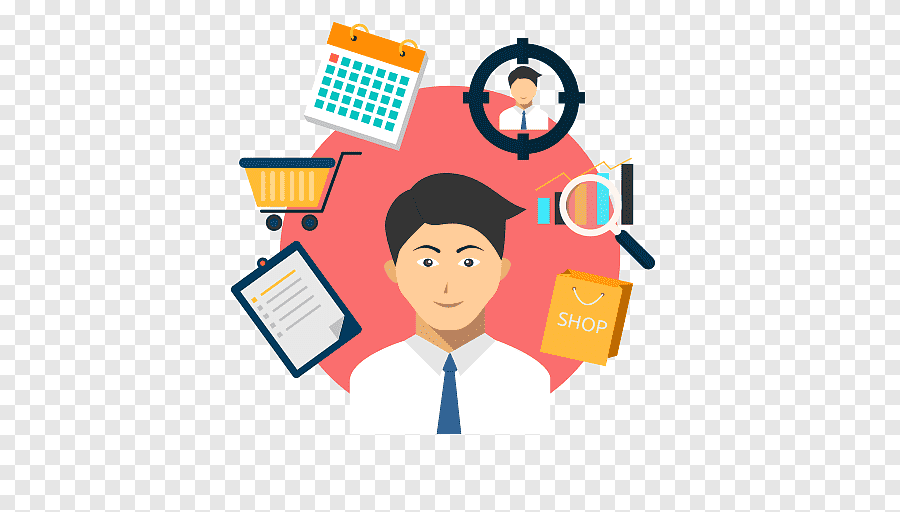
**RAPPORT : PROJET ESSALAF**

****

**NOM ET PRENOM : EL GHAZI LOUBNA**

**GROUPE : 1**

**CNE : H139225843**

**ENCADRE PAR LES PROFS. : EL MOKHTAR ENAIMI**

**: LOTFI ELACHAK**

**REMERCIMENTS :**

**Je tiens à exprimer ma sincère gratitude envers mes deux professeurs encadrants, pour leur soutien et leur expertise tout au long de la réalisation de ce projet d'application JavaFX. Leurs connaissances approfondies de ce domaine ont été d'une grande aide pour moi dans la résolution de problèmes techniques, la prise de décisions et l'amélioration de la qualité du produit final.**

**Plan**

**1-Composants du projet & Outils utilisés**

**2-Difficultés rencontrées**

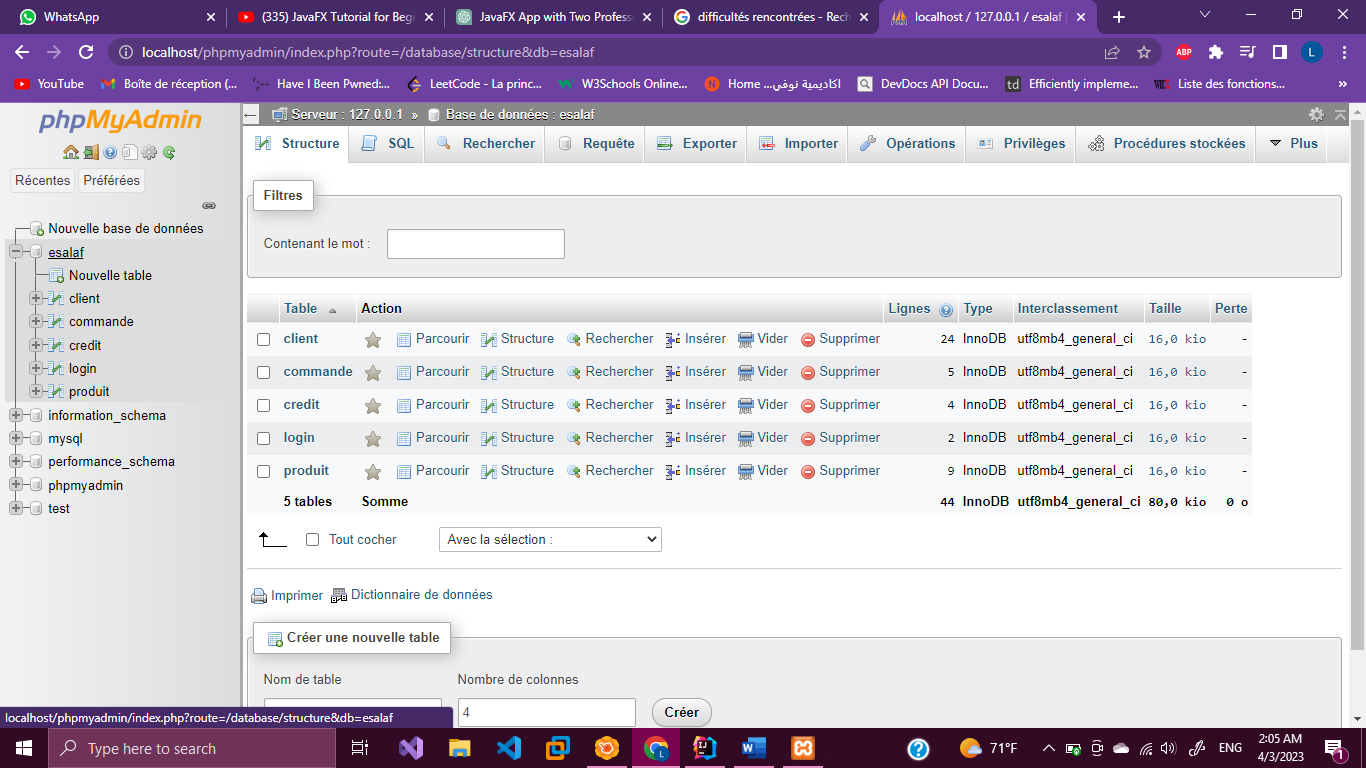
**3-Conclusion**

**1-Composants du projet & Outils utilisés**

**IDE utilisé :** IntelliJ IDEA

**Outil de base de données** : PhpMyadmin (XAMP)

**Scene builder , JDBC,JAVAFX**

Notre base de données (esalaf):

Contient toutes les tables necessaires (Client,produit,credit,commande ,user)

**Les classes du projet :**

**Classes Client, Produit,Commande ,Credit : Pour** representer les entités dans la base de données,.Elles contiennent des getters, setters, et meme constructeurs surchargés .

**Les objets d accès aux données DAO(**DAOoutils(Contrôle les interfaces login et inscription)..…..)

Dont les méthodes Crud sont implementés pour interagir avec la base de données .

**Les classesControlleurs : Pour chaque classe**:

Cordonne les Dao et les interfaces utilisateurs(fichiers fxml).

Et c’est dans ces classes qu’on implémente les méthodes qui donnent actions aux boutons, tableaux …

**Les fichiers fxml**

Representent les interfaces utilisateurs et gérés par Scene builder pour une performance facile .

Note :

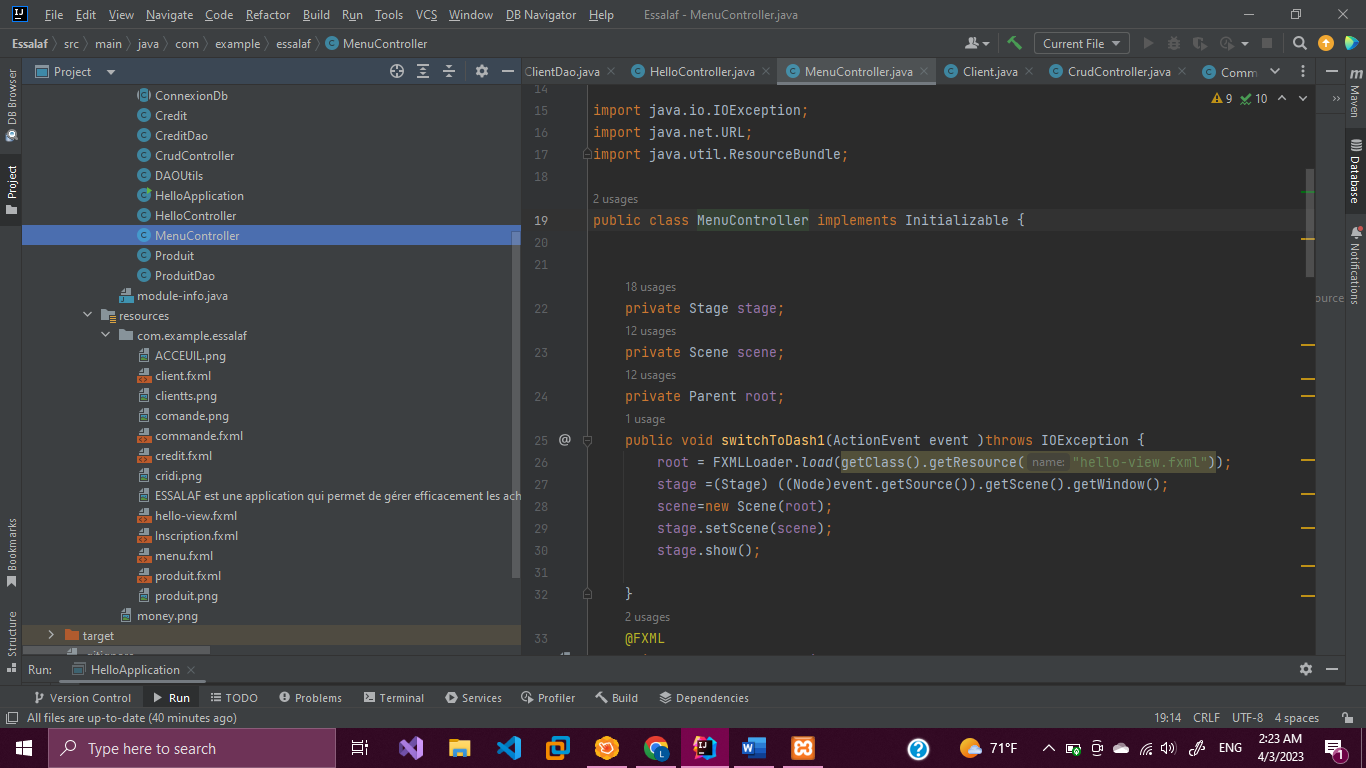
Y’en a d’autres plusieurs classes comme CrudController qui controles les actions du menu de l’application,

DbConnexion qui definit la connexion avec la base de données esalaf……

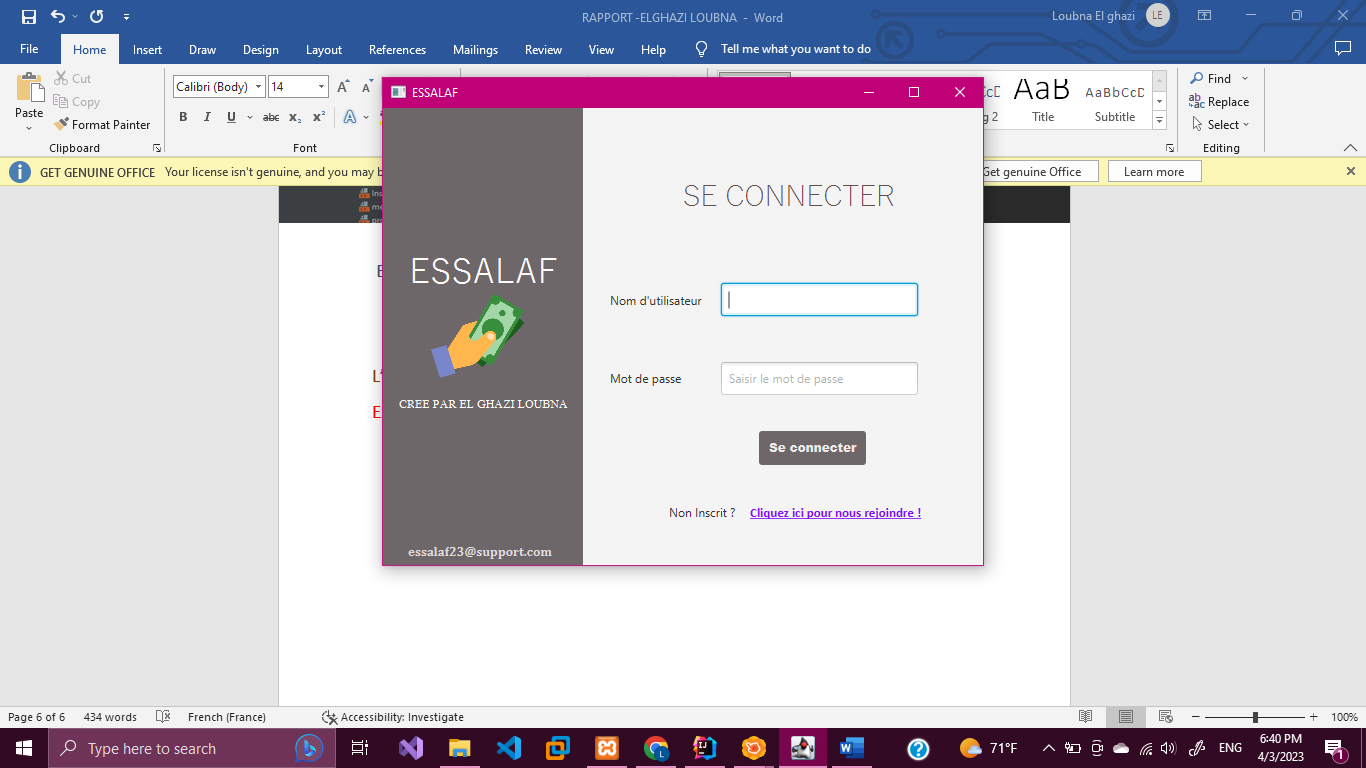
Exemples de classe  (CrudController):

package com.example.essalaf;  
  
import javafx.event.ActionEvent;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Node;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
  
public class CrudController {  
  
 private Stage stage;  
 private Scene scene;  
 private Parent root;  
 public void switchToMenu(ActionEvent event )throws IOException {  
 root = FXMLLoader.*load*(getClass().getResource("menu.fxml"));  
 stage =(Stage) ((Node)event.getSource()).getScene().getWindow();  
 scene=new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
  
 }  
  
 public void switchToDash2(ActionEvent event )throws IOException {  
 root = FXMLLoader.*load*(getClass().getResource("hello-view.fxml"));  
 stage =(Stage) ((Node)event.getSource()).getScene().getWindow();  
 scene=new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
  
 }  
 public void switchToProduct(ActionEvent event )throws IOException {  
 root = FXMLLoader.*load*(getClass().getResource("produit.fxml"));  
 stage =(Stage) ((Node)event.getSource()).getScene().getWindow();  
 scene=new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
  
 }  
 public void switchToCommande(ActionEvent event )throws IOException {  
 root = FXMLLoader.*load*(getClass().getResource("commande.fxml"));  
 stage =(Stage) ((Node)event.getSource()).getScene().getWindow();  
 scene=new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
  
 }  
  
 public void switchToClient(ActionEvent event )throws IOException {  
 root = FXMLLoader.*load*(getClass().getResource("client.fxml"));  
 stage =(Stage) ((Node)event.getSource()).getScene().getWindow();  
 scene=new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
  
 }  
 public void switchToCredit(ActionEvent event)throws IOException{  
  
 root = FXMLLoader.*load*(getClass().getResource("credit.fxml"));  
 stage =(Stage) ((Node)event.getSource()).getScene().getWindow();  
 scene=new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
 }  
  
  
  
  
}

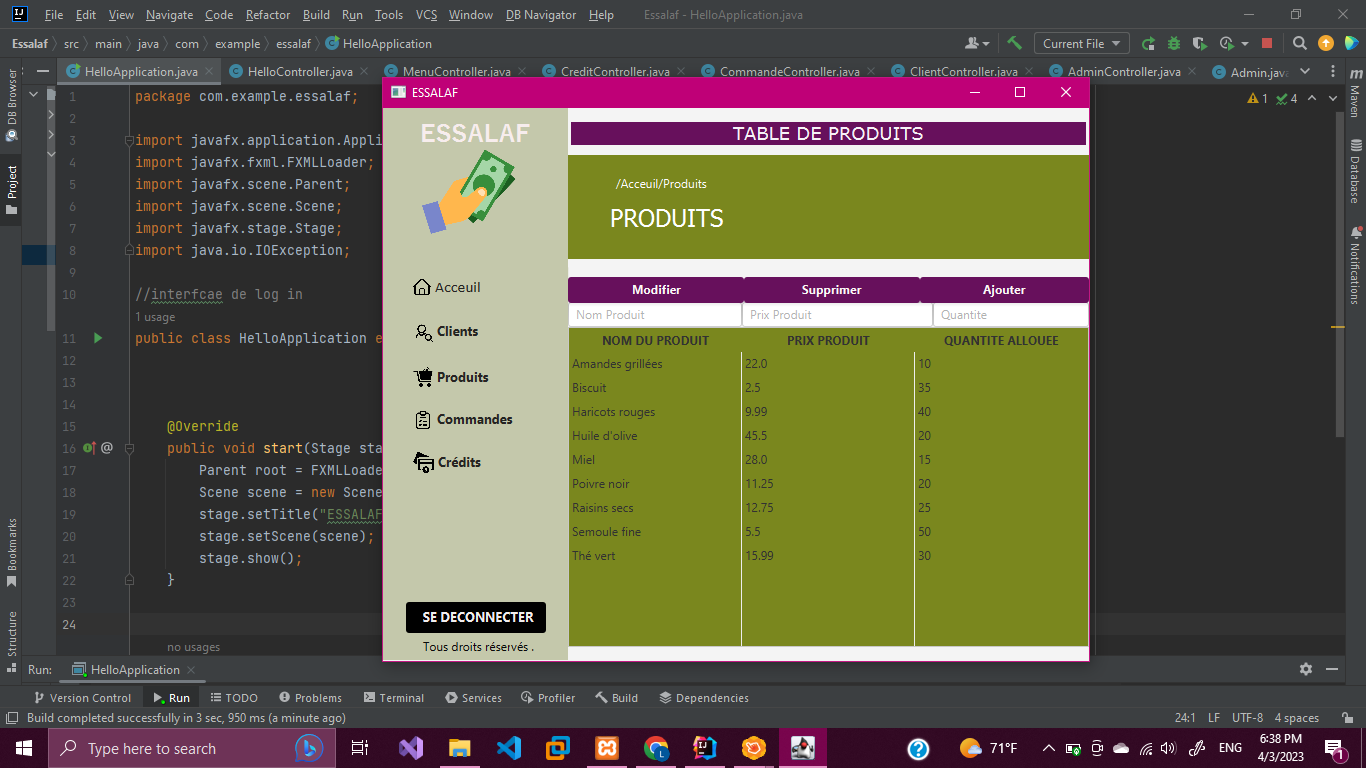
Les images dans le projet ne sont jamais lus si ne sont pas dans Ressources et recuperes au de la :

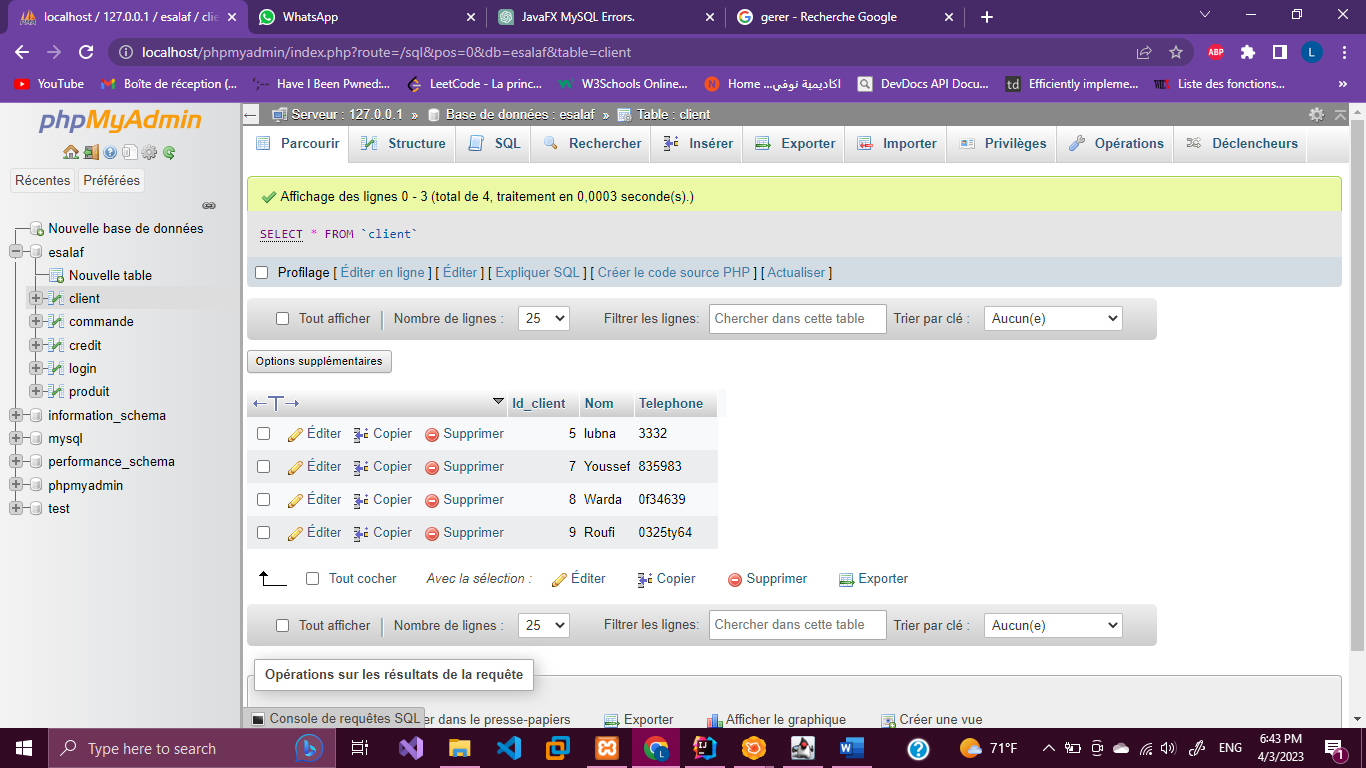


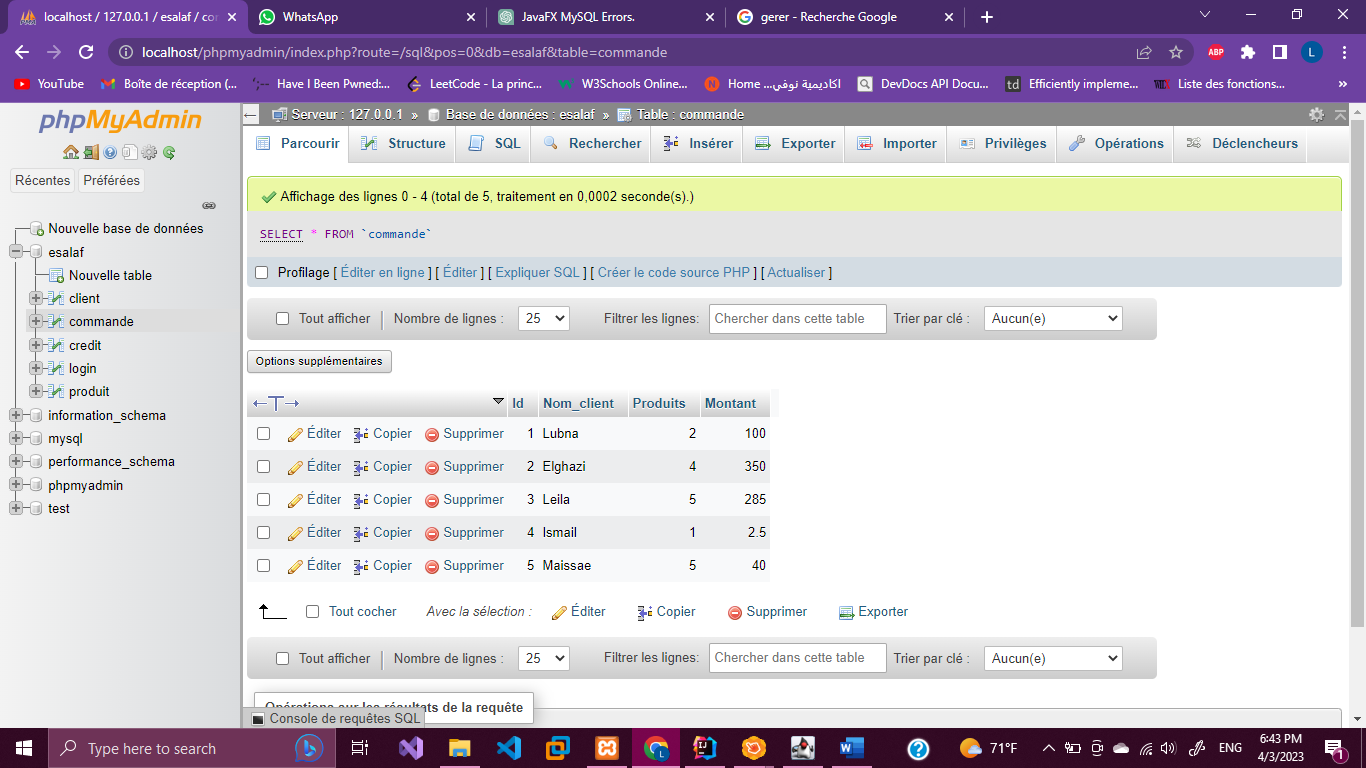
Exemple d’Interface de connexion :

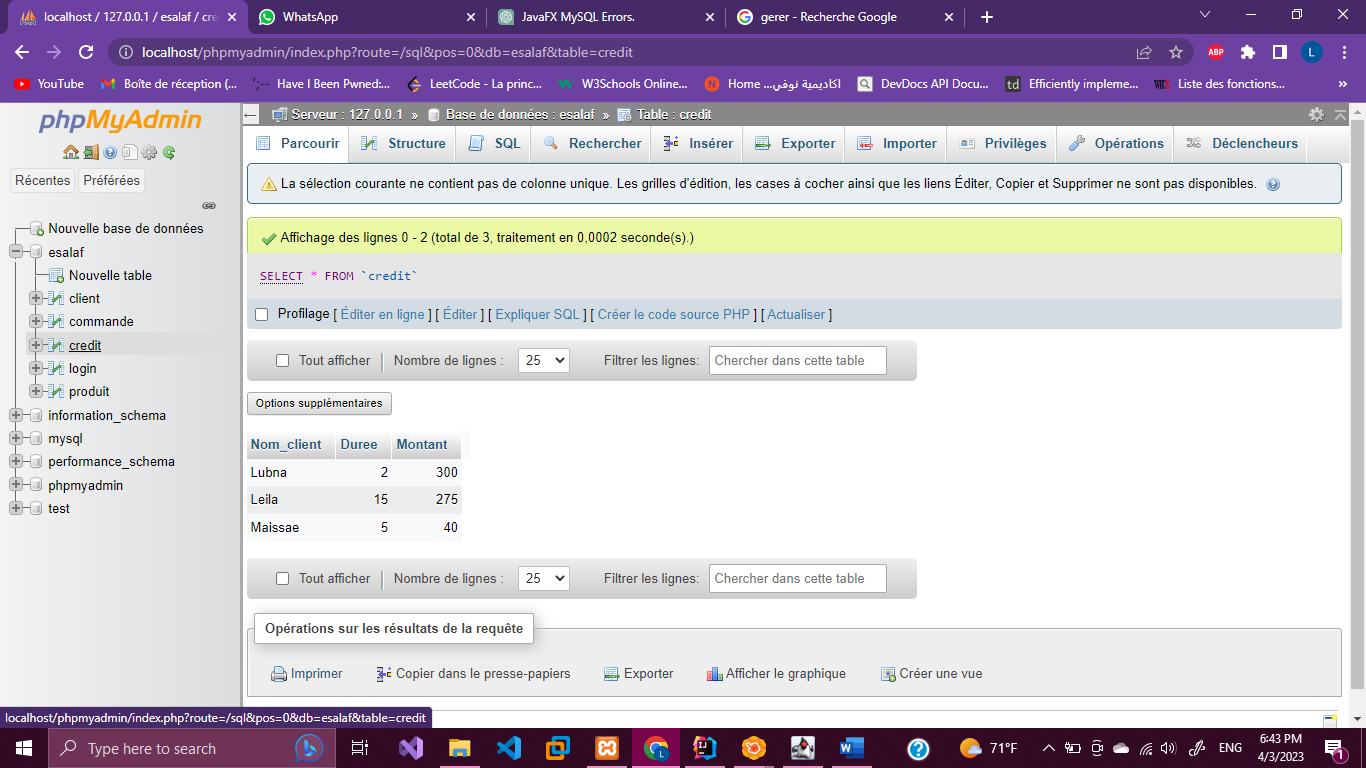


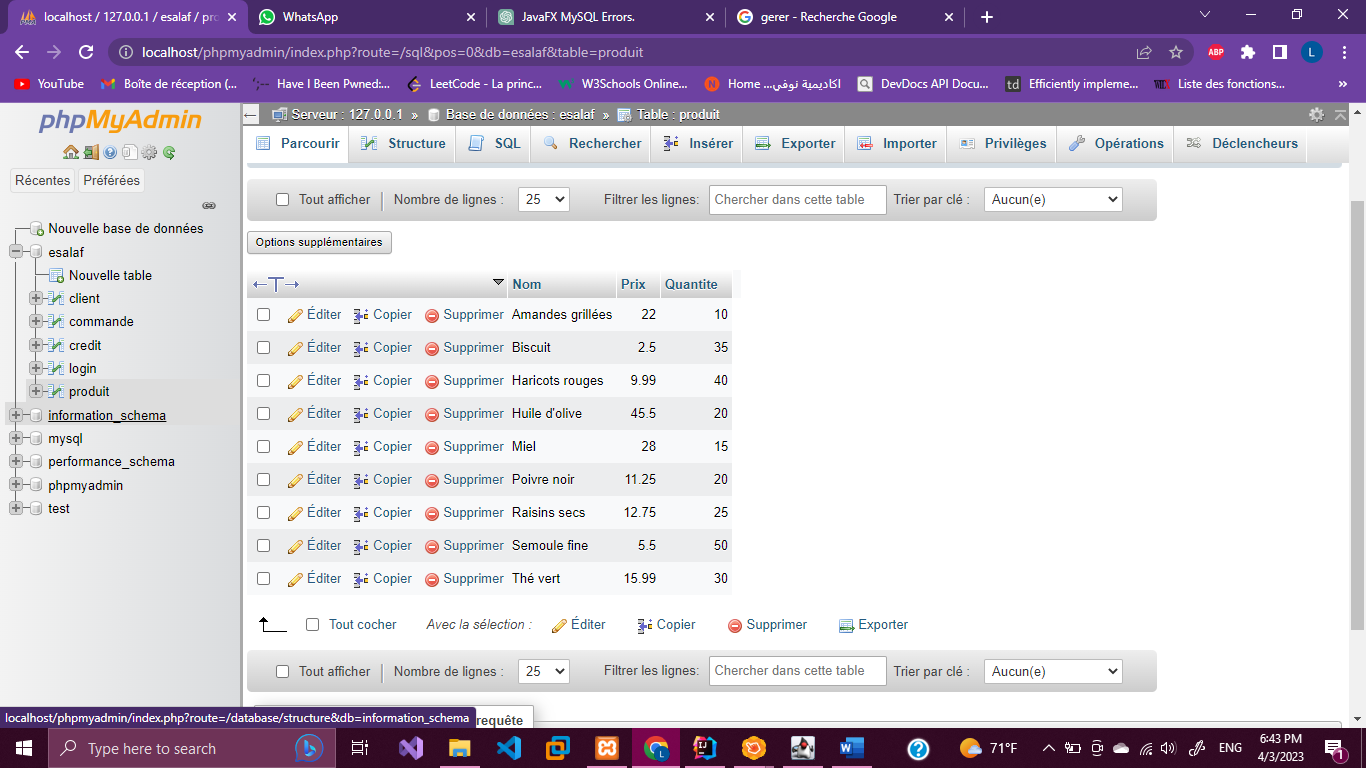
L’APPLICATION :

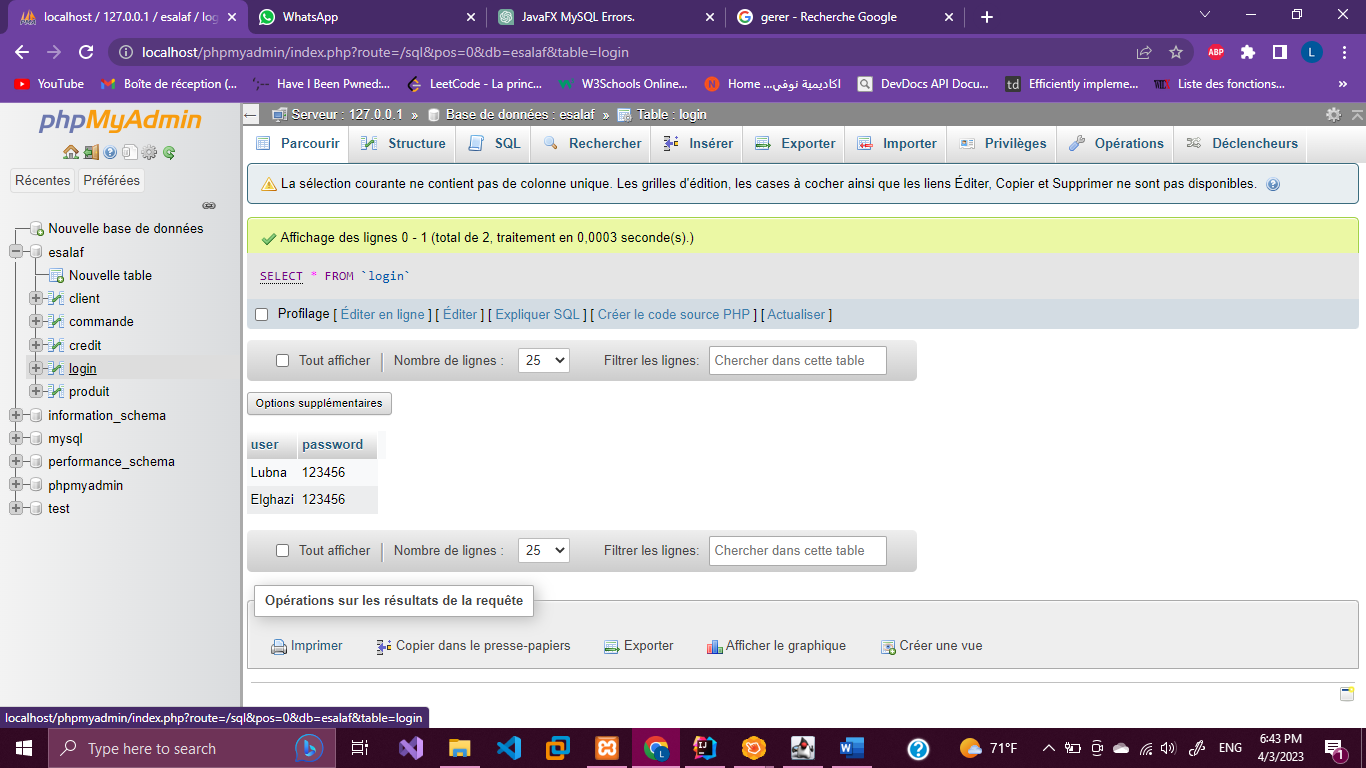
Exemple : Interface de produits :

Les tables de bases de donnes essalaf :





****

****

**Les methodes CRUD :**

**Pour toutes les classes (Client,Produit,Commande,Credit)**

**Les méthodes Crud sont les mêmes, On change juste les informations liées aux requêtes pour chacune des classe .**

**J’ai utilisé la méthode suivante : Exemple classe Produit(Constructeur, et getters et setters)+ CLASSE PODUITCONTROLLER(J appelle la connexionDb pour se connecter a la base de données + j’implémente les méthodes CRUD et je contrôle chaque action dedans.)**

**Exemple du code de la classe ProduitController(Et de la même façon j’ai travaille pour toutes les autres classes )**

public class ProduitController implements Initializable {  
  
  
 Produit produit;  
 private Stage stage;  
 private Scene scene;  
 private Parent root;  
 @FXML  
 private TextField Tnom;  
  
 @FXML  
 private TextField Tprod;  
  
 @FXML  
 private TextField Tqu;  
  
 @FXML  
 private TableView<Produit> tab;  
  
 @FXML  
 private TableColumn<Produit, String> colN;  
  
 @FXML  
 private TableColumn<Produit, Float> colP;  
  
 @FXML  
 private TableColumn<Produit, Integer> colQ;  
  
  
 public Connection getConnection() {  
 Connection conn;  
 try {  
 conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/esalaf", "root", "");  
 return conn;  
 } catch (Exception ex) {  
 System.*out*.println("Error " + ex.getMessage());  
 return null;  
 }  
 }  
  
 public ObservableList<Produit> getproduitsListe() {  
 ObservableList<Produit> ProduitsList = FXCollections.*observableArrayList*();  
 Connection conn = getConnection();  
 String req = "SELECT \* FROM produit";  
 Statement st;  
 ResultSet rs;  
  
 try {  
 st = conn.createStatement();  
 rs = st.executeQuery(req);  
 Produit produits;  
 while (rs.next()) {  
 produits = new Produit(rs.getString("Nom")  
 , rs.getFloat("Prix")  
 , rs.getInt("Quantite"));  
 ProduitsList.add(produits);  
 }  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 return ProduitsList;  
 }

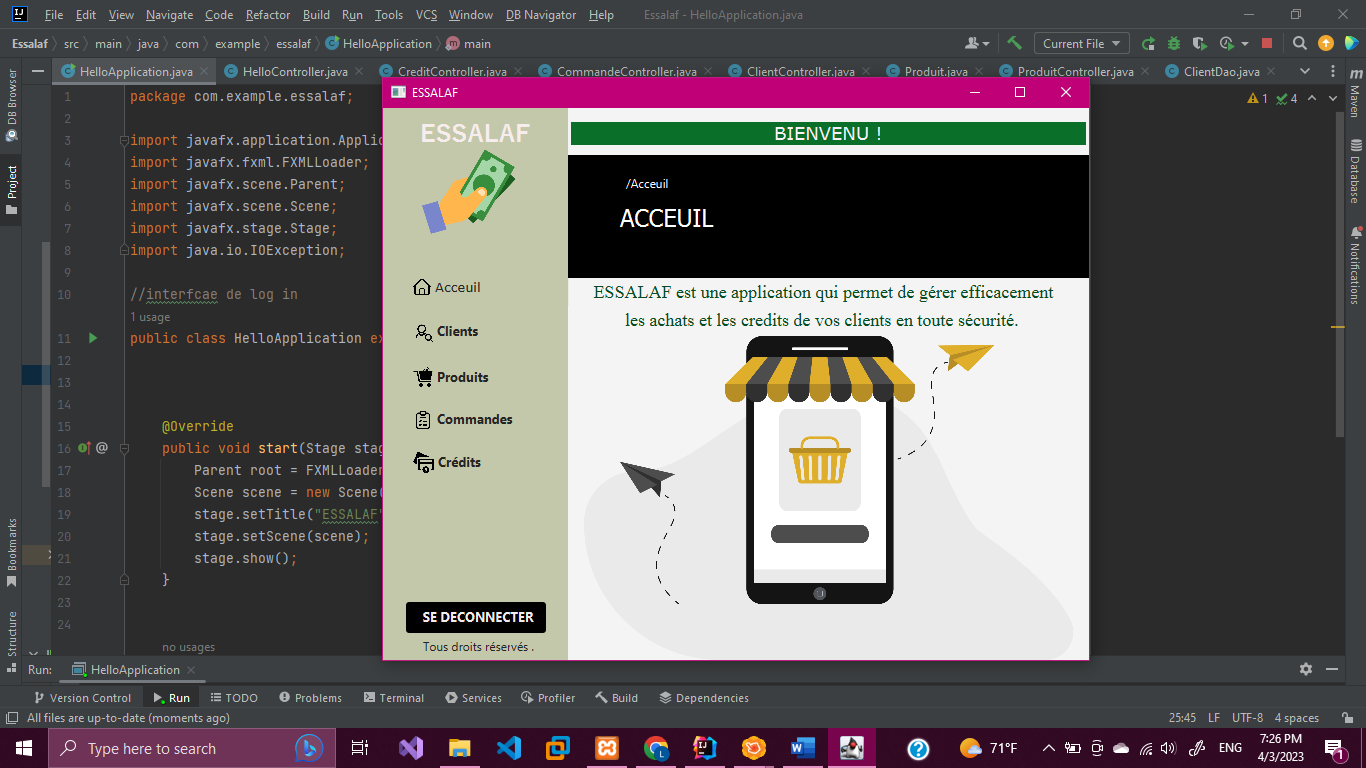
public void showproduits() {  
 ObservableList<Produit> Liste = getproduitsListe() ;  
 colQ.setCellValueFactory(new PropertyValueFactory<>("Quantite"));  
 colN.setCellValueFactory(new PropertyValueFactory<Produit, String>("Nom"));  
 colP.setCellValueFactory(new PropertyValueFactory<Produit, Float>("Prix"));  
 tab.setItems(Liste);  
}

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

@Override  
 public void initialize(URL url, ResourceBundle resourceBundle) {  
 showproduits();  
 }  
  
  
  
  
  
  
}

**Par ce principe, j’ai dû travailler avec toutes les autres classes.**

**EXPLICTION DU FONCTIONNEMENT DES INTERFACES G. :**

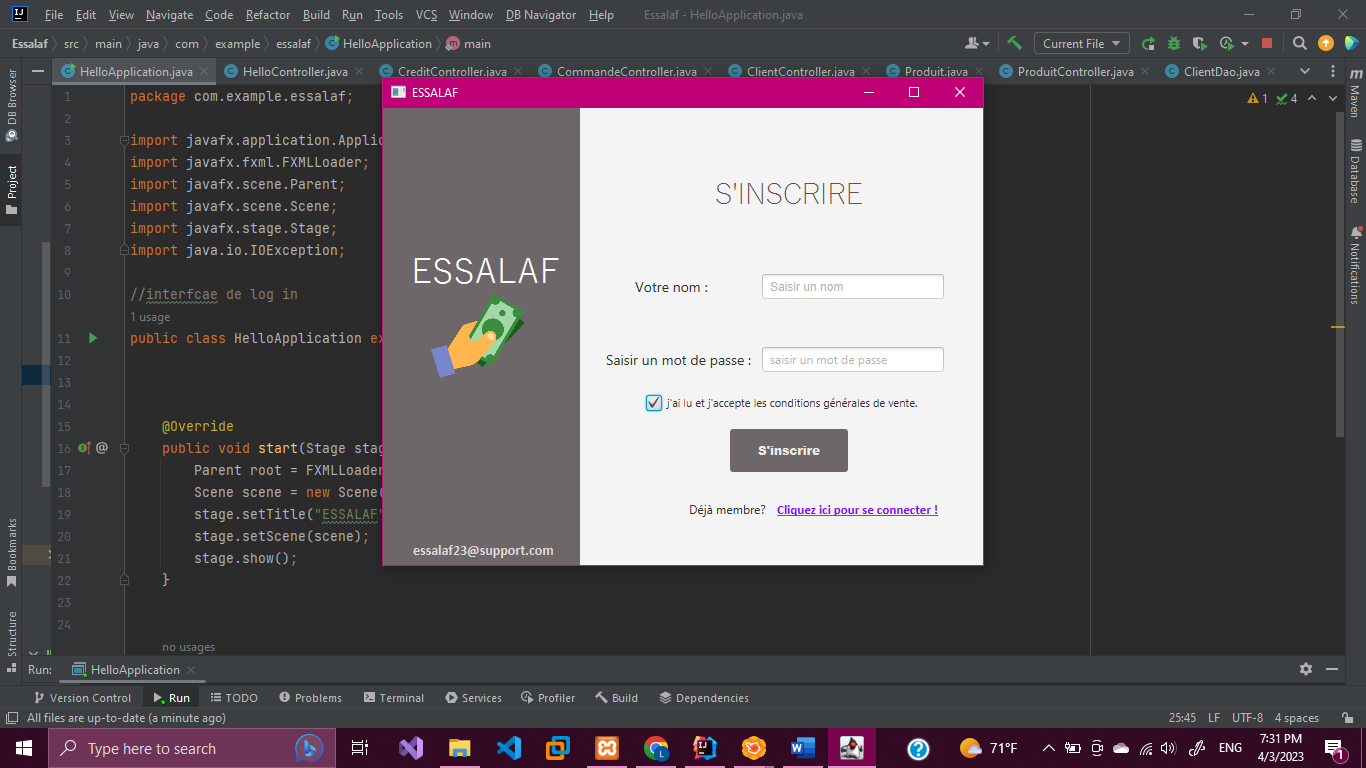
****

**J’ai préféré de travailler sur un menu que pour chaque élément, en cliquant**

**On change de scène en affichant les différentes catégories : Les tables du client, Produit,Commande,Credit .Se deconnecter nous fait revenir A l interface du login.**

**Interface Inscription :**

**Si l’utilisateur n’est pas inscrit, il peut remplir un formulaire et s’inscrire, le bouton va le ramener a l interface login pour se connecter a nouveau**

****

**3-Difficultés rencontrées :**

Parmi les difficultés que j’ai trouvé et que j’ai pas réussi à le régler facilement

LA CONNEXION A MA BASE DE DONNES .J’avais un problème très grave puisque j’avais pas des erreurs dans mon code, ni dans la base de donnes ,ni dans les informations de connexion, ni dans la configuration du JDBC …

La solution était de refaire le projet a zéro dans les dernières jours .ET heureusement j’ai réussi a le régler .

**CONCLUSION**

En conclusion, la réalisation de cette application de gestion des crédits des clients en utilisant JavaFX a permis de mettre en pratique les concepts clés de la programmation orientée objet et de la conception de interfaces graphiques. Le développement de cette application a nécessité une compréhension approfondie des besoins des utilisateurs et de l'architecture du projet.

En somme, ce projet a permis d'acquérir des compétences clés en programmation et en conception d'interfaces graphiques tout en répondant à un besoin pratique dans le domaine de la gestion des crédits des clients. Il peut être amélioré davantage en ajoutant de nouvelles fonctionnalités et en l'intégrant à des systèmes plus complexes.