LYCEE GASTON BACHELARD – BTS SN 2019

PROJET RESTAURATION SCOLAIRE

BOURDON Randy / LUSTGARTEN Léo / MALLEDANT William

SOMMAIRE

I. PARTIE GENERALE

PRESENTATION DU PROJET

Dans le cadre du BTS Systèmes numériques, informatique et réseaux, il nous a été demandé de répondre, en équipe, au cahier des charges du Lycée Gaston Bachelard.

Pour cela, un groupe de 3 personnes a été formé, il se compose de :

- BOURDON Randy

- LUSGARTEN Léo

- MALLEDANT William

L'objectif de cette épreuve est de savoir gérer et planifier un projet imposé tout en tenant compte des moyens techniques du lycée et de la volonté du client.

Chacun des élèves du groupe a en charge une partie différente dans l'amélioration du système de gestion d'accès de la restauration scolaire.

- Permettre au personnel et aux élèves de réserver les repas de la semaine.
- Récupérer les effectifs en début d'année et d'imprimer les cartes de cantine où la photo du porteur figure. Aujourd'hui seule une saisie manuelle est possible, car le logiciel actuel ne permet pas d'associer la liste des élèves (fichier PDF) et les fichiers de photos rangés dans un dossier.
- Gérer le paiement des repas sur une seule base concernant l'ensemble des convives (élèves et employés du lycée).
- Distribuer automatiquement un plateau dès que la carte est reconnue, si le repas a été réservé.

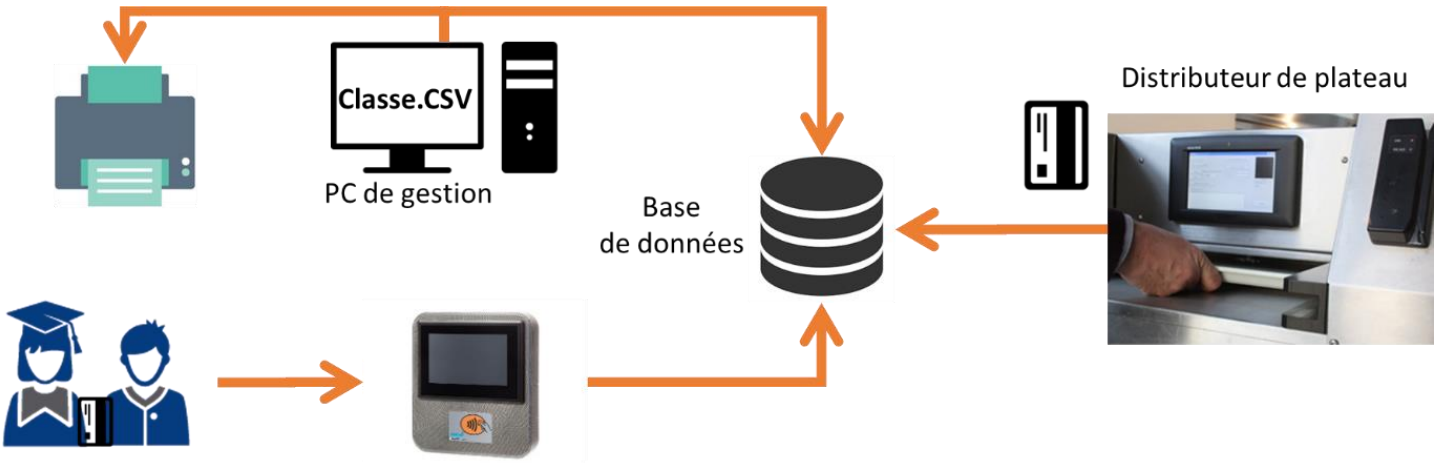
Cahier des charges

Besoins du client :

Dans le cadre de la rénovation de la cantine, l'intendance de notre lycée nous a solliciter pour rénover le système d'accès à la cantine scolaire.

L'objet de ce projet est de réaliser un système d'accès à la cantine scolaire. Ce système doit permettre de :

- Gérer en temps réel tout type de convives (possibilité de débloquer des cartes pour des personnes qui souhaitent déjeuner au dernier moment…),
- Gérer les différents modes de reconnaissance : tickets, cartes jetables, forfaits (en cas d'évolution de la tarification),
- Gestion depuis la cantine de diverses situations inhabituelles (oubli de carte, plage horaire non respectée…),
- Indiquer les refus de passage de la borne du self suivant différents critères de refus (solde de compte, nombre de passages autorisés, jours interdits, horaires interdits, périodes de stage).



Diagramme

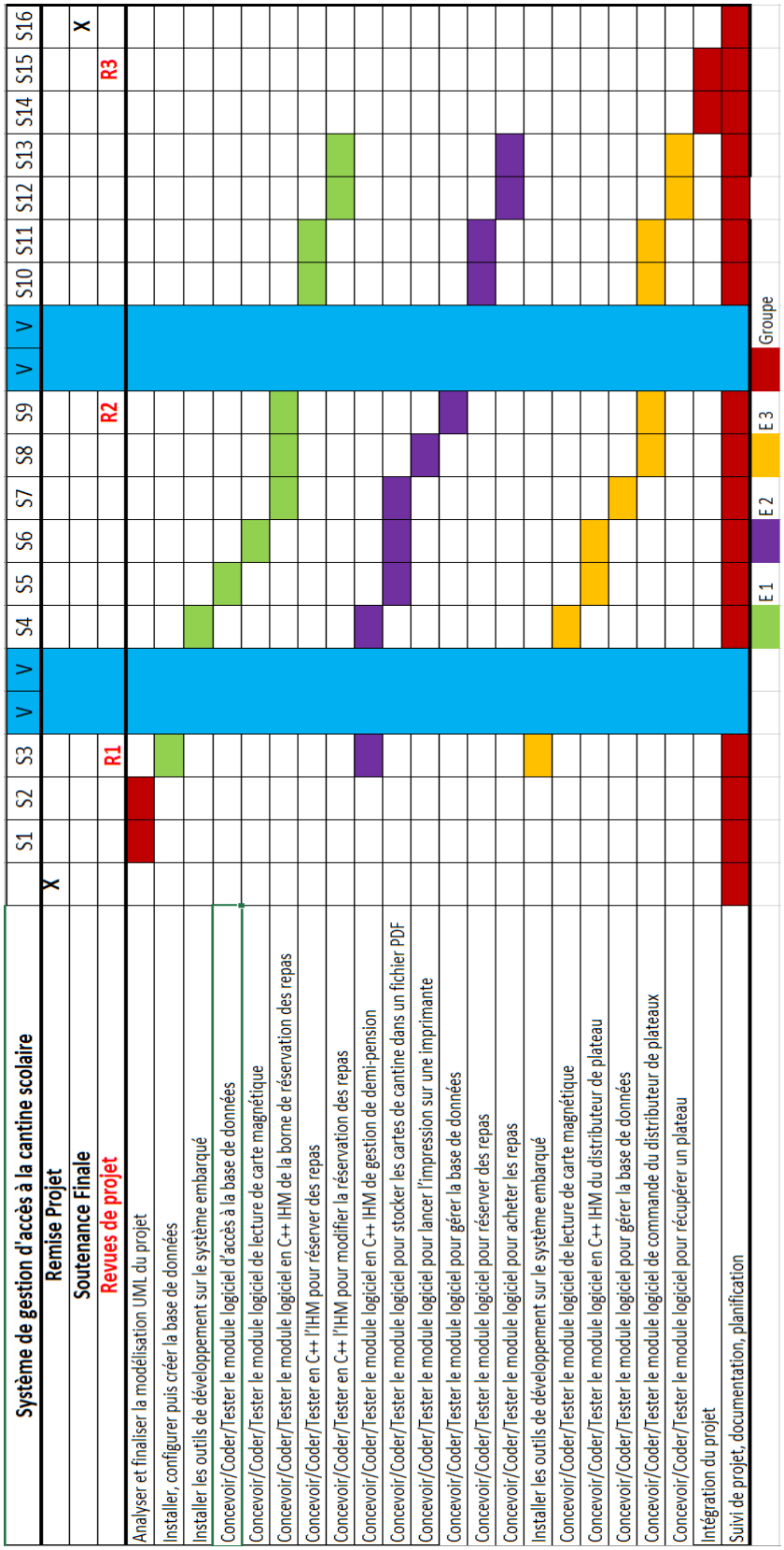


Diagramme de Gantt prévisionnel

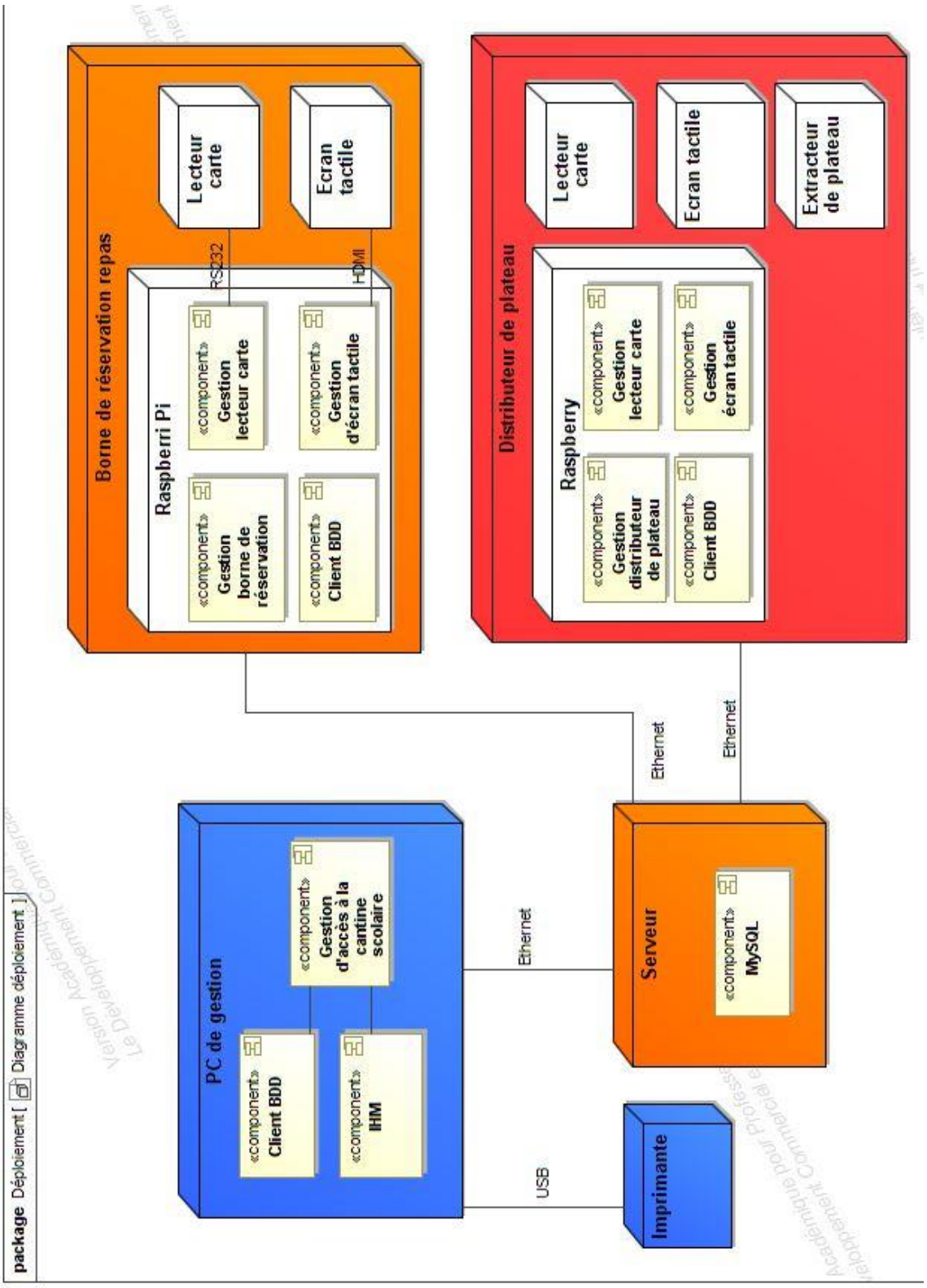


Diagramme de déploiement

II. PARTIE PERSONNELLE

ETUDIANT 1 – BOURDON RANDY

Ressources

Ressources matérielles

Les matériels à ma disposition sont :

- Une douchette OPR 2001 OPTICON

- Un ordinateur avec Windows

Ressources logicielles

Afin de procéder à mes taches, je possède les logiciels suivants :

- Qt (Windows / Linux)

- MySQL Workbench

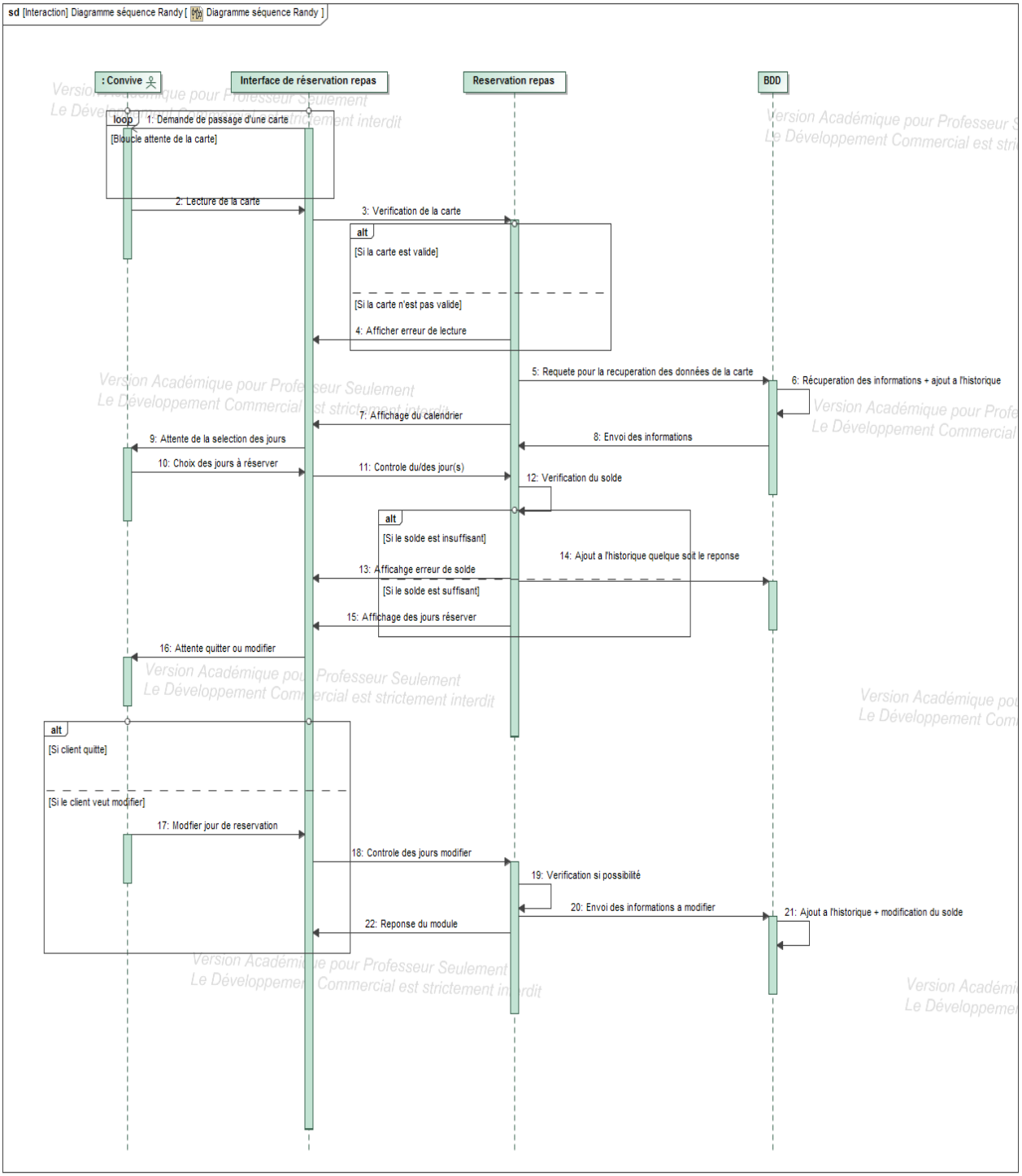
- WampServer

Présentation de ma partie personnelle

L'étudiant un a pour but de l'installation et la configuration de la base de données ainsi que de la conception et la réalisation de la borne de réservation des repas.

Diagramme

Diagramme de séquence :



1.1 Installation, configuration de la base de données

Dans ce projet, plusieurs types d'informations doivent être stockés pour le bon fonctionnement du projet, par exemple dans ma partie, je dois sauvegarder toutes les réservations de repas, des étudiants ou des employés de l'établissement.

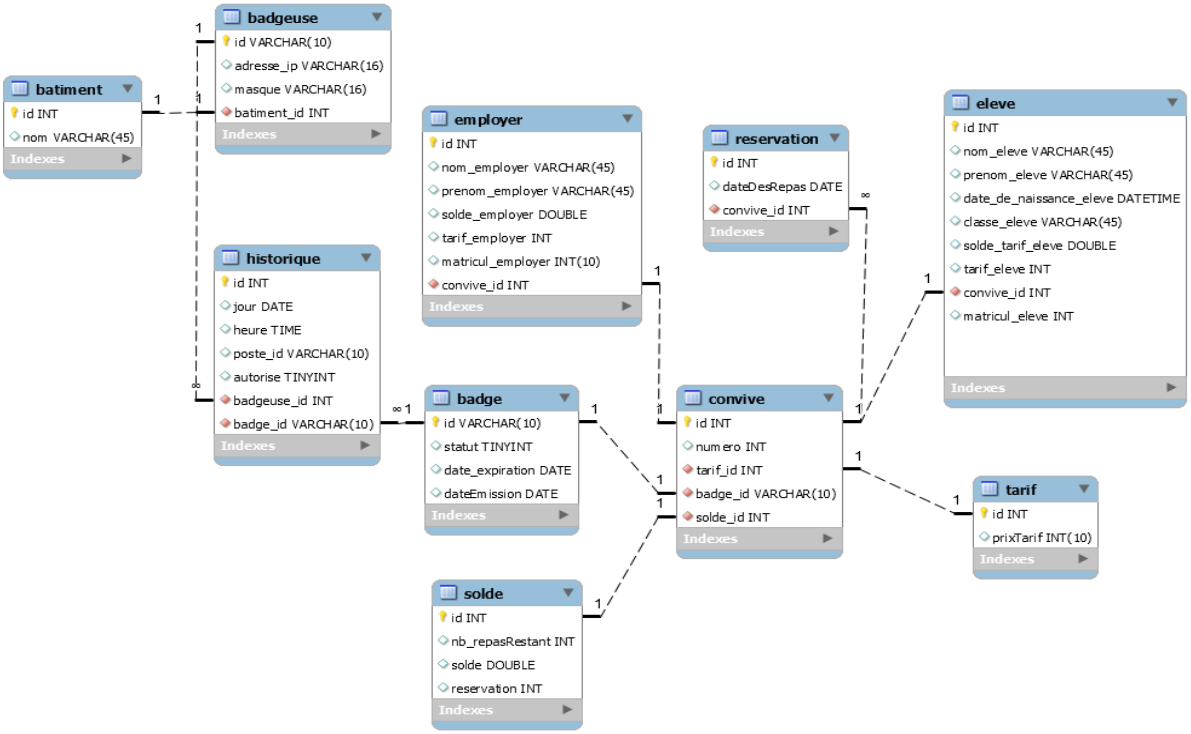
Afin de répondre à cette problématique, j'ai décidé d'utiliser MySQL pour mettre en place la base de données

Pour cette tâche j'ai utilisé plusieurs outils informatiques pour la création et la configuration de la base de données.
Tout d'abord, j'ai créé un modèle sur MySQL Workbench 8.0 (Diagramme ci-dessous)

Afin d'héberger la base de données, j'ai utilisé WampServer, ce logiciel permet de créer un serveur local.
Pour la gestion de la base de données j'ai utilisé PHPMyAdmin, ce qui m'as permis la modification des tables de la base de données

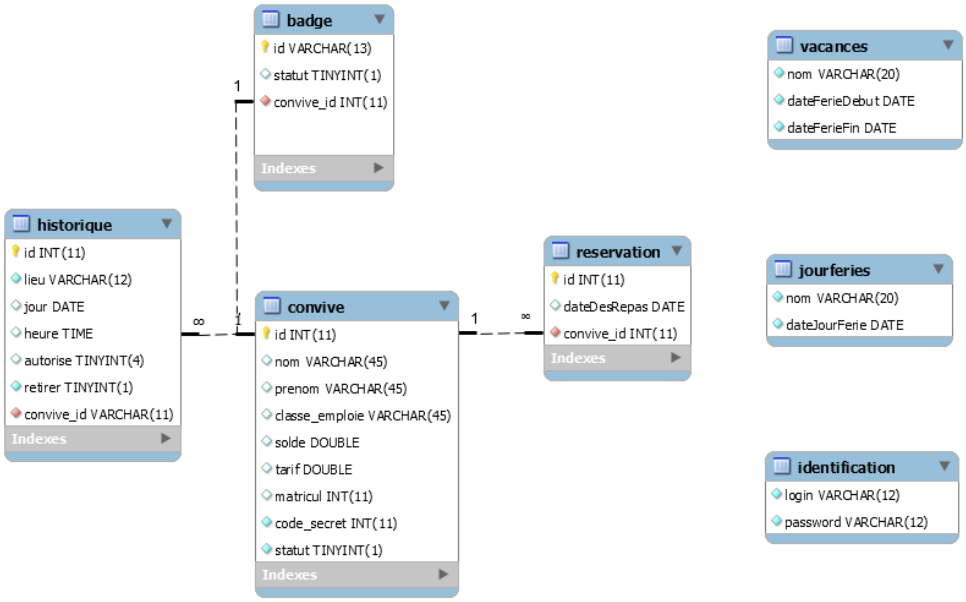
Avec l'aide de mon groupe de projet ainsi que mes professeurs, j'ai au fil du projet modifié la base de données de restauration scolaire afin qu'elle soit plus simple, comme par exemple regrouper la table professeur et élève dans la même table

La première base de données été construite comme ci-dessous :



La table badgeuse ou bâtiment était en trop, ou par exemple élève/employer ont été réunie dans la table convive afin de faciliter l'utilisation de la base de données

La base de données va contenir au final les tables suivantes :



Dans ma table principale, réservation, on pourra retrouver la date des repas réserver effectuer sous le format « yyyy-mm-jj » ainsi que convive_id (l'identifiant du convive) qui permet la réservation d'un repas.

Dans la table convive, on peut retrouver différents types d'informations concernant les étudiants ou employés comme son nom, son prénom ou son solde restant permettant de réserver des repas ultérieurement ou son tarif pour le prix d'un repas.

Afin de supprimer les réservations expirées de la table réservation, j'ai donc utilisé un « Evènement » programmé directement sur PhpMyAdmin.

Exécuter une fois par jour, l'évènement supprime toutes les réservations inférieures à la date courante avec une requête SQL, l'état du planificateur d'évènements doit être sur : Activé

Voir Annexe pour voir les différentes tables (1.0)

1.2 Concevoir, tester le module logiciel d'accès à la base de données

La conception d'accès à la base de données a été faite sous QT en C++.

« QT += sql » permet de lier le logiciel au module QtSql, cette ligne est ajoutée dans le .pro

#include <QSql> permet d'inclure les définitions des classes du module

J'ai créé une classe GestionBdd, avec plusieurs méthodes dont un destructeur et du constructeur qui prend en arguments :

Host : adresse IP du serveur sous la forme "xxx.xxx.xxx.xxx"

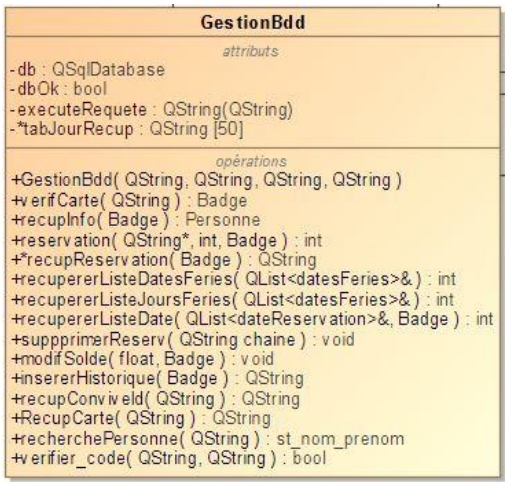
Bdd : nom de la base de données

Login : login de l'utilisateur

Pwd : mot de passe de l'utilisateur

Ainsi qu'un objet db de type QSqlDatabase, la classe QSqlDatabase représente une connexion à une base de données.

GestionBdd.h :



J'ai ensuite défini les paramètres de connexion grâce à setDatabaseName(), setUserName(), setPassword(), setHostName(), setPort()
Le port **3306** est le port pour la connexion au serveur de base de données MySQL
Grace au If (db.open()) on peut tester si la base de données est bien connecter avec dBOk de type booléen sinon en cas d'erreur le programme affiche avec lastError() la dernière erreur comme par exemple un mauvais mot de passe de la bbd
Constructeur de GestionBdd :

```
GestionBdd::GestionBdd(QString host, QString bdd, QString login, QString pwd)
{
    dbOk = false;
    db = QSqlDatabase::addDatabase("QMYSQL");
    db.setHostName(host);
    db.setUserName(login);
    db.setPassword(pwd);
    db.setDatabaseName(bdd);
    db.setPort(3306);

    if(db.open())
    {
        dbOk = true;
    }
    else
    {
        qDebug()<< db.lastError().text();
    }

}
```

addDatabase() est une méthode statique qui reçoit en paramètre « QMYSQL », cela permet de se connecter à une base de données de type MySQL

Les arguments seront transmis par l'appel du constructeur

```
static GestionBdd bdd("192.168.0.243","restauration_scolaire_bdd","root","randyleo");
```

1.3 Concevoir, tester le module IHM de la borne de réservation des repas

L'IHM (Interface Homme-Machine) sera affichée sur un écran tactile, elle doit être simple et intuitif pour tout le monde, assez grand car la taille de l'écran est limitée (une dimension de 800x480 pixels), l'étudiant ou l'employé pourra soit scanner sa carte par le biais d'une douchette USB a sa disposions ou par le biais d'un clavier tactile si la personne n'a pas sa carte.

Chaque carte sera imprimée d'un code barre, les codes-barres seront imprimés par le biais de l'application de Léo Lustgarten

Le code barre contiendra le numéro de chaque étudiant ou employer, codé sous forme de mots binaires, c'est à la base de l'électronique numérique.

Le code sera demandé grâce à un pop-up **(voir annexe 1.1)**

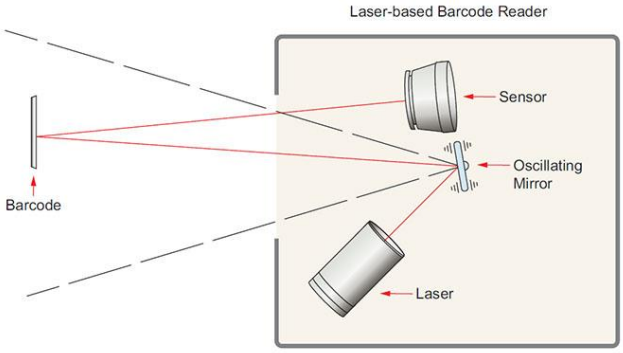
Pour lire ce code barre, comme dit précédemment, les utilisateurs utiliseront une douchette directement connectée au raspberry pi, lecteur optique qui consiste à envoyer sur le code-barres un faisceau lumineux (souvent un laser de très faible puissance) puis, à analyser la lumière réfléchie.

Utilisation de la douchette :

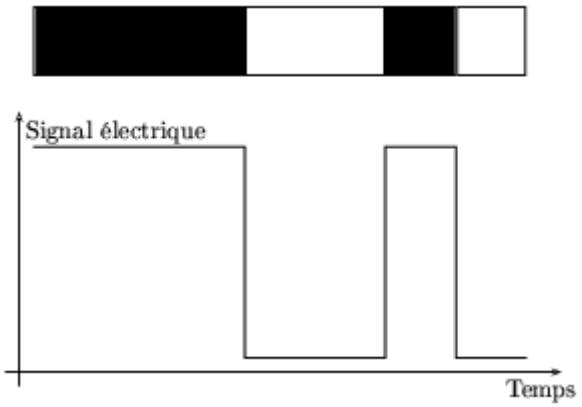
La douchette OPR 2001 qui sera à disposition permettra de lire le code barre imprimé sur la carte de cantine

Elle utilise une diode laser qui projette sur le code barre un faisceau laser (peu puissant mais suffisant) avec une longueur d'onde de 650 nm (nanomètres) cela signifie que le faisceau est de couleur rouge

Le rayon laser du lecteur va passer sur le code, selon la zone (sombre ou claire) au-dessus de laquelle le laser va passer, il va être réfléchi différemment, et le lecteur pourra par conséquent savoir si le laser est au-dessus d'une zone claire ou sombre grâce à une photodiode qui récupéra les photons renvoyés par les zones blanches, puis seront transformée en signal électrique selon le temps puis traduites en langages binaire (1 ou 0).



Le signal électrique sera ressemblant à ceci :



Source : Wikipédia

Le code barre :

On peut distinguer 2 types de code-barres :

Unidimensionnel (1D) : ils sont représentés par une série de lignes parallèles d'épaisseur variable.

Exemples : EAN, « **interleaved 2 of 5** » ou « **2 parmi 5** », …

Bidimensionnel (2D) : ces codes utilisent une variété de symboles (rectangles, points, hexagones et autres formes géométriques). Cette forme matricielle permet d'enregistrer d'avantage d'informations comme un site internet.

Exemples : Code QR, Code Datamatrix, …

Exemple de QR code :

Scannez-moi

La normalisation EAN (Non utilisé dans ce projet) :

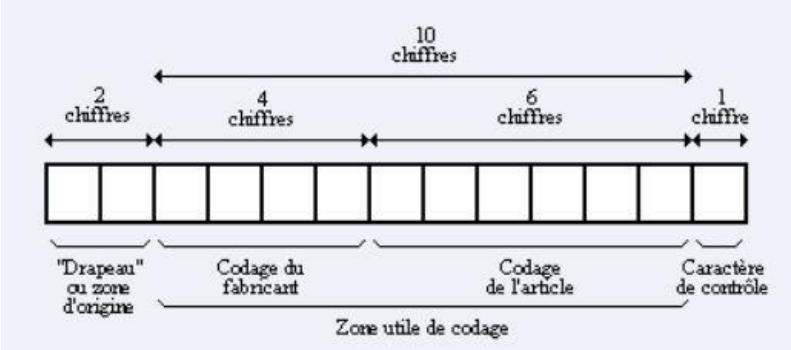
L'EAN ou European Article Number est une norme garantissant que le code-barres d'un article sera reconnu dans tous les pays de l'Union Européenne. L'EAN assure aussi une compatibilité avec les codes U.P.C. utilisés en Amérique du Nord.

Le mot codé est constitué de 8 ou 13 caractères répartis entre trois zones. En lisant de la gauche vers la droite, on trouve :

Le « drapeau » codant sur deux caractères l'origine du code

La « zone utile de codage » comprend 10 caractères dont les 4 premiers représentent le fabricant et les 6 derniers l'article (cas d'un code sur 13 chiffres). Dans le cas de cette zone, le codage dépend de la zone d'origine.

Le dernier caractère est le « code de contrôle » déterminant la validité du code. Il est calculé à l'aide d'un algorithme normalisé.



Dans ce projet, on utilisera des codes-barres a code linéaire de type **Interleaved 2 of 5** (en français le 2 parmi 5 entrelacé), développé par David Allais en 1972

En effet, la première donnée est encodée sous forme de barres alors que la seconde l'est sous forme d'espaces, barres et espaces étant entrelacés.

L'Interleaved 2 of 5 est de longueur variable, uniquement numérique et le nombre de caractères est TOUJOURS pair.

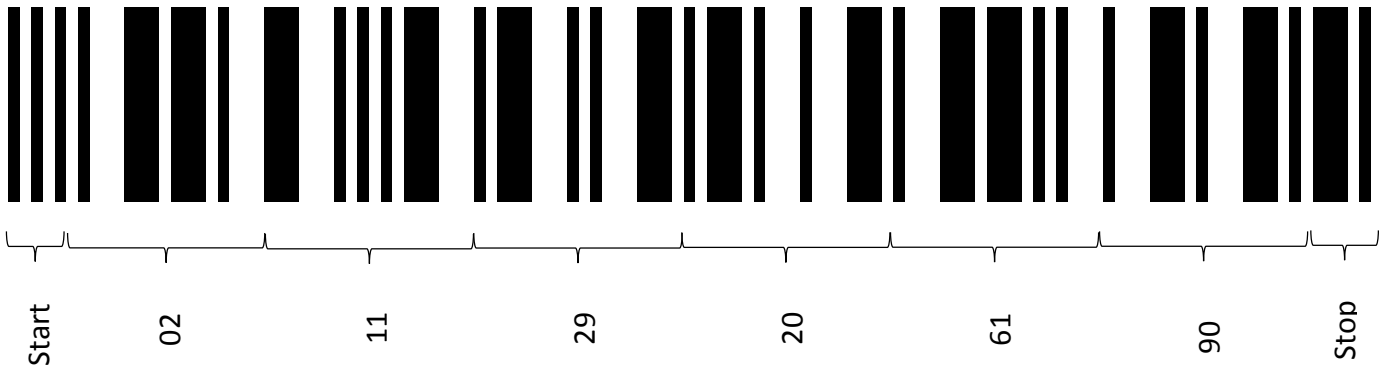
Donc pour coder 123, on écrira 0123

Les différents caractères sont représentés par la concaténation de deux symboles de base de largeur égales à 0,5 mm. Il existe deux symboles de base :

La barre noire, qui représente le 1 binaire,

La barre blanche, qui représente le 0 binaire.

Un exemple de code barre utiliser pour les cartes de cantine (021129206195) :



Le symbole comprend :

Le caractère de début (Start) (Barre Etroite - Espace Etroit - Barre Etroite Espace Etroit)

Les données

Le caractère de fin (Stop) (Barre Large - Espace Etroit - Barre Etroite)

Dans l'image ci-dessus (021129206195) le premier couple de données, soit 0 et 2, sera codé comme suit :

0 = E E L L E (0, première donnée du couple représentée sous forme de barres)

2 = E L E E L (2, seconde donnée du couple représentée sous forme d'espaces)

02 sera donc codé :

Barre **E**troit - espace **E**troit - barre **E**troite - espace **L**arge - barre **L**arge - espace **E**troit - barre **L**arge - espace **E**troit - barre **E**troit - espace **L**arge

Les autres données seront codées ainsi d'après le tableau suivant :

Chiffre	Représentation					Utilisations des poids
0	E	E	L	L	E	4+7 = 11, remplacé par 0
1	L	E	E	E	L	1+0 = 1
2	E	L	E	E	L	2+0 = 2
3	L	L	E	E	E	1+2 = 3
4	E	E	L	E	L	4+0 = 4
5	L	E	L	E	E	1+4 = 5
6	E	L	L	E	E	2+4 = 6
7	E	E	E	L	L	7+0 = 7
8	L	E	E	L	E	1+7 = 8
9	E	L	E	L	E	2+7 = 9
Poids des L	1	2	4	7	0	

02 = E E L L E - E L E E L

11 = L E E E L - L E E E L

29 = E L E E L - E L E L E

20 = E L E E L - E E L L E

61 = E L L E E - L E E E L

95 = E L E L E- L E L E E

Une fois le code barre converti en décimal, le numéro de la carte obtenu servira à accéder au service de réservation si le numéro est bien dans la base de données

Vérifié grâce à l'appelle de la méthode verifCarte :

```
badge = bdd.verifCarte(text);
```
(voir annexe 1.2)

Si le badge est autorisé, cela appellera la fenêtre pour la réservation **(Voir annexe 1.3)**

Fenêtre de réservation

La page de réservation sera composée d'un calendrier cliquable en boutons, de flèches de navigation entre chaque page du calendrier ainsi que des boutons valider et quitter et devra prendre en compte les vacances, journées banalisées, ect... **(Voir annexe 1.4)**

Quelques informations sur l'utilisateur seront affichées comme son nom, prénom, sa classe ou son emploie et le restant de crédit.

Si la carte est oubliée, la personne peut, en connaissant sont code secret, accéder à la fenêtre de réservation, la personne a juste à rechercher son nom dans la fenêtre adéquate puis entrer son code secret **(voir annexe 1.5)**, cette partie a été créer en collaboration avec William Malledant

La conception de l'IHM a été faite sous Windows avec Qt en C++, l'application utilise des widgets, des labels (QLabel), boutons (QPushButton) ainsi que des QGroupBox

Tous les QGroupBox sont mis dans un QGridLayout, ils sont ensuite positionnés dans l'application

```
QGridLayout *grid = new QGridLayout;

grid->addWidget(labelSuperieurC(),0,1);
grid->addWidget(labelSuperieurD(),0,3);
grid->addWidget(boutonChangement(), 2, 1);
grid->addWidget(calendrier(), 1, 1,1,3);
grid->addWidget(boutonInferieur(), 2, 3);
```

Avec ce code, les widgets seront placés de la façon suivante :

labelSuperieurC()	labelSuperieurD()
calendrier()	
boutonChangement()	boutonInferieur()

(Code annexe 1.6)

LabelSuperieurC contiendra la bannière du lycée Gaston Bachelard

LabelSuperieurD aura toutes informations concernant le client

boutonChangement possèdera deux boutons, droite et gauche

boutonInferieur disposera des boutons Valider et Quitter

Création du calendrier

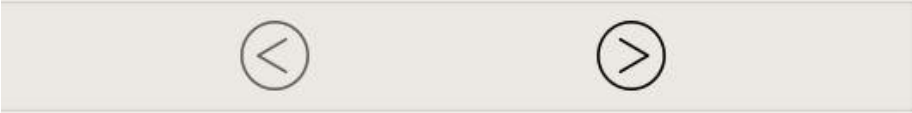
C'est au lancement de la fenêtre Réservation que le calendrier est créer dans le QGroupBox calendrier()

Pour la création des différents noms des jours de la semaine j'effectue simplement une boucle for ou dans un tableau jourDeLaSemaine j'ajoute les différents jours puis j'ajoute le tableau dans un QHBoxLayout

Afin que le calendrier soit à la bonne date, le programme doit récupérer l'heure du serveur et l'initialiser dans les variables tampons **(Voir annexe 1.7)**, les variables seront utilisées dans différentes boucles for **(Voir annexe 1.8)**

Passage d'une page a une autre

L'application est dotée de deux boutons droite et gauche afin de pouvoir permettre aux convives de réserver pour des prochains jours



Les boutons sont connectés aux slots changementDroit et changementGauche

connect(Droit,SIGNAL(clicked()),this,SLOT(changementDroit()));

connect(Gauche, SIGNAL(clicked()), this, SLOT(changementGauche()));

A chaque appel des slots, changementGauche() et changementDroit(), je réutilise les valeurs des boutons et je réutilise des boucles afin de tester si il y a des jours fériés par exemple **(Voir annexe 1.9)**

Bloquer les jours précédents

Pour améliorer mon application, j'ai décidé de bloquer les jours précédents par rapport au jour présent, pour cela je récupère le jour actuel (voir annexe) par exemple si le jour actuel est mardi l'application bloque juste le lundi

```
for(int h=0; h<jour; h++)
  {
    calendrierButon[h] = new QPushButton(this);
    calendrierButon[h]->setEnabled(false);
    calendrierButon[h]->setText(" ");

   }
```

Récupération des jours fériés de la base de données

Pour éviter que le client puisse réserver un jour férié, je dois les bloquer en avance, c'est pour cela que je dois récupérer dans la base de données toutes les dates fériées
Dans le constructeur, je récupère la liste des dates fériés (vacances...) et les jours fériés (8 mai,ect…)
Avec la requête SQL suivante :

```
QString requete ;
requete = "SELECT `dateFerieDebut`,`dateFerieFin` FROM `vacances`";
```

Vacances est une table de la base de données restauration scolaire

nom	dateFerieDebut	dateFerieFin
Toussaint	2019-10-19	2019-11-03
vacances d'hiver	2019-02-23	2019-03-10
vacances Printemps	2019-04-20	2019-05-05
vacances d'été	2019-07-06	2019-09-01
Noel	2019-12-21	2020-01-05

QList permet de stocker une liste de valeur (datesFeries qui est une structure)

```
struct datesFeries
{
   QString dateFeriesRecupDebut;
   QString dateFeriesRecupFin;
   QString jourFerie;

};
QList<datesFeries> listsDatesFeries;
QList<datesFeries>::iterator j;
```

```
/****************************** Recupere toutes les periodes feriees ************************************/
    bdd.recupererListeDatesFeries(listsDatesFeries);
    if(!listsDatesFeries.empty())
    {
        for(j= listsDatesFeries.begin();j<listsDatesFeries.end();j++)//
        {
            tabDatesFeriesRecupDebut[nombreJour] = j->dateFeriesRecupDebut;
            tabDatesFeriesRecupFin[nombreJour] = j->dateFeriesRecupFin;
            nombreJour = nombreJour+1;
        }
    }
```

Les valeurs sont ensuite mises dans un tableau afin d'être utilisé plus tard

1.4 Coder/Tester en C++ l'IHM pour réserver des repas

Pour réserver des repas, il faut d'abord récupérer les jours déjà réserver dans la base de données, comme pour les jours fériés, j'utilise QList qui récupère toutes les réservations par rapport au convive.id récupérer plus tôt

;

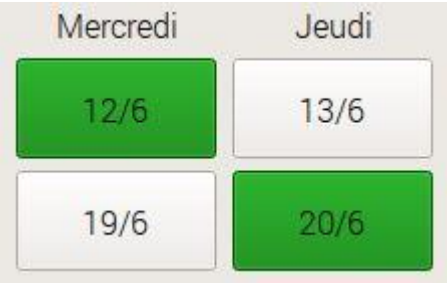
```
/****************************** Recupere toutes les reservations du client ******************
    bdd.recupererListeDate(listsDate,general);
    if(!listsDate.empty())
    {
        for(p= listsDate.begin();p<listsDate.end();p++)
        {
            tabJourRecup[nombreJour] = p->dateRecup;
            tabJourReserv[nombreJour] = tabJourRecup[nombreJour];
            nombreJour= nombreJour+1;
        }
    }
```

General est une variable générale de type Badge

La requete utiliser est : requete = "SELECT dateDesRepas FROM reservation WHERE convive_id = "+badge.convive_id+"";

Dans la création du calendrier, j'effectue une boucle afin tester si la date dans les boutons correspond à une date réserver dans la bdd,si oui cela change la couleur des boutons où le date est réserver

```
for(m=0;m<nombreDeJour;m++)//Jours deja resrever
{
    autre = tabJourRecup[m];
    strcpy(texte,autre.toStdString().c_str());
    sscanf(texte,"%d-%d-%d",&anneel,&moisl,&jourl);//Extraire le jour, le mois et l'année
    datel =QString::number(jourl)+"/"+QString::number(moisl);
    if(datel == calendrierButon[i]->text())
    {
        calendrierButon[i]->setStyleSheet("background-color: green;");
    }
```



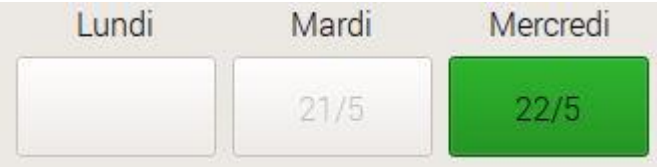
Dans cet exemple, le 12 juin ainsi que le 20 juin 2019 sont réservé

Si l'étudiant souhaite réserver un jour ou plusieurs jours, il n'a qu'à appuyer sur les jours où il veut manger, dès qu'il appuie cela envoie un signal ce qui appelle le slot enregistrement

Pour connecter les boutons, j'utilise un pointeur de type QSignalMapper :

QSignalMapper *signalMapper = new QSignalMapper(this);

Cela fournit un moyen d'établir une relation entre un ensemble de signaux, L'appel de setMapping() dans la boucle for établit une liaison entre un bouton et une valeur entière, par exemple



Le bouton [2] est associé à la valeur «21/5» et le bouton [3] est associé à la valeur «22/5 »

Avec cette ligne de code :

```
signalMapper->setMapping(calendrierButon[i], i);
```

Lorsque le signal clicked() d'un bouton est émis, le slot map() du QSignalMapper est invoqué (grâce au connect() dans la boucle for).

```
connect(calendrierButon[i], SIGNAL(clicked()), signalMapper, SLOT(map()));
```

Si une liaison existe pour l'objet qui a émis le signal, le signal mapped(int) est émis avec la valeur entière définie par setMapping()

Ce signal est à son tour connecté au signal choixJour(int)

```
QObject::connect(signalMapper, SIGNAL(mapped(int)), this, SLOT(choixJour(int)));
```

choixJour reçois en argument un int **(Voir annexe 1.10 pour le code complet)**, cela est la position du bouton cliqué, c'est un slot permettant de rendre le bouton cliqué vert ainsi qu'insérer la valeur du bouton dans un tableau, si le bouton est déjà dans le tableau, le programme rend le bouton dans sa couleur initial (blanc) et retire la valeur du tableau

En même temps, le slot met à jour le solde du crédit en temps réel graphiquement

Permet le changement de couleur du bouton :

```
calendrierButon[pos]->setStyleSheet("background-color: green;");
```

Permet l'insertion dans le tableau des réservations :

```
tabJourReserv[nombreDeJour] = date.toStdString().c_str();
```

Boucle permettant le test si la date est déjà dans le tableau :

```
for(i=0;i<nombreDeJour;i++)
   {
      if(date== tabJourReserv[i])
      {
         tabJourReserv[i]="";
         calendrierButon[pos]->setStyleSheet(0);
         soldeAffichage = soldeAffichage +personne.tarif.toFloat();
         solde->setText("Crédit : "+ QString::number(soldeAffichage) + " €");
         vrai=true;
      }
   }
   if(vrai == false)
   {
      soldeAffichage = soldeAffichage -personne.tarif.toFloat();
      solde->setText("Crédit : "+ QString::number(soldeAffichage) + " €");

   }
```

1.5 Coder/Tester en C++ l'IHM pour la modification des repas

Si l'étudiant ou l'employé souhaite réserver, il a juste à cliquer sur le/les jours qu'il veut de réserver (expliquer précédemment), et pour valider ses choix il n'a qu'à appuyer sur valider. Cela entraine l'appel du slot enregistrement **(Voir annexe 1.11)**, l'application va demander la confirmation au client s'il souhaite sauvegarder ses choix, si oui, l'application va vérifier si le solde est suffisant afin de pouvoir continuer la procédure de validation

L'application va ensuite procéder à la réservation des jours réserver dans la base de données ainsi qu'à la modification du solde dans la table convive, elle informera le client de la bonne sauvegarde des jours réservé.

L'application va déjà supprimer toutes les réservations enregistrer dans la base de données grâce à la requête suivante :

```
QString requete = "DELETE FROM reservation WHERE dateDesRepas = \""+chaine+"\"";
```

Puis dans une boucle, la requête insère toutes les valeurs (dates) se trouvant dans le tableau tabJourReserv.

```
 for(int i=0;i<nb;i++) // nb est le nombre de jours sauvegardé dans la tableau
   {
      chaine = tab[i]; //Chaine est de type QString
      requete = "INSERT INTO  reservation(dateDesRepas, convive_id) VALUES
(\""+chaine+"\",\""+badge.convive_id+"\")";
      executerRequete(requete);
   }
```

Si le solde n'est pas suffisant, l'application affiche un pop-up afin de prévenir que le solde est insuffisant

Chaque action de sauvegarde est enregistrer dans la base de données dans la table historique (**Voir annexe 1.12)**

Conclusion personnelle :

Ma partie est actuellement en phase terminal (Amélioration de la charte graphique, modification du code pour une meilleur compréhension).

L'écran de réservation (réservation possible jusqu'à 10h) permet bien de réserver ou modifier des repas, l'enregistrement et l'insertion de l'historique se fait correctement.

Il est aussi possible de consulter son solde (solde affiché en haut à droite de l'écran).

Le cahier des charges a été remplis.

Problèmes rencontrés :

Durant le projet, j'ai rencontré plusieurs problèmes :

Des problèmes matériels, par rapport à comment lire la carte du convive car dans le système actuel de réservation n'est pas le même que celui qu'on utilise (douchette)

Problèmes avec le changement des semaines dans le calendrier

Problèmes d'années bissextiles

Problèmes de réservation, une trop petite allocation de mémoire avait été faite, j'ai donc palier à ce problème en bloquant le nombre de réservations possibles.

Tous les problèmes rencontrés ont été résolu.

III. PARTIE PERSONNELLE

ETUDIANT 2 – LUSTGARTEN Léo

1. Mon rôle dans le projet

Je suis chargé de l'IHM du poste de gestion de demi-pensionnaire.
Je dois créer une application qui permettra à l'intendance d'effectuer simplement les actions suivantes :

- Ajouter ou supprimer des convives de la base de données.
- Imprimer des cartes pour les convives.
- Gérer le solde ou le tarif des convives.
- Réserver pour les convives .

- Possibilité de débloquer des cartes pour des personnes qui souhaitent déjeuner au dernier moment.

- Entrer les jours de fermeture.

- Nous avons ajouté cette tâche afin qu'il soit impossible de réserver un jour où le lycée est fermé (vacances, jours fériés, fermeture occasionnelle).

2. Ressources

- 2.1 Ressources matérielles :

- **Un ordinateur sous l'OS Windows**
- **Une imprimante à carte Zebra P330i**

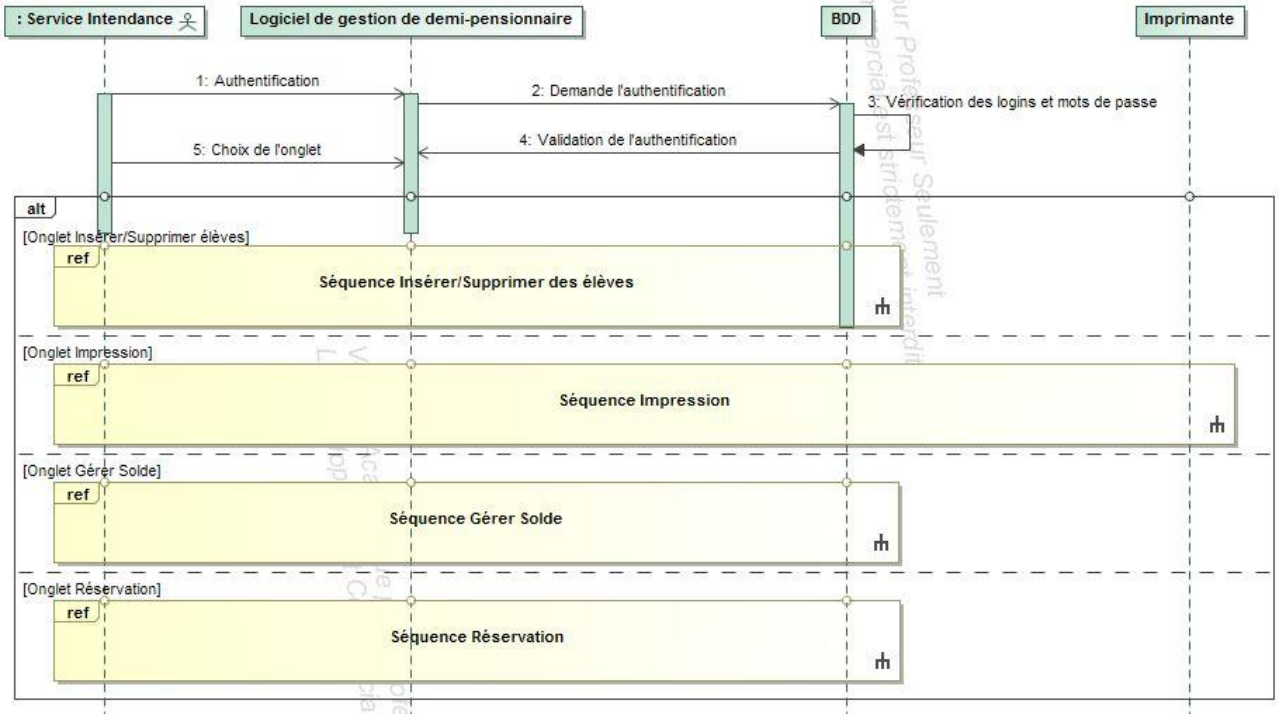
- 2.2 Ressources logicielles :

- L'environnement de développement intégré C/C++ QtCreator pour la programmation.
- Le logiciel MagicDraws pour faire les différents diagrammes.

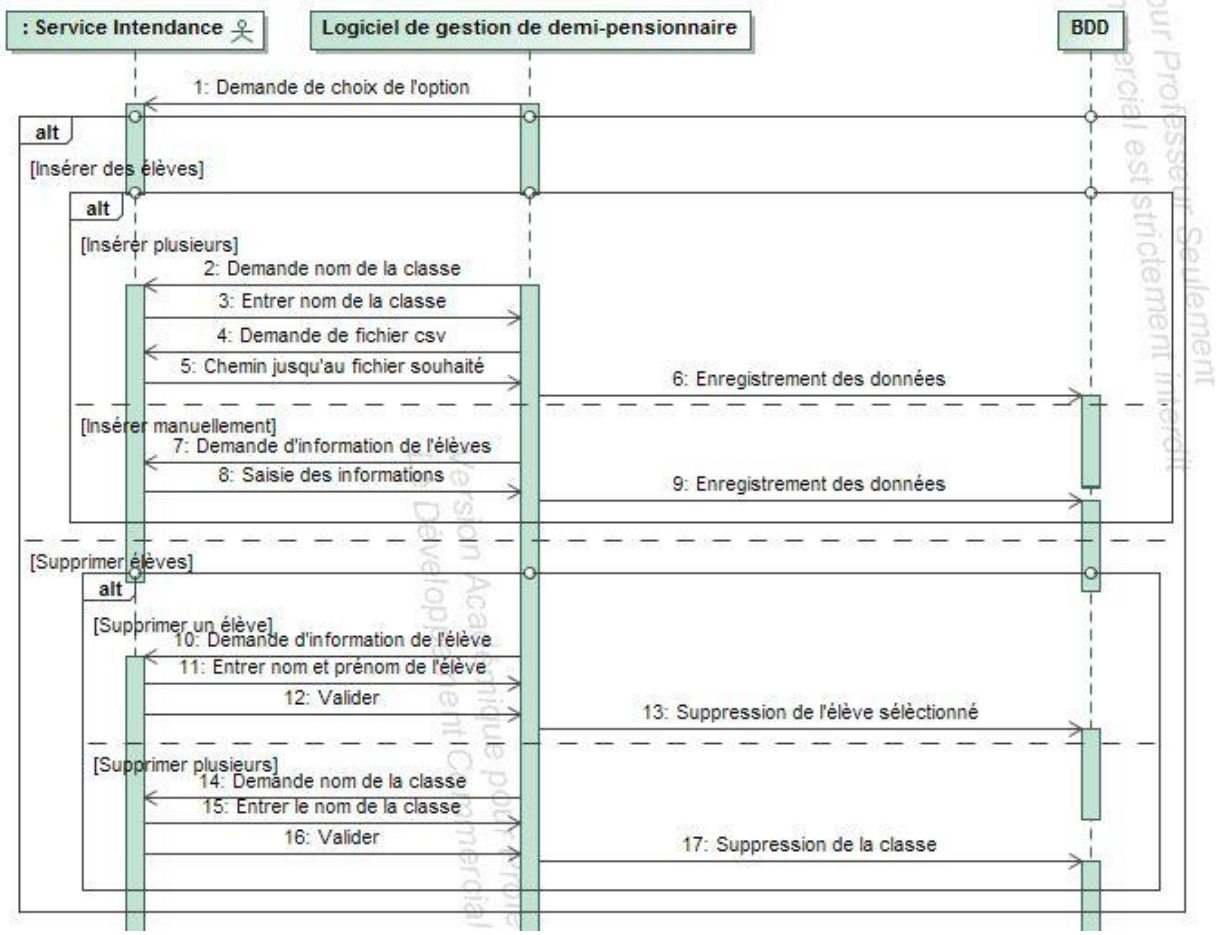
- Le navigateur Mozilla Firefox pour accéder à la base de donnée via PHPMyAdmin pour vérifier le bon fonctionnement des requêtes SQL de l'application, et pour effectuer des recherches si nécessaire.

3. Diagrammes

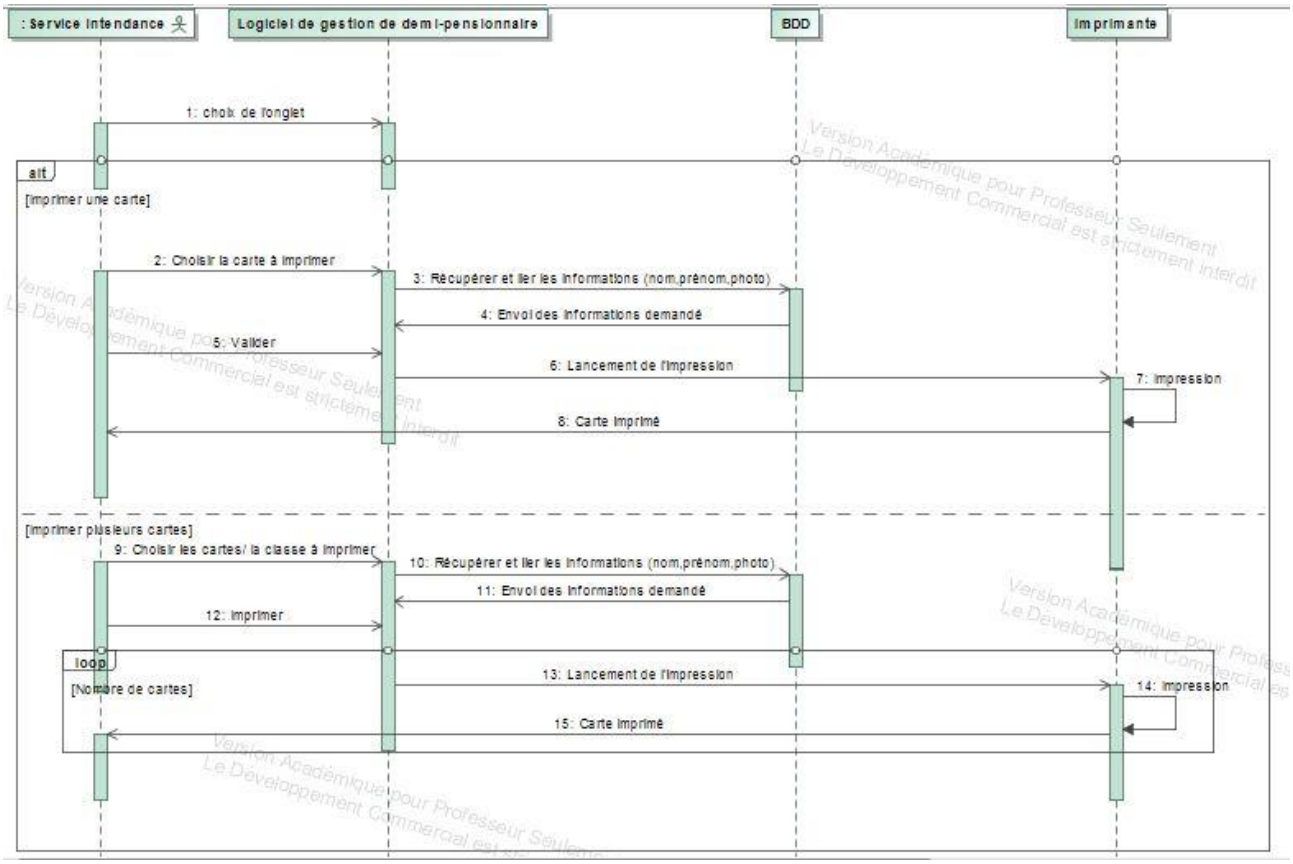
Mon diagramme de séquence se présente d'abord sous la forme d'un diagramme général avec des sous-parties redirigeant vers les diagrammes correspondants.



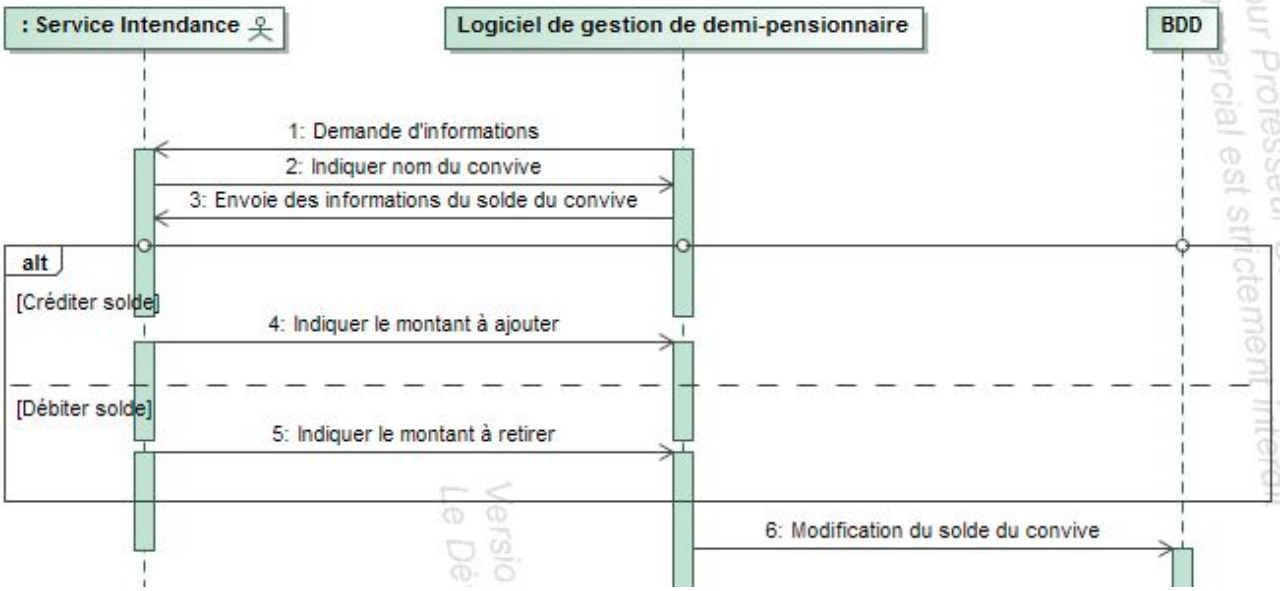
Séquence Insérer/Supprimer des élèves



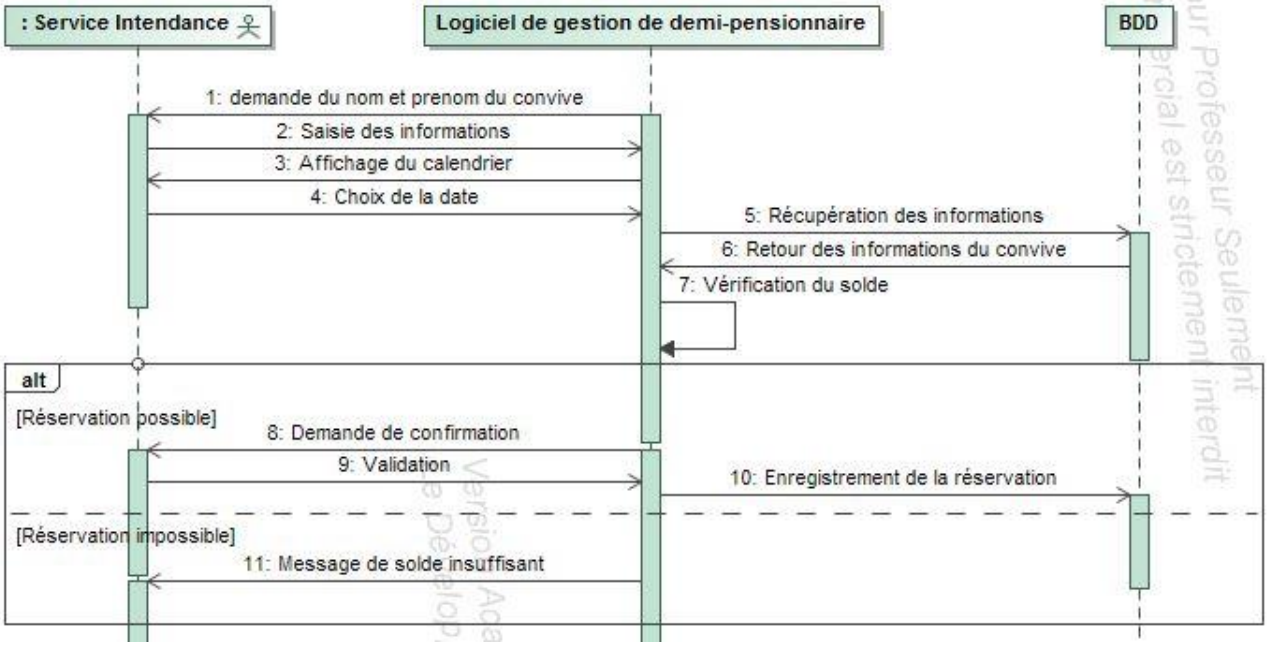
Séquence Impression



Séquence Gérer Solde



Séquence Réservation



4. Mes tâches

- 4.1 Concevoir/Coder/Tester le module logiciel en C++ IHM de gestion de demi-pension

- 4.1.1 Créer des onglets

Afin que l'intendance puisse gérer le système de demi-pension, j'ai dû concevoir une application en langage C++ sur Qt Creator où l'utilisateur pourrait choisir de naviguer simplement entre les différentes actions qu'il doit effectuer. Pour cela, j'ai décidé que mon IHM (Interface Humain Machine) comporterait un système d'onglet, les différentes tâches de mon cahier des charges seraient réparties entre les onglets.

J'ai donc pour cela utilisé un QTabWidget.

Je devais bien évidemment pour cela, tout d'abord inclure l'en-tête nécessaire à l'utilisation de Widget sur Qt avec la ligne suivante : #include "QtWidgets" (ajouté au début de mon fichier « .h »).

```
private:
    QWidget *page1 = new QWidget;
    QWidget *page2 = new QWidget;
    QWidget *page3 = new QWidget;
    QWidget *page4 = new QWidget;
    QWidget *page5 = new QWidget;
    QWidget *page6 = new QWidget;
```

J'ai donc déclaré un QWidget pour chaque future page, dans mon fichier header .

Voici donc à quoi ressemble mon constructeur :

Dans un premier temps, je déclare un layout pincipal dans lequel je mets mon QTabWidget, puis j'applique ce layout à ma fenêtre.

```
Gestion_Intendance::Gestion_Intendance(QWidget *parent)
    : QWidget(parent)
{
    QVBoxLayout *principale = new QVBoxLayout;
    // Créer le QTabWidget qui contiendra toutes les pages
    QTabWidget *onglets = new QTabWidget;
    principale->addWidget(onglets);
    setLayout(principale);
```

Ensuite j'ajoute mes pages en leur donnant leur nom :

```
// Ajouter chaque pages dans le QTabWidget
onglets->addTab(page1, "Accueil");
onglets->addTab(page2, "Insérer/Supprimer convives");
onglets->addTab(page3, "Impression");
onglets->addTab(page4, "Gérer Solde et Tarif");
onglets->addTab(page5, "Réservation");
onglets->addTab(page6, "Fermeture");
```

Je garde active uniquement ma page 1, car elle comportera un système de connexion qui permettra de débloquer toutes les autres pages.

```
// Désactivation des pages (sauf page d'accueil)
page2->setDisabled(true);
page3->setDisabled(true);
page4->setDisabled(true);
page5->setDisabled(true);
page6->setDisabled(true);
```

Enfin j'appelle mes fonctions, l'une d'elles servant à se connecter à la base de donnée, les autres à créer les pages.

```
    // Connexion à la base de donnée
    connexionBDD();
    // Création des pages
    creationPage1();
    creationPage2();
    creationPage3();
    creationPage4();
    creationPage5();
    creationPage6();
}
```

- 4.1.2 Connexion à la base de donnée

Afin de me connecter à la base de données dès le lancement de l'application, puis ensuite de pouvoir y accéder et faire des requêtes, j'ai ajouté la ligne :

```
QT += sql
```

dans mon fichier « .pro », et les lignes :

```
#include <QSqlDatabase>
#include <QDebug>
#include <QtSql>
#include <QSqlQuery>
#include <QSqlError>
```

dans mon fichier «.h ».

J'ai ensuite créé une simple fonction de connexion à la base de données avec

```
void Gestion_Intendance::connexionBDD()
{
    db = QSqlDatabase::addDatabase("QMYSQL");
    db.setHostName("192.168.0.243");
    db.setUserName("root");
    db.setPassword("randyleo");
    db.setDatabaseName("restauration_scolaire_bdd");
    if(db.open())
    {
        qDebug() << "Connexion à la base de donnée reussi" ;
    }
    else
    {
        QMessageBox::information(this,("ERREUR"),("Impossible de se connecter à la base de données: <br/> <br/>"+db.lastError().text()));
    }
}
```

un QmessageBox qui s'ouvre en cas d'erreur et qui indique l'erreur.

- 4.1.3 Fonction pour requêtes simples

Comme j'aurais à exécuter de nombreuses requêtes SQL tout au long de la programmation de cette application, j'ai décidé de créer une fonction « executerRequete » à laquelle on passerait la requête comme argument pour que celle-ci s'exécute.

```
QString Gestion_Intendance::executerRequete(QString requete)
{
    QSqlQuery query;
    qDebug() << requete;
    query.exec(requete);
    qDebug() << query.lastError().text();
    return  query.lastError().text();
}
```

- 4.1.4 Page d'accueil : connexion/déconnexion

Pour pouvoir disposer l'affichage de mon IHM à ma convenance, j'ai suivi les conseils de mon camarade de projet Randy Bourdon, et j'ai opté pour la création de différents QgroupBox disposés dans un QGridLayout final sur chaque page.

J'apporte aussi différentes modification à mes widgets avec les fonctions suivantes :

setStyleSheet() : Pour modifier la police (taille, couleur, etc.).

setPlaceholderText() : Pour indiquer ce qu'il y' a à remplir des les QlineEdit.

setEchoMode() : Pour que le texte du champ « mot de passe » soit remplacé par des points.

setFixedSize() : Pour modifier la taille des QlineEdit et des QpushButton.

setAlignment() : Pour placer les widgets dans le layout.

Si j'ai facilement réussi à trouver comment centrer un Qlabel dans un layout.

QLabel->setAlignment(Qt::AlignCenter);

J'ai eu plus de difficultés à trouver comment centrer un QLineEdit ou un QPushButton car la ligne

QLineEdit->setAlignment(Qt::AlignCenter);

centre uniquement le texte à l'intérieur du QLineEdit et pour un QPushButton la ligne

QPushButton->setAlignment(Qt::AlignCenter);

ne change rien. C'est donc après un certain temps de recherche que j'ai trouvé la solution à mon problème. Il fallait utiliser différemment la fonction setAlignment().

Layout->setAlignment(QLineEdit,Qt::AlignCenter);

Layout->setAlignment(QPushButton,Qt::AlignCenter);

J'ai pu par la suite placer mes widgets où bon me semblait dans mes layout.

À la fin de la création d'une page, je place tout mes QGroupBox dans le layout final que je définis comme layout du QWidget de la page correspondante.

```
finalGrid->addWidget(gBoxTop,0,0,1,3);
finalGrid->addWidget(gBoxGauche,1,0,2,1);
finalGrid->addWidget(gBox1,1,1,2,1);
finalGrid->addWidget(gBox1V2,1,1,2,1);
finalGrid->addWidget(gBoxDroite,1,2,2,1);
page1->setLayout(finalGrid);
```

Les déclarations des widgets et les fonctions de création de page étant répétitives, le fichier gestion_intendance.h sera disponible en annexe pour plus de précisions ainsi que les définitions de toutes les fonctions de création de page.

Voici donc à quoi ressemble ma page d'accueil une fois mise en place.

L'utilisateur peut déjà changer d'onglet mais les QlineEdit et les QpushButton sont inutilisables car les pages ont été désactivées dans le constructeur. L'utilisateur doit donc s'identifier pour déverrouiller les onglets suivants.

Il n'y a pas de comptes différents, cette partie d'identification sert de sécurité, pour que l'application ne soit pas utilisable par n'importe qui.

Le bouton valider de cette page est lié à la fonction « connexion » que voici :

```
void Gestion_Intendance::connexion()
{
    QSqlQuery query;
    QString requete = "SELECT * FROM `identification`";
    QString log = "", pwd = "";
    query.exec(requete);
    if(query.lastError().text()==" ")
    {
        while(query.next()) // Récupérer
        {
            log = query.value(0).toString();
            pwd = query.value(1).toString();
        }
    }
```

Les champs de la table identification de la base de données sont récupérés dans les variables « log » et « pwd ».

login	password
admin	admin

```
    if (editLogin->text()==log && editPassword->text()==pwd) // Si le login et le mdp sont da
    {
        page2->setDisabled(false);
        page3->setDisabled(false);
        page4->setDisabled(false);
        page5->setDisabled(false);
        txtLoginRecup->setText("Compte: "+log);
        gBox1->hide();
        gBox1V2->show();
    }
    else
        QMessageBox::information(this,("ERREUR"),("Mot de passe ou identifiant incorrect"));
}
```

Si les logins et mot de passe entrés par l'utilisateur sont identiques à ceux récupérés dans la base de données, les pages sont déverrouillées, le QGroupBox servant à se connecter est caché et un

autre vient prendre sa place, affichant le compte connecté et un bouton « Déconnexion ».

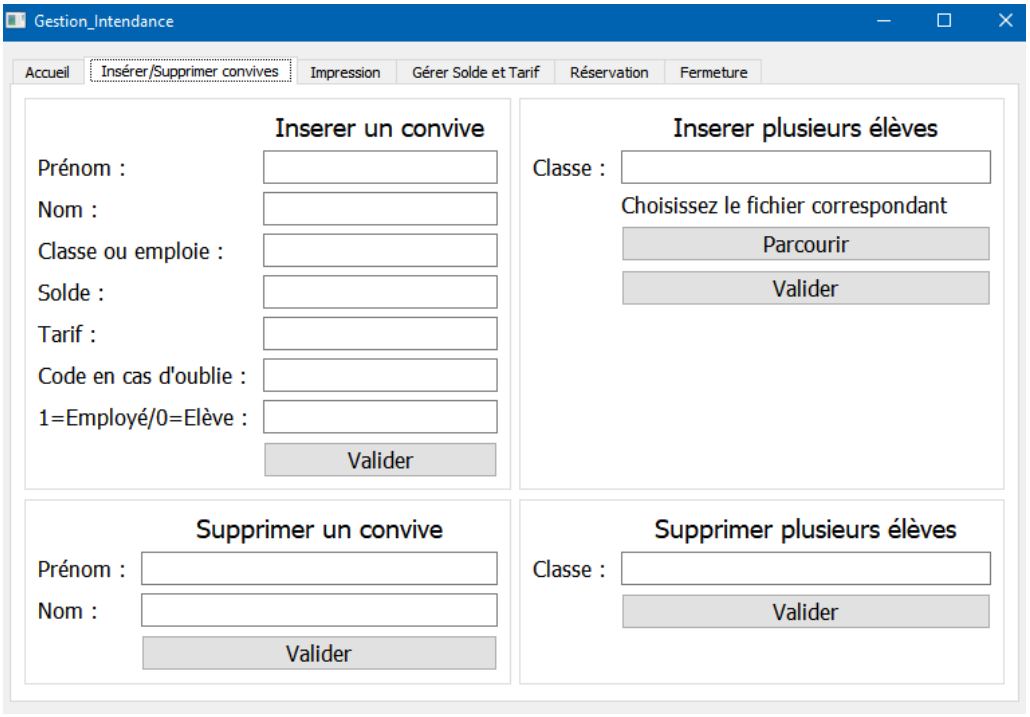
Ce bouton est lié à une fonction « déconnexion » :

```
void Gestion_Intendance::deconnexion()
{
    // Clear et verrouille les onglets
    foreach(QLineEdit* le, findChildren<QLineEdit*>()) {
        le->clear();
    }
    page2->setDisabled(true);
    page3->setDisabled(true);
    page4->setDisabled(true);
    page5->setDisabled(true);
    gBox1->show();
    gBox1V2->hide();

}
```

qui vide tout les champs de toutes les pages, qui les reverrouille puis qui ré-affiche le groupe box de connexion.

- 4.2 Concevoir/Coder/Tester le module logiciel pour gérer la base de données

L'intendance doit pouvoir ajouter ou supprimer des élèves de la base de données, en cas d'élève transféré et à chaque nouvelle année, ajouter les nouveaux élèves et supprimer ceux ayant obtenu leur BAC.
Mon application doit donc permettre d'ajouter ou de supprimer des convives à l'unité mais aussi de pouvoir ajouter ou supprimer des classes directement.



La fonction « creationPage2 » est disponible en annexe.

- 4.2.1 Ajouter un convive

Le bouton valider du groupe box « Insérer un convive » est lié à la fonction « insererConvive » que voici :

```
void Gestion_Intendance::insererConvive()
{
    QString requete = "INSERT INTO convive
    executerRequete(requete);
    editPrenomInser->clear();
    editNomInser->clear();
    editClasse_EmploieInser->clear();
    editSoldeInser->clear();
    editTarifInser->clear();
    editCodeInser->clear();
    editStatutInser->clear();
}
```

requête entière : "INSERT INTO convive (nom,prenom,classe_emploie,solde,tarif,code_secret,statut) SELECT '"+editNomInser->text()+"','"+editPrenomInser->text()+"','"+editClasse_EmploieInser->text()+"','"+editSoldeInser->text()+"','"+editTarifInser->text()+"','"+editCodeInser->text()+"','"+editStatutInser->text()+"' AS tmp WHERE NOT EXISTS (SELECT convive.nom, convive.prenom FROM convive WHERE convive.nom = '"+editNomInser->text()+ "' AND convive.prenom = '"+editPrenomInser->text()+"')";

Cette fonction ajoute toutes les informations entrées dans les champs par l'utilisateur pour remplir une ligne de la table « convive » sauf si le convive existe déjà dans la base de données.
Elle vide aussi tous les champs du groupe box.

Pour cette fonction, j'utilisais à l'origine une requête un peu différente, jusqu'à ce que mon camarade de projet William Malledant, découvre en testant mon application que la requête ne permettait pas de remplir de champ par une valeur identique. Par exemple, si le convive est un élève, alors l'utilisateur rentrera « 0 » dans le champ du statut, s' il souhaitait mettre le solde de ce convive à « 0 » une erreur survenait et le convive n'était pas inséré. Nous avons donc vérifier si c'était aussi le cas pour une même chaîne de caractère dans les champs nom et prénom, et effectivement la même erreur empêchait l'insertion du convive. J'ai donc dû rechercher une requête acceptant des valeurs identiques dans des champs différents, et j'ai finalement trouvé celle-ci :

id ▾ 1	nom	prenom	classe_emploie	solde	tarif	matricul	code_secret	statut 1 = employé / 0 = eleve
2311	BOURDON	Randy	SN3	1000	3	0	7777	0
2306	PIGAL	Romain	SN2	0	4	0	8483	0
2305	PAO	Eric	SN2	0	4	0	6040	0
2304	PAIN	Ludovic	SN2	0	4	0	6039	0
2303	MELOULI	Dalil	SN2	0	4	0	6179	0

Extrait de la table « convive » de la base de données :

- 4.2.2 Ajouter plusieurs convives

La difficulté supplémentaire pour l'ajout de plusieurs convives, est d'aller récupérer les informations d'un fichier « .csv » pour les insérer dans la base de données. Il m'a fallu effectuer de longues recherches pour enfin trouver la ligne de code qui convenait pour remplir au mieux cette tâche.
Le bouton parcourir, lié à la fonction du même nom, sert à récupérer le chemin jusqu'à ce fichier. La fonction ouvre un explorateur de fichier pour que l'utilisateur puisse aller chercher lui même son fichier.

```
void Gestion_Intendance::parcourir()
{
    dossierChoisi = QFileDialog::getOpenFileName(this, "Fichier CSV", "P:");
}
```

Ensuite la fonction « insererPlusieursConvives » lié au bouton valider de ce groupe box, lit par ligne le fichier et sépare les champs pour récupérer le nom, le prénom, et le code d'un convive, récupère le nom de la classe entré par l'utilisateur pour les insérer dans la base de données, la requête met au passage des valeurs par défaut pour les autres champs.
Ensuite une boucle permet de passer à la ligne suivante pour ajouter l'élève suivant et ainsi de suite, jusqu'à ce qu'il n'y ait plus d'élève à ajouter. Bien sûr, si l'élève existe déjà dans la base de données il n'est pas ajouté.

```
Nom de famille;Prénom 1;Date de naiss
AZAK;Mehmet;01/07/1999;7250;2 A rue d
BOURDON;Randy;25/09/1998;7819;9 avenu
CARRIERE;Yann;02/02/1999;8469;117 rue
GALTIER;Brice;11/12/1998;8472;36, RUE
GAYAUD;Loic;07/05/1999;8520;43, Rue L
HANDAS;Daoud;22/10/1997;7823;2, Bis R
HANNES TOURNIER;Alex;20/03/1999;8473;
HANNES TOURNIER;Alex;20/03/1999;8473;
IKACHE;Mohamed-Bilel;29/05/1997;8513;
KOHLER;Nicolas;27/06/1999;8474;23, Al
KOVACS;Florian;14/04/1999;8475;5 Rue
LAKHMECHE;Vincent;15/08/1999;8478;22
LUSTGARTEN;Léo;23/09/1999;6523;33 rue
MACQUE;Sébastien;03/11/1998;5987;4 ru
MALLEDANT;William;31/12/1999;8481;15
MALLEDANT;William;31/12/1999;8481;68
MARTIN;Damien;16/04/1999;8482;7 RUE A
MELOULI;Dalil;05/10/1998;6179;12 aven
PAIN;Ludovic;30/05/1998;6039;5 rue Sa
PAO;Eric;19/01/1998;6040;161 rue Paul
PIGAL;Romain;22/11/1998;8483;22 rue d
PIGAL;Romain;22/11/1998;8483;29 chemi
```

Extrait du fichier SN2.csv
ouvert avec Notepad++

Là aussi j'ai eu pas mal de recherche à faire avant d'arriver au résultat souhaité. Il m'a fallu faire quelques recherches pour trouver comment ouvrir le fichier, mais surtout pour comprendre comment utiliser la fonction « readLine », que j'ai choisi d'utiliser après avoir essayé avec la fonction « readAll » sans succès. Je n'avais pas compris que l'argument de « readLine » était le nombre de caractère maximum de la ligne à lire. Une fois ces problèmes réglés, j'ai pu réussir l'insertion d'une classe grâce à son fichier « .csv » avec la fonction suivante :

```
void Gestion_Intendance::insererPlusieursConvives()
{
    QFile fichier(dossierChoisi);
    fichier.open(QIODevice::ReadWrite); // ouverture du fichier
    QTextStream flux(&fichier); // récupération du contenu du fichier
    QString ligne = flux.readLine(200); // lecture de la première ligne
    while (ligne != "") // tant qu'il y'a une ligne a lire
    {
        QStringList champ = ligne.split(";");
        QString requete = "INSERT INTO convive (nom,prenom,classe_emploi
        executerRequete(requete);
        ligne = flux.readLine(200); // ligne suivante
    }
    editClasseInserPlus->clear();
}
```

Requête entière : "INSERT INTO convive

(nom,prenom,classe_emploie,solde,tarif,code_secret,statut) SELECT '"+champ.at(0)+"','"+champ.at(1)+"','"+editClasseInserPlus->text()+"','0','4','"+champ.at(3)+"','0' AS tmp WHERE NOT EXISTS (SELECT convive.nom, convive.prenom FROM convive WHERE convive.nom = '"+champ.at(0)+"' AND convive.prenom = '"+champ.at(1)+"')";

Une fois cela fait le QlineEdit est vidé.

- 4.2.3 Supprimer des convives

Pour la suppression de convive tout est plus simple. Pour en supprimer un seul, le bouton valider du groupe box « Supprimer un convive » appelle la fonction suivante :

```
void Gestion_Intendance::supprimerConvive()
{
    QString requete = "DELETE FROM convive WHERE nom='"+editNomSup->text()+"' AND prenom='"+editPrenomSup->text()+"'";
    executerRequete(requete);
    editNomSup->clear();
    editPrenomSup->clear();
}
```

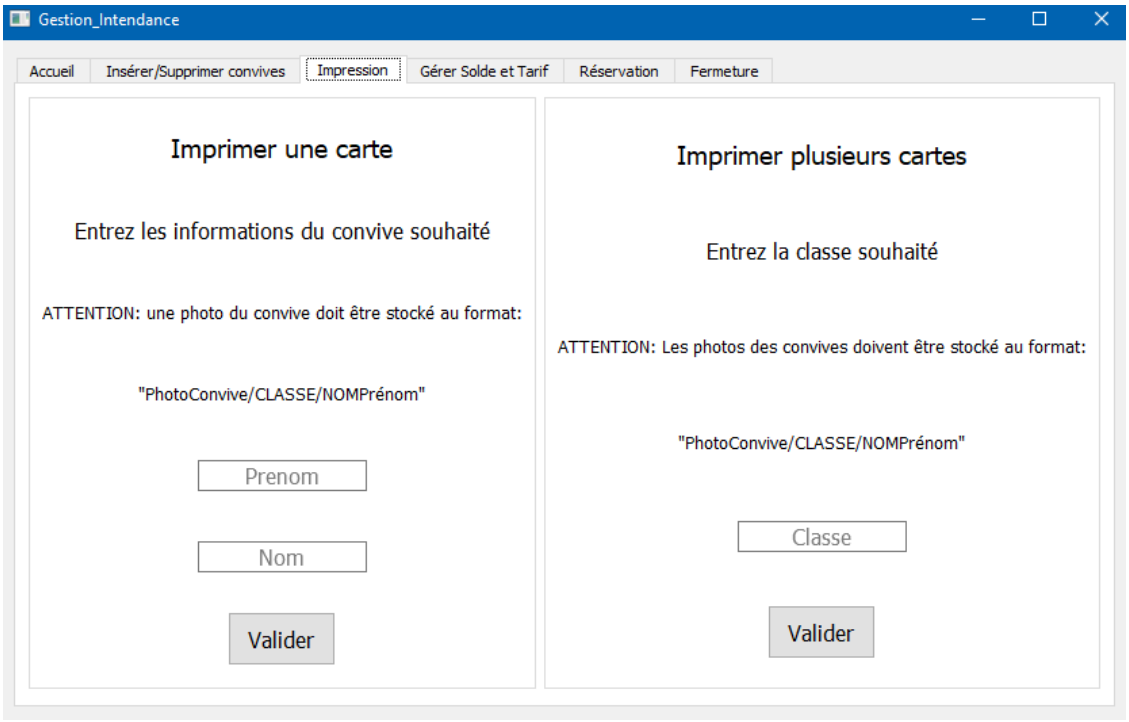
Elle supprime simplement dans la base de données le convive ayant le nom et prénom entré par l'utilisateur, avant de vider les champs de son groupe box.

Pour supprimer plusieurs convives, le bouton valider de ce groupe box appelle une fonction très proche de la précédente.

```
void Gestion_Intendance::supprimerPlusieursConvives()
{
    QString requete = "DELETE FROM convive WHERE classe_emploie='"+editClasseSup->text()+"'";
    executerRequete(requete);
    editClasseSup->clear();
}
```

Elle supprime tous les convives ayant pour classe, la classe entrée dans le QlineEdit avant de vider celui-ci.

- 4.3 Concevoir/Coder/Tester le module logiciel pour stocker les images de cartes de cantines dans un fichier PDF et pour lancer l'impression sur une imprimante



Depuis le début du projet, je m'imaginais la partie impression comme étant la plus complexe du projet. Cela s'est confirmé quand j'ai dû m'y atteler. Mon professeur m'avait fournit un programme qui permettait d'imprimer un document texte, et de créer un PDF, seulement la création de PDF ne fonctionnait pas.

Après pas mal de recherche, j'ai compris que la fonction « print » ne pourrait fonctionner qu'avec un document texte mais cette partie contenait un QPrintDialog qui ouvre une fenêtre permettant de choisir son imprimante et ses paramètres. Cependant, mon but étant d'automatiser la création des cartes, cette alternative demandait trop d'action pour l'utilisateur, et était donc pour lui une perte de temps. De plus, l'imprimante à carte étant branchée par USB, elle était par défaut utilisée pour l'impression sans que l'on ait besoin de la sélectionner.

Je me suis donc tourné vers l'utilisation de QPrinter pour paramétrer l'impression et de QPainter servant à « dessiner ». J'ai rapidement réussi a créer un fichier PDF grâce a l'extrait de programme de mon professeur. Avec un

QPainter je plaçais simplement une image de fond pour la carte avant d'y ajouter les informations de l'élève et sa photo. Cependant, si j'arrivais facilement à créer un PDF, je n'arrivais pas à trouver comment imprimer une carte.

Après 2 jours de recherche, et après avoir lu plusieurs fois la documentation de QPainter et celle de QPrinter sur le site de Qt Creator (donc en anglais), j'ai finalement trouvé l'endroit où la doc de QPrinter évoquait ce dont j'avais besoin, mais je n'arrivais pas à la comprendre correctement. J'ai donc demandé de l'aide à mon camarade de projet William Malledant, ce qu'il comprenait de ce passage et ses explications m'ont permis de me débloquer.

QPrinter permet au choix, de créer un PDF, dessiner avec un QPainter, ou d'imprimer directement le dessin du QPainter. Pour pouvoir simplement imprimer il ne faut pas donner de format et de nom de sortie à son QPrinter, alors le dessin s'imprime automatiquement sur l'imprimante branchée en USB.

J'ai donc enfin pu créer une fonction permettant au choix de créer un PDF de la carte ou d'imprimer celle-ci directement.

- 4.3.1 Imprimer une carte/créer un PDF

Le bouton valider du groupe box « Imprimer une carte » appelle la fonction qui récupère dans la base de données les informations du convive.

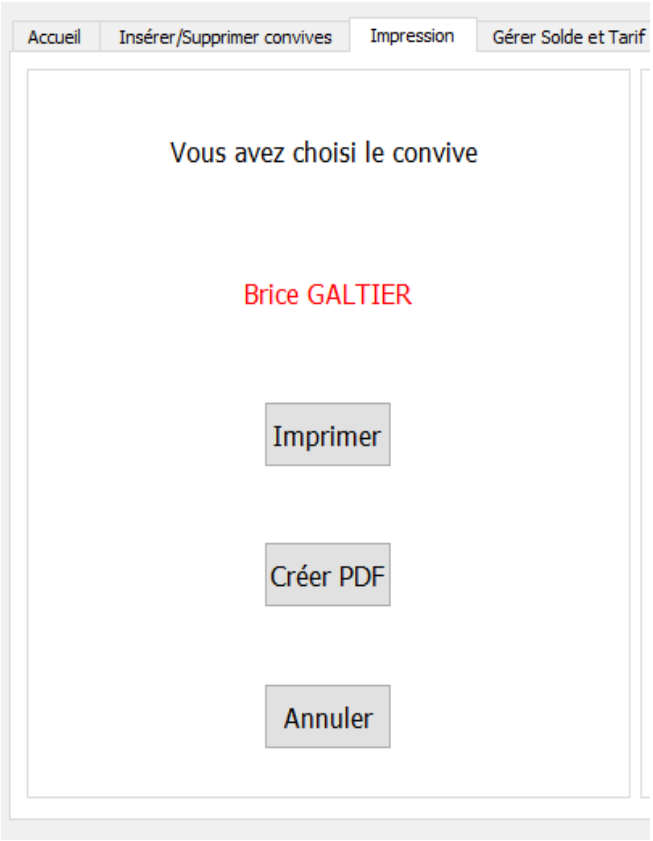
```
void Gestion_Intendance::selectImpress()
{
    QSqlQuery query;
    QString requete = "SELECT prenom, nom, classe_em
    query.exec(requete);
    recupPrenom = "";
    recupNom = "";
    if(query.lastError().text()==" ")
    {
        while(query.next()) // Récupérer
        {
            recupPrenom = query.value(0).toString();
            recupNom = query.value(1).toString();
            recupClasse = query.value(2).toString();
            recupCode = query.value(3).toInt();
        }
```

Requête entière : "SELECT prenom, nom, classe_emploie, code_secret FROM `convive` WHERE convive.nom='"+editNomImpress->text()+"' AND convive.prenom='"+editPrenomImpress->text()+"'"

Ensuite la fonction vérifie que des valeurs ont bien été récupérées. Si ce n'est pas le cas, cela signifie que les informations entrées par l'utilisateur sont incorrectes, la fonction ouvre donc un QMessageBox pour prévenir l'utilisateur de son erreur.

```
        if (recupNom=="" && recupPrenom=="") // Si la personne n'a pas été trouvé
            QMessageBox::information(this,("ERREUR"),("Les informations entrées sont incorrect, veuillez réessayer"));
        else
        {
            txtAvantImpressInfo->setText(recupPrenom+" "+recupNom);
            gBoxGauche3->hide();
            gBoxGauche3V2->show();
        }
    }
}
```

Si les informations sont correctes, un nouveau groupe box vient remplacer l'actuel.



Nous avons donc trois nouveaux boutons.

Le bouton annuler lié à la fonction « annulerCarte » sert simplement à revenir en arrière, il ré-affiche le premier groupe box et vide ses QLineEdit.

```
void Gestion_Intendance::annulerCarte()
{
    editNomImpress->clear();
    editPrenomImpress->clear();
    gBoxGauche3V2->hide();
    gBoxGauche3->show();
}
```

```
void Gestion_Intendance::imprimer()
{
    IMPouPDF = true;
    creerCarte();
}
void Gestion_Intendance::creerPDF()
{
    IMPouPDF = false;
    creerCarte();
}
```

La fonction « imprimer » lié au bouton Imprimer et la fonction « creerPDF » lié au bouton Créer PDF, servent juste à modifier la valeur d'une variable de type bool.

Avant de rediriger vers une dernière fonction qui imprimera ou créera un PDF en fonction de l'option choisie.

```
void Gestion_Intendance::creerCarte()
{
    if (IMPouPDF==false)
    {
        printer.setOutputFormat(QPrinter::PdfFormat); // Définir le format
        printer.setOutputFileName("CarteConvive/Carte"+recupNom+recupPrenom+".pdf"); // Définir le nom du fichier
    }
```

Si c'est le bouton Créer PDF qui a été cliqué alors la variable « IMPouPDF » aura la valeur false, le printer aura donc le type de format PDF de prédéfini ainsi qu'un nom de sortie, un PDF de ce nom sera alors créé.
Si c'est le bouton Imprimer qui a été cliqué, alors la variable « IMPouPDF » aura la valeur true, le printer n'aura ni format ni nom de sortie et partira directement vers l'imprimante.
Quelques paramètres sont ensuite indiqués pour que le format colle bien à la taille de la carte.

```
printer.setOrientation(QPrinter::Landscape); // Définir l'orientation
printer.setPaperSize(QSize(54,86),QPrinter::Millimeter); // Définir la taille
printer.setPageMargins(0,0,0,0,QPrinter::Millimeter); // Définir les marges
```

Puis un QPainter lié à notre QPrinter permet de créer la carte en ajoutant les images et le texte.

```
QFont font("Times", 12, QFont::Bold);
QPixmap templateCarte("template.png");
QPixmap photoConvive("PhotoConvive/"+recupClasse+"/"+recupNom+recupPrenom);
painter.begin(&printer);
painter.drawPixmap(0,0,printer.width(),printer.height(),templateCarte); // Ajouter une image
painter.drawPixmap(200,30,90,110,photoConvive);
painter.setFont(font);
painter.drawText(15, 100, recupNom); // Ajouter du texte
painter.drawText(15, 120, recupPrenom);
painter.drawText(105, 140, QString::number(recupCode));
painter.end();
```

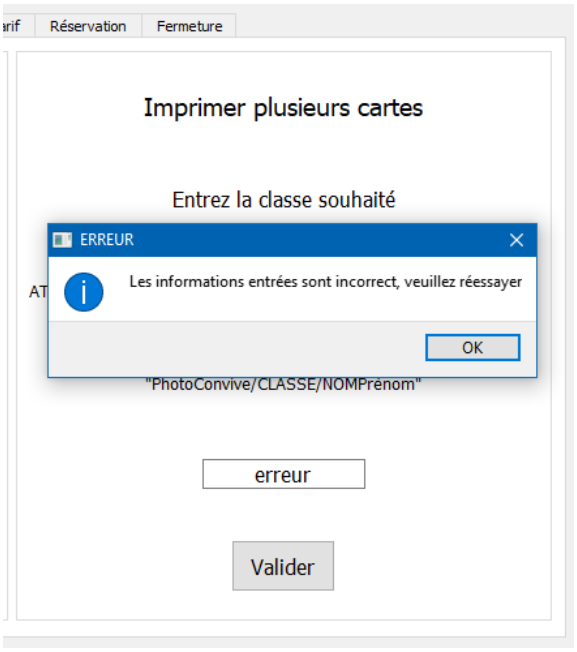
J'ai modifié la police du texte et j'ai placé chaque ligne de texte et photo en fonction des pixels.

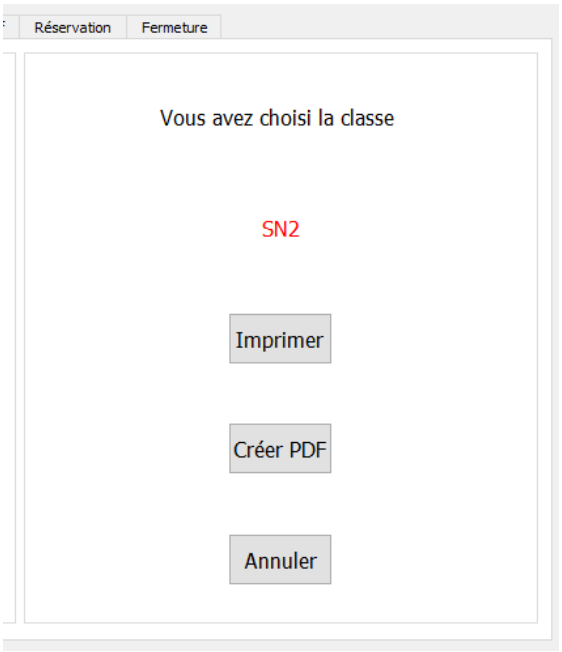
```
    editNomImpress->clear();
    editPrenomImpress->clear();
    gBoxGauche3V2->hide();
    gBoxGauche3->show();
}
```

Enfin le premier groupe box est ré-affiché avec ses QLineEdit vidés.

- 4.3.2 Imprimer plusieurs cartes/créer plusieurs PDF

Pour imprimer plusieurs cartes c'est le même principe au départ. On vérifie d'abord que la classe entrée par l'utilisateur existe dans la base de données, avant d'afficher soit un message d'erreur soit un autre groupe box.





On retrouve ensuite le même principe que pour le groupe box « Imprimer une carte » pour les trois boutons affichés.

```
void Gestion_Intendance::imprimer2()
{
    IMPouPDF = true;
    creerPlusieursCartes();
}
void Gestion_Intendance::creerPDF2()
{
    IMPouPDF = false;
    creerPlusieursCartes();
}
void Gestion_Intendance::annulerCarte2()
{
    editClasseImpress->clear();
    gBoxDroite3V2->hide();
    gBoxDroite3->show();
}
```

La fonction « creerPlusieursCartes » est, elle, bien différente. La partie création du PDF ou envoi de l'impression à l'imprimante est cette fois, dans la boucle de récupération des informations de la base de données. Les informations d'un élève sont récupérées, sa carte est créée, puis c'est au tour de l'élève suivant, et ce jusqu'à ce que tous les élèves de la classe sélectionnée soient passés.

```
void Gestion_Intendance::creerPlusieursCartes()
{
    QMessageBox::information(this,("Patientez un instant"),("Les cartes sont en cours de création"));
    QSqlQuery query;
    QString requete = "SELECT prenom, nom, classe_emploie, code_secret FROM `convive` WHERE convive.cl
    query.exec(requete);
    recupClasse = "";
    if(query.lastError().text()==" ")
    {
        while(query.next()) // Récupérer
        {
            recupPrenom = query.value(0).toString();
            recupNom = query.value(1).toString();
            recupClasse = query.value(2).toString();
            recupCode = query.value(3).toInt();
            if (IMPouPDF==false)
            {
                printer.setOutputFormat(QPrinter::PdfFormat); // Définir le format
                printer.setOutputFileName("CarteConvive/Carte"+recupNom+recupPrenom+".pdf"); // Défini
            }
            printer.setOrientation(QPrinter::Landscape); // Définir l'orientation
            printer.setPaperSize(QSize(54,86),QPrinter::Millimeter); // Définir la taille
            printer.setPageMargins(0,0,0,0,QPrinter::Millimeter); // Définir les marges
            QFont font("Times", 12, QFont::Bold);
            QPixmap templateCarte("template.png");
            QPixmap photoConvive("PhotoConvive/"+recupClasse+"/"+recupNom+recupPrenom);
            painter.begin(&printer);
            painter.drawPixmap(0,0,printer.width(),printer.height(),templateCarte); // Ajouter une ima
            painter.drawPixmap(200,30,90,110,photoConvive);
            painter.setFont(font);
            painter.drawText(15, 100, recupNom); // Ajouter du texte
            painter.drawText(15, 120, recupPrenom);
            painter.drawText(105, 140, QString::number(recupCode));
            painter.end();
        }
        editClasseImpress->clear();
        gBoxDroite3V2->hide();
        gBoxDroite3->show();
    }
}
```

Requête entière : "SELECT prenom, nom, classe_emploie, code_secret FROM `convive` WHERE convive.classe_emploie='"+editClasseImpress->text()+"'";

Ensuite le premier groupe box reprend sa place avec son QLineEdit vide.

- 4.3.3 Résultat de création de PDF et résultat d'impression

Voici un exemple de résultat obtenu :

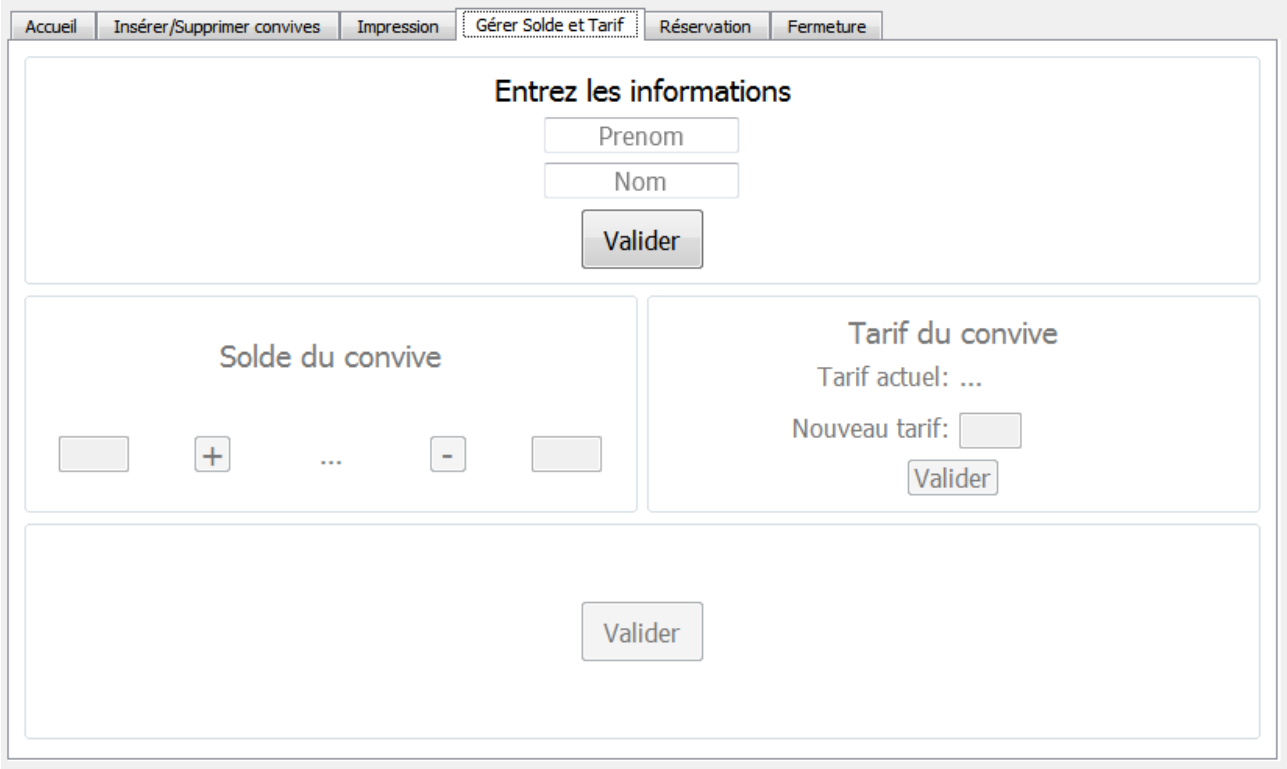
Voici le fichier PDF créé pour l'élève Brice Galtier.

Et voici le résultat de l'impression.

Une bande sur le haut de la carte reste non imprimé, ainsi que la bande bleu sur la droite. Le PDF étant correctement créé nous avons supposé qu'il s'agissait d'un problème sur l'imprimante, mais nous n'avons pas pu trouver un moyen de régler ce problème.

- 4.4 Concevoir/Coder/Tester le module logiciel pour acheter les repas

L'intendance doit pouvoir gérer le solde et le tarif des convives.



L'utilisateur doit d'abord entrer le nom et le prénom du convive de son choix. Comme pour la partie impression une vérification est faite, un QmessageBox est affiché en cas d'erreur sinon le groupe box d'identification affiche les informations du convive entré et les autres groupes box s'activent.

Le solde et le tarif du convive sont récupérés dans la base de données pour être affichés dans les groupes box correspondants.

- 4.4.1 Gérer le solde

Pour la partie solde, les boutons « + » et « - » servent à ajouter ou retirer les valeurs entrées dans les QLineEdit correspondant.

```
void Gestion_Intendance::additionnerSolde()
{
    solde = solde + editPlusSolde->text().toDouble();
    soldeAffichage->setText(QString::number(solde)+" €");
    editPlusSolde->clear();
}
void Gestion_Intendance::soustraireSolde()
{
    solde = solde - editMoinsSolde->text().toDouble();
    soldeAffichage->setText(QString::number(solde)+" €");
    editMoinsSolde->clear();
}
```

La variable étant de type double, les valeurs décimales sont prises en compte.

J'ai effectué de nombreux tests différents avant de réussir à afficher correctement le nouveau solde au moment où l'utilisateur clique sur le bouton. Au début, le nouveau solde était affiché par dessus l'ancien et était donc illisible.

C'est mon camarade de projet Randy Bourdon qui m'a expliqué comment régler simplement ce problème.

- 4.4.2 Gérer le tarif

Pour la partie tarif le bouton valider sert à remplacer le tarif actuel par le nouveau tarif.

```
void Gestion_Intendance::validerTarif()
{
    if (editNouveauTarif->text().toDouble()<0)
        QMessageBox::information(this,("ERREUR"),("Le tarif ne peut pas être inférieur à 0"));
    else
    {
        tarif = editNouveauTarif->text().toDouble();
        ancienTarif->setText(QString::number(tarif)+" €");
        editNouveauTarif->clear();
    }
}
```

Il m'a fallu un peu de temps pour réussir à placer mes widgets comme je le souhaitais.

Enfin le bouton valider en bas de la page sert à envoyer les informations modifiées dans la base de données.

Requête : `"UPDATE convive SET solde='"+QString::number(solde)+"', tarif='"+QString::number(tarif)+"' WHERE convive.nom='"+txtNomSolde->text()+"' AND convive.prenom='"+txtPrenomSolde->text()+"'"`

Puis ré-affiche la page comme elle était à l'origine.

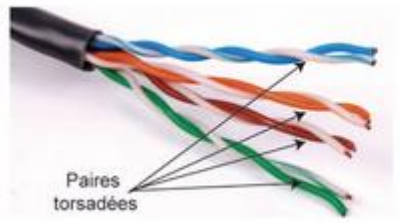
- 4.5 Concevoir/Coder/Tester le module logiciel pour réserver des repas

Cette partie n'a pas encore été conçue par manque de temps.

5. Physique Appliquée

- 5.1 RJ45

Composition d'un cable RJ45



Le câble RJ-45 correspond au connecteur 8P8C, qui signifie 8 positions et 8 contacts électriques. Il est utilisé principalement pour les **connexions Ethernet** ainsi que pour les **connecteurs téléphoniques**. Comme le précise son nom 8P8C, ce connecteur présente huit broches de connexion électrique. Un connecteur rj45 est une interface physique, qui est régulièrement utilisée pour finir les **câbles de type paire torsadée**.

Les fils **1-2** et **3-6** sont utilisés pour le câblage du **réseau Ethernet 10/100 Mbit/s**
Alors que pour transmettre ces mêmes informations en **1 000Mbit/s** (1 Gbit/s), les **4 paires** sont utilisées.
Les fils **4-5** sont utilisés pour le **téléphone** via France Télécom
Les fils **7-8** sont utilisés pour la **télévision** ou la ligne 2 du téléphone.

Il existe des câbles comme le câble ethernet blindé, le câble ethernet à double blindage et le câble ethernet non blindé :

- si il y a un feuillet d'aluminium autour de l'ensemble des paires
- si il y a un feuillet d'aluminium autour de chacune des paires
- ou les 2 en mêmes temps

Le débit d'un câble Ethernet.

Le débit d'un câble ethernet en 10/100 Mbps représente ce que l'on appel le Fast Ethernet, le 1000 Mbps lui est aussi appelé Gigabit et enfin le 10000 Mbps lui est appelé 10G ou 10 Gigabits. Cela désigne la quantité d'informations transmises par unité de temps, c'est-à-dire la vitesse à laquelle les informations circulent sur le réseau. Il se mesure en bits ou en octets par secondes.

- 5.1 USB

Vitesse de transfert de l'USB

L'USB supporte 3 vitesses :

Low Speed à 1.5Mbit/s – (USB 1.1)
Full Speed à 12Mbit/ s – (USB 1.1)
High Speed à 480Mbit/ s – (USB 2.0)

Tout les PC supporte actuellement deux vitesses de bus, le Full Speed et le Low Speed. La vitesse High Speed a été ajoutée avec l'apparition de la spécification USB 2.0. Cependant, pour pouvoir utiliser cette vitesse de transfert, il faut être équipé de cartes mères et de contrôleurs USB supportant l'USB 2.0.

L'imprimante Zebra P330 une liaison USB 1.1.

Les débits de l'USB

Il faut noter que dans la norme USB le débit n'est pas proportionnel à la vitesse. L'USB « Low Speed » qui est consacré aux périphériques soit disant lents est limités à échanger aux maximum 8 octets toutes les 10ms, ce qui correspond à un débit maximum de 800 octets/sec soit 6400 bps. Par certains moyens détournés il est possible d'espérer atteindre 8Ko/s, mais c'est sans garantie. L'USB « Full Speed » qui est consacré aux HUBs et aux périphériques non lents peut échanger jusqu'à 1024 octets toutes les ms, soit un débit de 1 Mo/s.

Composition du câble USB

Chaque connecteur dispose de deux fils d'alimentation (5V et GND) et deux fils destinés au transfert de données (D+ et D-).

En version Low Speed le blindage n'est pas obligatoire (ce qui assure une plus grande souplesse de manipulation en particulier pour une liaison souris).

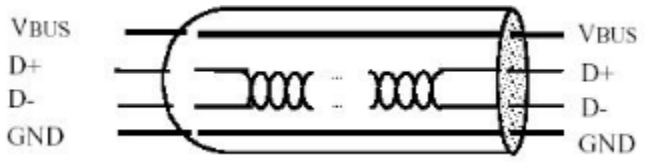


Figure 1 : Composition d'un câble USB

La longueur maximale autorisée par la norme est de 3m pour un câble non blindé donc généralement pour un périphérique Low USB (= 1.5Mb/s) et de 5m pour un câble blindé dans le cas d'un périphérique Full USB (=12Mb/s).

6. Amélioration possible

Concevoir/Coder/Tester le module logiciel pour gérer la base de données

L'interface que j'ai fait ne permet pas de voir les convives déjà dans la base de donnée, il faudrait faire une interface ou l'utilisateur puisse avoir une liste de tout les convives pour y apporter des modifications, en plus d'un partie insertion suppression.

Concevoir/Coder/Tester le module logiciel pour stocker les images de cartes de cantine dans un fichier PDF et pour lancer l'impression sur une imprimante.

J'ai pendant un long moment essayé d'ajouter un système d'aperçu avant impression mais malheureusement je n'ai pas trouvé comment faire malgré de très longues recherches. Ce serait une bonne idée amélioration si jamais elle est possible en C++ sur Qt.

Ajout d'un onglet « Fermeture »

Afin que les réservations ne soient pas possible le jour ou le lycée est fermé, nous avons rajouter des tables dans la base de donnée contenant les dates de vacances et les dates des

jours fériés. Nous avions donc pensé à ajouter un onglet fermeture à mon application afin que l'intendance puisse entrer les jours de fermetures, que ce sois les vacances, les jours fériés ou même des fermetures occasionelles.

7. Conclusion

Je n'ai pas totalement terminé ma partie du projet car il me manque l'onglet « Réservation ».

Cependant le reste de mon application fonctionne correctement et est prête a être utilisé.

IV. PARTIE PERSONNELLE

ETUDIANT 3 – MALLEDANT WILLIAM

Présentation partie personnelle étudiant trois.

L'étudiant trois à pour mission le développement de l'interface en cas d'oublie de carte, la vérification des réservations pour ensuite gérer le du distributeur de plateau, n'ayant pas accès à un véritable distributeur de plateau, cette partie ce fera en simulation à l'aide d'un raspberry Pi 3 et d'un module EXP-500.

Ressource :

- C++
- IDE QT Creator
- WampServeur
- PHPMyAdmin
- PC portable sous OS Windows 10
- Raspberry Pi 3
- Module EXP-500
- Raspberry Pi 3 avec écran tactile

L'écran tactile et son support qui intègre le raspberry Pi 3.

Diagrammes :

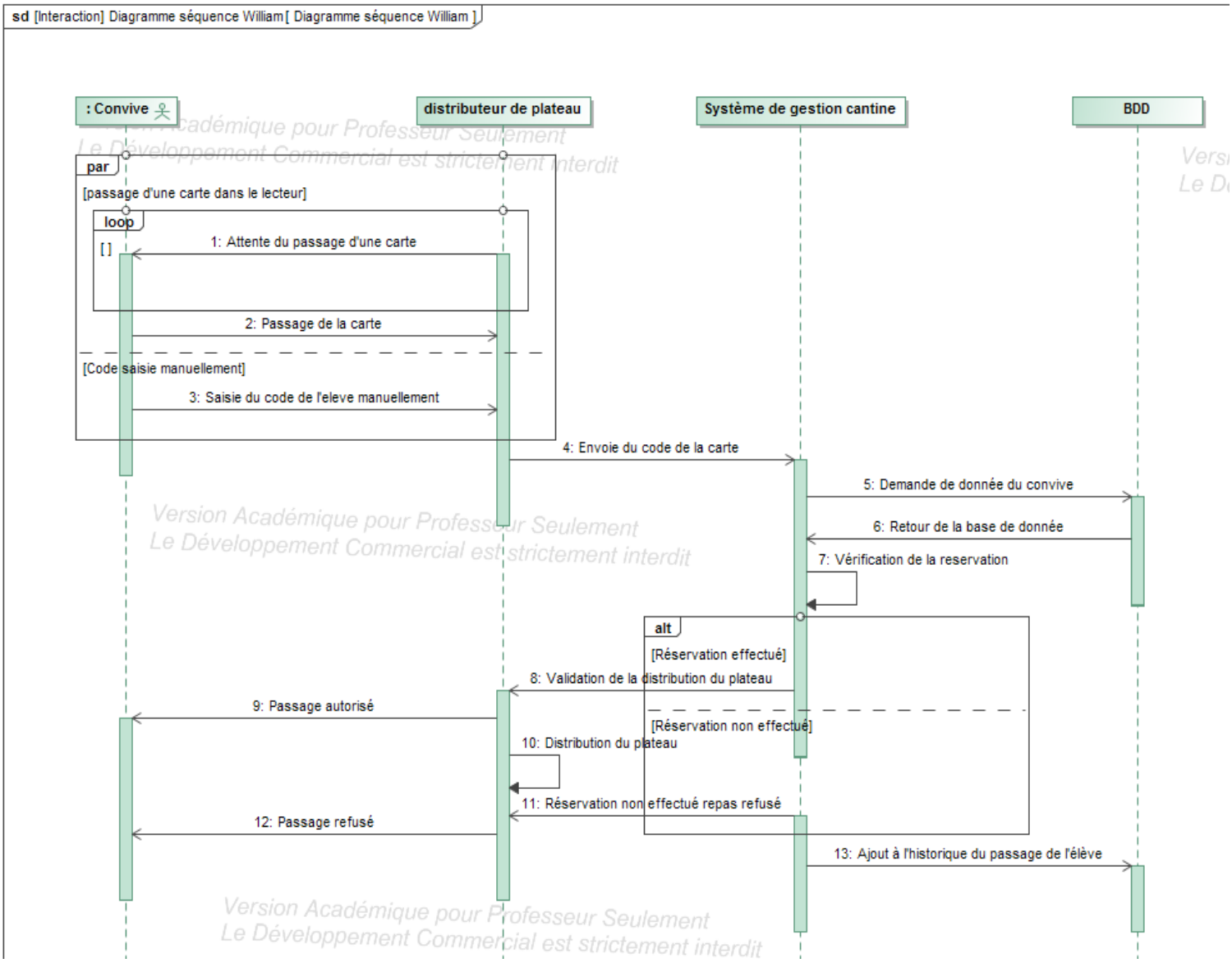


Diagramme de séquence

Installation des outils de développement sur les systèmes embarqués.

Pour débuter le projet il a d'abord fallu configurer tous les systèmes qui nous servirai par la suite.

J’ai donc commencé par récupérer un ordinateur portable (fourni par l’établissement) et par installer Windows 10 sur ce dernier. Après cela j’ai dû également configurer les deux raspberry PI 3 pour qu’il soit contrôlables en bureau à distance et donc fixé leurs IP (sur des adresses disponibles fournit par l’établissement) afin que nous puissions tous nous en servir. Pour ce faire après avoir fixé l’adresse IP j’ai installé VNC Server sur chaque raspberry et VNC Viewer sur les ordinateurs de mes collègues.

Principe de fonctionnement des écrans tactile :

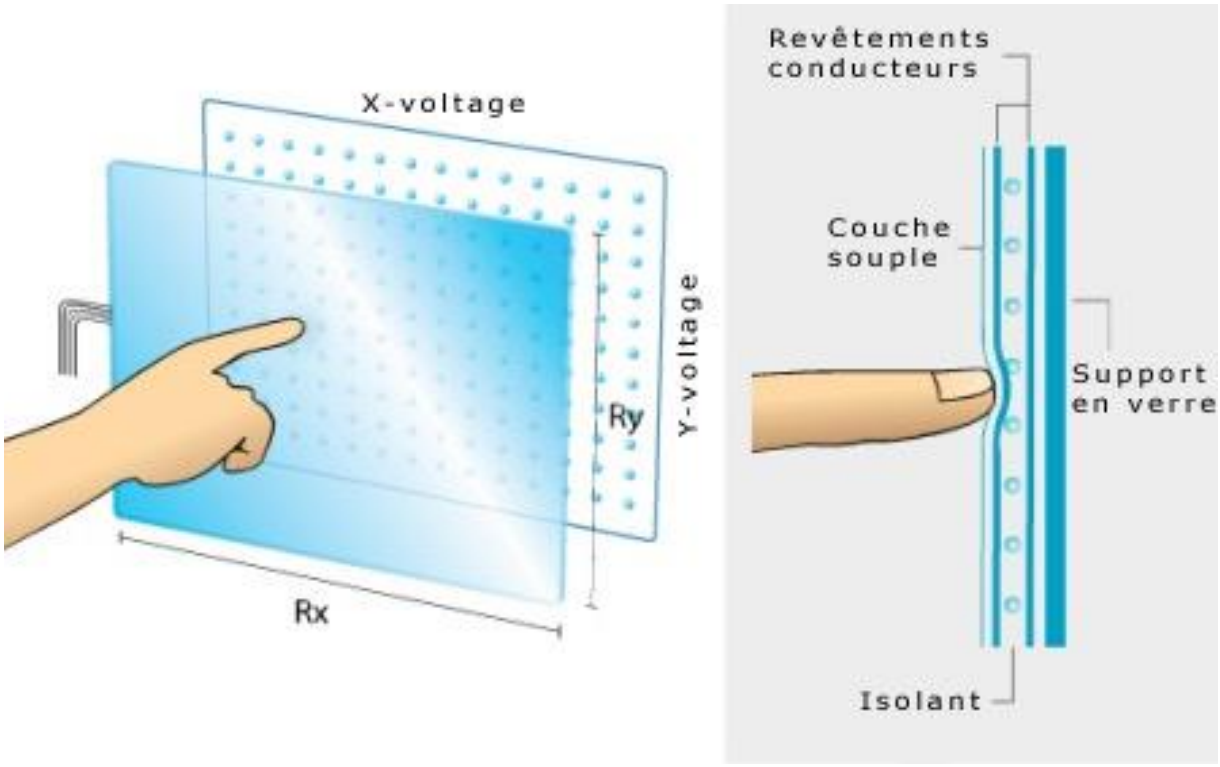
Pour les besoins du projet nous allons utiliser des écrans tactiles, sur les bornes de réservations comme sur le distributeur de plateaux.

Il existe deux grandes familles d'écran tactile, les écrans capacitifs et les écrans résistif

- Les écrans résistifs :

Sont plus économique, plus simple et supporte n'importe quel type de pointeur.

Mais ont une durée de vie plus courte, sont plus sensible aux griffures, coupures, etc., mais sont également pas multi-touches, moins réactivité et lumineux.

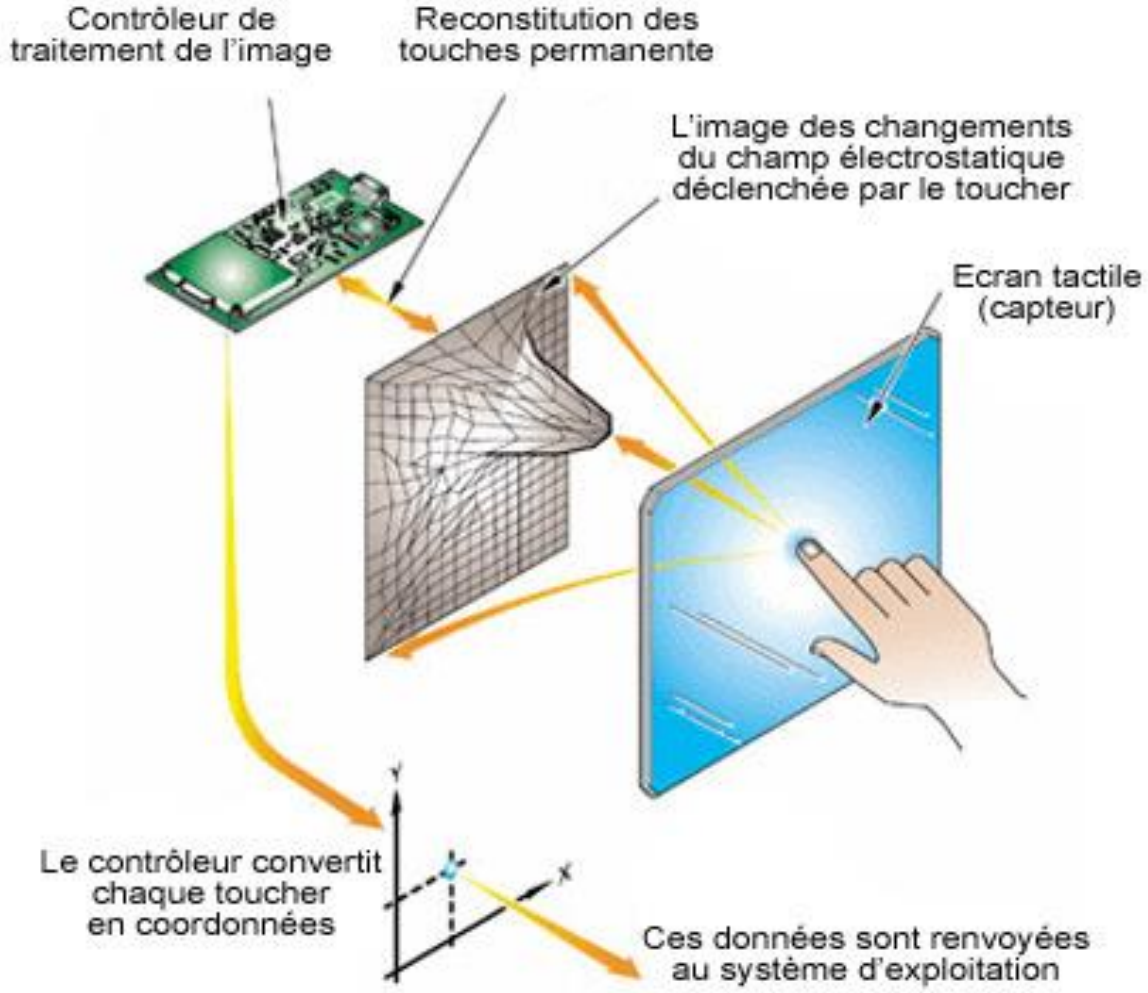


Cette technologie utilise deux couches de revêtements conducteurs souples séparés par un isolant le tout collé sur une plaque en verre. Quand une pression est exercé par un doigt ou autre, une liaison électrique ce fait entre les deux conducteurs ce qui permet de récupérer les coordonnées de la pression sur l'écran.

- Les écrans capacitifs :

Sont compatible avec le multi-touches, plus réactivité, plus lumineux, plus résistance et plus précision.

Mais sont plus cher à fabriquer, non compatible avec les stylets, très sensible à l'eau et à l'humidité.



Le principe de fonctionnement de l'écran contient une couche dite « capacitive », qui porte une certaine charge électrique. Lorsqu'un objet conducteur touche l'écran, il lui vole une partie de cette charge, ce qui est immédiatement détecté par des capteurs situés à chaque coin de l'écran.

Contrairement aux écrans tactiles résistifs, il n'est pas nécessaire d'appuyer très fort sur les écrans tactiles capacitifs... mais il faut absolument les toucher avec un objet qui conduit bien le courant, comme un doigt ou un morceau de métal.

Mise en place de la base de données.

Pour les besoins du projet une base de données devait être mise en place, cette base de données a été créée par Randy (l'étudiant 1) cette base de données va être hébergé sur mon ordinateur portable afin de faciliter la mobilité.

J'ai donc pour cela installé WampServer sur mon ordinateur afin de partager la base de données en local. Après avoir ouvert les ports et désactiver le part feu j'ai également fixé l'adresse IP (dans notre cas 192.168.0.243) de mon ordinateur pour le transformer en un serveur de base de données accessibles par tous.

J'ai ensuite activé le service PphMyAdmin pour héberger la base de données. Les autres utilisateurs n'ont plus qu'à se connecter directement à l'interface PphMyAdmin à l'aide d'un navigateur ou alors depuis leurs programmes à l'aide de l'adresse IP, du nom de la BDD, du login et du mot de passe.

Module logiciel d'accès à la base de données.

Pour que mes différentes applications puissent avoir accès aux données stocker dans la base, j'ai décidé de commencer par développer le module d'accès en c++.

Pour permettre à QT d'utilisé des méthodes SQL il est nécessaire de lui ajouter un module, il faut donc crée la ligne « QT += sql » dans le fichier qmake .pro. Il faut ensuite inclure les définitions des classes du module à l'aide de « #include <QtSql> » dans le fichier d'entête.

J'ai ensuite crée une classe « GestionBdd » qui est actuellement composé d'un constructeur et d'un destructeur. Par la suite il faudra simplement appeler le constructeur au lancement de l'application afin de se connecter à la BDD puis de crée et appelé la méthode souhaitée.

Le constructeur de GestionBdd prend en argument de quatre QString (voir annexe n° 3.1) :

- L'host qui va correspondre à l'adresse IP de la base de données
- Le nom de la base de donnée à la qu'elle se connecter
- Le login pour se connecter à la base de données
- Le mot de passe d'accès à la base de données

Programme pour s'authentifier en cas d'oublie de la carte.

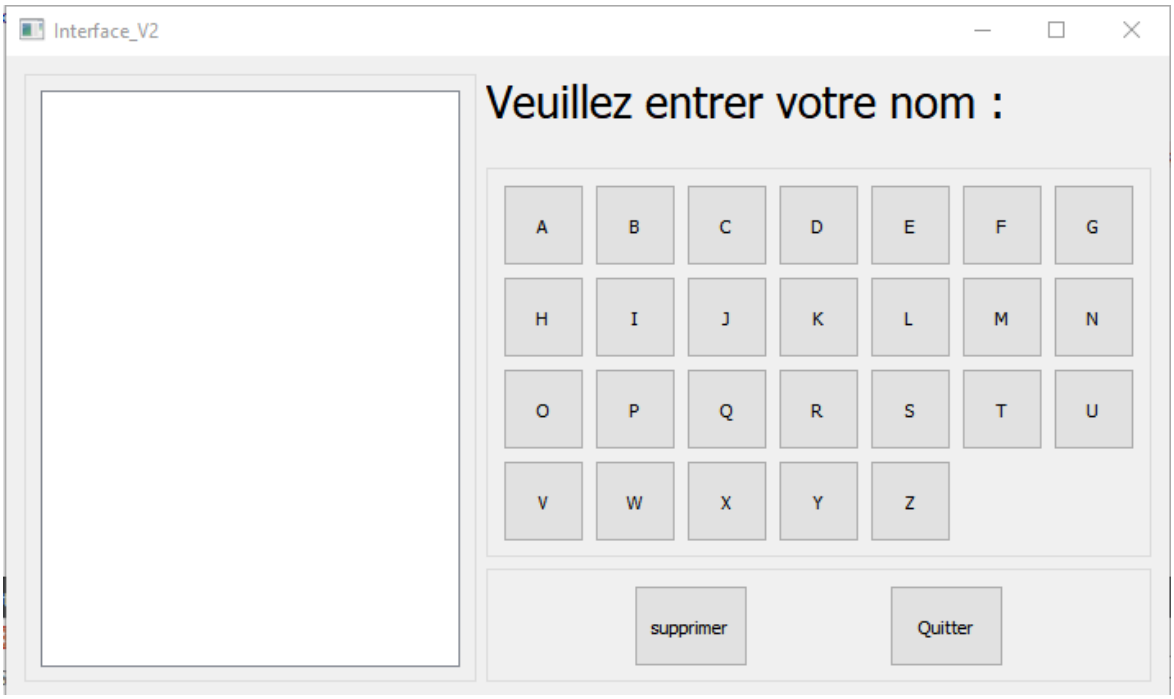
Généralités :

Le système de restauration authentifie les convives grâce à une carte munie d'un code barre, que ce soit au niveau de la borne de réservation ou du distributeur de plateau. Mais en cas d'oubli de la carte les utilisateurs doivent quand même pouvoir réserver ou récupérer sont repas. Pour ce faire nous avons mis en place une solution d'authentification via le nom du convive et un mot de passe à quatre chiffres qui lui est fourni en début d'année scolaire par le service intendance.

Il fallait donc développer une interface homme-machine (IHM) qui permettait ce type de connexion tout en respectant les contraintes du projet (Cad. L'écran tactile, l'ergonomie).

Pour ce faire j'ai donc commencé un nouveau projet à l'aide de l'IDE Qt, le projet sera dans son intégralité développer en C++.

J'ai commencé par réaliser une fenêtre principale (voir fichier H en annexe °3.2) qui sera organisé avec trois QGroupeBox principaux, le premier qui va servir à affiché le clavier visuel pour saisir le nom de famille, le second qui va contenir deux boutons, un pour supprimer le dernier caractère saisi et un pour quitter la fenêtre de recherche, et enfin un dernier qui sert à afficher le tableau de type QTableWidget pour lister les noms récupérer dans la BDD.



Clavier visuel :

Le premier groupeBox (partie_clavier) contenant le clavier visuel est une obligation du au contrainte du projet, pour crée ce dernier j'ai commencé par déclarer mon bouton comme un tableau de 26 caractères dans mon fichier d'entête.

```
private:
    Gestion_bdd     *bdd;
    QTableWidget    *la_liste = new QTableWidget(this);
    QPushButton     *bt_lettres[26];
    QPushButton     *bt_supprimer;
    QPushButton     *bt_quitter = new QPushButton;
    QString          nom_recherche;
    QLabel          *instruction = new QLabel;
```

Les boutons :

J'ai ensuite crée plusieurs boucles qui permette dans un premier temps d'alloué dynamiquement mon « bt_lettres », de lui attribué une lettre de l'alphabet via sa valeur décimale du tableau ASCII, la taille de se bouton et enfin le « connect » afin de pouvoir récupérer le clic sur le bouton.

```
/***************** Boucle de création des boutons du clavier ******************/

for(int i=0; i<26; i++)
{
    QString text = QString::number(i);
    bt_lettres[i] = new QPushButton(text, this); // Bouton qui servira pour l'alphabet

    bt_lettres[i]->setText(QChar(65+i));
    bt_lettres[i]->setFixedSize(50,50);

    signalMapper->setMapping(bt_lettres[i],i);

    connect(bt_lettres[i], SIGNAL(clicked()), signalMapper, SLOT(map()));
}
```

Dans une seconde boucle, je m'occupe du placement de ces boutons dans un QGridLayout pour des raisons d'ergonomie j'ai décide de placer mes boutons sur quatre lignes et sept colonnes.

```
/****** Placement des boutons dans le grid en fonction de leur contenu (alphabet) ******/

for(int i=0; i<4; i++) //boucle pour le placement en quatre lignes
{
    for(int j=0; j<7; j++) //boucle pour le placement en sept colonnes
    {
        if(j+(i*7)<26)
        {
            grilleLettre->addWidget(bt_lettres[j+(i*7)],i,j);
        }
    }
}
```

Une fois tout cela fait on obtient le clavier visuel suivant :

Affichage de la recherche

Pour la suite je récupère les lettres tapées par l'utilisateur et je récupère les noms correspondant dans la base de données à l'aide de cette requête : SELECT nom, prénom FROM convive WHERE nom LIKE '"+nom+"%'. L'élément % situé âpres le nom permet de récupérer tout les noms commençant par le ou les caractères rentrer par l'utilisateur.

Les noms récupérer par la requête sont stocker dans la structure « st_nom_premon » :

Cette structure contient deux tableaux de type QString.

```
struct st_nom_prenom
{

    QString nom[50];
    QString prenom[50];

};
```

Une fois les informations stocker dans la structure par la requête, j'appelle la méthode « ajouter_personne_tableau » pour inséré le nom des convives récupérer dans mon QTableWidget à l'aide de la méthode setItem et QTableWidgetItem en indiquant l'emplacement dans le QTableWidget (ligne et colonne, ici ligne b et colonne 0).

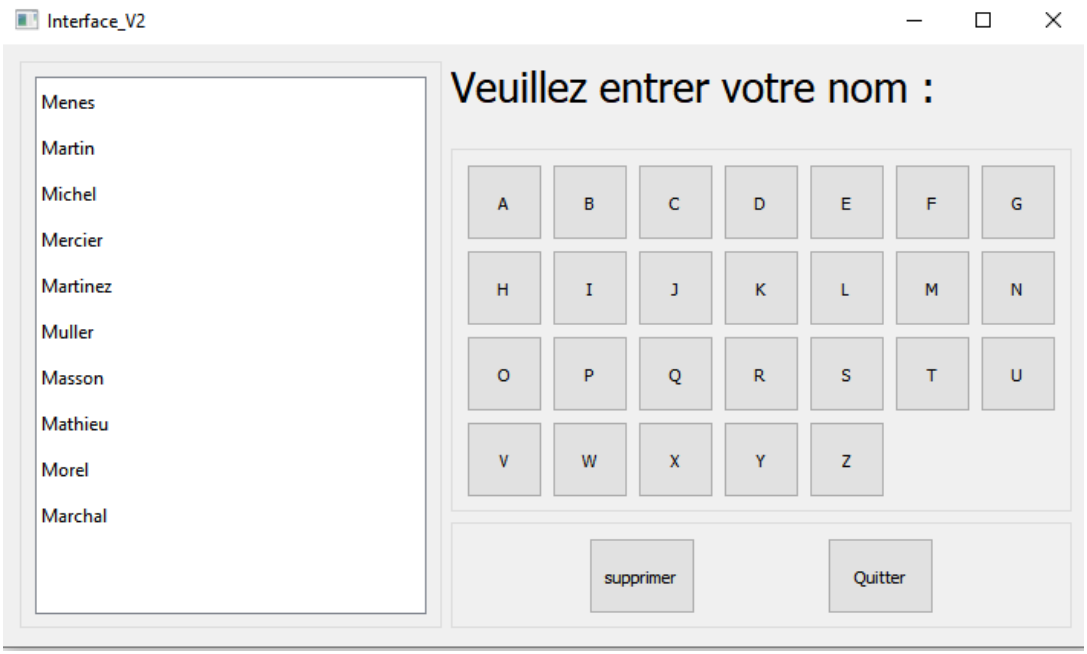
Définitions de mon QTableWidget

```
QHeaderView* header = la_liste->horizontalHeader();
header->setSectionResizeMode(QHeaderView::Stretch);

la_liste->setRowCount(10);
la_liste->setColumnCount(1);
la_liste->setShowGrid(false);
la_liste->verticalHeader()->setVisible(false);
la_liste->horizontalHeader()->setVisible(false);
la_liste->setEditTriggers(QAbstractItemView::NoEditTriggers);
la_liste->setSelectionMode(QAbstractItemView::SingleSelection);
la_liste->setStyleSheet("QTableView {selection-background-color: blue;}");
```

```
void Fenetre_principal::ajouter_personne_tableau()
{
    for(int b=0; b<z; b++)
    {
        la_liste->setItem(b, 0, new QTableWidgetItem(obj.nom[b])); // Ajout des noms dans le QTableWidget
    }
}
```

Méthode permettant d'ajouter les noms dans le tableau.

En fonction des caractères rentrés par l'utilisateur on obtient donc un tableau remplis par les noms stocker dans la BDD (ici l'utilisateur à appuyer sur M) et qui va affiné instantanément les propositions au fur et à mesure que l'utilisateur entrera de caractères.



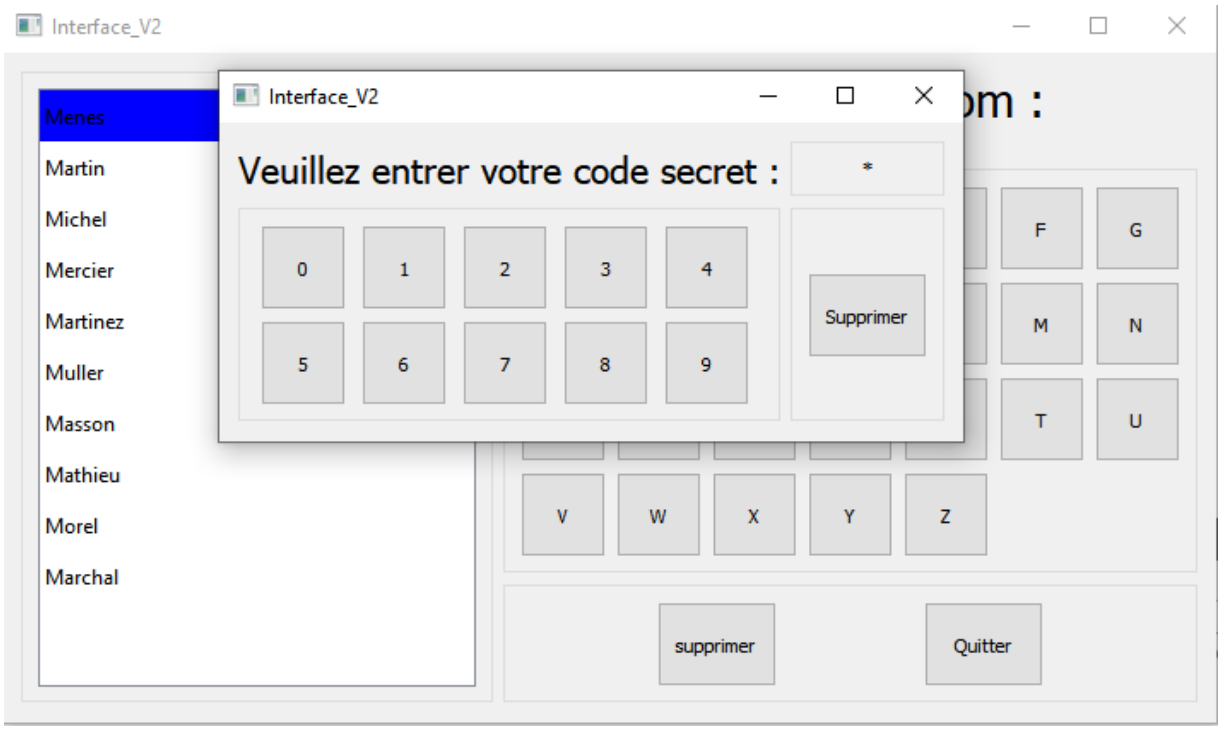
Sélection du nom par l'utilisateur :

Pour récupérer le nom sélectionné par l'utilisateur dans le tableau j'utilise un connect sur mon QTableWidget « la_liste » en récupérant le signal de « cellClicked » ce SIGNAL est connecté à ma méthode « cellule » qui appelle une nouvelle fenêtre à qui je passe l'indice de ligne selectionner (pour ensuite récupérer le nom dans la structure dans la nouvelle fenêtre) ainsi que mon objet de structure « obj ».

```
void Fenetre_principal::cellule(int ligne, int colonne)
{
    fenetre = new Fenetre_secondaire(ligne, obj);
    fenetre->show();
}
```

Deuxième fenêtre :

Cette deuxième fenêtre va permettre à l'utilisateur de saisir sont mot de passe, pour se faire la fenêtre récupère le nom sélectionner dans le QTableView et va comparer le mot de passe (code à quatre chiffres) saisie par l'utilisateur à celui présent dans la BDD.



Cette fenêtre nommée « fenetrecode.cpp » est composé composer comme la premiere de trois QGroupBox qui servent à la mise en forme de l'affichage (fichier d'en-tete en annexe n° 3.3).

Le premier GroupBox va contenir le pavé numérique réalisé de la même façon que le clavier visuel proposé dans la première page toujours dans l'optique de répondre au cahier des charges et de fournir une application totalement utilisable sur un écran tactile sans accès à un clavier ou une souris.

Le deuxième GroupBox contient simplement le bouton « supprimer » qui efface le dernier caractère saisie.

Enfin de troisième et dernier GroupBox permet d'afficher le code secret saisie par l'utilisateur, pour des raisons de sécurité ce dernier devait ne pas être visible par tous, j'ai donc décidé de remplacer les caractères saisie par des étoiles (*). L'affichage du code secret se fait par un QLabel, à défaut d'avoir une méthode qui « crypte » directement les caractères saisie je passe par une variable tampon appeler « compter_mdp » de type int qui me sert à compter le nombre de caractères saisie et a en fonction afficher le bon nombre d'étoiles avec un switch case.

```
code_secret = code_secret + bt_chiffres[pos]->text();
if(compteur_mdp<4)
{
    compteur_mdp ++;
}

switch (compteur_mdp) {
case 1:
    affichage_code->setText("*");
    break;
case 2:
    affichage_code->setText("**");
    break;
case 3:
    affichage_code->setText("***");
    break;
case 4:
    affichage_code->setText("****");
    break;
default:
    break;
}
```

Ici « code_secret » est de type QString et sert à stocker réellement le mot de passe saisie alors que « compteur_mdp » sert uniquement à l'affichage des étoiles dans le QLabel.

bt_chiffres[pos]->text() permet de récupérer la valeur contenu dans les QPushButton du pavé numérique.

Avant de passer à la vérification du code je vais simplement vous montrer l'action effectué par le bouton de suppression qui est similaire au précédent.

Ici je reprends le même principe que vu précédemment à la différence que quand l'utilisateur appuie sur « supprimer » je retire le dernier caractère contenu dans le QString « code_secret » et décrémente d'un le « compteur_mdp ».

```
void FenetreCode::signal_button_supp()
{
    if(code_secret.size()!=0)
    {
        code_secret.remove(code_secret.size()-1, 1);
        if(compteur_mdp >0 && compteur_mdp<5){
            compteur_mdp --;
        }
        switch (compteur_mdp) {
        case 1:
            affichage_code->setText("*");
            break;
        case 2:
            affichage_code->setText("**");
            break;
        case 3:
            affichage_code->setText("***");
            break;
        case 4:
            affichage_code->setText("****");
            break;
        default:
            affichage_code->setText("");
            break;
        }

    }
}
```

Vérification du mot de passe convive :

Pour éviter de devoir ajouter un bouton de validation qui surchargerai l'interface à mon sens j'ai décidé de vérifier le mot de passe dès que ça taille serai égal à quatre sachant tous les mots de passe sont des codes à quatre chiffres. Pour ce faire j'ai créé une nouvelle méthode dans Gestionbdd appeler « verifier_code » de type booléen, cette méthode prend en argument le code saisie par l'utilisateur ainsi que son nom récupérer dans le QTableWidget vu plus tôt et retourne vrai ou faux en fonction de si le mot de passe est correct ou non.

Un requête SQL doit donc être crée, elle sera de type SELECT avec comme paramètre le nom du convive, cette requête va donc récupérer le mot de passe contenu dans la BDD pour ensuite pouvoir le comparer à celui saisie précédemment.

```
requete = "SELECT code_secret FROM convive WHERE nom='"+nom+"' ";
```

Pour exécuter cette requête dans la base de données j'utilise « query.exec » avec query préalablement déclaré comme type QSqlQuery. Je vérifier ensuite si la requête ne retourne pas une erreur et ajoute le code retourné par la requête dans la variable code_recup de type QString.

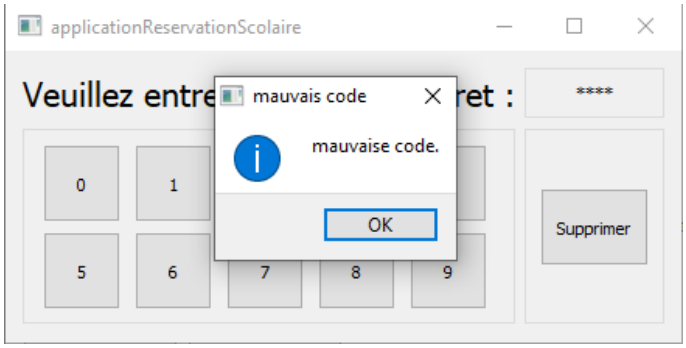
Pour finir je compare ma variable « code_recup » avec ma variable « code » et retourne true ou false en fonction de si le code est le même ou non.

```
query.exec(requete);
if(query.lastError().text()==" ")
{
    while(query.next()) // Récupérer les résultats
    {
        code_recup= query.value(0).toString();
    }
}
else {
}


if (code_recup == code)
{
    return 1;
}else {
    return 0;
}
```

Ce return est stocké dans la variable verif_mdp de type Booléen.

A la suite de quoi si le code saisi est bon j'affiche un message indiquant que le code secret est valide et passe à l'étape de vérification de la réservation du repas pour le jour en cours (étape détaillé plus tard) et si le code est faux un message d'erreur s'affiche indiquant que le code est faux.

```
verif_mdp=bdd.verifier_code(code_secret, nom_rechercher);

if(verif_mdp==true)
{
    QMessageBox::information(
                this,
                tr("Bon code"),
                tr("Bon code.") );
    close();
}
```



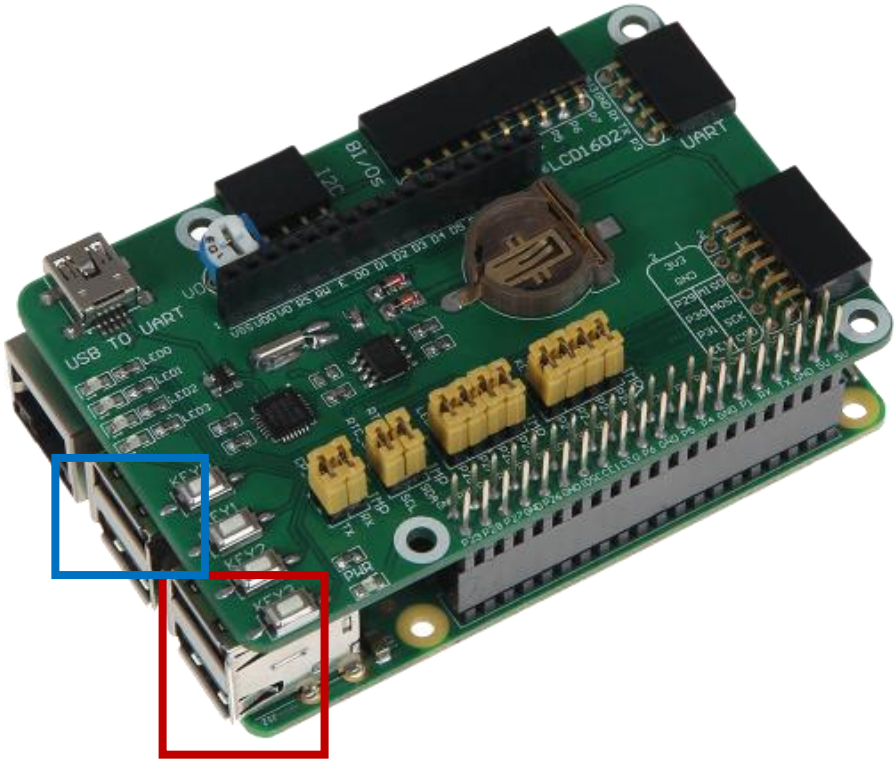
Conclusion :

Ce programme pour s'authentifier sans carte va être utilisé dans le distributeur de plateau au moment de la distribution des repas mais également par l'étudiant un dans les bornes de réservation. Malgré ça simplicité apparente il m'a fallu un certain temps pour réussir le placement des différents boutons (surtout le clavier visuel) mais également pour comprendre le fonctionnement du QTableWidget que ce soit pour l'ajout des items (ici les noms récupéré dans la base de donnée en « direct ») ou pour ensuite récupérer la case sélectionnée par l'utilisateur et son contenu.

Distributeur de plateau

En tant qu'étudiant trois une de mes taches principales est la gestion du distributeur de plateau, faute de matériel pour réaliser cette tâche, le distributeur va être simulé par un raspberry Pi 3 équipé d'un module EXP-500. Cette tâche n'est à ce jour pas terminé car des problèmes persiste avec ledit module, il m'est pour le moment pas possible de récupérer ou de communiqué des informations au module.

L'idée étant d'utilisé les boutons poussoir présent à la gauche du module (dans le cadre rouge) ainsi que les led (dans le cadre bleu) pour simuler une distribution de plateaux.



J'ai néanmoins commencé ma tâche en crée la requête SQL qui va permettre de vérifier si le convive authentifié précédemment a bien réservé ou non.

Vérification de la réservation :

La méthode pour vérifier la réservation est la même peu importe que l'utilisateur se soit authentifié avec sa carte ou avec son nom et son mot de passe. Je crée une requête SQL de type SELECT qui va récupérer l'ID de la réservation du convive en fonction de son ID personnel et de la date du jour.

```
QString requete ="SELECT id FROM reservation "
                 "WHERE reservation.convive_id= '"+badge.convive_id+"' "
                 "AND reservation.dateDesRepas= '"+date_du_jour+"'";
```

Cette requête est ensuite exécutée dans la méthode « verifier_reservation » de GestionBdd, je récupère ainsi sois un champ comportant l'id, ce qui veut dire que le convive à réservé soit rien ce qui veut dire qu'il n'a pas réservé. Je retourne ensuite en fonction un ou zéro et affiche si la réservation est effective ou pas.

```
bool f = bdd.verifier_reservation(requete);

if(f ==true)
{
    qDebug() <<"f egale a true donc repas reserver";
}
if(f ==false)
{
    qDebug() <<"f egale a false donc pas de repas reserver";
}
```

Comme expliqué plus tôt, à l'heure actuelle le retour ce fait uniquement grâce des qDebug.

Comme vous avez pu le voir pour effectuer la requête j'ai utilisé la date du jour comme point de repère, pour récupérer cette date j'inclue QTime et crée une structure *t et lui affecte la date du système.

```
time_t timestamp;
struct tm * t;
timestamp = time(NULL);
t = localtime(&timestamp);
```

Je déclare ensuite une variable date_du_jour de type QString et ajuste la date avec année, mois, jour.

```
QString date_du_jour;
date_du_jour = QString::number(t->tm_year+1900)+"-"+QString::number(t->tm_mon+1)+"-"+QString::number(t->tm_mday)
```

Problèmes rencontrés :

Lors du développement de ce projet j'ai dû faire face à plusieurs problèmes :

- La mise en réseau de la base de données, avec l'ouverture des ports pour que mes camarades puissent y accéder.
- La mise en place des boutons pour le clavier virtuel.
- L'affichage « en direct » des noms chercher par le convive dans un tableau.
- La simulation du distributeur de plateaux.

Conclusion :

La progression de ma partie est bien avancée, il me reste à m'occuper du « distributeur de plateaux » bien que cette partie semble compromise à l'heure actuel dù au défaut visiblement présent sur le module fourni.

ANNEXES :

1.0

Historique :

Éditer Copier Supprimer	1	reservation	2019-04-02	09:14:29	1	2005	0
Éditer Copier Supprimer	2	reservation	2019-04-02	09:42:10	1	2005	0
Éditer Copier Supprimer	3	reservation	2019-04-02	09:54:54	1	2005	0
Éditer Copier Supprimer	4	reservation	2019-04-02	10:32:28	1	2005	0
Éditer Copier Supprimer	5	reservation	2019-04-02	10:33:35	1	2005	0
Éditer Copier Supprimer	6	reservation	2019-04-02	10:34:29	1	2005	0
Éditer Copier Supprimer	7	reservation	2019-04-02	11:34:07	1	2005	0

Badge :

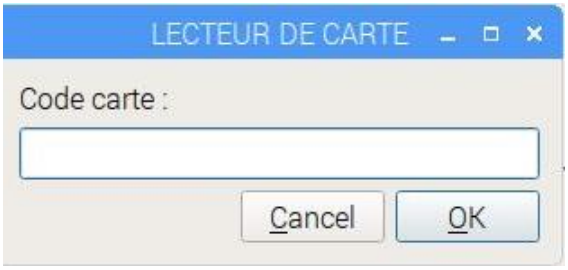
	id	statut	convive_id
Éditer Copier Supprimer	00001053521	1	2053
Éditer Copier Supprimer	021129206195	1	2005
Éditer Copier Supprimer	021129206468	1	2021
Éditer Copier Supprimer	9782851807151	1	2001

Convive :

	id	nom	prenom	classe_emploie	solde	tarif	matricul	code_secret	statut (1 = employé / 0 = eleve)
Éditer Copier Supprimer	2001	BOURBON	Dandy	SN3	2500	2	0	8514	0
Éditer Copier Supprimer	2005	BOURDON	Randy	SN3	24	3	0	7777	0
Éditer Copier Supprimer	2021	NEOU	Benoit	Prof	39544	6	0	6666	1
Éditer Copier Supprimer	2040	DOURBON	Bandy	SN3	4000	3	0	8516	0
Éditer Copier Supprimer	2042	ROURDON	Banby	SN3	4	3.5	0	9823	0
Éditer Copier Supprimer	2044	DOURDON	Hanry	SN3	400000	2.5	0	4846	0
Éditer Copier Supprimer	2045	Potter	harry	STI2D	100	2	0	1234	0

1.1

Le pop-up :



Code :

```
QInputDialog *Dialog = new QInputDialog;
QString text= Dialog->getText(this, tr("LECTEUR DE CARTE"),tr("Code carte:") ,QLineEdit
::NoEcho,QDir::home().dirName(), &ok);
```

1.2

Code :

```
Badge GestionBdd::verifCarte(QString numCarte)
{
   Badge badge;
   QSqlQuery query;
   QString requete;
   badge.validite =false;
   requete = "SELECT DISTINCT * FROM badge WHERE badge.id ='"+numCarte+"'";
   if(dbOk == true)
   {
      query.exec(requete);
      if(query.lastError().text()==" ")
      {
         while(query.next()) // Récupérer les résultats
         {
            qDebug()<<"OK";
            badge.numero = query.value(0).toString();
            badge.validite = query.value(1).toBool();
            badge.convive_id = query.value(2).toString();
            return   badge;
         }

      }
```

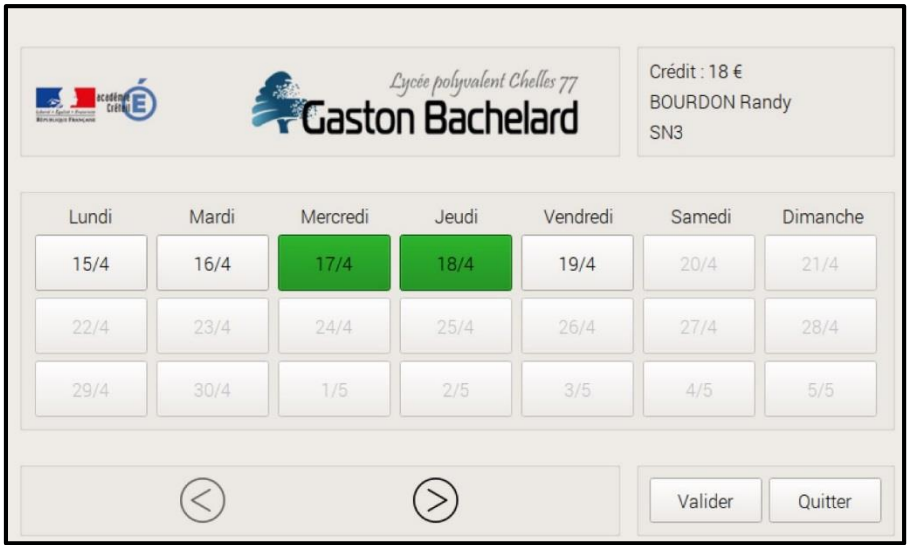
```
        return   badge;
    }
    return badge;
}
```

1.3

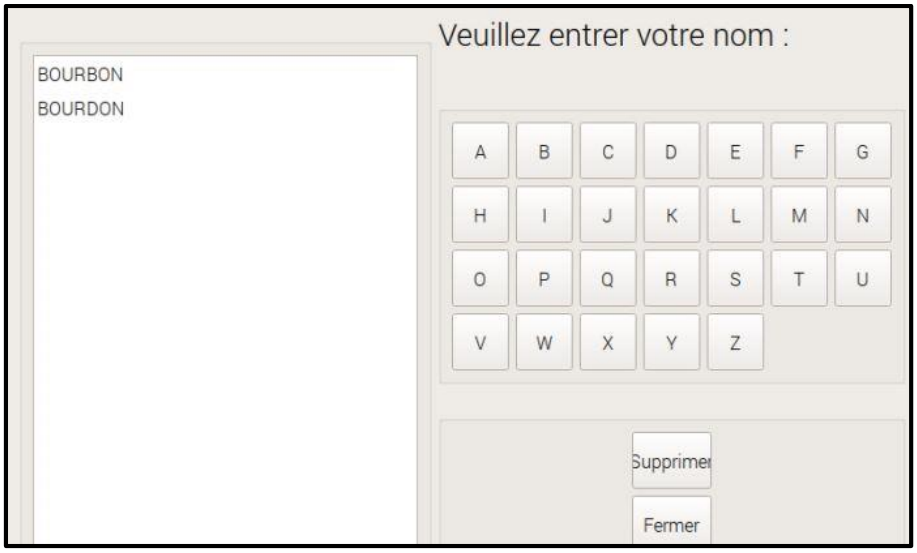
Code :

```
if(badge.validite == 1)
      {
         FenetreReservation *fenetre = new FenetreReservation(badge);
         fenetre->resize(500,300);
         fenetre->show();
      }
```

1.4



1.5



1.6

```
QGroupBox *FenetreReservation::boutonChangement()
{ QGroupBox *groupBox = new QGroupBox();
   groupBox->setFlat(false);
   Gauche = new QPushButton();
   Gauche->setFixedSize(QSize(100,40));
   Droit = new QPushButton();
   Droit->setFixedSize(QSize(100,40));
   Droit->setFlat(true);
   Gauche->setFlat(true);
   QHBoxLayout *vbox = new QHBoxLayout;
   vbox->addWidget(Gauche);
   vbox->addWidget(Droit);
   groupBox->setLayout(vbox);
   return groupBox;
}
QGroupBox *FenetreReservation::labelSuperieurC()
{
   QGroupBox *groupBox = new QGroupBox();
   groupBox->setFlat(false);
   QPixmap p; // load pixmap
   QImage image("I:\applicationReservationScolaireFINALE\marianne.png");
   imageLogo = new QLabel();
   imageLogo->setPixmap(QPixmap::fromImage(image));
   imageLogo->show();
   QVBoxLayout *vbox = new QVBoxLayout;
   vbox->addWidget(imageLogo);
```

```
    groupBox->setLayout(vbox);
    return groupBox;
}
QGroupBox *FenetreReservation::labelSuperieurD()
{
    QGroupBox *groupBox = new QGroupBox();
    groupBox->setFlat(false);
    solde = new QLabel ("solde");
    nom = new QLabel ("nom");
    classe = new QLabel ("classe");
    QVBoxLayout *vbox = new QVBoxLayout;
    vbox->addWidget(solde);
    vbox->addWidget(nom);
    vbox->addWidget(classe);
    groupBox->setLayout(vbox);
    return groupBox;
}



QGroupBox *FenetreReservation::boutonInferieur()
{
    QGroupBox *groupBox = new QGroupBox();
    groupBox->setFlat(false);
    Quitter = new QPushButton("Quitter");
    Quitter->setFixedSize(QSize(100,40));
    Valider = new QPushButton("Valider");
    Valider->setFixedSize(QSize(100,40));
    QHBoxLayout *vbox = new QHBoxLayout;
    vbox->setAlignment(Qt::AlignLeft);
    vbox->addWidget(Valider);
    vbox->addWidget(Quitter);
    groupBox->setLayout(vbox);
    return groupBox;
}
```

1.7

```
int tampondernierjour= t->tm_mday;
int tamponpremierjour= t->tm_mday;
int tamponpremiermois = t->tm_mon+1;
int tamponderniermois = t->tm_mon+1;
int tamponannee = t->tm_year+1900;
```

1.8

```
for(int i=jour; i<21; i++)
  {

    QString text = QString::number(i);
    calendrierButon[i] = new QPushButton(text, this);
    calendrierButon[i]->setFixedSize(QSize(100,50));

    signalMapper->setMapping(calendrierButon[i], i);
    connect(calendrierButon[i], SIGNAL(clicked()), signalMapper, SLOT(map()));
    calendrierButon[i]->setText
(QString::number(tamponderniерjour)+"/"+QString::number(tamponderniermois));
    tamponderniерjour++;
    switch (tamponderniermois)
    {
    case fevrier:
      if ((((tamponannee %4 == 0) && (tamponannee %100 != 0)) || (tamponannee %400
== 0))) // Test de l'année bissextile
      {
        if(tamponderniерjour > 29 )
        {
          tamponderniermois++;
          tamponderniерjour = 1;
        }
      }
      else
      {
        if(tamponderniерjour > 28)
        {
          tamponderniermois++;
          tamponderniерjour = 1;
        }
      }
      break;
    case avril:
    case juin:
    case septembre:
    case novembre:
      if(tamponderniерjour >30)// Test des mois avec 30 jours
      {
        tamponderniermois++;
        tamponderniерjour = 1;
      }
      break;
    case janvier:
    case mars:
```

```
    case mai:
    case juillet:
    case aout:
    case octobre:
    case decembre:
    {

      if(tampondernierjour >31) // Test des mois avec 31 jours
      {
        tamponderniermois++;
        tampondernierjour = 1;
      }
      if(tampondernierjour ==12 && tampondernierjour >31 )
      {
        tamponderniermois=1;
        tampondernierjour = 1;
        tamponannee = tamponannee +1;
      }
    }
      break;
    default:
      break;
    }
  }
```

1.9

```
 for(int i= 0 ;i<21;i++)
   {
     for(m=0;m<nombreDeFeries;m++)
     {
       debut = tabDatesFeriesRecupDebut[m];

       strcpy(texte9,debut.toStdString().c_str());
       sscanf(texte9,"%d-%d-%d",&annee9,&mois9,&jour9);//Extraire le jour, le mois et
l'année
       dateDebut =QString::number(jour9)+"/"+QString::number(mois9);
       fin = tabDatesFeriesRecupFin[m];
       strcpy(texte9,fin.toStdString().c_str());
       sscanf(texte9,"%d-%d-%d",&annee9,&mois9,&jour9);//Extraire le jour, le mois et
l'année
```

```
        dateFin =QString::number(jour9)+"/"+QString::number(mois9);
        qDebug()<<"DateDebut "<<dateDebut<<" Datefin "<<dateFin;
        if(dateDebut == calendrierButon[i]->text())
            {
            tamponFerieD =true;
             }
        if(dateFin == calendrierButon[i]->text())
           {
            tamponFerieD =false;
             }
        if(tamponFerieD ==  true)
           {
            calendrierButon[i]->setEnabled(false);
             }

    }
  }
```

1.10

```
void FenetreReservation::choixJour(int pos)
{
   QMessageBox msgbox;
   calendrierButon[pos]->setStyleSheet("background-color: green;");
   char    texte[10];            //Pour ranger la date sous forme de chaîne de caractères
   int     jour;
   int     mois;
   int annee;

   QString autre = calendrierButon[pos]->text();
   //Ranger la date sous la forme d'une chaîne de caractères
   strcpy(texte,autre.toStdString().c_str());
   sscanf(texte,"%d/%d",&jour,&mois);//Extraire le jour, le mois et l'année
   annee = anneetampon;
   qDebug()<<mois;
   QString jour1,mois1;

   jour1 = QString::number(jour);
   mois1 = QString::number(mois);



 //Ajout d'un 0 si mois ou jour inferrieur a 10
   if(9>=mois)
```

```
    {
        mois1 ="0"+mois1+"";
    }
    if(9>=jour)
    {
        jour1 ="0"+jour1;
    }

    QString date = QString::number(annee)+"-"+mois1+"-"+jour1;
    qDebug()<<date;
    int i=0;
    calendrierButon[pos]->setStyleSheet("background-color: green;");
    tabJourReserv[nombreDeJour] = date.toStdString().c_str();
    bool vrai=false;


    for(i=0;i<nombreDeJour;i++)
    {

        if(date== tabJourReserv[i])
        {
            tabJourReserv[i]="";
            tabJourReserv[nombreDeJour]="";
            calendrierButon[pos]->setStyleSheet(0);
            soldeAffichage = soldeAffichage +personne.tarif.toFloat();
            solde->setText("Crédit : "+ QString::number(soldeAffichage) + " €");
            vrai=true;
        }


    }
    if(vrai == false)
    {
        soldeAffichage = soldeAffichage -personne.tarif.toFloat();
        solde->setText("Crédit : "+ QString::number(soldeAffichage) + " €");

    }
    nombreDeJour= nombreDeJour+1;

}
```

1.11

```
void FenetreReservation::Enregistrement()
{
    QMessageBox msgBox;
    msgBox.setWindowTitle(tr("ENREGISTREMENT"));
```

```
msgBox.setText("Voulez-vous réserver les jours selectionés ?");
QAbstractButton* pButtonYes = msgBox.addButton(tr("Oui"), QMessageBox::YesRole);
msgBox.addButton(tr("Non"), QMessageBox::NoRole);
msgBox.exec();
if (msgBox.clickedButton()==pButtonYes)
{
   float tarif =personne.tarif.toFloat() ;
   float soldeFloat = personne.solde.toFloat();
   int nbDeJourSauvegarder=0;
   float soldeVerif=0;

   for(int i=0;i<=nombreDeJour;i++)
   {
      if(tabJourReserv[i]!="")
      {
         nbDeJourSauvegarder=nbDeJourSauvegarder+1;
      }
   }
   if(nbJourDejaSauv>nbDeJourSauvegarder) //ajouter
   {
      int differnce = nbJourDejaSauv-nbDeJourSauvegarder;
      soldeVerif = differnce * tarif;
      soldeVerif =  soldeFloat + soldeVerif;
   }
   if(nbJourDejaSauv<nbDeJourSauvegarder) //retirer argent
   {
      int differnce = nbDeJourSauvegarder-nbJourDejaSauv;
      soldeVerif = differnce * tarif;
      soldeVerif =  soldeFloat - soldeVerif;
   }
   if(nbJourDejaSauv==nbDeJourSauvegarder) //ne rien faire
   {
      soldeVerif = soldeFloat;
   }

   if(0<=soldeVerif)//solde suffisant
   {
      nbJourDejaSauv =0;
      personne.solde = QString::number(soldeVerif);
      bdd.modifSolde(soldeVerif,general);
      bdd.reservation(tabJourReserv,nombreDeJour,general);
      solde->setText(personne.solde + " €"); //pour afficher l'argent qui reste
      QMessageBox::information(this,tr("VALIDATION"),tr("Vous avez bien reserver"));
      close();
   }
   else//solde insuffisant
   {
```

```
        QMessageBox::information(this,tr("ERREUR"),tr("solde insuffisant, veuillez supprimer
un ou plusieurs jours "));
      }
   }
}
```

1.12

```
QString GestionBdd::insererHistorique(Badge badge)
{
   time_t timestamp;
   struct tm * t;
   timestamp = time(NULL);
   t = localtime(&timestamp);
   QSqlQuery query;
   QString requete;
   QString reserv = "reservation";
   // VALUES ("2019-02-04","09:14:29",1,021129206195,0);
   QString heure =QString::number( t->tm_hour) + ':' + QString::number(t->tm_min) + ':' +
QString::number( t->tm_sec) ;
   QString jour = QString::number(t->tm_year+1900) + '-' + QString::number(t->tm_mon+1) +
'-' + QString::number(t->tm_mday) ;

   requete = "INSERT INTO historique (lieu,jour,heure,autorise,convive_id,retirer) VALUES
('"+ reserv +"',\""+jour+"\",\""+heure+"\",1," + badge.convive_id + ",0)";
   qDebug()<<requete;
   return  executerRequete(requete);
}


/********* Récupère toutes les reservations du client ********************/
   bdd.recupererListeDate(listsDate,general);
   if(!listsDate.empty())
   {
      for(p= listsDate.begin();p<listsDate.end();p++)
      {
         tabJourRecup[nombreJour] = p->dateRecup;
         tabJourReserv[nombreJour] = tabJourRecup[nombreJour];
         nombreJour= nombreJour+1;
      }
   }

   nbJourDejaSauv = nombreJour;
   nombreDeJour=  nbJourDejaSauv;

/******************** Recupere touts les jours féries ********************/
   QList<datesFeries> listsDatesFeries;
   QList<datesFeries> listsJoursFeries;
```

```
    QList<datesFeries>::iterator j;

    bdd.recupererListeJoursFeries(listsJoursFeries);
    if(!listsJoursFeries.empty())
    {
       for(j= listsJoursFeries.begin();j<listsJoursFeries.end();j++) //
       {
          //qDebug() << j->jourFerie;
          tabJoursFeries[nombreDeJourFerie]=j->jourFerie;
          //tabJourRecup[nombreJour] = j->dateRecup;
          qDebug() <<"tabJourRecup[nombreJour]"<< tabJoursFeries[nombreDeJourFerie];
          nombreDeJourFerie = nombreDeJourFerie+1;
       }
    }
    nombreDeJoursFeries=nombreDeJourFerie;
/************* Recupere toutes les periodes fériees ********************/
    bdd.recupererListeDatesFeries(listsDatesFeries);
    if(!listsDatesFeries.empty())
    {
       for(j= listsDatesFeries.begin();j<listsDatesFeries.end();j++)//
       {
          tabDatesFeriesRecupDebut[nombreJour] = j->dateFeriesRecupDebut;
          tabDatesFeriesRecupFin[nombreJour] = j->dateFeriesRecupFin;
          nombreJour = nombreJour+1;
       }
    }
```

ANNEXE N° 2.1

- 2.1 gestion_intendance .h :

```
#ifndef GESTION_INTENDANCE_H
#define GESTION_INTENDANCE_H

#include "QtWidgets"
#include <QSqlDatabase>
#include <QDebug>
#include <QtSql>
#include <QSqlQuery>
#include <QSqlError>
#include <QFile>
#include <QFileDialog>
#include <QPrinter>
#include <QPainter>

class Gestion_Intendance : public QWidget
{
    Q_OBJECT

public:
    Gestion_Intendance(QWidget *parent = 0);
    ~Gestion_Intendance();
private slots:
    void creationPage1();
    void creationPage2();
    void creationPage3();
    void creationPage4();
    void creationPage5();
    void creationPage6();
    void connexionBDD();
    QString executerRequete(QString);
    //Page 1
    void connexion();
    void deconnexion();
    //Page 2
    void insererConvive();
    void supprimerConvive();
    void supprimerPlusieursConvives();
    void insererPlusieursConvives();
    void parcourir();
    //Page 3
    void selectImpress();
    void creerPDF();
    void imprimer();
    void annulerCarte();
    void creerCarte();
    void selectImpress2();
    void creerPDF2();
    void imprimer2();
    void annulerCarte2();
    void creerPlusieursCartes();
    //Page 4
    void connecterConviveSolde();
    void additionnerSolde();
    void soustraireSolde();
```

```
    void validerTarif();
    void validerSoldeTarif();
    //Page 5
    //Page 6

private:
    QWidget *page1 = new QWidget;
    QWidget *page2 = new QWidget;
    QWidget *page3 = new QWidget;
    QWidget *page4 = new QWidget;
    QWidget *page5 = new QWidget;
    QWidget *page6 = new QWidget;
    QSqlDatabase db;

    // ////////////////////////////////////// Page 1
//////////////////////////////////////
    QLabel *txtLogin = new QLabel("Entrez votre Login");
    QLineEdit *editLogin = new QLineEdit;
    QLabel *txtPassword = new QLabel("Entrez votre mot de passe");
    QLineEdit *editPassword = new QLineEdit;
    QPushButton *boutonValider1 = new QPushButton("Valider");
    QPushButton *boutonDeconnexion = new QPushButton("Déconnexion");
    QLabel *imageLogo = new QLabel;
    QLabel *imageRestoSco = new QLabel;
    QLabel *imageRestoSco2 = new QLabel;
    QLabel *txtLoginRecup = new QLabel;

    // ***** Groupe Box *****
    QGroupBox *gBox1 = new QGroupBox;
    QGroupBox *gBox1V2 = new QGroupBox;
    QGroupBox *gBoxTop = new QGroupBox;
    QGroupBox *gBoxBottom = new QGroupBox;
    QGroupBox *gBoxGauche = new QGroupBox;
    QGroupBox *gBoxDroite = new QGroupBox;
    // ***** Layout *****
    QHBoxLayout *hBoxTop = new QHBoxLayout;
    QVBoxLayout *vBoxBottom = new QVBoxLayout;
    QVBoxLayout *vBox1 = new QVBoxLayout;
    QVBoxLayout *vBox1V2 = new QVBoxLayout;
    QVBoxLayout *vBoxGauche = new QVBoxLayout;
    QVBoxLayout *vBoxDroite = new QVBoxLayout;
    QGridLayout *finalGrid = new QGridLayout;

    // ////////////////////////////////////// Page 2
//////////////////////////////////////
    QLabel *inserer = new QLabel("Inserer un convive");
    QLabel *supprimmer = new QLabel("Supprimer un convive");
    QLabel *insererPlusieurs = new QLabel("Inserer plusieurs élèves");
    QLabel *choisirFichier = new QLabel("Choisissez le fichier
correspondant");
    QLabel *supprimmerPlusieurs = new QLabel("Supprimer plusieurs élèves");
    QLineEdit *editNomInser = new QLineEdit;
    QLineEdit *editPrenomInser = new QLineEdit;
    QLineEdit *editNomSup = new QLineEdit;
    QLineEdit *editPrenomSup = new QLineEdit;
    QLineEdit *editClasseSup = new QLineEdit;
    QLineEdit *editClasseInserPlus = new QLineEdit;
    QLineEdit *editNaissanceInser = new QLineEdit;
    QLineEdit *editClasse_EmploieInser = new QLineEdit;
    QLineEdit *editSoldeInser = new QLineEdit;
```

```
    QLineEdit *editTarifInser = new QLineEdit;
    QLineEdit *editCodeInser = new QLineEdit;
    QLineEdit *editStatutInser = new QLineEdit;
    QString dossierChoisi;

    QPushButton *boutonValiderInser = new QPushButton("Valider");
    QPushButton *boutonValiderSup = new QPushButton("Valider");
    QPushButton *boutonValiderSupPlus = new QPushButton("Valider");
    QPushButton *boutonValiderInserPlus = new QPushButton("Valider");
    QPushButton *boutonParcourir = new QPushButton("Parcourir");
    // ***** Groupe Box *****
    QGroupBox *gBoxHautGauche2 = new QGroupBox;
    QGroupBox *gBoxBasGauche2 = new QGroupBox;
    QGroupBox *gBoxHautDroite2 = new QGroupBox;
    QGroupBox *gBoxBasDroite2 = new QGroupBox;
    // ***** Layout *****
    QFormLayout *formHautGauche2 = new QFormLayout;
    QFormLayout *formBasGauche2 = new QFormLayout;
    QFormLayout *formHautDroite2 = new QFormLayout;
    QFormLayout *formBasDroite2 = new QFormLayout;
    QGridLayout *finalGrid2 = new QGridLayout;

    // ////////////////////////////////////// Page 3
//////////////////////////////////////
    QLabel *txtImpressUneCarte = new QLabel("Imprimer une carte");
    QLabel *txtImpressPlusCarte = new QLabel("Imprimer plusieurs cartes");
    QLabel *txtImpressInfoUneCarte = new QLabel("Entrez les informations du
convive souhaité");
    QLabel *txtImpressInfoPlusCarte = new QLabel("Entrez la classe
souhaité");
    QLabel *txtImportantPhoto = new QLabel("ATTENTION: une photo du convive
doit être stocké au format:");
    QLabel *txtImportantPhotoPlusieurs = new QLabel("ATTENTION: Les photos
des convives doivent être stocké au format:");
    QLabel *txtImportantPhoto2 = new
QLabel("\"PhotoConvive/CLASSE/NOMPrénom\"");
    QLabel *txtImportantPhotoPlusieurs2 = new
QLabel("\"PhotoConvive/CLASSE/NOMPrénom\"");
    QLineEdit *editPrenomImpress = new QLineEdit;
    QLineEdit *editNomImpress = new QLineEdit;
    QLineEdit *editClasseImpress = new QLineEdit;
    QLabel *txtAvantImpress = new QLabel("Vous avez choisi le convive ");
    QLabel *txtAvantImpressInfo = new QLabel;
    QLabel *txtAvantImpress2 = new QLabel("Vous avez choisi la classe ");
    QLabel *txtAvantImpressInfo2 = new QLabel;
    QString recupPrenom = "";
    QString recupNom = "";
    QString recupClasse = "";
    int recupCode = 0;
    QPrinter printer;
    QPainter painter;
    QString nomFichier;
    bool IMPouPDF;

    QPushButton *boutonSelectImpress = new QPushButton("Valider");
    QPushButton *boutonImprimer = new QPushButton("Imprimer");
    QPushButton *boutonCreerPDF = new QPushButton("Créer PDF");
    QPushButton *boutonAnnulerCarte = new QPushButton("Annuler");
    QPushButton *boutonSelectImpress2 = new QPushButton("Valider");
    QPushButton *boutonImprimer2 = new QPushButton("Imprimer");
    QPushButton *boutonCreerPDF2 = new QPushButton("Créer PDF");
```

```
    QPushButton *boutonAnnulerCarte2 = new QPushButton("Annuler");
    // ***** Groupe Box *****
    QGroupBox *gBoxGauche3 = new QGroupBox;
    QGroupBox *gBoxGauche3V2 = new QGroupBox;
    QGroupBox *gBoxDroite3 = new QGroupBox;
    QGroupBox *gBoxDroite3V2 = new QGroupBox;
    // ***** Layout *****
    QVBoxLayout *vBoxGauche3 = new QVBoxLayout;
    QVBoxLayout *vBoxGauche3V2 = new QVBoxLayout;
    QVBoxLayout *vBoxDroite3 = new QVBoxLayout;
    QVBoxLayout *vBoxDroite3V2 = new QVBoxLayout;
    QGridLayout *finalGrid3 = new QGridLayout;

    // ////////////////////////////////////// Page 4
//////////////////////////////////////
    QLabel *txtInfoConviveSolde = new QLabel("Entrez les informations");
    QLineEdit *editPrenomSolde = new QLineEdit;
    QLineEdit *editNomSolde = new QLineEdit;
    QLabel *txtSoldeConvive = new QLabel("Solde du convive");
    double solde = 0;
    double tarif = 4;
    QLabel *soldeAffichage = new QLabel("...");
    QLineEdit *editPlusSolde = new QLineEdit;
    QLineEdit *editMoinsSolde = new QLineEdit;
    QLabel *txtTarifConvive =new QLabel("Tarif du convive");
    QLabel *txtAncienTarif =new QLabel("Tarif actuel:");
    QLabel *ancienTarif =new QLabel("...");
    QLabel *txtNouveauTarif =new QLabel("Nouveau tarif:");
    QLineEdit *editNouveauTarif = new QLineEdit;

    QLabel *txtPrenomSolde = new QLabel;
    QLabel *txtNomSolde = new QLabel;
    QLabel *txtClasseSolde = new QLabel;
    QPushButton *boutonValiderConviveSolde = new QPushButton("Valider");
    QPushButton *boutonPlusSolde = new QPushButton("+");
    QPushButton *boutonMoinsSolde = new QPushButton("-");
    QPushButton *boutonValiderTarif = new QPushButton("Valider");
    QPushButton *boutonValiderSoldeTarif = new QPushButton("Valider");
    // ***** Groupe Box *****
    QGroupBox *gBoxHaut4 = new QGroupBox;
    QGroupBox *gBoxHaut4V2 = new QGroupBox;
    QGroupBox *gBoxBas4 = new QGroupBox;
    QGroupBox *gBoxDroite4 = new QGroupBox;
    QGroupBox *gBoxGauche4 = new QGroupBox;
    // ***** Layout *****
    QVBoxLayout *vBoxHaut4 = new QVBoxLayout;
    QVBoxLayout *vBoxHaut4V2 = new QVBoxLayout;
    QVBoxLayout *vBoxBas4 = new QVBoxLayout;
    QGridLayout *gridDroite4 = new QGridLayout;
    QGridLayout *gridGauche4 = new QGridLayout;
    QGridLayout *finalGrid4 = new QGridLayout;

    // ////////////////////////////////////// Page 5
//////////////////////////////////////

    // ////////////////////////////////////// Page 6
//////////////////////////////////////

};
```

```
#endif // GESTION_INTENDANCE_H
```

- 2.2 creationPage1 :

```
void Gestion_Intendance::creationPage1()
{
    // *************** Groupe centre ***************
    vBox1->addWidget(txtLogin);
    vBox1->addWidget(editLogin);
    vBox1->addWidget(txtPassword);
    vBox1->addWidget(editPassword);
    vBox1->addWidget(boutonValider1);
    gBox1->setLayout(vBox1);
    gBox1->setStyleSheet("font: 12pt");
    txtLogin->setStyleSheet("font: 14pt");
    txtPassword->setStyleSheet("font: 14pt");
    editLogin->setPlaceholderText("Login");
    editPassword->setPlaceholderText("Mot de passe");
    editPassword->setEchoMode(QLineEdit::Password);
    editLogin->setFixedSize(125,23);
    editPassword->setFixedSize(125,23);
    boutonValider1->setFixedSize(80,40);
    txtLogin->setAlignment(Qt::AlignCenter);
    txtPassword->setAlignment(Qt::AlignCenter);
    editLogin->setAlignment(Qt::AlignCenter);
    editPassword->setAlignment(Qt::AlignCenter);
    vBox1->setAlignment(editLogin,Qt::AlignCenter);
    vBox1->setAlignment(editPassword,Qt::AlignCenter);
    vBox1->setAlignment(boutonValider1,Qt::AlignCenter);

    // *************** Groupe centre V2 ***************
    vBox1V2->addWidget(txtLoginRecup);
    vBox1V2->addWidget(boutonDeconnexion);
    gBox1V2->setLayout(vBox1V2);
    gBox1V2->setStyleSheet("font : 12pt");
    txtLoginRecup->setStyleSheet("font : 16pt");
    boutonDeconnexion->setFixedSize(100,40);
    txtLoginRecup->setAlignment(Qt::AlignCenter);
    vBox1V2->setAlignment(boutonDeconnexion,Qt::AlignCenter);
    gBox1V2->hide();

    // *************** Groupe Gauche Droite ***************
    imageRestoSco->setPixmap(QPixmap("restauSco.png").scaled(220,200));
    imageRestoSco2->setPixmap(QPixmap("restauSco.png").scaled(220,200));
    vBoxGauche->addWidget(imageRestoSco);
    gBoxGauche->setLayout(vBoxGauche);
    imageRestoSco->setAlignment(Qt::AlignCenter);
    vBoxDroite->addWidget(imageRestoSco2);
    gBoxDroite->setLayout(vBoxDroite);
    imageRestoSco2->setAlignment(Qt::AlignCenter);

    // *************** Goupe Haut ***************
    imageLogo->setPixmap(QPixmap("marianne.png"));
    hBoxTop->addWidget(imageLogo);
    gBoxTop->setLayout(hBoxTop);
    imageLogo->setAlignment(Qt::AlignCenter);

    // *************** Placement Grid ***************
    finalGrid->addWidget(gBoxTop,0,0,1,3);
    finalGrid->addWidget(gBoxGauche,1,0,2,1);
    finalGrid->addWidget(gBox1,1,1,2,1);
```

```
    finalGrid->addWidget(gBox1V2,1,1,2,1);
    finalGrid->addWidget(gBoxDroite,1,2,2,1);
    page1->setLayout(finalGrid);

    // *************** Bouton ***************
    connect(boutonValider1,SIGNAL(clicked(bool)), this, SLOT(connexion()));
    connect(boutonDeconnexion,SIGNAL(clicked(bool)), this,
SLOT(deconnexion()));
}
```

- 2.3 creationPage2 :

```
void Gestion_Intendance::creationPage2()
{
    // *************** Insérer un convive ***************
    formHautGauche2->addWidget(inserer);
    inserer->setStyleSheet("font: 14pt");
    inserer->setAlignment(Qt::AlignCenter);
    formHautGauche2->addRow("Prénom : ", editPrenomInser);
    formHautGauche2->addRow("Nom : ", editNomInser);
    formHautGauche2->addRow("Classe ou emploie : ",
editClasse_EmploieInser);
    formHautGauche2->addRow("Solde : ", editSoldeInser);
    formHautGauche2->addRow("Tarif : ", editTarifInser);
    formHautGauche2->addRow("Code en cas d'oublie : ", editCodeInser);
    formHautGauche2->addRow("1=Employé/0=Elève : ", editStatutInser);
    formHautGauche2->addWidget(boutonValiderInser);
    gBoxHautGauche2->setLayout(formHautGauche2);
    gBoxHautGauche2->setStyleSheet("font: 12pt");

    // *************** Supprimer un convive ***************
    formBasGauche2->addWidget(supprimmer);
    supprimmer->setStyleSheet("font: 14pt");
    supprimmer->setAlignment(Qt::AlignCenter);
    formBasGauche2->addRow("Prénom : ", editPrenomSup);
    formBasGauche2->addRow("Nom : ", editNomSup);
    formBasGauche2->addWidget(boutonValiderSup);
    gBoxBasGauche2->setLayout(formBasGauche2);
    gBoxBasGauche2->setStyleSheet("font: 12pt");

    // *************** Insérer des élèves ***************
    formHautDroite2->addWidget(insererPlusieurs);
    insererPlusieurs->setStyleSheet("font: 14pt");
    insererPlusieurs->setAlignment(Qt::AlignCenter);
    formHautDroite2->addRow("Classe : ", editClasseInserPlus);
    formHautDroite2->addWidget(choisirFichier);
    formHautDroite2->addWidget(boutonParcourir);
    formHautDroite2->addWidget(boutonValiderInserPlus);
    gBoxHautDroite2->setLayout(formHautDroite2);
    gBoxHautDroite2->setStyleSheet("font: 12pt");

    // *************** Supprimer des élèves ***************
    formBasDroite2->addWidget(supprimmerPlusieurs);
    supprimmerPlusieurs->setStyleSheet("font: 14pt");
    supprimmerPlusieurs->setAlignment(Qt::AlignCenter);
    formBasDroite2->addRow("Classe : ", editClasseSup);
    formBasDroite2->addWidget(boutonValiderSupPlus);
    gBoxBasDroite2->setLayout(formBasDroite2);
    gBoxBasDroite2->setStyleSheet("font: 12pt");
```

```
    // *************** Placement Grid ***************
    finalGrid2->addWidget(gBoxHautGauche2,0,0);
    finalGrid2->addWidget(gBoxBasGauche2,1,0);
    finalGrid2->addWidget(gBoxHautDroite2,0,1);
    finalGrid2->addWidget(gBoxBasDroite2,1,1);
    page2->setLayout(finalGrid2);

    // *************** Bouton ***************
    connect(boutonValiderInser,SIGNAL(clicked(bool)), this,
SLOT(insererConvive()));
    connect(boutonValiderSup,SIGNAL(clicked(bool)), this,
SLOT(supprimerConvive()));
    connect(boutonValiderSupPlus,SIGNAL(clicked(bool)), this,
SLOT(supprimerPlusieursConvives()));
    connect(boutonParcourir,SIGNAL(clicked(bool)), this,
SLOT(parcourir()));
    connect(boutonValiderInserPlus,SIGNAL(clicked(bool)), this,
SLOT(insererPlusieursConvives()));
}
```

- 2.4 creationPage3 :

```
void Gestion_Intendance::creationPage3()
{
    // *************** Groupe Gauche ***************
    vBoxGauche3->addWidget(txtImpressUneCarte);
    txtImpressUneCarte->setStyleSheet("font: 14pt");
    vBoxGauche3->addWidget(txtImpressInfoUneCarte);
    vBoxGauche3->addWidget(txtImportantPhoto);
    vBoxGauche3->addWidget(txtImportantPhoto2);
    vBoxGauche3->addWidget(editPrenomImpress);
    vBoxGauche3->addWidget(editNomImpress);
    vBoxGauche3->addWidget(boutonSelectImpress);
    txtImpressUneCarte->setAlignment(Qt::AlignCenter);
    txtImpressInfoUneCarte->setAlignment(Qt::AlignCenter);
    txtImportantPhoto->setAlignment(Qt::AlignCenter);
    txtImportantPhoto->setStyleSheet("font: 10pt");
    txtImportantPhoto2->setStyleSheet("font: 10pt");
    txtImportantPhoto2->setAlignment(Qt::AlignCenter);
    editPrenomImpress->setAlignment(Qt::AlignCenter);
    editNomImpress->setAlignment(Qt::AlignCenter);
    vBoxGauche3->setAlignment(editPrenomImpress,Qt::AlignCenter);
    vBoxGauche3->setAlignment(editNomImpress,Qt::AlignCenter);
    vBoxGauche3->setAlignment(boutonSelectImpress,Qt::AlignCenter);
    gBoxGauche3->setLayout(vBoxGauche3);
    gBoxGauche3->setStyleSheet("font: 12pt");
    editPrenomImpress->setPlaceholderText("Prenom");
    editNomImpress->setPlaceholderText("Nom");
    editPrenomImpress->setFixedSize(125,23);
    editNomImpress->setFixedSize(125,23);
    boutonSelectImpress->setFixedSize(80,40);
    gBoxGauche3->setLayout(vBoxGauche3);

    // *************** Groupe Gauche V2 ***************
    vBoxGauche3V2->addWidget(txtAvantImpress);
    txtAvantImpress->setAlignment(Qt::AlignCenter);
    vBoxGauche3V2->addWidget(txtAvantImpressInfo);
    txtAvantImpressInfo->setAlignment(Qt::AlignCenter);
    txtAvantImpressInfo->setStyleSheet("color: red");
    vBoxGauche3V2->addWidget(boutonImprimer);
    vBoxGauche3V2->addWidget(boutonCreerPDF);
```

```
vBoxGauche3V2->addWidget(boutonAnnulerCarte);
vBoxGauche3V2->setAlignment(boutonImprimer,Qt::AlignCenter);
vBoxGauche3V2->setAlignment(boutonCreerPDF,Qt::AlignCenter);
vBoxGauche3V2->setAlignment(boutonAnnulerCarte,Qt::AlignCenter);
gBoxGauche3V2->setStyleSheet("font: 12pt");
boutonImprimer->setFixedSize(80,40);
boutonCreerPDF->setFixedSize(80,40);
boutonAnnulerCarte->setFixedSize(80,40);
gBoxGauche3V2->setLayout(vBoxGauche3V2);
gBoxGauche3V2->hide();

// *************** Groupe Droite ***************
vBoxDroite3->addWidget(txtImpressPlusCarte);
txtImpressPlusCarte->setStyleSheet("font: 14pt");
vBoxDroite3->addWidget(txtImpressInfoPlusCarte);
vBoxDroite3->addWidget(txtImportantPhotoPlusieurs);
vBoxDroite3->addWidget(txtImportantPhotoPlusieurs2);
vBoxDroite3->addWidget(editClasseImpress);
vBoxDroite3->addWidget(boutonSelectImpress2);
txtImpressPlusCarte->setAlignment(Qt::AlignCenter);
txtImpressInfoPlusCarte->setAlignment(Qt::AlignCenter);
txtImportantPhotoPlusieurs->setAlignment(Qt::AlignCenter);
txtImportantPhotoPlusieurs->setStyleSheet("font: 10pt");
txtImportantPhotoPlusieurs2->setStyleSheet("font: 10pt");
txtImportantPhotoPlusieurs2->setAlignment(Qt::AlignCenter);
editClasseImpress->setAlignment(Qt::AlignCenter);
vBoxDroite3->setAlignment(editClasseImpress,Qt::AlignCenter);
vBoxDroite3->setAlignment(boutonSelectImpress2,Qt::AlignCenter);
gBoxDroite3->setLayout(vBoxDroite3);
gBoxDroite3->setStyleSheet("font: 12pt");
editClasseImpress->setPlaceholderText("Classe");
editClasseImpress->setFixedSize(125,23);
boutonSelectImpress2->setFixedSize(80,40);
gBoxDroite3->setLayout(vBoxDroite3);

// *************** Groupe Droite V2 ***************
vBoxDroite3V2->addWidget(txtAvantImpress2);
txtAvantImpress2->setAlignment(Qt::AlignCenter);
vBoxDroite3V2->addWidget(txtAvantImpressInfo2);
txtAvantImpressInfo2->setAlignment(Qt::AlignCenter);
txtAvantImpressInfo2->setStyleSheet("color: red");
vBoxDroite3V2->addWidget(boutonImprimer2);
vBoxDroite3V2->addWidget(boutonCreerPDF2);
vBoxDroite3V2->addWidget(boutonAnnulerCarte2);
vBoxDroite3V2->setAlignment(boutonImprimer2,Qt::AlignCenter);
vBoxDroite3V2->setAlignment(boutonCreerPDF2,Qt::AlignCenter);
vBoxDroite3V2->setAlignment(boutonAnnulerCarte2,Qt::AlignCenter);
gBoxDroite3V2->setStyleSheet("font: 12pt");
boutonImprimer2->setFixedSize(80,40);
boutonCreerPDF2->setFixedSize(80,40);
boutonAnnulerCarte2->setFixedSize(80,40);
gBoxDroite3V2->setLayout(vBoxDroite3V2);
gBoxDroite3V2->hide();

// *************** Placement Grid ***************
finalGrid3->addWidget(gBoxGauche3,0,0);
finalGrid3->addWidget(gBoxGauche3V2,0,0);
finalGrid3->addWidget(gBoxDroite3,0,1);
finalGrid3->addWidget(gBoxDroite3V2,0,1);
page3->setLayout(finalGrid3);
```

```
    // *************** Bouton ***************
    connect(boutonSelectImpress,SIGNAL(clicked(bool)), this,
SLOT(selectImpress()));
    connect(boutonImprimer,SIGNAL(clicked(bool)), this, SLOT(imprimer()));
    connect(boutonCreerPDF,SIGNAL(clicked(bool)), this, SLOT(creerPDF()));
    connect(boutonAnnulerCarte,SIGNAL(clicked(bool)), this,
SLOT(annulerCarte()));
    connect(boutonSelectImpress2,SIGNAL(clicked(bool)), this,
SLOT(selectImpress2()));
    connect(boutonImprimer2,SIGNAL(clicked(bool)), this,
SLOT(imprimer2()));
    connect(boutonCreerPDF2,SIGNAL(clicked(bool)), this,
SLOT(creerPDF2()));
    connect(boutonAnnulerCarte2,SIGNAL(clicked(bool)), this,
SLOT(annulerCarte2()));
}
```

- 2.5 creationPage4 :

```
void Gestion_Intendance::creationPage4()
{
    // *************** Groupe Haut ***************
    vBoxHaut4->addWidget(txtInfoConviveSolde);
    txtInfoConviveSolde->setStyleSheet("font: 14pt");
    vBoxHaut4->addWidget(editPrenomSolde);
    vBoxHaut4->addWidget(editNomSolde);
    vBoxHaut4->addWidget(boutonValiderConviveSolde);
    txtInfoConviveSolde->setAlignment(Qt::AlignCenter);
    editPrenomSolde->setAlignment(Qt::AlignCenter);
    editNomSolde->setAlignment(Qt::AlignCenter);
    vBoxHaut4->setAlignment(editPrenomSolde,Qt::AlignCenter);
    vBoxHaut4->setAlignment(editNomSolde,Qt::AlignCenter);
    vBoxHaut4->setAlignment(boutonValiderConviveSolde,Qt::AlignCenter);
    gBoxHaut4->setLayout(vBoxHaut4);
    gBoxHaut4->setStyleSheet("font: 12pt");
    editPrenomSolde->setPlaceholderText("Prenom");
    editNomSolde->setPlaceholderText("Nom");
    editPrenomSolde->setFixedSize(125,23);
    editNomSolde->setFixedSize(125,23);
    boutonValiderConviveSolde->setFixedSize(80,40);

    // *************** Groupe Haut V2 ***************
    txtPrenomSolde->setAlignment(Qt::AlignCenter);
    txtNomSolde->setAlignment(Qt::AlignCenter);
    txtClasseSolde->setAlignment(Qt::AlignCenter);
    vBoxHaut4V2->addWidget(txtPrenomSolde);
    vBoxHaut4V2->addWidget(txtNomSolde);
    vBoxHaut4V2->addWidget(txtClasseSolde);
    gBoxHaut4V2->setStyleSheet("font: 14pt");
    gBoxHaut4V2->setLayout(vBoxHaut4V2);
    gBoxHaut4V2->hide();

    // *************** Groupe Gauche ***************
    gridGauche4->addWidget(txtSoldeConvive,0,1,1,3);
    txtSoldeConvive->setStyleSheet("font: 14pt");
    txtSoldeConvive->setAlignment(Qt::AlignCenter);
    gridGauche4->addWidget(editPlusSolde,1,0);
    gridGauche4->addWidget(boutonPlusSolde,1,1);
    gridGauche4->addWidget(soldeAffichage,1,2);
    gridGauche4->addWidget(boutonMoinsSolde,1,3);
    gridGauche4->addWidget(editMoinsSolde,1,4);
```

```
    editPlusSolde->setFixedSize(45,23);
    boutonPlusSolde->setFixedSize(25,25);
    boutonMoinsSolde->setFixedSize(25,25);
    editMoinsSolde->setFixedSize(45,23);
    gridGauche4->setAlignment(editPlusSolde,Qt::AlignCenter);
    editPlusSolde->setAlignment(Qt::AlignCenter);
    gridGauche4->setAlignment(boutonPlusSolde,Qt::AlignCenter);
    soldeAffichage->setAlignment(Qt::AlignCenter);
    gridGauche4->setAlignment(boutonMoinsSolde,Qt::AlignCenter);
    gridGauche4->setAlignment(editMoinsSolde,Qt::AlignCenter);
    editMoinsSolde->setAlignment(Qt::AlignCenter);
    boutonPlusSolde->setStyleSheet("font: 16pt");
    boutonMoinsSolde->setStyleSheet("font: 16pt");
    gBoxGauche4->setLayout(gridGauche4);
    gBoxGauche4->setStyleSheet("font: 12pt");

    // *************** Groupe Droite ***************
    gridDroite4->addWidget(txtTarifConvive,0,0,1,2);
    txtTarifConvive->setStyleSheet("font: 14pt");
    txtTarifConvive->setAlignment(Qt::AlignCenter);
    gridDroite4->addWidget(txtAncienTarif,1,0);
    gridDroite4->addWidget(ancienTarif,1,1);
    gridDroite4->addWidget(txtNouveauTarif,2,0);
    gridDroite4->addWidget(editNouveauTarif,2,1);
    txtAncienTarif->setAlignment(Qt::AlignRight);
    ancienTarif->setAlignment(Qt::AlignLeft);
    txtNouveauTarif->setAlignment(Qt::AlignRight);
    gridDroite4->setAlignment(editNouveauTarif,Qt::AlignLeft);
    editNouveauTarif->setFixedSize(40,23);
    gridDroite4->addWidget(boutonValiderTarif,3,0,1,2);
    boutonValiderTarif->setFixedSize(60,25);
    gridDroite4->setAlignment(boutonValiderTarif,Qt::AlignCenter);
    gBoxDroite4->setLayout(gridDroite4);
    gBoxDroite4->setStyleSheet("font: 12pt");

    // *************** Groupe Bas ***************
    vBoxBas4->addWidget(boutonValiderSoldeTarif);
    vBoxBas4->setAlignment(boutonValiderSoldeTarif,Qt::AlignCenter);
    boutonValiderSoldeTarif->setStyleSheet("font: 12pt");
    gBoxBas4->setLayout(vBoxBas4);
    boutonValiderSoldeTarif->setFixedSize(80,40);

    // *************** Placement Grid ***************
    finalGrid4->addWidget(gBoxHaut4,0,0,1,2);
    finalGrid4->addWidget(gBoxHaut4V2,0,0,1,2);
    finalGrid4->addWidget(gBoxGauche4,1,0);
    finalGrid4->addWidget(gBoxDroite4,1,1);
    finalGrid4->addWidget(gBoxBas4,2,0,1,2);
    page4->setLayout(finalGrid4);

    gBoxGauche4->setDisabled(true);
    gBoxDroite4->setDisabled(true);
    gBoxBas4->setDisabled(true);

    // *************** Bouton ***************
    connect(boutonValiderConviveSolde,SIGNAL(clicked(bool)), this,
SLOT(connecterConviveSolde()));
    connect(boutonPlusSolde,SIGNAL(clicked(bool)), this,
SLOT(additionnerSolde()));
```

```
    connect(boutonMoinsSolde,SIGNAL(clicked(bool)), this,
SLOT(soustraireSolde()));
    connect(boutonValiderTarif,SIGNAL(clicked(bool)), this,
SLOT(validerTarif()));
    connect(boutonValiderSoldeTarif,SIGNAL(clicked(bool)), this,
SLOT(validerSoldeTarif()));
}
```

ANNEXE N° 3.1

```
/************************************ GestionBdd ***************************************
*   Constructeur de la classe GestionBdd                                                *
*   Param  IN           : host : adresse IP du serveur sous la forme "xxx.xxx.xxx.xxx"  *
*   Param  IN           : bdd  : nom de la base de données                              *
*   Param  IN           : login: login de l'utilisateur                                 *
*   Param  IN           : pwd  : mot de passe de l'utilisateur                          *
*****************************************************************************************/

GestionBdd::GestionBdd(QString host, QString bdd, QString login, QString pwd)
{
    dbOk = false;
    db = QSqlDatabase::addDatabase("QMYSQL");
    db.setHostName(host);
    db.setDatabaseName(bdd);
    db.setUserName(login);
    db.setPassword(pwd);
    db.setPort(3306);

    if(db.open())
    {
        dbOk = true;
    }
    else
    {
        qDebug()<< db.lastError().text();
    }

}
```

ANNEXE N° 3.2

```
class Fenetre_principal : public QWidget
{
    Q_OBJECT

public:
    Fenetre_principal(QWidget *parent =0);
    ~Fenetre_principal();

    QGroupBox            *partie_clavier();
    QGroupBox            *bt_suppression();
    QGroupBox            *gb_tableau();
    void                  ajouter_personne_tableau();
    Fenetre_secondaire   *fenetre;

private:
    Gestion_bdd      *bdd;
    QTableWidget     *la_liste = new QTableWidget(this);
    QPushButton      *bt_lettres[26];
    QPushButton      *bt_supprimer;
    QPushButton      *bt_quitter = new QPushButton;
    QString           nom_recherche;
    QLabel           *instruction = new QLabel;


public slots:
    void signal_button(int);
    void signal_button_supp();
    void cellule(int, int);
    void quitter();

};
```

ANNEXE N° 3.3

```
class FenetreCode: public QWidget
{
     Q_OBJECT
public:
    FenetreCode(int,st_nom_prenom, QWidget *parent =0);
    ~FenetreCode();


private:
    QGroupBox           *pave_numerique();
    QGroupBox           *bt_suppression();
    QGroupBox           *affichage_code_secret();
    QPushButton         *bt_chiffres[10];
    QPushButton         *bt_supprimer;
    QString              nom_rechercher;
    QString              code_secret;
    QLabel              *affichage_code = new QLabel;
    QLabel              *instruction = new QLabel;
    int                  compteur_mdp;


public slots:
    void signal_button(int);
    void signal_button_supp();

};

#endif // FENETRECODE_H
```