Université Cadi Ayyad
École Supérieure De Technologie - Safi
Département : Informatique
Filière : Génie Informatique (GI)

Compte Rendu TP2

Gestion des Congés en Java (DAO/MVC)

Réalisée par : DOUAE SAMTI

Encadrée par : Mme Asmae Elkourchi

Contents

1 Introduction **2**

2 Architecture et Organisation du Projet **2**
2.1 Architecture du Projet 2
2.2 Organisation des Packages 2
2.3 `MainView` 2

3 Ètapes Suivies **3**

4 Explications des Codes **3**
4.1 Modèle (`Holiday`, `HolidayModel`, `Type`) 3
4.2 Classe `Holiday` 4
4.3 Modèle (`Holiday`, `Type`) 4
4.4 Classe `HolidayModel` 5
4.5 Connexion à la base de données (`DBConnection`) 7
4.6 Interface `GenericDAOI` et classe `HolidayDAOImpl` 8
4.7 `View : ViewMain` 11
4.8 Classe `HolidayController` 16
4.9 Classe `Main` 22

5 Fonctionnement de l'application **23**
5.1 Ajout d'une demande de congé 23
5.2 Affichage des demandes de congé 23
5.3 Suppression d'une demande de congé 24
5.4 Modification d'une demande de congé 24
5.5 Modification et vérification des dates de congé 25
5.6 Modification, vérification des dates et gestion du solde 26

6 Conclusion **26**

1 Introduction

Le système de gestion des congés a été intégré au projet existant de gestion des employés pour permettre une administration efficace des absences des employés. Cette fonctionnalité inclut la planification, la modification et le suivi des congés en fonction des types définis (payé, non payé, maladie). Ce rapport détaille la structure et le fonctionnement de ce module, y compris les composants DAO, Modèle et Vue.

2 Architecture et Organisation du Projet

2.1 Architecture du Projet

L'architecture du projet repose sur le modèle MVC, séparant la logique en trois couches principales :

1. **Modèle (Model)** : Cette couche contient les règles métier et les classes représentant les entités, telles que `Holiday` (pour les congés) et `Type` (pour les catégories de congés).
2. **Vue (View)** : Elle est responsable de l'interface utilisateur et de la présentation des données, utilisant des composants Swing comme `JTable` et `JComboBox`.
3. **Contrôleur (Controller)** : Cette partie connecte la vue et le modèle, tout en gérant les interactions utilisateur. Par exemple, la classe `HolidayController` traite les événements des boutons (ajouter, modifier, supprimer) et assure la mise à jour des données entre la vue et le modèle.

2.2 Organisation des Packages

Le projet est divisé en quatre packages principaux :

- **DAO** : Contient la gestion de la connexion à la base de données et les classes pour les opérations CRUD.
- **Controller** : Inclut les classes responsables de la logique et des interactions entre la vue et le modèle, comme `HolidayController`.
- **Model** : Comprend les entités et les structures de données, notamment `Holiday`, `HolidayModel`, et `Type`.
- **View** : Contient les interfaces utilisateur, y compris `MainView`, qui combine la gestion des employés et la gestion des congés.

2.3 `MainView`

`MainView` offre une interface utilisateur unifiée pour gérer à la fois les employés et les congés. Elle intègre des composants interactifs comme des tableaux et des formulaires pour faciliter la navigation et la gestion.

3 Ètapes Suivies

Pour réaliser ce projet, nous avons suivi les étapes ci-dessous :

1. **Conception de la base de données** :
   - Création d'une table `Holiday`, structurée pour contenir les données liées aux congés, telles que l'identifiant, le nom de l'employé, les dates de début et de fin, ainsi que le type de congé.
2. **Développement du module DAO** :
   - Èlaboration d'une interface générique `GenericDAO` pour définir les opérations CRUD standard.
   - Implémentation de cette interface dans `HolidayDAOImpl`, en utilisant des requêtes SQL paramétrées pour interagir avec la base de données.
3. **Création des classes du modèle** :
   - Mise en place de la classe `Holiday`, qui représente un congé avec des propriétés comme `id`, `employeeName`, `startDate`, `endDate`, et `type`.
   - Définition de l'énumération `Type` pour représenter les différents types de congés, tels que `PAYE`, `NON PAYE` et `MALADIE`,.
4. **Développement de la vue** :
   - Conception d'une interface conviviale avec Swing, permettant de gérer les congés via des formulaires et des tableaux interactifs.
5. **Développement du contrôleur** :
   - Intégration de la logique pour gérer les actions des utilisateurs (ajout, suppression, modification) et assurer la synchronisation des données entre la base et l'interface utilisateur.

4 Explications des Codes

4.1 Modèle (`Holiday`, `HolidayModel`, `Type`)

- **Objectif** : Le modèle gère les données manipulées par l'application.
- **Détails** :
  - `Holiday` : Cette classe encapsule les informations relatives aux congés, avec des propriétés telles que `id`, `employeeName`, `startDate`, `endDate`, et `type`. Elle inclut également des `getters` et `setters` pour accéder à ces données.
  - `HolidayModel` : Responsable de la logique métier liée à la gestion des congés, comme la validation des dates ou le calcul des durées.
  - `Type` : Une énumération définissant les différentes catégories de congés, comme `PAYE`, `NON PAYE`,`MALADIE`, afin d'assurer la validité des données.

4.2 Classe `Holiday`

Listing 1: Classe Holiday

```
1
2  package Model;
3
4  import java.sql.Date;
5
6  public class Holiday{
7      private int id_holiday;
8      private int id_employe;
9      private Date startDate;
10     private Date endDate;
11     private HolidayType type;
12
13 public Holiday(int id_holiday, int id_employe,Date startDate, Date
       endDate , HolidayType type){
14     this.id_holiday = id_holiday;
15     this.id_employe = id_employe;
16     this.startDate = startDate;
17     this.endDate = endDate;
18     this.type = type;
19 }
20 public int getId_holiday() {
21     return id_holiday;
22 }
23 public Date getStartDate() {
24     return startDate;
25 }
26 public Date getEndDate() {
27     return endDate;
28 }
29 public HolidayType getType() {
30     return type;
31 }
32 public int getId_employe() {
33     return id_employe;
34 }
35     public Object getSolde() {
36         throw new UnsupportedOperationException("Not␣supported␣yet.");
37     }
38 }
```

4.3 Modèle (`Holiday`, `Type`)

- **Objectif** : Modéliser les informations liées aux congés au sein de l'application.
- **Détails** :
  - `Holiday` regroupe des propriétés telles que `employeeId`, `startDate`, `endDate`, etc., avec des méthodes `getters` et `setters` pour accéder et modifier ces données.
  - `Type` est une énumération qui spécifie les différentes catégories de congés (maladie, payé, non payé).

- **Code Source**

Listing 2: Enumération Type

Enumération `Type`

```
package Model;

public enum Type {
    CONGE_MALADIE,
    CONGE_PAYE,
    CONGE_NON_PAYE
}
```

4.4 Classe `HolidayModel`

- **Objectif** : La classe `HolidayModel` est responsable de la gestion des congés des employés. Elle interagit avec le DAO `HolidayDAOImpl` pour ajouter, afficher, supprimer et mettre à jour des congés dans la base de données.
- **Détails** :
  - `HolidayDAOImpl` : La classe de persistance qui permet d'interagir avec la base de données pour les congés.
  - `addHoliday` : Cette méthode permet d'ajouter un congé après avoir vérifié la validité des dates et du solde de congé de l'employé.
  - `displayHoliday` : Cette méthode permet d'afficher tous les congés enregistrés dans la base de données.
  - `deleteHoliday` : Cette méthode permet de supprimer un congé en fonction de son identifiant.
  - `updateHoliday` : Cette méthode permet de mettre à jour un congé en vérifiant si le solde de congé de l'employé permet d'effectuer la modification.
- **Code Source** :

Classe `HolidayModel` :

Listing 3: Code de la classe `HolidayModel`

```
package Model;

import Controller.EmployeController;
import java.util.List;
import DAO.HolidayDAOImpl;
import java.sql.Date;

public class HolidayModel {
    private HolidayDAOImpl dao;

    public HolidayModel(HolidayDAOImpl dao) {
        this.dao = dao;
    }

```

```
15      public boolean addHoliday(int id, int id_employe, Date startdate,
            Date enddate, HolidayType type , Employe targetEmploye) {
16          // V rification des dates
17          if(startdate.after(enddate)) return false;
18          if(startdate.equals(enddate)) return false;
19          if(startdate.before(new Date(System.currentTimeMillis())))
                return false;
20          if(enddate.before(new Date(System.currentTimeMillis()))) return
                false;
21
22          // V rification du solde de cong
23          long daysBetween = (enddate.toLocalDate().toEpochDay() -
                startdate.toLocalDate().toEpochDay());
24          if(daysBetween > targetEmploye.getSolde()) return false;
25
26          // Mise    jour du solde d'un employ
27          EmployeController.updateSolde(targetEmploye.getId(),
                targetEmploye.getSolde() - (int) daysBetween);
28
29          // Cr ation du cong  et ajout dans la base de donn es
30          Holiday e = new Holiday(id, id_employe, startdate, enddate,
                type);
31          dao.add(e);
32          return true;
33      }
34
35      public List<Holiday> displayHoliday() {
36          // Retourne la liste des cong s
37          List<Holiday> Holidays = dao.display();
38          return Holidays;
39      }
40
41      public boolean deleteHoliday(int id) {
42          // Supprime un cong  en fonction de son ID
43          dao.delete(id);
44          return true;
45      }
46
47      public boolean updateHoliday(int id, int id_employe, Date startdate
            , Date enddate, HolidayType type , Employe targetEmploye , int
            olddaysbetween ) {
48          // Calcul du nombre de jours entre les dates
49          long daysBetween = (enddate.toLocalDate().toEpochDay() -
                startdate.toLocalDate().toEpochDay());
50
51          // V rification des dates
52          if(startdate.after(enddate)) return false;
53          if(startdate.equals(enddate)) return false;
54          if(startdate.before(new Date(System.currentTimeMillis())))
                return false;
55          if(enddate.before(new Date(System.currentTimeMillis()))) return
                false;
56
```

```
57          // V rification du solde pour la mise    jour
58          if(daysBetween > (targetEmploye.getSolde() + olddaysbetween))
               return false;
59
60          // Mise    jour du solde d'un employ
61          EmployeController.updateSolde(targetEmploye.getId(), (
               targetEmploye.getSolde() + olddaysbetween) - (int)
               daysBetween);
62
63          // Mise    jour du cong  dans la base de donn es
64          Holiday e = new Holiday(id, id_employe, startdate, enddate,
               type);
65          dao.update(e);
66          return true;
67      }
68  }
```

4.5 Connexion à la base de données (DBConnection)

- **Objectif** : Assurer l'établissement et la gestion de la connexion avec la base de données MySQL dédiée à la gestion des congés.
- **Détails** :
  - Utilise le pilote JDBC pour établir une connexion à la base de données nommée conge.
  - Gère les exceptions SQL en fournissant des messages d'erreur clairs et adaptés.
- **Code Source**

Listing 4: Connexion à la base de données

Classe 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      private static final String URL = "jdbc:mysql://localhost:3306/
          conge";
9      private static final String USER = "root";
10     private static final String PASSWORD = "";
11
12     public static Connection getConnection() throws SQLException {
13         return DriverManager.getConnection(URL, USER, PASSWORD);
14     }
15 }
```

4.6 Interface `GenericDAOI` et classe `HolidayDAOImpl`

- **Objectif** : L'interface `GenericDAOI` définit les méthodes génériques pour effectuer des opérations CRUD (ajouter, supprimer, mettre à jour et afficher des entités), tandis que la classe `HolidayDAOImpl` implémente cette interface pour gérer spécifiquement les opérations sur les entités `Holiday`.
- **Détails** :
  - `GenericDAOI` : Cette interface définit les méthodes génériques suivantes :
    - `add(T e)` : Ajoute une entité.
    - `delete(int id)` : Supprime une entité en utilisant son identifiant.
    - `update(T e)` : Met à jour une entité.
    - `display()` : Affiche toutes les entités.
  - `HolidayDAOImpl` : Cette classe implémente `GenericDAOI` pour la gestion des congés. Elle inclut une logique supplémentaire pour vérifier le solde de congé d'un employé et le mettre à jour après chaque demande de congé.
- **Code Source**

Interface `GenericDAOI`

Listing 5: Interface GenericDAOI

```java
package DAO;

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();
}
```

Classe `HolidayDAOImpl`

Listing 6: Classe HolidayDAOImpl

```java
package DAO;

import Model.Holiday;
import Model.HolidayType;
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␣=␣?"
            ;
```

```
17          String insertHolidaySql = "INSERT␣INTO␣holiday␣(id_employe,␣
                startdate,␣enddate,␣type)␣VALUES␣(?,␣?,␣?,␣?)";
18
19          try (PreparedStatement checkStmt = DBConnexion.getConnexion().
                prepareStatement(checkSoldeSql)) {
20              // V rifier le solde de cong  de l'employ
21              checkStmt.setInt(1, e.getId_employe());
22              ResultSet rs = checkStmt.executeQuery();
23
24              if (rs.next()) {
25                  int solde = rs.getInt("solde");
26
27                  // Calculer le nombre de jours demand s
28                  long daysBetween = java.time.temporal.ChronoUnit.DAYS.
                        between(
29                      e.getStartDate().toLocalDate(),
30                      e.getEndDate().toLocalDate()
31                  );
32
33                  if (daysBetween > solde) {
34                      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                      insertStmt.executeUpdate();
46
47                      // Mettre    jour le solde de cong
48                      String updateSoldeSql = "UPDATE␣employe␣SET␣solde=␣
                            solde␣-␣?␣WHERE␣id␣=␣?";
49                      try (PreparedStatement updateStmt = DBConnexion.
                            getConnexion().prepareStatement(updateSoldeSql))
                             {
50                          updateStmt.setInt(1, (int) daysBetween);
51                          updateStmt.setInt(2, e.getId_employe());
52                          updateStmt.executeUpdate();
53                      }
54                  }
55              } else {
56                  System.err.println("Employ ␣introuvable.");
57              }
58          } catch (SQLException exception) {
59              exception.printStackTrace();
60          }
61      }
```

```
62
63      @Override
64      public void delete(int id) {
65          String sql = "DELETE␣FROM␣holiday␣WHERE␣id␣=␣?";
66          try (PreparedStatement stmt = DBConnexion.getConnexion().
                prepareStatement(sql)) {
67              stmt.setInt(1,id);
68              stmt.executeUpdate();
69          } catch (SQLException exception) {
70              System.err.println(" chec ␣de␣la␣suppression␣du␣cong .");
71          }
72      }
73
74      @Override
75      public void update(Holiday e) {
76          String sql = "UPDATE␣holiday␣SET␣id_employe␣=␣?,␣startdate␣=␣?,
                ␣enddate␣=␣?,␣type␣=␣?␣WHERE␣id␣=␣?";
77          try (PreparedStatement stmt = DBConnexion.getConnexion().
                prepareStatement(sql)){
78              stmt.setInt(1, e.getId_employe());
79              stmt.setDate(2, e.getStartDate());
80              stmt.setDate(3, e.getEndDate());
81              stmt.setString(4, e.getType().name());
82              stmt.setInt(5, e.getId_holiday());
83              stmt.executeUpdate();
84          } catch (SQLException exception) {
85              System.err.println(" chec ␣de␣la␣mise␣  ␣jour␣du␣cong .");
86          }
87      }
88
89      @Override
90      public List<Holiday> display() {
91          String sql = "SELECT␣*␣FROM␣holiday";
92          List<Holiday> Holidays = new ArrayList<>();
93          try (PreparedStatement stmt = DBConnexion.getConnexion().
                prepareStatement(sql)) {
94              ResultSet re = stmt.executeQuery();
95              while (re.next()) {
96                  int id = re.getInt("id");
97                  int id_employe = re.getInt("id_employe");
98                  Date startdate = re.getDate("startdate");
99                  Date enddate = re.getDate("enddate");
100                 String type = re.getString("type");
101                 Holiday e = new Holiday(id, id_employe, startdate,
                        enddate, HolidayType.valueOf(type));
102                 Holidays.add(e);
103             }
104         } catch (SQLException ex) {
105             System.err.println(" chec ␣de␣la␣r cup ration␣des␣cong s
                    ␣:␣" + ex.getMessage());
106         }
107         return Holidays; // Retourne une liste vide si une erreur se
                produit
```

```
    }
}
```

4.7 View : ViewMain

La classe `MainView` est une fenêtre principale qui permet de gérer les employés et leurs congés. Elle utilise le framework `Swing` pour l'interface graphique. Elle comporte deux onglets principaux : un pour la gestion des employés et un autre pour la gestion des congés. Chaque onglet contient des panels et des boutons permettant de gérer les entités correspondantes.

Code Source

```java
package View;

import DAO.EmployeDAOImpl;
import DAO.EmployeDAOImpl;
import Model.Employe;
import Model.EmployeModel;
import Model.Poste;
import Model.Role;
import Model.EmployeModel;
import Model.HolidayType;
import java.awt.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public class MainView 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");
```

```
43
44      // les labels du conge
45      private JLabel label_employe = new JLabel("Nom de l'employe");
46      private JLabel label_startDate = new JLabel("Date de debut
            (YYYY-MM-DD)");
47      private JLabel label_endDate = new JLabel("Date de fin
            (YYYY-MM-DD)");
48      private JLabel label_type = new JLabel("Type");
49      private JComboBox<HolidayType> TypeComboBox = new
            JComboBox<>(HolidayType.values());
50
51      // les textfield du l'employe
52      private JTextField text_nom = new JTextField();
53      private JTextField text_prenom = new JTextField();
54      private JTextField text_email = new JTextField();
55      private JTextField text_tele = new JTextField();
56      private JTextField text_salaire = new JTextField();
57
58      private JComboBox<Role> roleComboBox = new
            JComboBox<>(Role.values());
59      private JComboBox<Poste> posteComboBox = new
            JComboBox<>(Poste.values());
60
61      // les textfield du conge
62      private JComboBox<String> text_employe = new JComboBox<>();
63      private JTextField text_startDate = new JTextField("");
64      private JTextField text_endDate = new JTextField("");
65
66      // les boutons du l'employe
67      private JButton addButton_employe = new JButton("Ajouter");
68      private JButton updateButton_employe = new JButton("Modifier");
69      private JButton deleteButton_employe = new JButton("Supprimer");
70      private JButton displayButton_employe = new JButton("Afficher");
71
72      // les boutons du conge
73      private JButton addButton_holiday = new JButton("Ajouter");
74      private JButton updateButton_holiday = new JButton("Modifier");
75      private JButton deleteButton_holiday = new JButton("Supprimer");
76      private JButton displayButton_holiday = new JButton("Afficher");
77
78
79      // le tableau de l'employe
80      JPanel pan0 = new JPanel(new BorderLayout());
81      public static String[] columnNames_employe = {"ID", "Nom",
            "Prenom", "Email", "Telephone", "Salaire", "Role",
            "Poste","Solde"};
82      public static DefaultTableModel tableModel = new
            DefaultTableModel(columnNames_employe, 0);
83      public static JTable Tableau = new JTable(tableModel);
84
85      // le tableau du conge
86      JPanel pan1 = new JPanel(new BorderLayout());
87      public static String[] columnNames_holiday = {"ID",
```

```
        "nom_employe","date_debut","date_fin","type"};
    public static DefaultTableModel tableModel1 = new
        DefaultTableModel(columnNames_holiday, 0);
    public static JTable Tableau1 = new JTable(tableModel1);

    public MainView() {

        setTitle("Gestion des employes et des conges");
        setSize(1000, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        add(tabbedPane);

    // Employe Tab
        employeTab.setLayout(new BorderLayout());
        employeTab.add(Employepan, BorderLayout.CENTER);

        Employepan.setLayout(new BorderLayout());
        Employepan.add(Display_Table_employe, BorderLayout.CENTER);
        Tableau.setFillsViewportHeight(true);
        Dimension preferredSize = new Dimension(900, 500);
        Tableau.setPreferredScrollableViewportSize(preferredSize);
        pan0.add(new JScrollPane(Tableau), BorderLayout.CENTER);
        Display_Table_employe.add(pan0);

        Employepan.add(panButton_employe, BorderLayout.SOUTH);
        panButton_employe.add(addButton_employe);
        panButton_employe.add(updateButton_employe);
        panButton_employe.add(deleteButton_employe);
        panButton_employe.add(displayButton_employe);

        Employepan.add(Forme_employe, BorderLayout.NORTH);
        Forme_employe.setLayout(new GridLayout(7, 2, 10, 10));
        Forme_employe.add(label_nom);
        Forme_employe.add(text_nom);
        Forme_employe.add(label_prenom);
        Forme_employe.add(text_prenom);
        Forme_employe.add(label_email);
        Forme_employe.add(text_email);
        Forme_employe.add(label_tele);
        Forme_employe.add(text_tele);
        Forme_employe.add(label_salaire);
        Forme_employe.add(text_salaire);
        Forme_employe.add(label_role);
        Forme_employe.add(roleComboBox);
        Forme_employe.add(label_poste);
        Forme_employe.add(posteComboBox);

    // Holiday Tab
        holidayTab.setLayout(new BorderLayout());
        holidayTab.add(Holidaypan, BorderLayout.CENTER);
        Holidaypan.setLayout(new BorderLayout());
```

```
        Holidaypan.add(Display_Table_holiday, BorderLayout.CENTER);

        Tableau1.setFillsViewportHeight(true);
        Tableau1.setPreferredScrollableViewportSize(preferredSize);
        pan1.add(new JScrollPane(Tableau1), BorderLayout.CENTER);
        Display_Table_holiday.add(pan1);

        Holidaypan.add(Forme_holiday, BorderLayout.NORTH);
        Forme_holiday.setLayout(new GridLayout(4, 2, 10, 10));
        Forme_holiday.add(label_employe);
        Forme_holiday.add(text_employe);
        Forme_holiday.add(label_startDate);
        Forme_holiday.add(text_startDate);
        Forme_holiday.add(label_endDate);
        Forme_holiday.add(text_endDate);
        Forme_holiday.add(label_type);
        Forme_holiday.add(TypeComboBox);

        Holidaypan.add(panButton_holiday, BorderLayout.SOUTH);
        panButton_holiday.add(addButton_holiday);
        panButton_holiday.add(updateButton_holiday);
        panButton_holiday.add(deleteButton_holiday);
        panButton_holiday.add(displayButton_holiday);

    // TabbedPane
        tabbedPane.addTab("Employe", employeTab);
        tabbedPane.addTab("Holiday", holidayTab);

        remplaire_les_employes();
        setVisible(true);
    }

    public void remplaire_les_employes () {
        List<Employe> Employes = new EmployeModel(new
           EmployeDAOImpl()).displayEmploye();
       text_employe.removeAllItems();
       for (Employe elem : Employes) {
           text_employe.addItem(elem.getId() + " - " + elem.getNom()+"
              "+elem.getPrenom());
       }
    }

    // getters

        public int getId_employe() {
            return Integer.parseInt(text_employe.getSelectedItem().
            toString().split(" - ")[0]);
        }
        public String getNom() {
            return text_nom.getText();
        }

        public JTable getTable() {
```

```
190             return (JTable) Display_Table_employe.getComponent(0);
191         }
192 
193         public String getPrenom() {
194             return text_prenom.getText();
195         }
196 
197         public String getEmail() {
198             return text_email.getText();
199         }
200 
201         public String getTelephone() {
202             return text_tele.getText();
203         }
204 
205         public double getSalaire() {
206             return Double.parseDouble(text_salaire.getText());
207         }
208 
209         public Role getRole() {
210             return (Role) roleComboBox.getSelectedItem();
211         }
212 
213         public Poste getPoste() {
214             return (Poste) posteComboBox.getSelectedItem();
215         }
216 
217         public JButton getaddButton_employe () {
218             return addButton_employe;
219         }
220 
221         public JButton getupdateButton_employe () {
222             return updateButton_employe;
223         }
224 
225         public JButton getdeleteButton_employe () {
226             return deleteButton_employe;
227         }
228 
229         public JButton getdisplayButton_employe () {
230             return displayButton_employe;
231         }
232 
233         public JButton getaddButton_holiday () {
234             return addButton_holiday;
235         }
236 
237         public JButton getupdateButton_holiday () {
238             return updateButton_holiday;
239         }
240 
241         public JButton getdeleteButton_holiday () {
242             return deleteButton_holiday;
```

```
243         }
244
245         public JButton getdisplayButton_holiday () {
246             return displayButton_holiday;
247         }
248
249     }
```

4.8 Classe `HolidayController`

- **Objectif** : La classe `HolidayController` gère les interactions entre la vue (`MainView`), le modèle (`HolidayModel`) et la logique métier liée à la gestion des congés. Elle permet d'ajouter, de supprimer, de mettre à jour et d'afficher les congés d'un employé, tout en vérifiant les conditions comme le solde de congé et le chevauchement des dates.
- **Détails** :
  - `addHoliday()` : Ajoute un congé pour un employé en vérifiant les dates, le solde de congé et le chevauchement avec d'autres congés.
  - `deleteHoliday()` : Supprime un congé sélectionné en réajustant le solde de congé de l'employé concerné.
  - `displayHoliday()` : Affiche tous les congés existants dans un tableau, avec des informations sur les employés associés.
  - `updateHolidaybyselect()` : Permet de préremplir les champs de modification lorsque l'utilisateur sélectionne un congé à mettre à jour.
  - `updateHoliday()` : Met à jour un congé en vérifiant les nouvelles données et en réajustant le solde de congé.
- **Code Source** :

Classe `HolidayController` :

Listing 7: Code de la classe `HolidayController`

```
1  package Controller;
2
3  import DAO.EmployeDAOImpl;
4  import Model.*;
5  import View.*;
6  import java.sql.Date;
7  import java.util.List;
8  import javax.swing.table.DefaultTableModel;
9
10 public class HolidayController {
11
12     private final MainView View;
13     public HolidayModel model_holiday;
14     public static int id = 0;
15     public static int oldselectedrow = -1;
16     public static boolean test = false;
17     int id_employe = 0;
18     String nom_employe = "";
```

```
19      public static String OldstartDate = null;
20      public static String OldendDate = null;
21      HolidayType type = null;
22      int oldsolde = 0;
23      int solde = 0;
24      boolean updatereussi = false;
25      Employe targetEmploye = null;
26
27      // Constructeur de HolidayController qui initialise les listeners
           pour les boutons
28      public HolidayController(MainView view, HolidayModel model) {
29          this.View = view;
30          this.model_holiday = model;
31
32          // Ajout des listeners pour les boutons
33          View.getdeleteButton_holiday().addActionListener(e ->
               deleteHoliday());
34          View.getupdateButton_holiday().addActionListener(e ->
               updateHoliday());
35          MainView.Tableau1.getSelectionModel().addListSelectionListener(e
               -> updateHolidaybyselect());
36          View.getaddButton_holiday().addActionListener(e ->
               addHoliday());
37          View.getdisplayButton_holiday().addActionListener(e ->
               displayHoliday());
38      }
39
40      // M thode pour ajouter un cong
41      private void addHoliday() {
42          // R cup ration des donn es saisies dans la vue
43          int id_employe = View.getId_employe();
44          Date startDate = Date.valueOf(View.getStartDate());
45          Date endDate = Date.valueOf(View.getEndDate());
46          HolidayType type = View.getHolidayType();
47
48          View.viderChamps_ho(); // Vide les champs de la vue
49
50          Employe targetEmploye = null;
51
52          // Recherche de l'employ  correspondant
53          for (Employe employe : new EmployeModel(new
               EmployeDAOImpl()).displayEmploye()) {
54              if (employe.getId() == id_employe) {
55                  targetEmploye = employe;
56                  break;
57              }
58          }
59
60          if (targetEmploye == null) {
61              View.afficherMessageErreur("Cet employe n'existe pas.");
62              return;
63          }
64
```

```
65          // Calcul de la dur e du cong  demand
66          long daysBetween = java.time.temporal.ChronoUnit.DAYS.between(
67              startDate.toLocalDate(),
68              endDate.toLocalDate()
69          );
70
71          // V rification des dates
72          if (daysBetween <= 0) {
73              View.afficherMessageErreur("Les dates de d but et de fin
                    sont invalides.");
74              return;
75          }
76
77          // V rification du solde de cong  de l'employ
78          if (targetEmploye.getSolde() < daysBetween) {
79              View.afficherMessageErreur("Le solde de cong  de
                    l'employ  est insuffisant.");
80              return;
81          }
82
83          // V rification du chevauchement avec d'autres cong s
84          for (Holiday existingHoliday : model_holiday.displayHoliday())
                {
85              if (existingHoliday.getId_employe() == id_employe) {
86                  Date existingStartDate =
                        existingHoliday.getStartDate();
87                  Date existingEndDate = existingHoliday.getEndDate();
88
89                  // V rification si les dates se chevauchent
90                  if ((startDate.before(existingEndDate) &&
                        endDate.after(existingStartDate))) {
91                      View.afficherMessageErreur("Le cong  se chevauche
                            avec une autre p riode de cong .");
92                      return;
93                  }
94              }
95          }
96
97          // Tentative d'ajout du cong
98          try {
99              boolean addReussi = model_holiday.addHoliday(0,
                    id_employe, startDate, endDate, type, targetEmploye);
100
101             if (addReussi) {
102                 // Mise    jour du solde de cong  apr s l'ajout
103                 targetEmploye.setSolde(targetEmploye.getSolde() -
                        (int) daysBetween);
104                 View.afficherMessageSucces("Holiday est ajout e.");
105             }
106         } catch (Exception e) {
107             e.printStackTrace();
108             View.afficherMessageErreur("Erreur lors de l'ajout : " +
                    e.getMessage());
```

```
109         }
110     }
111
112     // M thode pour afficher les cong s
113     private void displayHoliday() {
114         List<Holiday> Holidays = model_holiday.displayHoliday();
115
116         // V rification si la liste est vide
117         if (Holidays == null || Holidays.isEmpty()) {
118             View.afficherMessageErreur("Aucune holiday.");
119             return; // Retourner pour ne pas continuer l'ex cution si
                    la liste est vide
120         }
121
122         DefaultTableModel tableModel1 = (DefaultTableModel)
                MainView.Tableau1.getModel();
123         tableModel1.setRowCount(0); // Vide le tableau existant
124
125         // Ajout des cong s dans le tableau
126         for (Holiday e : Holidays) {
127             String nom_employe = null;
128             List<Employe> Employes = new EmployeModel(new
                    EmployeDAOImpl()).displayEmploye();
129             for (Employe em : Employes) {
130                 if (em.getId() == e.getId_employe()) {
131                     nom_employe = em.getId() + " - " + em.getNom() + "
                            " + em.getPrenom();
132                     break;
133                 }
134             }
135             // Ajout de la ligne dans le tableau
136             tableModel1.addRow(new Object[]{e.getId_holiday(),
                    nom_employe, e.getStartDate(), e.getEndDate(),
                    e.getType()});
137         }
138         View.remplaire_les_employes();
139     }
140
141     // M thode pour supprimer un cong
142     private void deleteHoliday(){
143         int selectedrow = MainView.Tableau1.getSelectedRow();
144         if(selectedrow == -1){
145             View.afficherMessageErreur("Veuillez selectionner une
                    ligne.");
146         }else{
147             // R cup ration de l'ID et de l'employ
148             int id = (int) MainView.Tableau1.getValueAt(selectedrow,
                    0);
149             int id_employe =
                    Integer.parseInt((MainView.Tableau1.getValueAt(selectedrow,
                    1)).toString().split(" - ")[0]);
150             int olddaysbetween =  (int) (
                    (Date.valueOf(OldendDate).toLocalDate().toEpochDay() -
```

```
                Date.valueOf(OldstartDate).toLocalDate().toEpochDay()));
151
152             // Mise    jour du solde de cong  de l'employ  apr s
                    suppression
153             for(Employe e : new EmployeModel(new
                    EmployeDAOImpl()).displayEmploye()){
154                 if(e.getId() == id_employe){
155                     solde = e.getSolde();
156                     break;
157                 }
158             }
159             EmployeController.updateSolde(id_employe, solde +
                    olddaysbetween);
160
161             // Tentative de suppression du cong
162             boolean deletereussi = model_holiday.deleteHoliday(id);
163             if(deletereussi){
164                 View.afficherMessageSucces("Holiday est supprim e.");
165                 displayHoliday();
166             }else{
167                 View.afficherMessageErreur("Holiday n'est pas
                        supprim e.");
168             }
169         }
170     }
171
172     // M thode pour mettre    jour un cong  s lectionn
173     private void updateHolidaybyselect(){
174         int selectedrow = MainView.Tableau1.getSelectedRow();
175
176         if (selectedrow == -1) {
177             return;
178         }
179         try{
180             // R cup ration des donn es du cong  s lectionn
181             id = (int) MainView.Tableau1.getValueAt(selectedrow, 0);
182             nom_employe = (String)
                    MainView.Tableau1.getValueAt(selectedrow, 1);
183             id_employe = Integer.parseInt(nom_employe.split(" - ")[0]);
184             OldstartDate =
                    String.valueOf(MainView.Tableau1.getValueAt(selectedrow,
                    2));
185             OldendDate =
                    String.valueOf(MainView.Tableau1.getValueAt(selectedrow,
                    3));
186             type = (HolidayType)
                    MainView.Tableau1.getValueAt(selectedrow, 4);
187
188             // Remplissage des champs pour l'  dition
189             View.remplaireChamps_ho(id_employe, OldstartDate,
                    OldendDate, type);
190             test = true;
191         } catch(Exception e){
```

```
192             View.afficherMessageErreur("Erreur lors de la
                    r cup ration des donn es");
193         }
194     }
195
196     // M thode pour mettre    jour un cong
197     private void updateHoliday(){
198         if (!test) {
199             View.afficherMessageErreur("Veuillez d'abord s lectionner
                    une ligne    modifier.");
200             return;
201         }
202         try {
203             // R cup ration des nouvelles donn es pour le cong
204             nom_employe = View.getNom();
205             Date startDate_holiday = Date.valueOf(View.getStartDate());
206             Date endDate_holiday = Date.valueOf(View.getEndDate());
207             type = View.getHolidayType();
208             id_employe = View.getId_employe();
209
210             // Calcul de la nouvelle dur e du cong
211             long daysBetween =
                    java.time.temporal.ChronoUnit.DAYS.between(
212                         startDate_holiday.toLocalDate(),
                                endDate_holiday.toLocalDate());
213
214             // V rification du solde et des dates
215             if (daysBetween <= 0 || daysBetween > oldsolde) {
216                 View.afficherMessageErreur("Date invalide.");
217                 return;
218             }
219
220             // Tentative de mise    jour du cong
221             updatereussi = model_holiday.updateHoliday(id, id_employe,
                    startDate_holiday, endDate_holiday, type);
222
223             if (updatereussi) {
224                 View.afficherMessageSucces("Holiday a   t   mise
                        jour avec succ s.");
225                 displayHoliday();
226             } else {
227                 View.afficherMessageErreur(" chec  de la mise
                        jour.");
228             }
229
230         } catch (Exception e) {
231             View.afficherMessageErreur("Erreur lors de la mise a
                    jour.");
232         }
233     }
234 }
```

4.9 Classe `Main`

- **Objectif** : La classe `Main` est le point d'entrée de l'application. Elle initialise les composants de la vue (`MainView`), les objets de persistance (DAO), les modèles de données (`EmployeModel`, `HolidayModel`), et les contrôleurs (`EmployeController`, `HolidayController`).
- **Détails** :
  - `MainView` : La vue principale de l'application qui est responsable de l'affichage des informations.
  - `EmployeDAOImpl` et `HolidayDAOImpl` : Les implémentations des Data Access Objects (DAO) pour gérer les employés et les congés.
  - `EmployeModel` et `HolidayModel` : Les modèles qui interagissent avec les DAO pour manipuler les données des employés et des congés.
  - `EmployeController` et `HolidayController` : Les contrôleurs qui gèrent la logique métier de l'application et les interactions avec la vue.
- **Code Source** :

Classe `Main` :

Listing 8: Code de la classe `Main`

```
package Main;

import Controller.*;
import DAO.*;
import Model.*;
import View.*;

public class Main {

    public static void main(String[] args) {
        // Initialisation de la vue principale
        MainView view = new MainView();

        // Initialisation des DAO pour interagir avec la base de
           donnees
        EmployeDAOImpl dao = new EmployeDAOImpl();
        HolidayDAOImpl dao_holiday = new HolidayDAOImpl();

        // Initialisation des modeles en passant les DAO correspondants
        EmployeModel model_employe = new EmployeModel(dao);
        HolidayModel model_holiday = new HolidayModel(dao_holiday);

        // Initialisation des controleurs et liaison avec la vue et
           les modeles
        new EmployeController(view, model_employe);
        new HolidayController(view, model_holiday);
    }
}
```

5 Fonctionnement de l'application

Cette section présente les interfaces graphiques du projet et leurs fonctionnalités principales : ajout, affichage, suppression, et modification.

5.1 Ajout d'une demande de congé

La capture d'écran ci-dessous illustre l'interface permettant de créer une nouvelle demande de congé. L'utilisateur peut remplir les champs requis (nom de l'employé, dates de début et de fin, type de congé, etc.) et cliquer sur le bouton `Ajouter` pour enregistrer la demande dans le système.

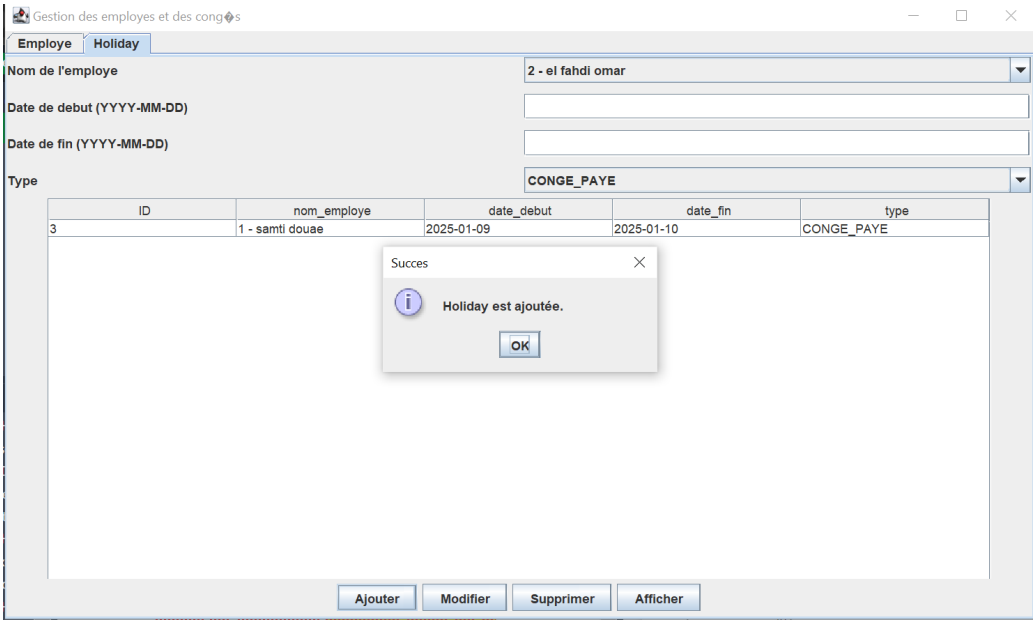

Figure 1: Interface pour l'ajout d'une demande de congé.

5.2 Affichage des demandes de congé

L'interface graphique pour l'affichage permet de visualiser toutes les demandes de congé enregistrées dans le système. Elle comprend une table listant les informations importantes telles que le nom de l'employé, les dates, le type de congé, et le statut de la demande.

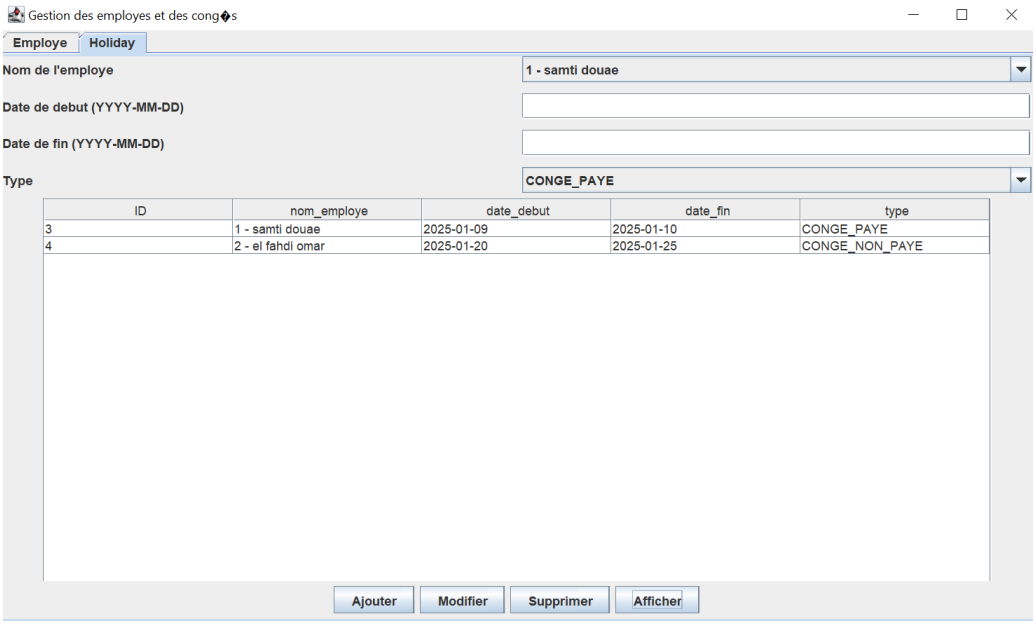

Figure 2: Interface pour l'affichage des demandes de congé.

5.3 Suppression d'une demande de congé

L'option de suppression permet de sélectionner une demande de congé dans la liste et de la retirer du système. Une confirmation peut être requise avant de supprimer la demande.

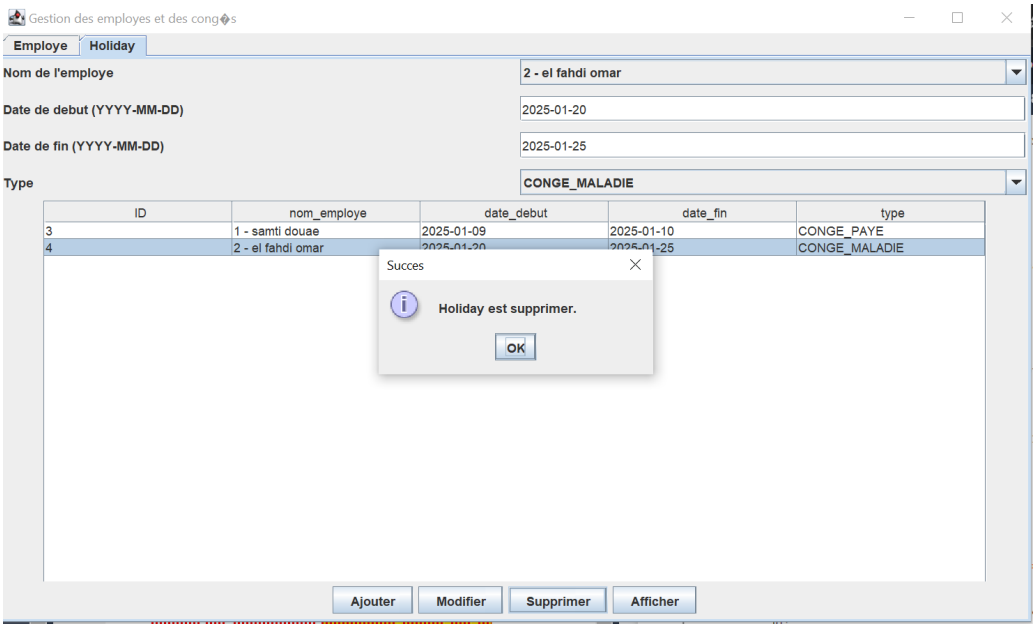

Figure 3: Interface pour la suppression d'une demande de congé.

5.4 Modification d'une demande de congé

Cette fonctionnalité permet de mettre à jour les informations relatives à une demande de congé existante. L'utilisateur sélectionne une demande, modifie les champs nécessaires (dates, type de congé, etc.), puis enregistre les modifications.

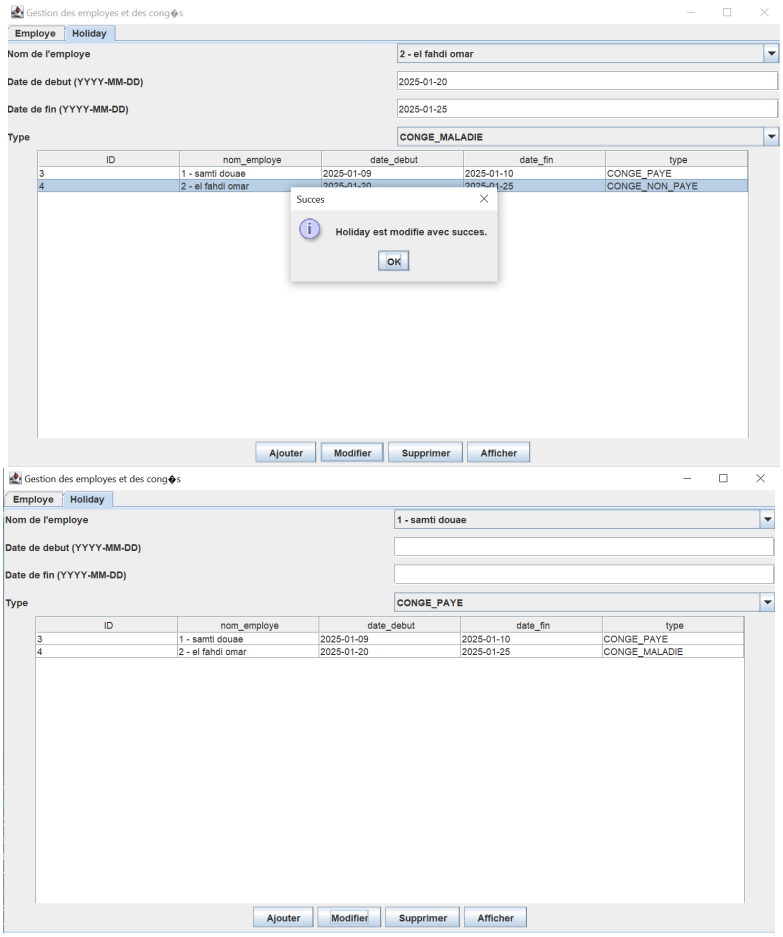

Figure 4: Interface pour la modification d'une demande de congé.

5.5 Modification et vérification des dates de congé

Cette fonctionnalité permet de gérer la modification des informations relatives à une demande de congé existante tout en s'assurant qu'aucune redondance de dates ne survienne. Lorsqu'un utilisateur sélectionne une demande de congé, il peut modifier les champs nécessaires (dates, type de congé, etc.), mais le système vérifie que les nouvelles dates ne chevauchent pas une période de congé existante pour le même employé.

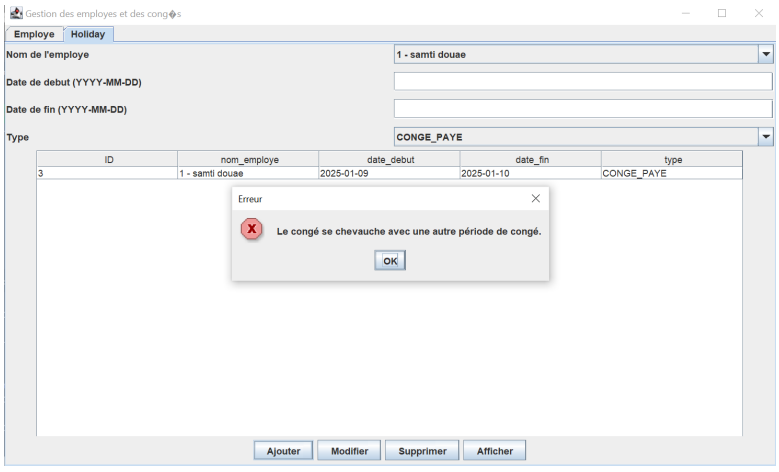

Figure 5: Interface pour la modification d'une demande de congé.

Avantages de cette vérification :

- Garantit l'intégrité des données en empêchant les redondances de congés pour un employé donné.
- Facilite la planification des congés et assure une meilleure organisation.

Grâce à cette amélioration, le système de gestion des congés devient plus robuste et minimise les erreurs liées à des périodes de congés conflictuelles.

5.6 Modification, vérification des dates et gestion du solde

Cette fonctionnalité permet de gérer la modification des informations relatives à une demande de congé existante tout en s'assurant qu'aucune redondance de dates ne survienne. Lorsqu'un utilisateur sélectionne une demande de congé, il peut modifier les champs nécessaires (dates, type de congé, etc.), mais le système vérifie que les nouvelles dates ne chevauchent pas une période de congé existante pour le même employé. De plus, le solde de congé de l'employé est automatiquement ajusté lors de la validation, garantissant un suivi précis des jours restants.

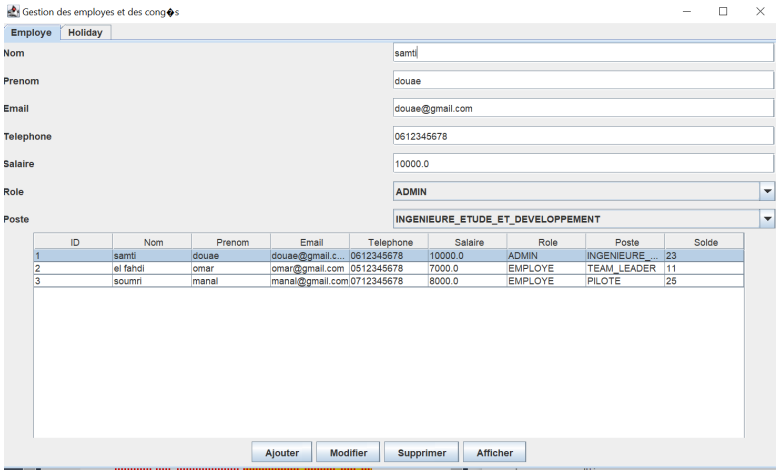

Figure 6: Interface pour la modification d'une demande de congé.

6 Conclusion

En conclusion, ce projet a permis de concevoir et de développer une application performante et bien structurée pour la gestion des congés, en s'appuyant sur une architecture solide combinant les modèles DAO et MVC. La couche DAO a assuré une interaction optimale avec la base de données, offrant un accès centralisé, fiable et sécurisé aux informations essentielles, tandis que la couche MVC a permis une séparation claire des responsabilités, favorisant une meilleure organisation du code.

Le modèle a permis au modèle de gérer la logique métier et les données liées aux congés, à la vue de fournir une interface utilisateur intuitive, et au contrôleur de coordonner efficacement les échanges entre ces deux composants. Cette modularité a renforcé la flexibilité et l'évolutivité de l'application, rendant son architecture prête à accueillir des améliorations futures, qu'il s'agisse de nouvelles fonctionnalités ou de mises à jour esthétiques.

Ce projet constitue donc une base robuste pour répondre aux besoins actuels de gestion des congés tout en ouvrant des perspectives d'évolution et d'optimisation dans le temps.