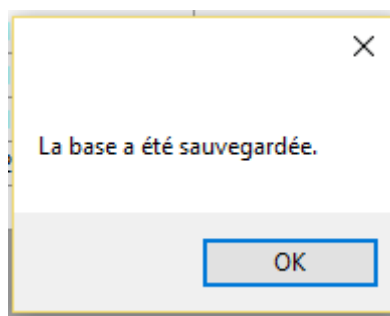
→ Créer (**Create**) un nouvel utilisateur :

- idUtilisateur : identifiant unique attribué à l'utilisateur
- nom
- prénom
- idClasse : l'identifiant de la classe d'appartenance de l'utilisateur
- codeRFID
- login : login de l'utilisateur
- motPasse : mot de passe utilisateur
- idStatut : le statut de l'utilisateur (personnels et étudiants ...)


→ Modifier (**Update**) un utilisateur existant :

L'administrateur peut modifier directement dans la grille toute les informations utilisateur et valider ses modifications à l'aide du bouton **Enregistrer les modifications**.

Une fois les informations enregistrées en base de données. Un pop-up s'affiche informant l'administrateur du succès de l'action



→ Supprimer (Delete) un utilisateur :

L'administrateur peut supprimer un utilisateur en cliquant sur le pictogramme  associé à l'utilisateur à supprimer.

Une fois l'utilisateur supprimé la liste des utilisateurs est rafraîchie.

→ Import en masse des utilisateurs :

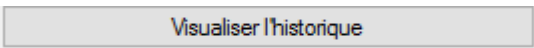
Pour faciliter la création des utilisateurs, l'administrateur peut importer une liste d'utilisateurs à partir d'un fichier au format CSV, on veille à ce que les informations issues du fichier CSV respectent certaines règles (contrôle de format) avant l'insertion en base de données.

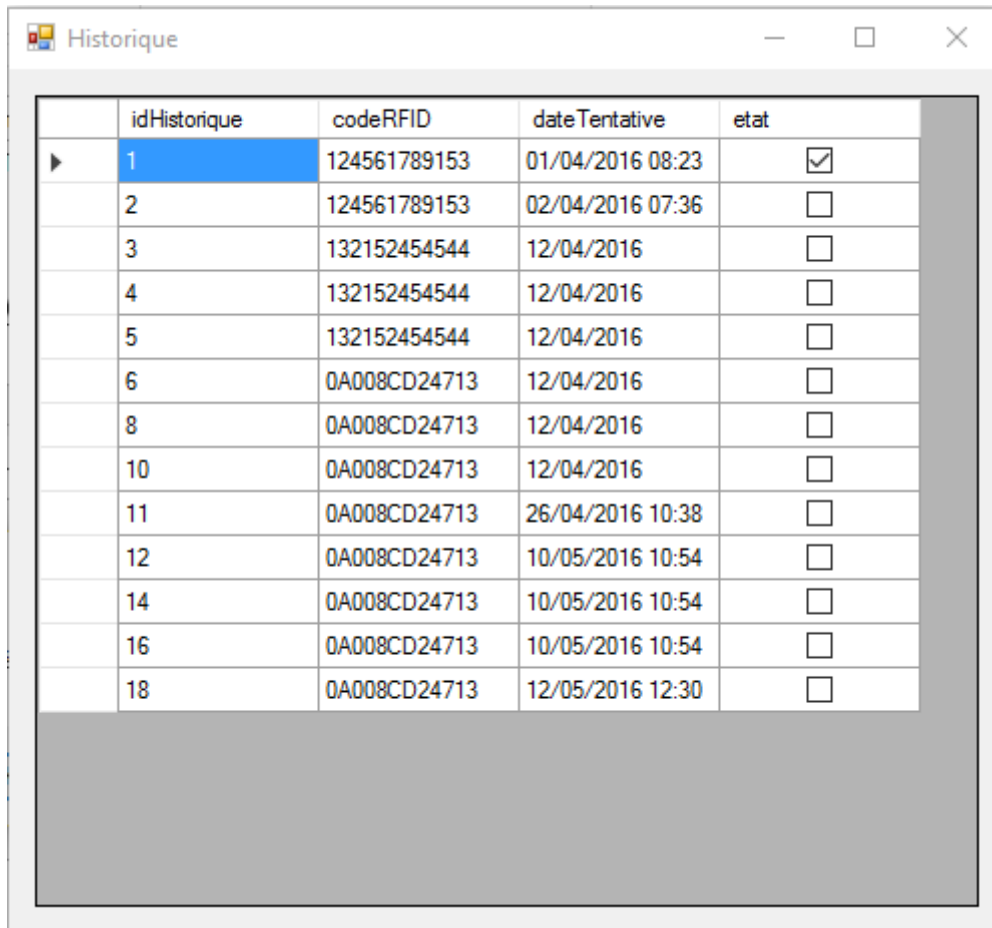
Pour se faire l'administrateur dispose d'un bouton « importer » sous le menu « import utilisateur »

Import utilisateurs			
	Importer	nom	prenom
	1	Baudelet	Maxence
	2	hamouten	yassine
	3	Et-Touil	abderrahmane



**b- La fonctionnalité « Visualiser l'historique »**

Le clic sur le bouton  permet d'ouvrir une fenêtre qui donne accès à l'historique des utilisateurs comme ci-dessous



The screenshot shows a window titled 'Historique' with a table of access attempts. The table has four columns: 'idHistorique', 'codeRFID', 'dateTentative', and 'etat'. The first row is highlighted in blue. The 'etat' column contains checkboxes, with the first one checked.

	idHistorique	codeRFID	dateTentative	etat
▶	1	124561789153	01/04/2016 08:23	<input checked="" type="checkbox"/>
	2	124561789153	02/04/2016 07:36	<input type="checkbox"/>
	3	132152454544	12/04/2016	<input type="checkbox"/>
	4	132152454544	12/04/2016	<input type="checkbox"/>
	5	132152454544	12/04/2016	<input type="checkbox"/>
	6	0A008CD24713	12/04/2016	<input type="checkbox"/>
	8	0A008CD24713	12/04/2016	<input type="checkbox"/>
	10	0A008CD24713	12/04/2016	<input type="checkbox"/>
	11	0A008CD24713	26/04/2016 10:38	<input type="checkbox"/>
	12	0A008CD24713	10/05/2016 10:54	<input type="checkbox"/>
	14	0A008CD24713	10/05/2016 10:54	<input type="checkbox"/>
	16	0A008CD24713	10/05/2016 10:54	<input type="checkbox"/>
	18	0A008CD24713	12/05/2016 12:30	<input type="checkbox"/>

## 4. Accès aux couches de données

La DAL (Data Access Layer) Permet de s'abstraire du support des données. Pour se faire elle met à disposition des méthodes génériques permettant d'accomplir des actions de maintenances sur les données. Les actions les plus communes sont regroupées sous l'acronyme CRUD (Create Read Update Delete). Basiquement la DAL va donc mettre à disposition des méthodes permettant d'ajouter, mettre à jour, lire, supprimer un enregistrement, et ce quel que soit le support de stockage des données.

La généricité par rapport au stockage est en général matérialisée par un paramètre permettant de spécifier la nature du support (on appelle ça des drivers). Ainsi les méthodes CRUD associées au support de stockage sont utilisées de manière transparente par le développeur. Concrètement il n'y a donc en théorie qu'un paramètre à changer pour qu'une application utilisant une DAL puisse changer de support.

D'un point de vue strictement théorique la DAL devrait offrir la possibilité de maintenir des données dans n'importe quelle base de données, dans des fichiers texte, dans des fichiers xml ... .

### Présentation des classes DAL

- Dal\_base

La classe DAL\_Base qui encapsule toute notre logique fonctionnelle comme la création de connexions, des commandes SQL, des procédures stockées, et des paramètres. La classe DAL\_Base contiendra également des méthodes pour obtenir nos deux principaux types de retour, DTO et la liste de DTO, à partir d'un SqlDataReader. Pour agir comme un guichet unique pour l'ensemble de nos méthodes d'accès aux données, pour obtenir et définir les données d'Utilisateurs, nous allons créer une classe dal utilisateurs, qui héritera de DAL\_Base et contiendra l'ensemble de nos méthodes qui retournent ou enregistrent des données d'utilisateurs

```

class DAL_Base
{
    public MySqlConnection bdd;

    string database = "gestionacces";
    string userDB = "gestionacces";
    string passwordDB = "gestionacces";

    O références
    public DAL_Base()
    {
        //Création de la connexion a la base de données
        this.bdd = new MySqlConnection("Database=" + database + ";server=37.187.118.104;User id=" + userDB + ";pwd=" + passwordDB + "");
    }

    O références
    ~DAL_Base()
    {
        this.bdd.Close();
    }
}

```

- DAL\_Utilisateurs

C'est la dal qui hérite de DAL\_Base et où il y a toutes les méthodes crud afin de pouvoir intervenir et modifier comme représenté ci-dessous :

```

class DAL_Utilisateurs : DAL_Base
{
    public Boolean userConnexion(string login, string mdp)
    {
        MySqlCommand cmd = new MySqlCommand("SELECT * FROM utilisateurs WHERE login like @log AND motPasse like @mdp", this.bdd);
        cmd.Connection.Open();
        cmd.Parameters.AddWithValue("@log", login);
        cmd.Parameters.AddWithValue("@mdp", mdp);
        MySqlDataReader reader = cmd.ExecuteReader();
        Boolean retour = false;
        if(reader.HasRows)
        {
            retour = true;
        }
        cmd.Connection.Close();
        return retour;
    }
}

```

## ➤ Insert

L'insertion de données dans une table s'effectue à l'aide de la commande INSERT INTO. Cette commande permet au choix d'inclure une seule ligne à la base existante ou plusieurs lignes d'un coup

```
public void insert(object nom, object prenom, object idClasse, object codeRFID, object login, object motPasse, object idStatut)
{
    MySqlCommand cmd = new MySqlCommand("INSERT INTO utilisateurs ('idUser', 'nom', 'prenom', 'idClasse', 'codeRFID', 'login', 'motPasse', 'idStatut') VALUES (NULL, @nom, @prenom, @idClasse, @codeRFID, @login, @motPasse, @idStatut)");
    //MySqlCommand cmd = new MySqlCommand("INSERT INTO utilisateurs ('idUser', 'nom', 'prenom', 'idClasse', 'codeRFID', 'login', 'motPasse', 'idStatut') VALUES (NULL, "+ nom +", "+ prenom +", "+ idClasse +", "+ codeRFID +", "+ login +", "+ motPasse +", "+ idStatut +)");
    cmd.Connection.Open();
    cmd.Parameters.AddWithValue("@nom", nom);
    cmd.Parameters.AddWithValue("@prenom", prenom);
    cmd.Parameters.AddWithValue("@idClasse", idClasse);
    cmd.Parameters.AddWithValue("@codeRFID", codeRFID);
    cmd.Parameters.AddWithValue("@login", login);
    cmd.Parameters.AddWithValue("@motPasse", motPasse);
    cmd.Parameters.AddWithValue("@idStatut", idStatut);
    cmd.ExecuteNonQuery(); //cette cmd permet de recuperer le resultat de la requete
    cmd.Connection.Close();
}
```

## ➤ Update

Pour mettre à jour des données en MySQL (et plus généralement en SQL), vous devrez utiliser la commande UPDATE. Celle-ci permet de mettre à jour des enregistrements. Bien entendu, vous allez pouvoir spécifier quels enregistrements vous souhaitez mettre à jour, au moyen de conditions.

```
public void update(object id, object nom, object prenom, object idClasse, object codeRFID, object login, object motPasse, object idStatut)
{
    MySqlCommand cmd = new MySqlCommand("UPDATE `utilisateurs` SET `nom` = @nom, `prenom` = @prenom, `idClasse` = @idClasse, `codeRFID` = @codeRFID, `login` = @login, `motPasse` = @motPasse, `idStatut` = @idStatut WHERE `idUser` = @idUser");
    cmd.Connection.Open();
    cmd.Parameters.AddWithValue("@id", id);
    cmd.Parameters.AddWithValue("@nom", nom);
    cmd.Parameters.AddWithValue("@prenom", prenom);
    cmd.Parameters.AddWithValue("@idClasse", idClasse);
    cmd.Parameters.AddWithValue("@codeRFID", codeRFID);
    cmd.Parameters.AddWithValue("@login", login);
    cmd.Parameters.AddWithValue("@motPasse", motPasse);
    cmd.Parameters.AddWithValue("@idStatut", idStatut);
    cmd.ExecuteNonQuery(); //cette cmd permet de recuperer le resultat de la requete
    cmd.Connection.Close();
}
```

1 référence

## ➤ Delete

Pour supprimer des données en MySQL (et plus généralement en SQL), vous devrez utiliser la commande DELETE FROM. Celle-ci permet de supprimer des enregistrements. Bien entendu, vous allez pouvoir spécifier quels enregistrements vous souhaitez supprimer, au moyen de conditions.

```

// 1. Supprimer un utilisateur
public void delete(string id)
{
    MySqlCommand cmd = new MySqlCommand("DELETE * FROM utilisateurs WHERE idUtilisateur like @id", this.bdd);
    cmd.Connection.Open();
    cmd.Parameters.AddWithValue("@id", id);
    cmd.ExecuteNonQuery(); // cette cmd permet de recuperer le resultat de la requete
    cmd.Connection.Close();
}

```

## 5. Objet de transfert de donnée

Un objet de transfert de données (data transfer object ou DTO en anglais) est un patron de conception utilisé dans les architectures logicielles objet.

Son but est de simplifier les transferts de données entre les sous-systèmes d'une application logicielle. Les objets de transfert de données sont souvent utilisés en conjonction des objets d'accès aux données.

### Présentation des classes DTO

```

public class DTO_Historique
{
    // Références
    public int idHistorique { get; set; }
    // Références
    public string codeRFID { get; set; }
    // Références
    public DateTime dateTentative { get; set; }
    // Références
    public string etat { get; set; }
}

```

```

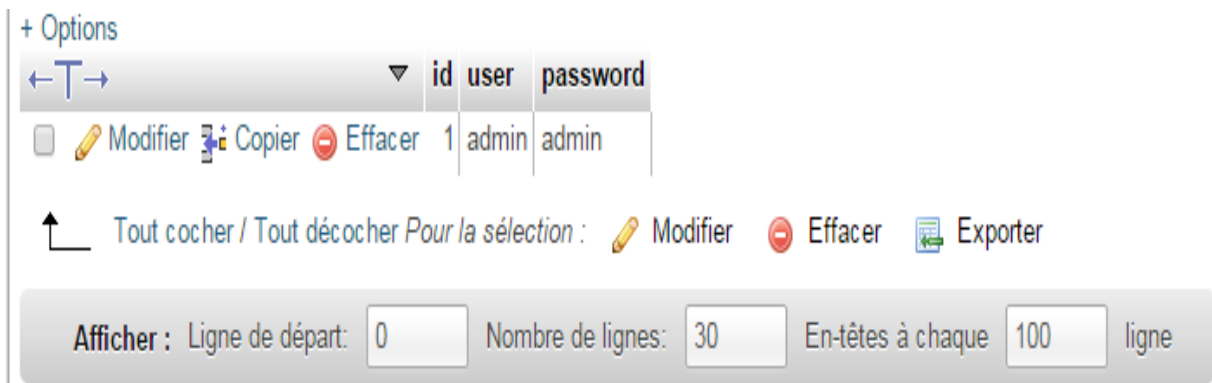
public class DTO_Historique_Utilisateurs
{
    Oréférences
    public int idHistorique { get; set; }
    Oréférences
    public string nom { get; set; }
    Oréférences
    public string prenom { get; set; }
    Oréférences
    public string codeRFID { get; set; }
    Oréférences
    public DateTime dateTentative { get; set; }
    Oréférences
    public string etat { get; set; }
}

```

## 6. Problématique

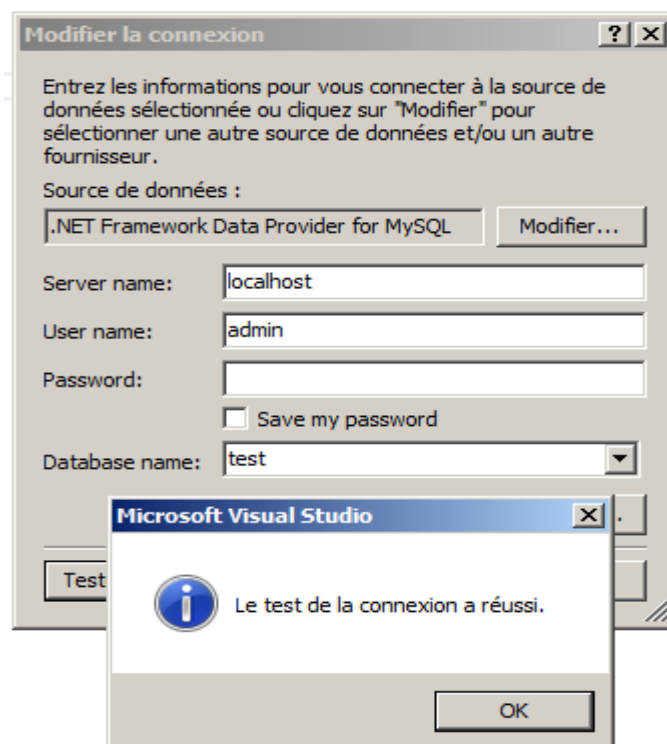
Pour tester la base de données j'ai tout d'abord utilisé easyphp pour créer une base de test, après la création de la base de données.





J'ai rencontré un problème durant cette phase de test. En effet, Visual studio n'inclus pas les librairies pour se connecter a une base de données MYSQL. Par conséquent j'ai dû chercher un connecteur MYSQL pour Visual Studio. J'ai essayé plusieurs versions différente mais elles ne fonctionnaient pas, la seule version qui fonctionnait était la 1.2.6.

Cela m'a permis de connecter la de base de données a Visual studio



Une fois que tout ceci était réglé, lorsque je codais il me signalait des erreurs avec MYSQL en m'indiquant qu'il manquait des DLL.



MySQL.Data.dll



MySQL.Data.Entity.dll



MySQL.Web.dll

## DLL

Une Dynamic Link Library (en français, bibliothèque de liens dynamiques) est une bibliothèque logicielle dont les fonctions sont chargées en mémoire par un programme, au besoin, lors de son exécution, par opposition aux bibliothèques logicielles statiques ou partagées dont les fonctions sont chargées en mémoire avant le début de l'exécution du programme.

.dll est une extension de nom de fichier utilisée par des fichiers contenant une Dynamic Link Library.



## 7. Tests

<b>Test fonctionnel :</b> Test d'identification				
<b>Objectif :</b>		réussir à se connecter en tant qu'administrateur		
<b>Éléments à tester :</b>		espace administrateur		
<b>Pré requis :</b>		– Un compte administrateur avec un mot de passe – Espace administrateur		
<b>Initialisation :</b>		ouverture de l'application et se connecter avec le compte admin		
<b>Scénario :</b>				
<b>ID</b>	<b>Démarche</b>	<b>Données</b>	<b>Comportement attendu</b>	<b>Validation</b>
1	Saisir le login et le mot de passe de l'administrateur	Login: admin Mdp: admin	Après connexion une fenêtre s'ouvre sur l'écran de gestion d'utilisateur	
<b>Rapport de test</b> <b>Testé par :</b> <b>Le :</b>				
<b>Fonctionnalité :</b>		<b>Conformité :</b>		<b>Ergonomie :</b>
<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente		
<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne		
<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne		
<input type="checkbox"/> Faible	<input type="checkbox"/> Faible	<input type="checkbox"/> Faible		
<b>Commentaire :</b>			<b>Approbation :</b>	
Fiches d'anomalies émises :				

<b>Test fonctionnel : Importer une liste d'utilisateurs</b>				
<b>Objectif :</b>		Réussir à importer une liste d'utilisateurs		
<b>Éléments à tester :</b>		Bouton importer		
<b>Pré requis :</b>		– Un compte administrateur avec un mot de passe – Espace administrateur		
<b>Initialisation :</b>		ouverture de l'application et se connecter avec le compte admin		
<b>Scénario :</b>				
<b>ID</b>	<b>Démarche</b>	<b>Données</b>	<b>Comportement attendu</b>	<b>Validation</b>
1	On choisit dans le menu du haut « Importer ».		Une fenêtre s'ouvre et l'on peut choisir quel fichier importer	
<b>Rapport de test</b> <b>Testé par :</b> <b>Le :</b>				
<b>Fonctionnalité :</b>		<b>Conformité :</b>		<b>Ergonomie :</b>
<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente		
<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne		
<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne		
<input type="checkbox"/> Faible	<input type="checkbox"/> Faible	<input type="checkbox"/> Faible		
<b>Commentaire :</b>			<b>Approbation :</b>	
Fiches d'anomalies émises :				

<b>Test fonctionnel : Visualiser l'historique</b>				
<b>Objectif :</b>		Avoir la liste des tentatives d'accès		
<b>Éléments à tester :</b>		Bouton visualisation historique		
<b>Pré requis :</b>		<ul style="list-style-type: none"> <li>– Un compte administrateur avec un mot de passe</li> <li>– Espace administrateur</li> </ul>		
<b>Initialisation :</b>		ouverture de l'application et se connecter avec le compte admin		
<b>Scénario :</b>				
<b>ID</b>	<b>Démarche</b>	<b>Données</b>	<b>Comportement attendu</b>	<b>Validation</b>
1	Cliquer sur le bouton visualisation historique		Une fenêtre s'ouvre avec toutes les tentatives d'accès	
<b>Rapport de test</b> <b>Testé par :</b> <b>Le :</b>				
<b>Fonctionnalité :</b>		<b>Conformité :</b>		<b>Ergonomie :</b>
<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente		
<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne		
<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne		
<input type="checkbox"/> Faible	<input type="checkbox"/> Faible	<input type="checkbox"/> Faible		
<b>Commentaire :</b>			<b>Approbation :</b>	
Fiches d'anomalies émises :				

<b>Test fonctionnel : Ajouter un utilisateur</b>				
Objectif :		L'ajout d'un utilisateur en base de données		
Éléments à tester :		Méthode insert		
Pré requis :		– Un compte administrateur avec un mot de passe – Espace administrateur		
Initialisation :		ouverture de l'application et se connecter avec le compte admin		
<b>Scénario :</b>				
ID	Démarche	Données	Comportement attendu	Validation
1	On choisit dans le menu gestion d'utilisateurs « éditer les utilisateurs »		Une fenêtre s'ouvre avec la liste des utilisateurs	
2	Remplir les champs			
3	Valider en cliquant sur le bouton « enregistrer les modifications »		Un pop-up s'ouvre validant le succès de l'action	
Rapport de test		Testé par :		Le :
Fonctionnalité :		Conformité :		Ergonomie :
<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente	<input type="checkbox"/> Excellente	
<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne	<input type="checkbox"/> Bonne	
<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne	<input type="checkbox"/> Moyenne	
<input type="checkbox"/> Faible	<input type="checkbox"/> Faible	<input type="checkbox"/> Faible	<input type="checkbox"/> Faible	
Commentaire :		Approbation :		
Fiches d'anomalies émises :				

**Test l'ajout et la modification d'un utilisateur**

## Méthode de test

```

dataGridView1.Update();
    dataGridView1.Refresh();
    // Fonction D'update des informations sur le datagridview directement
dans la BDD
    int x = 0;
    int nb = dataGridView1.RowCount - 1;
    string config =
"server=37.187.118.104;userid=gestionacces;database=gestionacces;pwd=gestionacces;Co
nvert Zero Datetime=True";

    MySqlConnection connexion = new MySqlConnection(config);
    connexion.Open();
    while (x < nb)
    {
        MySqlCommand cmd = new MySqlCommand("SELECT * FROM utilisateurs
WHERE `idUtilisateur` = " + this.dataGridView1.Rows[x].Cells[0].Value, connexion);
        MySqlDataReader reader = cmd.ExecuteReader();
        //on verifie si l'utilisateur doit etre modifier ou ajouter
        if(reader.HasRows){
            reader.Close();
            DAL_user.update(this.dataGridView1.Rows[x].Cells[0].Value,
this.dataGridView1.Rows[x].Cells[1].Value,
this.dataGridView1.Rows[x].Cells[2].Value,
this.dataGridView1.Rows[x].Cells[3].Value,
this.dataGridView1.Rows[x].Cells[4].Value,
this.dataGridView1.Rows[x].Cells[5].Value,
this.dataGridView1.Rows[x].Cells[6].Value,
this.dataGridView1.Rows[x].Cells[7].Value);
        }
        else
        {
            reader.Close();
            DAL_user.insert(this.dataGridView1.Rows[x].Cells[1].Value,
this.dataGridView1.Rows[x].Cells[2].Value,
this.dataGridView1.Rows[x].Cells[3].Value,
this.dataGridView1.Rows[x].Cells[4].Value,
this.dataGridView1.Rows[x].Cells[5].Value,
this.dataGridView1.Rows[x].Cells[6].Value,
this.dataGridView1.Rows[x].Cells[7].Value);
        }
        x = x + 1;
    }
    connexion.Close();
    MessageBox.Show("La base a été sauvegardée.");
}

```

## Résultats attendus

L'ajout, modification et la mise à jour de l'utilisateur

## Validation

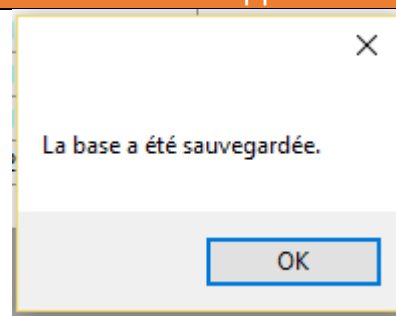
```

MySqlConnection connexion = new MySqlConnection(config);
    connexion.Open();
    while (x < nb)
    {
        MySqlCommand cmd = new MySqlCommand("SELECT * FROM utilisateurs
WHERE `idUtilisateur` = " + this.dataGridView1.Rows[x].Cells[0].Value, connexion);
        MySqlDataReader reader = cmd.ExecuteReader();
        //on verifie si l'utilisateur doit etre modifier ou ajouter
        if(reader.HasRows){

```

```
        reader.Close();
        DAL_user.update(this.dataGridView1.Rows[x].Cells[0].Value,
this.dataGridView1.Rows[x].Cells[1].Value,
this.dataGridView1.Rows[x].Cells[2].Value,
this.dataGridView1.Rows[x].Cells[3].Value,
this.dataGridView1.Rows[x].Cells[4].Value,
this.dataGridView1.Rows[x].Cells[5].Value,
this.dataGridView1.Rows[x].Cells[6].Value,
this.dataGridView1.Rows[x].Cells[7].Value);
    }
    else
    {
        reader.Close();
        DAL_user.insert(this.dataGridView1.Rows[x].Cells[1].Value,
this.dataGridView1.Rows[x].Cells[2].Value,
this.dataGridView1.Rows[x].Cells[3].Value,
this.dataGridView1.Rows[x].Cells[4].Value,
this.dataGridView1.Rows[x].Cells[5].Value,
this.dataGridView1.Rows[x].Cells[6].Value,
this.dataGridView1.Rows[x].Cells[7].Value);
    }
    x = x + 1;
}
```

### Résultat dans l'application



## DOSSIER DE MAINTENANCE

- Application embarquée (Yassine Hamouten)

Préventive :

Problème de lecture des badges RFID	<ul style="list-style-type: none"><li>• Vérifier que le lecteur RFID est bien branché</li><li>• Vérifier le lecteur RFID</li><li>• Le changer si défectueux</li></ul>
Problème de connexion au serveur	<ul style="list-style-type: none"><li>• Vérifier la connexion au réseau</li><li>• Faire un Ping vers l'adresse IP du serveur</li><li>• Vérifier si le serveur est bien exécuté</li></ul>

Curative :

Badges défectueux ou usagés	<ul style="list-style-type: none"><li>• Remplacement des badges</li></ul>
Lecteur RFID usagé	<ul style="list-style-type: none"><li>• Remplacement du lecteur</li></ul>

## INSTALLATION

- Application embarquée (Yassine Hamouten)

Déposer le fichier .exe ainsi que le fichier config.ini dans le répertoire projet qui est à la racine du FTP. Modifier le fichier config.ini pour spécifier l'adresse IP et le port du serveur distant.

Il faut ensuite déposer le fichier autoexec.bat à la racine du FTP. Ce fichier va permettre d'exécuter l'application directement au lancement du module BECK DK40.

L'arborescence du FTP sur le module BECK DK40 une fois l'application déployée :

<b>PROJET</b>	
CONFIG.INI	18 octets
PROJET.EXE	64 Ko
AUTOEXEC.BAT	21 octets
CHIP.INI	120 octets

Le fichier config.ini :

1	192.168.1.61
2	5001

Sur la première ligne il faut spécifier l'adresse IP, sur la seconde le port.



## NOTICE D'UTILISATION

- Application embarquée (Yassine Hamouten)

Vérifier que le lecteur RFID soit connecté et que le serveur distant soit lancé.

Présenter le badge RFID de l'utilisateur voulant accéder à la salle.

La porte s'ouvre si l'utilisateur est autorisé à y accéder.

## CONCLUSION

Au terme de ces 4 mois de projet, la plupart des tâches attendues ont été accomplies.

L'application embarquée de contrôle d'accès pour le module BECK DK40 est fonctionnelle et réalise l'intégralité des fonctions demandées dans le cahier des charges.

Cependant, certaines fonctionnalités ne sont pas encore mise en œuvre à la date de la remise du dossier mais devrait l'être au moment de la présentation.

Ce projet nous a permis tout d'abord d'apprendre à travailler en équipe sur un même projet à partir d'un cahier des charges dont le référentiel a été respecté, et a respecté le délai imparti.

De plus, le développement de l'application embarqué a été faite sur un processeur ancien, ce qui m'a permis (Yassine Hamouten) de mettre en œuvre mes connaissances et d'apprendre à utiliser une librairie spécifique.

Voici les tâches qu'il nous reste à faire avant l'orale de projet :

- Importer un fichier csv pour le client lourd C#
- Améliorer le serveur C#
- Finaliser le client lourd

## ANNEXES

### 1. Code source de l'application embarquée (Yassine Hamouten)

Le main :

```
#include <stdio.h>
#include <dos.h>
#include <cstring>
#include <stdio.h>
#include "clib.h"

#include "beckdk40.h"
#include "client.h"
#include "rfidgrovel25.h"
#include "serrure.h"

/*****
// Main
*****/
void main (void)
{
    beckdk40 * beck;
    client * cl;
    rfidGrove125 * rfid;
    serrure * ser;
    char * ip;
    int port;
    char * salle = "B01";
    int portCOM = 0;

    FILE * fichier;
    char str[20];
    fichier = fopen ("config.ini","r");
    if (fichier!=NULL){
        int i=0;
        while(fgets (str, sizeof(str), fichier)!=NULL){
            if(i==0){
                strcpy(ip, str);
                ip[strlen(ip)-1] = 0;
            }else{
                port = atoi(str);
            }
            i++;
        }
        fclose (fichier);
    }else{
        printf("impossible d'ouvrir le fichier de config!");
        exit(0);
    }

    beck = new beckdk40(portCOM);
    beck->focus(1);
```

```
printf("ip : %s port : %d", ip, port);
cl = new client(ip, port);
cl->connexion();

if(cl->connecte){
    printf("Connecte au serveur distant!\n");
    rfid = new rfidGrove125(portCOM);
    ser = new serrure();

    while(true){
        rfid->readCode(); // Bloquant
        char * code = rfid->getCode();
        char message[100];
        strcpy(message, "<code>");
        strcat(message, code);
        strcat(message, "</code><salle>");
        strcat(message, salle);
        strcat(message, "</salle>");
        cl->sendCode(message);
        int rep = cl->waitResponse();
        if(rep == 1){
            ser->ouvrir();
        }else if(rep == 2){
            printf("\nPas de reponse!\n");
        }else {
            printf("\nAcces refuse!\n");
        }
    }
}else{
    printf("Impossible de se connecter au serveur!\n");
    printf("Verifier que le serveur est bien lance\n");
    printf("Verifier le fichier de configuration(config.ini)");
}

beck->focus(0);
exit(0);
}
```

## La classe rs232 :

```
#include "rs232.h"

rs232::rs232(int port)
{
    // Initialisation du port COM

    //Init Fossil
    this->portCom = port;
    if(fossil_init(this->portCom)!=6484){
        printf("Erreur Init fossil!\n");
    }else{
        printf("Pas d'erreur Init fossil!\n");
    }

    // Purge des buffers
    fossil_purge_output(this->portCom);
    fossil_purge_input(this->portCom);

    // Definition des parametres pour le portCom
    fossil_setbaud (this->portCom , 9600L, 0, 8, 1);
    fossil_set_flowcontrol (this->portCom,0);

    printf("RS232 OK!\r\n");
}

rs232::~rs232(){
    fossil_deinit(this->portCom);
}

char rs232::getBytes(){
    // Lit un caractere sur le port COM
    return fossil_getbyte_wait(this->portCom);
}
```

## La classe rfidgrove125 :

```
#include "rfidGrove125.h"

rfidGrove125::rfidGrove125(int port): rs232(port)
{
    printf("RFID OK!\r\n");
}

rfidGrove125::~rfidGrove125(){
}

void rfidGrove125::readCode(){
    printf("\n\nEn attente du code RFID\n");

    int i = 0;
    char c;
    while(i<12){
        c = this->getBytes();
        if(c != 0x02 && c != 0x03){
```

```
        code[i] = c;
        i++;
    }
}

printf("Code RFID lu : %s\n",code);
}

char * rfidGrove125::getCode(){
    return code;
}
```

### La classe pio :

```
#include "pio.h"

pio::pio()
{
    // Initialisation PIO
    pfe_enable_bus(0xFFFF, 0);
    pfe_enable_pcs(0x06);
    printf("PIO OK!\r\n");
}

pio::~~pio(){
}

void pio::writePIO(unsigned char value){
    //0x10 -> PIN5
    //0x20 -> PIN6
    //0x40 -> PIN7
    //0x80 -> PIN8

    // ecrit les donnees
    hal_write_bus(0x600, value, 0xFFFF, 0x0000);

    if (value&0x80){
        //exemple: 0x10 & 0x80 -> 1 0000 & 1000 0000 -> pin n4 (5eme led)
        allumee
        hal_write_pio(13, 1);
    }else{
        hal_write_pio(13, 0);
    }
}

void pio::clearPIO(){
    this->writePIO(0x00);
}
```

## La classe serrure :

```
#include "serrure.h"

serrure::serrure()
{
    printf("SERRURE OK!\r\n");
}

serrure::~serrure() {
}

void serrure::ouvrir() {
    printf("\nOuverture de la porte\n");
    this->clearPIO();
    this->writePIO(0xC0);
    RTX_Sleep_Time(2000);
    this->clearPIO();
    printf("Fermeture de la porte\n");
}
```

## La classe client :

```
#include "client.h"

client::client(char * ip, int port){
    this->IPServeur = ip ;
    this->PortServeur = port ;
    this->bufLen = 1024;
    this->addr.sin_family = PF_INET ;
    this->addr.sin_addr.s_addr = 0 ;
    // Convertit entier depuis l'ordre des octets de l'hivers celui du r鳥au
    this->addr.sin_port = htons (this->PortServeur) ;
    // Convertit l'adresse ip en binaire
    inet_addr (this->IPServeur, &addr.sin_addr.s_addr);

    this->sd = opensocket(1, 0);
    if(this->sd!=-1){
        printf("\nSocket ouvert!\n");
    }
}

client::~~client(){
    closesocket(this->sd, &this->error_code);
}

void client::connexion(){
    result = connect( sd, (const struct sockaddr far *)&addr, &error_code
);
    if(result == 0){
        //Connexion reussie
        this->connecte = true;
    }else{
        //Connexion echouee
        this->connecte = false;
    }
}
```

```

    }
}

void client::deconnexion() {
    closesocket(this->sd, &this->error_code);
}

void client::sendCode(char * code) {
    int rez = send(this->sd, code, bufLen, MSG_BLOCKING, 0);
    if(rez!=-1){
        printf("Message envoye : %s\n",code);
    }else{
        printf("Message PAS envoye !\n");
    }
}

int client::waitResponse() {
    printf("En attente d'une reponse du serveur\n");
    char reponse[3] = " ";
    int retour;
    int len = 3;
    int rep = recv(this->sd, reponse, len, MSG_TIMEOUT , 15000, &this->error);
    printf("message reçu : '%s'\n", reponse);
    if(reponse[0] == 'o' && reponse[1] == 'k'){
        // reponse ok
        retour = 1;
    }else if(reponse[0] == ' '){
        // pas de reponse
        retour = 2;
    }else {
        // reponse nok
        retour = 0;
    }
    return retour;
}

```

### La classe becdk40 :

```

#include "becdk40.h"

becdk40::becdk40(int port)
{
    this->portCom = port;
}

becdk40::~becdk40() {
}

int becdk40::status() {
    return fossil_status_request(this->portCom);
}

void becdk40::reboot() {
    BIOS_Reboot();
}

```



```
void becdk40::focus(int param){  
    // Permet de definir le focus  
    if(param == 0){  
        BIOS_Set_Focus(FOCUS_SHELL);  
    }else if(param == 1){  
        BIOS_Set_Focus(FOCUS_APPLICATION);  
    }else{  
        BIOS_Set_Focus(FOCUS_BOTH);  
    }  
}
```