Application de Gestion D'étudiants

Realise par : **Ahmed Belkahla, Med Ali Zaabi , Montassar Amri**

Professeur : **Kamel Karoui**

Annexe

I) Introduction

II) Technologies utilisées

III) Présentation de chaque technologie

IV) Conception de l'application

V) Côté Serveur

1) Features
2) La base de données

VI) Côté Client

1) Création d'un Socket et Connexion au serveur :
2) Envoie des données
3) Réception des données
4)Fermeture du Socket

VII) Interface Graphique

VIII) Prochaines Etapes

Introduction :

Le but de cette application est de retrouver les informations d'un étudiant à partir de son identifiant unique , enregistré dans un serveur dédié , afin de faciliter le contrôle de flux de données et se basant sur une interface graphique intuitive et facile à utiliser .

Technologies utilisées :

Lors de ce projet on a utilisé :

-Java
-Communication avec les sockets
-Le système de gestion de base de données : MySQL
-Un VPS (Virtual private Server)

Présentation de chaque technologie :

- Java :

Java est langage de programmation orienté objet multi platforme (Cross Platform Language) puissant et permet de développer des applications Client-Serveur. Dans Ce projet on s'est basé sur le langage Java .

- Communication avec les sockets :

Les sockets permettent d'établir une connexion TCP/IP entre 2 programmes sans restriction sur la localisation des deux programmes. Afin d'assurer la communication entre la partie Client et Serveur nous avons utilisé les sockets.

- Le SGBD MySQL :

Afin de faire la bonne gestion de la base de données nous avons utilisé MySQL dans la partie Serveur

- Un VPS (Virtual Private Server) :

Un VPS est un serveur dédié qui utilise des mécanismes de virtualisation. Afin de simuler un environnement de production réel on a utilisé un VPS pour mettre en place la partie serveur + la base de données .

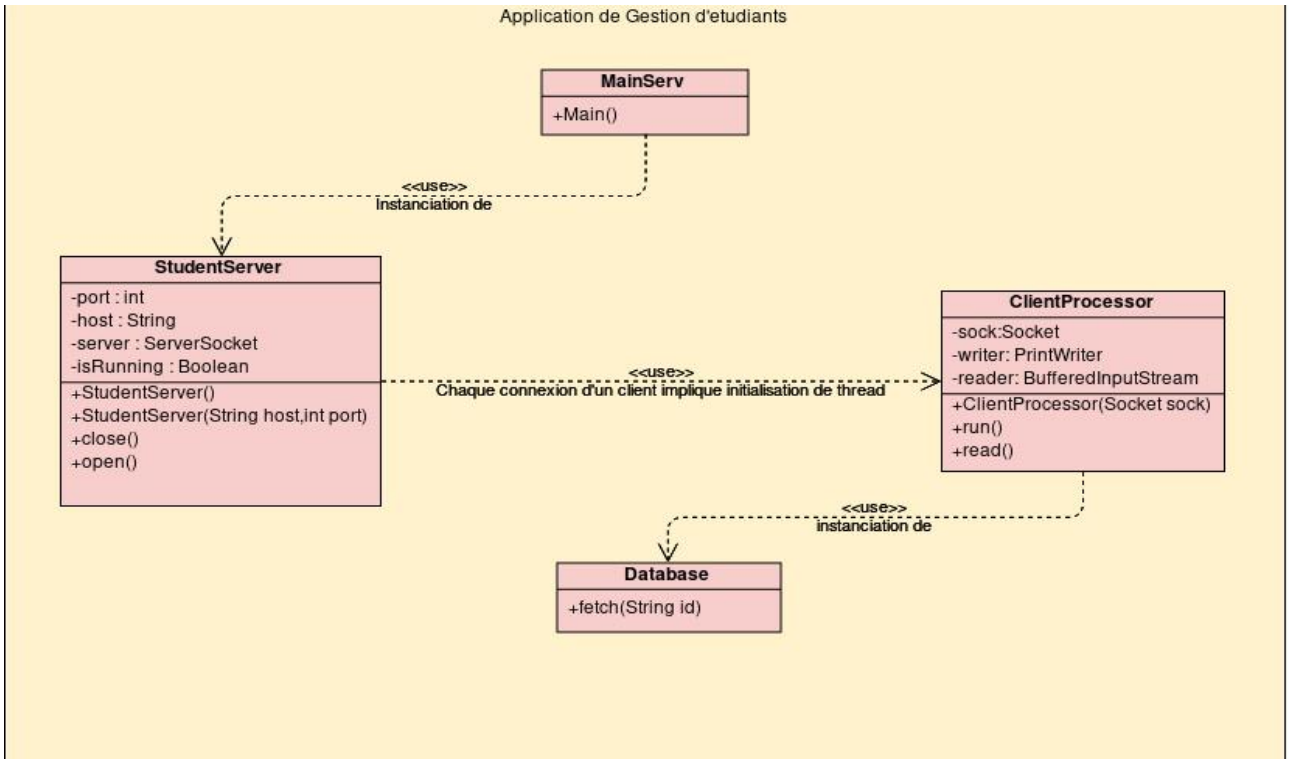
Conception de l'application :

En termes d'introduction voici un diagramme d'activité qui décrit de façon générale les échanges au cours de l'activité de l'application :

- Partie Serveur :

Diagramme de Classes :

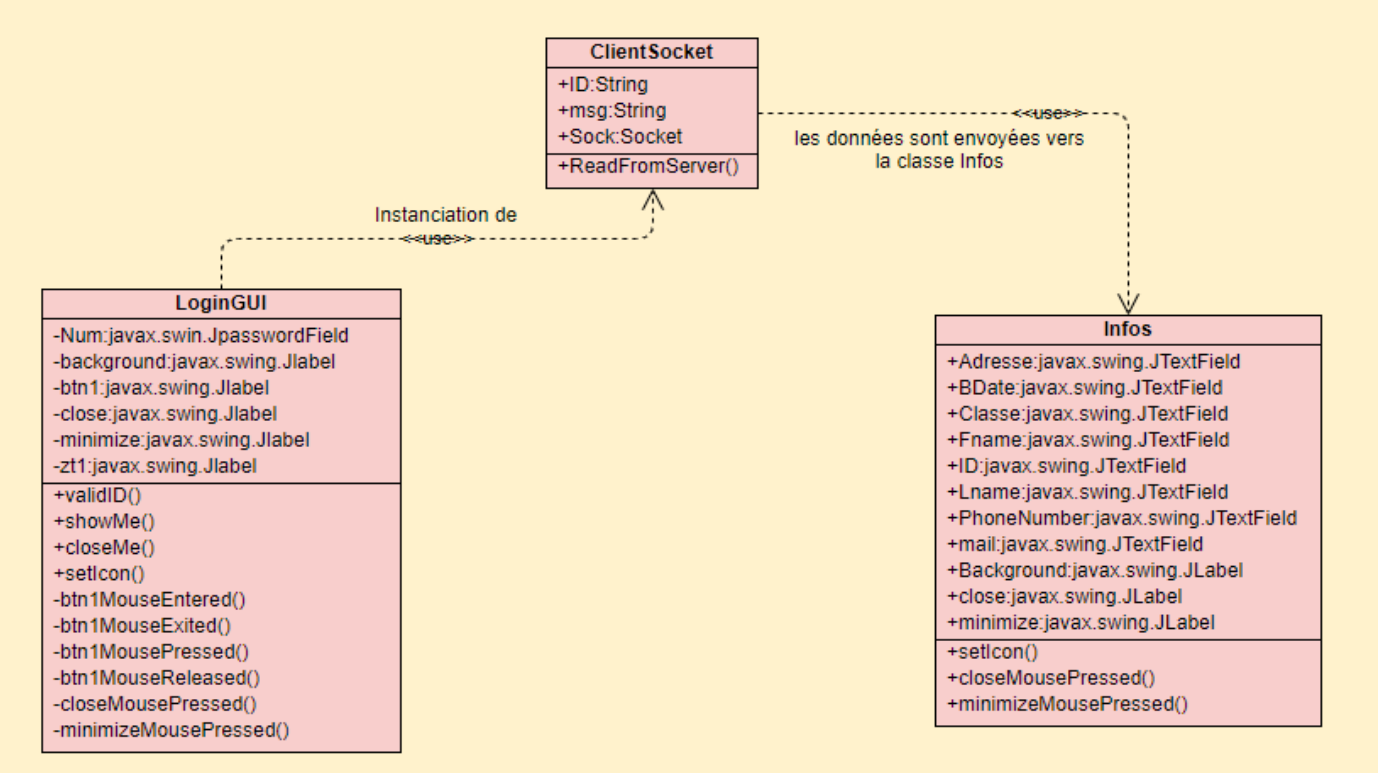
Ceci est le diagramme de classes de la partie serveur de l'application



-Partie Client :

Diagramme de classe :

Ceci est le diagramme de classe de la partie client :



-Côté Serveur :

On présentera dans cette partie la partie serveur de l'application

Features:

Multi Connexions:

L'application peut supporter plusieurs connexions en même temps grâce à l'utilisation des threads, chaque connexion sera traitée de façon indépendante dans une thread à part, d'où on peut accepter un grand nombre de connexions simultanées.

```java
//On lance notre serveur
public void open(){

    //Toujours dans un thread à part vu qu'il est dans une boucle infinie
    Thread t = new Thread(new Runnable(){
        public void run(){
            while(isRunning == true){

                try {
                    //On attend une connexion d'un client
                    Socket client = server.accept();

                    //Une fois reçue, on la traite dans un thread séparé
                    System.out.println("Connexion cliente reçue.");
                    Thread t = new Thread(new ClientProcessor(client));
                    t.start();

                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            try {
                server.close();
            } catch (IOException e) {
                e.printStackTrace();
```

Logging:

Nous avons assuré que les informations des clients (IP , Port , Commande Reçue) seront écrites au console , et l'administrateur peut choisir de les enregistrer dans un fichier ou non (Par exemple pour un système Linux on ajoute cette redirection de stdout a la fin de la commande : 1>>logs.txt)

```java
//On affiche quelques infos, pour le logging
String debug = "";
debug = "Thread : " + Thread.currentThread().getName() + ". ";
debug += "Demande de l'adresse : " + remote.getAddress().getHostAddress() +".";
debug += " Sur le port : " + remote.getPort() + ".\n";
debug += "\t → Commande reçue : " + response + "\n";
System.err.println("\n" + debug);
```

```
^Cubumtu@ip-172-31-80-220:~/Projet/usr/lib/jvm/java-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 -cp .:lib/mariadb-java-client-2.6.0.jar MainServ
Serveur initialisé.
Connexion cliente reçue.
Lancement du traitement de la connexion cliente

Thread : Thread-1. Demande de l'adresse : 102.175.195.74. Sur le port : 61415.
        → Commande reçue : 0002


Connecting database ...
Database connected!
```

La Sécurité :

Afin d'assurer la sécurité de la base de données on a utilisé les "Prepared Statements" dans l'exécution de la requête SQL pour éviter les attaques de SQL injection, et le SGBD MySQL n'est pas exposé .

```
String finalresult="";
String sql = "select * from students where id=?";
PreparedStatement preparedStatement = connection.prepareStatement(sql);
id=id.replace("\n", "");
preparedStatement.setString(1, id);
ResultSet result = preparedStatement.executeQuery();
while(result.next()){

finalresult=id+", "+result.getString(3)+" , "+result.getString(4)+", "+result.getString(5)+", "+result.getString(6)+" , "+result.getString(7)+" , "+result.getString(

}
connection.close();
return finalresult;
}
```

La configuration de l'application de la partie serveur dans le VPS a été faite de façon sécurisée, on a employé la containerisation en utilisant le logiciel docker pour isoler l'application et ses dépendances

```
ubuntu@ip-172-31-95-109:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS                    NAMES
4a9c945ffb26   ubuntu    "bash"    33 hours ago     Up 33 hours     0.0.0.0:1234→1234/tcp    awesome_rosalind
ubuntu@ip-172-31-95-109:~$ sudo docker exec -it 4a9 /bin/bash
root@4a9c945ffb26:/# ls data
 ClientProcessor.class   Database.class   MainServ.class  'StudentServer$1.class'   StudentServer.java   hs_err_pid7619.log   start.sh
 ClientProcessor.java    Database.java    MainServ.java    StudentServer.class      hs_err_pid7576.log   lib
root@4a9c945ffb26:/# exit
exit
ubuntu@ip-172-31-95-109:~$
```

La base de données :

L'utilisation de MySQL nous a permet de gérer la base de données aisément, on a d'abord créé un nouvel utilisateur MySQL conçu pour ce projet :

> CREATE USER "ahmed"@"localhost" IDENTIFIED BY "123456"

Après on donne les privilèges au nouvel utilisateur à partir de l'utilisateur root pour créer et modifier les bases de données :

> GRANT ALL PRIVILEGES ON *.* TO "kahla"@"localhost"

Maintenant on crée une nouvelle base nommée "etudiants" :

> CREATE DATABASE etudiants ;

Après on crée la table "students" contenant les informations des étudiants :

```
mysql> CREATE TABLE students (cle smallint unsigned not null auto_increment,id varchar(100) not null,prenom varchar(200) not null,nom varchar(200) not null,date varch
ar(200) not null,class varchar(200) not null,tel varchar(200) not null,adresse varchar(200) not null,email varchar(200) not null,constraint pk primary key(cle));
```

On insère maintenant les informations des étudiants :

```
mysql> INSERT INTO students (cle,id,prenom,nom,date,class,tel,adresse,email) VALUES (null,"0001","Ahmed","Belkahla","13/09/1999","RT 2","95460918","Kairouan,Tunisie",
"ahmed-belkahla@ieee.org"),(null,"0002","Montasar","Amri","24/02/1999","RT 2","22543184","Ariana,Tunisie","monta99@gmail.com"),(null,"0003","Med Ali","Zaabi","22/06/1
999","RT 2","23548197","Mednine,Tunisie","medalizaabi@gmail.com"),(null,"0004","Brahim","Ayadhi","18/07/1997","RT 4","5517964","Mannouba,Tunisie","brahimtrgfx@gmail.c
om");
```

D'où on obtient :

```
mysql> use etudiants
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM students ;
+-----+------+----------+----------+------------+-------+----------+-----------------+-------------------------+
| cle | id   | prenom   | nom      | date       | class | tel      | adresse         | email                   |
+-----+------+----------+----------+------------+-------+----------+-----------------+-------------------------+
|   1 | 0001 | Ahmed    | Belkahla | 13/09/1999 | RT 2  | 95460918 | Kairouan,Tunisie | ahmed-belkahla@ieee.org |
|   2 | 0002 | Montasar | Amri     | 24/02/1999 | RT 2  | 22543184 | Ariana,Tunisie  | monta99@gmail.com       |
|   3 | 0003 | Med Ali  | Zaabi    | 22/06/1999 | RT 2  | 23548197 | Mednine,Tunisie | medalizaabi@gmail.com   |
|   4 | 0004 | Brahim   | Ayadhi   | 18/07/1997 | RT 4  | 5517964  | Mannouba,Tunisie | brahimtrgfx@gmail.com   |
+-----+------+----------+----------+------------+-------+----------+-----------------+-------------------------+
4 rows in set (0.00 sec)
```

Côté Client :

Dans cette partie nous allons présenter la côté client qui aura la tâche de se connecter au serveur.
Le processus CLIENT établit une connexion au serveur à travers Les Sockets.

Création d'un Socket et Connexion au serveur :

La classe ClientSocket est responsable d'établir la connexion au serveur, elle prend en paramètres l'ID récupéré à l'interface de l'application.
Au Début nous devons créer notre socket en donnant comme argument l'adresse IP de notre serveur (VPS) et le numéro de Port.

```
String ip="52.71.185.177";
int port=1234;
Socket sock=new Socket(ip,port);
```

Envoi des données :

Maintenant pour communiquer avec notre serveur nous commençons par la déclaration et l'initialisation du flux de transport des données(BufferedOutputStream), puis l'envoie des informations à travers ce flux.

```
BufferedOutputStream bos = new BufferedOutputStream(sock.getOutputStream());
bos.write(ID.getBytes());
bos.flush();
```

Réception des données :

Pour recevoir la réponse du serveur on doit déclarer le flux entrant (BufferedInputStream) puis tout ce qui nous reste est de lire les données sous forme de (String) reçue depuis la fonction ReadFromServer( ). Cette chaine de caractères représente les données de l'Etudiant (ID, nom, prénom …) séparées par des virgules. Enfin nous envoyons ces données à l'interface graphique (Infos) pour les traiter et les afficher.

```
public String ReadFromServer(){
    String response="";
    try{
        BufferedInputStream bis = new BufferedInputStream(sock.getInputStream());
        int stream=0;
        byte[] b = new byte[4096];
        stream = bis.read(b);
        response = new String(b, 0, stream);
        bis.close();}
    catch(Exception e){
        msg="Connection Error , Try again please !";
        }
    return response;
}
```

```
String response=ReadFromServer();
final String copieStr=new String(response);
java.awt.EventQueue.invokeLater
   (new Runnable()
          {public void run() { new Infos(copieStr,in).setVisible(true);}
          }
   );
this.in.hideMe();//cacher la 1ere fénêtre
```

Fermeture du Socket :

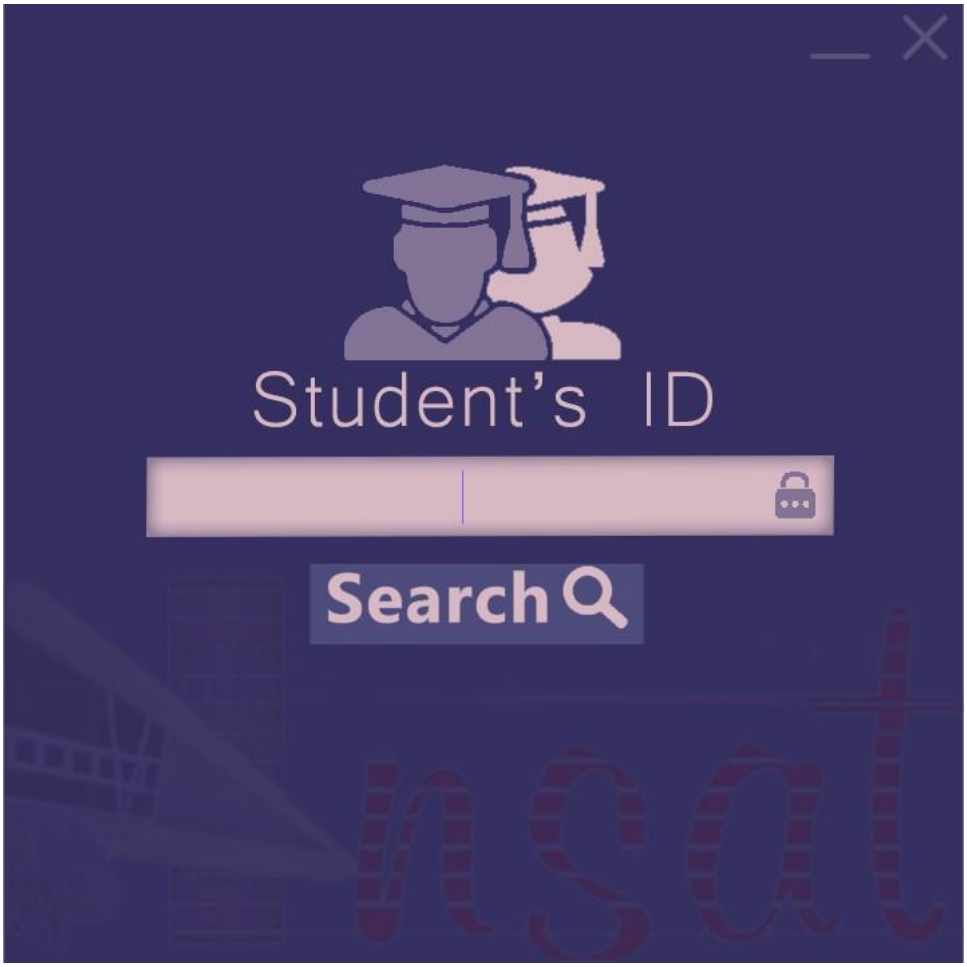
On finit par la fermeture de nos canaux de transport (Buffered(Input/Output)Stream) de données et la fermeture de notre Socket.

```
bis.close();
bos.close();
sock.close();
```

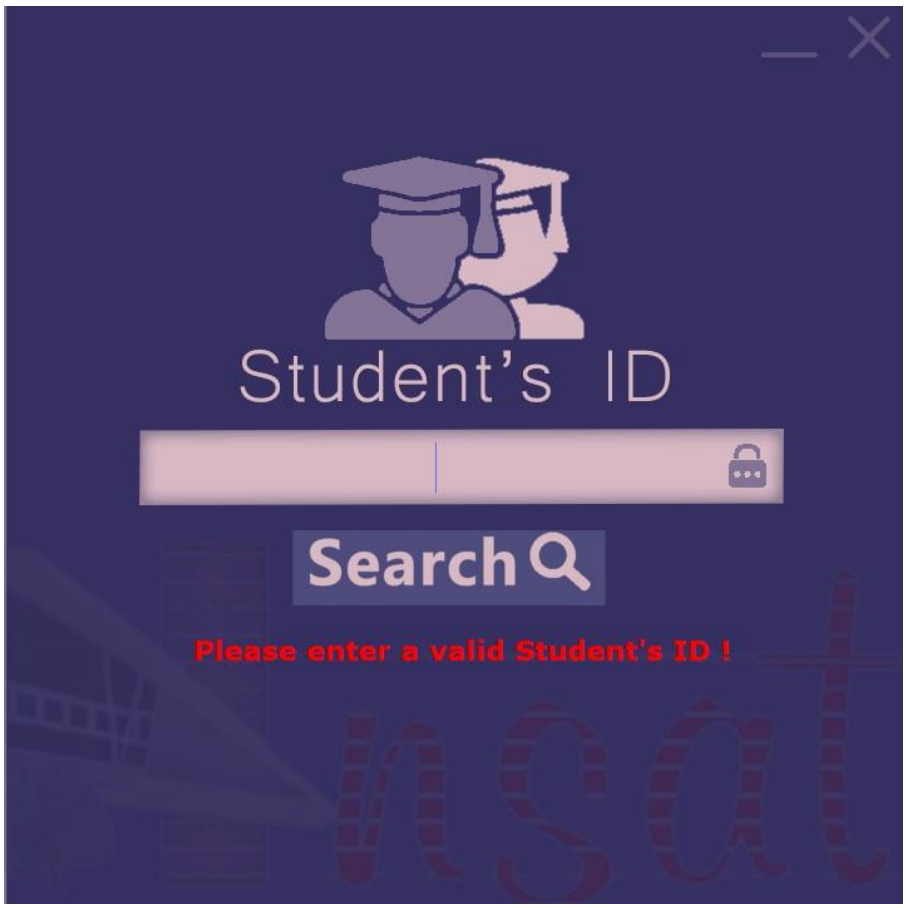
Interface graphique :

Pour bien présenter notre application on a utilisé la bibliothèque **Swing.**
On a créé les “backgrounds” de l’application et les buttons avec le logiciel Photoshop.
Puis on a attribué à chaque bouton sa fonctionnalité nécessaire (MousePressed, MouseEntered …).
On a utilisé deux interfaces graphiques pour les deux classes LoginGUI et Infos :

1- Ici le Client(l’Etudiant) va saisir son ID (dans l’application allant de 0001 à 0004) et il doit cliquer sur le bouton “Search” :

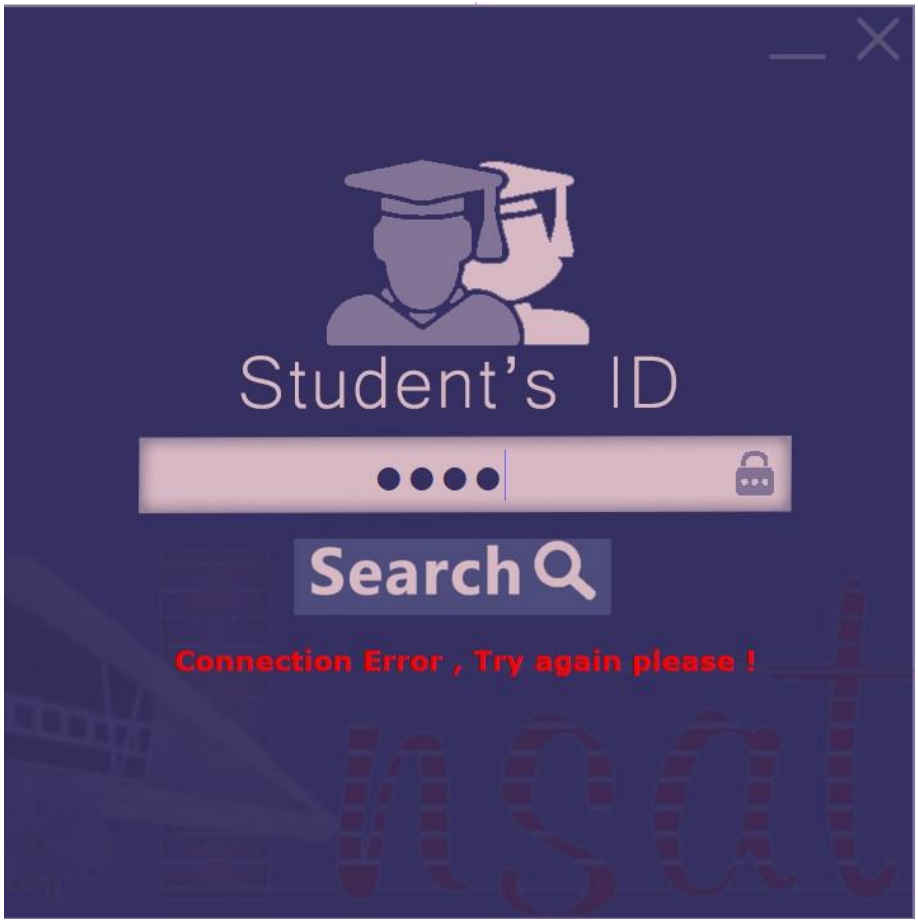


2- Si l'ID n'existe pas dans la base de données il reçoit un msg d'erreur comme suit :



3- Si l'ID existe dans la base de données une instance de ClientSocket se crée et celle-ci va faire l'appel à la classe Infos qui reçoit les informations depuis le serveur , d'où une nouvelle fenêtre s'affiche contenant toutes les données nécessaires organisées comme suit :

4- En cas de problème de connexion, une exception génère un message d'erreur qui s'affiche comme suit :



Prochaines étapes :

Afin d'améliorer l'application, on compte ajouter des fonctionnalités pour ajouter ou supprimer des étudiants de façon sécurisé avec une interface d'authentification, et d'installer la partie serveur dans un environnement SSL ou crypter les informations des étudiants pour assurer leur sécurité lors de leur transfert.