



PROJET SQLITE EN C++

DÉVELOPPÉ PAR RAFFANEL GUILHEM, RICCI BASTIEN ET GUILLEMOT KILLIAN

SOMMAIRE

- I. Objectif du projet
- II. Idée de conception
- III. Développement point par point
- IV. Images

OBJECTIFS DU PROJET

- Projet développé en C++ et utilisant la Programmation Orienté Objet
- Utilisation de la bibliothèque QT permettant le développement d'interface graphique
- Utilisation d'une base de donnée SQLite

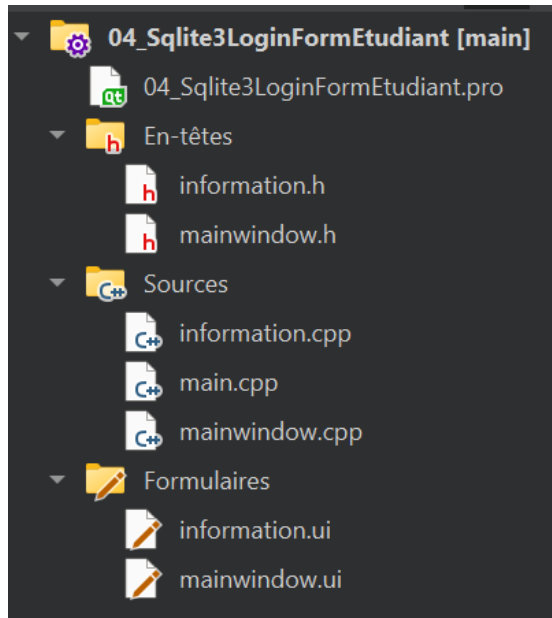
IDÉE DE CONCEPTION

L'idée de conception de ce projet est assez basique :

- Tout d'abord avoir une page de connexion du type Nom d'utilisateur / Mot de passe
- Si l'utilisateur n'est pas reconnu renvoyer une erreur de connexion
- Sinon accéder à la deuxième page
- La deuxième page affiche les données liées à l'utilisateur, ici elle affiche ce que l'utilisateur aime

DÉVELOPPEMENT POINT PAR POINT

Image du projet



Le projet est conçu de la manière suivante :

- 2 Formulaire liés à 2 classes :
 - MainWindow est la classe principale
 - Information est la classe secondaire héritant de MainWindow

DÉVELOPPEMENT POINT PAR POINT (MAINWINDOW)

Constructeur de la classe MainWindow

- Sa fonction première est de créer la base de l'UI
- Ensuite il ouvre la base de données

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    if(MainWindow::ouvreDb()) {
        ui->status->setText( "Connexion à la Base de Données OK.");
    } else {
        ui->status->setText("Problème de connexion à la Base de Données");
    }
}
```

Destructeur de la classe MainWindow

```
MainWindow::~MainWindow()
{
    delete ui;
    fermeDb();
}
```

DÉVELOPPEMENT POINT PAR POINT (MAINWINDOW)

Cette fonction permet de se connecter à la base de données

```
QSqlDatabase MainWindow::db;

bool MainWindow::ouvreDb()
{
    MainWindow::db = QSqlDatabase::addDatabase("QSQLITE");
    MainWindow::db.setDatabaseName("../accounts.sql3");
    if(MainWindow::db.open()) {
        //qDebug() <<"OuvreDb : connexion OK";
        return true;
    }
    else {
        //qDebug() <<"OuvreDb : connexion KO";
        return false;
    }
}
```

Cette fonction permet de fermer la base de données quand on ne l'utilisera plus

```
void MainWindow::fermeDb()
{
    MainWindow::db.close();
    MainWindow::db.removeDatabase(db.connectionName());
}
```

DÉVELOPPEMENT POINT PAR POINT (MAINWINDOW)

Cette fonction gère les connexions des clients

- Tout d'abord elle récupère les entrées textes
- Puis elle appelle la base de données pour vérifier si l'utilisateur a entré des informations correctes
- Si oui elle définit l'id de l'utilisateur
- Puis elle appelle la fonction `on_connect()`
- Sinon elle renvoie des erreurs

```
void MainWindow::on_pb_connect_clicked()
{
    QString login = ui->le_login->text();
    QString pass = ui->le_password->text();
    QString accord;
    QSqlQuery qry;
    qDebug() <<"Traitement connexion";
    QString sql = "SELECT COUNT(*) as nb, idAccount, sex FROM accounts WHERE login='"+login+"' AND password='"+pass+"'";
    if(qry.exec(sql)) {
        qry.next();
        int nb=qry.value(0).toInt();
        switch(nb) {
            case 0:
                ui->status->setText("Mot de passe incorrect...");
                break;
            case 1: {
                accord = qry.value(2).toInt() == 1 ? "e" : "";
                ui->status->setText("Tu es connecté"+accord+" avec le compte "+QString::number(qry.value(1).toInt()));
                // ouvrir 2eme
                sql = "SELECT idAccount From accounts where login = '" + login + "'";
                if (qry.exec(sql)) {
                    qry.next();
                    int nb = qry.value(0).toInt();
                    setId(nb);
                } else {
                    qDebug() << "Erreur sql ??";
                }
                on_connect();
                break;
            }
            default:
                ui->status->setText("Duplication de compte !");
                break;
        }
    } else {
        qDebug() <<"Erreur sql ??";
    }
}
```


DÉVELOPPEMENT POINT PAR POINT (MAINWINDOW)

Création de la deuxième page

- Tout d'abord elle appelle la fonction `hide()`
- Elle supprime la page actuelle
- Elle crée un nouvel objet `information`
- Elle définit l'id
- Puis elle appelle les différentes fonctions qui vont afficher les données
- Enfin elle affiche le tout

```
void MainWindow::on_connect()
{
    this->hide();
    delete ui;
    Information *information = new Information;
    information->setId(this->get_id());
    qDebug() << information->get_id();
    information->hide();
    information->Print_lb();
    information->Print_lst();
    information->show();
}
```

DÉVELOPPEMENT POINT PAR POINT (MAINWINDOW)

Cette fonction vérifie si l'id est correcte

- Si l'id est supérieur à 0 et inférieur ou égal au nombre d'utilisateurs total
- Alors on cache les informations du Widget principal

```
void MainWindow::hide()
{
    qDebug() << this -> get_id() << "id";
    qDebug() << get_taille() << "taille";
    if (this -> get_id() > 0 && this -> get_id() <= get_taille()){
        ui -> centralwidget -> hide();
    }
}
```

Cette fonction récupère le nombre d'utilisateur enregistrés

```
int MainWindow::get_taille()
{
    int taille = 0;
    QSqlQuery qry;
    QString sql = "SELECT count(*) as idAccount from accounts";
    if(qry.exec(sql)) {
        qry.next();
        taille = qry.value(0).toInt();
    }
    return taille;
}
```

DÉVELOPPEMENT POINT PAR POINT (MAINWINDOW)

Cette fonction est un Setter de la variable
Id

```
void MainWindow::setId(int newId)
{
    id = newId;
}
```

Cette fonction est un Getter de la variable
Id

```
int MainWindow::get_id()
{
    return id;
}
```

DÉVELOPPEMENT POINT PAR POINT (MAINWINDOW)

Cette fonction récupère le nom de l'utilisateur

```
QString MainWindow::get_Name(int id)
{
    QSqlQuery qry;
    QString sql = "SELECT login From accounts where idAccount = '" + QString::number(id) + "'";
    if(qry.exec(sql)) {
        qry.next();
        return qry.value(0).toString();
    }
    return QString();
}
```

Cette fonction récupère les informations de ce que l'utilisateur aime

```
QList<QString> MainWindow::get_Aime(int id)
{
    QList<QString> lst = QList<QString>();
    QSqlQuery qry;
    QString sql = "SELECT libelle From aime where idAccount = '" + QString::number(id) + "'";
    if(qry.exec(sql)) {
        while (qry.next()){
            lst.append(qry.value(0).toString());
        }
        return lst;
    }
    return QList<QString>();
}
```

DÉVELOPPEMENT POINT PAR POINT (INFORMATION)

Constructeur de la classe Information

```
Information::Information(QWidget *parent) :  
    ui(new Ui::Information)  
{  
    ui->setupUi(this);  
}
```

Destructeur de la classe Information

```
Information::~~Information()  
{  
    delete ui;  
}
```

DÉVELOPPEMENT POINT PAR POINT (INFORMATION)

Cette fonction affiche le nom d'utilisateur sur la fenêtre

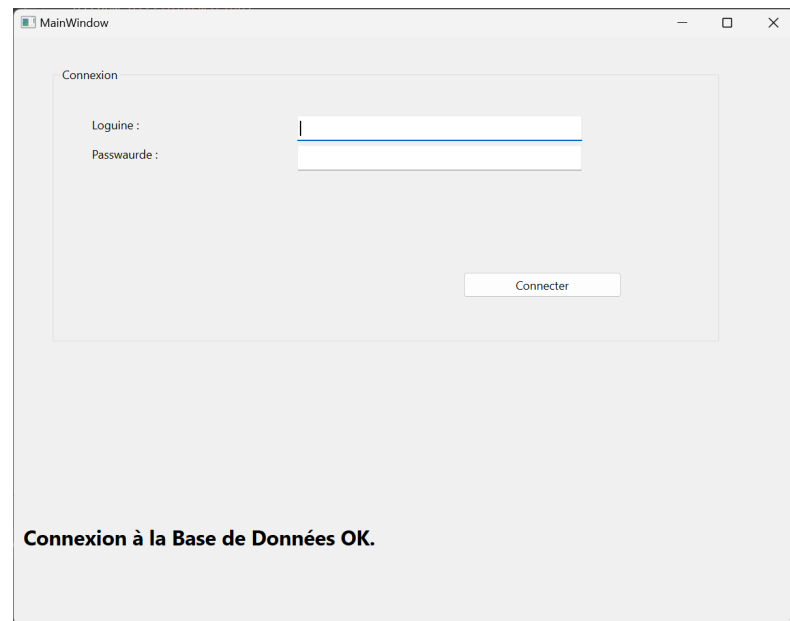
```
void Information::Print_lb()
{
    ui -> lb_nom -> setText(get_Name(get_id()));
}
```

Cette fonction affiche tout les éléments renvoyés par get_Aime()

```
void Information::Print_lst()
{
    for(int i = 0; i < get_Aime(get_id()).size(); i++) {
        ui -> listeAime -> addItem(get_Aime(get_id())[i]);
    }
}
```

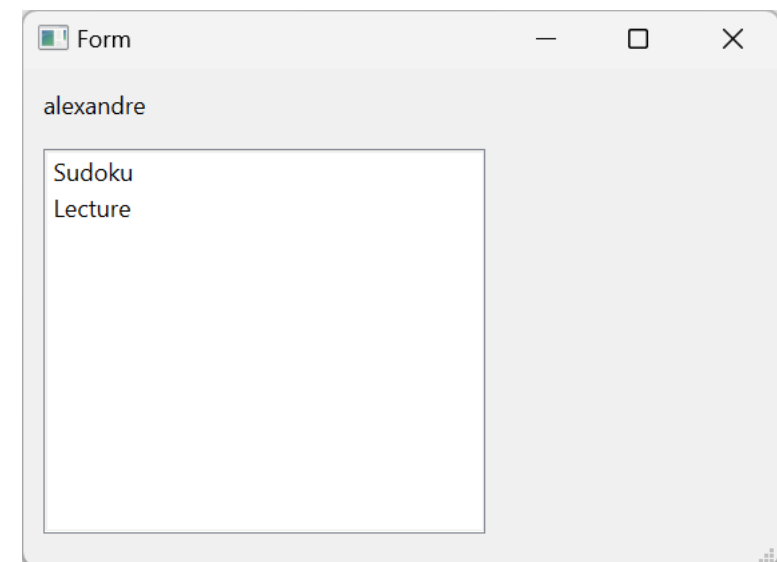
IMAGES

MainWindow



The screenshot shows a window titled "MainWindow" with a light gray background. Inside, there is a "Connexion" section with two input fields: "Loguine :" and "Passwaurde :". The "Loguine :" field contains a single character "l". Below these fields is a "Connecter" button. At the bottom of the window, a status message reads "Connexion à la Base de Données OK."

Information



The screenshot shows a window titled "Form" with a light gray background. The window displays the name "alexandre" at the top. Below it, there is a section titled "Sudoku" with a sub-label "Lecture". The main area of the window is a large, empty white rectangle, likely intended for displaying a Sudoku puzzle.



MERCI

Pour nous contacter :

- GUILHEM.RAFFANEL@YNOV.COM
- BASTIEN.RICCI@YNOV.COM
- KILLIAN.GUILLEMOT@YNOV.COM