

UNIVERSITÉ CADI AYAD
ECOLE SUPERIEURE DE TECHNOLOGIE-SAFI
DUT GÉNIE INFORMATIQUE

Compte rendu

TP 1 : Système de Gestion des Employés
(MVC, DAO, Swing)

Réalisée par :

SAISSI Zahra

Enseigné par :

M.EL ABDELLAOUI Said

Année universitaire :2024/2025

Table de matières:

Introduction:	1
Objectif:	1
Architecture du Projet:	1
Lien GitHub:	1
Étapes de Création du Projet	2
1. 1ère étape : Couche Model - Implémentation du Modèle	2
2. 2ème étape : Couche DAO - Création de la connexion	2
3. 3ème étape : Interface EmployeeDAOI	3
4. 4ème étape : Classe EmployeeDAOImpl	4
5. 5ème étape : Couche Model - EmployeeModel	5
6. 6ème étape : Couche View - Interface graphique	6
7. 7ème étape : Couche Controller - Implémentation du Contrôleur	7
8. 8ème étape : Main - Application principale	8
Réalisation:	9
1. Page d'Accueil:	9
2. Afficher un Employé:	10
3. Ajouter un Employé:	10
4. Supprimer un Employé:	11
5. Modifier un Employé:	12
Conclusion:	13

Introduction:

Ce rapport présente le développement d'une application de gestion des employés en Java, en adoptant l'architecture MVC (Modèle-Vue-Contrôleur). Cette approche permet de structurer le code en séparant la logique métier, la gestion des données et l'affichage. De plus, une couche DAO (Data Access Object) est utilisée pour gérer les interactions avec les données, assurant ainsi une meilleure organisation et maintenabilité du projet.

L'objectif principal de cette application est de répondre aux besoins de l'entreprise SEA en centralisant les informations relatives aux employés. Ces informations incluent leurs noms, postes, rôles et statuts. L'application propose également des fonctionnalités essentielles, telles que l'ajout, la modification et la suppression des employés, ainsi que la recherche selon des critères spécifiques.

Objectif:

- Concevoir une application robuste et fonctionnelle en s'appuyant sur l'architecture **MVC**.
- Permettre la gestion centralisée des employés, incluant leurs rôles et postes.
- Séparer les différentes couches du projet en utilisant des classes **DAO** pour l'accès aux données.
- Fournir une interface graphique intuitive pour faciliter les interactions utilisateur.

Architecture du Projet:

L'application repose sur l'architecture **MVC**, qui est divisée en trois couches principales :

- **Modèle** : Cette couche représente les données et les règles de gestion. Elle utilise la couche **DAO** pour accéder aux données stockées, telles que les informations sur les employés, les rôles et les postes.
- **Vue** : Cette couche correspond à l'interface graphique de l'application. Elle permet d'afficher les informations et de recueillir les interactions de l'utilisateur.
- **Contrôleur** : La couche contrôle relie le Modèle et la Vue. Elle gère les actions de l'utilisateur, transmet les données à la Vue, et applique les modifications sur le Modèle.

Lien GitHub:

<https://github.com/saissizahra/GestionRessourcesHumain.git>

Étapes de Création du Projet

1. 1ère étape : Couche Model - Implémentation du Modèle

Je commence par créer les classes et énumérations nécessaires pour représenter les données. Les énumérations Poste et Role définissent les différents rôles et postes possibles pour les employés. La classe Employee contient les attributs et méthodes nécessaires pour gérer les informations des employés, avec un constructeur et des accesseurs (getters et setters).

Code :

```
1 package Model;
2
3 public class Employee {
4     private int id;
5     private String nom;
6     private String prenom;
7     private String email;
8     private String phone;
9     private double salaire;
10    private Role role;
11    private Poste poste;
12
13    public Employee(String nom, String prenom, String email, String phone, double salaire, Role role, Poste poste) {
14
15        this.nom = nom;
16        this.prenom = prenom;
17        this.email = email;
18        this.phone = phone;
19        this.salaire = salaire;
20        this.role = role;
21        this.poste = poste;
22    }
23
24    public Employee() {
25
26    }
27
28    public int getId() { return id; }
29    public void setId(int id) { this.id = id; }
30    public String getNom() { return nom; }
31    public void setNom(String nom) { this.nom = nom; }
32    public String getPrenom() { return prenom; }
33    public void setPrenom(String prenom) { this.prenom = prenom; }
34    public String getEmail() { return email; }
35    public void setEmail(String email) { this.email = email; }
36    public String getPhone() { return phone; }
37    public void setPhone(String phone) { this.phone = phone; }
38    public double getSalaire() { return salaire; }
39    public void setSalaire(double salaire) { this.salaire = salaire; }
40    public Role getRole() { return role; }
41    public void setRole(Role role) { this.role = role; }
42    public Poste getPoste() { return poste; }
43    public void setPoste(Poste poste) { this.poste = poste; }
44
45    public void setnom(String nouveauNom) {
46    }
47 }
```

```
1 package Model;
2
3 public enum Poste {
4     INGENIEURE_ETUDE_ET_DEVELOPPEMENT,
5     TEAM_LEADER,
6     PILOTE
7 }
```

```
1 package Model;
2
3 public enum Role {
4     ADMIN,
5     EMPLOYE
6 }
```

2. 2ème étape : Couche DAO - Création de la connexion

2.1. Création des tables dans la base de données

Je conçois les tables nécessaires dans une base de données MySQL : **Employee**, **Poste**, et **Role**, qui contiendront les informations essentielles sur les employés et leurs rôles.

Code SQL :

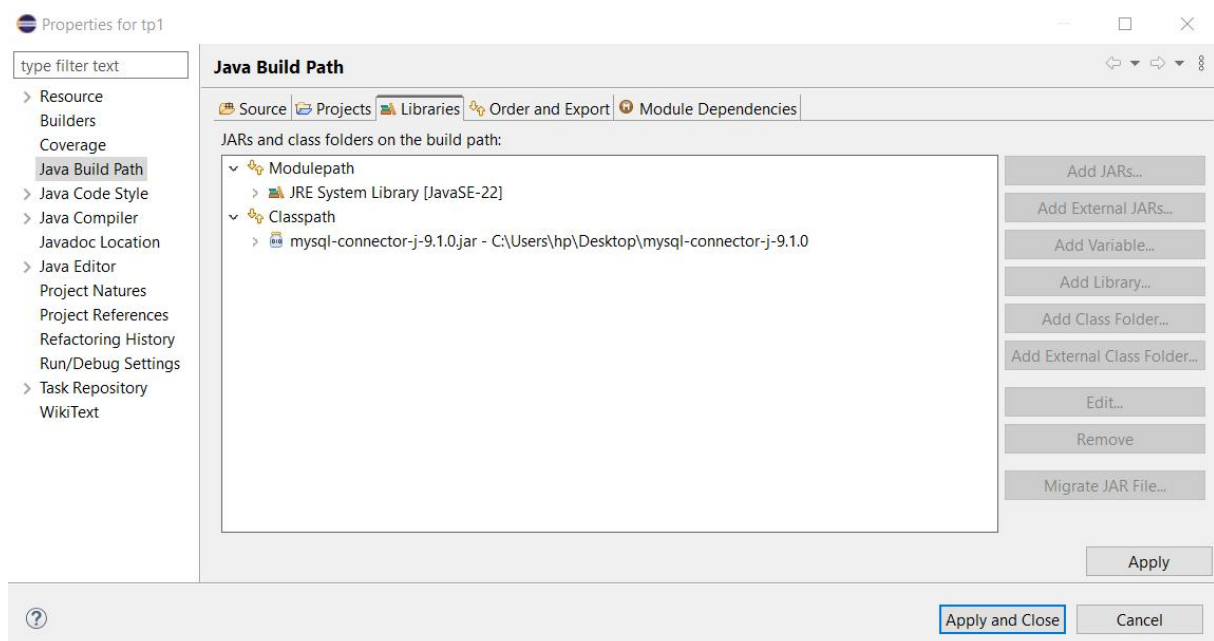
```

1 CREATE DATABASE TP7;
2 USE TP7;
3
4 CREATE TABLE Employe (
5     id INT AUTO_INCREMENT PRIMARY KEY,
6     nom VARCHAR(50) NOT NULL,
7     prenom VARCHAR(50) NOT NULL,
8     email VARCHAR(100) UNIQUE NOT NULL,
9     phone VARCHAR(15) NOT NULL,
10    salaire DOUBLE NOT NULL,
11    role ENUM('ADMIN', 'EMPLOYE') NOT NULL,
12    poste ENUM('INGENIEURE_ETUDE_ET_DEVELOPPEMENT',
13              'TEAM_LEADER',
14              'PILOTE') NOT NULL
15 );

```

2.2. Ajout du pilote JDBC de MySQL

J'ajoute le fichier JAR du pilote JDBC MySQL à mon projet pour établir la connexion entre l'application et la base de données.



2.3. Création de la classe pour la connexion à la base de données

Je crée une classe **DBConnection** dans le dossier **DAO** pour gérer la connexion sécurisée à la base de données.

Code :

```

1 package DAO;
2
3 import java.sql.Connection;
4
5
6 public class DBConnection {
7     private static final String URL = "jdbc:mysql://localhost:3306/TP7";
8     private static final String USER = "root";
9     private static final String PASSWORD = "";
10
11     public static Connection getConnection() throws SQLException {
12         return DriverManager.getConnection(URL, USER, PASSWORD);
13     }
14 }
15

```

3. 3ème étape : Interface EmployeeDAOI

Je définis une interface **EmployeeDAOI** qui contient les méthodes nécessaires pour gérer les données des employés (ajouter, supprimer, lister, etc.).

Code :

```
1 package DAO;
2
3 import Model.Employee;
4
5
6 public interface EmployeeDAOI {
7     void add(Employee employee); // Ajouter un employé
8     void delete(int id); // Supprimer un employé
9     void update(Employee employee, int id); // Mettre à jour un employé
10    List<Employee> listAll(); // Lister tous les employés
11    Employee findById(int id); // Trouver un employé par ID
12 }
```

4. 4ème étape : Classe EmployeeDAOImpl

J'implémente l'interface **EmployeeDAOI** dans une classe **EmployeeDAOImpl** pour fournir une gestion concrète des données et interagir avec la base de données.

Code :

```
1 package DAO;
2
3 import Model.Employee;
4
5
6 public class EmployeeDAOImpl implements EmployeeDAOI {
7
8     @Override
9     public void add(Employee employee) {
10         String sql = "INSERT INTO Employee (nom, prenom, email, phone, salaire, role, poste) VALUES (?, ?, ?, ?, ?, ?, ?)";
11         try (Connection conn = DBConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {
12             stmt.setString(1, employee.getNom());
13             stmt.setString(2, employee.getPrenom());
14             stmt.setString(3, employee.getEmail());
15             stmt.setString(4, employee.getPhone());
16             stmt.setDouble(5, employee.getSalaire());
17             stmt.setString(6, employee.getRole().name());
18             stmt.setString(7, employee.getPoste().name());
19             stmt.executeUpdate();
20         } catch (SQLException e) {
21             e.printStackTrace();
22         }
23     }
24
25     @Override
26     public void delete(int id) {
27         String sql = "DELETE FROM Employee WHERE id = ?";
28         try (Connection conn = DBConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {
29             stmt.setInt(1, id);
30             stmt.executeUpdate();
31         } catch (SQLException e) {
32             e.printStackTrace();
33         }
34     }
35
36     @Override
37     public void update(Employee employee, int id) {
38         String sql = "UPDATE Employee SET nom = ?, prenom = ?, email = ?, phone = ?, salaire = ?, role = ?, poste = ? WHERE id = ?";
39         try (Connection conn = DBConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {
40             stmt.setString(1, employee.getNom());
41             stmt.setString(2, employee.getPrenom());
42             stmt.setString(3, employee.getEmail());
43             stmt.setString(4, employee.getPhone());
44             stmt.setDouble(5, employee.getSalaire());
45             stmt.setString(6, employee.getRole().name()); // Envoi du rôle en tant que chaîne (avec la méthode .name())
46             stmt.setString(7, employee.getPoste().name()); // Idem pour le poste
47             stmt.setInt(8, id); // L'ID de l'employé à mettre à jour
48
49             int rowsUpdated = stmt.executeUpdate();
50
51             if (rowsUpdated > 0) {
52                 System.out.println("L'employé a été mis à jour avec succès.");
53             } else {
54                 System.out.println("Aucun employé trouvé avec cet ID.");
55             }
56         } catch (SQLException e) {
57             e.printStackTrace();
58         }
59     }
60 }
```

```

68  @Override
69  public List<Employee> listAll() {
70      List<Employee> employees = new ArrayList<>();
71      String sql = "SELECT * FROM Employee";
72      try (Connection conn = DBConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(sql);
73           ResultSet rs = stmt.executeQuery()) {
74
75          while (rs.next()) {
76              String roleStr = rs.getString("role").toUpperCase();
77              String posteStr = rs.getString("poste").toUpperCase();
78
79              Role role = null;
80              Poste poste = null;
81              try {
82                  role = Role.valueOf(roleStr);
83              } catch (IllegalArgumentException e) {
84                  System.out.println("Role non valide : " + roleStr);
85                  role = Role.EMPLOYEE; // Valeur par défaut
86              }
87
88              try {
89                  poste = Poste.valueOf(posteStr);
90              } catch (IllegalArgumentException e) {
91                  System.out.println("Poste non valide : " + posteStr);
92                  poste = Poste.INGENIEUR_ETUDE_ET_DEVELOPPEMENT; // Valeur par défaut
93              }
94
95              Employee employee = new Employee(
96                  rs.getString("nom"),
97                  rs.getString("prenom"),
98                  rs.getString("email"),
99                  rs.getString("phone"),
100                  rs.getDouble("salaire"),
101                  role,
102                  poste
103              );
104              employee.setId(rs.getInt("id"));
105              employees.add(employee);
106          }
107      } catch (SQLException e) {
108          e.printStackTrace();
109      }
110      return employees;
111  }
112  }

```

```

116
117  @Override
118  public Employee findById(int id) {
119      String sql = "SELECT * FROM Employee WHERE id = ?";
120      try (Connection conn = DBConnection.getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {
121          stmt.setInt(1, id);
122          ResultSet rs = stmt.executeQuery();
123          if (rs.next()) {
124              return new Employee(
125                  rs.getString("nom"),
126                  rs.getString("prenom"),
127                  rs.getString("email"),
128                  rs.getString("phone"),
129                  rs.getDouble("salaire"),
130                  Role.valueOf(rs.getString("role")),
131                  Poste.valueOf(rs.getString("poste"))
132              );
133          }
134      } catch (SQLException e) {
135          e.printStackTrace();
136      }
137      return null;
138  }
139
140  }

```

5. 5ème étape : Couche Model - EmployeeModel

```

1  package Model;
2  import DAO.EmployeeDAOImpl;
3  public class EmployeeModel
4  {
5      private EmployeeDAOImpl dao;
6      public EmployeeModel(EmployeeDAOImpl dao){
7          this.dao = dao;
8      }
9      public boolean addEmployee(String nom, String prenom, String email, String telephone, double salaire, Role role, Poste poste){
10         if(salaire <= 0){
11             System.out.println("erreur : salaire doit etre > 0");
12             return false;
13         }
14         if( email == null || !email.contains("@")){
15
16             System.out.println("Email invalid");
17             return false;
18         }
19         Employee nouveauEmployee = new Employee(nom, prenom, email, telephone, salaire, role, poste);
20         dao.add(nouveauEmployee);
21         return true;
22     }
23
24 }

```


6. 6ème étape : Couche View - Interface graphique

Je conçois l'interface graphique pour afficher les informations et permettre les interactions avec l'utilisateur. J'utilise des composants comme des tableaux, des boutons et des champs de texte pour implémenter les différentes fonctionnalités.

Code :

```
1 package View;
2
3 import javax.swing.*;
4
5
6 public class EmployeeView extends JFrame {
7     public JTable employeeTable;
8     public JButton addButton, listButton, deleteButton, modifyButton;
9     public JTextField nameField, surnameField, emailField, phoneField, salaryField;
10    public JComboBox<String> roleCombo, posteCombo;
11
12    public EmployeeView() {
13        setTitle("Gestion des Employés");
14        setSize(800, 600);
15        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16        setLayout(new BorderLayout());
17
18        JPanel inputPanel = new JPanel(new GridLayout(0, 2, 10, 10));
19
20        inputPanel.add(new JLabel("Nom:"));
21        nameField = new JTextField();
22        inputPanel.add(nameField);
23
24        inputPanel.add(new JLabel("Prénom:"));
25        surnameField = new JTextField();
26        inputPanel.add(surnameField);
27
28        inputPanel.add(new JLabel("Email:"));
29        emailField = new JTextField();
30        inputPanel.add(emailField);
31
32        inputPanel.add(new JLabel("Téléphone:"));
33        phoneField = new JTextField();
34        inputPanel.add(phoneField);
35
36        inputPanel.add(new JLabel("Salaire:"));
37        salaryField = new JTextField();
38        inputPanel.add(salaryField);
39
40        inputPanel.add(new JLabel("Rôle:"));
41        roleCombo = new JComboBox<>(new String[]{"Admin", "Employe"});
42        inputPanel.add(roleCombo);
43
44        inputPanel.add(new JLabel("Poste:"));
45        posteCombo = new JComboBox<>(new String[]{"INGENIEURE_ETUDE_ET_DEVELOPPEMENT", "TEAM_LEADER", "PILOTE"});
46        inputPanel.add(posteCombo);
47
48        add(inputPanel, BorderLayout.NORTH);
49
50        employeeTable = new JTable();
51        add(new JScrollPane(employeeTable), BorderLayout.CENTER);
52
53        JPanel buttonPanel = new JPanel();
54        addButton = new JButton("Ajouter");
55        buttonPanel.add(addButton);
56        listButton = new JButton("Afficher");
57        buttonPanel.add(listButton);
58        deleteButton = new JButton("Supprimer");
59        buttonPanel.add(deleteButton);
60        modifyButton = new JButton("Modifier");
61        buttonPanel.add(modifyButton);
62
63
64        add(buttonPanel, BorderLayout.SOUTH);
65    }
66 }
```


7. 7ème étape : Couche Controller - Implémentation du Contrôleur

J'ai créé une classe **EmployeeController** pour gérer les interactions entre l'interface graphique et le modèle. Cette classe écoute les événements de l'interface graphique et coordonne les actions avec le modèle.

Code :

```
1 package Controller;
2
3 import DAO.EmployeeDAO;
4 import Model.Employee;
5 import Model.EmployeeModel;
6 import Model.Poste;
7 import Model.Role;
8 import View.EmployeeView;
9
10 import javax.swing.*;
11 import javax.swing.table.DefaultTableModel;
12 import java.awt.event.ActionEvent;
13 import java.awt.event.ActionListener;
14
15 import java.util.List;
16
17 public class EmployeeController {
18
19     private EmployeeView view;
20     private EmployeeDAO dao;
21     private EmployeeModel model;
22
23     public EmployeeController(EmployeeView view, EmployeeModel model) {
24         this.view = view;
25         this.model = model;
26
27         // Écouteur pour le bouton Ajouter
28         view.addButton.addActionListener(new ActionListener() {
29             @Override
30             public void actionPerformed(ActionEvent e) {
31                 addEmployee();
32             }
33         });
34
35         // Écouteur pour le bouton Lister
36         view.listButton.addActionListener(new ActionListener() {
37             @Override
38             public void actionPerformed(ActionEvent e) {
39                 listEmployees();
40             }
41         });
42
43         // Écouteur pour le bouton Supprimer
44         view.deleteButton.addActionListener(new ActionListener() {
45             @Override
46             public void actionPerformed(ActionEvent e) {
47                 deleteEmployee();
48             }
49         });
50
51         // Écouteur pour le bouton Modifier
52         view.modifyButton.addActionListener(new ActionListener() {
53             @Override
54             public void actionPerformed(ActionEvent e) {
55                 modifyEmployee();
56             }
57         });
58
59         // Méthode pour ajouter un employé
60         private void addEmployee() {
61             try {
62                 String nom = view.nameField.getText();
63                 String prenom = view.surnameField.getText();
64                 String email = view.emailField.getText();
65                 String phone = view.phoneField.getText();
66                 double salaire = Double.parseDouble(view.salaryField.getText());
67                 Role role = Role.valueOf(view.roleCombo.getSelectedItem().toString().toUpperCase());
68                 Poste poste = Poste.valueOf(view.posteCombo.getSelectedItem().toString().toUpperCase());
69
70                 Employee employee = new Employee(nom, prenom, email, phone, salaire, role, poste);
71                 dao.add(employee);
72                 JOptionPane.showMessageDialog(view, "Employé ajouté avec succès.");
73             } catch (Exception ex) {
74                 JOptionPane.showMessageDialog(view, "Erreur: " + ex.getMessage());
75             }
76         }
77     }
78 }
```

```

77
78 // Méthode pour afficher la liste des employés
79 private void listEmployees() {
80     List<Employee> employees = dao.listAll();
81     String[] columnNames = {"ID", "Nom", "Prénom", "Email", "Téléphone", "Salaire", "Rôle", "Poste"};
82     DefaultTableModel model = new DefaultTableModel(columnNames, 0);
83
84     for (Employee emp : employees) {
85         Object[] row = {emp.getId(), emp.getNom(), emp.getPrenom(), emp.getEmail(), emp.getPhone(), emp.getSalaire()};
86         model.addRow(row);
87     }
88
89     view.employeeTable.setModel(model);
90 }
91
92 // Méthode pour supprimer un employé
93 private void deleteEmployee() {
94     try {
95         int id = Integer.parseInt(JOptionPane.showInputDialog(view, "Entrez l'ID de l'employé à supprimer :"));
96         dao.delete(id);
97         JOptionPane.showMessageDialog(view, "Employé supprimé avec succès.");
98     } catch (Exception ex) {
99         JOptionPane.showMessageDialog(view, "Erreur: " + ex.getMessage());
100     }
101 }
102
103 // Méthode pour modifier un employé
104 private void modifyEmployee() {
105     try {
106         int selectedRow = view.employeeTable.getSelectedRow();
107         if (selectedRow == -1) {
108             JOptionPane.showMessageDialog(view, "Veuillez sélectionner un employé dans le tableau.");
109             return;
110         }
111
112         int id = (int) view.employeeTable.getValueAt(selectedRow, 0);
113
114         String nom = view.nameField.getText();
115         String prenom = view.surnameField.getText();
116         String email = view.emailField.getText();
117         String phone = view.phoneField.getText();
118         double salaire = Double.parseDouble(view.salaryField.getText());
119         Role role = Role.valueOf(view.roleCombo.getSelectedItem().toString().toUpperCase());
120         Poste poste = Poste.valueOf(view.posteCombo.getSelectedItem().toString().toUpperCase());
121
122         Employee updatedEmployee = new Employee(nom, prenom, email, phone, salaire, role, poste);
123
124         dao.update(updatedEmployee, id);
125
126         JOptionPane.showMessageDialog(view, "Employé mis à jour avec succès.");
127         listEmployees();
128     } catch (Exception ex) {
129         JOptionPane.showMessageDialog(view, "Erreur: " + ex.getMessage());
130     }
131 }
132 }

```

8. 8ème étape : Main - Application principale

Dans cette dernière étape, je crée la classe principale pour instancier le modèle, la vue, et le contrôleur, puis je lance l'application.

Code :

```

1 package Main;
2
3 import Controller.EmployeeController;
4 import DAO.EmployeeDAOImpl;
5 import Model.EmployeeModel;
6 import View.EmployeeView;
7
8 public class Main {
9     public static void main(String[] args) {
10         EmployeeView view = new EmployeeView();
11         EmployeeDAOImpl dao = new EmployeeDAOImpl();
12         EmployeeModel model = new EmployeeModel(dao);
13         new EmployeeController(view,model);
14         view.setVisible(true);
15     }
16 }
17 }

```

Réalisation:

1. Page d'Accueil:

Gestion des Employés

Nom:

Prénom:

Email:

Téléphone:

Salaire:

Rôle:

Poste:

2. Afficher un Employé:

Gestion des Employés

Nom:

Prénom:

Email:

Téléphone:

Salaire:

Rôle:

Poste:

ID	Nom	Prénom	Email	Téléphone	Salaire	Rôle	Poste
1	khaoula	kouis	khaoula@gmail.c...	0123456789	1000000.0	ADMIN	INGENIEURE_E...
2	nissrin	elouati	nissrin@gmail.co...	0123456789	50000.0	EMPLOYE	INGENIEURE_E...
3	inas	ouhidouss	inas@gmail.com	0123456789	120000.0	EMPLOYE	PILOTE
4	zahra	zahra	zahra@yuuio	5676889	68990.0	EMPLOYE	TEAM_LEADER
5	jihad	zahni	jihad@gmail.com	0123456789	600000.0	ADMIN	TEAM_LEADER

Ajouter Afficher Supprimer Modifier

3. Ajouter un Employé:

Gestion des Employés

Nom:

Prénom:

Email:

Téléphone:

Salaire:

Rôle:

Poste:

ID	Nom	Prénom	Email	Téléphone	Salaire	Rôle	Poste
1	khaoula	kouis	khaoula@gmail.c...	0123456789	1000000.0	ADMIN	INGENIEURE_E...
2	nissrin	elouati	nissrin@gmail.co...	0123456789	50000.0	EMPLOYE	INGENIEURE_E...
3	inas	ouhidouss	inas@gmail.com	0123456789	120000.0	EMPLOYE	PILOTE
5	jihad	zahni	jihad@gmail.com	0123456789	600000.0	ADMIN	TEAM_LEADER

Message

Employé ajouté avec succès.

OK

4. Supprimer un Employé:

Gestion des Employés

Nom: saissi

Prénom: zahra

Email: saissizahra5@gmail.com

Téléphone: 0123456789

Salaire: 100000.0

Rôle: Admin

Poste: ET_DEVELOPPEMENT

ID	Nom	Prénom	Salaire	Rôle	Poste	
1	khaoula	kouis	0.0	ADMIN	INGENIEURE E...	
2	nissrin	elouati	0.0	EMPLOYE	INGENIEURE E...	
3	inas	ouhidouss	0.0	EMPLOYE	PILOTE	
4	zahra	zahra	0.0	EMPLOYE	TEAM_LEADER	
5	jiha	zahn	00000.0	ADMIN	TEAM_LEADER	
6	saissi	zahra	0123456789	100000.0	ADMIN	INGENIEURE E...

Ajouter Afficher Supprimer Modifier

Gestion des Employés

Nom: saissi

Prénom: zahra

Email: saissizahra5@gmail.com

Téléphone: 0123456789

Salaire: 100000.0

Rôle: Admin

Poste: ET_DEVELOPPEMENT

ID	Nom	Prénom	Salaire	Rôle	Poste	
1	khaoula	kouis	00.0	ADMIN	INGENIEURE E...	
2	nissrin	elouati	0.0	EMPLOYE	INGENIEURE E...	
3	inas	ouhidouss	0.0	EMPLOYE	PILOTE	
4	zahra	zahra	0.0	EMPLOYE	TEAM_LEADER	
5	jiha	zahn	00000.0	ADMIN	TEAM_LEADER	
6	saissi	zahra	saissizahra5@g... 0123456789	100000.0	ADMIN	INGENIEURE E...

Ajouter Afficher Supprimer Modifier

5. Modifier un Employé:

Gestion des Employés

Nom: saissi

Prénom: zahra

Email: saissizahra5@gmail.com

Téléphone: 0123456789

Salaire: 45000.0

Rôle: Employee

Poste:

ID	Nom	Prénom	Salaire	Rôle	Poste
1	khaoula	kouli	100000.0	ADMIN	INGENIEURE E...
2	nissrin	elouati	0.0	EMPLOYEE	INGENIEURE E...
3	inas	ouhidouss	0.0	EMPLOYEE	PILOTE
5	jiha	zahn	0.0	ADMIN	TEAM LEADER
6	saissi	zahra	45000.0	ADMIN	INGENIEURE E...

Message

Veuillez sélectionner un employé dans le tableau.

OK

Ajouter Afficher Supprimer Modifier

Gestion des Employés

Nom: saissi

Prénom: zahra

Email: saissizahra5@gmail.com

Téléphone: 0123456789

Salaire: 45000.0

Rôle: Employee

Poste:

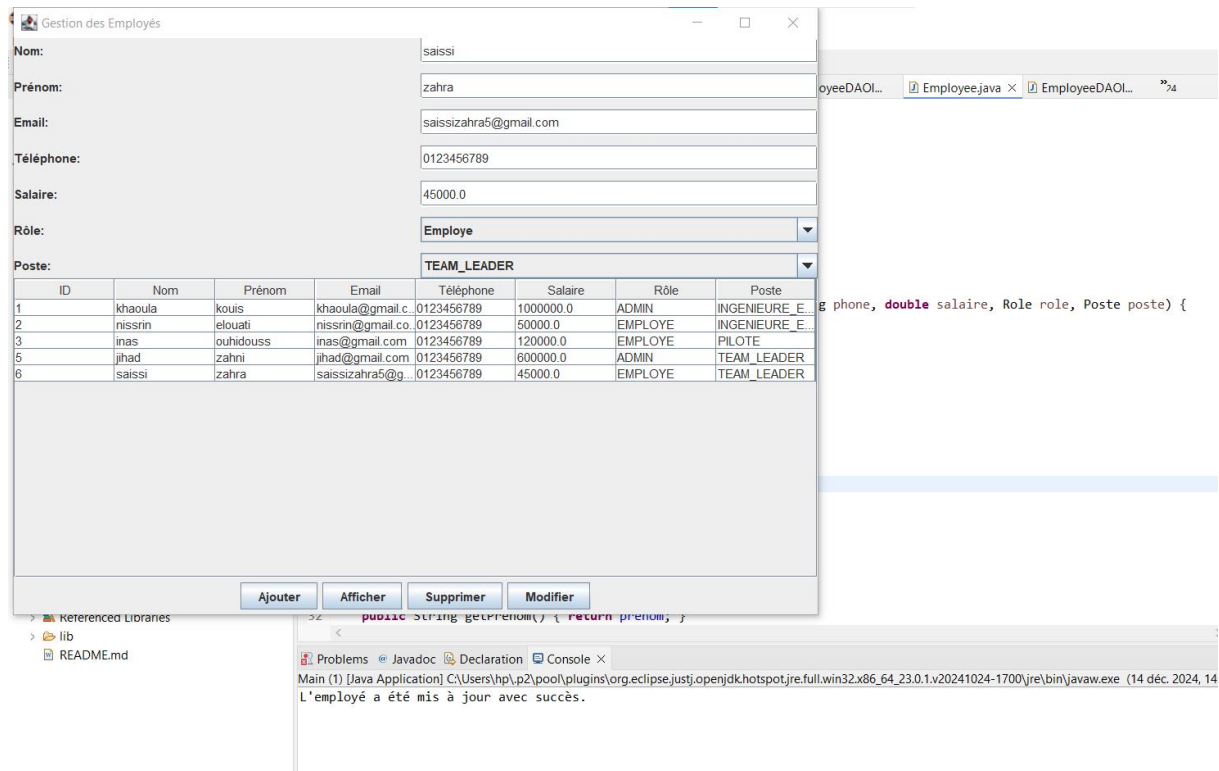
ID	Nom	Prénom	Salaire	Rôle	Poste
1	khaoula	kouli	100000.0	ADMIN	INGENIEURE E...
2	nissrin	elouati	0.0	EMPLOYEE	INGENIEURE E...
3	inas	ouhidouss	0.0	EMPLOYEE	PILOTE
5	jiha	zahn	0.0	ADMIN	TEAM LEADER
6	saissi	zahra	45000.0	ADMIN	INGENIEURE E...

Message

Employé mis à jour avec succès.

OK

Ajouter Afficher Supprimer Modifier



Gestion des Employés

Nom: saissi

Prénom: zahra

Email: saissizahra5@gmail.com

Téléphone: 0123456789

Salaire: 45000.0

Rôle: Employee

Poste: TEAM_LEADER

ID	Nom	Prénom	Email	Téléphone	Salaire	Rôle	Poste
1	khaoula	kouis	khaoula@gmail.c	0123456789	1000000.0	ADMIN	INGENIEURE E...
2	nissrin	elouati	nissrin@gmail.co	0123456789	50000.0	EMPLOYE	INGENIEURE E...
3	inas	ouhidouss	inas@gmail.com	0123456789	120000.0	EMPLOYE	PILOTE
5	ihad	zahni	ihad@gmail.com	0123456789	600000.0	ADMIN	TEAM_LEADER
6	saissi	zahra	saissizahra5@g	0123456789	45000.0	EMPLOYE	TEAM_LEADER

Ajouter Afficher Supprimer Modifier

Console:

```

Main (1) [Java Application] C:\Users\hp\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_23.0.1.v20241024-1700\jre\bin\javaw.exe (14 déc. 2024, 14
L'employé a été mis à jour avec succès.

```

Conclusion:

Ce projet illustre l'utilisation de l'architecture MVC et des DAO pour concevoir une application efficace et bien structurée. Grâce à une séparation claire des couches, l'application est facilement maintenable et évolutive. Cette expérience m'a permis de renforcer mes compétences en développement logiciel tout en découvrant l'importance de structurer et d'organiser un projet de manière rigoureuse.