



Université Abdelmalek Essaadi
Faculté des sciences et techniques de Tanger
Département Génie informatique

LST GI S6
JAVA

RAPPORT DE PROJET Tracking des Livreurs :
PROGRAMMATION ORIENTER OBJET JAVA



Encadre Par :

- EL ACHAK LOTFI
- El Mokhtar EN-NAIMI

Réaliser Par :

- ASSOULFI OUASSIM

Groupe :

- 2

1. LES FONCTIONS ET LES NOTES LES PLUS IMPORTANTES DANS MON PROJET

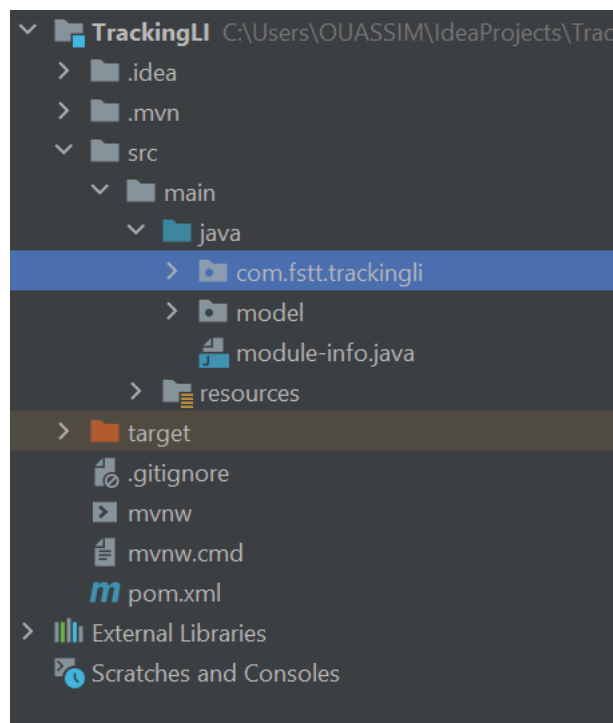
- Afin de développer une Interface Graphique **de Tracking des Livreurs** avec JDBC et JAVA FX :

- ✓ Applique le concept de MVC
- ✓ Création des fichier FXML qui représente la vue de notre interface
- ✓ Création des fichiers de css
- ✓ CRUD des Livreurs
- ✓ CRUD des commandes

- ✓ **CRUD des produits**
- ✓ **Connexion avec la base des données**
- ✓ **La fonction de switch entre les différent scènes de notre interface**
- ✓ **Le fonctionnement de Login**

ARCHETECTURE DE MVC

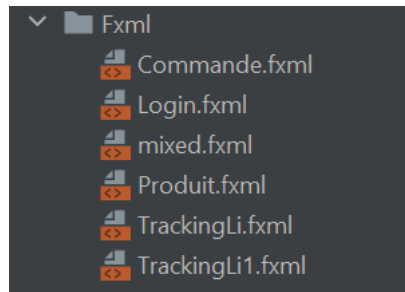
J'ai appliqué le concept de MVC sur mon projet comme pour faciliter de résoudre les erreurs



Le `com.fstt.trackingli` représente les classes contrôleurs et le `resources` contient la vue de projet.

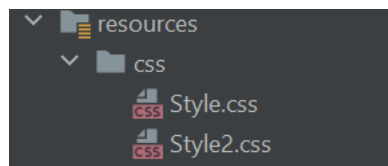
CREER LES FICHIERS FXML

Nous créons des fichiers `.fxml` pour manipuler la vue de notre interface, puis je le développe sur la scène Builder



CREER LES FICHER CSS

Pour créer un design a min projet css j'ai établi de créer des fichiers de feuille de style.



CRUD DES LIVREURS

Pour maintenir un CRUD a mon interface qui concerne les livreurs j'ai créé un class livreurs et une tableau livreurs sur ma base de données qui s'appelle trackingli

Parcourir Structure SQL Rechercher Insérer Exporter Importer Privilèges Opérations f										
Structure de table		Vue relationnelle								
	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1	id	int(11)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer
<input type="checkbox"/>	2	Nom	varchar(20)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer
<input type="checkbox"/>	3	Prenom	varchar(20)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer
<input type="checkbox"/>	4	Telephone	varchar(20)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer

Et la class livreurs contient les attributs de livreurs comme (id, Nom...) et aussi leurs getters and setters :

```

package model;

public class Livreur {
    private int id ;
    private String Nom;
    private String Prenom;
    private String Telephone ;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNom() {
        return Nom;
    }

    public void setNom(String nom) {
        Nom = nom;
    }

    public String getPrenom() {
        return Prenom;
    }

    public void setPrenom(String prenom) {
        Prenom = prenom;
    }

    public String getTelephone() {
        return Telephone;
    }

    public void setTelephone(String telephone) {
        Telephone = telephone;
    }
}

```

Et j'ai une class `LivreurController` qui réagit avec l'interface pour obtenir les différentes variables de l'interface comme le textfield et les buttons de CRUD ... (les détails de code tu peut leurs avoir sur [GitHub](#))

1. Afficher le tableau des livreurs :

j'ai trois fonction pour afficher les livreurs :

getAll() : pour obtenir les lignes de tableau de la base des données et leurs stocker sur une liste et cette fonction se trouve dans **livreurDAO**

getDataLivreur() : pour obtenir les lignes de la liste qui se trouve sur la fonction **getAll()** ;

showlivreurs() : la fonction qui affiche les lignes des informations dans notre tableau d'interface graphique

➤ **GetDataLivreur()** (dans le controller):

```
public static ObservableList<Livreur> getDataLivreurs() {  
  
    LivreurDAO livreurDAO = null;  
  
    ObservableList<Livreur> listfx = FXCollections.observableArrayList();  
  
    livreurDAO = new LivreurDAO();  
    for (Livreur ettemp : livreurDAO.getAll())  
        listfx.add(ettemp);  
  
    return listfx ;  
}
```

➤ **GetAll()** (dans le Livreur DAO):

```
@Override  
public List<Livreur> getAll() {  
  
    List<Livreur> mylist = new ArrayList<Livreur>();  
  
    String request = "select * from livreurs ";  
  
    try {  
        st= con.prepareStatement(request);  
        rs=st.executeQuery();  
  
        // parcours de la table  
        while ( this.rs.next()){  
  
        // mapping table objet  
            Livreur li = new Livreur();  
            li.setNom(rs.getString("Nom"));  
            li.setPrenom(rs.getString("Prenom"));  
            li.setTelephone(rs.getString("Telephone"));  
            li.setId(rs.getInt("Id"));  
  
            mylist.add(li);  
  
        }  
    } catch (SQLException e) {  
        throw new RuntimeException(e);  
    }  
}
```

```

        return mylist;
    }

```

➤ **Showlivreurs ()** (dans le class livreurController):

```

    public void showlivreurs(){
        ObservableList<Livreur> list = getDataLivreurs();
        table.setItems(list);
        colid1.setCellValueFactory(new
PropertyValuFactory<Livreur,Integer>("id"));
        colnom1.setCellValueFactory(new
PropertyValuFactory<Livreur,String>("Nom"));
        colprenom1.setCellValueFactory(new
PropertyValuFactory<Livreur,String>("Prenom"));
        coltele.setCellValueFactory(new
PropertyValuFactory<Livreur,String>("Telephone"));
    }

```

2. Ajouter une Livreur :

On ajoute une livreur a travers une fonction de ajouterLivreur() qui est inclue sur le Button de ajouter sur l'interface et puis on ajouter cette Livreur a nos base de données avec la fonction save() dans la class de LivreurDAO

➤ **Ajouterlivreur()** (dans le class de livreurController) :

```

@FXML
void ajouterlivreur(ActionEvent event) {

    Livreur li=new Livreur();
    li.setNom(tnom1.getText());
    li.setPrenom(tprenom1.getText());
    li.setTelephone(ttelephone1.getText());
    LivreurDAO livr = new LivreurDAO();
    livr.save(li);
    showlivreurs();
    clear();
}

```

➤ **Save()** (dans la class de livreurDAO) :

```

➤ public void save(Livreur object){
    String request = "insert into livreurs (Nom ,Prenom, Telephone) values
(?, ?, ?)";

    // mapping objet table

```

```

    try {
        this.st = this.con.prepareStatement(request);
        this.st.setString(1 , object.getNom());
        this.st.setString(2,object.getNom());
        this.st.setString(3 , object.getTelephone());
        this.st.execute();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
    // mapping
}

```

3. Modifier Livreur:

On modifier une livreur a travers une fonction de modifierlivreur() qui est inclue sur le Button de Modifier sur l'interface et puis on modifier cette Livreur a nos base de données avec la fonction update() dans la class de LivreurDAO

➤ **ModifierLivreur()** (dans Livreurcontroller) :

```

@FXML
void modifierlivreur(ActionEvent event) {
    Livreur li= new Livreur();
    LivreurDAO livr = new LivreurDAO();
    li.setNom(tnom1.getText());
    li.setPrenom(tprenom1.getText());
    li.setTelephone(ttelephone1.getText());
    Livreur check =table.getSelectionModel().getSelectedItem();
    id = check.getId();
    livr.update(li,id);

    showlivreurs();
    clear();
}

```

➤ **Update ()** (dans livreurDAO):

```

public void update(Livreur object,int id){
    String query ="update livreurs set Nom = ?, Prenom = ?, Telephone= ?
where id = ?";
    try {
        this.st = this.con.prepareStatement(query);
        this.st.setString(1 , object.getNom());
        this.st.setString(2,object.getPrenom());
        this.st.setString(3 , object.getTelephone());
        this.st.setInt(4, id);
        this.st.executeUpdate();
    }
}

```



```

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}

```

4. Supprimer un livreur:

On supprime un livreur à travers une fonction de `SupprimerLivreur()` qui est incluse sur le Button de supprimer sur l'interface et puis on supprime ce livreur à nos bases de données avec la fonction `supprimer()` dans la classe de `LivreurDAO`

➤ `SupprimerLivreur()` :

```

➤ @FXML
void supprimerlivreur(ActionEvent event) {
    LivreurDAO liv=new LivreurDAO();
    Livreur li =table.getSelectionModel().getSelectedItem();
    id = li.getId();
    liv.supprimer(id);

    showlivreurs();
    clear();
}

```

➤ `Supprimer()`:

```

    public void supprimer(int id) {
        String query = "Delete from livreurs where id = ?";
        try {
            st=con.prepareStatement(query);

            st.setInt(1,id);
            st.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}

```

5. Effacer tous le tableau:

On supprimer tous les livreurs à travers une fonction de `ClearTable()` qui est incluse sur le Button de clear sur l'interface et puis on applique cette action à nos bases de données avec la fonction `clear()` dans la classe de `LivreurDAO`

✓ **ClearTable():**

```
@FXML
void clearTable(ActionEvent event){
    LivreurDAO livr = new LivreurDAO();
    livr.clear();
    showLivreurs();
    clear();
}
```

✓ **Clear():**

```
public void clear() {
    String query = "Delete from livreurs ";

    try {
        st=con.prepareStatement(query);

        st.executeUpdate();

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

CRUD DES COMMANDES

Eh bien, nous allons faire la même chose avec les commandes également, les mêmes structures de fichiers et le même code la seule différence ici est dans quelle table de la base de données nous travaillons dessus

<(le code source se trouve dans GitHub)>

- ❖ on peut voir dans chaque commande les produits qui l'incluent, en utilisant trois class (mixed, mixedDAO , mixedController), et une interface pour afficher les produits de chaque commande mixed.fxml.

on exécute la fonction de getData() dans chaque fois on sélectionne une ligne dans la tableau des commandes et on passe le idCommande a use fonction qui remplir une liste de type mixed et a travers une fonction qui remplace une autre scène de les produits de commande on peut juste affiche les ligne des produit de table mixed qui contient cette valeur de idCommande au notre scène de mixed

- ❖ cette scène de mixed qui contient les produit de commande peut réaliser des fonction de CRUD() aussi :

- ❖ La fonction getallm() qui retourne une liste des ligne choisir par idcommande :

```
static List<Integer> counter =new ArrayList<>();
public void setCount(int count) {
    counter.add(count);
}

@Override
public void clear() {
}
@Override
public List<mixed> getallm() {

    List<mixed> mylist = new ArrayList<>();

    String request = "SELECT * FROM mixed where idcommande =?";

    try {
        Iterator<Integer> it = counter.iterator();
        st= con.prepareStatement(request);
        this.st.setInt(1 ,it.next());

        rs=st.executeQuery();

// parcours de la table
        while ( this.rs.next()){

// mapping table objet
            mixed li = new mixed();
            li.setLabel(rs.getString("Label"));
            li.setPrix(rs.getFloat("Prix"));
            li.setIdproduit(rs.getInt("idproduit_commande"));
            li.setQte(rs.getInt("qte"));
            li.setIdcommande(rs.getInt("idcommande"));

            mylist.add(li);
```

```
    }  
    } catch (SQLException e) {  
        throw new RuntimeException(e);  
    }  
  
    return mylist;  
}
```

CRUD DES PRODUITS

Eh bien, nous allons faire la même chose avec les Produits également, les mêmes structures de fichiers et le même code la seule différence ici est dans quelle table de la base de données nous travaillons dessus

<(le code source se trouve dans GitHub)>

CONNEXION AVEC LA BASE DES DONNEES

- ❖ A travers une Class abstraite qui va être hériter dans une ensemble des class qui nécessite la connexion avec la base des donnees comme LivreureDAO ,CommandeDAOA ... Qui s'appelle BaseDAO on peut connecter avec notre base des données.

On maintenir la connexion sur notre constructeur et on l'appeler dans plusieurs class qui leur hérite utilisant la méthode `super()`

- ❖ Cette class contient aussi des méthodes abstraite que on va utiliser dans les autres modèles comme LivreurDAO, CommandeDAO...

```
❖ package model;

import java.sql.*;
import java.util.List;

public abstract class BaseDAO {
    static String user = "root";
    static String password = "";
    static String url = "jdbc:mysql://localhost/TrackingLi";
    static String driver = "com.mysql.cj.jdbc.Driver";
    public Connection con = null;
    public PreparedStatement st = null;
    public ResultSet rs = null;
    public BaseDAO() {

        try {
            Class.forName(driver);
            try {
                con = DriverManager.getConnection(url, user, password);
            } catch (SQLException e) {
                throw new RuntimeException(e);
            }
        } catch (ClassNotFoundException e) {
            throw new RuntimeException(e);
        }

    }

    public abstract void save(Livreur object);
    public abstract void update(Livreur object, int id);
    public abstract void supprimer(int id);
    public abstract void clear();
    public boolean login(Login object) {
        return true;
    }

    public List<Livreur> getall() {

        return null;
    }
    public List<Commande> getallc() {

        return null;
    }
    public List<Produit> getallp() {

        return null;
    }
    public List<mixed> getallm() {

        return null;
    }

}
}
```

LA FONCTION DE SWITCH ENTRE LES DIFFERENTS SCENES

- ❖ On peut changer une scène par un autre d'une manière manuelle, C'est juste de appeler cette méthode sur une Button utilisant l'attribut on Action
- ❖ La fonction getresource() permet de retourner la scène que nous voulons aller

```
@FXML
void switchToTracking1(ActionEvent event) throws IOException {

    Parent trackingParent =
FXMLLoader.load(getClass().getResource("/FXML/Commande.fxml"));
    Scene trackingScene = new Scene(trackingParent);

    Stage window = (Stage) ((Node)
event.getSource()).getScene().getWindow();
    window.setScene(trackingScene);
    window.show();

}
```

LA FONCTIONNEMENT DE LOGIN

- ❖ Pour un admin faire une connexion il faut insérer (userName , password) et le système vérifier que cette password et username existe dans la base des données, Puis si il existe il apparaitre une autre scène de Dashboard utilisant la fonction que j'ai déjà expliquer en haut qui contient les livreurs.
- On a trois Class qui sont responsable a cette fonctionnement se sont : Login , LoginController, LoginDAO(herite de BaseDAO).

❖ Login :

```
package model;

public class Login {
    protected String username;
    protected String password;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

❖ LoginDAO:

```
❖ package model;

import javafx.event.ActionEvent;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonType;
import javafx.stage.Stage;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

public class LoginDAO extends BaseDAO{

    public LoginDAO(){
        super();
    }

    @Override
    public boolean login(Login object){
        try {
            st=con.prepareStatement("select * from login where username=?
AND password=?");
```

```

        st.setString(1,object.getUsername());
        st.setString(2, object.getUsername());
        rs=st.executeQuery();
        if(rs.next()){

            return true;
        }
        else{
            return false;
        }
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}

@Override
public void save(Livreur object) {

}

@Override
public void update(Livreur object, int id) {

}

@Override
public void supprimer(int id) {

}

@Override
public void clear() {

}

@Override
public List<Livreur> getall() {
    return null;
}

}

```

❖ LoginController:

```

package com.fstt.trackingli;

import javafx.event.ActionEvent;

```



```

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Group;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import model.Login;
import model.LoginDAO;

import java.io.IOException;
import java.net.URL;
import java.util.Objects;
import java.util.ResourceBundle;

public class LoginController implements Initializable {
    public TextField tname;
    public TextField tpass;
    public Button BtnCon;

    public void initialize(URL url, ResourceBundle resourceBundle){

    }

    private Stage stage;
    private Scene scene;
    @FXML
    void switchToTracking(ActionEvent event) throws IOException {
        Login lo=new Login();
        LoginDAO log=new LoginDAO();
        lo.setUsername(tname.getText());
        lo.setPassword(tpass.getText());

        boolean test =log.login(lo);
        if(test){
            Parent trackingParent =
FXMLLoader.load(getClass().getResource("/FXML/TrackingLil.fxml"));
            Scene trackingScene = new Scene(trackingParent);

            Stage window = (Stage) ((Node)
event.getSource()).getScene().getWindow();
            window.setScene(trackingScene);
            window.show();}
        else{
            Alert alert = new Alert(Alert.AlertType.ERROR, "Invalid username or
password");
            alert.showAndWait();
        }
    }
}

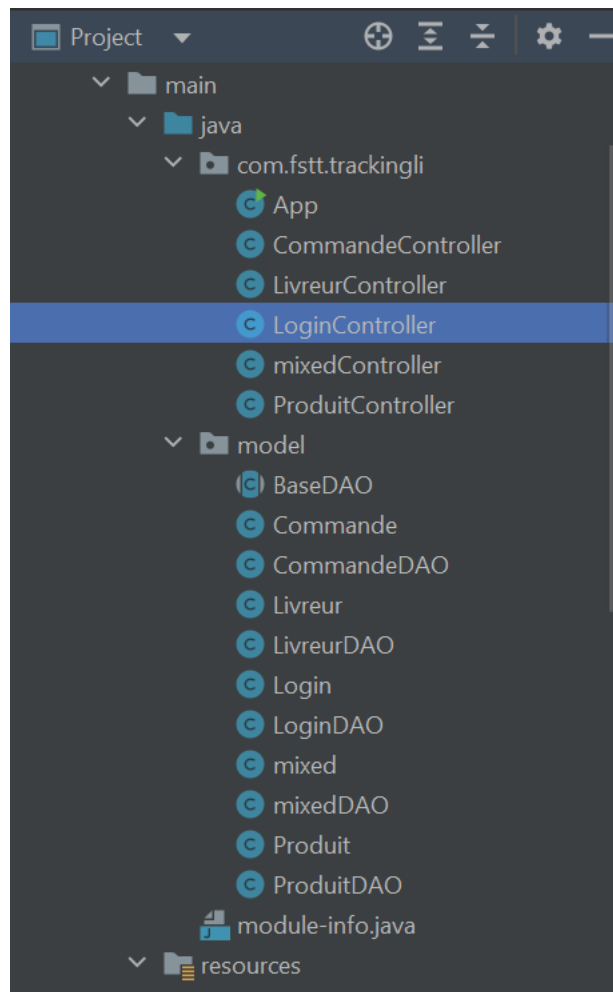
```

```
}  
  
}
```

2. LES SCENES QUE J'AI DEVELOPPER

Dans ce projet j'ai développé 5 Scène Login,Livreur,Commandes,Produits,mixed(les produits de commande)

Mes Class JAVA que j'ai fait :



Dans les fichier ces class j'ai ajouté tous les fonctions et les astuces qu' on a discuté en haut comme :

- 1. CRUD des livreurs**
- 2. CRUD des commandes**
- 3. CRUD des produits**
- 4. La connexion avec La base des données**
- 5. Les fonction de remplacement des scènes**
- 6. Remplacer la scène après une favorable insertion de username et password**
- 7. Ajouter les buttons avec ces fonctions**

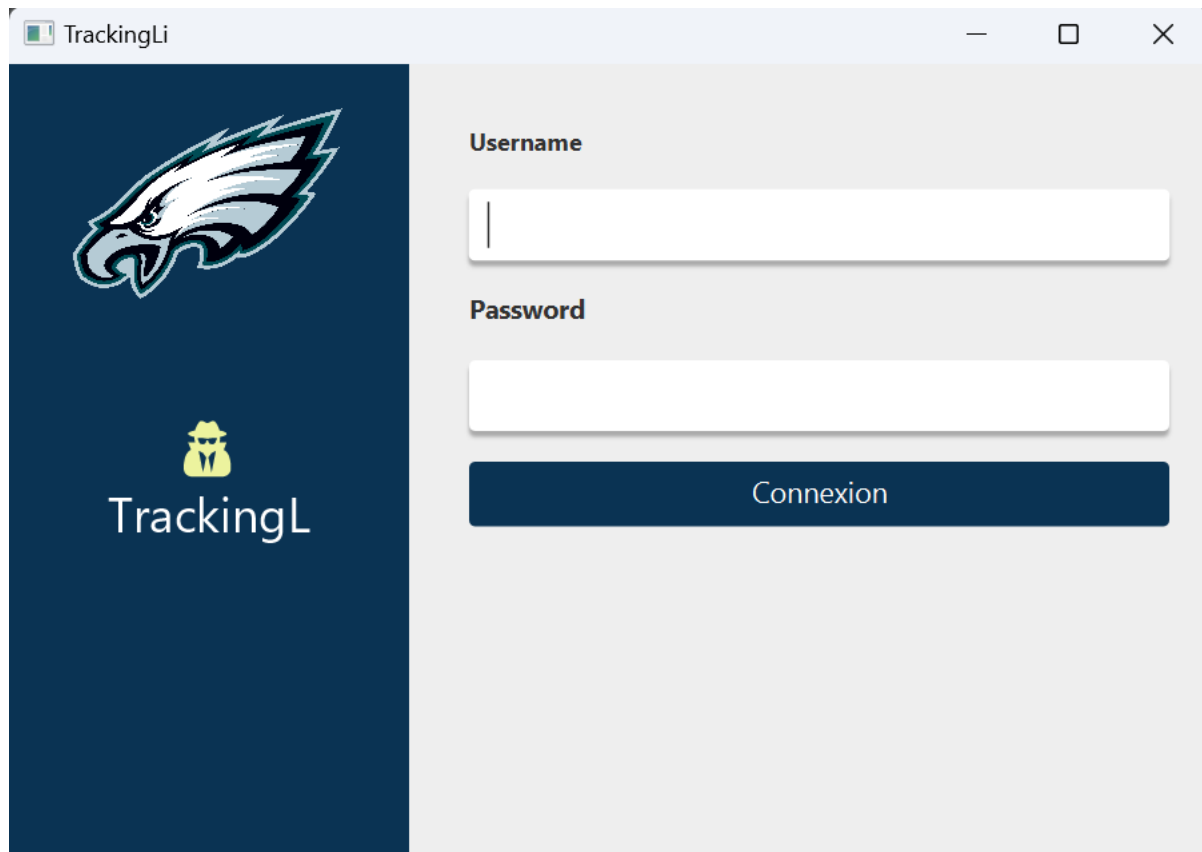
❖ Ces fonctions sera répété dans presque 90% des 5 Scènes.

LES PROBLEMES

- Les problèmes que j'ai combattre c'est pour accéder aux produits des commande, enfin j'ai *résoudre* cette problème en utilisant une liste statique que on stocke a cette liste le idcommande que on veut explorer a partir de fonction de getData() de tableau

LOGIN

- ✓ Dans cette scène l'admin faut insérer le username et le mot de passe pour accéder au Dashboard qui donne la possibilité au admin de choisir qu'elle champs veut il



The screenshot shows a web browser window titled "TrackingLi". The interface is split into two main sections. On the left, there is a dark blue vertical sidebar containing a white eagle head logo at the top and a yellow icon of two figures in a trench coat below it, with the text "TrackingL" underneath. On the right, the main content area has a light gray background. It contains two white input fields: the first is labeled "Username" and the second is labeled "Password". Below these fields is a dark blue button with the white text "Connexion".

LIVREURS :

Cette scène représente Les livreurs actif dans notre système, cette scène a une relation avec les class suivant : `Livreur` , `LivreurDAO` , `LivreurController`

TrackingLi

Commandes

Produits

Logout

LES LIVREURS

Nom:

Prenom:

Telephone:

Ajouter
Modifier
Supprimer
Clear

Id	Nom	Prenom	Telephone
14	assoufi	assoufi	3424545

LES COMMANDES :

Cette scène représente Les commandes actif dans notre système, cette scène a une relation avec les class suivant : Comamnde , commandeDAO , commandeController

TrackingLi

Commandes

Produits

Logout

LES COMMANDES

Adresse

date_debut

date_fin

Id_livreure

11

Ajouter
Modifier
Supprimer
Clear

Idcommande	Adresse	date_debut	date_fin	id_Livreure
3	bni makada	1/4/2023	3/4/2023	11
4	boukhalf	4/4/2023	8/4/2023	11

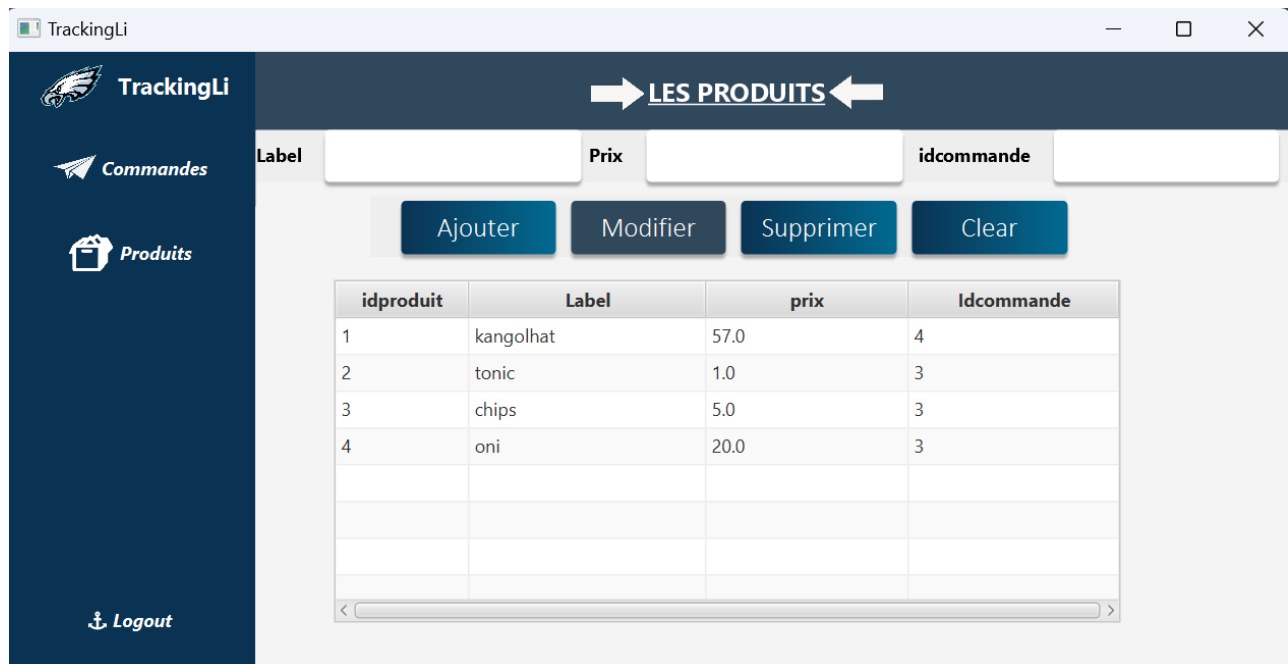
NOTE :

Quand je veux accéder a les produit d'une commande c'est juste de sélectionne une commande après cliquer le Button rouge , cette scène a une relation avec les class suivant : `mixed` , `mixedDAO` , `mixedController`



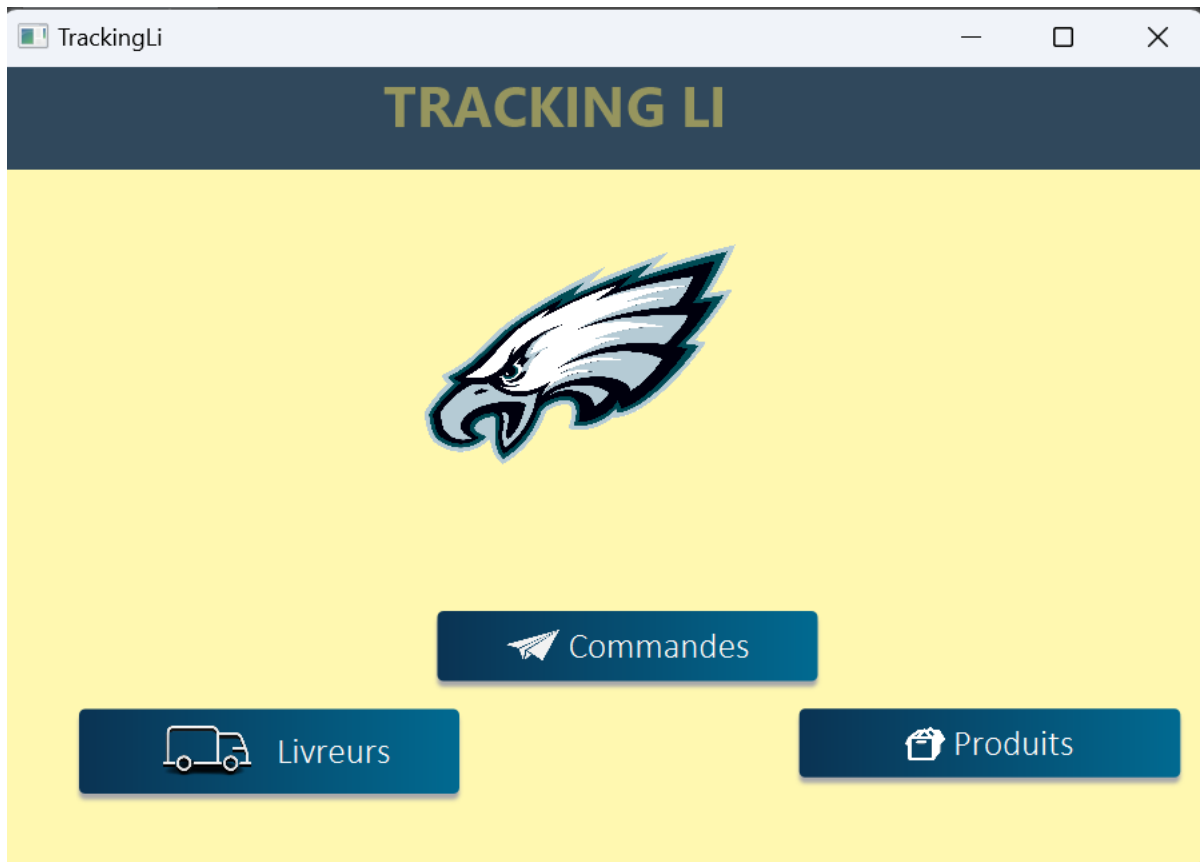
LES PRODUITS :

Cette scène représente Les produits dans notre système, cette scène a une relation avec les class suivant : `Prduit` , `ProduitDAO` , `ProduitController`



LE DASHBOARD :

Cette scène Donne la possibilité au admin de choisir les champs qui veut-il de leurs accéder, cette scène a une relation avec les class suivant : Dashboard



Les outils utiliser :

- Photoshop.
- Visual Studio Community
- Intelij Idea
- Scene Builder

LA FIN

Merci Pour Votre Attention