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

Rapport du TP N°2 (MVC, DAO et Java Swing)

Gestion des employés et des Congés

Réalisé par : CHABLAOUI Salma

Encadré par : Mme. KACHBAL Ilham

Année Universitaire : 2024/2025

Table des matières

Table des figures

Introduction

Ce travail pratique (TP) s'intéresse à la création d'une application Java pour la gestion des congés des employés. Il est structuré autour de l'architecture MVC (Model-View-Controller), un modèle de conception qui sépare clairement les données, la logique métier et l'interface utilisateur. Ce projet vise à renforcer la maîtrise des concepts de la programmation orientée objet (POO) et à développer des interfaces graphiques avec la bibliothèque Swing. L'approche suivie dans ce TP permet également de perfectionner la gestion du code et de garantir une organisation optimale grâce à une séparation des responsabilités.

L'application développée dans ce TP a pour but de simplifier la gestion des congés. Elle permet de gérer efficacement les informations relatives aux congés des employés, avec des fonctionnalités permettant leur ajout, modification, suppression et consultation. L'interface utilisateur, conçue pour être intuitive et conviviale, assure une interaction fluide et efficace avec l'application. L'utilisation de l'architecture MVC facilite la gestion du code, permettant une maintenance aisée et l'ajout futur de nouvelles fonctionnalités.

Les principales fonctionnalités de l'application incluent :

- L'ajout de congés avec toutes les informations nécessaires.
- La modification des congés existants.
- La suppression des congés du système.
- L'affichage de la liste des congés enregistrés dans la base de données.

Ce projet a pour objectif de démontrer l'efficacité de la programmation orientée objet et de l'architecture MVC dans la création d'applications logicielles robustes et évolutives. Il constitue une étape clé dans la formation permettant de maîtriser les concepts fondamentaux avant d'aborder des projets plus complexes à l'avenir.

Environnement et outils utilisés

1 Environnement de travail

Figure 1 – Eclipse logo

,

- **Eclipse** : Eclipse est un environnement de développement intégré (IDE) open-source, principalement utilisé pour le développement Java, mais aussi extensible pour d'autres langages via des plugins. Il offre des fonctionnalités comme la complétion automatique, le débogage et la gestion de projets, facilitant ainsi la création et le test d'applications logicielles.

2 Outils de travail

Figure 2 – MySQL Workbench logo

- **MySQL Workbench** : un outil de travail graphique conçu pour faciliter la conception, l'administration, et la gestion des bases de données MySQL. Il fournit une interface utilisateur intuitive permettant de travailler avec des bases de données sans avoir à utiliser uniquement des commandes en ligne.

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

FIGURE 4 – java developpement kit logo

- **java developpement kit** : st un ensemble d'outils logiciels nécessaires pour développer des applications Java. Il inclut les composants essentiels pour coder, compiler, exécuter et déboguer des programmes Java.

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

```
1
2  -- Cr ation de la table des holiday
3  CREATE TABLE holiday (
4    id INT(11) NOT NULL AUTO_INCREMENT,-- correction de 'auto incremente' en '
       AUTO_INCREMENT'
5    id_employe INT(11) NOT NULL,
6    startdate DATE DEFAULT NULL,
7    enddate DATE DEFAULT NULL,
8    type ENUM('ANNUAL', 'SICK', 'MATERNITY', 'OTHER') NOT NULL,
9    PRIMARY KEY (id),-- Ajout de la c l  primaire sur l'ID
10   FOREIGN KEY (id_employe) REFERENCES Employes(id) ON DELETE CASCADE -- Ajout de la
       contrainte de c l    trangre
11 );
12
13
14 -- Cr ation de la table de Login
15 CREATE TABLE login (
16   id INT(11) NOT NULL AUTO_INCREMENT,
17   username VARCHAR(50) NOT NULL UNIQUE,
18   password VARCHAR(255) NOT NULL,
19   PRIMARY KEY (id)
20 );
21
22
23 -- Insertion des donn es de l'admin
24 INSERT INTO login (username, password) VALUES ('salma', 'salma2006');
```

Listing 1.1 – Script SQL de la base de données

- Ce script est ecrit sur MySQL Workbench pour creation la base de donnée pour etre lier à au code via le driver JDBC pour garantir la gestion .

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 Model

Le modèle représente les données et la logique métier de l'application. Il gère l'accès aux données, effectue les calculs nécessaires et fournit les informations à la vue.

Holiday

```java
package Model;

import java.sql.Date;

public class Holiday{
    private int id_holiday;
    private int id_employe;
    private Date startDate;
    private Date endDate;
    private Type_holiday type;

public Holiday(int id_holiday, int id_employe,Date startDate, Date endDate
    , Type_holiday type){
    this.id_holiday = id_holiday;
    this.id_employe = id_employe;
    this.startDate = startDate;
    this.endDate = endDate;
    this.type = type;
}


public int getId_holiday() {
    return id_holiday;
}

public Date getStartDate() {
    return startDate;
}

public Date getEndDate() {
    return endDate;
}

public Type_holiday getType() {
    return type;
}
```

```
36
37 public int getId_employe() {
38     return id_employe;
39 }
40
41 }
```

TypeHoliday

```
1 package Model;
2
3 public enum TypeHoliday {
4   ANNUAL, SICK, MATERNITY, OTHER
5 }
```

Login

```
1 package Model;
2
3 public class Login {
4
5     private String username;
6     private String password;
7
8     public Login(String username, String password) {
9         this.username = username;
10        this.password = password;
11    }
12
13    public String getUsername() {
14        return username;
15    }
16
17    public void setUsername(String username) {
18        this.username = username;
19    }
20
21    public String getPassword() {
22        return password;
23    }
24
25    public void setPassword(String password) {
26        this.password = password;
27    }
28
29    public boolean verifyCredentials(String storedPassword) {
30        return this.password.equals(storedPassword);
```

```
31      }
32
33      @Override
34      public String toString() {
35          return "Login [username=" + username + ", password=" + password + "
    ]";
36      }
37  }
```

LoginModel

```
1  package Model;
2
3  import DAO.LoginDAOimpl;
4
5  public class LoginModel {
6
7      private LoginDAOimpl loginDAO;
8
9      public LoginModel(LoginDAOimpl loginDAO) {
10         this.loginDAO = loginDAO;
11     }
12
13     // Authentication method
14     public boolean authenticate(String username, String password) {
15         return loginDAO.authenticate(username, password); // Delegate to
    DAO
16     }
17 }
```

HolidayModel

```
1  package Model;
2
3
4  import Controller.EmployeController;
5  import java.util.List;
6
7  import DAO.HolidayDAOimpl;
8  import java.sql.Date;
9  public class HolidayModel {
10     private HolidayDAOimpl dao;
11
12     public HolidayModel(HolidayDAOimpl dao) {
13         this.dao = dao;
14     }
15
```

```
16      public boolean addHoliday(int id, int id_employe, Date startdate, Date
    enddate, Type_holiday type , Employe targetEmploye) {
17
18          if(startdate.after(enddate)) return false;
19          if(startdate.equals(enddate)) return false;
20          if(startdate.before(new Date(System.currentTimeMillis()))) return
    false;
21          if(enddate.before(new Date(System.currentTimeMillis()))) return
    false;
22
23          long daysBetween = (enddate.toLocalDate().toEpochDay() - startdate.
    toLocalDate().toEpochDay());
24          if(daysBetween > targetEmploye.getSolde()) return false;
25          EmployeController.updateSolde(targetEmploye.getId(), targetEmploye.
    getSolde() - (int) daysBetween);
26          Holiday e = new Holiday(id, id_employe, startdate, enddate, type);
27          dao.add(e);
28          return true;
29      }
30
31
32      public List<Holiday> displayHoliday() {
33          List<Holiday> Holidays = dao.display();
34          return Holidays;
35      }
36
37      public boolean deleteHoliday(int id) {
38          dao.delete(id);
39          return true;
40      }
41
42      public boolean updateHoliday(int id, int id_employe, Date startdate,
    Date enddate, Type_holiday type , Employe targetEmploye , int
    olddaysbetween ) {
43
44          long daysBetween = (enddate.toLocalDate().toEpochDay() - startdate.
    toLocalDate().toEpochDay());
45
46          if(startdate.after(enddate)) return false;
47          if(startdate.equals(enddate)) return false;
48          if(startdate.before(new Date(System.currentTimeMillis()))) return
    false;
49          if(enddate.before(new Date(System.currentTimeMillis()))) return
    false;
50
51          if(daysBetween > (targetEmploye.getSolde()+olddaysbetween)) return
    false;
52          EmployeController.updateSolde(targetEmploye.getId(), (targetEmploye
    .getSolde()+olddaysbetween) - (int) daysBetween);
```

```
53
54          Holiday e = new Holiday(id, id_employe, startdate, enddate, type);
55          dao.update(e);
56          return true;
57      }
58
59  }
```

2.2 DAO

Le DAO est une couche qui permet de gérer l'interaction avec une base de données, en effectuant des opérations telles que la création, la lecture, la mise à jour et la suppression (CRUD) des données.

- **DBConnexion**

```
1  package DAO;
2
3  import java.sql.*;
4
5  class DBConnexion {
6      public static final String url = "jdbc:mysql://localhost:3306/
       gestion";
7      public static final String user = "root";
8      public static final String password = "";
9      public static Connection conn = null;
10
11     public static Connection getConnexion() throws
       ClassNotFoundException {
12         if (conn != null) {
13             return conn;
14         }
15         try {
16
17             Class.forName("com.mysql.cj.jdbc.Driver");
18             conn = DriverManager.getConnection(url, user, password);
19             System.out.println("correct");
20         } catch (SQLException e) {
21             throw new RuntimeException("Error de connexion", e);
22         }
23
24         return conn;
25     }
26 }
```

GenericDAOI

```
1  package DAO;
2
```

```
import java.util.List;
public interface GenericDAOI <T> {
    public void add(T e);
    public void delete(int id);
    public void update(T e);
    public List<T> display();
}
```

HolidayDAOimpl

```
package DAO;
import Model.Holiday;
import Model.Type_holiday;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class HolidayDAOimpl implements GenericDAOI<Holiday> {

    @Override
    public void add(Holiday e) {
        String checkSoldeSql = "SELECT solde FROM employe WHERE id = ?";
        String insertHolidaySql = "INSERT INTO holiday (id_employe,
    startdate, enddate, type) VALUES (?, ?, ?, ?)";
        String updateSoldeSql = "UPDATE employe SET solde = solde - ?
    WHERE id = ?";

        try (PreparedStatement checkStmt = DBConnexion.getConnexion().
    prepareStatement(checkSoldeSql)) {
            // R cup rer le solde de cong  de l'employ
            checkStmt.setInt(1, e.getId_employe());
            ResultSet rs = checkStmt.executeQuery();

            if (rs.next()) {
                int solde = rs.getInt("solde");

                // Calculer le nombre de jours demand s
                long daysBetween = java.time.temporal.ChronoUnit.DAYS.
    between(
                    e.getStartDate().toLocalDate(),
                    e.getEndDate().toLocalDate()
                );

                if (daysBetween > solde) {
                    System.err.println("Le solde de cong  est
```

```
insuffisant.");
35                      return;
36                  }
37
38                  // Ins rer la demande de cong
39                  try (PreparedStatement insertStmt = DBConnexion.
getConnexion().prepareStatement(insertHolidaySql)) {
40                      insertStmt.setInt(1, e.getId_employe());
41                      insertStmt.setDate(2, e.getStartDate());
42                      insertStmt.setDate(3, e.getEndDate());
43                      insertStmt.setString(4, e.getType().name());
44
45                      int rowsInserted = insertStmt.executeUpdate();
46                      if (rowsInserted > 0) {
47                          System.out.println("Cong  ajout  avec succ s.
");
48                      }
49
50                      // Mettre    jour le solde de cong
51                      try (PreparedStatement updateStmt = DBConnexion.
getConnexion().prepareStatement(updateSoldeSql)) {
52                          updateStmt.setInt(1, (int) daysBetween);
53                          updateStmt.setInt(2, e.getId_employe());
54                          updateStmt.executeUpdate();
55                      }
56                  }
57              } else {
58                  System.err.println("Employ  introuvable.");
59              }
60          } catch (SQLException | ClassNotFoundException exception) {
61              System.err.println("Erreur lors de l'ajout du cong  : " +
exception.getMessage());
62              exception.printStackTrace();
63          }
64      }
65
66      @Override
67      public void delete(int id) {
68          String sql = "DELETE FROM holiday WHERE id = ?";
69          try (PreparedStatement stmt = DBConnexion.getConnexion().
prepareStatement(sql)) {
70              stmt.setInt(1, id);
71              int rowsDeleted = stmt.executeUpdate();
72              if (rowsDeleted > 0) {
73                  System.out.println("Cong  supprim  avec succ s.");
74              }
75          } catch (SQLException | ClassNotFoundException exception) {
76              System.err.println(" chec  de la suppression du cong  : " +
 exception.getMessage());
```

```
77          }
78      }
79
80      @Override
81      public void update(Holiday e) {
82          String sql = "UPDATE holiday SET id_employe = ?, startdate = ?,
    enddate = ?, type = ? WHERE id = ?";
83          try (PreparedStatement stmt = DBConnexion.getConnexion().
    prepareStatement(sql)) {
84              stmt.setInt(1, e.getId_employe());
85              stmt.setDate(2, e.getStartDate());
86              stmt.setDate(3, e.getEndDate());
87              stmt.setString(4, e.getType().name());
88              stmt.setInt(5, e.getId_holiday());
89
90              int rowsUpdated = stmt.executeUpdate();
91              if (rowsUpdated > 0) {
92                  System.out.println("Cong  mis    jour avec succ s.");
93              }
94          } catch (SQLException | ClassNotFoundException exception) {
95              System.err.println(" chec  de la mise    jour du cong  : "
    + exception.getMessage());
96          }
97      }
98
99      @Override
100     public List<Holiday> display() {
101         String sql = "SELECT * FROM holiday";
102         List<Holiday> holidays = new ArrayList<>();
103         try (PreparedStatement stmt = DBConnexion.getConnexion().
    prepareStatement(sql)) {
104             ResultSet re = stmt.executeQuery();
105             while (re.next()) {
106                 int id = re.getInt("id");
107                 int id_employe = re.getInt("id_employe");
108                 Date startdate = re.getDate("startdate");
109                 Date enddate = re.getDate("enddate");
110                 String type = re.getString("type");
111                 Holiday holiday = new Holiday(id, id_employe, startdate,
     enddate, Type_holiday.valueOf(type));
112                 holidays.add(holiday);
113             }
114
115             if (holidays.isEmpty()) {
116                 System.out.println("Aucun cong  trouv .");
117             }/* else {
118                 System.out.println("Liste des cong s :");
119                 for (Holiday holiday : holidays) {
120                     //System.out.println(holiday);
```

```
                }
            }*/

        } catch (SQLException | ClassNotFoundException ex) {
            System.err.println(" chec  de la r cup ration des cong s : " + ex.getMessage());
        }
        return holidays;
    }
}
```

LoginDAOimpl

```

package DAO;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class LoginDAOimpl {

    // Method to check if the username and password are valid
    public boolean authenticate(String username, String password) {
        String sql = "SELECT password FROM login WHERE username = ?";
        try (PreparedStatement stmt = DBConnexion.getConnexion().prepareStatement(sql)) {
            stmt.setString(1, username);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    String storedPassword = rs.getString("password");
                    return storedPassword.equals(password); // Return true if passwords match
                }
            }
        } catch (SQLException | ClassNotFoundException exception) {
            System.err.println("Error during authentication: " + exception.getMessage());
            exception.printStackTrace();
        }

        return false; // Return false if authentication fails
    }
}
```

2.3 View

EmployeHolidayView

```java
package view;

import Model.Employe;
import Model.Employemodel;
import Model.Post;
import Model.Role;
import Model.Type_holiday;


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

import DAO.EmployeDAOimpl;

import java.util.List;

public class EmployeHolidayView extends JFrame {

    // le tableau de employe et conge
    private JTabbedPane tabbedPane = new JTabbedPane();

    // les tabs
    private JPanel employeTab = new JPanel();
    private JPanel holidayTab = new JPanel();

    // les panels
    private JPanel Employepan = new JPanel();
    private JPanel Holidaypan = new JPanel();
    private JPanel Display_Table_employe = new JPanel();
    private JPanel Display_Table_holiday = new JPanel();
    private final JPanel Forme_employe = new JPanel();
    private final JPanel Forme_holiday = new JPanel();
    private JPanel panButton_employe = new JPanel();
    private JPanel panButton_holiday = new JPanel();

    // les labels du l'employe
    private JLabel label_nom = new JLabel("Nom");
    private JLabel label_prenom = new JLabel("Prenom");
    private JLabel label_email = new JLabel("Email");
    private JLabel label_tele = new JLabel("Telephone");
    private JLabel label_salaire = new JLabel("Salaire");
    private JLabel label_role = new JLabel("Role");
    private JLabel label_poste = new JLabel("Poste");

```

```
46      // les labels du conge
47      private JLabel label_employe = new JLabel("Nom de l'employe");
48      private JLabel label_startDate = new JLabel("Date de debut (YYYY-MM-
        DD)");
49      private JLabel label_endDate = new JLabel("Date de fin (YYYY-MM-DD)"
        );
50      private JLabel label_type = new JLabel("Type");
51      private JComboBox<Type_holiday> TypeComboBox = new JComboBox<>(
        Type_holiday.values());
52
53      // les textfield du l'employe
54      private JTextField text_nom = new JTextField();
55      private JTextField text_prenom = new JTextField();
56      private JTextField text_email = new JTextField();
57      private JTextField text_tele = new JTextField();
58      private JTextField text_salaire = new JTextField();
59
60      private JComboBox<Role> roleComboBox = new JComboBox<>(Role.values()
        );
61      private JComboBox<Post> posteComboBox = new JComboBox<>(Post.values
        ());
62
63      // les textfield du conges
64      private JComboBox<String> text_employe = new JComboBox<>();
65      private JTextField text_startDate = new JTextField("");
66      private JTextField text_endDate = new JTextField("");
67
68      // les boutons du l'employe
69      private JButton addButton_employe = new JButton("Ajouter");
70      private JButton updateButton_employe = new JButton("Modifier");
71      private JButton deleteButton_employe = new JButton("Supprimer");
72      private JButton displayButton_employe = new JButton("Afficher");
73
74      // les boutons du conges
75      private JButton addButton_holiday = new JButton("Ajouter");
76      private JButton updateButton_holiday = new JButton("Modifier");
77      private JButton deleteButton_holiday = new JButton("Supprimer");
78      private JButton displayButton_holiday = new JButton("Afficher");
79
80
81      // le tableau de l'employe
82      JPanel pan0 = new JPanel(new BorderLayout());
83      public static String[] columnNames_employe = {"ID", "Nom", "Prenom",
        "Email", "Telephone", "Salaire", "Role", "Poste","solde"};
84      public static DefaultTableModel tableModel = new DefaultTableModel(
        columnNames_employe, 0);
85      public static JTable Tableau = new JTable(tableModel);
86
87      // le tableau du conges
```

```
88      JPanel pan1 = new JPanel(new BorderLayout());
89      public static String[] columnNames_holiday = {"ID", "nom_employe","
        date_debut","date_fin","type"};
90      public static DefaultTableModel tableModel1 = new DefaultTableModel(
        columnNames_holiday, 0);
91      public static JTable Tableau1 = new JTable(tableModel1);
92
93      public Employe_HolidayView() {
94
95          setTitle("Gestion des employes et des conges");
96          setSize(1000, 600);
97          setDefaultCloseOperation(EXIT_ON_CLOSE);
98          setLocationRelativeTo(null);
99
100         add(tabbedPane);
101
102     // Employe Tab
103         employeTab.setLayout(new BorderLayout());
104         employeTab.add(Employepan, BorderLayout.CENTER);
105
106         Employepan.setLayout(new BorderLayout());
107         Employepan.add(Display_Table_employe, BorderLayout.CENTER);
108         Tableau.setFillsViewportHeight(true);
109         Dimension preferredSize = new Dimension(900, 500);
110         Tableau.setPreferredScrollableViewportSize(preferredSize);
111         pan0.add(new JScrollPane(Tableau), BorderLayout.CENTER);
112         Display_Table_employe.add(pan0);
113
114         Employepan.add(panButton_employe, BorderLayout.SOUTH);
115         panButton_employe.add(addButton_employe);
116         panButton_employe.add(updateButton_employe);
117         panButton_employe.add(deleteButton_employe);
118         panButton_employe.add(displayButton_employe);
119
120         Employepan.add(Forme_employe, BorderLayout.NORTH);
121         Forme_employe.setLayout(new GridLayout(7, 2, 10, 10));
122         Forme_employe.add(label_nom);
123         Forme_employe.add(text_nom);
124         Forme_employe.add(label_prenom);
125         Forme_employe.add(text_prenom);
126         Forme_employe.add(label_email);
127         Forme_employe.add(text_email);
128         Forme_employe.add(label_tele);
129         Forme_employe.add(text_tele);
130         Forme_employe.add(label_salaire);
131         Forme_employe.add(text_salaire);
132         Forme_employe.add(label_role);
133         Forme_employe.add(roleComboBox);
134         Forme_employe.add(label_poste);
```

```
135         Forme_employe.add(posteComboBox);
136 
137     // Holiday Tab
138         holidayTab.setLayout(new BorderLayout());
139         holidayTab.add(Holidaypan, BorderLayout.CENTER);
140         Holidaypan.setLayout(new BorderLayout());
141         Holidaypan.add(Display_Table_holiday, BorderLayout.CENTER);
142 
143         Tableau1.setFillsViewportHeight(true);
144         Tableau1.setPreferredScrollableViewportSize(preferredSize);
145         pan1.add(new JScrollPane(Tableau1), BorderLayout.CENTER);
146         Display_Table_holiday.add(pan1);
147 
148         Holidaypan.add(Forme_holiday, BorderLayout.NORTH);
149         Forme_holiday.setLayout(new GridLayout(4, 2, 10, 10));
150         Forme_holiday.add(label_employe);
151         Forme_holiday.add(text_employe);
152         Forme_holiday.add(label_startDate);
153         Forme_holiday.add(text_startDate);
154         Forme_holiday.add(label_endDate);
155         Forme_holiday.add(text_endDate);
156         Forme_holiday.add(label_type);
157         Forme_holiday.add(TypeComboBox);
158 
159         Holidaypan.add(panButton_holiday, BorderLayout.SOUTH);
160         panButton_holiday.add(addButton_holiday);
161         panButton_holiday.add(updateButton_holiday);
162         panButton_holiday.add(deleteButton_holiday);
163         panButton_holiday.add(displayButton_holiday);
164 
165     // TabbedPane
166         tabbedPane.addTab("Employe", employeTab);
167         tabbedPane.addTab("Holiday", holidayTab);
168 
169 
170         remplir_les_employes();
171         setVisible(true);
172     }
173 
174     public void remplir_les_employes () {
175         List<Employe> Employes = new Employemodel(new EmployeDAOimpl()).
    displayEmploye();
176         text_employe.removeAllItems();
177         for (Employe elem : Employes) {
178             text_employe.addItem(elem.getId() + " - " + elem.getNom()+" "
    +elem.getPrenom());
179         }
180     }
181 
```

```
182     // getters
183         public int getId_employe() {
184             return Integer.parseInt(text_employe.getSelectedItem().
    toString().split(" - ")[0]);
185         }
186         public String getNom() {
187             return text_nom.getText();
188         }
189 
190         public JTable getTable() {
191             return (JTable) Display_Table_employe.getComponent(0);
192         }
193 
194         public String getPrenom() {
195             return text_prenom.getText();
196         }
197 
198         public String getEmail() {
199             return text_email.getText();
200         }
201 
202         public String getTelephone() {
203             return text_tele.getText();
204         }
205 
206         public double getSalaire() {
207             return Double.parseDouble(text_salaire.getText());
208         }
209 
210         public Role getRole() {
211             return (Role) roleComboBox.getSelectedItem();
212         }
213 
214         public Post getPoste() {
215             return (Post) posteComboBox.getSelectedItem();
216         }
217 
218         public JButton getaddButton_employe () {
219             return addButton_employe;
220         }
221 
222         public JButton getupdateButton_employe () {
223             return updateButton_employe;
224         }
225 
226         public JButton getdeleteButton_employe () {
227             return deleteButton_employe;
228         }
229 
```

```
230         public JButton getdisplayButton_employe () {
231             return displayButton_employe;
232         }
233 
234         public JButton getaddButton_holiday () {
235             return addButton_holiday;
236         }
237 
238         public JButton getupdateButton_holiday () {
239             return updateButton_holiday;
240         }
241         public JButton getdeleteButton_holiday () {
242             return deleteButton_holiday;
243         }
244 
245         public JButton getdisplayButton_holiday () {
246             return displayButton_holiday;
247         }
248         public String getStartDate () {
249             return text_startDate.getText();
250         }
251 
252         public String getEndDate() {
253             return text_endDate.getText();
254         }
255 
256         public Type_holiday getType_holiday(){
257             return (Type_holiday) TypeComboBox.getSelectedItem();
258         }
259 
260      // methods d'affichage des messages
261         public void afficherMessageErreur(String message) {
262             JOptionPane.showMessageDialog(this, message, "Erreur",
    JOptionPane.ERROR_MESSAGE);
263         }
264 
265         public void afficherMessageSucces(String message) {
266             JOptionPane.showMessageDialog(this, message, "Succes",
    JOptionPane.INFORMATION_MESSAGE);
267         }
268 
269      // methodes de vider les champs
270         public void viderChamps_em() {
271             text_nom.setText("");
272             text_prenom.setText("");
273             text_email.setText("");
274             text_tele.setText("");
275             text_salaire.setText("");
276             roleComboBox.setSelectedIndex(0);
```

```
277             posteComboBox.setSelectedIndex(0);
278         }
279 
280         public void viderChamps_ho() {
281             text_startDate.setText("");
282             text_endDate.setText("");
283             TypeComboBox.setSelectedIndex(0);
284         }
285 
286      // methodes de remplir les champs
287         public void remplaireChamps_em (int id, String nom, String
    prenom, String email, String telephone, double salaire, Role role,
    Post poste) {
288             text_nom.setText(nom);
289             text_prenom.setText(prenom);
290             text_email.setText(email);
291             text_tele.setText(telephone);
292             text_salaire.setText(String.valueOf(salaire));
293             roleComboBox.setSelectedItem(role);
294             posteComboBox.setSelectedItem(poste);
295         }
296 
297         public void remplaireChamps_ho(int id_employe, String date_debut
    , String date_fin, Type_holiday type) {
298             List<Employe> Employes = new Employemodel(new EmployeDAOimpl
    ()).displayEmploye();
299             text_employe.removeAllItems();
300             for (Employe elem : Employes) {
301                 if (elem.getId() == id_employe) {
302                     text_employe.addItem(elem.getId() + " - " + elem.
    getNom()+" "+elem.getPrenom());
303                     text_employe.setSelectedItem(elem.getId() + " - " +
    elem.getNom()+" "+elem.getPrenom());
304                 }
305             }
306             text_startDate.setText(date_debut);
307             text_endDate.setText(date_fin);
308             TypeComboBox.setSelectedItem(type);
309         }
310 
311      // methodes de test des champs
312         public boolean testChampsVide_em (){
313             return text_nom.getText().equals("") || text_prenom.getText
    ().equals("") || text_email.getText().equals("") || text_tele.getText
    ().equals("") || text_salaire.getText().equals("");
314         }
315 
316         public boolean testChampsVide_ho () {
317             return text_employe.getSelectedItem().equals("") ||
```

```
    text_startDate.getText().equals("") || text_endDate.getText().equals(
    "") || TypeComboBox.getSelectedItem().equals("");
318         }
319         public static void main(String[] args) {
320             new Employe_HolidayView();
321         }
322
323 }
```

LoginView

```
1  package view;
2
3  import javax.swing.*;
4  import java.awt.*;
5  import java.awt.event.ActionListener;
6
7  public class LoginView extends JFrame {
8
9      private JTextField usernameField;
10     private JPasswordField passwordField;
11     private JButton loginButton;
12
13     public LoginView() {
14         setTitle("Login");
15         setSize(1000, 600);
16         setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
17
18         usernameField = new JTextField(20);
19         passwordField = new JPasswordField(20);
20         loginButton = new JButton("Login");
21
22         // Cr er un panneau avec GridBagLayout
23         JPanel panel = new JPanel();
24         panel.setLayout(new GridBagLayout());
25         GridBagConstraints gbc = new GridBagConstraints();
26
27         // D finir les marges (padding) entre les   lments
28         gbc.insets = new Insets(10, 10, 10, 10);
29
30         // Ajouter le label et le champ username
31         gbc.gridx = 0;
32         gbc.gridy = 0;
33         panel.add(new JLabel("Username:"), gbc);
34
35         gbc.gridx = 1;
36         panel.add(usernameField, gbc);
37
```

```
38          // Ajouter le label et le champ password
39          gbc.gridx = 0;
40          gbc.gridy = 1;
41          panel.add(new JLabel("Password:"), gbc);
42
43          gbc.gridx = 1;
44          panel.add(passwordField, gbc);
45
46          // Ajouter le bouton Login
47          gbc.gridx = 1;
48          gbc.gridy = 2;
49          panel.add(loginButton, gbc);
50
51          // Ajouter le panneau    la fen tre et le centrer
52          setLayout(new BorderLayout());
53          add(panel, BorderLayout.CENTER);
54
55          // Centrer la fen tre    l' cran
56          setLocationRelativeTo(null);
57      }
58
59      public String getUsername() {
60          return usernameField.getText();
61      }
62
63      public String getPassword() {
64          return new String(passwordField.getPassword());
65      }
66
67      public void addLoginListener(ActionListener listener) {
68          loginButton.addActionListener(listener);
69      }
70
71      public void showError(String message) {
72          JOptionPane.showMessageDialog(this, message, "Error",
    JOptionPane.ERROR_MESSAGE);
73      }
74
75      public void close() {
76          this.setVisible(false);
77      }
78
79      public void showSuccess(String message) {
80          JOptionPane.showMessageDialog(this, message, "Success",
    JOptionPane.INFORMATION_MESSAGE);
81      }
82  }
```

2.4 Controller

Le contrôleur gère les actions de l'utilisateur. Il reçoit les événements de la vue, interagit avec le modèle pour effectuer des opérations (par exemple, ajout, modification, suppression de données), puis met à jour la vue en conséquence.

HolidayController

```
1
2  package Controller;
3
4  import Model.*;
5
6  import view.*;
7
8  import java.sql.Date;
9  import java.util.Calendar;
10 import java.util.List;
11
12 import javax.swing.table.DefaultTableModel;
13
14
15
16 public class EmployeController {
17
18     private final Employe_HolidayView View;
19     public static Employemodel model_employe ;
20     public static int id = 0;
21     public static int oldselectedrow = -1;
22     public static boolean test = false;
23     String nom = "";
24     String prenom = "";
25     String email = "";
26     String telephone = "";
27     double salaire = 0;
28     Role role = null;
29     Post poste = null;
30     int solde = 0;
31     boolean updatereussi = false;
32
33     public EmployeController(Employe_HolidayView view, Employemodel
       model) {
34         this.View = view;
35         this.model_employe = model;
36         View.getaddButton_employe().addActionListener(e -> addEmploye())
       ;
37         View.getdeleteButton_employe().addActionListener(e ->
       deleteEmploye());
38         View.getupdateButton_employe().addActionListener(e ->
```

```
updateEmploye());
39          View.getdisplayButton_employe().addActionListener(e ->
displayEmploye());
40          Employe_HolidayView.Tableau.getSelectionModel().
addListSelectionListener(e -> updateEmployebyselect());
41      }
42
43
44
45      public void displayEmploye() {
46          List<Employe> Employes = model_employe.displayEmploye();
47          if(Employes.isEmpty()){
48              View.afficherMessageErreur("Aucun employe.");
49          }
50          DefaultTableModel tableModel = (DefaultTableModel)
Employe_HolidayView.Tableau.getModel();
51          tableModel.setRowCount(0);
52          for(Employe e : Employes){
53              tableModel.addRow(new Object[]{e.getId(), e.getNom(), e.
getPrenom(), e.getEmail(), e.getTelephone(), e.getSalaire(), e.
getRole(), e.getPost(),e.getSolde()});
54          }
55          View.remplir_les_employes();
56      }
57
58
59      // function of add Employe :
60
61      private void addEmploye() {
62          String nom = View.getNom();
63          String prenom = View.getPrenom();
64          String email = View.getEmail();
65          String telephone = View.getTelephone();
66          double salaire = View.getSalaire();
67          Role role = View.getRole();
68          Post poste = View.getPoste();
69
70          View.viderChamps_em();
71          boolean addreussi = model_employe.addEmploye(0,nom, prenom,
email, telephone, salaire, role, poste ,25);
72
73          if(addreussi == true){
74              View.afficherMessageSucces("L'employe a bien ete ajoutee.");
75              displayEmploye();
76          }else{
77              View.afficherMessageErreur("L'employe n'a pas ete ajoutee.")
;
78          }
79      }
```

```
80
81
82
83      // function of delete Employe :
84
85      private void deleteEmploye(){
86          int selectedrow = Employe_HolidayView.Tableau.getSelectedRow();
87          if(selectedrow == -1){
88              View.afficherMessageErreur("Veuillez selectionner une ligne.
    ");
89          }else{
90              int id = (int) Employe_HolidayView.Tableau.getValueAt(
    selectedrow, 0);
91              if(model_employe.deleteEmploye(id)){
92                  View.afficherMessageSucces("L'employe a bien ete
    supprimer.");
93                  displayEmploye();
94              }else{
95                  View.afficherMessageErreur("L'employe n'a pas ete
    supprimer.");
96              }
97          }
98      }
99
100     // function of Update :
101
102     private void updateEmployebyselect(){
103         int selectedrow = Employe_HolidayView.Tableau.getSelectedRow();
104
105         if (selectedrow == -1) {
106             return;
107         }
108         try{
109             id = (int) Employe_HolidayView.Tableau.getValueAt(
    selectedrow, 0);
110             nom = (String) Employe_HolidayView.Tableau.getValueAt(
    selectedrow, 1);
111             prenom = (String) Employe_HolidayView.Tableau.getValueAt(
    selectedrow, 2);
112             email = (String) Employe_HolidayView.Tableau.getValueAt(
    selectedrow, 3);
113             telephone = (String) Employe_HolidayView.Tableau.getValueAt(
    selectedrow, 4);
114             salaire = (double) Employe_HolidayView.Tableau.getValueAt(
    selectedrow, 5);
115             role = (Role) Employe_HolidayView.Tableau.getValueAt(
    selectedrow, 6);
116             poste = (Post) Employe_HolidayView.Tableau.getValueAt(
    selectedrow, 7);
```

```
117             solde = (int) Employe_HolidayView.Tableau.getValueAt(
    selectedrow, 8);
118             View.remplaireChamps_em(id, nom, prenom, email, telephone,
    salaire, role, poste);
119             test = true;
120             displayEmploye();
121 
122         }catch(Exception e){
123              View.afficherMessageErreur("Erreur lors de la recuperation
    des donn es");
124         }
125      }
126 
127      private void updateEmploye(){
128         if (!test) {
129             View.afficherMessageErreur("Veuillez d'abord selectionner
    une ligne a modifier.");
130             return;
131         }
132         try {
133             nom = View.getNom();
134             prenom = View.getPrenom();
135             email = View.getEmail();
136             telephone = View.getTelephone();
137             salaire = View.getSalaire();
138             role = View.getRole();
139             poste = View.getPoste();
140 
141             boolean updateSuccessful = model_employe.updateEmploye(id,
    nom, prenom, email, telephone, salaire, role, poste , solde);
142 
143             if (updateSuccessful) {
144                 test = false;
145                 View.afficherMessageSucces("L'employe modifier avec
    succes.");
146                 displayEmploye();
147                 View.viderChamps_em();
148             } else {
149                 View.afficherMessageErreur("Erreur lors de la mise a
    jour de l'employe");
150             }
151         } catch (Exception e) {
152 
153             View.afficherMessageErreur("Erreur lors de la mise a jour");
154         }
155      }
156 
157      public void resetSolde(){
158         Calendar now = Calendar.getInstance();
```

```
159         if(now.get(Calendar.DAY_OF_YEAR) == 1){
160             for (Employe employe : model_employe.displayEmploye()) {
161                 updateSolde(employe.getId(), 25);
162             }
163         }
164         displayEmploye();
165 
166     }
167 
168     public static void updateSolde(int  id , int solde){
169         boolean updateSuccessful = model_employe.updateSolde(id, solde);
170     }
171 }
```

GenericDAOI

```
1  package Controller;
2 
3 
4  import Model.LoginModel;
5  import view.LoginView;
6 
7  import java.awt.event.ActionEvent;
8  import java.awt.event.ActionListener;
9 
10 public class LoginController {
11 
12     private final LoginView loginView;
13     private final LoginModel loginModel;
14 
15     public LoginController(LoginView loginView, LoginModel loginModel) {
16         this.loginView = loginView;
17         this.loginModel = loginModel;
18 
19         // Attach the login button listener
20         this.loginView.addLoginListener(new LoginListener());
21     }
22 
23     private class LoginListener implements ActionListener {
24 
25         @Override
26         public void actionPerformed(ActionEvent e) {
27             // Get the username and password from the login view
28             String username = loginView.getUsername();
29             String password = loginView.getPassword();
30 
31             // Validate input fields
32             if (username.isEmpty() || password.isEmpty()) {
33                 loginView.showError("Username or password cannot be
```

```
    empty.");
34                  return;
35              }
36
37              // Attempt to authenticate the user using the model
38              boolean isAuthenticated = loginModel.authenticate(username,
    password);
39
40              if (isAuthenticated) {
41                  // If authentication is successful, notify the user
42                  loginView.showSuccess("Login successful!");
43                  loginView.close(); // Close the login view or proceed to
     the next view
44              } else {
45                  // If authentication fails, show an error message
46                  loginView.showError("Invalid username or password.
    Please try again.");
47              }
48          }
49      }
50  }
```

2.5 MAIN

Main

```
1  package Main;
2
3
4  import Controller.EmployeController;
5  import Controller.HolidayController;
6  import Controller.LoginController;
7  import DAO.EmployeDAOimpl;
8  import DAO.HolidayDAOimpl;
9  import DAO.LoginDAOimpl;
10 import Model.Employemodel;
11 import Model.HolidayModel;
12 import Model.LoginModel;
13 import view.Employe_HolidayView;
14 import view.LoginView;
15
16 public class Main {
17     public static void main(String[] args) {
18
19         LoginDAOimpl loginDAO = new LoginDAOimpl();
20         EmployeDAOimpl employeDAO = new EmployeDAOimpl();
21         HolidayDAOimpl holidayDAO = new HolidayDAOimpl();
```

```
22
23
24          LoginModel loginModel = new LoginModel(loginDAO);
25          Employemodel employeModel = new Employemodel(employeDAO);
26          HolidayModel holidayModel = new HolidayModel(holidayDAO);
27
28
29          LoginView loginView = new LoginView();
30          Employe_HolidayView employeHolidayView = new Employe_HolidayView
        ();
31
32
33          new LoginController(loginView, loginModel);
34
35          // Show the login view
36          loginView.setVisible(true);
37
38          // Listen for login success to show the main application
39          loginView.addLoginListener(e -> {
40              if (loginModel.authenticate(loginView.getUsername(),
        loginView.getPassword())) {
41                  // Login successful
42                  loginView.setVisible(false); // Hide login view
43
44                  // Initialize controllers for employee and holiday
        management
45                  new EmployeController(employeHolidayView, employeModel);
46                  new HolidayController(employeHolidayView, holidayModel);
47
48                  // Show the main application view
49                  employeHolidayView.setVisible(true);
50              } else {
51                  // Login failed
52                  loginView.showError("Invalid username or password.
        Please try again.");
53              }
54          });
55      }
56  }
```

Résultats

1 Résultats de la partie View

La couche View représente l'interface utilisateur de l'application et permet l'interaction entre l'utilisateur et le système. Dans ce projet, l'interface a été conçue avec le framework Swing en Java, qui fournit des composants graphiques riches et personnalisables.

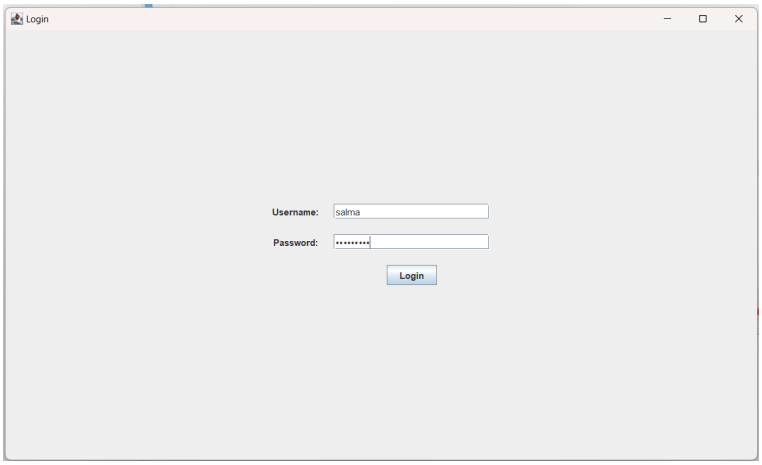
- 

FIGURE 2.1 – Interface de Login

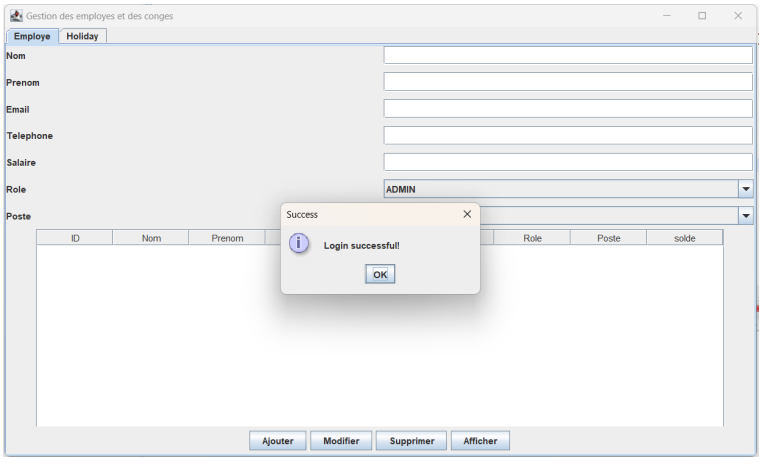


FIGURE 2.2 – Seccess de Login

2 Affichage des Conges

L'application affiche une liste des congés avec leurs informations principales. L'utilisateur peut sélectionner un congé et cliquer sur "Afficher" pour voir ses détails de manière simple et claire.

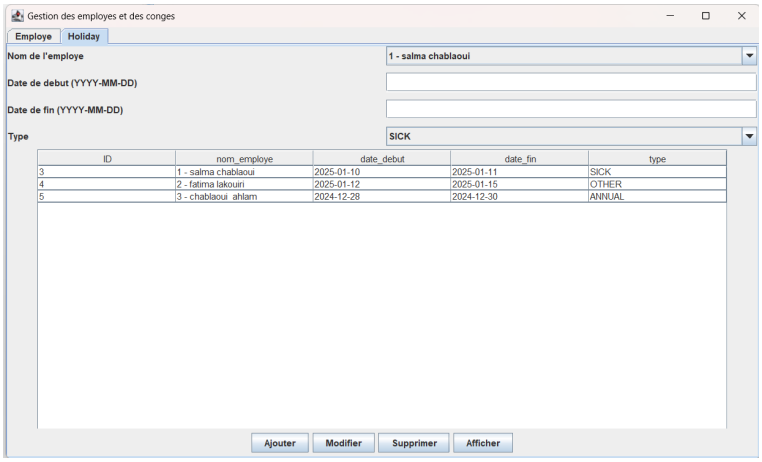


FIGURE 2.3 – Resultat d'afficher

3 Après Ajout

Lors de l'ajout d'un congé, une vérification est effectuée pour s'assurer que le solde de congés de l'employé ne dépasse pas 25 jours. Cette condition garantit une gestion conforme et équilibrée des droits de congés pour chaque employé.

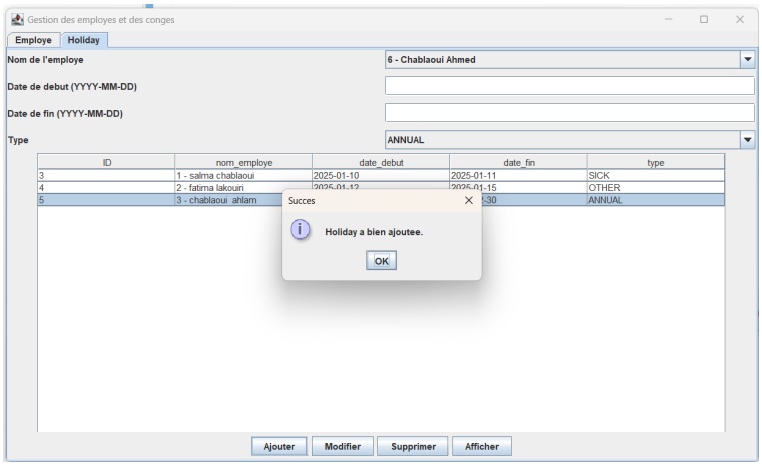


FIGURE 2.4 – Resultat d'Ajout

4 Après modification

Après la mise à jour d'un congé, les nouvelles informations saisies par l'utilisateur dans le panneau d'entrée sont validées et transmises à la couche Controller, qui assure leur traitement via la logique

métier.

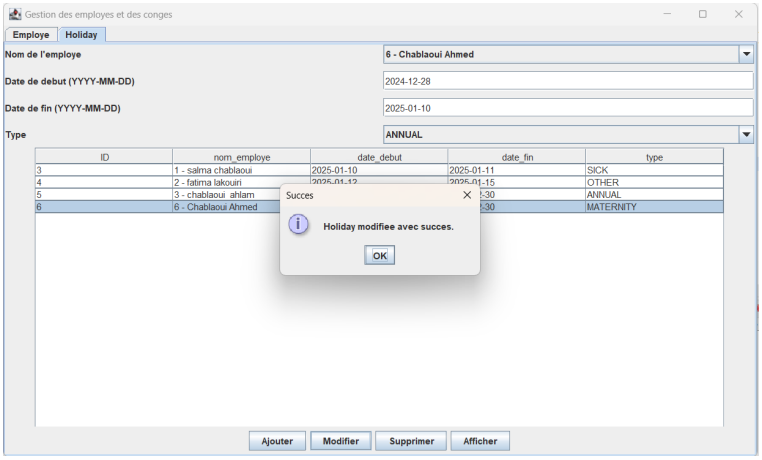


FIGURE 2.5 – Resultat de modification

5 Apres Suppression

Lorsqu'un congé est supprimé, l'utilisateur sélectionne l'congé concerné dans la liste affichée et confirme l'action en cliquant sur le bouton Supprimer. Cette demande est transmise à la couche Controller, qui s'assure de la suppression de l'enregistrement via la logique métier.

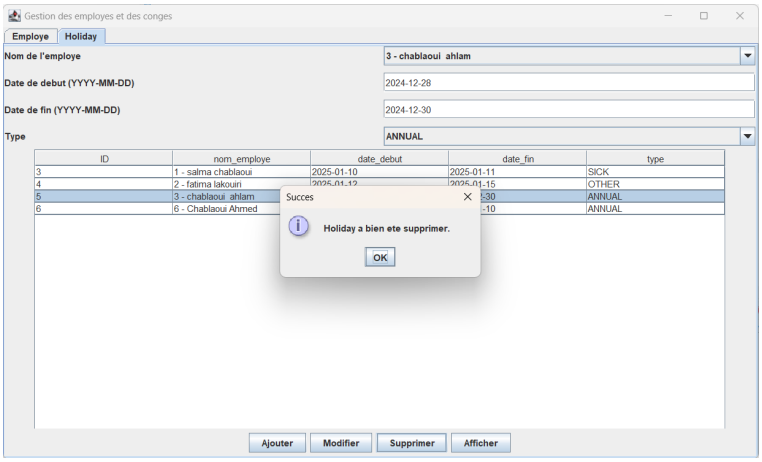
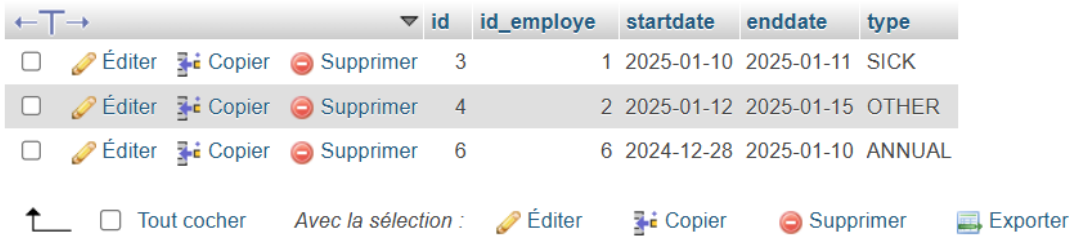


FIGURE 2.6 – Resultat de suppression

Tous ca est stocké dans la base de donnée voici le resultat.



	id	id_employe	startdate	enddate	type
Éditer Copier Supprimer	3	1	2025-01-10	2025-01-11	SICK
Éditer Copier Supprimer	4	2	2025-01-12	2025-01-15	OTHER
Éditer Copier Supprimer	6	6	2024-12-28	2025-01-10	ANNUAL

Tout cocher *Avec la sélection :* Éditer Copier Supprimer Exporter

FIGURE 2.7 – affichage de tous ce qui est stocké à la base de donnée

Conclusion générale

Ce travail pratique a permis de concevoir et de mettre en œuvre une application complète de gestion des congés, offrant des fonctionnalités essentielles adaptées aux besoins d'une entreprise. Parmi celles-ci, on retrouve l'affichage clair et organisé des congés, la possibilité d'ajouter de nouveaux congés en respectant des conditions spécifiques, telles que la vérification du solde disponible, et la visualisation détaillée des informations relatives à chaque congé.

L'application a été pensée pour garantir une utilisation intuitive, avec une interface graphique conviviale permettant de simplifier les processus de gestion. Des validations ont été mises en place pour assurer le respect des règles définies, comme la limitation du solde de congés à 25 jours par employé, ce qui contribue à une meilleure organisation et équité dans l'entreprise.

Ce projet nous a permis de développer nos compétences techniques dans plusieurs domaines. Nous avons consolidé nos connaissances en programmation orientée objet, en particulier pour le développement d'interfaces graphiques, et nous avons également approfondi notre maîtrise des bases de données, tant au niveau de la conception que de l'interaction avec des requêtes SQL.

En répondant à un besoin réel, ce travail pratique illustre l'importance des solutions numériques dans la gestion efficace des ressources humaines et constitue une expérience enrichissante pour notre formation professionnelle.

Références

java :

— https://www.java.com/en/download/
Eclipse :
— https://www.eclipse.org/downloads/
XAMPP :
— https://www.apachefriends.org/fr/index.html
jdk 23 :
— https://www.oracle.com/java/technologies/downloads/