



Rapport Mini-Projet Tracking Des livreurs :

Intitulé :

Département SI

Encadre par :

Mr El-Mokhtar En-Naimi
Mr Lotfi El ACHAAK

Réalise par :

Abdelmajid Benjelloun
LST G.INFO Gr 2
H130407540
20000203

Vous Trouverez Le code Source ainsi qu'un vidéo et la bdd dans Classroom +Lien GitHub

Introduction :

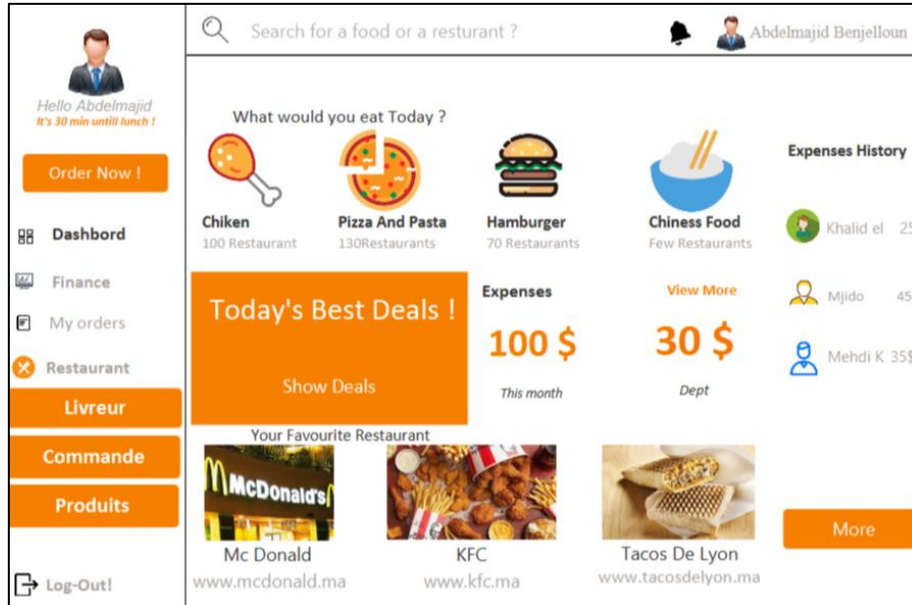
Bienvenue dans ce rapport de projet. Dans ce document, vous trouverez une analyse détaillée de mon projet ainsi que les résultats obtenus. J'ai travaillé dur pour atteindre les objectifs que nous nous étions fixés et je suis heureux de pouvoir vous présenter le fruit de mon travail. Ce rapport contient des informations clés sur le projet, y compris la portée, les méthodes utilisées, les résultats obtenus et les conclusions tirées. J'espère que ce rapport vous sera utile et vous donnera un aperçu complet de notre projet.

Je tiens à remercier Mr.Achaak , Mr Naimi Pour le témoignage et leur conseils et orientation durant ce projet- la .

But de ce projet :

Le projet Tracking_L sert à tracker les livreurs, qui livrent les commandes des clients qui commandent des produits.

J'ai créé un Dash-board



Le Dashboard se divise en plusieurs parties :

Part1 : Partie Menu où on trouve Dashboard, Finance, My orders, Restaurant, Ainsi que 3 Buttons qui mènent à 3 interfaces ou fichiers .FXML

Part2 : une interface où on trouve des catégories des restaurants avec le nombre des restaurants disponibles Avec 3 derniers restaurants et l'historique des dépenses.

Langage utilisé :

JavaFx, JDBC, MySQL, POO en JAVA

Outils utilisés :

GitHub, IntelliJ Idea,

Démarche :

On a commencer le projet par faire un setup D'inteliJ, Installer Sdk ,
Driver pour la base de donnees , Cree une base de donnees ,

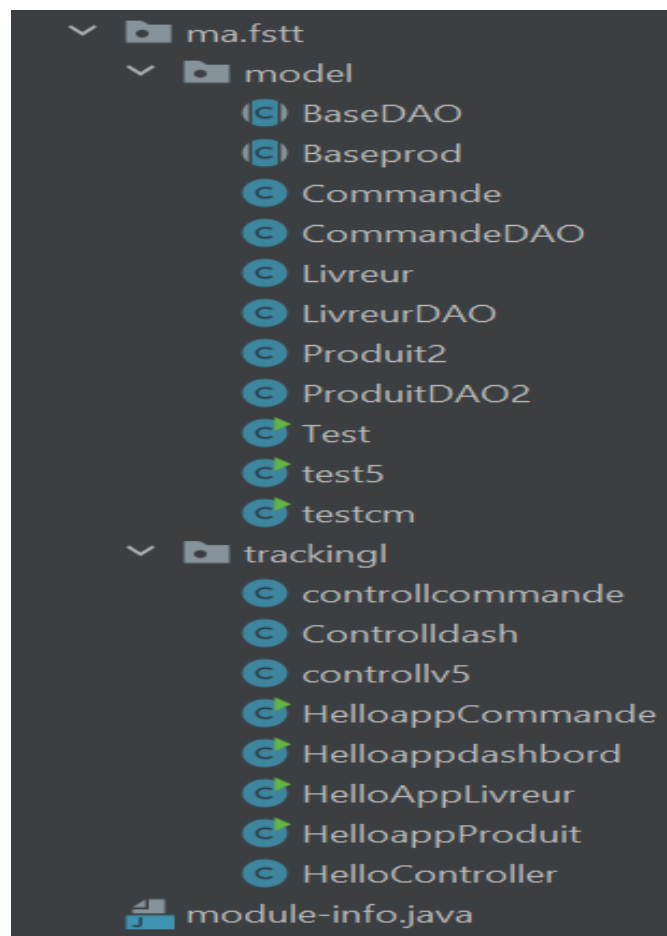
Pour la base de donnees :

```
CREATE TABLE `livreur` (  
  `id_livreur` int(11) NOT NULL,  
  `nom` varchar(255) DEFAULT NULL,  
  `telephone` varchar(255) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;  
  
--  
-- Dumping data for table `livreur`  
--  
  
INSERT INTO `livreur` (`id_livreur`, `nom`, `telephone`) VALUES  
(668922, 'Abdelmajid', '0661898183'),  
(668931, 'adil', '76379384'),  
(668966, 'jamal', '34567'),  
(668967, 'AHMED', '99999.0');  
  
-----  
  
--  
-- Table structure for table `produit2`  
--  
  
CREATE TABLE `produit2` (  
  `id_produit` int(11) NOT NULL,  
  `prix` varchar(255) NOT NULL,  
  `description` varchar(255) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;  
  
--  
-- Dumping data for table `produit2`  
--  
  
INSERT INTO `produit2` (`id_produit`, `prix`, `description`) VALUES  
(34, '344.89', 'tet'),  
(1, '67.98', 'ALIMEN');  
  
--  
-- Indexes for dumped tables  
--  
  
--
```

Tracking Des Livreurs

```
-- Indexes for table `livreur`  
--  
ALTER TABLE `livreur`  
  ADD PRIMARY KEY (`id_livreur`);  
  
--  
-- AUTO_INCREMENT for dumped tables  
--  
  
--  
-- AUTO_INCREMENT for table `livreur`  
--  
ALTER TABLE `livreur`  
  MODIFY `id_livreur` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=668968;  
COMMIT;
```

Définition élément du travail :



Ma.fstt.model : Sert a definir les classes .

Définition et rôle de chaque classe :

Base DAO :

```
package ma.fstt.model;

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

public abstract class BaseDAO <T>{

    protected Connection connection ;
    protected Statement statement ;

    protected PreparedStatement preparedStatement;

    protected ResultSet resultSet ;

    // connexion avec bdd

    private String url = "jdbc:mysql://127.0.0.1:3306/glovo";
    private String login = "root";
    private String password = "";

    BaseDAO() throws SQLException {
        this.connection = DriverManager.getConnection(url , login ,password
    );
    }

    public abstract void save( T object ) throws SQLException;

    public abstract void modify(T object) throws SQLException;

    public abstract void delete(T object ) throws SQLException ;

    public abstract List<T> getAll( ) throws SQLException ;

    public abstract T getOne(Long id) throws SQLException;
}
```

Cet class abstraite sert a définir les méthodes qu'on va implémenter dans notre projet { On a utilisé le concept des Template pour éviter écrire les mêmes fonction pour un type différent}

Livreur DAO :

Cette classe la sert à définir les méthodes créées dans BASEDAO
Elle a une relation a travers du Drivers avec la base donne qu'on a créer

```
public void save(Livreur object) throws SQLException {  
    String request = "insert into livreur (nom , telephone) values (?,  
    ?)";  
    // mapping objet table  
    this.preparedStatement = this.connection.prepareStatement(request);  
    // mapping  
    this.preparedStatement.setString(1 , object.getNom());  
    this.preparedStatement.setString(2 , object.getTelephone());  
    this.preparedStatement.execute();  
}
```

on prend cette exemple-là

définition de la méthode **Save (Livreur Obj)**

On écrit la demande ou requête (utilisation du concept de la bdd my Sql
Qu'on étudier avec Mr.Zouhair)

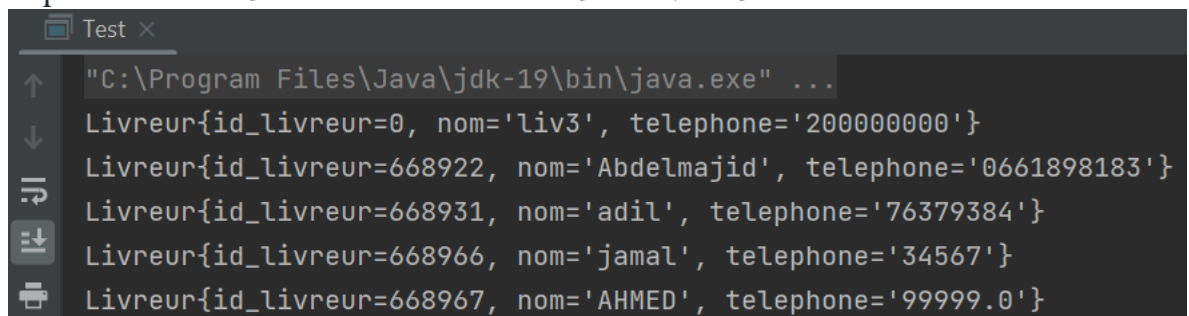
Apres on prepare notre demande par this.preparedStatement.set(TYPE)

Et on l'exécute

```
public List<Livreur> getAll() throws SQLException {  
  
    List<Livreur> mylist = new ArrayList<Livreur>();  
    String request = "select * from livreur";  
    this.statement = this.connection.createStatement();  
    this.resultSet = this.statement.executeQuery(request);  
}
```

la méthode **getAll()** Sert à afficher le contenu de la table dans ce cas la
table est livreur

On a créé une class TEST qui sert à remarquer et corriger s'il y a une
erreur il permet aussi d'afficher le contenu de la table dans le terminal



```
Test x  
"C:\Program Files\Java\jdk-19\bin\java.exe" ...  
Livreur{id_livreur=0, nom='liv3', telephone='200000000'}  
Livreur{id_livreur=668922, nom='Abdelmajid', telephone='0661898183'}  
Livreur{id_livreur=668931, nom='adil', telephone='76379384'}  
Livreur{id_livreur=668966, nom='jamal', telephone='34567'}  
Livreur{id_livreur=668967, nom='AHMED', telephone='99999.0'}
```

Ma.fstt.tracking

Controller :

Cet class la a une relation avec 2 classes directement (Ficher.FXML,HelloApplication)

FichierFXML un fichier lors on créer notre scène ou notre interface graphique

Hello Application sert à lancer notre scène et démarrer notre application

```
public void start(Stage stage) throws IOException {  
    FXMLLoader fxmlLoader = new  
FXMLLoader(HelloAppLivreur.class.getResource("Dashbord.fxml"));  
    Scene scene = new Scene(fxmlLoader.load(), 800, 500);  
    stage.setTitle("Food Delivery ");  
    stage.setScene(scene);  
    stage.show();  
}
```

voici un exemple de HelloApplcation du Dashboard

On configure aussi les dimensions et le titre de L'application

Alors on continue avec Hellocontroler

On configure aussi le comportement de l'interface lorsque interagie avec l'interface (Les buttons , faire des Maj,ajouter des buttons ,des fonctions ...)

Module info.java

Sert à définir les bibliothèques qu'on va utiliser .


```
module ma.fstt.tracking1 {  
    requires javafx.controls;  
    requires javafx.fxml;  
  
    requires org.controlsfx.controls;  
    requires org.kordamp.bootstrapfx.core;  
  
    opens ma.fstt.tracking1 to javafx.fxml;  
    exports ma.fstt.tracking1;  
  
    exports ma.fstt.model;  
  
    requires java.sql;  
}
```

On détaille les fichiers **.FXML**

Dans notre projet là on a créé 4 fichiers = 4 interfaces

Interface Livreur , Produit, Commande, Dashboard

Tracking Des Livreurs




Hello Abdelmajid
Choose Your delivery person

- Dashboard
- Finance
- My Orders
- Restaurant


Back

Entrez Nom du Restaurant


Les Livreurs Disponibles Aujourd'hui




Mehdi Kellat
+212 678 876 345



Abderahman nej
+212 789 098 456



Simo El
+212 567 345 128



Zakariya cha
+212 678 982 356

Nom du Livreur

Entrez Le nom

Save


Telephone Du Livreur

Entrez Le Numero

Modifier

Valider

id_Livreur	nom livreur	telephone du livreur
No content in table		



Hello Abdelmajid
Edit Your Commande

- Dashboard
- Finance
- Commande
- Restaurant

Back

Entrez Nom du Restaurant

RAMADAN
SPECIAL PROMO
UP TO 80% DISCOUNT

Nom client

Entrez Le nom Du client


Save

Categorie

Indiquer la categorie

Modify

Date_Debut	Nom client	Categorie
No content in table		




Hello Abdelmajid
Choose Your Prouct Here


- Dashboard
- Finance
- Orders
- Restaurants

Back


Search for a restaurants ...




Produit Tech



Sante & Beaute



Alimentation



Bebe et enfant

Prix Du produit

Save

Description

modify

Code Prod	Prix Prod	Categorie
No content in table		

Tracking Des Livreurs

Tous ces boutons -la bien que **Textfield,tableau,collones** sont définies dans **Controller**

```
@FXML
private TextField nom ;
public long idproduitSelectionne;
@FXML
private Button save;

@FXML Button btn1;
@FXML
private Button modify;
@FXML
private TextField tele ;

@FXML
private TableView<Livreur> mytable ;

@FXML
private TableColumn<Livreur ,Long> col_id ;

@FXML
private TableColumn <Livreur ,String> col_nom ;

@FXML
private TableColumn <Livreur ,String> col_tele ;
```

On étudie Un scenario d'ajout des donnees dans une bdd a travers une interface

Pour un utilisateur :

Remplie les infos dans les 2 textfield après il les constate dans le tableau

Dans le programme :

```
@FXML
protected void onSaveButtonClick() {

    // acces a la bdd

    try {
        LivreurDAO livreurDAO = new LivreurDAO();

        Livreur liv = new Livreur(01 , nom.getText() , tele.getText());

        livreurDAO.save(liv);

        UpdateTable();
```

Le programme récupère les 2 donnees et fait appeler à 2 fonctions **Save et update**

Save :

```
@Override
public void save(Livreur object) throws SQLException {
    String request = "insert into livreur (nom , telephone) values (?, ?)";
    // mapping objet table
    this.preparedStatement = this.connection.prepareStatement(request);
    // mapping
    this.preparedStatement.setString(1 , object.getNom());
    this.preparedStatement.setString(2 , object.getTelephone());
    this.preparedStatement.execute();
}
```

➔ Insertion dans la bdd .

Update :

```
public void UpdateTable() {
    col_id.setCellValueFactory(new
    PropertyValueFactory<Livreur,Long>("id_livreur"));
    col_nom.setCellValueFactory(new
    PropertyValueFactory<Livreur,String>("nom"));

    col_tele.setCellValueFactory(new
    PropertyValueFactory<Livreur,String>("telephone"));

    mytable.setItems(this.getDataLivreurs());
    addButtonsToTable("Modifier",0);
    addButtonsToTable("Supprimer",1);
}
```

On fait une mise à jour sur le tableau et on ajout 2 nouveau bouton **Modifier et supprimer** qui font à leurs tours d'appeler leur fonction .

Scenario 2 : Supprimer et modifier .

```
private void addButtonToTable(String ButtonName,int btnId) {
    if(mytable.getColumns().size()==5) {
        return;
    }
    TableColumn<Livreur, Void> colBtn = new TableColumn(ButtonName);
    Button btn1 = new Button();
    Callback<TableColumn<Livreur, Void>, TableCell<Livreur, Void>>
    cellFactory = new Callback<TableColumn<Livreur, Void>, TableCell<Livreur,
    Void>>() {
        @Override
        public TableCell<Livreur, Void> call(final TableColumn<Livreur,
        Void> param) {
            final TableCell<Livreur, Void> cell = new TableCell<Livreur,
            Void>() {

```

Tracking Des Livreurs

```
private final Button btn = new Button(ButtonName);

{
    btn.setId(""+btnId);
    btn.setOnAction((ActionEvent event) -> {
        LivreurDAO livreurDAO = null;
        try {
            livreurDAO = new LivreurDAO();
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        Livreur livreur =
getTableView().getItems().get(getIndex());
        System.out.println("selectedData: " + livreur);
        if (Objects.equals(btn.getId(), "1")){

            try {
                livreurDAO.delete(livreur);
                UpdateTable();
            } catch (SQLException e) {
                throw new RuntimeException(e);
            }
        }else if(Objects.equals(btn.getId(), "0")){
            save.setDisable(true);
            modify.setDisable(false);
            idproduitSelectionne=livreur.getId_livreur();
        }
    });
}

@Override
public void updateItem(Void item, boolean empty) {
    super.updateItem(item, empty);
    if (empty) {
        setGraphic(null);
    } else {
        setGraphic(btn);
    }
}

};
return cell;
}

};
```

L'ajout des 2 buttons dans une ligne d'un tableau avec l'étude de cas

Modifier

```
public void modify(Livreur object) throws SQLException {
    String request = "update livreur set telephone = ?, nom=? where
id_livreur =?" ;
    this.preparedStatement=this.connection.prepareStatement(request);
    this.preparedStatement.setFloat(1,parseFloat(object.getTelephone()));
    this.preparedStatement.setString(2,object.getNom());
    this.preparedStatement.setLong(3,object.getId_livreur());

    this.preparedStatement.execute();
}
```

supprimer

```
public void delete(Livreur object) throws SQLException {
    String request = "DELETE FROM livreur WHERE id_livreur = ?";
    System.out.println(object);
    // mapping objet table
    this.preparedStatement = this.connection.prepareStatement(request);
    // mapping
    this.preparedStatement.setLong(1, object.getId_livreur());
    this.preparedStatement.execute();
}
```

et le code s'exécute de la même manière que **Save** .

On a ajouté des buttons dans les fichiers FXML pour faciliter la navigation mais j'ai laissé Hello application de tous les fichiers si vous voulez tester chaque fichier individuellement

Pour naviguer entre les fichiers :

```
public void change() throws IOException
{
    Parent root =
FXMLLoader.load(getClass().getResource("Livreurscene.fxml"));
    Stage window = (Stage) btnliv.getScene().getWindow();
    window.setScene(new Scene(root, 800, 500));
}
```

on aussi la destination et Régler les dimensions .

Conclusion

En conclusion, ce projet a été une expérience très enrichissante pour moi . J'ai atteint les objectifs que J'ai fixe en utilisant des méthodes efficaces. J'ai également pu relever des défis et surmonter des obstacles tout au long du projet, ce qui ma permis de développer de nouvelles compétences et de renforcer ma capacité à travailler avec pression.

Les résultats obtenus sont significatifs et répondent aux attentes fixées au début du projet. J'ai pu identifier des opportunités d'amélioration et proposer des solutions concrètes pour y remédier. J'ai également mis en place des outils et des méthodes pour assurer la durabilité de mon projet et pour faciliter sa mise en œuvre.il reste des petits trucs que je vais les faire pour augmenter le projet .