



المدرسة العليا
للتكنولوجيا-اسفي
ECOLE SUPÉRIEURE
DE TECHNOLOGIE -SAFI

Compte Rendu du TP N°2 : Gestion Congés (Généricités, MVC et DAO)

Réalisé par : Aya EL ALAMA

Année universitaire : 2024-2025

Elément : Java Avancé


























Groupe 2

Introduction :

Ce compte rendu présente les étapes et les résultats obtenus lors de la réalisation d'un TP visant à implémenter une application de gestion des congés. Le projet a été développé en suivant une architecture MVC et intègre la notion de généricité pour garantir une flexibilité et une réutilisabilité accrues dans la gestion des données. Les travaux ont couvert l'ensemble des couches de l'application, depuis la définition du modèle de données jusqu'à la conception de l'interface utilisateur. Les principaux défis abordés incluent la gestion des congés via une base de données, l'implémentation de règles métier spécifiques, et la création d'une interface graphique intuitive pour interagir avec l'utilisateur.

Organisation des fichiers Java

La structure du projet repose sur une organisation claire et modulaire des fichiers Java, conformément au modèle MVC. Chaque couche est représentée par des classes spécifiques regroupées dans des packages :

- ▼  GC
 - >  JRE System Library [JavaSE-22]
 - ▼  src
 - ▼  Controller
 - >  EmployeeController.java
 - >  HolidayController.java
 - >  ViewController.java
 - ▼  DAO
 - >  DBConnection.java
 - >  EmployeeDAOImpl.java
 - >  GenericDAO.java
 - >  HolidayDAOImpl.java
 - ▼  Main
 - >  Main.java
 - ▼  Model
 - >  Employee.java
 - >  Holiday.java
 - >  Poste.java
 - >  Role.java
 - >  Type.java
 - ▼  View
 - >  EmployeeView.java
 - >  HolidayView.java
 - ▼  Referenced Libraries
 - >  mysql-connector-j-9.1.0.jar - C:\

Dans cette section, nous présentons des captures d'écran illustrant les principales fonctionnalités de l'application, avant d'entrer dans les détails des codes qui les implémentent.

Gestion des Employés

Nom:

Prénom:

Email:

Téléphone:

Salaire:

Rôle:

Poste:

ID	Nom	Prénom	Email	Téléphone	Salaire	Rôle	Poste
2	AMRANI	Nawal	nwlam@email.c...	0623456789	7000.0	EMPLOYE	INGENIEURE_E...
3	El Yassir	Rachid	rachid.elyassir@...	0612345678	5000.0	EMPLOYE	TEAM_LEADER

Ajouter Afficher Supprimer Modifier Gérer les Congés

Figure 1 interface gestion des employés

Gestion des Congés

Employé Nom Complet:

Date Début:

Date Fin:

Type:

ID	Employé	Date Début	Date Fin	Type
1	AMRANI Nawal	2024-12-01	2024-12-10	CONGE_PAYE
2	El Yassir Rachid	2024-12-15	2024-12-20	CONGE_MALADIE

Ajouter Afficher Supprimer Modifier Gerer les Employés

Figure 2 interface gestion des congés

Gestion des Employés

Nom: EL ALAMA

Prénom: Aya

Email: ael@gmail.com

Téléphone: 0674603637

Salaire: 8000

Rôle: Employe

Poste:

ID	Nom	Prénom	Salaire	Rôle	Poste
2	AMRANI	Nawal		EMPLOYE	INGENIEURE_E..
3	El Yassir	Rachid		EMPLOYE	TEAM_LEADER

Message

Employé ajouté avec succès.

OK

Ajouter Afficher Supprimer Modifier Gérer les Congés

Figure 3 Ajout réussi

Gestion des Employés

Nom: EL ALAMA

Prénom: Aya

Email: ael@gmail.com

Téléphone: 0674603637

Salaire: 8000

Rôle: Employe

Poste: PILOTE

ID	Nom	Prénom	Email	Téléphone	Salaire	Rôle	Poste
2	AMRANI	Nawal	nwlam@email.c...	0623456789	7000.0	EMPLOYE	INGENIEURE_E..
3	El Yassir	Rachid	rachid.elyassir@...	0612345678	5000.0	EMPLOYE	TEAM_LEADER
5	EL ALAMA	Aya	ael@gmail.com	0674603637	8000.0	EMPLOYE	PILOTE

Ajouter Afficher Supprimer Modifier Gérer les Congés

Figure 4 affichage réussi

	id	nom	prenom	email	phone	salaire	role	poste
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	AMRANI	Nawal	nwlam@email.com	0623456789	7000.00	EMPLOYE	INGENIEURE_ETUDE_ET_DEVELOPPEMENT
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	3	El Yassir	Rachid	rachid.elyassir@email.com	0612345678	5000.00	EMPLOYE	TEAM_LEADER
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	5	EL ALAMA	Aya	ael@gmail.com	0674603637	8000.00	EMPLOYE	PILOTE

Gestion des Employés

Nom: EL ALAMA

Prénom: Aya

Email: ael@gmail.com

Téléphone: 0674603637

Salaire: 8000

Rôle: Employe

Poste:

ID	Nom	Prénom	Salaire	Rôle	Poste
2	AMRANI	Nawal		EMPLOYE	INGENIEURE_E...
3	El Yassir	Rachid		EMPLOYE	TEAM_LEADER
5	EL ALAMA	Aya		EMPLOYE	PILOTE

Ajouter Afficher Supprimer Modifier Gérer les Congés

Suppression d'un employé

Entrez l'ID de l'employé à supprimer :

3

OK Cancel

Confirmation de suppression

Êtes-vous sûr de vouloir supprimer l'employé avec l'ID 3 ?

Yes No

Message

Employé supprimé avec succès.

OK

Figure 5 suppression avec succès

Gestion des Employés

Nom: AMRANI

Prénom: Sara

Email: nwlam@email.com

Téléphone: 0623456789

Salaire: 7000.0

Rôle: Employe

Poste: ET_DEVELOPPEMENT

ID	Nom	Prénom	Salaire	Rôle	Poste
2	AMRANI	Nawal		EMPLOYE	INGENIEURE_E...
5	EL ALAMA	Aya		EMPLOYE	PILOTE

Message

Employé mis à jour avec succès.

OK

Ajouter Afficher Supprimer Modifier Gérer les Congés

Gestion des Employés

Nom: AMRANI

Prénom: Sara

Email: nwlam@email.com

Téléphone: 0623456789

Salaire: 7000.0

Rôle: Employe

Poste: INGENIEURE_ETUDE_ET_DEVELOPPEMENT

ID	Nom	Prénom	Email	Téléphone	Salaire	Rôle	Poste
2	AMRANI	Sara	nwlam@email.c...	0623456789	7000.0	EMPLOYE	INGENIEURE_E
5	EL ALAMA	Aya	ael@gmail.com	0674603637	8000.0	EMPLOYE	PILOTE

Ajouter Afficher Supprimer Modifier Gérer les Congés

Figure 6 modification réussie

Gestion des Congés

Employé Nom Complet: AMRANI Sara


Date Début: 2024-12-17

Date Fin: 2024-12-20

Type: CONGE_MALADIE

ID	Employé	Date Début	Date Fin	Type
3	AMRANI Sara	2024-12-01	2024-12-10	CONGE_PAYE

Message

 Congé modifié avec succès.

OK

Ajouter Afficher Supprimer Modifier Gerer les Employés

Figure 7 modification de la date début réussie

Gestion des Congés

Employé Nom Complet: EL ALAMA Aya

Date Début: 2024-12-01

Date Fin: 2024-12-10

Type: CONGE_PAYE

ID	Employé	Date Début	Date Fin	Type
3	AMRANI Sara	2024-12-01	2024-12-17	CONGE_PAYE
6	EL ALAMA Aya	2024-12-01	2024-12-10	CONGE_PAYE

Ajouter Afficher Supprimer Modifier Gerer les Employés

Figure 8 ajout d'un nouveau congé réussi

	Employé	Date Début	Date Fin	
	AMRANI Sara	2024-12-01	2024-12-17	CON
	EL ALAMA Aya	2024-12-01	2024-12-10	CON

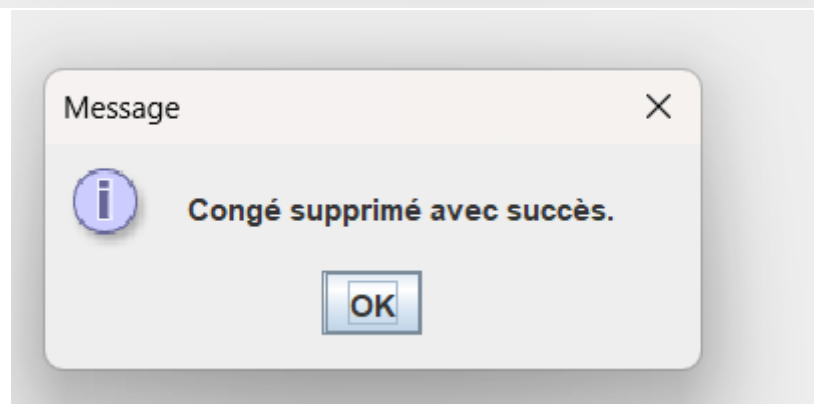
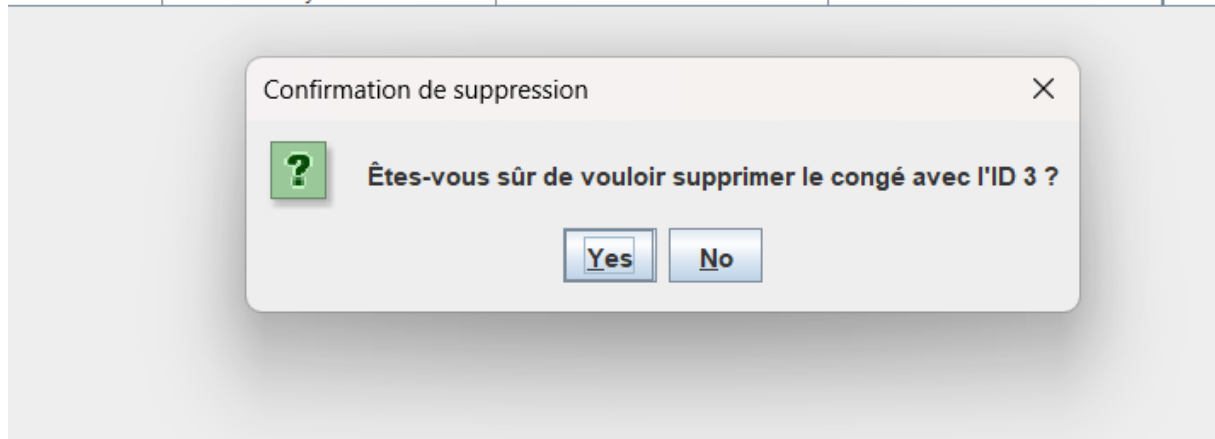


Figure 9 suppression réussie

Après avoir présenté les captures d'écran illustrant les principales fonctionnalités de l'application, nous allons maintenant passer à l'explication détaillée du code, en commençant par les différentes classes et leurs rôles

Holiday.java :

1. Attributs (Variables d'instance) :

- **id** : Identifiant unique du congé.
- **employeeId** : ID de l'employé associé.
- **employeeName** : Nom de l'employé.
- **startDate et endDate** : Dates de début et de fin du congé.
- **type** : Type de congé (défini par une énumération `Type`).

2. Constructeurs :

- **Avec tous les paramètres** : Inclut `id`, `employeeName`, `startDate`, `endDate`, et `type`.

- **Sans `employeeId`** : Utilisé quand l'ID de l'employé n'est pas nécessaire (par exemple pour l'affichage).
- **Sans `employeeName`** : Pour ajouter ou mettre à jour un congé (lorsqu'on n'a pas le nom complet).

3. Méthodes :

- **Getters** : Permettent d'obtenir les valeurs des attributs.
- **Setter pour `employeeName`** : Permet de modifier le nom de l'employé après création.

Code :

```
package Model;

public class Holiday {
    private int id; // Identifiant du congé
    private int employeeId; // ID de l'employé
    private String employeeName; // Nom complet de l'employé
    private String startDate; // Date de début
    private String endDate; // Date de fin
    private Type type; // Type de congé (enum)

    // Constructeur avec employeeName pour listAll()
    public Holiday(int id, String employeeName, String startDate,
String endDate, Type type) {
        this.id = id;
        this.employeeName = employeeName;
        this.startDate = startDate;
        this.endDate = endDate;
        this.type = type;
    }
    public Holiday(String employeeName, String startDate, String
endDate, Type type) {
        this.employeeName = employeeName;
        this.startDate = startDate;
        this.endDate = endDate;
        this.type = type;
    }

    // Constructeur pour add() et update() (sans employeeName)
    public Holiday(int employeeId, String startDate, String endDate,
Type type) {
        this.employeeId = employeeId;
        this.startDate = startDate;
        this.endDate = endDate;
        this.type = type;
    }

    // Getters et Setters
```

```

    public int getId() {
        return id;
    }

    public int getEmployeeId() {
        return employeeId;
    }

    public String getEmployeeName() {
        return employeeName; // Retourne le nom complet
    }

    public String getStartDate() {
        return startDate;
    }

    public String getEndDate() {
        return endDate;
    }

    public Type getType() {
        return type;
    }

    public void setEmployeeName(String employeeName) {
        this.employeeName = employeeName;
    }
}

```

Type.java :

La classe `Type` est une **énumération** (enum) qui définit les différents types de congés disponibles dans l'application. Elle contient trois valeurs possibles :

- `CONGE_MALADIE` : Représente un congé pour raison de maladie.
- `CONGE_PAYE` : Représente un congé payé, tel que des vacances.
- `CONGE_NON_PAYE` : Représente un congé non payé.

Les énumérations sont utilisées pour définir un ensemble de constantes, ici pour le type de congé, ce qui permet d'assurer une gestion claire et sécurisée des différents types de congés dans l'application.

Code :

```

package Model;

public enum Type {
    CONGE_MALADIE,
    CONGE_PAYE,

```

```
        CONGE_NON_PAYE  
    }  
}
```

GenericDAO.java :

La classe `GenericDAO` est une interface générique utilisée pour définir des opérations de base sur des entités dans une application, telles que l'ajout, la suppression, la mise à jour et la recherche. Voici une explication de ses méthodes :

1. **add(T entity)** : Ajoute un objet de type générique `T` (par exemple, une entité `Holiday` ou une autre) à la base de données.
2. **delete(int id)** : Supprime un objet de type `T` en fonction de son identifiant unique `id`.
3. **listAll()** : Retourne une liste de tous les objets de type `T` présents dans la base de données.
4. **findById(int id)** : Recherche et retourne un objet de type `T` à partir de son identifiant unique `id`.
5. **update(T entity, int id)** : Met à jour un objet existant dans la base de données en fonction de son `id`.

Cette interface permet de centraliser les opérations courantes de gestion des données pour différentes entités, tout en offrant la flexibilité d'utiliser des types différents grâce à la généricité (`<T>`). Cela permet une réutilisation facile et une gestion uniforme des entités dans le projet.

Code :

```
package DAO;  
  
import java.util.List;  
public interface GenericDAO<T> {  
    void add(T entity); // Ajouter un objet  
    void delete(int id); // Supprimer un objet par ID  
    List<T> listAll(); // Lister tous les objets  
    T findById(int id); // Trouver un objet par ID  
    void update(T entity, int id); // Mettre à jour un objet  
}
```

HolidayDAOImpl.java :

Cette classe gère l'interaction avec la base de données pour manipuler les informations relatives aux congés des employés. Elle permet d'ajouter, de supprimer, de lister, de mettre à jour et de rechercher des congés, tout en effectuant des vérifications sur les employés et les types de congé. L'utilisation de SQL dans les méthodes garantit une gestion dynamique et efficace des données.

Code :

```

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) = ?";
    // Méthode pour ajouter un congé
    @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 : Employé 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("Congé ajouté avec succès.");
        } catch (SQLException e) {
            System.err.println("Erreur lors de l'ajout du congé : "
+ e.getMessage());
            e.printStackTrace();
        }
    }
}

```

```

    }

    // Méthode pour supprimer un congé 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("Congé supprimé avec succès.");
            } else {
                System.out.println("Aucun congé trouvé avec cet
ID.");
            }
        } catch (SQLException e) {
            System.err.println("Erreur lors de la suppression du
congé : " + e.getMessage());
            e.printStackTrace();
        }
    }

    // Méthode pour lister tous les congés
    @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 récupération des
congés : " + e.getMessage());
            e.printStackTrace();
        }
        return holidays;
    }
}

```

```

// Méthode pour trouver un congé 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 congé
: " + e.getMessage());
        e.printStackTrace();
    }
    return null;
}

// Méthode pour mettre à jour un congé
@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 : Employé 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("Congé mis à jour avec succès.");
        } else {
            System.out.println("Aucun congé trouvé avec cet
ID.");
        }
    }
}

```

```

    }
    } catch (SQLException e) {
        System.err.println("Erreur lors de la mise à jour du
congé : " + e.getMessage());
        e.printStackTrace();
    }
}

// Méthode pour récupérer l'ID de l'employé par nom complet
public int getEmployeeIdByName(String employeeName) {
    try (Connection conn = DBConnection.getConnection());
PreparedStatement stmt =
conn.prepareStatement(SELECT_EMPLOYEE_ID_BY_NAME_SQL)) {
        stmt.setString(1, employeeName);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            return rs.getInt("id");
        }
    } catch (SQLException e) {
        System.err.println("Erreur lors de la récupération de
l'ID employé : " + e.getMessage());
        e.printStackTrace();
    }
    return -1;
}

// Méthode pour récupérer tous les noms des employés
public List<String> getAllEmployeeNames() {
    List<String> employeeNames = new ArrayList<>();
    String query = "SELECT CONCAT(nom, ' ', prenom) AS fullName
FROM employe";

    try (Connection conn = DBConnection.getConnection();
        PreparedStatement stmt = conn.prepareStatement(query);
        ResultSet rs = stmt.executeQuery()) {

        while (rs.next()) {
            String fullName = rs.getString("fullName");
            System.out.println("Nom récupéré : " + fullName); //
Pour debug
            employeeNames.add(fullName);
        }
    } catch (SQLException e) {
        System.err.println("Erreur lors de la récupération des
noms des employés : " + e.getMessage());
    }

    return employeeNames;
}

```

```

// Méthode pour récupérer les types de congés (Enum)
public List<Type> getAllHolidayTypes() {
    List<Type> holidayTypes = new ArrayList<>();
    for (Type type : Type.values()) {
        holidayTypes.add(type);
    }
    return holidayTypes;
}
}

```

HolidayController.java :

La classe HolidayController fait le lien entre la vue, le modèle et la base de données, en gérant la logique de validation et en mettant à jour la vue en conséquence. Elle utilise des actions utilisateur (ajout, modification, suppression) pour interagir avec la base de données et garantir la cohérence des données.

Code :

```

package Controller;

import DAO.HolidayDAOImpl;
import Model.Holiday;
import Model.Type;
import View.HolidayView;

import javax.swing.*.*;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.List;

public class HolidayController {
    private final HolidayView view;
    private final HolidayDAOImpl dao;

    public HolidayController(HolidayView view) {
        this.view = view;
        this.dao = new HolidayDAOImpl();

        loadEmployeeNames();
        refreshHolidayTable();

        view.addButton.addActionListener(e -> addHoliday());
        view.deleteButton.addActionListener(e -> deleteHoliday());
        view.modifyButton.addActionListener(e -> modifyHoliday());
    }
}

```



```

        // Ajout d'un écouteur de sélection sur la table

view.holidayTable.getSelectionModel().addListSelectionListener(e ->
{
    if (!e.getValueIsAdjusting()) { // Vérifie si la
sélection est terminée
        int selectedRow =
view.holidayTable.getSelectedRow();
        if (selectedRow != -1) {
            // Récupérer l'ID et les autres informations de
la ligne sélectionnée
            int id = (int)
view.holidayTable.getValueAt(selectedRow, 0); // Récupère l'ID
depuis la colonne 0
            String employeeName = (String)
view.holidayTable.getValueAt(selectedRow, 1);
            String startDate = (String)
view.holidayTable.getValueAt(selectedRow, 2);
            String endDate = (String)
view.holidayTable.getValueAt(selectedRow, 3);
            Type type =
Type.valueOf(view.holidayTable.getValueAt(selectedRow,
4).toString());

            // Remplir les champs de modification avec ces
valeurs

view.employeeNameComboBox.setSelectedItem(employeeName);
view.startDateField.setText(startDate);
view.endDateField.setText(endDate);
view.typeCombo.setSelectedItem(type.toString());

            // Modifier l'action du bouton de modification
pour utiliser l'ID de la ligne sélectionnée

view.modifyButton.setActionCommand(String.valueOf(id));
        }
    }
});
}

private void loadEmployeeNames() {
    view.employeeNameComboBox.removeAllItems();
    List<String> names = dao.getAllEmployeeNames();

    if (names.isEmpty()) {
        System.out.println("Aucun employé trouvé.");
    } else {

```

```

        System.out.println("Noms des employés chargés : " +
names);
        for (String name : names) {
            view.employeeNameComboBox.addItem(name);
        }
    }

    private void refreshHolidayTable() {
        List<Holiday> holidays = dao.listAll();
        String[] columnNames = {"ID", "Employé", "Date Début", "Date
Fin", "Type"};
        Object[][] data = new Object[holidays.size()][5];

        for (int i = 0; i < holidays.size(); i++) {
            Holiday h = holidays.get(i);
            data[i] = new Object[]{h.getId(), h.getEmployeeName(),
h.getStartDate(), h.getEndDate(), h.getType()};
        }

        view.holidayTable.setModel(new
javax.swing.table.DefaultTableModel(data, columnNames));
    }

    private boolean isValidDate(String date) {
        try {
            DateTimeFormatter formatter =
DateTimeFormatter.ISO_LOCAL_DATE;
            LocalDate.parse(date, formatter);
            return true;
        } catch (Exception e) {
            return false;
        }
    }

    private boolean isEndDateAfterStartDate(String startDate, String
endDate) {
        DateTimeFormatter formatter =
DateTimeFormatter.ISO_LOCAL_DATE;
        LocalDate start = LocalDate.parse(startDate, formatter);
        LocalDate end = LocalDate.parse(endDate, formatter);

        return end.isAfter(start);
    }

    private void addHoliday() {
        try {
            String employeeName = (String)
view.employeeNameComboBox.getSelectedItem();

```

```

        String startDate = view.startDateField.getText();
        String endDate = view.endDateField.getText();
        Type type =
Type.valueOf(view.typeCombo.getSelectedItem().toString().toUpperCase
());

        if (!isValidDate(startDate) || !isValidDate(endDate)) {
            throw new IllegalArgumentException("Les dates
doivent être au format YYYY-MM-DD.");
        }

        if (!isEndDateAfterStartDate(startDate, endDate)) {
            throw new IllegalArgumentException("La date de fin
doit être supérieure à la date de début.");
        }

        Holiday holiday = new Holiday(employeeName, startDate,
endDate, type);
        dao.add(holiday);
        loadEmployeeNames();
        refreshHolidayTable();
        JOptionPane.showMessageDialog(view, "Congé ajouté avec
succès.");
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(view, "Erreur : " +
ex.getMessage());
    }
}

private void modifyHoliday() {
    try {
        // Récupérer l'ID du congé à partir de l'action du
bouton
        String actionCommand =
view.modifyButton.getActionCommand();
        if (actionCommand != null &&
!actionCommand.trim().isEmpty()) {
            int id = Integer.parseInt(actionCommand.trim());

            String employeeName = (String)
view.employeeNameComboBox.getSelectedItem();
            String startDate = view.startDateField.getText();
            String endDate = view.endDateField.getText();
            Type type =
Type.valueOf(view.typeCombo.getSelectedItem().toString().toUpperCase
());

            if (!isValidDate(startDate) ||
!isValidDate(endDate)) {

```

```

        throw new IllegalArgumentException("Les dates
doivent être au format YYYY-MM-DD.");
    }

    if (!isEndDateAfterStartDate(startDate, endDate)) {
        throw new IllegalArgumentException("La date de
fin doit être supérieure à la date de début.");
    }

    Holiday holiday = new Holiday(employeeName,
startDate, endDate, type);
    dao.update(holiday, id);
    loadEmployeeNames();
    refreshHolidayTable();
    JOptionPane.showMessageDialog(view, "Congé modifié
avec succès.");
    } else {
        JOptionPane.showMessageDialog(view, "Veuillez
sélectionner un congé à modifier.");
    }
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(view, "Erreur : " +
ex.getMessage());
    }
}

private void deleteHoliday() {
    try {
        // Afficher un message pour entrer un ID de congé
        String input = JOptionPane.showInputDialog(view,
"Veuillez entrer l'ID du congé à supprimer:");
        if (input != null && !input.trim().isEmpty()) {
            int id = Integer.parseInt(input.trim());

            // Demander confirmation avant de supprimer
            int confirm = JOptionPane.showConfirmDialog(view,
"Êtes-vous sûr de vouloir supprimer le congé
avec l'ID " + id + " ?",
"Confirmation de suppression",
JOptionPane.YES_NO_OPTION);

            if (confirm == JOptionPane.YES_OPTION) {
                dao.delete(id); // Supprimer le congé
                loadEmployeeNames();
                refreshHolidayTable();
                JOptionPane.showMessageDialog(view, "Congé
supprimé avec succès.");
            } else {

```

```

JOptionPane.showMessageDialog(view, "Suppression
annulée.");
    }
    } else {
        JOptionPane.showMessageDialog(view, "ID de congé non
valide ou action annulée.");
    }
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(view, "ID invalide.
Veuillez entrer un nombre valide.");
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(view, "Erreur : " +
ex.getMessage());
    }
}
}
}

```

HolidayView.java :

La classe HolidayView définit la structure et les composants de l'interface utilisateur pour la gestion des congés. Elle permet à l'utilisateur de saisir des informations relatives aux congés, de visualiser la liste des congés existants dans une table, et de réaliser des actions (ajout, suppression, modification) via des boutons.

Code : **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 Congés");
        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("Employé Nom Complet:"));
        employeeNameComboBox = new JComboBox<>();
    }
}

```

```

inputPanel.add(employeeNameComboBox);

inputPanel.add(new JLabel("Date Début:"));
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 congés
holidayTable = new JTable();
add(new JScrollPane(holidayTable), BorderLayout.CENTER);

// Boutons d'action
JPanel buttonPanel = new JPanel();
addButton = new JButton("Ajouter");
buttonPanel.add(addButton);
listButton = new JButton("Afficher");
buttonPanel.add(listButton);
deleteButton = new JButton("Supprimer");
buttonPanel.add(deleteButton);
modifyButton = new JButton("Modifier");
buttonPanel.add(modifyButton);

switchViewButton = new JButton("Gerer les Employés");
buttonPanel.add(switchViewButton);

add(buttonPanel, BorderLayout.SOUTH);
}
}

```

HolidayView.java :

Code :

```

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 Congés");
        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("Employé Nom Complet:"));
        employeeNameComboBox = new JComboBox<>();
        inputPanel.add(employeeNameComboBox);

        inputPanel.add(new JLabel("Date Début:"));
        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 congés
        holidayTable = new JTable();
        add(new JScrollPane(holidayTable), BorderLayout.CENTER);

        // Boutons d'action
        JPanel buttonPanel = new JPanel();
        addButton = new JButton("Ajouter");
        buttonPanel.add(addButton);
        listButton = new JButton("Afficher");
        buttonPanel.add(listButton);
        deleteButton = new JButton("Supprimer");
        buttonPanel.add(deleteButton);
        modifyButton = new JButton("Modifier");
        buttonPanel.add(modifyButton);

```

```

        switchViewButton = new JButton("Gerer les Employés");
        buttonPanel.add(switchViewButton);

        add(buttonPanel, BorderLayout.SOUTH);
    }
}

```

Main.java :

La classe Main configure et gère les vues et contrôleurs pour l'application de gestion des employés et des congés. Elle permet de basculer entre la vue des employés et celle des congés en utilisant des boutons de commutation. Ce modèle facilite l'organisation de l'application en termes de séparation des préoccupations : chaque vue est responsable de l'affichage, tandis que chaque contrôleur gère la logique d'affaires associée.

Code : `package` Main;

```

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

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

        // Créer les contrôleurs pour chaque vue
        new EmployeeController(employeeView, holidayView);
        new HolidayController(holidayView);

        // Définir quelle vue sera affichée par défaut (exemple :
vue des employés)
        employeeView.setVisible(true);

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

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

```


Conclusion :

Le projet présente une application de gestion des employés et des congés utilisant l'architecture **Model-View-Controller (MVC)**, où chaque composant joue un rôle spécifique :

1. **Vues** : Les classes `HolidayView` et `EmployeeView` servent à afficher les informations et à interagir avec l'utilisateur. Elles permettent de gérer les congés et les employés respectivement.
2. **Contrôleurs** : Les classes `HolidayController` et `EmployeeController` sont responsables de la logique d'affaires et de la gestion des actions des utilisateurs, telles que l'ajout, la modification, et la suppression des données.
3. **Modèles** : Les objets comme `Holiday` et `Employee` (gérés via des classes DAO comme `HolidayDAOImpl` et `EmployeeDAOImpl`) représentent les données, permettant leur gestion et leur persistance.

Les actions utilisateurs sont gérées par les contrôleurs, et l'interface utilisateur est définie par les vues. L'application offre une fonctionnalité de navigation entre la gestion des employés et celle des congés via des boutons de commutation. Cela permet à l'utilisateur de basculer facilement entre les deux vues.

En résumé, ce projet illustre une implémentation claire et fonctionnelle de l'architecture MVC pour gérer des données relatives aux employés et aux congés. Le code est conçu pour être extensible et facile à maintenir.

Remarque :

L'explication détaillée et l'analyse des autres parties du code relatives à la gestion des employés (modèles, contrôleurs et autres fonctionnalités associées) sont disponibles dans l'autre document réalisé avec **LaTeX**. Vous y trouverez des informations complémentaires sur la gestion des employés et d'autres fonctionnalités de l'application.

Ce dernier compte rendu présente aussi les modifications apportées à la gestion des employés, qui ont été implémentées pour améliorer l'efficacité et l'ergonomie de l'application.