

# Esalaf

## (Project)

**Encadré par :**

***Prof. El Mokhtar EN-NAIMI***

***Réalisé par :***

***Chibani Fahd***

# SOMMAIRE

SOMMAIRE.....	2
INTRODUCTION.....	3
LE PROCESSUS DU DÉVELOPPEMENT ET LES OPTIONS DÉVELOPPÉES .....	4
<b>1. Login interface</b> .....	4
<b>2) Main interface</b> .....	5
<b>3) Interface Client</b> .....	5
<b>4) Interface Produit</b> .....	6
<b>5) Interface Crédit</b> .....	6
<b>6) Interface Commande</b> .....	7
<b>Les fichier de travail</b> .....	7
• <b>Client.java</b> .....	8
• <b>ClientController</b> .....	9

# INTRODUCTION

L'objectif principal de ce projet est de maîtriser les interfaces graphiques et la programmation orientée objet en Java par la mise en place d'une application desktop Java, basée sur JDBC et JavaFX, l'application doit gérer les crédits, les commandes, les produits, les clients, ainsi elle donne un tableau de bord générique à l'aide de l'IDE (Integrated Development Environment) IntelliJ IDEA qu'il est un environnement de développement intégré destiné au développement de logiciels informatiques reposant sur la technologie Java. .

L'application desktop Java développée s'appelle Esalaf. Cette application se compose de six interfaces :

- Login Interface
- Main Interface
- Interface Client
- Interface Crédit
- Interface Produit
- Interface Commande

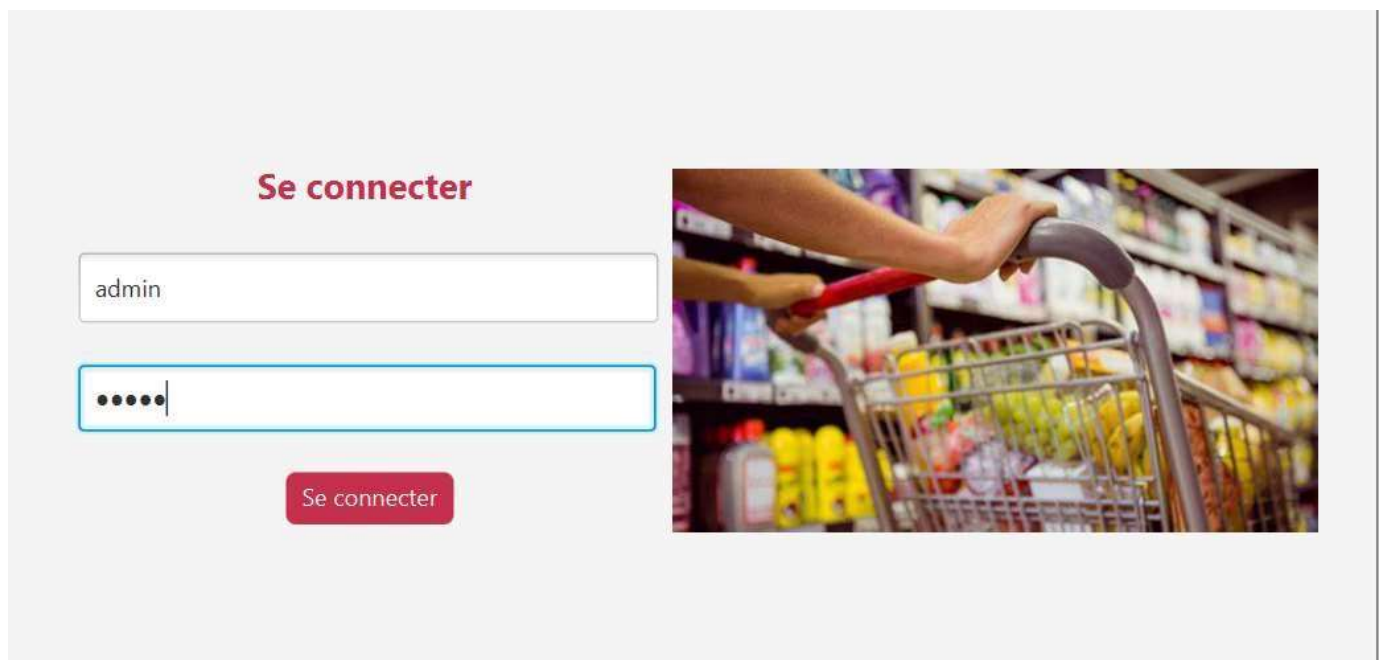
Outils : IntelliJ IDEA, JavaFX, JDBC, MySQL, Java, SceneBuilder.

# LE PROCESSUS DU DÉVELOPPEMENT ET LES OPTIONS DÉVELOPPÉES

Afin de développer notre application desktop, on avait besoin de créer plusieurs Interface comme déjà indiquer, qui sont les différents interfaces visualiser par l'utilisateur et qui le permettre de stocker les différents information (Client, Crédit, Produit, Commande), et ajouter ensuite des méthodes permettant la navigation d'une interface vers une autre.

Les interfaces créées sont les suivants :

## 1. Login interface :



The image displays a login interface on the left and a photograph of a shopping cart on the right. The login interface features a light gray background. At the top, the text "Se connecter" is written in red. Below this, there are two input fields: the first contains the text "admin", and the second contains five dots, indicating a password field. A red button with the text "Se connecter" is positioned below the input fields. The photograph on the right shows a person's hand pushing a metal shopping cart filled with various groceries, including bottles of cleaning products and bags of food, through a supermarket aisle.

## 2) Main interface :



## 3) Interface Client :

The 'Client' interface in the Esalaf application includes a sidebar with the same four buttons as the main interface. The main area is titled 'Client' and contains a form with four input fields for client information: 'Id' (value: 1), 'Nom Complet' (value: Chibani Fahd), 'CIN' (value: GI11014), and 'Telephone' (value: 0695200202). Below the form is a table with the following data:

Id	Nom Complet	CIN	Telephone
1	Chibani Fahd	GI11014	695200202

At the bottom of the interface are three buttons: 'Modifier', 'Ajouter', and 'Supprimer'.

#### 4) Interface Produit :

Client

Produit

Crédit

Commande

Esalaf

Produit

ID

Brande

Produit

Prix

Id_Produit	Brande	Produit	Prix
1	Samsung	S23	12000.0

Ajouter

Modifier

Supprimer

#### 5) Interface Crédit :

Client

Produit

Crédit

Commande

Esalaf

Crédit

1

Chibani Fahd

100.0

Id_Crédit	Nom et Prénom	Montant Crédit
1	Chibani Fahd	100.0

Ajouter

Modifier

Supprimer

## 6) Interface Commande :

**Commande**

Client:

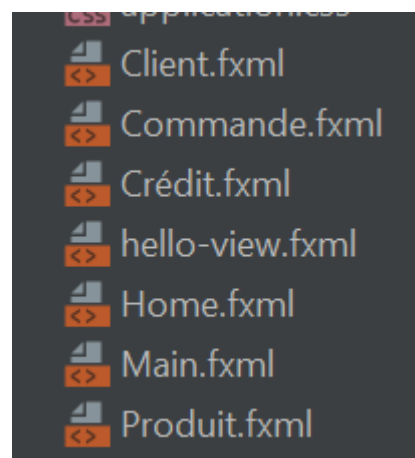
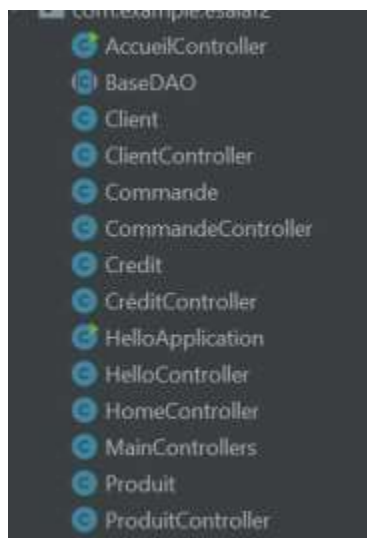
Produit:

Id	Nom_Complet	Brande_Produit	Nom_Produit	Num_télépho...	Date_Commande
8	Chibani Fahd	Apple	Iphone 13	0678890987	2023-04-05
9	Chibani Fahd	Apple	Iphone 12	0678890987	2023-04-05

Ajouter Modifier Supprimer

## Les fichier de travail :

Chaque Interface se constitue de deux classe, une classe contient le constructeur pour initialiser les attributs d'interface et une classe Controller qui gère la connexion avec la base donne et les méthode CRUD est un fichier.fxml qui contient le design de chaque interfaces .



Si en prend l'interface Client par exemple :

- **Client.java :**

```
• public class Client {  
    private int IdC ;  
    private String NomprenomC ;  
    private String Cni ;  
    private String Telephone ;  
    public Client(int IdC, String NomprenomC, String Cni, String  
Telephone) {  
        this.IdC = IdC;  
        this.NomprenomC= NomprenomC;  
        this.Telephone = Telephone;  
        this.Cni = Cni;  
  
    }  
  
    public int getIdC() {  
        return IdC;  
    }  
    public void setidC(int IdC) {  
        this.IdC = IdC;  
    }  
    public String getNomprenomC() {  
        return NomprenomC;  
    }  
    public void setNomprenomC(String NomprenomC) {  
        this.NomprenomC = NomprenomC;  
    }  
    public String getTelephone() {  
        return Telephone;  
    }  
    public void setTelephone(String Telephone) {  
        this.Telephone = Telephone;  
    }  
    public String getCni() {  
        return Cni;  
    }  
    public void setCni(String Cni) {  
        this.Cni = Cni;  
    }  
}
```



- **ClientController :**
- Les méthodes CRUD :

```

- public ObservableList<Client> getclientsListe() {
    ObservableList<Client> ClientsList =
    FXCollections.observableArrayList();
    Connection conn = getConnection();
    String req = "SELECT * FROM client";
    Statement st;
    ResultSet rs;

    try{
        st = conn.createStatement();
        rs = st.executeQuery(req);
        Client clients;
        while(rs.next()){
            clients = new Client(rs.getInt("IdC")
                                ,rs.getString("NomprenomC")
                                ,rs.getString("Cni")
                                ,rs.getString("Telephone"));
            ClientsList.add(clients);
        }
    }catch (Exception ex){
        ex.printStackTrace();
    }
    return ClientsList;
}

public void showclients(){
    ObservableList<Client> Liste = getclientsListe();
    colid.setCellValueFactory(new
    PropertyValueFactory<>("IdC"));
    colnom.setCellValueFactory(new
    PropertyValueFactory<Client,String>("NomprenomC"));
    colcni.setCellValueFactory(new
    PropertyValueFactory<Client,String>("Cni"));
    coltel.setCellValueFactory(new
    PropertyValueFactory<Client,String>("Telephone"));
    tvBox.setItems(Liste);
}

public void onAjouterButtClick() {
    String c="NULL";
    String req ="INSERT INTO client VALUES (" + c
    + "," + ttfnom.getText() + "," + ttfcni.getText() + "," + ttfnum.getTe
    xt() + ") ";
    executeQuery(req);
}

```

```

        showclients();
    }

    public void onmodButtClick() {
        String req = "UPDATE client SET NomprenomC = '" +
            tfnom.getText() + "', Cni = '" + tfcni.getText()
                + "', Telephone= '" + tfnum.getText() + "' WHERE idC
            = '" + tfid.getText() + "'";
        executeQuery(req);
        showclients();
    }

    public void onsuppButtClick() {
        String req = "DELETE FROM client WHERE IdC =" +
            tfid.getText() + "'";
        executeQuery(req);
        showclients();
    }

    public void executeQuery(String req) {
        Connection conn = getConnection();
        Statement st;
        try{
            st = conn.createStatement();
            st.executeUpdate(req);
        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
}

```

C'est méthode CRUD nous aides a afficher, ajouter, modifier et supprimer les élément des tableaux.

- *Méthode de connexion au data base :*

```
- public Connection getConnection() {  
    Connection conn;  
    try {  
        conn =  
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/esalaf", "root", "");  
        return conn;  
    } catch (Exception ex) {  
        System.out.println("Error " + ex.getMessage());  
        return null;  
    }  
}
```

