

# Projet DAO/MVC en Java : Application de Gestion des Employés

AMINE ZAGHLOUL AKRAM ELBAHAR

December 8, 2024

## Contents

|          |                                      |           |
|----------|--------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                  | <b>2</b>  |
| <b>2</b> | <b>Objectifs</b>                     | <b>2</b>  |
| <b>3</b> | <b>Architecture du Projet</b>        | <b>2</b>  |
| <b>4</b> | <b>Étapes de Création du Projet</b>  | <b>2</b>  |
| 4.1      | Création du Modèle (DAO) . . . . .   | 2         |
| 4.1.1    | Classe Employee . . . . .            | 2         |
| 4.1.2    | Classe DAO . . . . .                 | 5         |
| 4.2      | Création du Contrôleur . . . . .     | 6         |
| 4.3      | Création de la Vue . . . . .         | 9         |
| 4.4      | Création de la Classe Main . . . . . | 12        |
| <b>5</b> | <b>Conclusion</b>                    | <b>12</b> |

# 1 Introduction

Ce rapport présente une application développée en Java basée sur l'architecture **MVC** (Modèle-Vue-Contrôleur) avec un accès aux données via la couche **DAO** (Data Access Object). L'application permet la gestion des employés dans une entreprise en incluant différents rôles et postes.

## 2 Objectifs

- Implémenter une architecture MVC robuste.
- Créer une application avec des rôles et des postes spécifiques.
- Utiliser des classes DAO pour séparer la logique d'accès aux données.

## 3 Architecture du Projet

L'architecture MVC est composée de :

- **Modèle** : Gère les données (DAO).
- **Vue** : Affiche les informations à l'utilisateur.
- **Contrôleur** : Gère les actions de l'utilisateur.

## 4 Étapes de Création du Projet

### 4.1 Création du Modèle (DAO)

Le modèle représente les objets métiers et l'accès aux données.

#### 4.1.1 Classe Employee

```
1
2
3
4 public class Employee {
5     public enum Role {
6         EMPLOYEE,
7         ADMIN
8     }
9
10    public enum Poste {
11        INGENIEUR_ETUDE_ET_DEVELOPPEMENT,
12        TEAM_LEADER,
13        PILOTE
14    }
15    private int id;
16    private String nom;
```

```

17     private String prenom;
18     private String tel;
19     private String email;
20
21
22
23     private Double salaire ;
24     private Poste poste ;
25
26     public Role getRole() {
27         return role;
28     }
29
30     public void setRole(Role role) {
31         this.role = role;
32     }
33
34     private Role role ;
35
36     public Employee(int id, String nom, String prenom, String tel
37         , String email, double salaire , Role role , Poste poste) {
38         this.salaire = salaire ;
39         this.id = id;
40         this.nom = nom;
41         this.prenom = prenom;
42         this.tel = tel;
43         this.email = email;
44         this.role = role ;
45         this.poste = poste ;
46     }
47     public Double getSalaire() {
48         return salaire;
49     }
50     public Poste getPoste() {
51         return poste;
52     }
53
54     public void setPoste(Poste poste) {
55         this.poste = poste;
56     }
57     public void setSalaire(Double salaire) {
58         this.salaire = salaire;
59     }
60     public int getId() {
61         return id;
62     }
63
64     public void setId(int id) {
65         this.id = id;
66     }

```

```

67     public String getNom() {
68         return nom;
69     }
70
71     public void setNom(String nom) {
72         this.nom = nom;
73     }
74
75     public String getPrenom() {
76         return prenom;
77     }
78
79     public void setPrenom(String prenom) {
80         this.prenom = prenom;
81     }
82
83     public String getTel() {
84         return tel;
85     }
86
87     public void setTel(String tel) {
88         this.tel = tel;
89     }
90
91     public String getEmail() {
92         return email;
93     }
94
95     public void setEmail(String email) {
96         this.email = email;
97     }
98
99
100    @Override
101    public String toString() {
102        return String.format(
103            "%-5s|%-15s|%-15s|%-15s|%-25s|%-10s|
104            %-20s|%-10s|",
105            id, nom, prenom, tel, email, salaire, poste, role
106        );
107    }
108    package Model;
109
110    import DAO.EmployeeDAOImpl;
111    import java.util.List;
112
113    public class EmployeeModel {
114        private final EmployeeDAOImpl dao;
115
116        public EmployeeModel(EmployeeDAOImpl dao) {

```

```

117         this.dao = dao;
118     }
119
120     public void addEmployee(Employee emp) {
121         dao.add(emp);
122     }
123
124     public List<Employee> getAllEmployees() {
125         return dao.findAll();
126     }
127
128     public void deleteEmployee(int id) {
129         dao.delete(id);
130     }
131
132     public void updateEmployee(Employee emp, int id) {
133         dao.update(emp, id);
134     }
135
136     public static int parseEmployeeId(String
137         selectedEmployeeString) {
138         try {
139             return Integer.parseInt(selectedEmployeeString.split(
140                 "\\|")[1].trim());
141         } catch (Exception e) {
142             throw new IllegalArgumentException("Invalid employee
143                 format.");
144         }
145     }
146 }

```

#### 4.1.2 Classe DAO

```

1 import java.util.ArrayList;
2 import java.util.List;
3
4 public class EmployeeDAO {
5     private List<Employee> employees = new ArrayList<>();
6
7     public void addEmployee(Employee e) {
8         employees.add(e);
9     }
10
11     public List<Employee> getAllEmployees() {
12         return employees;
13     }
14 }

```

## 4.2 Création du Contrôleur

Le contrôleur gère la logique métier.

```
1 package Controller;
2
3 import Model.Employee;
4 import Model.EmployeeModel;
5 import Vue.Vue;
6
7 import javax.swing.*;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 public class EmployeeController {
12     private final EmployeeModel model;
13     private final Vue view;
14
15     public EmployeeController(EmployeeModel model, Vue view) {
16         this.model = model;
17         this.view = view;
18
19         initializeListeners();
20     }
21
22     private void initializeListeners() {
23         view.getAjouter().addActionListener(e -> {
24             try {
25                 Employee emp = new Employee(
26                     0,
27                     view.getNom().getText(),
28                     view.getPrenom().getText(),
29                     view.getTel().getText(),
30                     view.getEmail().getText(),
31                     Double.parseDouble(view.getSal().getText()),
32                     Employee.Role.valueOf((String) view.
33                         getRoleComboBox().getSelectedItem()),
34                     Employee.Poste.valueOf((String) view.
35                         getPostesComboBox().getSelectedItem())
36                 );
37                 model.addEmployee(emp);
38                 JOptionPane.showMessageDialog(view, "Employee_
39                     added_successfully!", "Success", JOptionPane.
40                     INFORMATION_MESSAGE);
41                 view.getAfficher().doClick();
42             } catch (NumberFormatException ex) {
43                 JOptionPane.showMessageDialog(view, "Invalid_
44                     salary_value!", "Error", JOptionPane.
45                     ERROR_MESSAGE);
46             } catch (IllegalArgumentException ex) {
```

```

41         JOptionPane.showMessageDialog(view, ex.getMessage
42             (), "Error", JOptionPane.ERROR_MESSAGE);
43     }
44 });
45
46 view.getAfficher().addActionListener(e -> {
47     List<Employee> allEmployees = model.getAllEmployees()
48     ;
49     List<String> employeeStrings = new ArrayList<>();
50     employeeStrings.add(String.format(
51         "%-5s%-15s%-15s%-15s%-25s%-10s%-20s%-10s",
52         "ID", "Nom", "Prenom", "Tel", "Email", "
53         Salaire", "Poste", "Role"
54     ));
55     for (Employee emp : allEmployees) {
56         employeeStrings.add(emp.toString());
57     }
58     String[] employeeArray = employeeStrings.toArray(new
59         String[0]);
60     JList<String> updatedList = new JList<>(employeeArray
61         );
62     view.setEmployeeList(updatedList);
63     JPanel p3 = view.getP3();
64     p3.removeAll();
65     p3.add(new JScrollPane(updatedList));
66     p3.revalidate();
67     p3.repaint();
68 });
69
70 view.getSupprimer().addActionListener(e -> {
71     String selectedEmployeeString = view.getEmployeeList
72     ().getSelectedValue();
73     if (selectedEmployeeString == null) {
74         JOptionPane.showMessageDialog(view, "No employee
75         selected!", "Error", JOptionPane.ERROR_MESSAGE
76         );
77         return;
78     }
79
80     try {
81         int id = EmployeeModel.parseEmployeeId(
82             selectedEmployeeString);
83         int confirm = JOptionPane.showConfirmDialog(view,
84             "Are you sure you want to delete this
85             employee?", "Confirm", JOptionPane.
86             YES_NO_OPTION);
87         if (confirm == JOptionPane.YES_OPTION) {
88             model.deleteEmployee(id);
89             JOptionPane.showMessageDialog(view, "Employee
90             deleted successfully!", "Success",

```

```

78         JOptionPane.INFORMATION_MESSAGE);
79         view.getAfficher().doClick();
80     }
81     } catch (IllegalArgumentException ex) {
82         JOptionPane.showMessageDialog(view, ex.getMessage
83             (), "Error", JOptionPane.ERROR_MESSAGE);
84     }
85 }));
86
87 view.getModifier().addActionListener(e -> {
88     String selectedEmployeeString = view.getEmployeeList
89         ().getSelectedValue();
90     if (selectedEmployeeString == null) {
91         JOptionPane.showMessageDialog(view, "No employee
92             selected!", "Error", JOptionPane.ERROR_MESSAGE
93             );
94         return;
95     }
96
97     try {
98         int id = EmployeeModel.parseEmployeeId(
99             selectedEmployeeString);
100         Employee emp = new Employee(
101             id,
102             view.getNom().getText(),
103             view.getPrenom().getText(),
104             view.getTel().getText(),
105             view.getEmail().getText(),
106             Double.parseDouble(view.getSal().getText
107                 ()),
108             Employee.Role.valueOf((String) view.
109                 getRoleComboBox().getSelectedItem()),
110             Employee.Poste.valueOf((String) view.
111                 getPostesComboBox().getSelectedItem())
112             );
113         model.updateEmployee(emp, id);
114         JOptionPane.showMessageDialog(view, "Employee
115             updated successfully!", "Success", JOptionPane
116             .INFORMATION_MESSAGE);
117         view.getAfficher().doClick();
118     } catch (NumberFormatException ex) {
119         JOptionPane.showMessageDialog(view, "Invalid
120             salary value!", "Error", JOptionPane.
121             ERROR_MESSAGE);
122     } catch (IllegalArgumentException ex) {
123         JOptionPane.showMessageDialog(view, ex.getMessage
124             (), "Error", JOptionPane.ERROR_MESSAGE);
125     }
126 }));
127 }
128 }

```



## 4.3 Création de la Vue

La vue affiche les résultats.

```
1 package Vue;
2 import DAO.*;
3 import Model.Employee;
4
5 import javax.print.DocFlavor;
6 import javax.swing.*;
7 import javax.swing.border.Border;
8 import java.awt.*;
9 import java.util.ArrayList;
10 import java.util.List;
11
12 public class Vue extends JFrame {
13
14     private JPanel p1;
15     private JPanel p2;
16     private JPanel p3;
17     public JPanel getP3(){
18         return p3 ;
19     }
20
21     private JPanel p4;
22     private JComboBox<String> postesComboBox ;
23     private JComboBox<String> roleComboBox ;
24     private JButton ajouter ;
25     private JButton modifier;
26     private JButton supprimer ;
27     private JButton afficher ;
28
29
30     private JTextField tel;
31     private JTextField sal ;
32     private JTextField nom ;
33     private JTextField prenom ;
34     private JTextField email ;
35     private JList<String> employeeList ;
36     public JList<String> getEmployeeList(){
37         return employeeList ;
38     };
39     public void setEmployeeList(JList<String> employeeList){
40         this.employeeList =employeeList ;
41     } ;
42     public Vue() {
43         setTitle("App");
44         setSize(1920, 1080);
45         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46         EmployeeDAOImpl eImp = new EmployeeDAOImpl();
47         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
48         p1 = new JPanel();
```

```

49     p2 = new JPanel();
50     p3 = new JPanel();
51     p4 = new JPanel();
52     p1.setLayout(new BorderLayout());
53     p2.setLayout(new GridLayout(7,2,10,10));
54
55     p3.setLayout(new GridLayout());
56     p4.setLayout(new GridLayout());
57     add(p1);
58     p1.add(p2 , BorderLayout.NORTH);
59     p1.add(p3 , BorderLayout.CENTER);
60     p1.add(p4 , BorderLayout.SOUTH);
61
62
63     p2.add(new JLabel("Nom:"));
64     nom = new JTextField();p2.add(nom);
65     p2.add(new JLabel("Prenom:"));
66     prenom = new JTextField();p2.add(prenom);
67     p2.add(new JLabel("Email:"));
68     email = new JTextField();p2.add(email);
69     p2.add(new JLabel("Telephone:"));
70     tel = new JTextField();p2.add(tel);
71     p2.add(new JLabel("Salaire:"));
72     sal = new JTextField();p2.add(sal);
73
74     p2.add(new JLabel("Role:"));
75     List<Employee.Role> roles = eImp.findAllRoles() ;
76     String[] roleStrings = roles.stream()
77         .map(Enum::name)
78         .toArray(String[]::new);
79     roleComboBox = new JComboBox<String>(roleStrings);
80     p2.add(roleComboBox);
81
82     p2.add(new JLabel("Poste:"));
83     List<Employee.Poste> postes = eImp.findAllPostes() ;
84     String[] postesStrings = postes.stream()
85         .map(Enum::name)
86         .toArray(String[]::new);
87     postesComboBox = new JComboBox<String>(postesStrings);
88     p2.add(postesComboBox);
89
90
91
92     //P3 Container
93     List<Employee> all_e = eImp.findAll();
94     List<String> allString = new ArrayList<String>();
95     allString.add(String.format(
96         "%-5s|%-15s|%-15s|%-15s|%-25s|%-10s|%-20s|%-10s|",
97         "ID", "Nom", "Prenom", "Tel", "Email", "Salaire",
98         "Poste", "Role"

```

```

98     ));
99     for (Employee e : all_e){
100         allString.add(e.toString());
101     }
102     String[] allStringArray = allString.toArray(new String
        [0]);
103
104     employeeList = new JList<String>(allStringArray);
105     p3.add(employeeList);
106
107
108     //p4 Container
109
110     p4.setLayout(new FlowLayout());
111     this.ajouter = new JButton("Ajouter");
112     this.modifier = new JButton("Modifier");
113     this.supprimer = new JButton("Supprimer");
114     this.afficher = new JButton("Afficher");
115     p4.add(this.ajouter);p4.add(this.modifier);p4.add(this.
        supprimer);p4.add(this.afficher);
116     setVisible(true);
117 }
118
119 public JComboBox<String> getPostesComboBox() {
120     return postesComboBox;
121 }
122
123 public JComboBox<String> getRoleComboBox() {
124     return roleComboBox;
125 }
126
127 public JButton getAjouter() {
128     return ajouter;
129 }
130
131 public JButton getModifier() {
132     return modifier;
133 }
134
135 public JButton getSupprimer() {
136     return supprimer;
137 }
138
139 public JButton getAfficher() {
140     return afficher;
141 }
142
143 public JTextField getTel() {
144     return tel;
145 }
146

```

```

147     public JTextField getSal() {
148         return sal;
149     }
150
151     public JTextField getNom() {
152         return nom;
153     }
154
155     public JTextField getPrenom() {
156         return prenom;
157     }
158
159     public JTextField getEmail() {
160         return email;
161     }
162
163     public JPanel getP1() {
164         return p1;
165     }
166
167 }

```

## 4.4 Création de la Classe Main

```

1 import DAO.EmployeeDAOImpl;
2 import Model.EmployeeModel;
3 import Vue.Vue;
4 import Controller.EmployeeController;
5
6 public class Main {
7     public static void main(String[] args) {
8         EmployeeDAOImpl dao = new EmployeeDAOImpl();
9         EmployeeModel model = new EmployeeModel(dao);
10        Vue view = new Vue();
11
12        // Pass the model and view to the controller
13        new EmployeeController(model, view);
14    }
15 }

```

## 5 Conclusion

Ce projet met en œuvre l'architecture MVC et l'utilisation des DAO pour une application simple de gestion des employés. Chaque couche est bien séparée, permettant une évolutivité et une maintenance optimales.