



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 // LEILA ELKHROF

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Architecture et Étapes Suivies</b>	<b>2</b>
2.1	Architecture du Projet . . . . .	2
2.2	Étapes Suivies . . . . .	2
<b>3</b>	<b>Explications des Codes</b>	<b>3</b>
3.1	Modèle (Holiday, Type) . . . . .	3
3.2	Classe Holiday . . . . .	3
3.3	Modèle (Holiday, Type) . . . . .	4
3.4	Code Source . . . . .	4
3.5	Connexion à la base de données (DBConnection) . . . . .	5
3.6	Code Source . . . . .	5
3.7	DAO (HolidayDAOImpl) . . . . .	5
3.8	Code Source . . . . .	6
3.9	Vue (HolidayView) . . . . .	10
3.10	Contrôleur (HolidayController) . . . . .	11
3.11	Classe principale (Main) . . . . .	17
<b>4</b>	<b>Fonctionnement de l'application</b>	<b>18</b>
4.1	Ajout d'une demande de congé . . . . .	18
4.2	Cas de dépassement de solde de congé : . . . . .	19
4.3	Cas de chevauchement de congé : . . . . .	19
4.4	Affichage des demandes de congé . . . . .	20
4.5	Suppression d'une demande de congé . . . . .	21
4.6	Modification d'une demande de congé . . . . .	23
<b>5</b>	<b>Conclusion</b>	<b>23</b>

# 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 congé
6     private int employeeId; // ID de l'employé
7     private String employeeName; // Nom complet de l'employé
8     private String startDate; // Date de début
9     private String endDate; // Date de fin
10    private Type type; // Type de congé (enum)
11
12    // Constructeur avec employeeName pour listAll()
13    public Holiday(int id, String employeeName, String startDate,
14        String endDate, Type type) {
15        this.id = id;
16        this.employeeName = employeeName;
17        this.startDate = startDate;
18        this.endDate = endDate;
19        this.type = type;
20    }
21    public Holiday(String employeeName, String startDate, String
22        endDate, Type type) {
23        this.employeeName = employeeName;
24        this.startDate = startDate;
25        this.endDate = endDate;
26        this.type = type;
27    }
28
29    // Constructeur pour add() et update() (sans employeeName)
```

```

28     public Holiday(int employeeId, String startDate, String endDate,
29                    Type type) {
30         this.employeeId = employeeId;
31         this.startDate = startDate;
32         this.endDate = endDate;
33         this.type = type;
34     }
35
36     // Getters et Setters
37     public int getId() {
38         return id;
39     }
40
41     public int getEmployeeId() {
42         return employeeId;
43     }
44
45     public String getEmployeeName() {
46         return employeeName; // Retourne le nom complet
47     }
48
49     public String getStartDate() {
50         return startDate;
51     }
52
53     public String getEndDate() {
54         return endDate;
55     }
56
57     public Type getType() {
58         return type;
59     }
60
61     public void setEmployeeName(String employeeName) {
62         this.employeeName = employeeName;
63     }
64 }

```

## 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

```

1 package Model;
2
3 public enum Type {
4     CONGE_MALADIE,
5     CONGE_PAYE,
6     CONGE_NON_PAYE
7 }

```

## 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

```

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/
9         conge";
10    private static final String USER = "root";
11    private static final String PASSWORD = "";
12
13    public static Connection getConnection() throws SQLException {
14        return DriverManager.getConnection(URL, USER, PASSWORD);
15    }
16 }

```

## 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

```
1 package DAO;
2
3 import Model.Holiday;
4 import Model.Type;
5
6 import java.sql.*;
7 import java.time.LocalDate;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 public class HolidayDAOImpl implements GenericDAO<Holiday> {
12
13     private static final String INSERT_HOLIDAY_SQL = "INSERT INTO
14         holiday(employeeId, startDate, endDate, type) VALUES(?, ?, ?, ?)";
15
16     private static final String DELETE_HOLIDAY_SQL = "DELETE FROM
17         holiday WHERE id=?";
18
19     private static final String SELECT_ALL_HOLIDAY_SQL = "SELECT h.id,
20         CONCAT(e.nom, ' ', e.prenom) AS employeeName, h.startDate, h.
21         endDate, h.type FROM holiday h JOIN employee e ON h.employeeId=e.
22         id";
23
24     private static final String SELECT_HOLIDAY_BY_ID_SQL = "SELECT h.id
25         , CONCAT(e.nom, ' ', e.prenom) AS employeeName, h.startDate, h.
26         endDate, h.type FROM holiday h JOIN employee e ON h.employeeId=e.
27         id WHERE h.id=?";
28
29     private static final String SELECT_EMPLOYEE_ID_BY_NAME_SQL = "
30         SELECT id FROM employee WHERE CONCAT(nom, ' ', prenom)=?";
31
32     @Override
33     public void add(Holiday holiday) {
34         try (Connection conn = DBConnection.getConnection();
35             PreparedStatement stmt = conn.prepareStatement(
36                 INSERT_HOLIDAY_SQL)) {
37             int employeeId = getEmployeeIdByName(holiday.
38                 getEmployeeName());
39             if (employeeId == -1) return;
40             stmt.setInt(1, employeeId);
41             stmt.setString(2, holiday.getStartDate());
42             stmt.setString(3, holiday.getEndDate());
43             stmt.setString(4, holiday.getType().name());
44             stmt.executeUpdate();
45         } catch (SQLException e) {
46             e.printStackTrace();
47         }
48     }
49
50     @Override
51     public void delete(int id) {
52         try (Connection conn = DBConnection.getConnection();
53             PreparedStatement stmt = conn.prepareStatement(
```

```

        DELETE_HOLIDAY_SQL)) {
37         stmt.setInt(1, id);
38         stmt.executeUpdate();
39     } catch (SQLException e) {
40         e.printStackTrace();
41     }
42 }
43
44 @Override
45 public List<Holiday> listAll() {
46     List<Holiday> holidays = new ArrayList<>();
47     try (Connection conn = DBConnection.getConnection();
48         PreparedStatement stmt = conn.prepareStatement(
49             SELECT_ALL_HOLIDAY_SQL); ResultSet rs = stmt.executeQuery())
50     {
51         while (rs.next()) {
52             holidays.add(new Holiday(
53                 rs.getInt("id"),
54                 rs.getString("employeeName"),
55                 rs.getString("startDate"),
56                 rs.getString("endDate"),
57                 Type.valueOf(rs.getString("type"))
58             ));
59         }
60     } catch (SQLException e) {
61         e.printStackTrace();
62     }
63     return holidays;
64 }
65
66 @Override
67 public Holiday findById(int id) {
68     try (Connection conn = DBConnection.getConnection();
69         PreparedStatement stmt = conn.prepareStatement(
70             SELECT_HOLIDAY_BY_ID_SQL)) {
71         stmt.setInt(1, id);
72         ResultSet rs = stmt.executeQuery();
73         if (rs.next()) {
74             return new Holiday(
75                 rs.getInt("id"),
76                 rs.getString("employeeName"),
77                 rs.getString("startDate"),
78                 rs.getString("endDate"),
79                 Type.valueOf(rs.getString("type"))
80             );
81         }
82     } catch (SQLException e) {
83         e.printStackTrace();
84     }
85     return null;
86 }
87
88 @Override

```



```

84 public void update(Holiday holiday, int id) {
85     try (Connection conn = DBConnection.getConnection();
        PreparedStatement stmt = conn.prepareStatement("UPDATE_
        holiday SET employeeId=?, startDate=?, endDate=?, type
        =? WHERE id=?")) {
86         int employeeId = getEmployeeIdByName(holiday.
            getEmployeeName());
87         if (employeeId == -1) return;
88         stmt.setInt(1, employeeId);
89         stmt.setString(2, holiday.getStartDate());
90         stmt.setString(3, holiday.getEndDate());
91         stmt.setString(4, holiday.getType().name());
92         stmt.setInt(5, id);
93         stmt.executeUpdate();
94     } catch (SQLException e) {
95         e.printStackTrace();
96     }
97 }
98
99 public int getEmployeeIdByName(String employeeName) {
100     try (Connection conn = DBConnection.getConnection();
        PreparedStatement stmt = conn.prepareStatement(
        SELECT_EMPLOYEE_ID_BY_NAME_SQL)) {
101         stmt.setString(1, employeeName);
102         ResultSet rs = stmt.executeQuery();
103         if (rs.next()) return rs.getInt("id");
104     } catch (SQLException e) {
105         e.printStackTrace();
106     }
107     return -1;
108 }
109 public int getTotalDaysTakenThisYear(String employeeName, int year)
    {
110     int totalDays = 0;
111     String query = ""
112     SELECT SUM(DATEDIFF(endDate, startDate)+1) AS totalDays
113     FROM holiday
114     WHERE employeeId=?
115     AND YEAR(startDate)=?
116     """;
117
118     try (Connection conn = DBConnection.getConnection();
        PreparedStatement stmt = conn.prepareStatement(query)) {
119         int employeeId = getEmployeeIdByName(employeeName);
120         if (employeeId == -1) return 0;
121
122         stmt.setInt(1, employeeId);
123         stmt.setInt(2, year);
124
125         ResultSet rs = stmt.executeQuery();
126         if (rs.next()) {
127             totalDays = rs.getInt("totalDays");
128         }
129     }

```

```

130     } catch (SQLException e) {
131         e.printStackTrace();
132     }
133     return totalDays;
134 }
135
136
137 public List<String> getAllEmployeeNames() {
138     List<String> employeeNames = new ArrayList<>();
139     String query = "SELECT CONCAT(nom, ' ', prenom) AS fullName
140                     FROM employee";
141     try (Connection conn = DBConnection.getConnection();
142          PreparedStatement stmt = conn.prepareStatement(query);
143          ResultSet rs = stmt.executeQuery()) {
144         while (rs.next()) employeeNames.add(rs.getString("fullName"));
145     } catch (SQLException e) {
146         e.printStackTrace();
147     }
148     return employeeNames;
149 }
150
151 public List<Type> getAllHolidayTypes() {
152     List<Type> holidayTypes = new ArrayList<>();
153     for (Type type : Type.values()) holidayTypes.add(type);
154     return holidayTypes;
155 }
156
157 public boolean hasOverlappingHoliday(String employeeName, String
158                                     startDate, String endDate) {
159     String query = ""
160     ""SELECT COUNT(*) AS overlapCount
161     FROM holiday
162     WHERE employeeId = ?
163     AND (
164     (startDate <= ? AND endDate >= ?) OR
165     (startDate <= ? AND endDate >= ?) OR
166     (startDate >= ? AND endDate <= ?)
167     )
168     """;
169
170     try (Connection conn = DBConnection.getConnection();
171          PreparedStatement stmt = conn.prepareStatement(query)) {
172         int employeeId = getEmployeeIdByName(employeeName);
173         if (employeeId == -1) return false;
174
175         stmt.setInt(1, employeeId);
176         stmt.setString(2, startDate);
177         stmt.setString(3, startDate);
178         stmt.setString(4, endDate);
179         stmt.setString(5, endDate);
180         stmt.setString(6, startDate);
181         stmt.setString(7, endDate);

```

```

178         ResultSet rs = stmt.executeQuery();
179         if (rs.next()) {
180             return rs.getInt("overlapCount") > 0;
181         }
182     } catch (SQLException e) {
183         e.printStackTrace();
184     }
185     return false;
186 }
187
188
189 }

```

## 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 :

```

1 package View;
2
3 import javax.swing.*.*;
4 import java.awt.*.*;
5
6 public class HolidayView extends JFrame {
7     public JTable holidayTable;
8     public JButton addButton, listButton, deleteButton, modifyButton,
9         switchViewButton;
10    public JComboBox<String> employeeNameComboBox;
11    public JTextField startDateField, endDateField;
12    public JComboBox<String> typeCombo;
13
14    public HolidayView() {
15        setTitle("Gestion des Congés");
16        setSize(800, 600);
17        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18        setLayout(new BorderLayout());
19
20        // Panneau de saisie des champs
21        JPanel inputPanel = new JPanel(new GridLayout(0, 2, 10, 10));
22        inputPanel.add(new JLabel("Employé Nom Complet:"));
23        employeeNameComboBox = new JComboBox<>();
24        inputPanel.add(employeeNameComboBox);
25
26        inputPanel.add(new JLabel("Date Début:"));
27        startDateField = new JTextField();
28        inputPanel.add(startDateField);

```

```

29     inputPanel.add(new JLabel("Date Fin:"));
30     endDateField = new JTextField();
31     inputPanel.add(endDateField);
32
33     inputPanel.add(new JLabel("Type:"));
34     typeCombo = new JComboBox<>(new String[]{"CONGE_PAYE",
35         "CONGE_MALADIE", "CONGE_NON_PAYE"});
36     inputPanel.add(typeCombo);
37
38     add(inputPanel, BorderLayout.NORTH);
39
40     // Table des congés
41     holidayTable = new JTable();
42     add(new JScrollPane(holidayTable), BorderLayout.CENTER);
43
44     // Boutons d'action
45     JPanel buttonPanel = new JPanel();
46     addButton = new JButton("Ajouter");
47     buttonPanel.add(addButton);
48     listButton = new JButton("Afficher");
49     buttonPanel.add(listButton);
50     deleteButton = new JButton("Supprimer");
51     buttonPanel.add(deleteButton);
52     modifyButton = new JButton("Modifier");
53     buttonPanel.add(modifyButton);
54
55     switchViewButton = new JButton("Gerer les Employés");
56     buttonPanel.add(switchViewButton);
57
58     add(buttonPanel, BorderLayout.SOUTH);
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             -> {
31                 if (!e.getValueIsAdjusting()) { // V rifie si la
32                     s lection est termin e
33                     int selectedRow = view.holidayTable.getSelectedRow();
34                     if (selectedRow != -1) {
35                         // R cup rer l'ID et les autres informations de
36                         la ligne s lectionn e
37                         int id = (int)
38                             view.holidayTable.getValueAt(selectedRow, 0);
39                         // R cup re l'ID depuis la colonne 0
40                         String employeeName = (String)
41                             view.holidayTable.getValueAt(selectedRow, 1);
42                         String startDate = (String)
43                             view.holidayTable.getValueAt(selectedRow, 2);
44                         String endDate = (String)
45                             view.holidayTable.getValueAt(selectedRow, 3);
46                         Type type =
47                             Type.valueOf(view.holidayTable.getValueAt(selectedRow,
48                                 4).toString());
49
50                         // Remplir les champs de modification avec ces
51                         valeurs
52                         view.employeeNameComboBox.setSelectedItem(employeeName);
53                         view.startDateField.setText(startDate);
54                         view.endDateField.setText(endDate);
55                         view.typeCombo.setSelectedItem(type.toString());
56                     }
57                 }
58             }
59         );
60     }
61 }

```

```

45
46         // Modifier l'action du bouton de modification
47         // pour utiliser l'ID de la ligne sélectionnée
48         view.modifyButton.setActionCommand(String.valueOf(id));
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 employé trouvé.");
59     } else {
60         System.out.println("Noms des employés chargés : " +
61             names);
62         for (String name : names) {
63             view.employeeNameComboBox.addItem(name);
64         }
65     }
66
67 private void refreshHolidayTable() {
68     List<Holiday> holidays = dao.listAll();
69     String[] columnNames = {"ID", "Employé", "Date Début", "Date
70         Fin", "Type"};
71     Object[][] data = new Object[holidays.size()][5];
72
73     for (int i = 0; i < holidays.size(); i++) {
74         Holiday h = holidays.get(i);
75         data[i] = new Object[]{h.getId(), h.getEmployeeName(),
76             h.getStartDate(), h.getEndDate(), h.getType()};
77     }
78
79     view.holidayTable.setModel(new
80         javax.swing.table.DefaultTableModel(data, columnNames));
81 }
82
83 private boolean isValidDate(String date) {
84     try {
85         DateTimeFormatter formatter =
86             DateTimeFormatter.ISO_LOCAL_DATE;
87         LocalDate.parse(date, formatter);
88         return true;
89     } catch (Exception e) {
90         return false;
91     }
92 }
93
94 private boolean isEndDateAfterStartDate(String startDate, String
95     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 private void addHoliday() {
98     try {
99         String employeeName = (String)
100             view.employeeNameComboBox.getSelectedItemAt();
101         String startDate = view.startDateField.getText();
102         String endDate = view.endDateField.getText();
103         Type type =
104             Type.valueOf(view.typeCombo.getSelectedItemAt().toString().toUpperCase());
105
106         if (!isValidDate(startDate) || !isValidDate(endDate)) {
107             throw new IllegalArgumentException("Les dates doivent
108                 tre au format YYYY-MM-DD.");
109         }
110
111         if (!isEndDateAfterStartDate(startDate, endDate)) {
112             throw new IllegalArgumentException("La date de fin
113                 doit tre sup rieure la date de d but.");
114         }
115
116         DateTimeFormatter formatter =
117             DateTimeFormatter.ISO_LOCAL_DATE;
118         LocalDate start = LocalDate.parse(startDate, formatter);
119         LocalDate end = LocalDate.parse(endDate, formatter);
120
121         // V rifier si les dates chevauchent avec un cong
122         // existant
123         if (dao.hasOverlappingHoliday(employeeName, startDate,
124             endDate)) {
125             throw new IllegalArgumentException("Chevauchement
126                 d tect avec un cong existant pour cet
127                 employ .");
128         }
129
130         // Calcul de la dur e en jours
131         long daysBetween =
132             java.time.temporal.ChronoUnit.DAYS.between(start, end)
133             + 1;
134
135         if (daysBetween > 25) {
136             throw new IllegalArgumentException("La dur e du
137                 cong ne peut pas d passer 25 jours.");
138         }
139
140         // V rifier le total des jours pris cette ann e
141         int year = start.getYear();
142         long existingDays =
143             dao.getTotalDaysTakenThisYear(employeeName, year);

```

```

131         if (existingDays + daysBetween > 25) {
132             throw new IllegalArgumentException(
133                 "L'employé a déjà pris " + existingDays + "
134                 jours de congé cette année. "
135                 + "Le total ne peut pas dépasser 25 jours."
136             );
137         }
138
139         // Ajouter le congé
140         Holiday holiday = new Holiday(employeeName, startDate,
141             endDate, type);
142         dao.add(holiday);
143         loadEmployeeNames();
144         refreshHolidayTable();
145         JOptionPane.showMessageDialog(view, "Congé ajouté avec
146             succès.");
147     } catch (Exception ex) {
148         JOptionPane.showMessageDialog(view, "Erreur : " +
149             ex.getMessage());
150     }
151 }
152
153 private void modifyHoliday() {
154     try {
155         // Récupérer l'ID du congé à partir de l'action du
156         // bouton
157         String actionCommand =
158             view.modifyButton.getActionCommand();
159         if (actionCommand != null &&
160             !actionCommand.trim().isEmpty()) {
161             int id = Integer.parseInt(actionCommand.trim());
162
163             String employeeName = (String)
164                 view.employeeNameComboBox.getSelectedItem();
165             String startDate = view.startDateField.getText();
166             String endDate = view.endDateField.getText();
167             Type type =
168                 Type.valueOf(view.typeCombo.getSelectedItem().toString().toUpperCase());
169
170             if (!isValidDate(startDate) || !isValidDate(endDate)) {
171                 throw new IllegalArgumentException("Les dates
172                     doivent être au format YYYY-MM-DD.");
173             }
174
175             if (!isEndDateAfterStartDate(startDate, endDate)) {
176                 throw new IllegalArgumentException("La date de fin
177                     doit être supérieure à la date de début.");
178             }
179
180             if (dao.hasOverlappingHoliday(employeeName, startDate,
181                 endDate)) {
182                 throw new IllegalArgumentException("L'employé a
183                     déjà un congé durant cette période.");
184             }
185         }
186     }
187 }

```



```

171
172
173         Holiday holiday = new Holiday(employeeName, startDate,
174             endDate, type);
175         dao.update(holiday, id);
176         loadEmployeeNames();
177         refreshHolidayTable();
178         JOptionPane.showMessageDialog(view, "Cong    modifi
179             avec succ s.");
180     } else {
181         JOptionPane.showMessageDialog(view, "Veuillez
182             s lectionner un cong    modifier.");
183     }
184 } catch (Exception ex) {
185     JOptionPane.showMessageDialog(view, "Erreur : " +
186         ex.getMessage());
187 }
188
189 private void deleteHoliday() {
190     try {
191         // V rifier si une ligne est s lectionn e dans la table
192         int selectedRow = view.holidayTable.getSelectedRow();
193         if (selectedRow == -1) {
194             JOptionPane.showMessageDialog(view, "Veuillez
195                 s lectionner un cong    supprimer.");
196             return;
197         }
198
199         // R cup rer l'ID du cong    partir du mod le de table
200         int id = (int) view.holidayTable.getValueAt(selectedRow,
201             0); // Supposons que la colonne 0 contient l'ID
202
203         // Demander confirmation avant de supprimer
204         int confirm = JOptionPane.showConfirmDialog(view,
205             " tes -vous s r de vouloir supprimer le cong
206                 avec l'ID " + id + " ?",
207             "Confirmation de suppression",
208             JOptionPane.YES_NO_OPTION);
209
210         if (confirm == JOptionPane.YES_OPTION) {
211             // Supprimer le cong    via le DAO
212             dao.delete(id);
213
214             // Rafra chir les donn es dans la vue
215             loadEmployeeNames();
216             refreshHolidayTable();
217
218             JOptionPane.showMessageDialog(view, "Cong    supprim
219                 avec succ s.");
220         } else {
221             JOptionPane.showMessageDialog(view, "Suppression
222                 annul e.");
223         }
224     }

```

```

215     }
216     } catch (Exception ex) {
217         JOptionPane.showMessageDialog(view, "Erreur : " +
            ex.getMessage());
218     }
219 }
220
221 }

```

## 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 :

```

1 // Classe principale pour démarrer l'application
2 package Main;
3
4 import Controller.EmployeeController;
5 import Controller.HolidayController;
6 import View.EmployeeView;
7 import View.HolidayView;
8
9 public class Main {
10     public static void main(String[] args) {
11         // Créer les vues
12         EmployeeView employeeView = new EmployeeView();
13         HolidayView holidayView = new HolidayView();
14
15         // Créer les contrôleurs pour chaque vue
16         new EmployeeController(employeeView, holidayView);
17         new HolidayController(holidayView);
18
19         // Définir quelle vue sera affichée par défaut (exemple :
20         // vue des employés)
21         employeeView.setVisible(true);
22
23         // Ajouter un gestionnaire pour passer de l'une à l'autre
24         employeeView.switchViewButton.addActionListener(e -> {
25             employeeView.setVisible(false);
26             holidayView.setVisible(true);
27         });
28
29         holidayView.switchViewButton.addActionListener(e -> {

```

```

29         holidayView.setVisible(false);
30         employeeView.setVisible(true);
31     });
32 }
33 }

```

## Fonctionnement de l'application

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

### 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.

**Gestion des Congés**

Employé Nom Complet: amal ellaoui

Date Début: 2024-01-10

Date Fin: 2024-01-20

Type: CONGE\_MALADIE

ID	Employé	Date Début	Date Fin	Type
8	amal ellaoui	2024-11-10	2024-11-25	CONGE_MALADIE
12	houssam mazwara	2024-04-05	2024-04-15	CONGE_MALADIE
13	hafsia fekari	2024-01-10	2024-01-20	CONGE_MALADIE

**Message**

i Congé ajouté avec succès.

OK

Ajouter Afficher Supprimer Modifier Gerer les Employés

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

## Cas de dépassement de solde de congé :

Lorsqu'un congé a été ajouté pour l'employé avec l'ID 4, avec une date de début le 10 janvier 2024 et une date de fin le 10 janvier 2024, tout s'est bien passé. Cependant, lorsque nous avons tenté d'ajouter un autre congé pour le même employé, avec une date de début le 1er mars 2024 et une date de fin le 20 mars 2024, le système a affiché un message d'erreur. Ce message indiquait que le solde de congé de l'employé avait été dépassé, car la durée totale des congés demandés pour l'année 2024 dépassait le maximum autorisé de 25 jours.

The screenshot shows a web application window titled "Gestion des Congés". It contains a form for adding a new leave request. The form fields are: "Employé Nom Complet:" with a dropdown menu showing "hafsa fekari"; "Date Début:" with a date picker showing "2024-03-01"; "Date Fin:" with a date picker showing "2024-03-20"; and "Type:" with a dropdown menu showing "CONGE\_MALADIE". Below the form is a table displaying existing leave requests. The table has columns: ID, Employé, Date Début, Date Fin, and Type. The data rows are: ID 8, Employé amal ellaoui, Date Début 2024-11-10, Date Fin 2024-11-25, Type CONGE\_MALADIE; ID 12, Employé houssam mazwara, Date Début 2024-04-05, Date Fin 2024-04-15, Type CONGE\_MALADIE; and ID 13, Employé hafsa fekari, Date Début 2024-01-10, Date Fin 2024-01-20, Type CONGE\_MALADIE. At the bottom of the window are five buttons: "Ajouter", "Afficher", "Supprimer", "Modifier", and "Gerer les Employés". An error message dialog box is overlaid on the form. The dialog box has a title bar "Message" and a close button. It contains an information icon and the text: "Erreur : L'employé a déjà pris 11 jours de congé cette année. Le total ne peut pas dépasser 25 jours." Below the text is an "OK" button.

ID	Employé	Date Début	Date Fin	Type
8	amal ellaoui	2024-11-10	2024-11-25	CONGE_MALADIE
12	houssam mazwara	2024-04-05	2024-04-15	CONGE_MALADIE
13	hafsa fekari	2024-01-10	2024-01-20	CONGE_MALADIE

Message

Erreur : L'employé a déjà pris 11 jours de congé cette année. Le total ne peut pas dépasser 25 jours.

OK

Ajouter Afficher Supprimer Modifier Gerer les Employés

Figure 2: Interface pour l'ajout d'une demande de congé qui dépasse le solde.

## Cas de chevauchement de congé :

Lorsqu'un congé a été ajouté pour l'employé Houssam Mazwara, avec une date de début le 5 avril 2024 et une date de fin le 15 avril 2024, tout s'est bien passé. Cependant, lorsque nous avons tenté d'ajouter un autre congé pour le même employé, avec une date de début le 10 avril 2024 et une date de fin le 20 avril 2024, le système a affiché un message d'erreur. Ce message indiquait que l'employé avait déjà un congé durant cette période.

Gestion des Congés

Employé Nom Complet:

hafsa fekari

Date Début:

2024-03-01

Date Fin:

2024-03-20

Type:

CONGE\_MALADIE

ID	Employé	Date Début	Date Fin	Type
8	amal ellaoui	2024-11-10	2024-11-25	CONGE_MALADIE
12	houssam mazwara	2024-04-05	2024-04-15	CONGE_MALADIE
13	hafsa fekari	2024-01-10	2024-01-20	CONGE_MALADIE

Message

i

Erreur : L'employé a déjà pris 11 jours de congé cette année. Le total ne peut pas dépasser 25 jours.

OK

Ajouter

Afficher

Supprimer

Modifier

Gerer les Employés

Figure 3: Interface pour l'ajout d'une demande de congé qui dépasse le solde.

## 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.

Gestion des Congés

Employé Nom Complet:

Date Début:

Date Fin:

Type:

ID	Employé	Date Début	Date Fin	Type
8	amal ellaoui	2024-11-10	2024-11-25	CONGE_MALADIE
12	houssam mazwara	2024-04-05	2024-04-15	CONGE_MALADIE

Figure 4: Interface pour l’affichage des demandes de congé.

## 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.

Gestion des Congés

Employé Nom Complet:

MOAA ellaoui

Date Début:

2024-04-05

Date Fin:

2024-04-25

Type:

CONGE\_MALADIE

ID	Employé	Date Début	Date Fin	Type
8	amal ellaoui	2024-11-10	2024-11-25	CONGE_MALADIE
12	houssam mazwara	2024-04-05	2024-04-15	CONGE_MALADIE
13	hafsa fekari	2024-01-10	2024-01-20	CONGE_MALADIE
15	MOAA ellaoui	2024-04-05	2024-04-25	CONGE_MALADIE

Confirmation de suppression

?

Êtes-vous sûr de vouloir supprimer le congé avec l'ID 15 ?

Oui

Non

Ajouter

Afficher

Supprimer

Modifier

Gerer les Employés

Gestion des Congés

Employé Nom Complet:

amal ellaoui

Date Début:

2024-04-05

Date Fin:

2024-04-25

Type:

CONGE\_MALADIE

ID	Employé	Date Début	Date Fin	Type
3	amal ellaoui	2024-11-10	2024-11-25	CONGE_MALADIE
12	houssam mazwara	2024-04-05	2024-04-15	CONGE_MALADIE
13	hafsa fekari	2024-01-10	2024-01-20	CONGE_MALADIE

Message

i

Congé supprimé avec succès.

OK

## Modification d'une demande de congé

Cette fonctionnalité permet de modifier les informations d'une demande d'un congé existant. L'utilisateur sélectionne un congé, met à jour les champs nécessaires, puis enregistre les modifications

The screenshot shows a window titled "Gestion des Congés". It contains a form with the following fields:

- Employé Nom Complet: amal ellaoui
- Date Début: 2024-11-01
- Date Fin: 2024-11-24
- Type: CONGE\_MALADIE

Below the form is a table with the following data:

ID	Employé	Date Début	Date Fin	Type
8	amal ellaoui	2024-11-01	2024-11-24	CONGE_MALADIE
13	hafsa fekari	2024-01-10	2024-01-20	CONGE_MALADIE
18	hafsa fekari	2024-03-10	2024-03-20	CONGE_MALADIE
20	houssam mazware	2024-01-10	2024-01-25	CONGE_MALADIE

A modal dialog box titled "Message" is displayed in the center, showing an information icon and the text "Congé modifié avec succès." with an "OK" button.

At the bottom of the window are five buttons: "Ajouter", "Afficher", "Supprimer", "Modifier", and "Gerer les Employés".

Figure 6: Interface pour la modification des informations d'un employé.

## 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.