

Rapport Mini Projet :

Application Qt de gestion d'école



Réalisé par : AIT MOUSSA Mariam

Sommaire

- Introduction
- Première interface affichée
- Menu de l'application
- Section « Students »
- Manipulations dans la section « Students »
- Résultats de manipulations divers sur les sections
- Conclusion : points en cours de développement

Introduction

Qt est une application de développement des applications, dédiée principalement à la création des fenêtres.

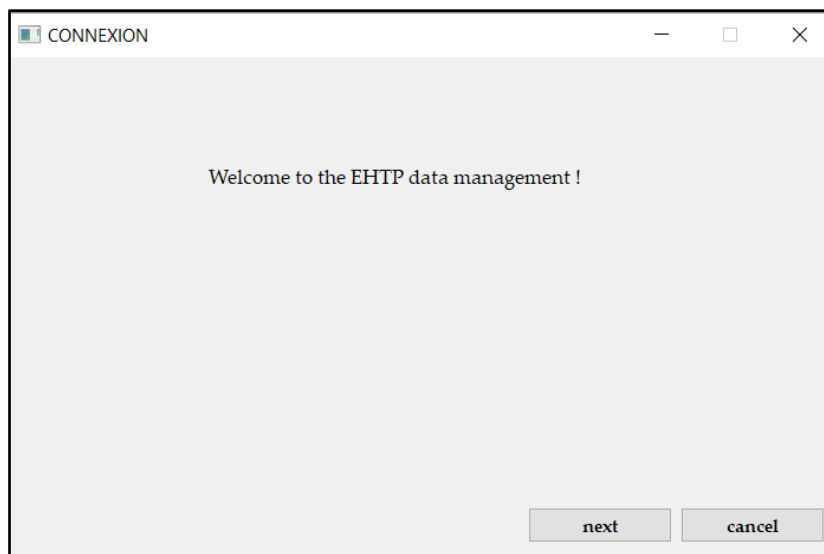
Qt est un « Framework » qui rassemble un ensemble de bibliothèques : SQL, XML, GUI...



Dans ce rapport nous allons décrire les différentes phases de l'élaboration de notre application Qt, qui sert à la gestion l'ensemble des étudiants, enseignants et modules de notre école.

Première interface affichée

Comme toute application, le lancement implique l'apparition d'un message de bienveillance :



Fichier .ui :

Cette première interface a été créée grâce à l'outil de création d'interfaces graphiques sur « Qt Creator », ci-dessous un aperçu des différents objets **widgets** qu'on nous avons utilisé :

Objet	Classe
CONNEXION	QMainWindow
centralwidget	QWidget
horizontalLayout	QHBoxLayout
pushButton_cancel	QPushButton
pushButton_connect	QPushButton
label_welcome	QLabel
menubar	QMenuBar
statusbar	QStatusBar

La fenêtre contient un **central widget** dans lequel on a mis l'**horizontal layout** (sert à organiser des objets horizontalement) est un label (message de bienveillance).

Dans l'**horizontal layout** on a disposé deux **PushButtons** : **Next** et **Cancel**, qui servent respectivement à **passer à la fenêtre suivante** et **quitter l'application**.

Fichier header :

Pour traduire les deux actions : **passer à la fenêtre suivante** et **quitter l'application**, nous avons initialisé des **slots** liées au **signal** clicked :

```
public:
    Options *window2;           | //DONNEE DE TYPE "OPTIONS", 2EME FENETRE A APPARAÎTRE
private slots:
    void on_pushButton_connect_clicked(); //SLOTS LIEE CHACUN AU SIGNAL "CLICKED" DU BOUTON/ NEXT OU CANCEL
    void on_pushButton_cancel_clicked();
```

Comme mentionné dans le script, on a initialisé un objet de type « Options », Options correspond à la classe de la deuxième fenêtre qui apparaîtra en cliquant le bouton Next.

Remarque : -on inclut le fichier header de la classe « options ».

- les noms des slots sont générés automatiquement, il suffit- au lieu d'utiliser la méthode « connect »- d'un clic droit sur le bouton, aller au slot, définir le signal correspondant et définir le slot dans le fichier source.

Fichier source :

Nous allons –dans notre fichier source- définir nos slots :

```
18 void CONNEXION::on_pushButton_connect_clicked() //definition du slot liee au signal next
19 {
20     hide();
21     window2= new Options (this);
22     window2->show();
23 }
24 void CONNEXION::on_pushButton_cancel_clicked() //definition du slot liee au signal cancel
25 {
26     close();
27 }
```

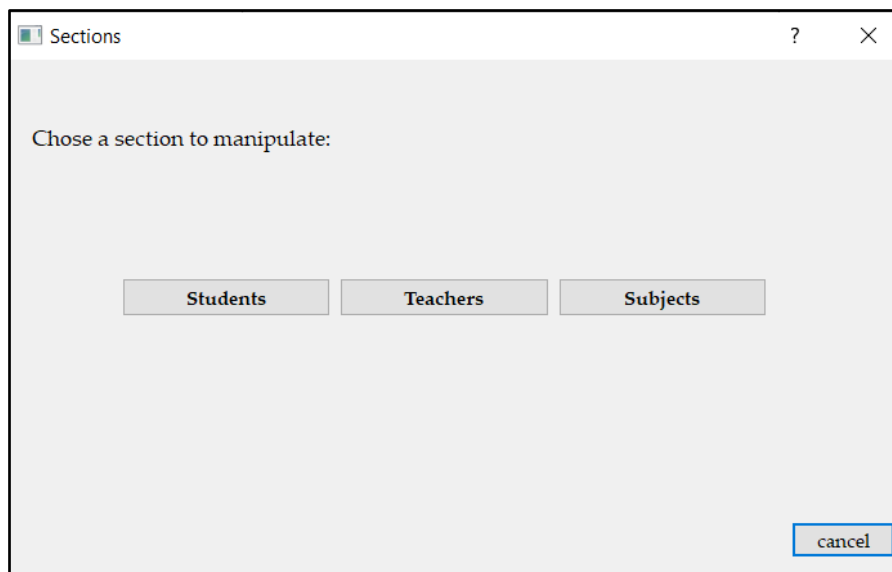
-Pour quitter l'application on utilise la méthode **close()**, ce bouton est présent dans toutes les fenêtres de l'application pour permettre à l'utilisateur de quitter l'application sur n'importe quel niveau.

-Un clic sur le bouton Next permet de masquer la fenêtre de bienveillance (en utilisant la méthode **hide()**) et de visualiser (**Show()**) la fenêtre suivante (objet de classe options).

Remarque : Pour ne pas revenir à chaque fois vers les mêmes reflexes dont on a déjà parlé, on mentionner que cette logique **d'initialiser un objet de la classe suivante et le visualiser après masquer la fenêtre (this)** est adopté sur tout le projet pour définir les slots qui servent à passer à la fenêtre qui suit.

Menu de l'application

Notre application sert à gérer les données correspondantes aux étudiants enseignants et modules, il est alors trivial de penser dans un premier lieu à diviser les volets de manipulation en trois sections, et sur ce, l'utilisateur est mené à faire son choix, la conception de la fenêtre des sections est la suivante :



Fichier .ui:

Les widgets utilisés sont quatre boutons : trois correspondent aux sections, (organisés dans un horizontal layout) et le quatrième c'est le bouton connu : cancel.

Un clic sur l'un des boutons : **Students**, **Teachers** ou **Subjects**, se traduit par la volonté de l'utilisateur à manipuler une telle section, donc selon son choix une nouvelle fenêtre va y apparaître.

Selon cette logique on a créé trois nouvelles classes de type « **QDialog** » correspondant chacune à une section (les classes sont : student, teacher et subject).

Fichier header:

Comme déjà mentionner un choix correspond à un clic (signal), on initialise un slot correspondant à chaque signal (student.clicked, teacher.clicked et subject.clicked)

```
private slots:                                     //SOLTS LIEE CHACUN AU SIGNAL CLICKED D4UN BOUTTON
    void on_pushButton_cancel_clicked();
    void on_pushButton_student_clicked();
    void on_pushButton_teacher_clicked();
    void on_pushButton_subject_clicked();
|
```

Fichier source:

Dans notre fichier source on aura affaire à définir les slots.

```

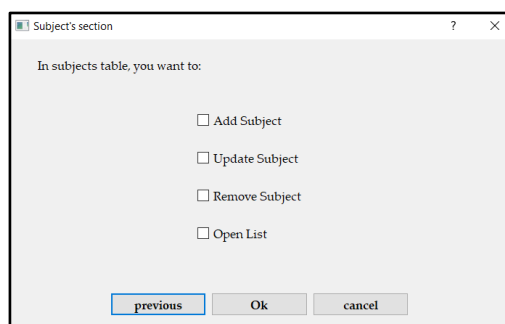
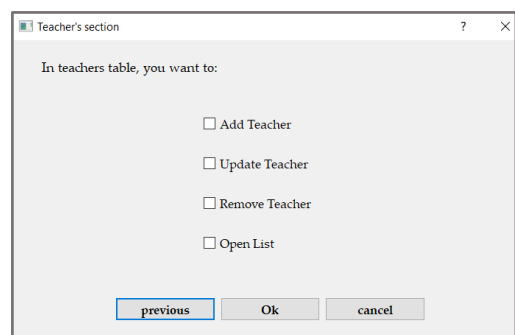
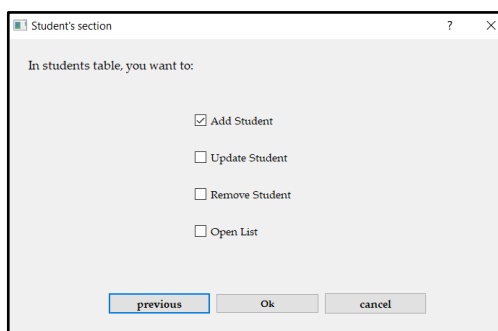
20 void Options::on_pushButton_cancel_clicked() //definition des slots liees a chaque signal
21 {
22     close(); //methode qui permet de quitter l'application
23 }
24 void Options::on_pushButton_student_clicked()
25 {
26     hide(); //methode qui permet de masquer la fenetre actuelle
27     Student *window3; //nouveau objet
28     window3= new Student(this);
29     window3->show(); //apparition d'une fenetre de type "student"
30 }
31 void Options::on_pushButton_teacher_clicked()
32 {
33     hide();
34     teacher *window4;
35     window4=new teacher(this);
36     window4->show();
37 }
38 void Options::on_pushButton_subject_clicked()
39 {
40     hide();
41     subject *window5;
42     window5=new subject(this);
43     window5->show();
44 }

```

Le raisonnement adopté est celui détaillé précédemment (ouvrir une fenêtre d'après l'autre).

Il suffit juste d'inclure les fichiers headers correspondant à chaque classe.

Selon le choix de l'utilisateur une des fenêtres suivantes va apparaître :



Remarque-1 : on mentionne à ce niveau que le travail sur les différentes sections est presque identique ; quelques spécifications que l'on résume ci-après :

- Au niveau de la base de données chaque table (étudiants, enseignants et matières) a ses propres attributs- on y viendra pour parler de la base de données juste après-.

Remarque-2 : l'application qu'on vient de créer sert la manipulation des données supposées déjà stockées, donc il faut assurer un espace de stockage adéquat. Nous, nous avons choisis d'associer notre application à une base de données pour assurer des critères de performances. La question qui se pose est la suivante :

QUEL SYSTÈME DE GESTION DE BASE DE DONNEES CHOISIR ?

Dans un premier lieu on a essayé « PostgreSQL » ; car c'est le SGBD qu'on a déjà travaillé avec.

Le problème c'était au niveau de la documentation insuffisante –presque rare- concernant l'association avec une application Qt.

Est pour la raison bar, on a travaillé avec le SGBD « **SqLite** ».

Maintenant le SGBD est choisi, comment alors l'associer à notre application ou plutôt **comment associer notre application Qt a une base de données de type SqLite ?**

Pour ne plus tarder, voilà ce qu'on a écrit comme code pour le faire : (pour établir une connexion vers la base de données et l'enlever)

```
public:
    CONNEXION(QWidget *parent = nullptr);
    ~CONNEXION();
    QSqlDatabase DB;
    bool open_connection() //METHODE POUR ETABLIR LA CONNEXION VERS LA DB
    {
        DB = QSqlDatabase::addDatabase("QSQLITE");
        DB.setDatabaseName("C:/Users/M/Desktop/S2's all/C++/build-AIT_MOUSSA_Mariam-Desktop_Qt_5_14_2_MinGW_64_bit-Debug/EHTP_DB.db");
        if (!DB.open())
        {return false;}
        else
        {return true;}
    }
    void close_connection() //METHODE POUR ARRETER LA CONNEXION VERS LA DB
    {
        DB.close();
        DB.removeDatabase(QSqlDatabase::defaultConnection);
    }
}
```

Notre base de données est placée dans le fichier build du projet !

Ces fonctions ont été définies dans la classe connexion (on avouera vers la fin de ce rapport la cause☺).

Comme cité dans le code on a suit la logique suivante :

- inclure la classe **QSqlDataBase**
- initialiser un objet de type QSqlDataBase
- déclarer le type (« SQLite »)
- associer l'objet crée a la base de données concrète (en utilisant la méthode **setdatabase**)
- la connexion est établie, il suffit de s'assurer de cela a chaque utilisation avec un test if.
- et pour enlever la connexion on ne fait que fermer la base de données (**close**) est arrêter la connexion (**removedatabase**).

Revenant maintenant à notre application, nous étions en train de faire un choix, et nous avons

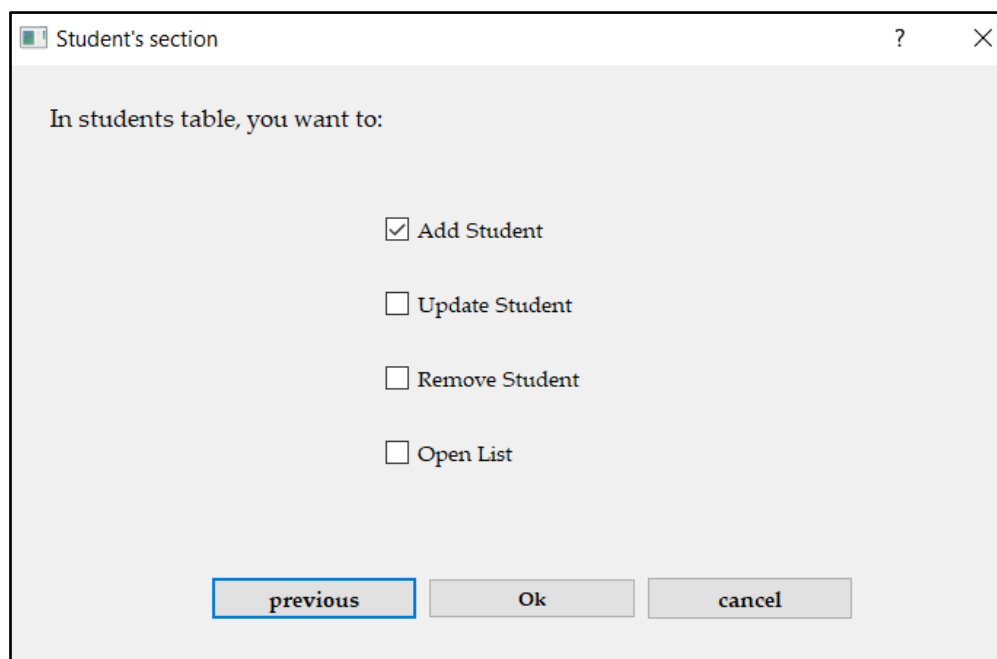
dit que le travail sur les différentes sections est presque identique, c'est pour cette raison que nous allons faire un choix et ne détailler dans ce rapport qu'une seule section (Sur ce, les autres sections sont automatiquement détaillées).

On choisit la section... « Students » (à noter que la section « Teachers » suit exactement la même logique et la section « Subjects » est plus simple que ça.

C'est parti !

Section « Students »

L'utilisateur a cliqué le bouton « Students », la fenêtre qui apparait est donnée ci-dessous :



Fichier .ui:

La fenêtre est composée d'un text_label, pour afficher un message, un vertical layout, dans lequel on a placé quatre check_Box correspond chacun a une manipulation possible, et un horizontal layout, dans lequel trois pushButtons sont placées.

Objet	Classe
Student	QDialog
horizontalLayout	QHBoxLayout
pushButton_Ok	QPushButton
pushButton_cancel	QPushButton
pushButton_previous	QPushButton
label_manipulate	QLabel
verticalLayout	QVBoxLayout
checkBox_add	QCheckBox
checkBox_delete	QCheckBox
checkBox_update	QCheckBox
checkBox_visualize	QCheckBox

Détaillons un peu les boutons qu'on a met ; en plus du bouton cancel on a le bouton previous qui sert à **revenir en arrière** vers la fenêtre des options, et le bouton Ok qui permet l'utilisateur de **confirmer son choix**.

Fichier header:

Dans le fichier header on initialise trois slots correspondant chacun a un signal et quatre checkBox correspondant chacun a un type de manipulation possible.

L'idée est de définir pour chaque choix une classe correspondant à une fenêtre dans laquelle on réalise la manipulation choisie.

Donc, au niveau du fichier header voilà ce qu'on a met :

```
25     private slots:
26         void on_pushButton_3_clicked();
27     |
28         void on_pushButton_2_clicked();
29
30         void on_pushButton_clicked();
31
32     private:
33         Ui::subject *ui;
34         window5_1 *add;
35         window5_2 *update;
36         window5_3 *remove;
37         window5_4 *open;
```

Fichier source:

Dans le fichier source on définit chaque slot comme donné ci-dessous :

```

18 void Student::on_pushButton_cancel_clicked()
19 {
20     close();
21 }
22 void Student::on_pushButton_Ok_clicked() //en couchant un choix et en cliquant le bouton "ok";
23 {
24     if(ui->checkBox_add->isChecked())
25     {
26         hide();
27         add=new window3_1(this);
28         add->show();
29     }
30     if(ui->checkBox_update->isChecked())
31     {
32         hide();
33         update=new window3_2(this);
34         update->show();
35     }
36     if(ui->checkBox_delete->isChecked())
37     {
38         hide();
39         remove=new window3_3(this);
40         remove->show();
41     }
42     if(ui->checkBox_visualize->isChecked())
43     {
44         visualize=new window3_4(this);
45         visualize->show();
46     }
47 }
48 void Student::on_pushButton_previous_clicked() //un click sur le bouton "previous" permet de revenir en arriere
49 {
50     hide();
51     Options *previous;
52     previous=new Options(this);
53     previous->show();
54 }

```

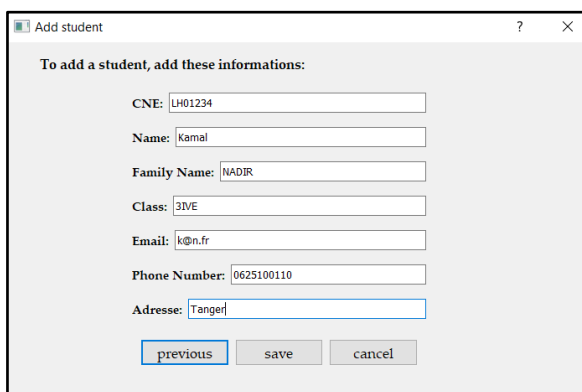
Le bouton « cancel » est toujours défini de la même manière mentionnée avant.

Le bouton « previous », sert à revenir à la fenêtre des options, pour réaliser cela, on masque (hide()) la fenêtre actuelle et on visualise (show()) un objet options qu'on a initialisé.

Pour choisir une manipulation, on coche un checkBox. Un clic sur le bouton « Ok » confirme le choix, et on affiche la fenêtre qui lui

correspond (ceci toujours avec les méthodes `hide()` et `show()` sans oublier l'inclure les fichiers headers correspondant à chaque classe utilisée).

Les différentes fenêtres qui correspondent à chaque choix sont illustrées ci-après :



Add student

To add a student, add these informations:

CNE:

Name:

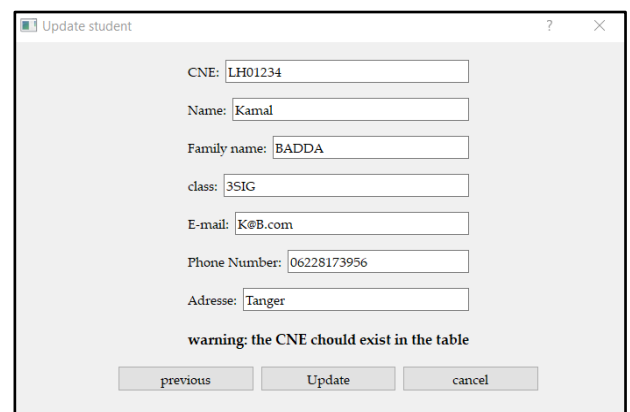
Family Name:

Class:

Email:

Phone Number:

Adresse:



Update student

CNE:

Name:

Family name:

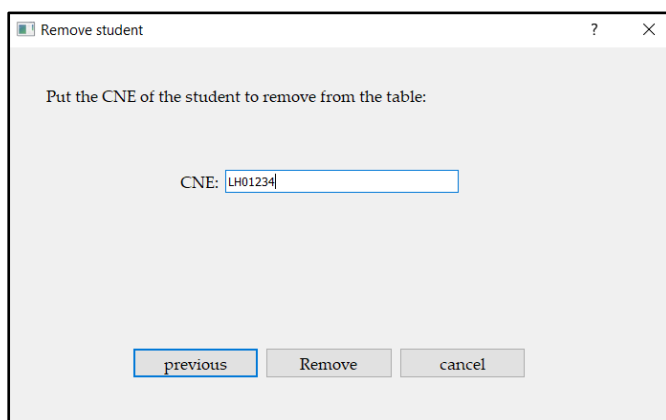
class:

E-mail:

Phone Number:

Adresse:

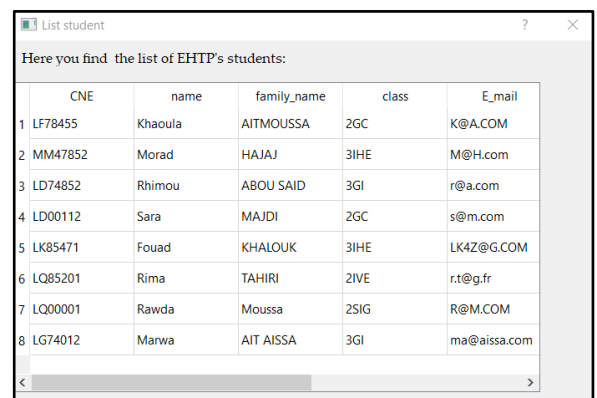
warning: the CNE should exist in the table



Remove student

Put the CNE of the student to remove from the table:

CNE:



List student

Here you find the list of EHTP's students:

	CNE	name	family_name	class	E_mail
1	LF78455	Khaoula	AITMOUSSA	2GC	K@A.COM
2	MM47852	Morad	HAJAJ	3IHE	M@H.com
3	LD74852	Rhimou	ABOU SAID	3GI	r@a.com
4	LD00112	Sara	MAJDI	2GC	s@m.com
5	LK85471	Fouad	KHALOUK	3IHE	LK4Z@G.COM
6	LQ85201	Rima	TAHIRI	2IVE	r.t@g.fr
7	LQ00001	Rawda	Moussa	2SIG	R@M.COM
8	LG74012	Marwa	AIT AISSA	3GI	ma@aissa.com

< >

Manipulations dans la section « Students »

Dans cette partie du rapport nous allons effectuer les différents types de manipulations possibles.

Ces manipulations sont exactement celles qu'on effectue dans une base de données ; ajouter un élément a la table, modifier un élément, retirer un élément ou afficher les éléments (y en a bien d'autres détails qu'un SGBD offre et qu'on ne détaille pas ici).

Notre base de données créée sous SQLite « EHTTP_BD.db » est constitué de trois tables (students, teachers et subjects) ; les manipulations qu'on va effectuer dans cette section de l'application « Students » vont être effectué dans la table « students » de la base de données.

- Ajout d'un Etudiant à la table :

On a coché la case « Add Student » et puis cliqué le bouton Ok, la fenêtre qui apparait est la suivante :

The screenshot shows a Qt-style dialog box titled "Add student". Inside, there is a label "To add a student, add these informations:". Below this are seven input fields, each with a label and a text box: "CNE:" with "LH01234", "Name:" with "Kamal", "Family Name:" with "NADIR", "Class:" with "3IVE", "Email:" with "k@n.fr", "Phone Number:" with "0625100110", and "Adresse:" with "Tanger". At the bottom of the dialog are three buttons: "previous" (highlighted with a blue border), "save", and "cancel".

Fichier .ui:

Les objets utilisés sont illustrés ci-après :

Objet	Classe
✓ window3_1	QDialog
horizontalLayout_8	QHBoxLayout
pushButton_cancel	QPushButton
pushButton_previous	QPushButton
pushButton_save	QPushButton
label	QLabel
verticalLayout	QVBoxLayout
horizontalLayout	QHBoxLayout
CNE	QLabel
lineEdit1	QLineEdit
horizontalLayout_2	QHBoxLayout
Name	QLabel
lineEdit_2	QLineEdit
horizontalLayout_3	QHBoxLayout
FAMILYNAME	QLabel
lineEdit_3	QLineEdit
horizontalLayout_4	QHBoxLayout
class_2	QLabel
lineEdit_4	QLineEdit
horizontalLayout_5	QHBoxLayout
lineEdit_5	QLineEdit
mail	QLabel
horizontalLayout_6	QHBoxLayout
lineEdit_6	QLineEdit
phonenumber	QLabel
horizontalLayout_7	QHBoxLayout
adresse	QLabel
lineEdit_7	QLineEdit

Y'en a des labels pour définir quelle information saisir dans quelle emplacement, des line_edit pour permettre à l'utilisateur la saisie de ces informations (tout cela est organisé selon des vertical et horizontal layout), en plus y'en les boutons fondamentaux cancel et previous et en plus le bouton « save ».

Fichier header :

On ne fait qu'initialiser les slots liées aux signaux des boutons :

```
private slots:  
    void on_pushButton_cancel_clicked();  
  
    void on_pushButton_save_clicked();  
|  
    void on_pushButton_previous_clicked();
```

Fichier source :

On définit les slots liées aux signaux des boutons ; pour le bouton cancel et previous c'est le même principe adopté précédemment.

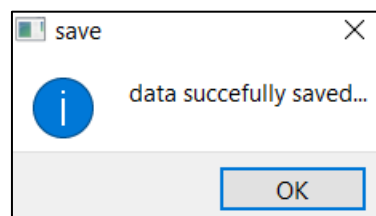
On détaille le slot liée au signal clic du bouton save.

Un clic sur le bouton save veut dire que l'utilisateur enregistre les informations saisies comme élément dans la table students de la base de données :

```
30 void window3_1::on_pushButton_save_clicked()
31 {
32     CONNEXION connexion; //Besoin d'etablir la connexion vers la DB
33     QString cne,name,Fname,classe,mail,phone,adresse; //On retient les inputs qu'on a définie l'utilisateur
34     cne=ui->lineEdit1->text();
35     name=ui->lineEdit_2->text();
36     Fname=ui->lineEdit_3->text();
37     classe=ui->lineEdit_4->text();
38     mail=ui->lineEdit_5->text();
39     phone=ui->lineEdit_6->text();
40     adresse=ui->lineEdit_7->text();
41     if (!connexion.open_connection())
42     {QMessageBox::warning(this,"warning","Failed to connect to the database...");
43     return;}
44     connexion.open_connection();
45     QSqlQuery query; //pour definir une requete SQL
46     query.prepare("insert into students values('"+cne+"','"+name+"','"+Fname+"','"+classe+"','"+mail+"','"+phone+"','"+adresse+"')");
47     if(query.exec())
48     {QMessageBox::information(this,tr("save"),tr("data succefully saved..."));
49     connexion.close_connection();}
50     else
51     {QMessageBox::critical(this,tr("error::"),query.lastError().text());}
52 }
53
```

- D'abord il faut se connecter à la base de données ; on créant un objet de la classe connexion (pour pouvoir utiliser les fonctions open_connection et close_connection).
- On retient les informations de l'élément à ajouter dans des variables string.

- On vérifie la connexion avec la base de données par un test if (sinon un message « warning » va être affiche).
- On crée un objet de la classe QSqlQuery, on le prépare (définition de la requête SQL à exécuter).
- Le Query s'exécute pour le vérifier on utilise la classe « QMessageBox » pour afficher une confirmation (s'il a été exécuté) et un message warning sinon. Dans notre exemple le message qui se visualise est le suivant :



On vérifie au niveau de la base de données et on trouve que l'élément est ajouté avec succès :

	CNE	name	family_name	class	E_mail	phone_number	adresse
1	LF78455	Khaoula	AITMOUSSA	2GC	K@A.COM	0612368545	Fnideq
2	MM47852	Morad	HAJAJ	3IHE	M@H.com	0618271580	Dakhla
3	LD74852	Rhimou	ABOU SAID	3GI	r@a.com	0654781239	Meknes
4	LD00112	Sara	MAJDI	2GC	s@m.com	0633221166	Taounate
5	LK85471	Fouad	KHALOUK	3IHE	LK4Z@G.COM	0614255869	LARACHE
6	LQ85201	Rima	TAHIRI	2IVE	r.t@g.fr	0612457896	Ouazzane
7	LQ00001	Rawda	Moussa	2SIG	R@M.COM	0652857845	Tetouan
8	LG74012	Marwa	AIT AISSA	3GI	ma@aissa.com	0569874132	Rabat
9	LF98765	Anwar	DEBDI	2SIG	na@d.fr	0628173945	AL-Hociema
10	LH01234	Kamal	NADIR	3IVE	k@n.fr	0625100110	Tanger

Result: 10 enregistrements ramenés en 45ms

- Modification d'un Etudiant dans la table :

On a coché la case « Update Student » et puis cliqué le bouton Ok, la fenêtre qui apparait est la suivante :

Update student

CNE: LH01234

Name: Kamal

Family name: BADDA

class: 3SIG

E-mail: K@B.com

Phone Number: 06228173956

Adresse: Tanger

warning: the CNE should exist in the table

previous Update cancel

Fichier .ui : Les objets utilisés sont exactement les mêmes que la fenêtre précédente (Ajout d'un étudiant).

Fichier header :

```
18     private slots:
19         void on_cancel_clicked();
20
21         void on_Update_clicked();
22
23         void on_previous_clicked();
24
```

Slots liées à chaque signal de chaque bouton.

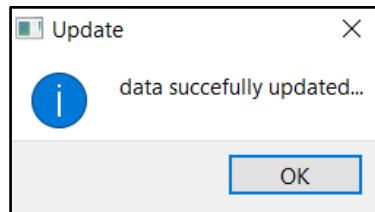
Fichier source :

Les slots qu'on a définie pour les boutons de la fenêtre « add student » sont les mêmes !

Le seul changement est au niveau des définitions de la requête SQL à appliquer :

```
29 void window3_2::on_Update_clicked()
30 {
31     CONNEXION connexion;
32     QString cne,name,Fname,classe,mail,phone,adresse;
33     cne=ui->lineEdit1->text();
34     name=ui->lineEdit_2->text();
35     Fname=ui->lineEdit_3->text();
36     classe=ui->lineEdit_4->text();
37     mail=ui->lineEdit_5->text();
38     phone=ui->lineEdit_6->text();
39     adresse=ui->lineEdit_7->text();
40     if (!(connexion.open_connection()))
41     {QMessageBox::warning(this,"warning","Failed to connect to the database...");
42     return;}
43     connexion.open_connection();
44     QSqlQuery query;
45     query.prepare("update students set CNE='"+cne+"',name='"+name+"',family_name='"+Fname+"',
46     |'class='"+classe+"',E_mail='"+mail+"',phone_number='"+phone+"',adresse='"+adresse+"' where CNE='"+cne+"'");
47     if(query.exec())
48     {QMessageBox::information(this,tr("Update"),tr("data succefully updated..."));
49     connexion.close_connection();}
50     else
51     {QMessageBox::critical(this,tr("error::"),query.lastError().text());}
52 }
```

On clique le bouton Update pour modifier les informations de l'élément créé précédemment :



Dans la base de données on trouve qu'effectivement l'élément est modifié :

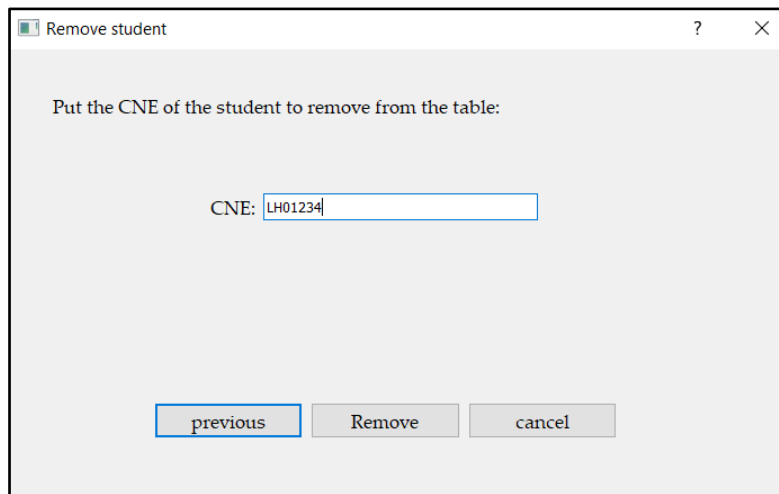
	CNE	name	family_name	class	E_mail	phone_number	adresse
1	LF78455	Khaoula	AITMOUSSA	2GC	K@A.COM	0612368545	Fnideq
2	MM47852	Morad	HAJAJ	3IHE	M@H.com	0618271580	Dakhla
3	LD74852	Rhimou	ABOU SAID	3GI	r@a.com	0654781239	Meknes
4	LD00112	Sara	MAJDI	2GC	s@m.com	0633221166	Taounate
5	LK85471	Fouad	KHALOUK	3IHE	LK4Z@G.COM	0614255869	LARACHE
6	LQ85201	Rima	TAHIRI	2IVE	r.t@g.fr	0612457896	Ouazzane
7	LQ00001	Rawda	Moussa	2SIG	R@M.COM	0652857845	Tetouan
8	LG74012	Marwa	AIT AISSA	3GI	ma@aissa.com	0569874132	Rabat
9	LF98765	Anwar	DEBDI	2SIG	na@d.fr	0628173945	AL-Hociema
10	LH01234	Kamal	BADDA	3SIG	K@B.com	06228173956	Tanger

Result: 10 enregistrements ramenés en 94ms
At line 1:

- Suppression d'un Etudiant de la table :

La tache devient plus simple ; dans le langage SQL la suppression d'un élément nécessite de

déterminer sa clé primaire (ici le CNE) donc notre fenêtre est la suivante :



Fichier .ui :

On a utilisé les mêmes types d'objets.

Fichier header :

Que la définition des slots :

```
18     private slots:
19         void on_cancel_clicked();
20
21         void on_Remove_clicked();
22
23         void on_previous_clicked();
24
25     private:
26         Ui::window2_2 *ui;
```

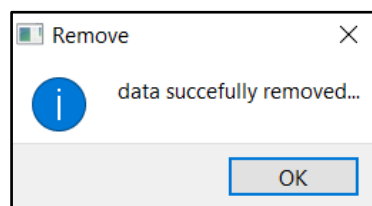
Fichier source :

Comme peut être remarqué le slot liée au bouton remove se base à son tour sur le slot liée au bouton save de la fenêtre add ;

```
30 void window3_3::on_Remove_clicked()
31 {
32     CONNEXION connexion;
33     QString cne;
34     cne=ui->lineEdit->text();
35     if (!(connexion.open_connection()))
36     {QMessageBox::warning(this,"warning","Failed to connect to the database...");
37     return;}
38     connexion.open_connection();
39     QSqlQuery query;
40     query.prepare("delete from students where CNE='"+cne+"'");
41     if(query.exec())
42     {QMessageBox::information(this,tr("Remove"),tr("data succefully removed..."));
43     connexion.close_connection();}
44     else
45     {QMessageBox::critical(this,tr("error::"),query.lastError().text());}
46 }
47
```

Encore le seul changement qu'on mentionne est au niveau du Quercy !

On clique le bouton remove et voilà ce qui se passe :

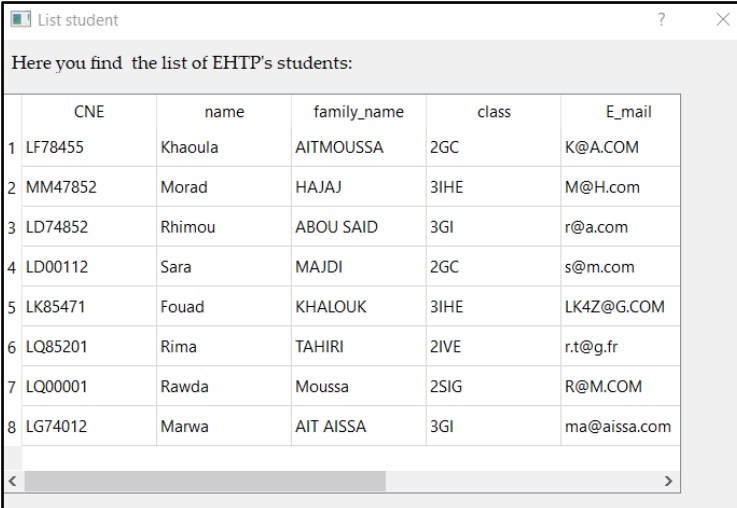


On vérifie la base de données et la suppression est effectuée avec succès :

	CNE	name	family_name	class	E_mail	phone_number	adresse
1	LF78455	Khaoula	AITMOUSSA	2GC	K@A.COM	0612368545	Fnideq
2	MM47852	Morad	HAJAJ	3IHE	M@H.com	0618271580	Dakhla
3	LD74852	Rhimou	ABOU SAID	3GI	r@a.com	0654781239	Meknes
4	LD00112	Sara	MAJDI	2GC	s@m.com	0633221166	Taounate
5	LK85471	Fouad	KHALOUK	3IHE	LK4Z@G.COM	0614255869	LARACHE
6	LQ85201	Rima	TAHIRI	2IVE	r.t@g.fr	0612457896	Ouazzane
7	LQ00001	Rawda	Moussa	2SIG	R@M.COM	0652857845	Tetouan
8	LG74012	Marwa	AIT AISSA	3GI	ma@aissa.com	0569874132	Rabat
9	LF98765	Anwar	DEBDI	2SIG	na@d.fr	0628173945	AL-Hociema

- Affichage de la liste des étudiants :

On a coché la case « Open List » et puis cliqué le bouton Ok, la fenêtre qui apparait est la suivante :



	CNE	name	family_name	class	E_mail
1	LF78455	Khaoula	AITMOUSSA	2GC	K@A.COM
2	MM47852	Morad	HAJAJ	3IHE	M@H.com
3	LD74852	Rhimou	ABOU SAID	3GI	r@a.com
4	LD00112	Sara	MAJDI	2GC	s@m.com
5	LK85471	Fouad	KHALOUK	3IHE	LK4Z@G.COM
6	LQ85201	Rima	TAHIRI	2IVE	r.t@g.fr
7	LQ00001	Rawda	Moussa	2SIG	R@M.COM
8	LG74012	Marwa	AIT AISSA	3GI	ma@aissa.com

Fichier .ui :

On a besoin d'afficher la table de la base de données toute entière, on définit notre fenêtre avec les objets suivants :

Objet	Classe
▼ window3_4	QDialog
label	QLabel
tableView	QTableView

On va utiliser l'objet de type **QTableView** pour afficher la table d la base de donnée ;

Fichier source :

Le travail ici est différent que précédemment, la totalité du code est définie dans le constructeur de la classe ;

```

3  #include"connexion.h"
4  #include<QSqlQuery>
5  #include<QSqlQueryModel>
6  #include"student.h"
7
8  window3_4::window3_4(QWidget *parent) :
9      QDialog(parent),
10     ui(new Ui::window3_4)
11 {
12     ui->setupUi(this);
13     setFixedSize(700,450);
14     setWindowTitle("List student");
15     CONNEXION connexion;
16     QSqlQueryModel *model=new QSqlQueryModel(); //on definie un model
17     connexion.open_connection(); //on etablie la connexion vers la BD
18     QSqlQuery *query=new QSqlQuery(connexion.DB);
19     query->prepare("select * from students");
20     query->exec();
21     model->setQuery(*query); //on definie le query pour le model
22     ui->tableView->setModel(model); //on associe le model a la tableview
23 }

```

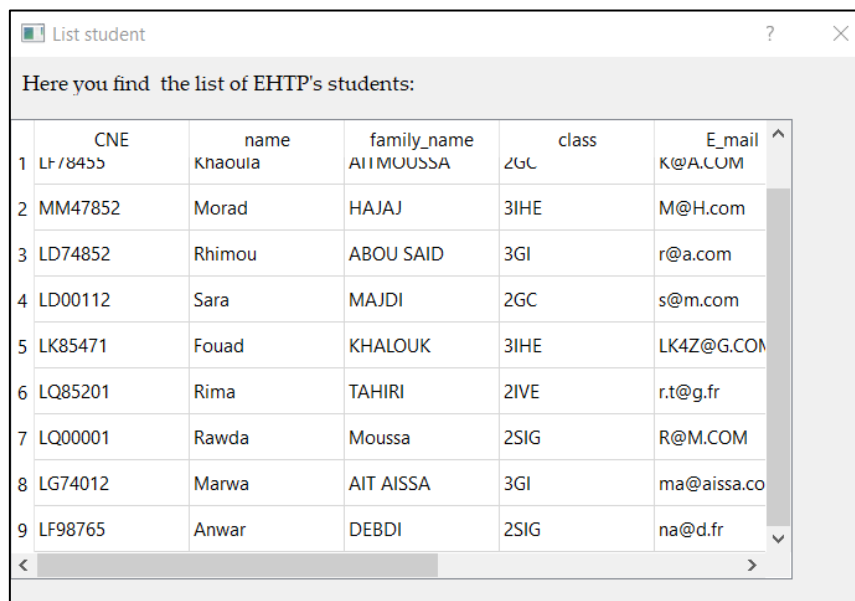
On utilise la classe **QSqlQueryModel** qui offre des model supportant les données retenus par une lecture (Query) SQL ; on définit un model

on lui associe un Quercy et on attribue le tout (par la méthode **setModel**) à la table view.

Comparaison :

	CNE	name	family_name	class	E_mail	phone_number	adresse
1	LF78455	Khaoula	AITMOUSSA	2GC	K@A.COM	0612368545	Fnideq
2	MM47852	Morad	HAJAJ	3IHE	M@H.com	0618271580	Dakhla
3	LD74852	Rhimou	ABOU SAID	3GI	r@a.com	0654781239	Meknes
4	LD00112	Sara	MAJDI	2GC	s@m.com	0633221166	Taounate
5	LK85471	Fouad	KHALOUK	3IHE	LK4Z@G.COM	0614255869	LARACHE
6	LQ85201	Rima	TAHIRI	2IVE	r.t@g.fr	0612457896	Ouazzane
7	LQ00001	Rawda	Moussa	2SIG	R@M.COM	0652857845	Tetouan
8	LG74012	Marwa	AIT AISSA	3GI	ma@aissa.com	0569874132	Rabat
9	LF98765	Anwar	DEBDI	2SIG	na@d.fr	0628173945	AL-Hociema

Dans la base de données



Here you find the list of EHTP's students:

	CNE	name	family_name	class	E_mail
1	LF78455	Khaoula	AITMOUSSA	2GC	K@A.COM
2	MM47852	Morad	HAJAJ	3IHE	M@H.com
3	LD74852	Rhimou	ABOU SAID	3GI	r@a.com
4	LD00112	Sara	MAJDI	2GC	s@m.com
5	LK85471	Fouad	KHALOUK	3IHE	LK4Z@G.COM
6	LQ85201	Rima	TAHIRI	2IVE	r.t@g.fr
7	LQ00001	Rawda	Moussa	2SIG	R@M.COM
8	LG74012	Marwa	AIT AISSA	3GI	ma@aissa.co
9	LF98765	Anwar	DEBDI	2SIG	na@d.fr

L'output de l'application

Par cette manipulation on termine la série de manipulations qu'on a adoptées dans ce rapport.

Résultats de manipulations divers sur les sections

Comme on a déjà mentionné, au niveau des outils utilisés et raisonnement adopté, il n'y a pas une grande différence entre les différentes sections de cette application ; c'est pour cette raison nous allons dans cette partie donner que des schémas récapitulatifs sur les résultats de l'utilisation des différentes sections (ceci appart la section des étudiants développée suffisamment plus haut):

- Section des enseignants « Teachers » :

Teacher's section

In teachers table, you want to:

☐ Add Teacher

☐ Update Teacher

☐ Remove Teacher

☐ Open List

previous Ok cancel

Cocher **Add Teacher**
puis cliquer **Ok**

Cocher **Add Teacher**
puis cliquer **Ok**

Cocher **Remove Teacher**
puis cliquer **Ok**

Add teacher

To add a teacher, add these informations:

CNE: LQ98765

Full Name: Farah KADIRRI

Departement: GI

E-mail: F@K.FR

Phone Number: 0600000000

Adresse: NADOR

previous save cancel

Passer les informations
demandées et puis
cliquer **Save**

Update teacher

CNE: LQ98765

Full Name: Farah KHADDARI

Departement: MIG

E-mail: F@K.FR

Phone Number: 0600000000

Adresse: NADOR

warning: the CNE should exist in the table

previous Update cancel

Passer les informations
demandées et puis
cliquer **Update**

Remove teacher

Put the CNE of the student to remove from the table:

CNE: LQ98765

previous Remove cancel

Passer les
informations
demandées et puis
cliquer **Remove**

save

data succefully saved...

OK

Au niveau BD

Update

data succefully updated...

OK

Au niveau BD

Remove

data succefully removed...

OK

Au niveau BD

```
1 SELECT * FROM teachers
```

	CNE	Full_Name	departement	E_mail	phone_number	adresse
1	If14258	Idriss TAOUNATI	MIG	t@l.com	0645129632	Casablanca
2	LV03284	Idriss THAMI	MIG	KL45FC@4.COM	0645125896	RABAT
3	LH45782	Rajae RAEFAT	GE	r@r.com	0512360985	Nador
4	LP01234	Ahmed KADIR	GI	lp@lp.fr	0628172400	EL-Hoclema
5	LS01234	Farah MORTADA	ENVIRONEMENT	fa@mo.com	0655667788	Tanger
6	LS78451	Jawad BAKKALI	GI	j@b.com	0600221144	Fes
7	frde124	rti	ndf	ern	0612345678	fs
8	LQ98765	Farah KADIRRI	GI	F@K.FR	0600000000	NADOR

Result: 8 enregistrements ramenés en 56ms
At line 1:
SELECT * FROM teachers

```
1 SELECT * FROM teachers
```

	CNE	Full_Name	departement	E_mail	phone_number	adresse
1	If14258	Idriss TAOUNATI	MIG	t@l.com	0645129632	Casablanca
2	LV03284	Idriss THAMI	MIG	KL45FC@4.COM	0645125896	RABAT
3	LH45782	Rajae RAEFAT	GE	r@r.com	0512360985	Nador
4	LP01234	Ahmed KADIR	GI	lp@lp.fr	0628172400	EL-Hoclema
5	LS01234	Farah MORTADA	ENVIRONEMENT	fa@mo.com	0655667788	Tanger
6	LS78451	Jawad BAKKALI	GI	j@b.com	0600221144	Fes
7	frde124	rti	ndf	ern	0612345678	fs
8	LQ98765	FARAH KHADDARI	MIG	F@K.FR	0600000000	NADOR
9	frde124	rti	ndf	ern	0612345678	fs
10	LQ98765	FARAH KHADDARI	MIG	F@K.FR	0600000000	NADOR

Result: 10 enregistrements ramenés en 220ms
At line 1:
SELECT * FROM teachers

```
1 select * from teachers
```

	CNE	Full_Name	departement	E_mail	phone_number	adresse
1	If14258	Idriss TAOUNATI	MIG	t@l.com	0645129632	Casablanca
2	LV03284	Idriss THAMI	MIG	KL45FC@4.COM	0645125896	RABAT
3	LH45782	Rajae RAEFAT	GE	r@r.com	0512360985	Nador
4	LP01234	Ahmed KADIR	GI	lp@lp.fr	0628172400	EL-Hoclema
5	LS01234	Farah MORTADA	ENVIRONEMENT	fa@mo.com	0655667788	Tanger
6	LS78451	Jawad BAKKALI	GI	j@b.com	0600221144	Fes
7	frde124	rti	ndf	ern	0612345678	fs
8	LQ98765	FARAH KHADDARI	MIG	F@K.FR	0600000000	NADOR
9	frde124	rti	ndf	ern	0612345678	fs

Result: 9 enregistrements ramenés en 69ms
At line 1:
select * from teachers

- Section des modules « Subjects » :

Subject's section

In subjects table, you want to:

☐ Add Subject

☐ Update Subject

☐ Remove Subject

☐ Open List

previous Ok cancel

Cocher **Add Subject** puis cliquer **Ok**

Cocher **Update Subject** puis cliquer **Ok**

Cocher **Remove Subject** puis cliquer **Ok**

Add subject

To add a subject, add these informations:

ID: 1070

Subject's label: DAO

Warning: the ID of the subject is unique!

previous Save cancel

Passer les informations demandées et puis cliquer **Save**

Update subject

ID: 1070

Subject's label: DESSIN

warning: theID should exist in the table:

previous Update cancel

Passer les informations demandées et puis cliquer **Update**

Remove subject

Put the ID of the student to remove from the table:

ID: 1070

previous Remove cancel

Passer les informations demandées et puis cliquer **Remove**

save

data succcessfully saved...

OK

Au niveau BD

```
1 select * from subjects
```

	id	label
1	1025	MRD
2	1026	Topographie
3	1027	MMC
4	1028	SIG Desktop
5	1029	POO
6	1070	DAO

Result: 6 enregistrements ramenés en 56ms
At line 1:
select * from subjects

Update

data succcessfully updated...

OK

Au niveau BD

```
1 select * from subjects
```

	id	label
1	1025	MRD
2	1026	Topographie
3	1027	MMC
4	1028	SIG Desktop
5	1029	POO
6	1070	DESSIN

Result: 6 enregistrements ramenés en 27ms
At line 1:
select * from subjects

Remove

data succcessfully removed...

OK

Au niveau BD

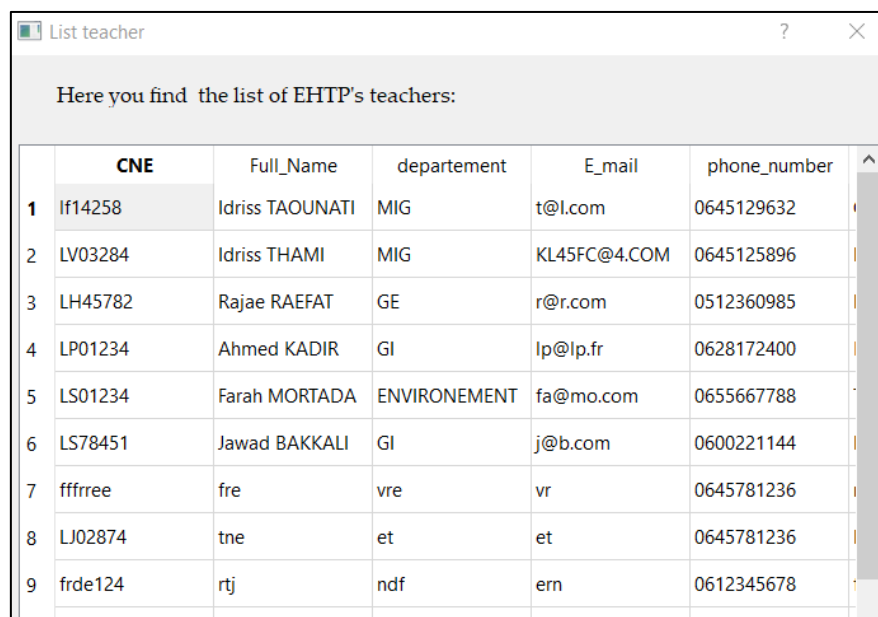
```
1 select * from subjects
```

	id	label
1	1025	MRD
2	1026	Topographie
3	1027	MMC
4	1028	SIG Desktop
5	1029	POO

Result: 5 enregistrements ramenés en 18ms
At line 1:
select * from subjects

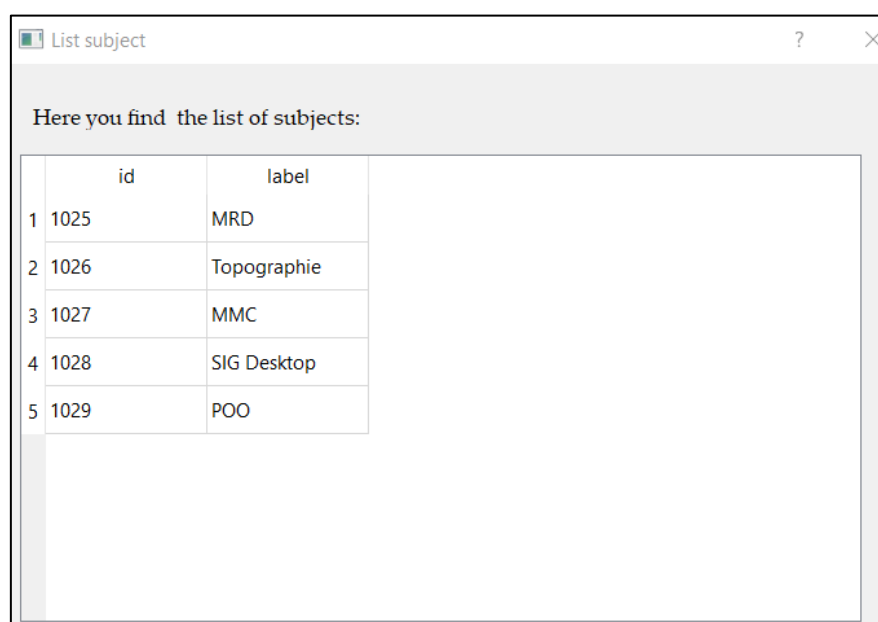
Remarque-1 : il nous reste la dernière manipulation, c'est la visualisation des listes :

Dans la fenêtre des manipulations on coche le check Box « Open List » et on clique « Ok », les outputs sont les suivants :



The screenshot shows a window titled 'List teacher' with a subtitle 'Here you find the list of EHTP's teachers:'. It contains a table with 6 columns: CNE, Full_Name, departement, E_mail, and phone_number. The table lists 9 teachers.

	CNE	Full_Name	departement	E_mail	phone_number
1	If14258	Idriss TAOUNATI	MIG	t@l.com	0645129632
2	LV03284	Idriss THAMI	MIG	KL45FC@4.COM	0645125896
3	LH45782	Rajae RAEFAT	GE	r@r.com	0512360985
4	LP01234	Ahmed KADIR	GI	lp@lp.fr	0628172400
5	LS01234	Farah MORTADA	ENVIRONEMENT	fa@mo.com	0655667788
6	LS78451	Jawad BAKKALI	GI	j@b.com	0600221144
7	fffree	fre	vre	vr	0645781236
8	LJ02874	tne	et	et	0645781236
9	frde124	rtj	ndf	ern	0612345678



The screenshot shows a window titled 'List subject' with a subtitle 'Here you find the list of subjects:'. It contains a table with 2 columns: id and label. The table lists 5 subjects.

	id	label
1	1025	MRD
2	1026	Topographie
3	1027	MMC
4	1028	SIG Desktop
5	1029	POO

Remarque-2 : « Personnalisation des fentres »

Dans ce volet on a traité les points suivants :

- Chaque fenêtre est intitulée.
- La taille de toutes les fenêtres est la même : (700,400)
- Tous les labels suivent le même style textuel :
(Platino Linotpe-Gras)

Remarque-3 : pourquoi on a défini les fonctions open_connection et close_connection dans la classe connexion (de bienveillance) ?

La réponse comme on a mentionné est vers la fin du rapport, voir alors le dernier point en cours de développement dans la conclusion !

Conclusion : points en cours de développement

Comme conclusion de ce travail on met ci-après quelques points traités dans la phase conceptuelle est non pas encore réalisé (d'une raison de délai du projet ou outil utilisé non encore acquis !) :

- Système de gestion de base de données :

En termes de performances pour des grandes bases de données, il y a d'autres SGBD plus performant que SQLite, citant par exemple PostgreSQL (SGBD déjà manipulé lors du cours Langage SQL), mais à cause de la rareté de la documentation on a travaillé avec le plus simple comme première initialisation vers le Framework Qt.

- Développement de requêtes ; sélection avec conditions :

Ici on mentionne que les manipulations adoptés dans ce projet sont les plus basiques qu'un SDBD offre (ca répond bien sûr au cahier de charges du projet !), y'en a bien d'autres requête qu'on peut implémenter et rend l'application plus recommandé pour des bases de données plus grandes.

- Attribution des critères de protection :

L'idée est d'attribuer à chaque utilisateur possible de l'application un mot de passe et un nom d'utilisateur, ainsi attribuer une fenêtre de protection pour vérifier si l'utilisateur appartient à une table (appelée par exemple : Admins dans la même base de données).

(Voilà la réponse a la question de la remarque 3)



Réalisé par : AIT MOUSSA Mariam dans le cadre du projet Qt/C++ qui s'inscrit dans le cadre du module « Programmation Orientée Objet/C++ ».