Université Cadi Ayyad
École Supérieure De Technologie-Safi
Département : Informatique
Filière : genie informatique first year

gestion des employés

Rapport du tp 1(java avancé)

Réalisé par : ELHIYANI Hanane

Encadré par : Mme. KACHBAL Ilham

Année Universitaire : 2024/2025

Table des matières

Table des figures

Inroduction

Dans le cadre de notre apprentissage des concepts avancés de développement d'applications, ce TP vise à mettre en pratique le modèle architectural **MVC (Model-View-Controller)** associé au **DAO (Data Access Object)**. Le projet consiste à développer une application de gestion des employés, permettant l'ajout, la modification, la suppression dans une base de données.

L'objectif principal est de séparer les différentes couches de l'application :
- Model : Gère la logique métier et l'interaction avec la base de données.
- View : Assure l'affichage et l'interaction utilisateur.
- Controller : Traite les requêtes de l'utilisateur et met à jour le modèle et la vue en conséquence.
- DAO :(Data Access Object) est une couche du modèle chargée des opérations de persistance. Elle gère l'accès aux données en exécutant les opérations CRUD (Create, Read, Update, Delete), tout en séparant la logique métier de l'accès à la base de données.

L'utilisation du DAO garantit une gestion centralisée et sécurisée des opérations sur les données, favorisant ainsi la maintenance et l'évolution de l'application. Ce TP permettra de mieux comprendre la structuration d'une application selon les bonnes pratiques du développement logiciel.

Outils & environnement de travail

1 Language de programmation

FIGURE 1 – JAVA logo

- **Java** : est un langage de programmation orienté objet, sécurisé, portable et indépendant de la plateforme. Il est utilisé pour développer des applications web, mobiles et de bureau, grâce à sa machine virtuelle Java (JVM), qui permet d'exécuter le même code sur différentes plateformes.

2 Environnement de travail

FIGURE 2 – intellij idea logo

- **intellij idea** : est un environnement de développement intégré (IDE) puissant et populaire, principalement utilisé pour le développement en Java. Développé par JetBrains, il offre des fonctionnalités avancées telles que l'autocomplétion intelligente, la navigation dans le code, le débogage, le contrôle de version intégré et la prise en charge de plusieurs langages de programmation. IntelliJ IDEA est apprécié pour sa productivité accrue et son interface conviviale.

FIGURE 3 – xampp logo

- **xampp** : En parallèle, le projet vise à fournir des outils de gestion robustes pour le corps administratif, avec une fonctionnalité de multi-rôle, permettant à chaque agent d'accéder à un compte adapté à ses responsabilités spécifique

- **MySQL Workbench** : est un outil visuel de gestion de bases de données développé par Oracle. Il permet de concevoir, modéliser, administrer et interagir avec des bases de données MySQL.

Elaboration & Réalisation

1 Création de la base de donnée

```
//
CREATE DATABASE EmployeDB;

//
USE EmployeDB;
//

CREATE TABLE Employes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    phone_number VARCHAR(20),
    salary DECIMAL(10, 2),
    role VARCHAR(50),
    poste VARCHAR(50)
);
//
CREATE TABLE Role (
  name varchar(30) NOT NULL
);
//
CREATE TABLE Poste (
  name varchar(30) NOT NULL
);
```

Listing 1.1 – Script SQL de la base de données

2 Implémentation de l'architecture MVC avec le pattern DAO

Les étapes pratiques suivies dans ce TP sont organisées selon l'architecture **MVC (Model-View-Controller)** combinée au **pattern DAO (Data Access Object)** pour assurer une bonne séparation des responsabilités. Après la création de la base de données, nous commençons par la gestion des données, en implémentant le DAO pour interagir avec la base. Les étapes sont les suivantes :

- **Étape 1 : Implémentation du modèle (couche Model)**
  Création de la classe `Employe` représentant l'entité métier, avec ses attributs, ses getters et setters.
- **Étape 2 : Gestion des données (DAO)**
  Création de l'interface `EmployerInterface` définissant les opérations CRUD, suivie de l'implémentation de la classe `EmployerDAO` pour la gestion des données de la base.
- **Étape 3 : Logique métier**
  Développement de la classe `EmployerLogic` contenant les règles métier et la gestion des opérations complexes.
- **Étape 4 : Interface graphique (couche View)**
  Création de l'interface utilisateur en utilisant `JTabbedPane` pour afficher les données et interagir avec l'application.
- **Étape 5 : Contrôleur (couche Controller)**
  Gestion des événements de l'interface utilisateur pour relier la vue et la logique métier.
- **Étape 6 : Main**
  Initialisation de l'application et lancement de l'interface principale.

Ces étapes assurent une application bien structurée, évolutive et facile à maintenir.

3 Couche Model

Employer

```java
package model;

import enums.*;

public class Employer {

    private int id;
    private String firstName;
    private String lastName;
    private String email;
    private int phoneNumber;
    private double salary;
    private Role role;
    private Poste poste;

```

```
    public Employer(int id, String firstName, String lastName, String email, int phoneNumber, double salary, Role role, Poste poste) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.phoneNumber = phoneNumber;
        this.salary = salary;
        this.role = role;
        this.poste = poste;
    }

    public int getId() {
        return id;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getEmail() {
        return email;
    }

    public int getPhoneNumber() {
        return phoneNumber;
    }

    public double getSalary() {
        return salary;
    }

    public Role getRole() {
        return role;
    }

    public Poste getPoste() {
        return poste;
    }
}
```

EmployerLogic

```
package model;
import enums.*;
import dao.EmployerDAO;
import java.util.List;
public class EmployerLogic {

    private EmployerDAO dao;

```

```
    public EmployerLogic (EmployerDAO dao) {
        this.dao = dao;
    }

    public boolean addEmployer(int id, String firstName, String lastName, String email,
   int phoneNumber, double salary, Role role, Poste poste) {

        if ( isValidEmail(email) ) {
            return dao.addEmployer( new Employer(
                id,
                firstName,
                lastName,
                email,
                phoneNumber,
                salary,
                role,
                poste
            ));
        }

        return false;
    }


    public boolean updateEmployer(int id, String firstName, String lastName, String
   email, int phoneNumber, double salary, Role role, Poste poste) {


        if ( isValidEmail(email) ) {

            Employer employer = new Employer(
                id,
                firstName,
                lastName,
                email,
                phoneNumber,
                salary,
                role,
                poste
            );

            return dao.updateEmployer(employer);
        }

        return false;
    }

    private boolean isValidEmail(String email) {
        return email.contains("@gmail.com") ? true : false;
    }


    public boolean deleteEmployer(int id) {
        return dao.deleteEmployer(id);
    }
```

```
62
63      public List<Employer> getAllEmployers() {
64          return dao.getAllEmployers();
65      }
66
67  }
```

4 Couche DAO

DBConnection

```
1  package dao;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.SQLException;
6
7  public class DBConnection {
8
9      private static final String URL = "jdbc:mysql://localhost:3306/
       gestion_employes";
10     private static final String USERNAME = "root";
11     private static final String PASSWORD = "";
12
13     public static Connection getConnection() throws SQLException {
14         return DriverManager.getConnection(URL, USERNAME, PASSWORD);
15     }
16 }
```

EmployerDAO

```
1  package dao;
2
3  import java.sql.*;
4  import java.util.ArrayList;
5  import java.util.List;
6  import model.Employer;
7  import enums.Role;
8  import enums.Poste;
9
10 public class EmployerDAO implements EmployerInterface {
11
12     private Connection connection;
13
14     public EmployerDAO() {
15         try {
16             connection = DBConnection.getConnection();
17         } catch (SQLException connectionException) {
18             connectionException.printStackTrace();
19         }
20     }
21
```

```
22
23      @Override
24      public boolean addEmployer(Employer employer) {
25          try (PreparedStatement addStatement = connection.prepareStatement(
26              "INSERT INTO employers (first_name, last_name, email, phone, salary,
    role, poste) VALUES (?, ?, ?, ?, ?, ?, ?)")) {
27
28              addStatement.setString(1, employer.getFirstName());
29              addStatement.setString(2, employer.getLastName());
30              addStatement.setString(3, employer.getEmail());
31              addStatement.setInt(4, employer.getPhoneNumber());
32              addStatement.setDouble(5, employer.getSalary());
33              addStatement.setString(6, employer.getRole().name());
34              addStatement.setString(7, employer.getPoste().name());
35
36              return addStatement.executeUpdate() > 0;
37
38          } catch (SQLException addException) {
39              addException.printStackTrace();
40              return false;
41          }
42      }
43
44
45      @Override
46      public boolean updateEmployer(Employer employer) {
47          try (PreparedStatement updateStatement = connection.prepareStatement("
    UPDATE employers SET first_name = ?, last_name = ?, email = ?, phone = ?,
    salary = ?, role = ?, poste = ? WHERE id = ?")) {
48
49              updateStatement.setString(1, employer.getFirstName());
50              updateStatement.setString(2, employer.getLastName());
51              updateStatement.setString(3, employer.getEmail());
52              updateStatement.setInt(4, employer.getPhoneNumber());
53              updateStatement.setDouble(5, employer.getSalary());
54              updateStatement.setString(6, employer.getRole().name());
55              updateStatement.setString(7, employer.getPoste().name());
56              updateStatement.setInt(8, employer.getId());
57
58              return updateStatement.executeUpdate() > 0;
59
60          } catch (SQLException updateException) {
61              updateException.printStackTrace();
62              return false;
63          }
64      }
65
66      @Override
67      public boolean deleteEmployer(int id) {
68          try (PreparedStatement deleteStatement = connection.prepareStatement("
    DELETE FROM employers WHERE id = ?")) {
69
70              deleteStatement.setInt(1, id);
71              return deleteStatement.executeUpdate() > 0;
72
```

```
        } catch (SQLException deleteException) {
            return false;
        }
    }

    @Override
    public List<Employer> getAllEmployers() {
        List<Employer> employers = new ArrayList<>();
        try (ResultSet getResult = connection.prepareStatement("SELECT * FROM
   employers").executeQuery()) {

            while (getResult.next()) {
                employers.add(new Employer(
                    getResult.getInt("id"),
                    getResult.getString("first_name"),
                    getResult.getString("last_name"),
                    getResult.getString("email"),
                    getResult.getInt("phone"),
                    getResult.getDouble("salary"),
                    Role.valueOf(getResult.getString("role")),
                    Poste.valueOf(getResult.getString("poste"))
                ));
            }

        } catch (SQLException getException) {
            getException.printStackTrace();
        }
        return employers;
    }
}
```

EmployerInterface

```
package dao;

import java.util.List;
import model.Employer;

public interface EmployerInterface {
    boolean addEmployer(Employer employer);
    boolean updateEmployer(Employer employer);
    boolean deleteEmployer(int id);
    List<Employer> getAllEmployers();
}
```

5 Couche Controller

EmployerController

```
package controller;

import view.*;
import dao.*;
import model.*;
import enums.*;
import java.util.List;
import javax.swing.JOptionPane;
public class EmployerController {

    private FormFrame frame;
    private EmployerLogic employerLogic;

    public EmployerController(FormFrame frame, EmployerLogic employerLogic) {
        this.frame = frame;
        this.employerLogic = employerLogic;

        frame.getBtnPanel().getAddBtn().addActionListener(addEvent -> addEmployer
    ());
        frame.getBtnPanel().getUpdateBtn().addActionListener(updateEvent ->
    updateEmployer());
        frame.getBtnPanel().getRemoveBtn().addActionListener(deleteEvent ->
    deleteEmployer());
        loadEmployers();
     }

    private void addEmployer() {
        try {

            if (employerLogic.addEmployer(
                    1,
                    frame.getInPanel().getFirstNameField().getText(),
                    frame.getInPanel().getLastNameField().getText(),
                    frame.getInPanel().getEmailField().getText(),
                    Integer.parseInt(frame.getInPanel().getTelephoneNumberField().
    getText()),
                    Double.parseDouble(frame.getInPanel().getSalaryField().getText
    ()),
                    Role.valueOf(frame.getInPanel().getSelectedRole().toString()),
                    Poste.valueOf(frame.getInPanel().getSelectedPoste().toString()
    )
                ))
            {
                JOptionPane.showMessageDialog(frame, "Employer added successfully
    .");
                loadEmployers();
            } else {
                JOptionPane.showMessageDialog(frame, "Failed to add employer.");
            }
        } catch (Exception e) {
```

```
44              JOptionPane.showMessageDialog(frame, "Invalid input: " + e.getMessage
    ());
45          }
46      }
47
48      private void updateEmployer() {
49       try {
50          if (employerLogic.updateEmployer(
51                  frame.getListPanel().getSelectedRowId(),
52                  frame.getInPanel().getFirstNameField().getText(),
53                  frame.getInPanel().getLastNameField().getText(),
54                  frame.getInPanel().getEmailField().getText(),
55                  Integer.parseInt(frame.getInPanel().getTelephoneNumberField().
    getText()),
56                  Double.parseDouble(frame.getInPanel().getSalaryField().getText())
    ,
57                  Role.valueOf(frame.getInPanel().getSelectedRole().toString()),
58                  Poste.valueOf(frame.getInPanel().getSelectedPoste().toString())
59              ))
60          {
61              JOptionPane.showMessageDialog(frame, "Employer updated successfully
    .");
62              loadEmployers();
63          } else {
64              JOptionPane.showMessageDialog(frame, "Failed to update employer.");
65          }
66       } catch (Exception e) {
67          JOptionPane.showMessageDialog(frame, "Invalid input: " + e.getMessage());
68       }
69      }
70
71      private void deleteEmployer() {
72       try {
73          if (employerLogic.deleteEmployer(frame.getListPanel().getSelectedRowId())
    ) {
74              JOptionPane.showMessageDialog(frame, "Employer deleted successfully
    .");
75              loadEmployers();
76          } else {
77              JOptionPane.showMessageDialog(frame, "Failed to delete employer.");
78          }
79       } catch (Exception e) {
80          JOptionPane.showMessageDialog(frame, "Invalid input: " + e.getMessage());
81       }
82      }
83
84      private void loadEmployers() {
85          frame.getListPanel().updateEmployerList(employerLogic.getAllEmployers());
86      }
87  }
```

6 Couche View

BtnPanel

```
package view;

import javax.swing.*;
import java.awt.*;

public class BtnPanel extends JPanel {

    private JButton addBtn, removeBtn , updateBtn;

    public BtnPanel() {
        setLayout(new FlowLayout());

        addBtn = new JButton("Add");
        removeBtn = new JButton("Remove");
        updateBtn = new JButton("Update");


        add(addBtn);
        add(removeBtn);
        add(updateBtn);
    }

    public JButton getAddBtn() {
        return addBtn;
    }


    public JButton getRemoveBtn() {
        return removeBtn;
    }

    public JButton getUpdateBtn() {
        return updateBtn;
    }
}
```

FormFrame

```
package view;

import javax.swing.*;
import java.awt.*;

public class BtnPanel extends JPanel {

    private JButton addBtn, removeBtn , updateBtn;

    public BtnPanel() {
        setLayout(new FlowLayout());
```

```
12
13          addBtn = new JButton("Add");
14          removeBtn = new JButton("Remove");
15          updateBtn = new JButton("Update");
16
17
18          add(addBtn);
19          add(removeBtn);
20          add(updateBtn);
21      }
22
23      public JButton getAddBtn() {
24          return addBtn;
25      }
26
27
28      public JButton getRemoveBtn() {
29          return removeBtn;
30      }
31
32      public JButton getUpdateBtn() {
33          return updateBtn;
34      }
35  }
```

InputPanel

```
1  package view;
2
3  import javax.swing.*;
4  import java.awt.*;
5  import enums.Role;
6  import enums.Poste;
7
8  public class InputPanel extends JPanel {
9
10     JTextField firstNameField, lastNameField, emailField, telephoneNumberField,
       salaryField;
11     JComboBox<Role> roleField;
12     JComboBox<Poste> posteField;
13
14     public InputPanel() {
15         setLayout(new GridLayout(7, 2, 5, 5));
16         setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
17
18         firstNameField = new JTextField(15);
19         lastNameField = new JTextField(15);
20         emailField = new JTextField(15);
21         telephoneNumberField = new JTextField(15);
22         salaryField = new JTextField(15);
23         roleField = new JComboBox<>(Role.values());
24         posteField = new JComboBox<>(Poste.values());
25
26         add(new JLabel("First Name"));
```

```
27          add(firstNameField);
28
29          add(new JLabel("Last Name"));
30          add(lastNameField);
31
32          add(new JLabel("Email"));
33          add(emailField);
34
35          add(new JLabel("Telephone Number"));
36          add(telephoneNumberField);
37
38          add(new JLabel("Salary"));
39          add(salaryField);
40
41          add(new JLabel("Role"));
42          add(roleField);
43
44          add(new JLabel("Poste"));
45          add(posteField);
46      }
47
48      public JTextField getFirstNameField() {
49          return firstNameField;
50      }
51
52      public JTextField getLastNameField() {
53          return lastNameField;
54      }
55
56      public JTextField getEmailField() {
57          return emailField;
58      }
59
60      public JTextField getTelephoneNumberField() {
61          return telephoneNumberField;
62      }
63
64      public JTextField getSalaryField() {
65          return salaryField;
66      }
67
68      public Role getSelectedRole() {
69          return (Role) roleField.getSelectedItem();
70      }
71
72      public Poste getSelectedPoste() {
73          return (Poste) posteField.getSelectedItem();
74      }
75  }
```

ListPane

```
1  package view;
2
```

```
import javax.swing.*;
import javax.swing.border.LineBorder;
import java.awt.*;
import java.util.ArrayList;
import java.util.List;
import model.Employer;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class ListPanel extends JPanel {

    private JPanel contentPanel;
    private int selectedRowId = -1;
    private List<JPanel> rowPanels = new ArrayList<>();

    public ListPanel() {
        setLayout(new BorderLayout());

        JPanel titlePanel = new JPanel();
        titlePanel.setLayout(new GridLayout(1, 5));
        titlePanel.add(new JLabel("Id", SwingConstants.CENTER));
        titlePanel.add(new JLabel("Nom", SwingConstants.CENTER));
        titlePanel.add(new JLabel("Prenom", SwingConstants.CENTER));
        titlePanel.add(new JLabel("Email", SwingConstants.CENTER));
        titlePanel.add(new JLabel("Salaire", SwingConstants.CENTER));
        titlePanel.setBorder(new LineBorder(Color.BLACK));
        add(titlePanel, BorderLayout.NORTH);

        contentPanel = new JPanel();
        contentPanel.setLayout(new BoxLayout(contentPanel, BoxLayout.Y_AXIS));
        JScrollPane scrollPane = new JScrollPane(contentPanel);
        add(scrollPane, BorderLayout.CENTER);
    }

    public void updateEmployerList(List<Employer> employers) {
        contentPanel.removeAll();
        rowPanels.clear();

        for (Employer employer : employers) {
            JPanel rowPanel = new JPanel(new GridLayout(1, 5));
            rowPanel.setBorder(new LineBorder(Color.GRAY));
            rowPanel.setMaximumSize(new Dimension(Integer.MAX_VALUE, 30));

            JLabel idLabel = new JLabel(String.valueOf(employer.getId()),
    SwingConstants.CENTER);
            JLabel lastNameLabel = new JLabel(employer.getLastName(),
    SwingConstants.CENTER);
            JLabel firstNameLabel = new JLabel(employer.getFirstName(),
    SwingConstants.CENTER);
            JLabel emailLabel = new JLabel(employer.getEmail(), SwingConstants.
    CENTER);
            JLabel salaryLabel = new JLabel(String.valueOf(employer.getSalary()),
    SwingConstants.CENTER);

            rowPanel.addMouseListener(new MouseAdapter() {
```

```
53                  @Override
54                  public void mouseClicked(MouseEvent e) {
55                      highlightRow(rowPanel, employer.getId());
56                  }
57              });
58
59              rowPanel.add(idLabel);
60              rowPanel.add(lastNameLabel);
61              rowPanel.add(firstNameLabel);
62              rowPanel.add(emailLabel);
63              rowPanel.add(salaryLabel);
64
65              contentPanel.add(rowPanel);
66              rowPanels.add(rowPanel);
67          }
68
69          contentPanel.revalidate();
70          contentPanel.repaint();
71      }
72
73      private void highlightRow(JPanel selectedRow, int employerId) {
74          for (JPanel row : rowPanels) {
75              row.setBackground(Color.WHITE);
76          }
77
78          selectedRow.setBackground(Color.LIGHT_GRAY);
79          selectedRowId = employerId;
80      }
81
82      public int getSelectedRowId() {
83          return selectedRowId;
84      }
85  }
```

Résultats

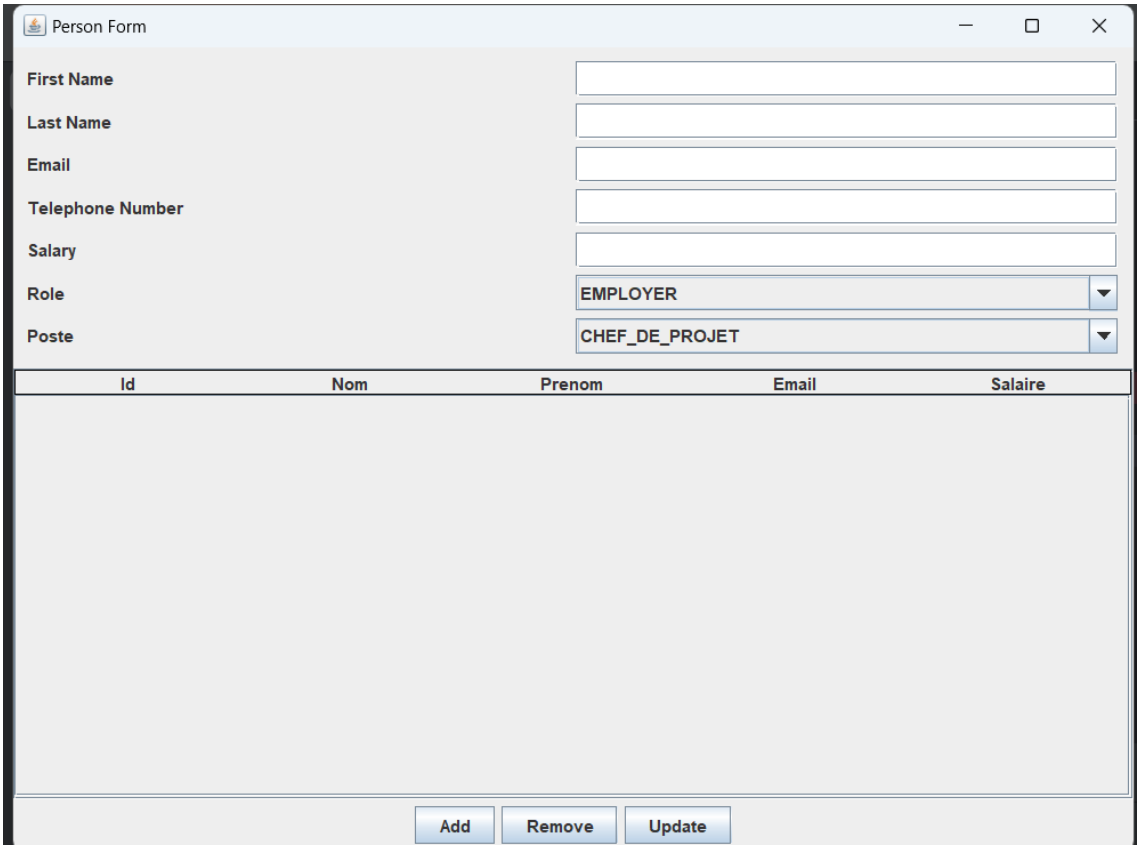

FIGURE 2.1 – Interface Utilisateur

1 Button ajouter

Lorsque l'utilisateur clique sur le bouton "Ajouter", les informations saisies sont enregistrées dans la base de données. Ensuite, ces données sont affichées automatiquement dans le tableau de l'interface graphique.

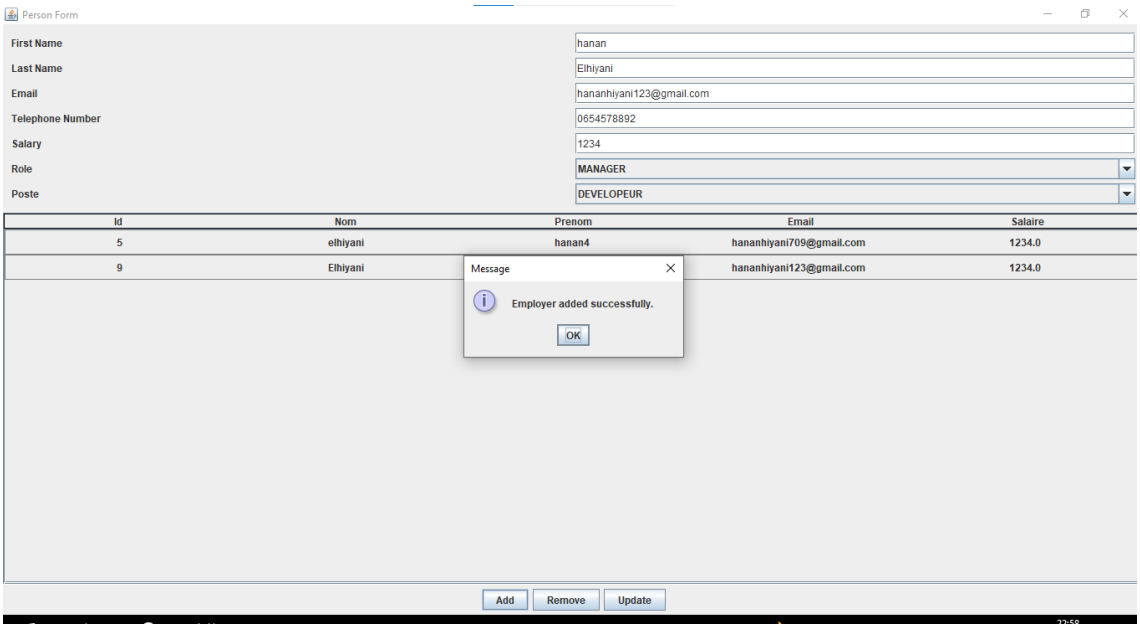

FIGURE 2.2 – Resultat Ajout

2 Button modifier

Pour modifier une entrée, l'utilisateur doit d'abord sélectionner la ligne correspondante dans le tableau de l'interface graphique. Ensuite, en cliquant sur le bouton "Modifier", les informations de cette ligne sont chargées dans les champs de saisie. Après avoir apporté les modifications nécessaires, un second clic sur le bouton "Modifier" met à jour les données dans la base de données et actualise le tableau.

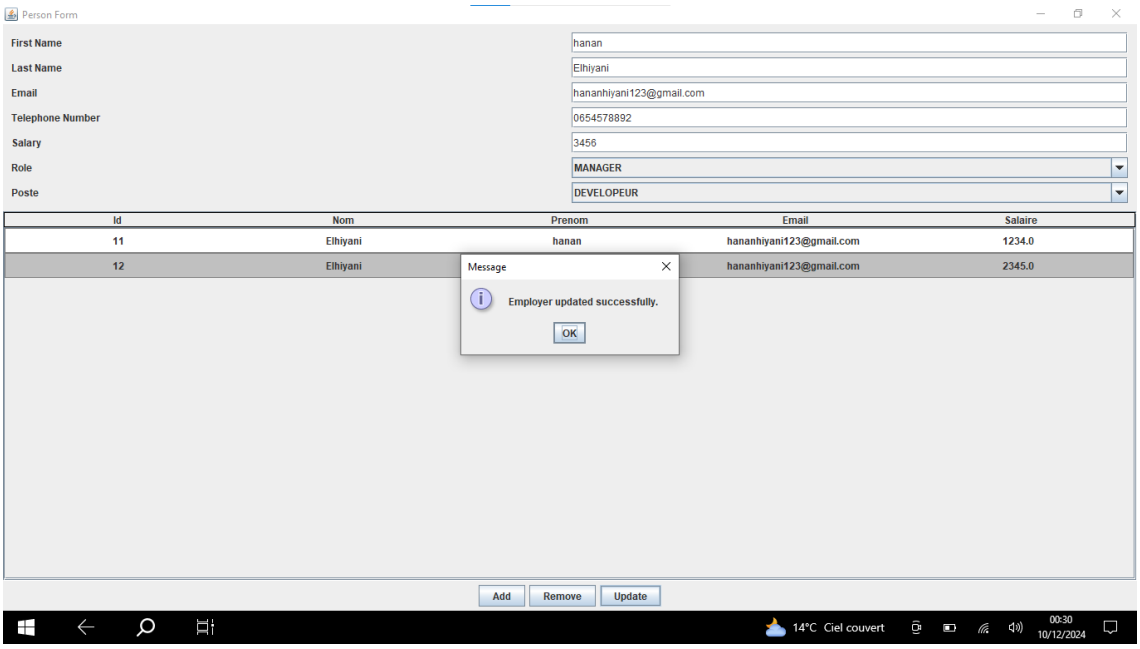

FIGURE 2.3 – Resultat de modification

3 Button suprimmer

Pour supprimer une entrée, l'utilisateur doit sélectionner la ligne correspondante dans le tableau de l'interface graphique. Ensuite, en cliquant sur le bouton "Supprimer", un message de confirmation peut s'afficher pour éviter les suppressions accidentelles. Si l'utilisateur confirme, l'entrée est supprimée de la base de données et le tableau est mis à jour automatiquement.

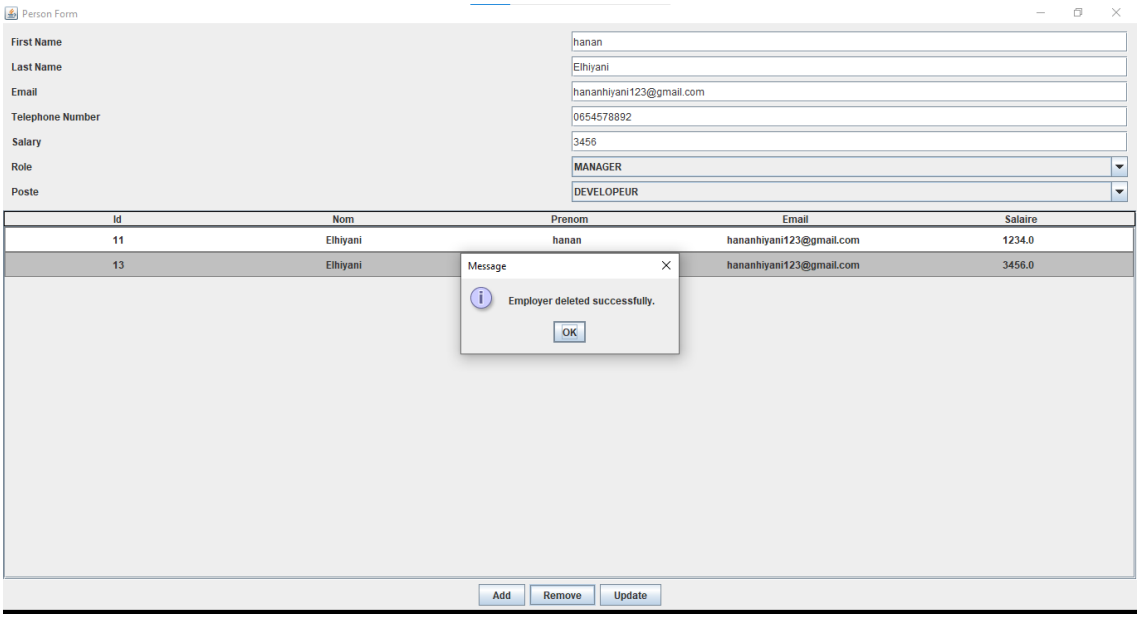

FIGURE 2.4 – Resultat de suppression