

UNIVERSITÉ CADI AYAD
ECOLE SUPERIEURE DE TECHNOLOGIE-SAFI
DUT GÉNIE INFORMATIQUE

Compte rendu

TP 2 : Système de Gestion des Congés

(Genericite,MVC, DAO, Swing)

Réalisée par :

SAISSI Zahra

Enseigné par :

M.EL ABDELLAOUI Said

Année universitaire :2024/2025

Table de matières:

Introduction:	1
Objectif:	1
Architecture du Projet:	1
Lien GitHub:	2
Étapes de Création du Projet	3
1. 1ère étape : Couche Model - Implémentation du Modèle	3
2. 2ème étape : gestion des donnée (DAO):	5
2.1. Création des tables dans la base de données	6
2.2. Interface GenericDAO:	6
2.3. Implémentation de HolidayDAOImpl:	6
2.4. Modification de EmployeDAOImpl:	9
3. 3ème étape : Logique métier:	10
4. 4ème étape : Couche View - Interface graphique	13
5. 5ème étape : Couche Controller - Implémentation du Contrôleur	19
6. 6ème étape : Main - Application principale	26
Réalisation:	27
1. Exécution du projet:	27
2. Affichage des employés:	27
3. La page du gestion de congé:	27
4. Ajouter un employé:	28
5. Afficher l'employé ajoute:	29
6. Ajouter un congé a l'employé:	29
7. Affichage des congés:	30
8. Le Solde change après l'ajout:	31
9. Si on ajoute un congé qui contient le même employé et la même date:	31
10. La modification d'un congé:	32
11. Supprimer un congé:	34
Conclusion:	36

Introduction:

Ce rapport L'entreprise SEA, confrontée à des difficultés organisationnelles récurrentes, fait face à un nouveau défi dans la gestion des congés de ses employés. Actuellement, les informations relatives aux congés sont dispersées et mal organisées, ce qui crée des inefficacités dans les processus de demande, d'approbation et de suivi. De plus, l'absence d'un système intégré de suivi des soldes de congés complique la gestion équitable des demandes par les managers et le service des ressources humaines.

En réponse à cette problématique, SEA souhaite développer un module intégré de gestion des congés, qui sera directement intégré à son application existante de gestion des employés. Cette partie permettra de centraliser les demandes, d'assurer un suivi précis des soldes de congés et de faciliter le processus d'approbation des demandes.

Objectif:

L'objectif principal de ce projet est de concevoir et de développer un module de gestion des congés qui répond aux besoins suivants :

- Permettre aux employés de soumettre leurs demandes de congé de manière simple et efficace.
- Offrir aux managers la possibilité d'approuver ou de rejeter les demandes de congé en fonction des soldes disponibles.
- Permettre au service des ressources humaines de suivre les soldes de congés de chaque employé de manière centralisée et transparente.
- Garantir que le solde de congé utilisé ne dépasse jamais le solde total disponible.
- Assurer une gestion équitable des congés en validant chaque demande en fonction du solde disponible.

Architecture du Projet:

Pour répondre aux exigences du projet de manière cohérente et scalable, nous adopterons des principes de généricité et des architectures éprouvées telles que le modèle MVC (Modèle-Vue-Contrôleur) et le design pattern DAO (Data Access Object).

- **Généricité** : Afin de rendre le système flexible et réutilisable, des classes génériques seront utilisées pour gérer les différentes entités du système (par exemple, les demandes de congés, les employés).

- **MVC (Modèle-Vue-Contrôleur)** : Cette architecture permettra de séparer les différentes couches du système, rendant ainsi l'application plus modulaire et plus facile à maintenir. Le modèle gérera la logique métier , la vue sera responsable de l'interface utilisateur, et le contrôleur s'occupera de la gestion des interactions entre le modèle et la vue .
- **DAO (Data Access Object)** : Le pattern DAO sera utilisé pour séparer la logique d'accès aux données du reste de l'application. Cela permettra de faciliter les opérations sur la base de données, telles que la création, la mise à jour et la suppression des congés, tout en garantissant une abstraction claire entre la couche de données et la couche de logique métier.

Lien GitHub:

<https://github.com/saissizahra/GestionRessourcesHumain.git>

Étapes de Création du Projet

1. 1ère étape : Couche Model - Implémentation du Modèle

1.1. Class Holiday.java:

la classe Holiday qui représente un congé dans le système de gestion des congés. Cette classe contient des attributs tels que l'ID du congé, l'ID de l'employé, les dates de début et de fin du congé, ainsi que le type de congé. Elle inclut un constructeur pour initialiser ces attributs, ainsi que des méthodes d'accès pour récupérer les informations liées à ces attributs. Une méthode getSolde() est également présente, mais elle n'est pas encore implémentée.

Code :

```
1 package Model;
2
3 import java.sql.Date;
4
5 public class Holiday {
6     private int id_holiday;
7     private int id_employe;
8     private Date startDate;
9     private Date endDate;
10    private HolidayType type;
11
12    // Constructeur de la classe Holiday
13    public Holiday(int id_holiday, int id_employe, Date startDate, Date endDate, HolidayType type) {
14        this.id_holiday = id_holiday;
15        this.id_employe = id_employe;
16        this.startDate = startDate;
17        this.endDate = endDate;
18        this.type = type;
19    }
20
21    // Méthode pour récupérer l'identifiant du congé
22    public int getId_holiday() {
23        return id_holiday;
24    }
25
26    // Méthode pour récupérer la date de début du congé
27    public Date getStartDate() {
28        return startDate;
29    }
30
31    // Méthode pour récupérer la date de fin du congé
32    public Date getEndDate() {
33        return endDate;
34    }
35
36    // Méthode pour récupérer le type de congé
37    public HolidayType getType() {
38        return type;
39    }
40
41    // Méthode pour récupérer l'identifiant de l'employé
42    public int getId_employe() {
43        return id_employe;
44    }
45
46    // Méthode pour récupérer le solde des congés (non implémentée)
47    // mais elle n'est pas encore implémentée et lève une exception pour signaler cela.
48    public Object getSolde() {
49        throw new UnsupportedOperationException("Not supported yet.");
50    }
51 }
```

Enumération des types de congé:

```
1 package Model;
2
3 public enum HolidayType {
4     CONGE_PAYE, // Congé payé
5     CONGE_NON_PAYE, // Congé non payé
6     CONGE_MALADIE // Congé maladie
7 }
```

1.2. Class Employe.java:

Dans la classe Employe, un attribut supplémentaire a été ajouté pour gérer le solde de congé de chaque employé. Cette fonctionnalité permet de suivre le nombre de jours de congé restants pour chaque employé. Le solde est géré grâce aux méthodes setSolde(int conge) et getSolde(), qui permettent de modifier et de récupérer le solde de congé respectivement. Ainsi, chaque employé dispose désormais d'une information clé sur ses droits à congé, facilitant la gestion des demandes de congé et le suivi des soldes dans le système.

Code :

```
1 package Model;
2
3 public class Employe {
4     // Déclaration des attributs de la classe Employe
5     private int id;
6     private String nom;
7     private String prenom;
8     private String email;
9     private String telephone;
10    private double salaire;
11    private Role role;
12    private Poste poste;
13    private int solde;
14
15    // Constructeur de la classe Employe
16    public Employe(int id, String nom, String prenom, String email, String telephone, double salaire, Role role, Poste poste, int solde) {
17        this.id = id;
18        this.nom = nom;
19        this.prenom = prenom;
20        this.email = email;
21        this.telephone = telephone;
22        this.salaire = salaire;
23        this.role = role;
24        this.poste = poste;
25        this.solde = solde;
26    }
}
```

```

27    // getters et setters
28    public int getId() {return id;}
29
30    public void setId(int id) {this.id = id;}
31
32    public String getNom() {return nom;}
33
34    public void setNom(String nom) {this.nom = nom;}
35
36    public String getPrenom() {return prenom;}
37
38    public void setPrenom(String prenom) {this.prenom = prenom;}
39
40    public String getEmail() {return email;}
41
42    public void setEmail(String email) {this.email = email;}
43
44    public String getTelephone() {return telephone;}
45
46    public void setTelephone(String telephone) {this.telephone = telephone;}
47
48    public double getSalaire() {return salaire;}
49
50    public void setSalaire(double salaire) {this.salaire = salaire;}
51
52    public Role getRole() {return role;}
53
54    public void setRole(Role role) {this.role = role;}
55
56    public Poste getPoste() {return poste;}
57
58    public void setPost(Poste poste) {this.poste = poste;}
59
60    public void setSolde(int conge) {this.solde = conge;}
61
62    public int getSolde() {return solde;}
63
64    }
65
..
```

2. 2ème étape : gestion des données (DAO):

La table holiday a été créée pour gérer les demandes de congé des employés, avec un attribut clé id_employé qui relie chaque demande de congé à un employé spécifique. Cette table permet de suivre les dates de début et de fin des congés, ainsi que le type de congé (payé, non payé ou maladie). De plus, un attribut solde a été ajouté à la table employé pour suivre le nombre de jours de congé restant pour chaque employé, facilitant ainsi une gestion efficace des congés.

Code SQL :

```

1 USE GestionRessourcesHumaines;
2
3 CREATE TABLE employe (
4     ID INT PRIMARY KEY AUTO_INCREMENT,
5     nom VARCHAR(100) NOT NULL,
6     prenom VARCHAR(100) NOT NULL,
7     email VARCHAR(150) NOT NULL,
8     telephone VARCHAR(15) DEFAULT NULL,
9     salaire DECIMAL(10, 2) NOT NULL,
10    solde INT(11) NOT NULL,
11    role ENUM('ADMIN', 'EMPLOYEE') NOT NULL,
12    poste ENUM('INGENIEURE_ETUDE_ET_DEVELOPPEMENT', 'TEAM_LEADER', 'PILOTE') NOT NULL
13 );
14
15 CREATE TABLE holiday (
16     id INT(11) NOT NULL AUTO_INCREMENT,
17     id_employe INT(11) NOT NULL,
18     startdate DATE DEFAULT NULL,
19     enddate DATE DEFAULT NULL,
20     type ENUM('CONGE_PAYE', 'CONGE_NON_PAYE', 'CONGE_MALADIE') NOT NULL,
21     PRIMARY KEY (id),
22     FOREIGN KEY (id_employe) REFERENCES employe(ID) ON DELETE CASCADE
23 );
24
25
```

2.1. Création des tables dans la base de données

2.2. Interface GenericDAO:

Le code fournit une interface générique GenericDAOI<T> qui définit des méthodes de base pour les opérations courantes sur une base de données, comme l'ajout, la suppression, la mise à jour et l'affichage d'objets de type générique T. Cette interface peut être implémentée pour différentes entités, permettant ainsi de travailler avec divers types de données dans une application.

Code :

```
1 package DAO;
2
3 import java.util.List;
4
5 public interface GenericDAOI<T> {
6     public void add(T e); // Ajouter une entité
7     public void delete(int id); // Supprimer une entité par son identifiant
8     public void update(T e); // Mettre à jour une entité
9     public List<T> display(); // Afficher toutes les entités
10 }
11
```

2.3. Implémentation de HolidayDAOImpl:

La classe HolidayDAOImpl, qui implémente l'interface GenericDAOI<Holiday>, permet de gérer les congés des employés dans une base de données via des opérations CRUD (création, suppression, mise à jour et affichage). Elle contient la logique nécessaire pour vérifier le solde de congés d'un employé avant l'ajout d'une nouvelle demande de congé, ainsi que pour actualiser le solde après validation. Les différentes méthodes utilisent des requêtes SQL et des objets PreparedStatement pour interagir avec la base de données.

2.3.1. Méthode add(Holiday e)

Étape 1 : Vérifie le solde de congé de l'employé dans la base de données avec une requête SQL.

Étape 2 : Calcule le nombre de jours entre les dates de début et de fin du congé demandé.

Étape 3 : Compare le nombre de jours demandés avec le solde disponible. Si le solde est insuffisant, affiche un message d'erreur et arrête le processus.

Étape 4 : Si le solde est suffisant, insère une nouvelle ligne dans la table holiday avec les détails du congé.

Étape 5 : Met à jour le solde de congé de l'employé dans la table employe.

Code :

```

1 package DAO;
2
3 import Model.Holiday;
4 import Model.HolidayType;
5 import java.sql.Date;
6 import java.sql.PreparedStatement;
7 import java.sql.ResultSet;
8 import java.sql.SQLException;
9 import java.util.ArrayList;
10 import java.util.List;
11
12 public class HolidayDAOImpl implements GenericDAO<Holiday> {
13
14@ Override
15 public void add(Holiday e) { // Ajouter une demande de congé
16     String checkSoldeSql = "SELECT solde FROM employe WHERE id = ?";
17     String insertHolidaySql = "INSERT INTO holiday (id_employe, startdate, enddate, type) VALUES (?, ?, ?, ?)";
18
19     try (PreparedStatement checkStmt = DBConnexion.getConnexion().prepareStatement(checkSoldeSql)) {
20
21         // Vérifier le solde de congé de l'employé
22         checkStmt.setInt(1, e.getId_employe());
23         ResultSet rs = checkStmt.executeQuery();
24
25         if (rs.next()) {
26             int solde = rs.getInt("solde");
27
28             // Calculer le nombre de jours demandés
29             long daysBetween = java.time.temporal.ChronoUnit.DAYS.between(
30                 e.getStartDate().toLocalDate(),
31                 e.getEndDate().toLocalDate()
32             );
33
34             // Vérification du solde suffisant
35             if (daysBetween > solde) {
36                 System.err.println("Le solde de congé est insuffisant.");
37                 return;
38             }
39
40             // Insérer la demande de congé
41             try (PreparedStatement insertStmt = DBConnexion.getConnexion().prepareStatement(insertHolidaySql)) {
42                 insertStmt.setInt(1, e.getId_employe());
43                 insertStmt.setDate(2, e.getStartDate());
44                 insertStmt.setDate(3, e.getEndDate());
45                 insertStmt.setString(4, e.getType().name());
46                 insertStmt.executeUpdate();
47
48             // Mettre à jour le solde de congé
49             String updateSoldeSql = "UPDATE employe SET solde= solde - ? WHERE id = ?";
50             try (PreparedStatement updateStmt = DBConnexion.getConnexion().prepareStatement(updateSoldeSql)) {
51                 updateStmt.setInt(1, (int) daysBetween);
52                 updateStmt.setInt(2, e.getId_employe());
53                 updateStmt.executeUpdate();
54             }
55         } else {
56             System.err.println("Employé introuvable.");
57         }
58     } catch (SQLException exception) {
59         exception.printStackTrace();
60     }
61 }
62

```

2.3.2. Méthode delete(int id)

Étape 1 : Prépare une requête SQL pour supprimer une ligne de la table holiday en fonction de l'ID.

Étape 2 : Exécute la requête pour supprimer la demande de congé correspondante.

Étape 3 : Gère les éventuelles erreurs SQL et affiche un message si la suppression échoue.

Code :

```

62
63@ Override
64 // Supprimer une demande de congé par ID
65 public void delete(int id) {
66     String sql = "DELETE FROM holiday WHERE id = ?";
67     try (PreparedStatement stmt = DBConnexion.getConnexion().prepareStatement(sql)) {
68         stmt.setInt(1, id);
69         stmt.executeUpdate();
70     } catch (SQLException exception) {
71         System.err.println("Échec de la suppression du congé");
72     }
73 }
74

```

2.3.3. Méthode update(Holiday e)

Étape 1 : Prépare une requête SQL pour mettre à jour une ligne existante dans la table holiday.

Étape 2 : Passe les nouveaux paramètres (ID de l'employé, dates de début/fin, type de congé) à la requête SQL.

Étape 3 : Exécute la requête pour appliquer les modifications dans la base de données.

Étape 4 : Gère les erreurs SQL et affiche un message si la mise à jour échoue.

Code :

```
/4
75@ Override
76 // Mettre à jour une demande de congé
77 public void update(Holiday e) {
78     String sql = "UPDATE holiday SET id_employe = ?, startdate = ?, enddate = ?, type = ? WHERE id = ?";
79     try (PreparedStatement stmt = DBConnexion.getConnexion().prepareStatement(sql)) {
80         stmt.setInt(1, e.getId_employe());
81         stmt.setDate(2, e.getStartDate());
82         stmt.setDate(3, e.getEndDate());
83         stmt.setString(4, e.getType().name());
84         stmt.setInt(5, e.getId_holiday());
85         stmt.executeUpdate();
86     } catch (SQLException exception) {
87         System.err.println("Échec de la mise à jour du congé");
88     }
89 }
```

2.3.4. Méthode display()

Étape 1 : Prépare une requête SQL pour récupérer toutes les lignes de la table holiday.

Étape 2 : Exécute la requête et parcourt les résultats rentrés (chaque ligne correspond à une demande de congé).

Étape 3 : Pour chaque ligne, récupère les colonnes (id, id_employe, startdate, enddate, type) et crée un objet Holiday.

Étape 4 : Ajoute chaque objet Holiday à une liste.

Étape 5 : Retourne la liste contenant toutes les demandes de congé ou affiche un message d'erreur en cas de problème.

Code :

```

9
@Override
// Afficher toutes les demandes de congé
public List<Holiday> display() { // Afficher toutes les demandes de congé
    String sql = "SELECT * FROM holiday";
    List<Holiday> Holidays = new ArrayList<>();
    try (PreparedStatement stmt = DBConnexion.getConnexion().prepareStatement(sql)) {
        ResultSet re = stmt.executeQuery();
        while (re.next()) {
            // Récupérer les données de chaque congé
            int id = re.getInt("id");
            int id_employe = re.getInt("id_employe");
            Date startdate = re.getDate("startdate");
            Date enddate = re.getDate("enddate");
            String type = re.getString("type");

            // Créer un objet Holiday et l'ajouter à la liste
            Holiday e = new Holiday(id, id_employe, startdate, enddate, HolidayType.valueOf(type));
            Holidays.add(e);
        }
    } catch (SQLException ex) {
        System.err.println("Échec de la récupération des congés: " + ex.getMessage());
    }
    return Holidays; // Retourner la liste des congés
}

```

2.4. Modification de EmployeDAOImpl:

La classe EmployeDAOImpl implémente l'interface GenericDAO<Employe> pour gérer les opérations liées aux employés, notamment l'ajout et la mise à jour du solde de congé.

- **Ajout du solde :** Lors de l'ajout d'un employé via la méthode add, un solde initial est assigné dans la base de données lors de l'insertion des informations de l'employé.

Code :

```

1 package DAO;
2 import Model.Employe;
10
11 public class EmployeDAOImpl implements GenericDAO<Employe> {
12
13@Override
14 public void add(Employe e) {
15     String sql = "INSERT INTO employe (nom, prenom, email, telephone, salaire, role, poste, solde) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
16     try (PreparedStatement stmt = DBConnexion.getConnexion().prepareStatement(sql)) {
17         stmt.setString(1, e.getNom());
18         stmt.setString(2, e.getPrenom());
19         stmt.setString(3, e.getEmail());
20         stmt.setString(4, e.getTelephone());
21         stmt.setDouble(5, e.getSalaire());
22         stmt.setString(6, e.getRole().name());
23         stmt.setString(7, e.getPoste().name());
24         stmt.setInt(8, e.getSolde());
25         stmt.executeUpdate();
26     } catch (SQLException exception) {
27         System.err.println("Failed to add employee: " + exception.getMessage());
28         exception.printStackTrace();
29     }
30 }
31
32@Override
33 public void delete(int id) {
34     String sql = "DELETE FROM employe WHERE id = ?";
35     try (PreparedStatement stmt = DBConnexion.getConnexion().prepareStatement(sql)) {
36         stmt.setInt(1, id);
37         stmt.executeUpdate();
38     } catch (SQLException exception) {
39         System.err.println("Failed to delete employee");
40     }
41 }
42

```

```

43@  Override
44  public void update(Employe e) {
45      String sql = "UPDATE employe SET nom = ?, prenom = ?, email = ?, telephone = ?, salaire = ?, role = ?, poste = ? WHERE id = ?";
46      try (PreparedStatement stmt = DBConnexion.getConnexion().prepareStatement(sql)) {
47          stmt.setString(1, e.getNom());
48          stmt.setString(2, e.getPrenom());
49          stmt.setString(3, e.getEmail());
50          stmt.setString(4, e.getTelephone());
51          stmt.setDouble(5, e.getSalaire());
52          stmt.setString(6, e.getRole().name());
53          stmt.setString(7, e.getPoste().name());
54          stmt.setInt(8, e.getId());
55          stmt.executeUpdate();
56      } catch (SQLException exception) {
57          System.err.println("failed of update employe");
58      }
59  }
60@ Override
61  public List<Employe> display() {
62      String sql = "SELECT * FROM employe";
63      List<Employe> Employes = new ArrayList<>();
64      try (PreparedStatement stmt = DBConnexion.getConnexion().prepareStatement(sql)) {
65          ResultSet re = stmt.executeQuery();
66          while (re.next()) {
67              int id = re.getInt("id");
68              String nom = re.getString("nom");
69              String prenom = re.getString("prenom");
70              String email = re.getString("email");
71              String telephone = re.getString("telephone");
72              double salaire = re.getDouble("salaire");
73              String role = re.getString("role");
74              String poste = re.getString("poste");
75              int solde = re.getInt("solde");
76              Employe e = new Employe(id,nom, prenom, email, telephone, salaire, Role.valueOf(role), Poste.valueOf(poste),solde);
77              Employes.add(e);
78          }
79          return Employes;
80      } catch (SQLException ex) {
81          System.err.println("failed of display employe");
82          return null;
83      }
84  }

```

- Méthode updateSolde :** Cette méthode permet de mettre à jour le solde de congé d'un employé en fonction de son identifiant (id). Elle prend en paramètre l'identifiant et le nouveau solde, puis exécute une requête SQL pour appliquer la mise à jour.

```

86
87@ public void updateSolde(int id, int solde) {
88    String sql = "UPDATE employe SET solde = ? WHERE id = ?";
89    try (PreparedStatement stmt = DBConnexion.getConnexion().prepareStatement(sql)) {
90        stmt.setInt(1, solde);
91        stmt.setInt(2, id);
92        stmt.executeUpdate();
93    } catch (SQLException exception) {
94        System.err.println("failed of update solde employe");
95    }
96  }
97
98 }

```

3. 3ème étape : Logique métier:

3.1. Classe HolidayModel

La classe HolidayModel gère la logique métier des congés des employés. Elle s'assure que les règles de gestion sont respectées avant toute action concernant les congés, notamment l'ajout, la suppression et la mise à jour des congés. Elle interagit avec la base de données via la classe HolidayDAOImpl pour effectuer les opérations CRUD, tout en appliquant des contrôles métier;

Ajout de congé (addHoliday) :

- Vérifie que la date de début n'est pas après la date de fin.
- Vérifie que la date de début n'est pas égale à la date de fin.

- Vérifie que les dates ne sont pas dans le passé.
- Vérifie que le nombre de jours du congé ne dépasse pas le solde de congé de l'employé.
- Si ces conditions sont remplies, le solde de l'employé est mis à jour et le congé est ajouté dans la base de données.

Code :

```

1 package Model;
2 import Controller.EmployeController;
3 import java.util.List;
4
5 import DAO.HolidayDAOImpl;
6 import java.sql.Date;
7
8 public class HolidayModel {
9     private HolidayDAOImpl dao;
10
11    public HolidayModel(HolidayDAOImpl dao) {
12        this.dao = dao;
13    }
14
15    // Ajoute un congé après avoir vérifié les dates et le solde de l'employé
16    public boolean addHoliday(int id, int id_employe, Date startdate, Date enddate, HolidayType type, Employe targetEmploye) {
17        if (startdate.after(enddate)) return false;
18        if (startdate.equals(enddate)) return false;
19        if (startdate.before(new Date(System.currentTimeMillis()))) return false;
20        if (enddate.before(new Date(System.currentTimeMillis()))) return false;
21
22        long daysBetween = (enddate.toLocalDate().toEpochDay() - startdate.toLocalDate().toEpochDay());
23        if (daysBetween > targetEmploye.getSolde()) return false;
24
25        EmployeController.updateSolde(targetEmploye.getId(), targetEmploye.getSolde() - (int) daysBetween);
26        Holiday e = new Holiday(id, id_employe, startdate, enddate, type);
27        dao.add(e);
28        return true;
29    }
30

```

Suppression de congé et affichage :

Code :

```

30
31    // Affiche la liste des congés
32    public List<Holiday> displayHoliday() {
33        return dao.display();
34    }
35
36    // Supprime un congé
37    public boolean deleteHoliday(int id) {
38        dao.delete(id);
39        return true;
40    }
41

```

Mise à jour de congé (updateHoliday) :

- Vérifie les mêmes conditions que pour l'ajout de congé (dates valides, solde suffisant..)
- Permet de modifier un congé existant en mettant à jour ses dates et en recalculant le solde de l'employé en fonction du nombre de jours de congé.
- Vérifie que le solde total après mise à jour (ancien solde + solde restant) est suffisant pour couvrir la durée du congé modifié.

Code :

```
41 // Met à jour un congé avec validation des nouvelles données
42 public boolean updateHoliday(int id, int id_employe, Date startdate, Date enddate, HolidayType type, Employe targetEmploye, int olldaysbetween) {
43     long daysBetween = (enddate.toLocalDate().toEpochDay() - startdate.toLocalDate().toEpochDay());
44
45     if (startdate.after(enddate)) return false;
46     if (startdate.equals(enddate)) return false;
47     if (startdate.before(new Date(System.currentTimeMillis()))) return false;
48     if (enddate.before(new Date(System.currentTimeMillis()))) return false;
49
50     if (daysBetween > (targetEmploye.getSolde() + olldaysbetween)) return false;
51     EmployeController.updateSolde(targetEmploye.getId(), (targetEmploye.getSolde() + olldaysbetween) - (int) daysBetween);
52
53     Holiday e = new Holiday(id, id_employe, startdate, enddate, type);
54     dao.update(e);
55     return true;
56 }
57
58 }
```

3.2. Classe EmployeModel

La classe EmployeModel gère les opérations relatives aux employés, telles que l'ajout, la suppression, la mise à jour des informations, et la gestion du solde de congé. Elle communique avec la base de données via la classe EmployeDAOImpl. Ici, je vais me concentrer sur la gestion du solde de congé des employés.

Ajout d'employé (addEmploye) :

- Lors de l'ajout d'un nouvel employé, le solde de congé est spécifié dans les paramètres. Le solde est directement affecté au nouvel employé au moment de sa création.
- Aucune logique spécifique n'est effectuée sur le solde dans cette méthode, car il est simplement attribué lors de la création de l'employé.

Mise à jour du solde de congé (updateSolde) :

- Cette méthode permet de mettre à jour le solde de congé d'un employé.
- Le solde est directement modifié dans la base de données via la méthode dao.updateSolde(id, solde).
- Il n'y a pas de validation sur la nouvelle valeur du solde dans ce code, ce qui signifie qu'il est supposé que cette mise à jour respecte les règles métier définies ailleurs (par exemple, vérifier que le solde ne devient pas négatif).

Code :

```

1 package Model;
2
3 import DAO.EmployeDAOImpl;
4 import java.util.List;
5
6 public class EmployeModel {
7     private EmployeDAOImpl dao;
8
9     public EmployeModel(EmployeDAOImpl dao) {
10         this.dao = dao;
11     }
12
13     // Ajoute un employé après validation des données
14     public boolean addEmploye(int id, String nom, String prenom, String email, String telephone, double salaire, Role role, Poste poste, int solde) {
15         if (salaire < 0) {
16             System.out.println("Erreur : le salaire doit être positif.");
17             return false;
18         }
19         if (id < 0) {
20             System.out.println("Erreur : l'id doit être positif.");
21             return false;
22         }
23         if (telephone.length() != 10) {
24             System.out.println("Erreur : le téléphone doit contenir 10 chiffres.");
25             return false;
26         }
27         if (!email.contains("@")) {
28             System.out.println("Erreur : l'email doit contenir '@'.");
29             return false;
30         }
31
32         Employe e = new Employe(id, nom, prenom, email, telephone, salaire, role, poste, solde);
33         dao.add(e);
34
35         return true;
36     }
37
38     // Supprime un employé par son ID
39     public boolean deleteEmploye(int id) {
40         dao.delete(id);
41         return true;
42     }
43
44     // Met à jour les informations d'un employé
45     public boolean updateEmploye(int id, String nom, String prenom, String email, String telephone, double salaire, Role role, Poste poste, int solde) {
46         Employe e = new Employe(id, nom, prenom, email, telephone, salaire, role, poste, solde);
47         dao.update(e);
48         return true;
49     }
50
51     // Met à jour le solde d'un employé
52     public boolean updateSolde(int id, int solde) {
53         dao.updateSolde(id, solde);
54         return true;
55     }
56
57     // Récupère et retourne la liste des employés
58     public List<Employe> displayEmploye() {
59         List<Employe> Employes = dao.display();
60         return Employes;
61     }
62 }

```

4. 4ème étape : Couche View - Interface graphique

Cette interface graphique, développée à l'aide de JTabbedPane, permet la gestion des employés et de leurs congés. L'application est divisée en deux onglets principaux :

Onglet "Employé" :

- Permet d'ajouter, de modifier, de supprimer et d'afficher les informations des employés (nom, prénom, email, téléphone, salaire, rôle, poste).
- Un tableau affiche les employés existants, et un formulaire permet de saisir ou de modifier les informations d'un employé.

Onglet "Congé" :

- Permet de gérer les congés des employés, incluant l'ajout, la modification, la suppression et l'affichage des congés.

- Un tableau affiche les congés existants, et un formulaire permet de saisir les informations concernant un congé (employé, date de début, date de fin, type de congé).

Code:

```

1 package View;
2
3 import DAO.EmployeDAOImpl;
4 import Model.Employe;
5 import Model.EmployeModel;
6 import Model.Poste;
7 import Model.Role;
8 import Model.HolidayType;
9 import javax.swing.*;
10 import javax.swing.table.DefaultTableModel;
11 import java.awt.*;
12 import java.util.List;
13
14 public class MainView extends JFrame {
15
16     // Déclaration des composants pour l'interface utilisateur
17     private JTabbedPane tabbedPane = new JTabbedPane();
18
19     // Les panneaux pour chaque onglet
20     private JPanel employeTab = new JPanel(); // Onglet pour afficher les employés
21     private JPanel holidayTab = new JPanel(); // Onglet pour afficher les congés
22
23     // Les panneaux pour les formulaires et les affichages
24     private JPanel Employepan = new JPanel();
25     private JPanel Holidaypan = new JPanel();
26     private JPanel Display_Table_employe = new JPanel();
27     private JPanel Display_Table_holiday = new JPanel();
28     private final JPanel Forme_employe = new JPanel();
29     private final JPanel Forme_holiday = new JPanel();
30     private JPanel panButton_employe = new JPanel();
31     private JPanel panButton_holiday = new JPanel();
32
33     // Labels pour le formulaire de l'employé
34     private JLabel label_nom = new JLabel("Nom");
35     private JLabel label_prenom = new JLabel("Prenom");
36     private JLabel label_email = new JLabel("Email");
37     private JLabel label_tele = new JLabel("Telephone");
38     private JLabel label_salaire = new JLabel("Salaire");
39     private JLabel label_role = new JLabel("Role");
40     private JLabel label_poste = new JLabel("Poste");
41
42     ...
43
44 }
```

```

41 // Labels pour le formulaire de congé
42 private JLabel label_employe = new JLabel("Nom de l'employé");
43 private JLabel label_startDate = new JLabel("Date de début (YYYY-MM-DD)");
44 private JLabel label_endDate = new JLabel("Date de fin (YYYY-MM-DD)");
45 private JLabel label_type = new JLabel("Type");
46 private JComboBox<HolidayType> TypeComboBox = new JComboBox<>(HolidayType.values());
47
48 // TextFields pour le formulaire de l'employé
49 private JTextField text_nom = new JTextField();
50 private JTextField text_prenom = new JTextField();
51 private JTextField text_email = new JTextField();
52 private JTextField text_tele = new JTextField();
53 private JTextField text_salaire = new JTextField();
54
55 // ComboBox pour le rôle et le poste de l'employé
56 private JComboBox<Role> roleComboBox = new JComboBox<>(Role.values());
57 private JComboBox<Poste> posteComboBox = new JComboBox<>(Poste.values());
58
59 // TextFields pour le formulaire de congé
60 private JComboBox<String> text_employe = new JComboBox<>();
61 private JTextField text_startDate = new JTextField("");
62 private JTextField text_endDate = new JTextField("");
63
64 // Boutons pour les actions des employés
65 private JButton addButton_employe = new JButton("Ajouter");
66 private JButton updateButton_employe = new JButton("Modifier");
67 private JButton deleteButton_employe = new JButton("Supprimer");
68 private JButton displayButton_employe = new JButton("Afficher");
69
70 // Boutons pour les actions des congés
71 private JButton addButton_holiday = new JButton("Ajouter");
72 private JButton updateButton_holiday = new JButton("Modifier");
73 private JButton deleteButton_holiday = new JButton("Supprimer");
74 private JButton displayButton_holiday = new JButton("Afficher");
75
76
77 // Tableau des employés
78 JPanel pan0 = new JPanel(new BorderLayout());
79 public static String[] columnNames_employe = {"ID", "Nom", "Prenom", "Email", "Telephone", "Salaire", "Role", "Poste", "Solde"};
80 public static DefaultTableModel tableModel = new DefaultTableModel(columnNames_employe, 0);
81 public static JTable Tableau = new JTable(tableModel);
82
83 // Tableau des congés
84 JPanel pan1 = new JPanel(new BorderLayout());
85 public static String[] columnNames_holiday = {"ID", "nom_employe", "date_debut", "date_fin", "type"};
86 public static DefaultTableModel tableModel1 = new DefaultTableModel(columnNames_holiday, 0);
87 public static JTable Tableau1 = new JTable(tableModel1);
88

```

```

```
89 public MainView() {
90 setTitle("Gestion des employés et des congés");
91 setSize(1000, 600);
92 setDefaultCloseOperation(EXIT_ON_CLOSE);
93 setLocationRelativeTo(null);
94
95 add(tabbedPane);
96
97 // Configuration de l'onglet Employé
98 employeTab.setLayout(new BorderLayout());
99 employeTab.add(EmployePanel, BorderLayout.CENTER);
100
101 EmployePanel.setLayout(new BorderLayout());
102 EmployePanel.add(DisplayTableEmploye, BorderLayout.CENTER);
103 Tableau.setFillsViewportHeight(true);
104 Dimension preferredSize = new Dimension(900, 500);
105 Tableau.setPreferredSize(preferredSize);
106 pan0.add(new JScrollPane(Tableau), BorderLayout.CENTER);
107 DisplayTableEmploye.add(pan0);
108
109 EmployePanel.add(buttonPanelEmploye, BorderLayout.SOUTH);
110 buttonPanelEmploye.add(addButtonEmploye);
111 buttonPanelEmploye.add(updateButtonEmploye);
112 buttonPanelEmploye.add(deleteButtonEmploye);
113 buttonPanelEmploye.add(displayButtonEmploye);
114
115 EmployePanel.add(FormeEmploye, BorderLayout.NORTH);
116 FormeEmploye.setLayout(new GridLayout(7, 2, 10, 10));
117 FormeEmploye.add(labelNom);
118 FormeEmploye.add(textNom);
119 FormeEmploye.add(labelPrenom);
120 FormeEmploye.add(textPrenom);
121 FormeEmploye.add(labelEmail);
122 FormeEmploye.add(textEmail);
123 FormeEmploye.add(labelTele);
124 FormeEmploye.add(textTele);
125 FormeEmploye.add(labelSalaire);
126 FormeEmploye.add(textSalaire);
127 FormeEmploye.add(labelRole);
128 FormeEmploye.add(roleComboBox);
129 FormeEmploye.add(labelPoste);
130 FormeEmploye.add(posteComboBox);
```

```

```

131
132     // Configuration de l'onglet Congé
133     holidayTab.setLayout(new BorderLayout());
134     holidayTab.add(Holidaypan, BorderLayout.CENTER);
135     Holidaypan.setLayout(new BorderLayout());
136     Holidaypan.add(Display_Table_holiday, BorderLayout.CENTER);
137
138     Tableau1.setFillsViewportHeight(true);
139     Tableau1.setPreferredScrollableViewportSize(preferredSize);
140     pan1.add(new JScrollPane(Tableau1), BorderLayout.CENTER);
141     Display_Table_holiday.add(pan1);
142
143     Holidaypan.add(Forme_holiday, BorderLayout.NORTH);
144     Forme_holiday.setLayout(new GridLayout(4, 2, 10, 10));
145     Forme_holiday.add(label_employe);
146     Forme_holiday.add(text_employe);
147     Forme_holiday.add(label_startDate);
148     Forme_holiday.add(text_startDate);
149     Forme_holiday.add(label_endDate);
150     Forme_holiday.add(text_endDate);
151     Forme_holiday.add(label_type);
152     Forme_holiday.add(TypeComboBox);
153
154     Holidaypan.add(panButton_holiday, BorderLayout.SOUTH);
155     panButton_holiday.add(addButton_holiday);
156     panButton_holiday.add(updateButton_holiday);
157     panButton_holiday.add(deleteButton_holiday);
158     panButton_holiday.add(displayButton_holiday);
159
160     // Ajouter les onglets au TabbedPane
161     tabbedPane.addTab("Employe", employeTab);
162     tabbedPane.addTab("Holiday", holidayTab);
163
164     // Remplir la liste des employés au démarrage
165     remplaire_les_employes();
166     setVisible(true);
167 }
168

```

La méthode **remplaire_les_employes** remplit un ComboBox avec la liste des employés, affichant leur ID et nom, pour permettre à l'utilisateur de choisir un employé parmi ceux enregistrés.

Code:

```

168
169     // Méthode pour remplir la liste des employés dans le ComboBox
170     public void remplaire_les_employes () {
171         List<Employe> Employes = new EmployeModel(new EmployeDAOImpl()).displayEmploye();
172         text_employe.removeAllItems();
173         for (Employe elem : Employes) {
174             text_employe.addItem(elem.getId() + " - " + elem.getNom()+" "+elem.getPrenom());
175         }
176     }
177

```

```

1// 
178 // Getters pour récupérer les informations des champs
179 public int getId_employe() { return Integer.parseInt(text_employe.getSelectedItem().toString().split(" - ")[0]); }
180 public String getNom() { return text_nom.getText(); }
181 public JTable getTable() { return (JTable) Display_Table_employe.getComponent(0); }
182 public String getPrenom() { return text_prenom.getText(); }
183 public String getEmail() { return text_email.getText(); }
184 public String getTelephone() { return text_tele.getText(); }
185 public double getSalaire() { return Double.parseDouble(text_salaire.getText()); }
186 public Role getRole() { return (Role) roleComboBox.getSelectedItem(); }
187 public Poste getPoste() { return (Poste) posteComboBox.getSelectedItem(); }
188
189 // Getters pour les boutons de l'employé
190 public JButton get addButton_employe() { return addButton_employe; }
191 public JButton get updateButton_employe() { return updateButton_employe; }
192 public JButton get deleteButton_employe() { return deleteButton_employe; }
193 public JButton get displayButton_employe() { return displayButton_employe; }
194
195 // Getters pour les boutons du congé
196 public JButton get addButton_holiday() { return addButton_holiday; }
197 public JButton get updateButton_holiday() { return updateButton_holiday; }
198 public JButton get deleteButton_holiday() { return deleteButton_holiday; }
199 public JButton get displayButton_holiday() { return displayButton_holiday; }
200

```

Ces méthodes sont utilisées pour afficher des messages d'erreur ou de succès à l'utilisateur via des boîtes de dialogue. Cela permet à l'utilisateur de savoir si une action a réussi ou échoué.

Code:

```

// methods d'affichage des messages
public void afficherMessageErreur(String message) {
    JOptionPane.showMessageDialog(this, message, "Erreur", JOptionPane.ERROR_MESSAGE);
}

public void afficherMessageSucces(String message) {
    JOptionPane.showMessageDialog(this, message, "Succes", JOptionPane.INFORMATION_MESSAGE);
}

```

Ces méthodes sont utilisées pour réinitialiser les champs d'un formulaire lorsqu'un utilisateur termine une action, comme l'ajout ou la modification d'un employé ou d'un congé. Elles permettent de préparer le formulaire pour une nouvelle entrée.

Code:

```

268
269 // méthodes de vider les champs
270 public void viderChamps_em() {
271     text_nom.setText("");
272     text_prenom.setText("");
273     text_email.setText("");
274     text_tele.setText("");
275     text_salaire.setText("");
276     roleComboBox.setSelectedIndex(0);
277     posteComboBox.setSelectedIndex(0);
278 }
279
280 public void viderChamps_ho() {
281     text_startDate.setText("");
282     text_endDate.setText("");
283     TypeComboBox.setSelectedIndex(0);
284 }
285

```

Ces méthodes sont utilisées pour pré-remplir les champs d'un formulaire avec des données spécifiques (comme les informations d'un employé ou d'un congé). Elles sont utiles lors de la modification des informations existantes, afin d'éviter que l'utilisateur ait à entrer à nouveau toutes les données.

Code:

```

~~~
286     // méthodes de remplir les champs
287     public void remplaireChamps_en (int id, String nom, String prenom, String email, String telephone, double salaire, Role role, Poste poste) {
288         text_nom.setText(nom);
289         text_prenom.setText(prenom);
290         text_email.setText(email);
291         text_tele.setText(telephone);
292         text_salaire.setText(String.valueOf(salaire));
293         roleComboBox.setSelectedItem(role);
294         posteComboBox.setSelectedItem(poste);
295     }
296
297     public void remplaireChamps_ho(int id_employe, String date_debut, String date_fin, HolidayType type) {
298         List<Employe> Employes = new EmployeModel(new EmployeDAOImpl()).displayEmploye();
299         text_employe.removeAllItems();
300         for (Employe elem : Employes) {
301             if (elem.getId() == id_employe) {
302                 text_employe.addItem(elem.getId() + " - " + elem.getNom() + " " + elem.getPrenom());
303                 text_employe.setSelectedItem(elem.getId() + " - " + elem.getNom() + " " + elem.getPrenom());
304             }
305         }
306         text_startDate.setText(date_debut);
307         text_endDate.setText(date_fin);
308         TypeComboBox.setSelectedItem(type);
309     }
310

```

Ces méthodes sont utilisées pour vérifier si des champs importants d'un formulaire sont vides avant de permettre la soumission ou l'enregistrement des données.

Code:

```

310
311     // méthodes de test des champs
312     public boolean testChampsVide_em (){
313         return text_nom.getText().equals("") || text_prenom.getText().equals("")
314             || text_email.getText().equals("") || text_tele.getText().equals("")
315             || text_salaire.getText().equals("");
316     }
317
318     public boolean testChampsVide_ho () {
319         return text_employe.getSelectedItem().equals("") || text_startDate.getText().equals("")
320             || text_endDate.getText().equals("") || TypeComboBox.getSelectedItem().equals("");
321     }
322 }
323

```

5. 5ème étape : Couche Controller - Implémentation du Contrôleur

5.1. HolidayController:

- Ajouter un congé :** Vérifie les informations (employé, dates, solde) et ajoute un congé si tout est valide.
- Afficher les congés :** Affiche la liste des congés dans un tableau.
- Supprimer un congé :** Supprime un congé sélectionné et met à jour le solde de congé de l'employé.
- Mettre à jour un congé :** Permet de modifier un congé sélectionné après validation des nouvelles informations.

- **Sélectionner un congé pour mise à jour :** Affiche les détails du congé sélectionné pour permettre la modification.

Code :

```

1 package Controller;
2
3 import DAO.EmployeDAOImpl;
4 import Model.*;
5 import View.*;
6 import java.sql.Date;
7 import java.util.List;
8 import javax.swing.table.DefaultTableModel;
9
10 public class HolidayController {
11
12     private final MainView View;
13     public HolidayModel model_holiday;
14     public static int id = 0;
15     public static int oldselectedrow = -1;
16     public static boolean test = false;
17     int id_employe = 0;
18     String nom_employe = "";
19     public static String OldstartDate = null;
20     public static String OldendDate = null;
21     HolidayType type = null;
22     int oldsolde = 0;
23     int solde = 0;
24     boolean updatereussi = false;
25     Employe targetEmploye = null;
26
27     public HolidayController(MainView view, HolidayModel model) {
28         this.View = view;
29         this.model_holiday = model;
30
31         // Ajout des écouteurs d'événements pour les boutons
32         View.getdeleteButton_holiday().addActionListener(e -> deleteHoliday());
33         View.getupdateButton_holiday().addActionListener(e -> updateHoliday());
34         MainView.Tableau1.getSelectionModel().addListSelectionListener(e -> updateHolidaybyselect());
35         View.get addButton_holiday().addActionListener(e -> addHoliday());
36         View.getdisplayButton_holiday().addActionListener(e -> displayHoliday());
37     }
38

```

```

38 // Méthode pour ajouter un congé
39 // Cette méthode vérifie la validité des dates, le solde et l'absence de chevauchement avant d'ajouter un congé
40 private void addHoliday() {
41     int id_employe = View.getId_employe();
42     Date startDate = Date.valueOf(View.getStartDate());
43     Date endDate = Date.valueOf(View.getEndDate());
44     HolidayType type = View.getHolidayType();
45
46     View.viderChamps_ho(); // Vide les champs de saisie
47
48     Employe targetEmploye = null;
49
50     // Recherche de l'employé correspondant à l'ID
51     for (Employe employe : new EmployeModel(new EmployeDAOImpl()).displayEmploye()) {
52         if (employe.getId() == id_employe) {
53             targetEmploye = employe;
54             break;
55         }
56     }
57
58     if (targetEmploye == null) {
59         View.afficherMessageErreur("Cet employé n'existe pas.");
60         return;
61     }
62
63     // Calcul de la durée du congé
64     long daysBetween = java.time.temporal.ChronoUnit.DAYS.between(
65         startDate.toLocalDate(),
66         endDate.toLocalDate()
67     );
68
69     if (daysBetween <= 0) {
70         View.afficherMessageErreur("Les dates de début et de fin sont invalides.");
71         return;
72     }
73
74     // Vérification du solde de congé
75     if (targetEmploye.getSolde() < daysBetween) {
76         View.afficherMessageErreur("Le solde de congé de l'employé est insuffisant.");
77         return;
78     }
79 }
80
80 // Vérification des chevauchements des congés existants
81 for (Holiday existingHoliday : model_holiday.displayHoliday()) {
82     if (existingHoliday.getId_employe() == id_employe) {
83         Date existingStartDate = existingHoliday.getStartDate();
84         Date existingEndDate = existingHoliday.getEndDate();
85
86         // Vérification du chevauchement des dates
87         if ((startDate.before(existingEndDate) && endDate.after(existingStartDate))) {
88             View.afficherMessageErreur("Le congé se chevauche avec une autre période de congé.");
89             return;
90         }
91     }
92 }
93
94 try {
95     // Ajout du congé
96     boolean addReussi = model_holiday.addHoliday(0, id_employe, startDate, endDate, type, targetEmploye);
97
98     if (addReussi) {
99         // Mise à jour du solde de congé après ajout
100        targetEmploye.setSolde(targetEmploye.getSolde() - (int) daysBetween);
101        View.afficherMessageSucces("Holiday est ajoutée.");
102    }
103 } catch (Exception e) {
104     e.printStackTrace();
105     View.afficherMessageErreur("Erreur lors de l'ajout : " + e.getMessage());
106 }
107
108 }
109

```

```

109
110 // Méthode pour afficher la liste des congés
111 // Cette méthode récupère les congés et les affiche dans le tableau
112 private void displayHoliday() {
113     List<Holiday> Holidays = model_holiday.displayHoliday();
114
115     if (Holidays == null || Holidays.isEmpty()) {
116         View.afficherMessageErreur("Aucune holiday.");
117         return; // Retourne pour éviter de continuer si la liste est vide
118     }
119
120     DefaultTableModel tableModel1 = (DefaultTableModel) MainView.Tableau1.getModel();
121     tableModel1.setRowCount(0); // Efface les lignes existantes dans le tableau
122
123     for (Holiday e : Holidays) {
124         String nom_employe = null;
125         List<Employe> Employes = new EmployeModel(new EmployeDAOImpl()).displayEmploye();
126         for (Employe em : Employes) {
127             if (em.getId() == e.getId_employe()) {
128                 nom_employe = em.getId() + " - " + em.getNom() + " " + em.getPrenom();
129                 break;
130             }
131         }
132         // Ajout de la ligne dans le tableau
133         tableModel1.addRow(new Object[]{e.getId_holiday(), nom_employe, e.getStartDate(), e.getEndDate(), e.getType()});
134     }
135     View.remplaire_les_employes(); // Remplir les données des employés
136 }
137
138 // Méthode pour supprimer un congé
139 // Cette méthode supprime un congé sélectionné dans le tableau
140 private void deleteHoliday() {
141     int selectedrow = MainView.Tableau1.getSelectedRow();
142     if (selectedrow == -1) {
143         View.afficherMessageErreur("Veuillez selectionner une ligne.");
144     } else {
145         int id = (int) MainView.Tableau1.getValueAt(selectedrow, 0);
146         int id_employe = Integer.parseInt((MainView.Tableau1.getValueAt(selectedrow, 1)).toString().split(" - ")[0]);
147         int olddaysbetween = (int) ((Date.valueOf(OldendDate).toLocalDate().toEpochDay() - Date.valueOf(OldstartDate).toLocalDate().toEpochDay()));
148         for (Employe e : new EmployeModel(new EmployeDAOImpl()).displayEmploye()) {
149             if (e.getId() == id_employe) {
150                 solde = e.getSolde();
151                 break;
152             }
153         }
154         EmployeController.updateSolde(id_employe, solde + olddaysbetween);
155         boolean deletereussi = model_holiday.deleteHoliday(id);
156         if (deletereussi) {
157             View.afficherMessageSucces("Holiday est supprimé.");
158             displayHoliday();
159         } else {
160             View.afficherMessageErreur("Holiday n'est pas supprimé.");
161         }
162     }
163 }
164
165 // Méthode pour mettre à jour un congé sélectionné
166 // Cette méthode permet de charger les informations du congé sélectionné pour les modifier
167 private void updateHolidaybyselect() {
168     int selectedrow = MainView.Tableau1.getSelectedRow();
169
170     if (selectedrow == -1) {
171         return;
172     }
173     try {
174         id = (int) MainView.Tableau1.getValueAt(selectedrow, 0);
175         nom_employe = (String) MainView.Tableau1.getValueAt(selectedrow, 1);
176         id_employe = Integer.parseInt(nom_employe.split(" - ")[0]);
177         OldstartDate = String.valueOf(MainView.Tableau1.getValueAt(selectedrow, 2));
178         OldendDate = String.valueOf(MainView.Tableau1.getValueAt(selectedrow, 3));
179         type = (HolidayType) MainView.Tableau1.getValueAt(selectedrow, 4);
180         View.remplaireChamps_ho(id_employe, OldstartDate, OldendDate, type);
181         test = true;
182     } catch (Exception e) {
183         View.afficherMessageErreur("Erreur lors de la recuperation des données");
184     }
185 }

```

```

186 // Méthode pour mettre à jour un congé
187 // Cette méthode vérifie les informations et effectue la mise à jour du congé
188 private void updateHoliday() {
189     if (!test) {
190         View.afficherMessageErreur("Veuillez d'abord sélectionner une ligne à modifier.");
191         return;
192     }
193     try {
194         nom_employe = View.getNom();
195         Date startDate_holiday = Date.valueOf(View.getStartDate());
196         Date endDate_holiday = Date.valueOf(View.getEndDate());
197         type = View.getHolidayType();
198         id_employe = View.getId_employe();
199
200         int olddaysbetween = (int) ( (Date.valueOf(OldendDate).toLocalDate().toEpochDay() - Date.valueOf(OldstartDate).toLocalDate().toEpochDay()));
201
202         for (Employe employe : new EmployeModel(new EmployeDAOImpl()).displayEmploye()) {
203             if (employe.getId() == id_employe) {
204                 targetEmploye = employe;
205                 break;
206             }
207         }
208     }
209     boolean updateSuccessful = model_holiday.updateHoliday(id, id_employe, startDate_holiday, endDate_holiday, type, targetEmploye, olddaysbetween);
210
211     if (updateSuccessful) {
212         test = false;
213         View.afficherMessageSucces("Holiday est modifié avec succès.");
214         displayHoliday();
215         View.viderChamps_ho(); // Réinitialiser les champs après modification
216     } else {
217         View.afficherMessageErreur("Erreur lors de la mise à jour de holiday.");
218     }
219 } catch (Exception e) {
220     View.afficherMessageErreur("Erreur lors de la mise à jour");
221 }
222 }
223 }
224 }
225

```

5.2. EmployeController:

La gestion du solde est effectuée via la fonction updateSolde et réinitialisée chaque début d'année.

- Affichage des employés :** Il récupère et affiche la liste des employés, y compris leur solde de congés.
- Ajout d'un employé :** Lorsqu'un employé est ajouté, un solde initial de 25 jours est attribué, et un message de succès ou d'erreur est affiché.
- Suppression d'un employé :** Un employé est supprimé à partir de la ligne sélectionnée dans le tableau. Un message de succès ou d'erreur est également affiché.
- Mise à jour d'un employé :** Permet de modifier les informations d'un employé, y compris son solde de congés. Le solde est mis à jour en fonction des changements apportés.
- Gestion du solde :** Lors du début de l'année, le solde de congé des employés est réinitialisé à 25 jours. Un employé peut voir et modifier son solde de congés dans l'interface.

Code :

```

1 package Controller;
2
3 import Model.*;
4 import View.*;
5
6 import java.util.Calendar;
7 import java.util.List;
8
9 import javax.swing.table.DefaultTableModel;
10
11 public class EmployeController {
12
13     private final MainView View;
14     public static EmployeModel model_employe ;
15     public static int id = 0;
16     public static int oldselectedrow = -1;
17     public static boolean test = false;
18     String nom = "";
19     String prenom = "";
20     String email = "";
21     String telephone = "";
22     double salaire = 0;
23     Role role = null;
24     Poste poste = null;
25     int solde = 0;
26     boolean updatereussi = false;
27
28     // Constructeur : Initialise le contrôleur avec la vue et le modèle, et ajoute des écouteurs pour les boutons
29     public EmployeController(MainView view, EmployeModel model) {
30         this.View = view;
31         this.model_employe = model;
32
33         View.getaddButton_employe().addActionListener(e -> addEmploye());
34         View.getdeleteButton_employe().addActionListener(e -> deleteEmploye());
35         View.getupdateButton_employe().addActionListener(e -> updateEmploye());
36         View.getdisplayButton_employe().addActionListener(e -> displayEmploye());
37         MainView.Tableau.getSelectionModel().addListSelectionListener(e -> updateEmployebyselect());
38     }
39
40
41     // Affiche tous les employés dans le tableau
42     public void displayEmploye() {
43         List<Employe> Employes = model_employe.displayEmploye();
44         if(Employes.isEmpty()){
45             View.afficherMessageErreur("Aucun employé.");
46         }
47         DefaultTableModel tableModel = (DefaultTableModel) MainView.Tableau.getModel();
48         tableModel.setRowCount(0);
49         for(Employe e : Employes){
50             tableModel.addRow(new Object[]{e.getId(), e.getNom(), e.getPrenom(), e.getEmail(), e.getTelephone(), e.getSalaire(), e.getRole(), e.getPoste(),e.getSolde()});
51         }
52         View.remplaire_les_employees();
53     }
54
55     // Ajouter un nouvel employé
56     private void addEmploye() {
57         String nom = View.getNom();
58         String prenom = View.getPrenom();
59         String email = View.getEmail();
60         String telephone = View.getTelephone();
61         double salaire = View.getSalaire();
62         Role role = View.getRole();
63         Poste poste = View.getPoste();
64
65         View.viderChamps_em();
66         boolean addreussi = model_employe.addEmploye(0,nom, prenom, email, telephone, salaire, role, poste ,25);
67
68         if(addreussi == true){
69             View.afficherMessageSucces("L'employé a bien été ajoutée.");
70             displayEmploye();
71         }else{
72             View.afficherMessageErreur("L'employé n'a pas été ajoutée.");
73         }
74     }

```

```

75 // Supprimer un employé en fonction de la ligne sélectionnée dans le tableau
76 private void deleteEmploye(){
77     int selectedrow = MainView.Tableau.getSelectedRow();
78     if(selectedrow == -1){
79         View.afficherMessageErreur("Veuillez selectionner une ligne.");
80     }else{
81         int id = (int) MainView.Tableau.getValueAt(selectedrow, 0);
82         if(model_employe.deleteEmploye(id)){
83             View.afficherMessageSucces("L'employé a bien été supprimé.");
84             displayEmploye();
85         }else{
86             View.afficherMessageErreur("L'employé n'a pas été supprimé.");
87         }
88     }
89 }
90
91 // Mettre à jour les données de l'employé en fonction de la ligne sélectionnée dans le tableau
92 private void updateEmployebyselect(){
93     int selectedrow = MainView.Tableau.getSelectedRow();
94
95     if (selectedrow == -1) {
96         return;
97     }
98     try{
99         id = (int) MainView.Tableau.getValueAt(selectedrow, 0);
100        nom = (String) MainView.Tableau.getValueAt(selectedrow, 1);
101        prenom = (String) MainView.Tableau.getValueAt(selectedrow, 2);
102        email = (String) MainView.Tableau.getValueAt(selectedrow, 3);
103        telephone = (String) MainView.Tableau.getValueAt(selectedrow, 4);
104        salaire = (double) MainView.Tableau.getValueAt(selectedrow, 5);
105        role = (Role) MainView.Tableau.getValueAt(selectedrow, 6);
106        poste = (Poste) MainView.Tableau.getValueAt(selectedrow, 7);
107        solde = (int) MainView.Tableau.getValueAt(selectedrow, 8);
108        View.remplirChamps_em(id, nom, prenom, email, telephone, salaire, role, poste);
109        test = true;
110    }catch(Exception e){
111        View.afficherMessageErreur("Erreur lors de la récupération des données");
112    }
113 }
114
115 // Mettre à jour un employé dans le modèle
116 private void updateEmploye(){
117     if (!test) {
118         View.afficherMessageErreur("Veuillez d'abord sélectionner une ligne à modifier.");
119         return;
120     }
121     try {
122         nom = View.getNom();
123         prenom = View.getPrenom();
124         email = View.getEmail();
125         telephone = View.getTelephone();
126         salaire = View.getSalaire();
127         role = View.getRole();
128         poste = View.getPoste();
129
130         boolean updateSuccessful = model_employe.updateEmploye(id, nom, prenom, email, telephone, salaire, role, poste, solde);
131
132         if (updateSuccessful) {
133             test = false;
134             View.afficherMessageSucces("L'employé a été modifié avec succès.");
135             displayEmploye();
136             View.viderChamps_em();
137         } else {
138             View.afficherMessageErreur("Erreur lors de la mise à jour de l'employé.");
139         }
140     } catch (Exception e) {
141         View.afficherMessageErreur("Erreur lors de la mise à jour");
142     }
143 }
144
145 // Réinitialiser le solde des employés au début de l'année
146 public void resetSolde(){
147     Calendar now = Calendar.getInstance();
148     if(now.get(Calendar.DAY_OF_YEAR) == 1){
149         for (Employe employe : model_employe.displayEmploye()) {
150             updateSolde(employe.getId(), 25);
151         }
152     }
153 }
154

```

```

154
155 // Mettre à jour le solde d'un employé
156 public static void updateSolde(int id , int solde){
157     boolean updateSuccessful = model_employe.updateSolde(id, solde);
158 }
159
160 }

```

6. 6ème étape : Main - Application principale

Le code de la classe Main initialise l'application en créant les objets nécessaires pour l'affichage, la gestion des employés et des congés. Il instancie la vue (MainView), les objets DAO pour l'accès aux données des employés (EmployeDAOImpl) et des congés (HolidayDAOImpl), ainsi que les modèles correspondants (EmployeModel et HolidayModel). Ensuite, il initialise les contrôleurs (EmployeController et HolidayController) pour gérer les interactions entre la vue et les modèles.

Code :

```

1 package Main;
2
3 import Controller.*;
4 import DAO.*;
5 import Model.*;
6 import View.*;
7
8 public class Main {
9
10    public static void main(String[] args) {
11        // Création de la vue principale de l'application
12        MainView view = new MainView();
13
14        // Création des objets DAO (Data Access Object) pour accéder aux données des employés et des congés
15        EmployeDAOImpl dao = new EmployeDAOImpl();
16        HolidayDAOImpl dao_holiday = new HolidayDAOImpl();
17
18        // Création des modèles, qui utilisent les DAO pour interagir avec les données
19        EmployeModel model_employe = new EmployeModel(dao);
20        HolidayModel model_holiday = new HolidayModel(dao_holiday);
21
22        // Création des contrôleurs, qui connectent la vue et les modèles
23        new EmployeController(view, model_employe);
24        new HolidayController(view, model_holiday);
25    }
26 }

```

Réalisation:

1. Exécution du projet:

ID	Nom	Prenom	Email	Telephone	Salaire	Role	Poste	Solde
1	SAISSI	Zahra	zahra@email.com	0612345678	100000.0	ADMIN	INGENIEURE_E...	9

2. Affichage des employés:

3. La page du gestion de congé:

Gestion des employés et des congés

Employe Holiday

Nom de l'employé	1 - SAISSI Zahra			
Date de début (YYYY-MM-DD)				
Date de fin (YYYY-MM-DD)				
Type	CONGE_PAYE			
ID	nom_employe	date_debut	date_fin	type

Ajouter **Modifier** **Supprimer** **Afficher**

4. Ajouter un employé:

Gestion des employés et des congés

Employe Holiday

Nom	OUHIDOUS							
Prenom	Inas							
Email	Inas@gmail.com							
Telephone	0612345678							
Salaire	25400.0							
Role	ADMIN							
Poste	PILOTE							
ID	Nom	Prenom	Email	Telephone	Salaire	Role	Poste	Solde
1	SAISSI	Zahra	zahra@email.com	0612345678	100000.0	ADMIN	INGENIEURE_E...	9
2	KOUISS	Khaoula	kouis@email.com	0612345678	25000.0	EMPLOYEE	INGENIEURE_E...	3
3	ZAHNI	jihad	jihad@email.com	0612345678	50000.0	EMPLOYEE	TEAM LEADER	15
4	jhndfkc	cfjiui	dffji@ujhc	0612345678	1000.0	ADMIN	INGENIEURE_E...	13

Ajouter **Modifier** **Supprimer** **Afficher**

Gestion des employés et des congés

Employe Holiday

Nom	
Prenom	
Email	
Telephone	
Salaire	
Role	
Poste	

Succès
L'employé a bien été ajoutée.

ID	Nom	Prenom	Email	Telephone	Salaire	Role	Poste	Solde
1	SAISSI	Zahra	zahra@email.com	0612345678	100000.0	ADMIN	INGENIEURE_E...	9
2	KOUISS	Khaoula	kouis@email.com	0612345678	25000.0	EMPLOYEE	INGENIEURE_E...	3
3	ZAHNI	jihad	jihad@email.com	0612345678	50000.0	EMPLOYEE	TEAM_LEADER	15
4	jhnfdk	cfjiuij	dfiji@ujhc	0612345678	1000.0	ADMIN	INGENIEURE_E...	13

Ajouter Modifier Supprimer Afficher

5. Afficher l'employé ajouté:

Gestion des employés et des congés

Employe Holiday

Nom	
Prenom	
Email	
Telephone	
Salaire	
Role	ADMIN
Poste	INGENIEURE_ETUDE_ET_DEVELOPPEMENT

Succès
L'employé a bien été ajoutée.

ID	Nom	Prenom	Email	Telephone	Salaire	Role	Poste	Solde
1	SAISSI	Zahra	zahra@email.com	0612345678	100000.0	ADMIN	INGENIEURE_E...	9
2	KOUISS	Khaoula	kouis@email.com	0612345678	25000.0	EMPLOYEE	INGENIEURE_E...	3
3	ZAHNI	jihad	jihad@email.com	0612345678	50000.0	EMPLOYEE	TEAM_LEADER	15
4	jhnfdk	cfjiuij	dfiji@ujhc	0612345678	1000.0	ADMIN	INGENIEURE_E...	13
8	OUHIDOUS	Inas	inas@gmail	0612345678	24000.0	ADMIN	PILOTE	25

Ajouter Modifier Supprimer Afficher

6. Ajouter un congé à l'employé:

Gestion des employés et des congés

Employe Holiday

Nom de l'employé	5 - OUHIDOUS Inas			
Date de début (YYYY-MM-DD)	2025-01-01			
Date de fin (YYYY-MM-DD)	2025-01-05			
Type	CONGE_MALADIE			
ID	nom_employe	date_debut	date_fin	type

Ajouter Modifier Supprimer Afficher

Gestion des employés et des congés

Employe Holiday

Nom de l'employé	5 - OUHIDOUS Inas			
Date de début (YYYY-MM-DD)				
Date de fin (YYYY-MM-DD)				
Type	CONGE_PAYE			
ID	nom_employe	date_debut	date_fin	type

Succès

Holiday est ajoutée.

OK

Ajouter Modifier Supprimer Afficher

7. Affichage des congés:

Gestion des employés et des congés

Employe Holiday

Nom de l'employé	1 - SAISSI Zahra			
Date de début (YYYY-MM-DD)				
Date de fin (YYYY-MM-DD)				
Type	CONGE_PAYE			
ID	nom_employé	date_debut	date_fin	type
1	1 - SAISSI Zahra	2024-12-25	2024-12-30	CONGE_NON_PAYE
2	1 - SAISSI Zahra	2025-01-01	2025-01-04	CONGE_NON_PAYE
4	3 - ZAHNI jihad	2024-12-25	2024-12-30	CONGE_MALADIE
5	1 - SAISSI Zahra	2024-12-25	2024-12-30	CONGE_PAYE
6	4 - jhndfkc cfjiiji	2025-01-24	2025-01-30	CONGE_PAYE
7	2 - KOUIS Khaoula	2025-12-24	2025-12-30	CONGE_PAYE
8	2 - KOUIS Khaoula	2025-02-01	2025-02-04	CONGE_MALADIE
13	8 - OUHIDOUS Inas	2025-01-01	2025-01-03	CONGE_PAYE

Ajouter Modifier Supprimer Afficher

8. Le Solde change après l'ajout:

Gestion des employés et des congés

Employe Holiday

Nom								
Prenom								
Email								
Telephone								
Salaire								
Role	ADMIN							
Poste	INGENIEURE_ETUDE_ET_DEVELOPPEMENT							
ID	Nom	Prenom	Email	Telephone	Salaire	Role	Poste	Solde
1	SAISSI	Zahra	zahra@email.com	0612345678	100000.0	ADMIN	INGENIEURE_E... 9	
2	KOUIS	Khaoula	kouis@email.com	0612345678	25000.0	EMPLOYE	INGENIEURE_E... 3	
3	ZAHNI	jihad	jihad@email.com	0612345678	50000.0	EMPLOYE	TEAM_LEADER 15	
4	Jhndfkc	cfjiiji	dfjji@ujhc	0612345678	1000.0	ADMIN	INGENIEURE_E... 13	
8	OUHIDOUS	Inas	inas@gmail	0612345678	24000.0	ADMIN	PILOTE 21	

Ajouter Modifier Supprimer Afficher

9. Si on ajoute un congé qui contient le même employé et la même date:

Gestion des employés et des congés

Employe		Holiday	
Nom de l'employé	8 - OUHIDOUS Inas		
Date de début (YYYY-MM-DD)	2025-01-01		
Date de fin (YYYY-MM-DD)	2025-01-03		
Type	CONGE_PAYE		
ID	nom_employe	date_debut	date_fin
1	1 - SAISSI Zahra	2024-12-25	2024-12-30
2	1 - SAISSI Zahra	2025-01-01	2025-01-04
4	3 - ZAHNI jihad	2024-12-25	2024-12-30
5	1 - SAISSI Zahra	2024-12-25	2024-12-30
6	4 - jhndfk cfiuji	2025-01-24	2025-01-30
7	2 - KOUIS Khaoula	2025-12-24	2025-12-30
8	2 - KOUIS Khaoula	2025-02-01	2025-02-04
13	8 - OUHIDOUS Inas	2025-01-01	2025-01-03

Ajouter **Modifier** **Supprimer** **Afficher**

Gestion des employés et des congés

Employe		Holiday	
Nom de l'employé	8 - OUHIDOUS Inas		
Date de début (YYYY-MM-DD)			
Date de fin (YYYY-MM-DD)			
Type	CONGE_PAYE		
ID	nom_employe	date_debut	date_fin
1	1 - SAISSI Zahra	2024-12-25	2024-12-30
2	1 - SAISSI Zahra	2025-01-01	2025-01-04
4	3 - ZAHNI jihad	2024-12-25	2024-12-30
5	1 - SAISSI Zahra	2024-12-25	2024-12-30
6	4 - jhndfk cfiuji	2025-01-24	2025-01-30
7	2 - KOUIS Khaoula	2025-12-24	2025-12-30
8	2 - KOUIS Khaoula	2025-02-01	2025-02-04
13	8 - OUHIDOUS Inas	2025-01-01	2025-01-03

X

Le congé se chevauche avec une autre période de congé.

OK

Ajouter **Modifier** **Supprimer** **Afficher**

10. La modification d'un congé:

Gestion des employés et des congés

Employe Holiday

Nom de l'employé	8 - OUHIDOUS Inas																																													
Date de début (YYYY-MM-DD)	2025-01-01																																													
Date de fin (YYYY-MM-DD)	2025-01-12																																													
Type	CONGE_PAYE																																													
<table border="1"> <thead> <tr> <th>ID</th> <th>nom_employe</th> <th>date_debut</th> <th>date_fin</th> <th>type</th> </tr> </thead> <tbody> <tr><td>1</td><td>1 - SAISSI Zahra</td><td>2024-12-25</td><td>2024-12-30</td><td>CONGE_NON_PAYE</td></tr> <tr><td>2</td><td>1 - SAISSI Zahra</td><td>2025-01-01</td><td>2025-01-04</td><td>CONGE_NON_PAYE</td></tr> <tr><td>4</td><td>3 - ZAHNI jihad</td><td>2024-12-25</td><td>2024-12-30</td><td>CONGE_MALADIE</td></tr> <tr><td>5</td><td>1 - SAISSI Zahra</td><td>2024-12-25</td><td>2024-12-30</td><td>CONGE_PAYE</td></tr> <tr><td>6</td><td>4 - ihndfkc cfjiuji</td><td>2025-01-24</td><td>2025-01-30</td><td>CONGE_PAYE</td></tr> <tr><td>7</td><td>2 - KOUIS Khaoula</td><td>2025-12-24</td><td>2025-12-30</td><td>CONGE_PAYE</td></tr> <tr><td>8</td><td>2 - KOUIS Khaoula</td><td>2025-02-01</td><td>2025-02-04</td><td>CONGE_MALADIE</td></tr> <tr><td>13</td><td>8 - OUHIDOUS Inas</td><td>2025-01-01</td><td>2025-01-03</td><td>CONGE_PAYE</td></tr> </tbody> </table>		ID	nom_employe	date_debut	date_fin	type	1	1 - SAISSI Zahra	2024-12-25	2024-12-30	CONGE_NON_PAYE	2	1 - SAISSI Zahra	2025-01-01	2025-01-04	CONGE_NON_PAYE	4	3 - ZAHNI jihad	2024-12-25	2024-12-30	CONGE_MALADIE	5	1 - SAISSI Zahra	2024-12-25	2024-12-30	CONGE_PAYE	6	4 - ihndfkc cfjiuji	2025-01-24	2025-01-30	CONGE_PAYE	7	2 - KOUIS Khaoula	2025-12-24	2025-12-30	CONGE_PAYE	8	2 - KOUIS Khaoula	2025-02-01	2025-02-04	CONGE_MALADIE	13	8 - OUHIDOUS Inas	2025-01-01	2025-01-03	CONGE_PAYE
ID	nom_employe	date_debut	date_fin	type																																										
1	1 - SAISSI Zahra	2024-12-25	2024-12-30	CONGE_NON_PAYE																																										
2	1 - SAISSI Zahra	2025-01-01	2025-01-04	CONGE_NON_PAYE																																										
4	3 - ZAHNI jihad	2024-12-25	2024-12-30	CONGE_MALADIE																																										
5	1 - SAISSI Zahra	2024-12-25	2024-12-30	CONGE_PAYE																																										
6	4 - ihndfkc cfjiuji	2025-01-24	2025-01-30	CONGE_PAYE																																										
7	2 - KOUIS Khaoula	2025-12-24	2025-12-30	CONGE_PAYE																																										
8	2 - KOUIS Khaoula	2025-02-01	2025-02-04	CONGE_MALADIE																																										
13	8 - OUHIDOUS Inas	2025-01-01	2025-01-03	CONGE_PAYE																																										

Ajouter Modifier Supprimer Afficher

Gestion des employés et des congés

Employe Holiday

Nom de l'employé	8 - OUHIDOUS Inas																																													
Date de début (YYYY-MM-DD)	2025-01-01																																													
Date de fin (YYYY-MM-DD)	2025-01-12																																													
Type	CONGE_PAYE																																													
<table border="1"> <thead> <tr> <th>ID</th> <th>nom_employe</th> <th>date_debut</th> <th>date_fin</th> <th>type</th> </tr> </thead> <tbody> <tr><td>1</td><td>1 - SAISSI Zahra</td><td>2024-12-25</td><td>2024-12-30</td><td>CONGE_NON_PAYE</td></tr> <tr><td>2</td><td>1 - SAISSI Zahra</td><td>2025-01-01</td><td>2025-01-04</td><td>CONGE_NON_PAYE</td></tr> <tr><td>4</td><td>3 - ZAHNI jihad</td><td></td><td>-30</td><td>CONGE_MALADIE</td></tr> <tr><td>5</td><td>1 - SAISSI Zahra</td><td></td><td>-30</td><td>CONGE_PAYE</td></tr> <tr><td>6</td><td>4 - ihndfkc cfjiuji</td><td></td><td>-30</td><td>CONGE_PAYE</td></tr> <tr><td>7</td><td>2 - KOUIS Khaoula</td><td></td><td>-30</td><td>CONGE_PAYE</td></tr> <tr><td>8</td><td>2 - KOUIS Khaoula</td><td></td><td>-04</td><td>CONGE_MALADIE</td></tr> <tr><td>13</td><td>8 - OUHIDOUS Inas</td><td></td><td>-03</td><td>CONGE_PAYE</td></tr> </tbody> </table>		ID	nom_employe	date_debut	date_fin	type	1	1 - SAISSI Zahra	2024-12-25	2024-12-30	CONGE_NON_PAYE	2	1 - SAISSI Zahra	2025-01-01	2025-01-04	CONGE_NON_PAYE	4	3 - ZAHNI jihad		-30	CONGE_MALADIE	5	1 - SAISSI Zahra		-30	CONGE_PAYE	6	4 - ihndfkc cfjiuji		-30	CONGE_PAYE	7	2 - KOUIS Khaoula		-30	CONGE_PAYE	8	2 - KOUIS Khaoula		-04	CONGE_MALADIE	13	8 - OUHIDOUS Inas		-03	CONGE_PAYE
ID	nom_employe	date_debut	date_fin	type																																										
1	1 - SAISSI Zahra	2024-12-25	2024-12-30	CONGE_NON_PAYE																																										
2	1 - SAISSI Zahra	2025-01-01	2025-01-04	CONGE_NON_PAYE																																										
4	3 - ZAHNI jihad		-30	CONGE_MALADIE																																										
5	1 - SAISSI Zahra		-30	CONGE_PAYE																																										
6	4 - ihndfkc cfjiuji		-30	CONGE_PAYE																																										
7	2 - KOUIS Khaoula		-30	CONGE_PAYE																																										
8	2 - KOUIS Khaoula		-04	CONGE_MALADIE																																										
13	8 - OUHIDOUS Inas		-03	CONGE_PAYE																																										

Succès
Holiday est modifié avec succès.

OK

Ajouter Modifier Supprimer Afficher

-Le Solde change:

Gestion des employés et des congés

Employe **Holiday**

Nom																																																															
Prenom																																																															
Email																																																															
Telephone																																																															
Salaire																																																															
Role	ADMIN																																																														
Poste	INGENIEURE_ETUDE_ET DEVELOPPEMENT																																																														
<table border="1"> <thead> <tr> <th>ID</th> <th>Nom</th> <th>Prenom</th> <th>Email</th> <th>Telephone</th> <th>Salaire</th> <th>Role</th> <th>Poste</th> <th>Solde</th> </tr> </thead> <tbody> <tr><td>1</td><td>SAISSI</td><td>Zahra</td><td>zahra@email.com</td><td>0612345678</td><td>100000.0</td><td>ADMIN</td><td>INGENIEURE_E... 9</td><td></td></tr> <tr><td>2</td><td>KOUISS</td><td>Khaoula</td><td>kouis@email.com</td><td>0612345678</td><td>25000.0</td><td>EMPLOYE</td><td>INGENIEURE_E... 3</td><td></td></tr> <tr><td>3</td><td>ZAHNI</td><td>jihad</td><td>jihad@email.com</td><td>0612345678</td><td>50000.0</td><td>EMPLOYE</td><td>TEAM LEADER 15</td><td></td></tr> <tr><td>4</td><td>jhnfdk</td><td>cjiuij</td><td>dfji@ujhc</td><td>0612345678</td><td>1000.0</td><td>ADMIN</td><td>INGENIEURE_E... 13</td><td></td></tr> <tr><td>8</td><td>OUHIDOUS</td><td>Inas</td><td>inas@gmail</td><td>0612345678</td><td>24000.0</td><td>ADMIN</td><td>PILOTE 12</td><td></td></tr> </tbody> </table>										ID	Nom	Prenom	Email	Telephone	Salaire	Role	Poste	Solde	1	SAISSI	Zahra	zahra@email.com	0612345678	100000.0	ADMIN	INGENIEURE_E... 9		2	KOUISS	Khaoula	kouis@email.com	0612345678	25000.0	EMPLOYE	INGENIEURE_E... 3		3	ZAHNI	jihad	jihad@email.com	0612345678	50000.0	EMPLOYE	TEAM LEADER 15		4	jhnfdk	cjiuij	dfji@ujhc	0612345678	1000.0	ADMIN	INGENIEURE_E... 13		8	OUHIDOUS	Inas	inas@gmail	0612345678	24000.0	ADMIN	PILOTE 12	
ID	Nom	Prenom	Email	Telephone	Salaire	Role	Poste	Solde																																																							
1	SAISSI	Zahra	zahra@email.com	0612345678	100000.0	ADMIN	INGENIEURE_E... 9																																																								
2	KOUISS	Khaoula	kouis@email.com	0612345678	25000.0	EMPLOYE	INGENIEURE_E... 3																																																								
3	ZAHNI	jihad	jihad@email.com	0612345678	50000.0	EMPLOYE	TEAM LEADER 15																																																								
4	jhnfdk	cjiuij	dfji@ujhc	0612345678	1000.0	ADMIN	INGENIEURE_E... 13																																																								
8	OUHIDOUS	Inas	inas@gmail	0612345678	24000.0	ADMIN	PILOTE 12																																																								

11. Supprimer un congé:

Gestion des employés et des congés

Employe **Holiday**

Nom de l'employé	8 - OUHIDOUS Inas			
Date de début (YYYY-MM-DD)	2025-01-01			
Date de fin (YYYY-MM-DD)	2025-01-12			
Type	CONGE_PAYE			
ID	nom_employe	date_debut	date_fin	type
1	1 - SAISSI Zahra	2024-12-25 2025-01-01	2024-12-30 2025-01-04	CONGE_NON_PAYE CONGE_NON_PAYE
2	1 - SAISSI Zahra			
4	3 - ZAHNI jihad			
5	1 - SAISSI Zahra			
6	4 - jhnfdk cjiuij			
7	2 - KOUISS Khaoula			
8	2 - KOUISS Khaoula			
13	8 - OUHIDOUS Inas			

Succès

i Holiday est supprimé.

Gestion des employés et des congés

Employe Holiday

Nom de l'employé	1 - SAISSI Zahra			
Date de début (YYYY-MM-DD)	2025-01-01			
Date de fin (YYYY-MM-DD)	2025-01-03			
Type	CONGE_PAYE			
ID	nom_employe	date_debut	date_fin	type
1	1 - SAISSI Zahra	2024-12-25	2024-12-30	CONGE_NON_PAYE
2	1 - SAISSI Zahra	2025-01-01	2025-01-04	CONGE_NON_PAYE
4	3 - ZAHNI jihad	2024-12-25	2024-12-30	CONGE_MALADIE
5	1 - SAISSI Zahra	2024-12-25	2024-12-30	CONGE_PAYE
6	4 - jhndfk cfiiji	2025-01-24	2025-01-30	CONGE_PAYE
7	2 - KOUIS Khaoula	2025-12-24	2025-12-30	CONGE_PAYE
8	2 - KOUIS Khaoula	2025-02-01	2025-02-04	CONGE_MALADIE

Ajouter **Modifier** **Supprimer** **Afficher**

Gestion des employés et des congés

Employe Holiday

Nom								
Prenom								
Email								
Telephone								
Salaire								
Role	ADMIN							
Poste	INGENIEURE_ETUDE_ET DEVELOPPEMENT							
ID	Nom	Prenom	Email	Telephone	Salaire	Role	Poste	Solde
1	SAISSI	Zahra	zahra@email.com	0612345678	100000.0	ADMIN	INGENIEURE_E...	9
2	KOUISS	Khaoula	kouis@email.com	0612345678	25000.0	EMPLOYEE	INGENIEURE_E...	3
3	ZAHNI	jihad	jihad@email.com	0612345678	50000.0	EMPLOYEE	TEAM_LEADER	15
4	jhndfk	cfiiji	dffji@ujhc	0612345678	1000.0	ADMIN	INGENIEURE_E...	13
8	OUHIDOUS	Inas	inas@gmail	0612345678	24000.0	ADMIN	PILOTE	23

Ajouter **Modifier** **Supprimer** **Afficher**

Conclusion:

En conclusion, ce projet met en avant une architecture MVC rigoureuse, garantissant une organisation claire et une bonne séparation des responsabilités entre les composants. Le système est conçu pour gérer efficacement les employés et leurs données, tout en facilitant l'intégration d'autres fonctionnalités comme la gestion des congés. Cette structuration améliore la maintenabilité et la compréhension globale du code.

L'introduction de la généricité renforcerait encore cette flexibilité en permettant une gestion standardisée des opérations sur différents types d'entités, telles que les employés et les congés. Cela réduirait la duplication de code, tout en rendant l'application plus modulaire et facile à étendre. Une approche générique, notamment dans les DAO et contrôleurs, favoriserait un code plus propre et réutilisable pour des systèmes encore plus complexes.