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

Rapport du TP java avance 1

Réalisé par : ELGOUDIMI Meryem

Encadré par : Mme. KACHBAL Ilham

Année Universitaire : 2024/2025

Table des matières

Table des figures

Inroduction

Ce travail pratique (TP) vise le développement d'une application Java de gestion des employés en adoptant l'architecture MVC (Modèle-Vue-Contrôleur), tout en mettant en œuvre des interfaces graphiques à l'aide de la bibliothèque Swing.

Ce projet s'inscrit dans une démarche pédagogique qui combine l'apprentissage des bases de la programmation orientée objet (POO) et des bonnes pratiques de conception logicielle. L'objectif est de concevoir une application intuitive et modulaire, permettant de gérer efficacement les données des employés à travers des fonctionnalités telles que l'ajout, la modification, la suppression et l'affichage des informations.

Grâce à une structure bien définie et à une séparation claire des responsabilités, ce TP offre une opportunité d'approfondir les notions de maintenabilité et d'évolutivité, tout en préparant les étudiants à des développements logiciels plus avancés.

Outils & environnement de travail

1 Environnement de travail

FIGURE 1 – intellij idea logo

,

- **IntelliJ IDEA** : est un IDE développé par JetBrains, conçu principalement pour Java, mais prenant en charge d'autres langages comme Kotlin, Scala et Python. Il se distingue par ses outils intelligents, sa complétion de code avancée, son analyse en temps réel et son intégration avec des outils comme Maven, Gradle et Git, offrant une productivité accrue aux développeurs.

2 Outils de travail

FIGURE 2 – MySQL Workbench logo

- **MySQL Workbench** : est un outil graphique développé pour simplifier la conception, l'administration et la gestion des bases de données MySQL. Il offre une interface intuitive qui permet de manipuler les bases de données visuellement, sans dépendre uniquement des commandes en ligne, rendant les tâches plus accessibles et efficaces.

FIGURE 3 – xampp logo

- **XAMPP** : est une solution open-source qui regroupe Apache, MySQL, PHP, et Perl pour faciliter le développement et le déploiement d'applications web. Il est conçu pour créer un environnement de serveur local rapide et simple, adapté aux développeurs souhaitant tester ou gérer des projets web sur leur machine avant le déploiement.

FIGURE 4 – java developpement kit logo

- **java developpement kit** : est un ensemble d'outils logiciels indispensables pour le développement d'applications Java. Il comprend des composants essentiels tels qu'un compilateur, une machine virtuelle Java (JVM), des bibliothèques standard, et des outils de débogage, permettant de coder, compiler, exécuter et tester des programmes Java efficacement.

3 Language de Programmation

FIGURE 5 – java logo

- **Java** : un langage de programmation orienté objet et une plateforme largement utilisée pour le développement d'applications logicielles. Il a été créé par Sun Microsystems (maintenant propriété d'Oracle) en 1995 et reste l'un des langages les plus populaires au monde, notamment pour les applications d'entreprise, le développement mobile (Android) et les applications web.

Réalisation

1 Création de la base de donnée

1.1 Script base de donnée

```sql
create database DBperson;
use DBperson;
CREATE TABLE employers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    email VARCHAR(50),
    phone INT(10),
    salary NUMERIC,
    role VARCHAR(50),
    poste VARCHAR(50)

);
CREATE TABLE Role (
  name varchar(30) NOT NULL
);
INSERT INTO Role (name) VALUES ('Admin'),('Employee');
INSERT INTO Poste (name) VALUES ('Ingenieur'),('Team_Leader'),('Pilote');

CREATE TABLE Poste (
  name varchar(30) NOT NULL
);
```

Listing 1.1 – Script SQL de la base de données

-

2 Architecture MVC (Model-View-Controller)

L'architecture MVC est un modèle de conception qui sépare les responsabilités au sein d'une application, facilitant ainsi la gestion et la maintenance du code. Elle repose sur trois composants principaux :

2.1 Implémentation du modèle (couche Model)

Employer

```
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) {

```

```
36
37          if ( isValidEmail(email) ) {
38
39              Employer employer = new Employer(
40                  id,
41                  firstName,
42                  lastName,
43                  email,
44                  phoneNumber,
45                  salary,
46                  role,
47                  poste
48              );
49
50              return dao.updateEmployer(employer);
51          }
52
53          return false;
54      }
55
56      private boolean isValidEmail(String email) {
57          return email.contains("@gmail.com") ? true  : false;
58      }
59
60
61      public boolean deleteEmployer(int id) {
62          return dao.deleteEmployer(id);
63      }
64
65      public List<Employer> getAllEmployers() {
66          return dao.getAllEmployers();
67      }
68
69  }
```

2.2 Gestion des données (DAO)

DBConnection

```
1
2
3
4                                   ;
5                                   ;
6
7
8
9                                   ng URL = "jdbc:mysql://localhost:3306/DBperson";
10                                  ing USERNAME = "root";
11                                  ng PASSWORD = "";
12
13                                   getConnection() throws SQLException
14                                  er.getConnection(URL, USERNAME, PASSWORD);
15
16
```

EmployerDAO

```
1
2
3
4
5
6
7
8
9
10                                  ements EmployerInterface
11
12                                  nection;
13
14
15
16                                  nnection.getConnection();
17                                  connectionException)
18                                  tion.printStackTrace();
19
20
21
22
23
24                                  r(Employer employer) {
25                                  t addStatement =
26                                  epareStatement( "INSERT INTO employers (first_name,
                                    ail, phone,
27                                  ES (?, ?, ?, ?, ?, ?, ?)")) {
28
29                                  String(1, employer.getFirstName());
30                                  String(2, employer.getLastName());
```

```
            addStatement.setInt(4, employer.getPhoneNumber());
            addStatement.setDouble(5, employer.getSalary());
            addStatement.setString(6, employer.getRole().name());
            addStatement.setString(7, employer.getPoste().name());

            return addStatement.executeUpdate() > 0;

        } catch (SQLException addException)
            { addException.printStackTrace();
            return false;
        }
    }


    @Override
    public boolean updateEmployer(Employer employer) {
        try (PreparedStatement updateStatement = connection.prepareStatement("
    UPDATE employers SET first_name = ?, last_name = ?, email = ?, phone = ?,
    salary = ?, role = ?, poste = ? WHERE id = ?")) {

            updateStatement.setString(1, employer.getFirstName());
            updateStatement.setString(2, employer.getLastName());
            updateStatement.setString(3, employer.getEmail());
            updateStatement.setInt(4, employer.getPhoneNumber());
            updateStatement.setDouble(5, employer.getSalary());
            updateStatement.setString(6, employer.getRole().name());
            updateStatement.setString(7, employer.getPoste().name());
            updateStatement.setInt(8, employer.getId());

            return updateStatement.executeUpdate() > 0;

        } catch (SQLException updateException) {
            updateException.printStackTrace();
            return false;
        }
    }

    @Override
    public boolean deleteEmployer(int id) {
        try (PreparedStatement deleteStatement = connection.prepareStatement("
    DELETE FROM employers WHERE id = ?")) {

            deleteStatement.setInt(1, id);
            return deleteStatement.executeUpdate() > 0;

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

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

```
82
83              while (getResult.next())
84                  { employers.add(new
85                  Employer(
86                          getResult.getInt("id"),
87                          getResult.getString("first_name"),
88                          getResult.getString("last_name"),
89                          getResult.getString("email"),
90                          getResult.getInt("phone"),
91                          getResult.getDouble("salary"),
92                          Role.valueOf(getResult.getString("role")),
93                          Poste.valueOf(getResult.getString("poste"))
94                  ));
95              }
96
97          } catch (SQLException getException)
98              { getException.printStackTrace();
99          }
100         return employers;
101     }
```

2.3 Contrôleur (couche Controller)

EmployerController

```
1  package controller;
2
3  import view.*;
4  import dao.*;
5  import model.*;
6  import enums.*;
7  import java.util.List;
8  import javax.swing.JOptionPane;
9
10 public class EmployerController {
11
12     private FormFrame frame;
13     private EmployerLogic employerLogic;
14
15     public EmployerController(FormFrame frame, EmployerLogic employerLogic)
           { this.frame = frame;
16         this.employerLogic = employerLogic;
17
18         frame.getBtnPanel().getAddBtn().addActionListener(addEvent -> addEmployer
19     ());
           frame.getBtnPanel().getUpdateBtn().addActionListener(updateEvent ->
20     updateEmployer());
           frame.getBtnPanel().getRemoveBtn().addActionListener(deleteEvent ->
21     deleteEmployer());
           loadEmployers();
22      }
23
24     private void addEmployer() {
25
```

```
26          try {
27
28              if (employerLogic.addEmployer(
29                      1,
30                      frame.getInPanel().getFirstNameField().getText(),
31                      frame.getInPanel().getLastNameField().getText(),
32                      frame.getInPanel().getEmailField().getText(),
33                      Integer.parseInt(frame.getInPanel().getTelephoneNumberField().
    getText()),
34                      Double.parseDouble(frame.getInPanel().getSalaryField().getText
    ()),
35                      Role.valueOf(frame.getInPanel().getSelectedRole().toString()),
36                      Poste.valueOf(frame.getInPanel().getSelectedPoste().toString()
    )
37                  ))
38              {
39                  JOptionPane.showMessageDialog(frame, "Employer added successfully
    .");
40                  loadEmployers();
41              } else {
42                  JOptionPane.showMessageDialog(frame, "Failed to add employer.");
43              }
44          } catch (Exception e) {
45              JOptionPane.showMessageDialog(frame, "Invalid input: " + e.getMessage
    ());
46          }
47      }
48
49      private void updateEmployer() {
50       try {
51           if (employerLogic.updateEmployer(
52                    frame.getListPanel().getSelectedRowId(),
53                    frame.getInPanel().getFirstNameField().getText(),
54                    frame.getInPanel().getLastNameField().getText(),
55                    frame.getInPanel().getEmailField().getText(),
56                    Integer.parseInt(frame.getInPanel().getTelephoneNumberField().
    getText()),
57                    Double.parseDouble(frame.getInPanel().getSalaryField().getText())
    ,
58                    Role.valueOf(frame.getInPanel().getSelectedRole().toString()),
59                    Poste.valueOf(frame.getInPanel().getSelectedPoste().toString())
60               ))
61           {
62               JOptionPane.showMessageDialog(frame, "Employer updated successfully
    .");
63               loadEmployers();
64           } else {
65               JOptionPane.showMessageDialog(frame, "Failed to update employer.");
66           }
67       } catch (Exception e) {
68           JOptionPane.showMessageDialog(frame, "Invalid input: " + e.getMessage());
69       }
70      }
71
72      private void deleteEmployer() {
```

```
     try {
         if (employerLogic.deleteEmployer(frame.getListPanel().getSelectedRowId())) {
             JOptionPane.showMessageDialog(frame, "Employer deleted successfully.");
             loadEmployers();
         } else {
             JOptionPane.showMessageDialog(frame, "Failed to delete employer.");
         }
     } catch (Exception e) {
          JOptionPane.showMessageDialog(frame, "Invalid input: " + e.getMessage());
     }
    }

    private void loadEmployers() {
         frame.getListPanel().updateEmployerList(employerLogic.getAllEmployers());
    }
}
```

2.4 Interface graphique (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 FormFrame extends JFrame

    { private InputPanel inPanel;
    private BtnPanel btnPanel;
    private ListPanel listPanel;

    public FormFrame()
        { setTitle("Person Form");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        inPanel = new InputPanel();
        btnPanel = new BtnPanel();
        listPanel = new ListPanel();

        add(inPanel, BorderLayout.NORTH);
        add(btnPanel, BorderLayout.SOUTH);

        JScrollPane scrollPane = new JScrollPane(listPanel);
        add(scrollPane, BorderLayout.CENTER);

        setVisible(true);
    }

    public InputPanel getInPanel()
        { return inPanel;
    }

    public BtnPanel getBtnPanel()
        { return btnPanel;
    }

    public ListPanel getListPanel()
        { return listPanel;
    }
}
```

InputPanel

```
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
```

```
    public JTextField getEmailField()
        { return emailField;
    }

    public JTextField getTelephoneNumberField()
        { return telephoneNumberField;
    }

    public JTextField getSalaryField()
        { return salaryField;
    }

    public Role getSelectedRole() {
        return (Role) roleField.getSelectedItem();
    }

    public Poste getSelectedPoste() {
        return (Poste) posteField.getSelectedItem();
    }
}
```

ListPanel

```
package view;

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() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    highlightRow(rowPanel, employer.getId());
                }
            });

            rowPanel.add(idLabel);
            rowPanel.add(lastNameLabel);
            rowPanel.add(firstNameLabel);
            rowPanel.add(emailLabel);
            rowPanel.add(salaryLabel);

            contentPanel.add(rowPanel);
            rowPanels.add(rowPanel);
        }

        contentPanel.revalidate();
        contentPanel.repaint();
    }

    private void highlightRow(JPanel selectedRow, int employerId) {
        for (JPanel row : rowPanels) {
            row.setBackground(Color.WHITE);
        }

        selectedRow.setBackground(Color.LIGHT_GRAY);
        selectedRowId = employerId;
    }

```

```
82      public int getSelectedRowId()
83          { return selectedRowId;
84      }
85  }
```

2.5 Main

```
1  import view.*;
2  import dao.*;
3  import model.*;
4  import controller.EmployerController;
5
6  public class Main {
7      public static void main(String[] args) {
8          new EmployerController( new FormFrame(), new EmployerLogic(new EmployerDAO
       ()));
9      }
10 }
```

Résultats

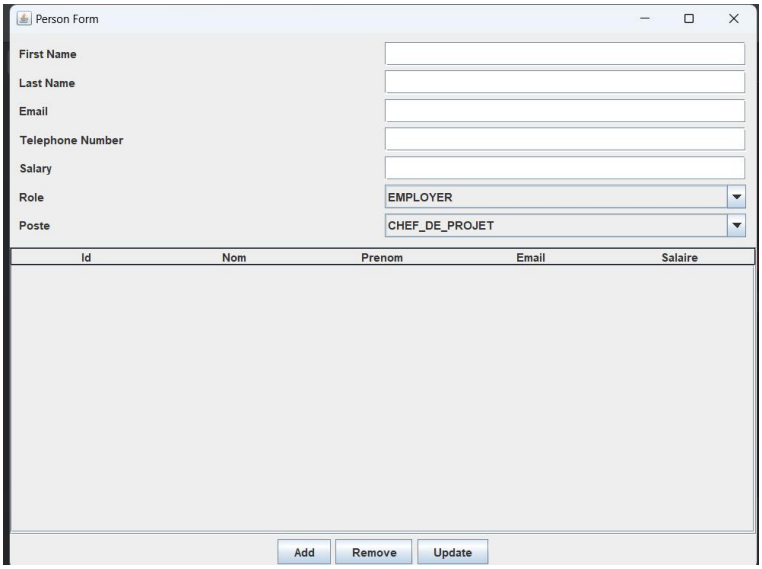

FIGURE 2.1 – Interface Utilisateur

1 Ajout

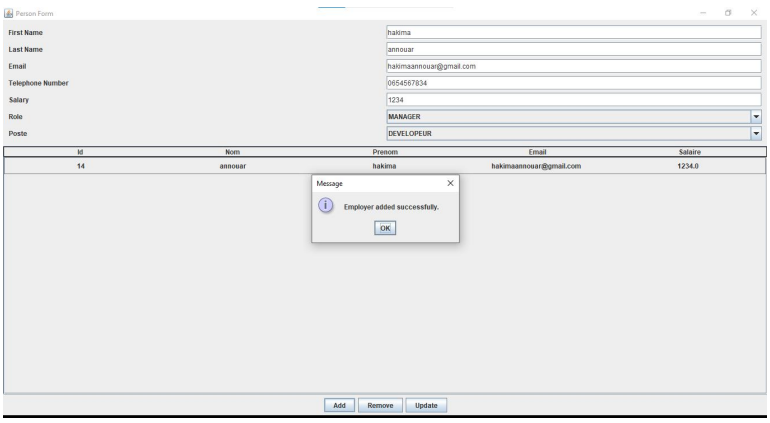

FIGURE 2.2 – Affichage de l'Ajout

2 Modification

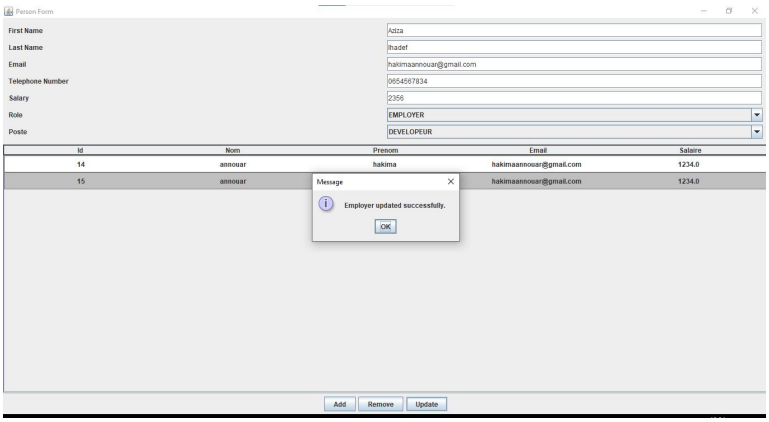

FIGURE 2.3 – Affichage de modification

3 Suppression

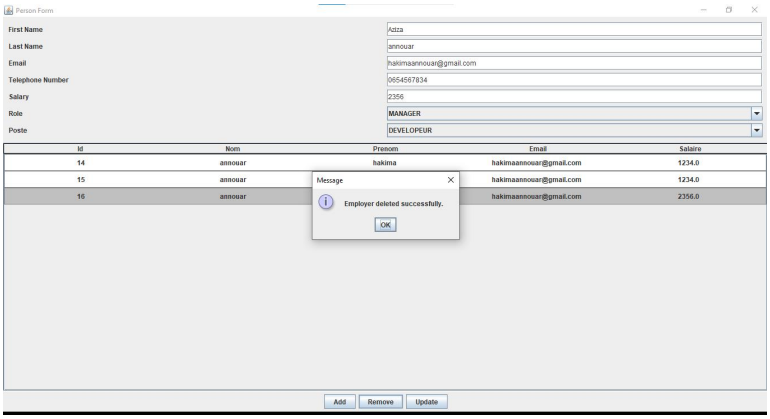

FIGURE 2.4 – Affichage de suppression