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 :** AMAL ELLAOUI

**Encadrée par :** Mme ILHAM KACHBAL

# Contents

# Introduction

Dans le cadre de ce projet, nous avons développé une application Java destinée à la gestion des congés. Cette application repose sur l'architecture DAO (Data Access Object) et le modèle MVC (Model-View-Controller), assurant une séparation claire des responsabilités. L'objectif principal était d'offrir une solution modulaire, maintenable et extensible pour la gestion des congés au sein d'une organisation.

# Architecture et Étapes Suivies

## Architecture du Projet

Le projet suit l'architecture MVC, divisant la logique du programme en trois couches distinctes :

1. **Modèle (Model)** : Cette couche contient la logique métier ainsi que les classes représentant les entités, notamment `Holiday` (pour les congés) et `Type` (pour les types de congés).
2. **Vue (View)** : Elle gère l'interface utilisateur et l'affichage des données, notamment à l'aide de composants Swing tels que `JTable` et `JComboBox`.
3. **Contrôleur (Controller)** : Cette couche relie le modèle à la vue et coordonne les interactions utilisateur. Par exemple, la classe `HolidayController` gère les événements liés aux boutons (ajouter, modifier, supprimer) et assure la synchronisation des données entre la vue et le modèle.

## Étapes Suivies

Pour mener à bien le développement de l'application, nous avons suivi les étapes suivantes :

1. **Conception de la base de données** :
   - Création d'une table `Holiday` avec des colonnes pour les informations relatives aux congés (ID, nom de l'employé, date de début, date de fin, type).
2. **Développement du module DAO** :
   - Création d'une interface générique `GenericDAO` définissant les opérations CRUD de base.
   - Implémentation de cette interface dans `HolidayDAOImpl`, en utilisant des requêtes SQL préparées pour interagir avec la base de données.
3. **Développement des classes du modèle** :
   - Création de la classe `Holiday`, qui représente un congé, avec des attributs comme `id`, `employeeName`, `startDate`, `endDate` et `type`.
   - Définition de l'énumération `Type` pour encapsuler les différents types de congés (par exemple, `ANNUEL`, `MALADIE`).
4. **Développement de la vue** :
   - Conception d'une interface utilisateur intuitive utilisant Swing pour permettre la gestion des congés via des formulaires et des tableaux interactifs.

5. **Développement du contrôleur** :

   - Mise en œuvre de la logique permettant de gérer les actions utilisateur (ajout, suppression, modification) et de synchroniser les données entre la base et l'affichage.

# Explications des Codes

## Modèle (`Holiday`, `Type`)

- **Objectif** : Le modèle représente les données manipulées par l'application.
- **Détails** :
  - `Holiday` : Classe encapsulant les informations d'un congé, avec des attributs comme `id`, `employeeName`, `startDate`, `endDate`, et `type`. Elle contient également des `getters` et `setters` pour accéder aux propriétés.
  - `Type` : Énumération définissant les types de congés possibles, tels que `ANNUEL`, `MALADIE`, etc., permettant d'assurer la cohérence des données.

## Classe `Holiday`

Listing 1: Classe Holiday

```
1
2  package Model;
3
4  public class Holiday {
5      private int id; // Identifiant du cong
6      private int employeeId; // ID de l'employ
7      private String employeeName; // Nom complet de l'employ
8      private String startDate; // Date de d but
9      private String endDate;   // Date de fin
10     private Type type;        // Type de cong  (enum)
11
12     // Constructeur avec employeeName pour listAll()
13     public Holiday(int id, String employeeName, String startDate,
           String endDate, Type type) {
14         this.id = id;
15         this.employeeName = employeeName;
16         this.startDate = startDate;
17         this.endDate = endDate;
18         this.type = type;
19     }
20     public Holiday(String employeeName, String startDate, String
           endDate, Type type) {
21         this.employeeName = employeeName;
22         this.startDate = startDate;
23         this.endDate = endDate;
24         this.type = type;
25     }
26
27     // Constructeur pour add() et update() (sans employeeName)
```

```
28      public Holiday(int employeeId, String startDate, String endDate,
            Type type) {
29          this.employeeId = employeeId;
30          this.startDate = startDate;
31          this.endDate = endDate;
32          this.type = type;
33      }
34
35      // Getters et Setters
36      public int getId() {
37          return id;
38      }
39
40      public int getEmployeeId() {
41          return employeeId;
42      }
43
44      public String getEmployeeName() {
45          return employeeName; // Retourne le nom complet
46      }
47
48      public String getStartDate() {
49          return startDate;
50      }
51
52      public String getEndDate() {
53          return endDate;
54      }
55
56      public Type getType() {
57          return type;
58      }
59
60      public void setEmployeeName(String employeeName) {
61          this.employeeName = employeeName;
62      }
63  }
```

## Modèle (`Holiday`, `Type`)

- **Objectif** : Représenter les données relatives aux congés dans l'application.
- **Détails** :
  - `Holiday` contient des attributs comme `employeeId`, `startDate`, `endDate`, etc., avec des `getters` et `setters`.
  - `Type` est une énumération définissant les types de congés (maladie, payé, non payé).

## Code Source

**Enumération** `Type`

Listing 2: Enumération Type

```java
package Model;

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

## Connexion à la base de données (DBConnection)

- **Objectif** : Gérer la connexion à la base de données MySQL pour les données relatives aux congés.
- **Détails** :
  - Utilise le driver JDBC pour se connecter à la base de données `conge`.
  - Gère les exceptions SQL en affichant des messages d'erreur pertinents.

## Code Source

Listing 3: Connexion à la base de données

**Classe** `DBConnection`

```java
package DAO;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/
       conge";
    private static final String USER = "root";
    private static final String PASSWORD = "";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

## DAO (HolidayDAOImpl)

- **Objectif** : Effectuer les opérations CRUD sur les congés des employés.
- **Fonctionnalités principales** :
  - `add()` : Ajoute un congé pour un employé dans la base de données.
  - `delete()` : Supprime un congé en fonction de son ID.
  - `listAll()` : Récupère tous les congés sous forme de liste.
  - `findById()` : Récupère un congé en fonction de son ID.
  - `update()` : Met à jour un congé existant.

## Code Source

Listing 4: DAO HolidayDAOImpl

**Classe** `HolidayDAOImpl`

```
package DAO;

import Model.Holiday;
import Model.Type;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class HolidayDAOImpl implements GenericDAO<Holiday> {

    // Constants for SQL queries
    private static final String INSERT_HOLIDAY_SQL = "INSERT INTO holiday (employeeId, startDate, endDate, type) VALUES (?, ?, ?, ?)";
    private static final String DELETE_HOLIDAY_SQL = "DELETE FROM holiday WHERE id = ?";
    private static final String SELECT_ALL_HOLIDAY_SQL = "SELECT h.id, CONCAT(e.nom, ' ', e.prenom) AS employeeName, h.startDate, h.endDate, h.type FROM holiday h JOIN employe e ON h.employeeId = e.id";
    private static final String SELECT_HOLIDAY_BY_ID_SQL = "SELECT h.id, CONCAT(e.nom, ' ', e.prenom) AS employeeName, h.startDate, h.endDate, h.type FROM holiday h JOIN employe e ON h.employeeId = e.id WHERE h.id = ?";
    private static final String SELECT_EMPLOYEE_ID_BY_NAME_SQL = "SELECT id FROM employe WHERE CONCAT(nom, ' ', prenom) = ?";

    // M thode pour ajouter un cong
    @Override
    public void add(Holiday holiday) {
        try (Connection conn = DBConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(
             INSERT_HOLIDAY_SQL)) {
             int employeeId = getEmployeeIdByName(holiday.
                getEmployeeName());
             if (employeeId == -1) {
                 System.out.println("Erreur : Employ introuvable.");
                 return;
             }
             stmt.setInt(1, employeeId);
             stmt.setString(2, holiday.getStartDate());
             stmt.setString(3, holiday.getEndDate());
             stmt.setString(4, holiday.getType().name());
             stmt.executeUpdate();
             System.out.println("Cong ajout avec succ s.");
         } catch (SQLException e) {
             System.err.println("Erreur lors de l'ajout du cong :" +
                e.getMessage());
             e.printStackTrace();
```

```
        }
    }

    // M thode pour supprimer un cong  par ID
    @Override
    public void delete(int id) {
        try (Connection conn = DBConnection.getConnection();
           PreparedStatement stmt = conn.prepareStatement(
           DELETE_HOLIDAY_SQL)) {
            stmt.setInt(1, id);
            int rowsDeleted = stmt.executeUpdate();
            if (rowsDeleted > 0) {
                System.out.println("Cong  supprim  avec succ s.");
            } else {
                System.out.println("Aucun cong  trouv  avec cet ID.")
                   ;
            }
        } catch (SQLException e) {
            System.err.println("Erreur lors de la suppression du cong
               : " + e.getMessage());
            e.printStackTrace();
        }
    }

    // M thode pour lister tous les cong s
    @Override
    public List<Holiday> listAll() {
        List<Holiday> holidays = new ArrayList<>();
        try (Connection conn = DBConnection.getConnection();
           PreparedStatement stmt = conn.prepareStatement(
           SELECT_ALL_HOLIDAY_SQL); ResultSet rs = stmt.executeQuery())
            {
            while (rs.next()) {
                Holiday holiday = new Holiday(
                        rs.getInt("id"),
                        rs.getString("employeeName"),
                        rs.getString("startDate"),
                        rs.getString("endDate"),
                        Type.valueOf(rs.getString("type"))
                );
                holidays.add(holiday);
            }
        } catch (SQLException e) {
            System.err.println("Erreur lors de la r cup ration des
               cong s : " + e.getMessage());
            e.printStackTrace();
        }
        return holidays;
    }

    // M thode pour trouver un cong  par ID
    @Override
    public Holiday findById(int id) {
```

```
        try (Connection conn = DBConnection.getConnection();
           PreparedStatement stmt = conn.prepareStatement(
           SELECT_HOLIDAY_BY_ID_SQL)) {
            stmt.setInt(1, id);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                return new Holiday(
                        rs.getInt("id"),
                        rs.getString("employeeName"),
                        rs.getString("startDate"),
                        rs.getString("endDate"),
                        Type.valueOf(rs.getString("type"))
                );
            }
        } catch (SQLException e) {
            System.err.println("Erreur lors de la recherche du cong :
               " + e.getMessage());
            e.printStackTrace();
        }
        return null;
    }

    // M thode pour mettre    jour un cong
    @Override
    public void update(Holiday holiday, int id) {
        try (Connection conn = DBConnection.getConnection();
           PreparedStatement stmt = conn.prepareStatement("UPDATE
           holiday SET employeeId = ?, startDate = ?, endDate = ?, type
           = ? WHERE id = ?")) {
            int employeeId = getEmployeeIdByName(holiday.
               getEmployeeName());
            if (employeeId == -1) {
                System.out.println("Erreur : Employ introuvable.");
                return;
            }
            stmt.setInt(1, employeeId);
            stmt.setString(2, holiday.getStartDate());
            stmt.setString(3, holiday.getEndDate());
            stmt.setString(4, holiday.getType().name());
            stmt.setInt(5, id);
            int rowsUpdated = stmt.executeUpdate();
            if (rowsUpdated > 0) {
                System.out.println("Cong mis  jour avec succ s.");
            } else {
                System.out.println("Aucun cong trouv avec cet ID.")
                   ;
            }
        } catch (SQLException e) {
            System.err.println("Erreur lors de la mise  jour du
               cong : " + e.getMessage());
            e.printStackTrace();
        }
    }
```

```
126
127     // M thode pour r cup rer l'ID de l'employ  par nom complet
128     public int getEmployeeIdByName(String employeeName) {
129         try (Connection conn = DBConnection.getConnection();
                PreparedStatement stmt = conn.prepareStatement(
                SELECT_EMPLOYEE_ID_BY_NAME_SQL)) {
130             stmt.setString(1, employeeName);
131             ResultSet rs = stmt.executeQuery();
132             if (rs.next()) {
133                 return rs.getInt("id");
134             }
135         } catch (SQLException e) {
136             System.err.println("Erreur lors de la r cup ration de l'
                   ID employ  : " + e.getMessage());
137             e.printStackTrace();
138         }
139         return -1;
140     }
141
142     // M thode pour r cup rer tous les noms des employ s
143     public List<String> getAllEmployeeNames() {
144         List<String> employeeNames = new ArrayList<>();
145         String query = "SELECT CONCAT(nom, ' ', prenom) AS fullName
               FROM employe";
146
147         try (Connection conn = DBConnection.getConnection();
148              PreparedStatement stmt = conn.prepareStatement(query);
149              ResultSet rs = stmt.executeQuery()) {
150
151             while (rs.next()) {
152                 String fullName = rs.getString("fullName");
153                 employeeNames.add(fullName);
154             }
155         } catch (SQLException e) {
156             System.err.println("Erreur lors de la r cup ration des
                   noms des employ s : " + e.getMessage());
157         }
158
159         return employeeNames;
160     }
161
162     // M thode pour r cup rer les types de cong s (Enum)
163     public List<Type> getAllHolidayTypes() {
164         List<Type> holidayTypes = new ArrayList<>();
165         for (Type type : Type.values()) {
166             holidayTypes.add(type);
167         }
168         return holidayTypes;
169     }
170 }
```

listings xcolor
frame=single, tabsize=4, showstringspaces=false, breaklines=true, breakatwhitespace=true,

## Vue (HolidayView)

**Objectif** : Fournir une interface utilisateur pour gérer les congés des employés.
**Détails** :

- Contient des champs de saisie pour les détails des congés (nom de l'employé, dates de début et de fin, type de congé).
- Des boutons permettent d'exécuter les différentes actions CRUD.

Code source :

```
package View;

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

public class HolidayView extends JFrame {
    public JTable holidayTable;
    public JButton addButton, listButton, deleteButton, modifyButton,
       switchViewButton;
    public JComboBox<String> employeeNameComboBox;
    public JTextField startDateField, endDateField;
    public JComboBox<String> typeCombo;

    public HolidayView() {
        setTitle("Gestion des Cong s");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Panneau de saisie des champs
        JPanel inputPanel = new JPanel(new GridLayout(0, 2, 10, 10));
        inputPanel.add(new JLabel("Employ  Nom Complet:"));
        employeeNameComboBox = new JComboBox<>();
        inputPanel.add(employeeNameComboBox);

        inputPanel.add(new JLabel("Date D but:"));
        startDateField = new JTextField();
        inputPanel.add(startDateField);

        inputPanel.add(new JLabel("Date Fin:"));
        endDateField = new JTextField();
        inputPanel.add(endDateField);

        inputPanel.add(new JLabel("Type:"));
        typeCombo = new JComboBox<>(new String[]{"CONGE_PAYE", "
           CONGE_MALADIE", "CONGE_NON_PAYE"});
        inputPanel.add(typeCombo);

        add(inputPanel, BorderLayout.NORTH);

        // Table des cong s
        holidayTable = new JTable();
        add(new JScrollPane(holidayTable), BorderLayout.CENTER);
```

```
42
43          // Boutons d'action
44          JPanel buttonPanel = new JPanel();
45          addButton = new JButton("Ajouter");
46          buttonPanel.add(addButton);
47          listButton = new JButton("Afficher");
48          buttonPanel.add(listButton);
49          deleteButton = new JButton("Supprimer");
50          buttonPanel.add(deleteButton);
51          modifyButton = new JButton("Modifier");
52          buttonPanel.add(modifyButton);
53
54          switchViewButton = new JButton("Gerer les Employ s");
55          buttonPanel.add(switchViewButton);
56
57          add(buttonPanel, BorderLayout.SOUTH);
58      }
59  }
```

## Contrôleur (HolidayController)

**Objectif** : Gérer les interactions utilisateur pour la gestion des congés et coordonner les actions entre la vue et le modèle.

**Fonctionnalités principales** :

- `addHoliday()`: Ajoute un congé en récupérant les entrées utilisateur et en appelant `HolidayDAOImpl.add`.
- `deleteHoliday()`: Supprime un congé en fonction de son ID après confirmation.
- `modifyHoliday()`: Modifie les informations d'un congé existant.
- `loadEmployeeNames()`: Charge et affiche les noms des employés dans la vue.
- `refreshHolidayTable()`: Rafraîchit la table des congés affichée dans la vue.

Code source :

```
1   package Controller;
2
3   import DAO.HolidayDAOImpl;
4   import Model.Holiday;
5   import Model.Type;
6   import View.HolidayView;
7
8   import javax.swing.*;
9   import java.time.LocalDate;
10  import java.time.format.DateTimeFormatter;
11  import java.util.List;
12
13  public class HolidayController {
14      private final HolidayView view;
15      private final HolidayDAOImpl dao;
16
```

```
17      public HolidayController(HolidayView view) {
18          this.view = view;
19          this.dao = new HolidayDAOImpl();
20
21          loadEmployeeNames();
22          refreshHolidayTable();
23
24          view.addButton.addActionListener(e -> addHoliday());
25          view.deleteButton.addActionListener(e -> deleteHoliday());
26          view.modifyButton.addActionListener(e -> modifyHoliday());
27
28          // Ajout d'un  couteur  de s lection sur la table
29          view.holidayTable.getSelectionModel().addListSelectionListener(
                e -> {
30              if (!e.getValueIsAdjusting()) { // V rifie si la
                    s lection est termin e
31                  int selectedRow = view.holidayTable.getSelectedRow();
32                  if (selectedRow != -1) {
33                      // R cup rer l'ID et les autres informations de
                            la ligne s lectionn e
34                      int id = (int) view.holidayTable.getValueAt(
                            selectedRow, 0); // R cup re l'ID depuis la
                            colonne 0
35                      String employeeName = (String) view.holidayTable.
                            getValueAt(selectedRow, 1);
36                      String startDate = (String) view.holidayTable.
                            getValueAt(selectedRow, 2);
37                      String endDate = (String) view.holidayTable.
                            getValueAt(selectedRow, 3);
38                      Type type = Type.valueOf(view.holidayTable.
                            getValueAt(selectedRow, 4).toString());
39
40                      // Remplir les champs de modification avec ces
                            valeurs
41                      view.employeeNameComboBox.setSelectedItem(
                            employeeName);
42                      view.startDateField.setText(startDate);
43                      view.endDateField.setText(endDate);
44                      view.typeCombo.setSelectedItem(type.toString());
45
46                      // Modifier l'action du bouton de modification pour
                             utiliser l'ID de la ligne s lectionn e
47                      view.modifyButton.setActionCommand(String.valueOf(
                            id));
48                  }
49              }
50          });
51      }
52
53      private void loadEmployeeNames() {
54          view.employeeNameComboBox.removeAllItems();
55          List<String> names = dao.getAllEmployeeNames();
56
```

```
57          if (names.isEmpty()) {
58              System.out.println("Aucun employ trouv .");
59          } else {
60              System.out.println("Noms des employ s charg s : " + names
                    );
61              for (String name : names) {
62                  view.employeeNameComboBox.addItem(name);
63              }
64          }
65      }
66
67      private void refreshHolidayTable() {
68          List<Holiday> holidays = dao.listAll();
69          String[] columnNames = {"ID", "Employ ", "Date D but", "Date
                Fin", "Type"};
70          Object[][] data = new Object[holidays.size()][5];
71
72          for (int i = 0; i < holidays.size(); i++) {
73              Holiday h = holidays.get(i);
74              data[i] = new Object[]{h.getId(), h.getEmployeeName(), h.
                    getStartDate(), h.getEndDate(), h.getType()};
75          }
76
77          view.holidayTable.setModel(new javax.swing.table.
                DefaultTableModel(data, columnNames));
78      }
79
80      private boolean isValidDate(String date) {
81          try {
82              DateTimeFormatter formatter = DateTimeFormatter.
                    ISO_LOCAL_DATE;
83              LocalDate.parse(date, formatter);
84              return true;
85          } catch (Exception e) {
86              return false;
87          }
88      }
89
90      private boolean isEndDateAfterStartDate(String startDate, String
            endDate) {
91          DateTimeFormatter formatter = DateTimeFormatter.ISO_LOCAL_DATE;
92          LocalDate start = LocalDate.parse(startDate, formatter);
93          LocalDate end = LocalDate.parse(endDate, formatter);
94
95          return end.isAfter(start);
96      }
97
98      private void addHoliday() {
99          try {
100             String employeeName = (String) view.employeeNameComboBox.
                    getSelectedItem();
101             String startDate = view.startDateField.getText();
102             String endDate = view.endDateField.getText();
```

```
103             Type type = Type.valueOf(view.typeCombo.getSelectedItem().
                    toString().toUpperCase());
104
105             if (!isValidDate(startDate) || !isValidDate(endDate)) {
106                 throw new IllegalArgumentException("Les␣dates␣doivent␣
                        tre ␣au␣format␣YYYY-MM-DD.");
107             }
108
109             if (!isEndDateAfterStartDate(startDate, endDate)) {
110                 throw new IllegalArgumentException("La␣date␣de␣fin␣doit
                        ␣ tre ␣sup rieure␣  ␣la␣date␣de␣d but.");
111             }
112
113             Holiday holiday = new Holiday(employeeName, startDate,
                    endDate, type);
114             dao.add(holiday);
115             loadEmployeeNames();
116             refreshHolidayTable();
117             JOptionPane.showMessageDialog(view, "Cong ␣ajout ␣avec␣
                    succ s.");
118         } catch (Exception ex) {
119             JOptionPane.showMessageDialog(view, "Erreur␣:␣" + ex.
                    getMessage());
120         }
121     }
122
123     private void modifyHoliday() {
124         try {
125             // R cup rer l'ID du cong     partir de l'action du
                    bouton
126             String actionCommand = view.modifyButton.getActionCommand()
                    ;
127             if (actionCommand != null && !actionCommand.trim().isEmpty
                    ()) {
128                 int id = Integer.parseInt(actionCommand.trim());
129
130                 String employeeName = (String) view.
                        employeeNameComboBox.getSelectedItem();
131                 String startDate = view.startDateField.getText();
132                 String endDate = view.endDateField.getText();
133                 Type type = Type.valueOf(view.typeCombo.getSelectedItem
                        ().toString().toUpperCase());
134
135                 if (!isValidDate(startDate) || !isValidDate(endDate)) {
136                     throw new IllegalArgumentException("Les␣dates␣
                            doivent␣ tre ␣au␣format␣YYYY-MM-DD.");
137                 }
138
139                 if (!isEndDateAfterStartDate(startDate, endDate)) {
140                     throw new IllegalArgumentException("La␣date␣de␣fin␣
                            doit␣ tre ␣sup rieure␣  ␣la␣date␣de␣d but.");
141                 }
142
```

```
143                 Holiday holiday = new Holiday(employeeName, startDate,
                        endDate, type);
144                 dao.update(holiday, id);
145                 loadEmployeeNames();
146                 refreshHolidayTable();
147                 JOptionPane.showMessageDialog(view, "Cong␣modifi␣
                        avec␣succ s.");
148             } else {
149                 JOptionPane.showMessageDialog(view, "Veuillez␣
                        s lectionner␣un␣cong ␣ ␣modifier.");
150             }
151         } catch (Exception ex) {
152             JOptionPane.showMessageDialog(view, "Erreur␣:␣" + ex.
                    getMessage());
153         }
154     }
155
156     private void deleteHoliday() {
157         try {
158             // Afficher un message pour entrer un ID de cong
159             String input = JOptionPane.showInputDialog(view, "Veuillez␣
                    entrer␣l'ID␣du␣cong ␣ ␣supprimer:");
160             if (input != null && !input.trim().isEmpty()) {
161                 int id = Integer.parseInt(input.trim());
162
163                 // Demander confirmation avant de supprimer
164                 int confirm = JOptionPane.showConfirmDialog(view,
165                         " tes -vous␣s r␣de␣vouloir␣supprimer␣le␣cong
                                ␣avec␣l'ID␣" + id + "␣?",
166                         "Confirmation␣de␣suppression",
167                         JOptionPane.YES_NO_OPTION);
168
169                 if (confirm == JOptionPane.YES_OPTION) {
170                     dao.delete(id); // Supprimer le cong
171                     loadEmployeeNames();
172                     refreshHolidayTable();
173                     JOptionPane.showMessageDialog(view, "Cong ␣
                            supprim ␣avec␣succ s.");
174                 } else {
175                     JOptionPane.showMessageDialog(view, "Suppression␣
                            annul e.");
176                 }
177             } else {
178                 JOptionPane.showMessageDialog(view, "ID␣de␣cong ␣non␣
                        valide␣ou␣action␣annul e.");
179             }
180         } catch (NumberFormatException ex) {
181             JOptionPane.showMessageDialog(view, "ID␣invalide.␣Veuillez␣
                    entrer␣un␣nombre␣valide.");
182         } catch (Exception ex) {
183             JOptionPane.showMessageDialog(view, "Erreur␣:␣" + ex.
                    getMessage());
184         }
```

```
    }

}
```

## Classe principale (Main)

**Objectif** : Démarrer l'application.

**Détails** :

- Crée une instance de `EmployeeView` et la passe à `EmployeeController`.
- Crée une instance de `HolidayView` et la passe à `HolidayController`.
- Révèle l'interface utilisateur avec `setVisible(true)`.
- Permet de basculer entre les vues des employés et des congés à l'aide des boutons de commutation.

Code source :

```
// Classe principale pour d marrer l'application
package Main;

import Controller.EmployeeController;
import Controller.HolidayController;
import View.EmployeeView;
import View.HolidayView;

public class Main {
    public static void main(String[] args) {
        // Cr er les vues
        EmployeeView employeeView = new EmployeeView();
        HolidayView holidayView = new HolidayView();

        // Cr er les contr leurs pour chaque vue
        new EmployeeController(employeeView, holidayView);
        new HolidayController(holidayView);

        // D finir quelle vue sera affich e par d faut (exemple :
           vue des employ s)
        employeeView.setVisible(true);

        // Ajouter un gestionnaire pour passer de l'une    l'autre
        employeeView.switchViewButton.addActionListener(e -> {
            employeeView.setVisible(false);
            holidayView.setVisible(true);
        });

        holidayView.switchViewButton.addActionListener(e -> {
            holidayView.setVisible(false);
            employeeView.setVisible(true);
        });
    }
}
```

# Conclusion

Ce projet fournit une gestion simple mais efficace des congés pour les employés, avec une interface graphique permettant une manipulation facile des données. La séparation des responsabilités entre le modèle, la vue et le contrôleur suit les bonnes pratiques de développement logiciel.