



Université Cadi Ayyad

École Supérieure de Technologie

Département : Informatique

Filière : Génie Informatique

---

Développement d'une application de gestion  
des Employés et Congés  
(MVC, DAO et Java Swing)

---

Réalisé par :  
ELKATMOUR Mounir

Enseigné par :  
Prof. EL ABDELLAOUI Said

Année Universitaire : 2024-2025



# Introduction générale

Dans le cadre des travaux pratiques réalisés (TP1 et TP2), nous avons développé une application desktop en utilisant le langage Java. Cette application vise à répondre à des besoins spécifiques dans la gestion des ressources humaines, en permettant la gestion des employés et des congés. Le choix de Java s'est imposé pour sa robustesse, sa portabilité, et sa large utilisation dans le développement d'applications professionnelles.

L'application a été conçue en suivant les principes de l'architecture MVC (Modèle-Vue-Contrôleur) afin de garantir une séparation claire des responsabilités et de rendre le code plus maintenable. Nous avons également appliqué le modèle DAO (Data Access Object) pour gérer les interactions avec la base de données de manière centralisée et efficace. Ces choix architecturaux ont permis d'assurer une gestion souple des données et une interface utilisateur claire et intuitive.

Ce projet a permis de mettre en pratique les concepts théoriques étudiés en cours, tout en nous offrant l'opportunité de développer des compétences techniques avancées, telles que la gestion des bases de données, la programmation orientée objet, et la gestion d'interfaces graphiques. Au-delà des aspects techniques, ce projet a renforcé notre capacité à travailler en équipe et à gérer un projet informatique de manière structurée et efficace.

# Table des matières

<b>Introduction</b>	<b>2</b>
<b>Chapitre 1 : Outils utilisées</b>	<b>1</b>
1.1 Introduction . . . . .	2
1.2 Outils et Environnements . . . . .	2
1.2.1 XAMPP . . . . .	2
1.2.2 Visual Studio Code (VS Code) . . . . .	2
1.2.3 JDBC (Java Database Connectivity) . . . . .	3
1.2.4 phpMyAdmin . . . . .	3
1.3 Langages et Frameworks . . . . .	3
1.3.1 Java . . . . .	3
1.3.2 Java Swing . . . . .	4
1.3.3 Java AWT . . . . .	4
1.3.4 SQL . . . . .	4
<b>Chapitre 2 : Base de donnée</b>	<b>5</b>
2.1 Introduction . . . . .	6
2.2 Les tables de ma base de données . . . . .	6
2.2.1 Table Employee . . . . .	6
2.2.2 Table Holiday . . . . .	7
<b>Chapitre 3 : Structure du projet</b>	<b>9</b>
3.1 Introduction . . . . .	10
3.2 Modèle MVC (Modèle-Vue-Contrôleur) . . . . .	10
3.3 Modèle DAO (Data Access Object) . . . . .	11
3.4 Conclusion . . . . .	11
<b>Chapitre 4 : Réalisation</b>	<b>12</b>
4.1 Gestion des employés . . . . .	13
4.1.1 Fonctionnalités principales . . . . .	13
4.1.2 Les fonctions implémentées . . . . .	15
4.2 Gestion des congés . . . . .	17
4.2.1 Les fonctionnalités principales . . . . .	17

4.2.2 Tests Effectués dans HolidayModel.java . . . . .	18
4.2.3 Les fonctions implémentées . . . . .	20
4.3 Conclusion . . . . .	20
<b>Conclusion Générale</b>	<b>22</b>
<b>Webographie</b>	<b>24</b>

# Liste des figures

1.1	XAMPP logo . . . . .	2
1.2	Visual Studio Code logo . . . . .	3
1.3	JDBC logo . . . . .	3
1.4	phpMyAdmin logo . . . . .	3
1.5	Java logo . . . . .	4
1.6	SQL logo . . . . .	4
2.7	Base de donnée . . . . .	6
2.8	Table employée . . . . .	7
2.9	Table holiday . . . . .	8
4.10	Interface gestion des employés . . . . .	13
4.11	Ajouter Employee Success . . . . .	14
4.12	Ajouter Employee Fail . . . . .	14
4.13	Modifier employee . . . . .	14
4.14	Affichage apres modification . . . . .	15
4.15	Supprimer Employee Confirmation . . . . .	15
4.16	Supprimer Employee . . . . .	15
4.17	Afficher Employee . . . . .	16
4.18	Rechercher Employee par Nom . . . . .	16
4.19	Intérface générique . . . . .	16
4.20	Intérface Employé . . . . .	16
4.21	Interface gestion des congés . . . . .	17
4.22	Ajouter un congé . . . . .	18
4.23	Modifier un congé . . . . .	18
4.24	Supprimer un congé . . . . .	18
4.25	Afficher les congés . . . . .	19
4.26	Afficher la liste des employés dans gestion de congés . . . . .	19
4.27	Test de date d'ajout du congé . . . . .	19

# Chapitre 1 : Outils utilisées

## 1.1 Introduction

Dans ce chapitre, nous allons détailler les outils, environnements, langages et frameworks utilisés pour le développement de l'application de gestion des employés et des congés. Chaque technologie a été choisie en fonction de son adéquation avec les besoins du projet et de ses capacités à assurer une interaction fluide entre les différentes couches de l'application.

## 1.2 Outils et Environnements

### 1.2.1 XAMPP

XAMPP est une solution de développement gratuite qui intègre Apache, MySQL, PHP et Perl. Cet outil a été choisi pour simuler un serveur local et faciliter le développement de l'application côté serveur.



Figure 1.1: XAMPP logo

### 1.2.2 Visual Studio Code (VS Code)

Visual Studio Code est un éditeur de code source puissant et léger qui permet de travailler efficacement avec différents langages. Il a été utilisé pour la gestion du code source, particulièrement en Java et PHP.



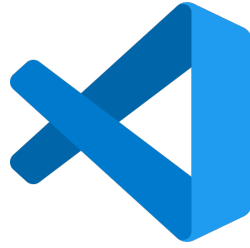


Figure 1.2: Visual Studio Code logo

### 1.2.3 JDBC (Java Database Connectivity)

JDBC permet à notre application Java d'interagir avec la base de données MySQL. Il facilite la gestion des requêtes SQL depuis l'application Java, assurant ainsi la communication entre le front-end et la base de données.



Figure 1.3: JDBC logo

### 1.2.4 phpMyAdmin

phpMyAdmin est un outil web permettant de gérer une base de données MySQL via une interface graphique. Cet outil est utilisé pour gérer et administrer les données liées aux employés et aux congés.



Figure 1.4: phpMyAdmin logo

## 1.3 Langages et Frameworks

### 1.3.1 Java

Java est un langage de programmation orienté objet largement utilisé pour le développement d'applications multiplateformes. Il est utilisé dans ce projet

pour la gestion des données et l'interaction avec la base de données.



Figure 1.5: Java logo

### 1.3.2 Java Swing

Java Swing est un framework pour créer des interfaces graphiques en Java. Il permet de créer des applications de bureau avec des composants interactifs, comme les boutons et les tableaux.

### 1.3.3 Java AWT

L'AWT est une bibliothèque graphique qui permet de gérer l'interface utilisateur. Bien que moins utilisée aujourd'hui, elle est encore présente dans certains cas pour des interfaces plus légères.

### 1.3.4 SQL

SQL est utilisé pour gérer les bases de données relationnelles. Dans ce projet, il est utilisé pour manipuler les données des employés et des demandes de congés à travers des requêtes SQL simples et complexes.



Figure 1.6: SQL logo

## Chapitre 2 : Base de donnée

## 2.1 Introduction

Dans ce chapitre, nous allons explorer la structure de la base de données utilisée pour le projet de gestion des employés et des congés. Nous détaillerons les tables principales et leur relation, ainsi que la manière dont elles permettent de gérer les données de manière cohérente et efficace.

## 2.2 Les tables de ma base de données

La base de données utilisée dans ce projet est structurée en plusieurs tables essentielles pour le bon fonctionnement de l'application. Ces tables ont été conçues pour gérer les informations des employés, leurs rôles, leurs postes, ainsi que les demandes de congé.

Table	Action
<input type="checkbox"/> <b>employee</b>	★ Browse Structure Search Insert Empty Drop
<input type="checkbox"/> <b>holiday</b>	★ Browse Structure Search Insert Empty Drop
<input type="checkbox"/> <b>login</b>	★ Browse Structure Search Insert Empty Drop
<input type="checkbox"/> <b>poste</b>	★ Browse Structure Search Insert Empty Drop
<input type="checkbox"/> <b>role</b>	★ Browse Structure Search Insert Empty Drop

Figure 2.7: Base de donnée

### 2.2.1 Table Employee

La table **Employee** contient des informations relatives aux employés de l'entreprise. Voici la description des colonnes de cette table :

- **Id** : (Type : `int(11)`) - Un identifiant unique pour chaque employé.
- **Nom** : (Type : `varchar(100)`) - Le nom de l'employé, limité à 100 caractères.
- **Prénom** : (Type : `varchar(100)`) - Le prénom de l'employé, limité à 100 caractères.
- **Salaire** : (Type : `double`) - Le salaire de l'employé, stocké sous forme de nombre à virgule flottante.

- **Email** : (Type : `varchar(100)`) - L'adresse e-mail de l'employé, limitée à 100 caractères.
- **Phone** : (Type : `varchar(100)`) - Le numéro de téléphone de l'employé, limité à 100 caractères.
- **Role** : (Type : `varchar(100)`) - Le rôle ou la fonction de l'employé dans l'entreprise (par exemple, "ADMIN", "MANAGER" et "EMPLOYEE").
- **Poste** : (Type : `varchar(100)`) - Le poste occupé par l'employé au sein de l'entreprise (par exemple, "PILOTE", "TEAM LEADER" et "INGENIEUR ETUDE ET DEVELOPPEMENT").
- **HolidayBalance** : (Type : `int(11)`) - Le solde des congés de l'employé, exprimé en jours.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	<b>id</b>	int(11)			No	None		AUTO_INCREMENT	Change  Drop  More
<input type="checkbox"/> 2	<b>nom</b>	varchar(100)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/> 3	<b>prenom</b>	varchar(100)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/> 4	<b>salaire</b>	double			No	None			Change  Drop  More
<input type="checkbox"/> 5	<b>email</b>	varchar(100)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/> 6	<b>phone</b>	varchar(100)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/> 7	<b>role</b>	varchar(100)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/> 8	<b>poste</b>	varchar(100)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/> 9	<b>holidayBalance</b>	int(11)			No	None			Change  Drop  More

Figure 2.8: Table employée

### 2.2.2 Table Holiday

La table `Holiday` contient les informations relatives aux congés des employés. Elle est structurée de la manière suivante :

- **ID** : (Type : `int(11)`) - Identifiant unique du congé. Il permet de distinguer chaque enregistrement de congé dans la base de données.
- **Employee\_ID** : (Type : `int(11)`) - L'identifiant de l'employé auquel le congé est attribué. Cette colonne fait référence à l'ID de la table `Employee`.
- **Type** : (Type : `varchar(255)`) - Le type de congé (par exemple, "Congé payé", "Congé non payé", "Congé maladie", etc.), limité à 255 caractères.

- **Start** : (Type : date) - La date de début du congé, stockée au format YYYY-MM-DD.
- **End** : (Type : date) - La date de fin du congé, également stockée au format YYYY-MM-DD.


















#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id 	int(11)			No	None		AUTO_INCREMENT	 Change  Drop  More
<input type="checkbox"/>	2 employee_id 	int(11)			No	None			 Change  Drop  More
<input type="checkbox"/>	3 type	varchar(255)	utf8mb4_general_ci		No	None			 Change  Drop  More
<input type="checkbox"/>	4 start	date			No	None			 Change  Drop  More
<input type="checkbox"/>	5 end	date			No	None			 Change  Drop  More

Figure 2.9: Table holiday

## Chapitre 3 : Structure du projet

## 3.1 Introduction

Dans ce chapitre, nous décrivons l'architecture de notre projet, qui repose sur le modèle **MVC (Modèle-Vue-Contrôleur)** et l'approche **DAO (Data Access Object)**. Ces deux paradigmes permettent une organisation claire et modulaire du code, facilitant ainsi sa maintenance.

## 3.2 Modèle MVC (Modèle-Vue-Contrôleur)

Le design **MVC** divise l'application en trois composants principaux pour séparer les responsabilités :

- **Vue (View) :**
  - La vue représente tout ce que l'utilisateur peut voir et interagir avec. Il s'agit de l'interface utilisateur qui affiche les données et recueille les entrées de l'utilisateur.
  - Par exemple, dans notre projet, **EmployeeView** est une classe qui affiche les informations des employés à l'écran.
- **Contrôleur (Controller) :**
  - Le contrôleur agit comme un intermédiaire entre la vue et le modèle. Il gère les entrées de l'utilisateur (comme des clics ou des saisies), les transmet au modèle, puis met à jour la vue avec les sorties appropriées.
  - Dans notre projet, **EmployeeController** est responsable de récupérer les données des employés depuis le modèle et de les transmettre à la vue, ou vice versa.
- **Modèle (Model) :**
  - Le modèle contient la logique métier de l'application. C'est ici que nous effectuons des validations, des tests, ou encore des calculs avant d'envoyer ou de récupérer les données via la couche DAO.
  - Par exemple, dans notre projet, le modèle comme **EmployeeModel** interagit directement avec les classes DAO pour gérer la persistance des données dans la base de données.



L'architecture MVC permet ainsi de séparer les rôles pour rendre le code plus lisible et modulaire.

### 3.3 Modèle DAO (Data Access Object)

Le modèle **DAO** est utilisé pour gérer les interactions avec la base de données de manière centralisée et uniforme. Cette couche se compose de :

- **DBConnection** : Responsable de la connexion à la base de données.
- **EmployeeDAOI** : Une interface définissant les opérations de base comme la récupération, l'insertion, la mise à jour ou la suppression.
- **EmployeeDAOImpl** : L'implémentation de **EmployeeDAOI**, contenant les requêtes SQL spécifiques.

Cette architecture présente plusieurs avantages :

- Un découplage entre la logique métier et la logique d'accès aux données.
- Une meilleure réutilisabilité du code.
- Une testabilité accrue grâce à la possibilité de remplacer DAO par des implémentations de test.

### 3.4 Conclusion

Voici un résumé de l'interaction entre les différentes couches :

1. L'utilisateur interagit avec la **vue** pour effectuer des actions ou consulter des données.
2. Le **contrôleur** capte ces actions, les traite, et communique avec le **modèle** pour appliquer les règles métier.
3. Le **modèle** effectue des validations ou des tests et utilise les classes DAO pour interagir avec la base de données.
4. Une fois les données récupérées ou mises à jour, le **contrôleur** met à jour la **vue** pour afficher les nouvelles informations à l'utilisateur.

Ce système assure une architecture propre et bien structurée, essentielle pour des projets robustes et maintenables.

## Chapitre 4 : Réalisation

## 4.1 Gestion des employés

Cette section détaille l'ensemble des fonctionnalités et des modules développés pour la gestion des employés.

Id	Nom	Prenom	Email	Salaire	Phone	Role	Poste	Holiday Bala...
3	Elkatmour	Mounir	mounirelkat@gmail.com	93000.0	0717728208	ADMIN	TEAM LEA...	11
4	Koubi	Mohamed-Yassi...	mohamedyassinkoubi@gmail	10000.22	0645593798	EMPLOYEE	INGENIEUR...	8
5	Datch	Adam	datchpro7@gmail.com	5500.0	0777994673	EMPLOYEE	PILOTE	25
8	Elaouani	Adam	adamelaouani@gmail.com	7000.0	0689638596	EMPLOYEE	INGENIEUR...	25
15	Adrar	Marouane	adrarmarouane@gmail.com	7820.0	0677262829	EMPLOYEE	PILOTE	25

Figure 4.10: Interface gestion des employés

### 4.1.1 Fonctionnalités principales

- Ajouter un nouvel employé : un formulaire permet de collecter les informations nécessaires (nom, prénom, email, téléphone, poste, rôle, etc.).
- Modifier les informations d'un employé : possibilité de mettre à jour les données existantes.
- Supprimer un employé : suppression des informations d'un employé depuis la base de données.
- Afficher la liste des employés : récupération et affichage des données dans un tableau interactif.

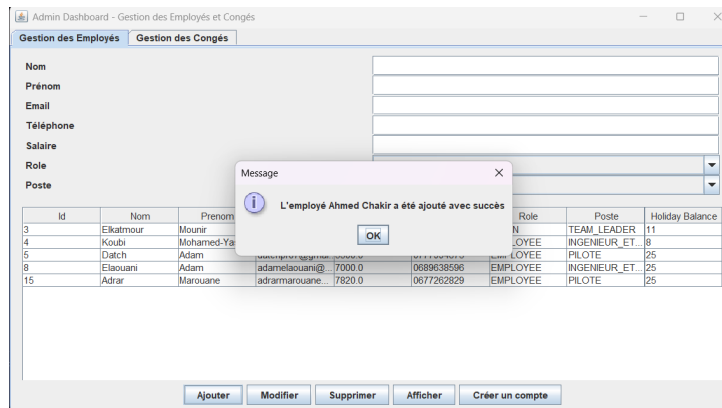


Figure 4.11: Ajouter Employee Success

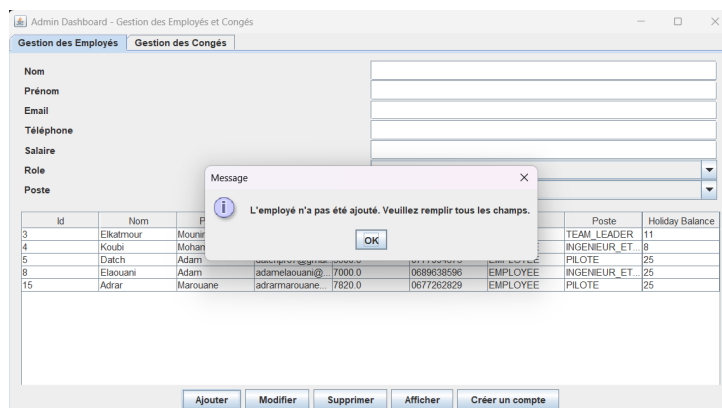


Figure 4.12: Ajouter Employee Fail

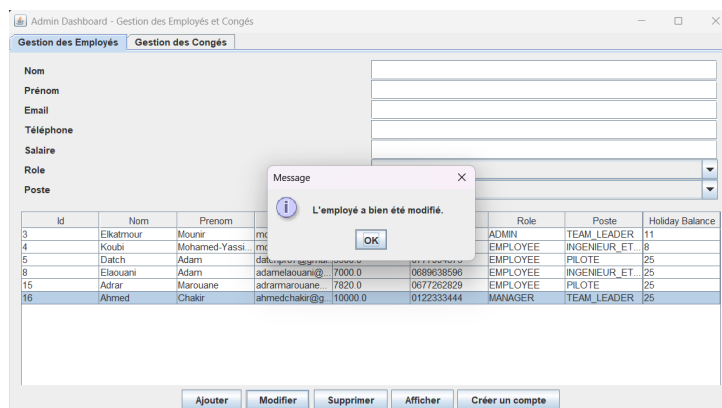


Figure 4.13: Modifier employee

- Recherche des employés : Effectuer des recherches des employés depuis la base de données selon leurs noms, prenom, email, telephone ou salaire.

Exemple:

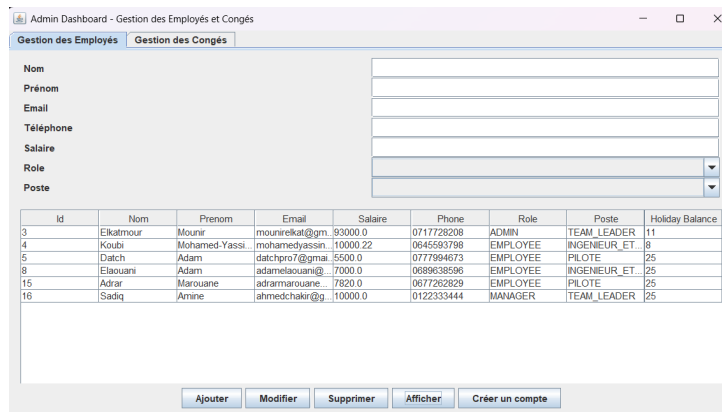


Figure 4.14: Affichage apres modification

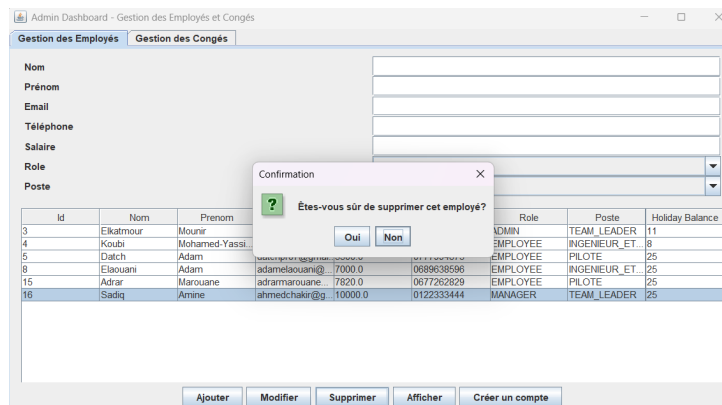


Figure 4.15: Supprimer Employee Confirmation

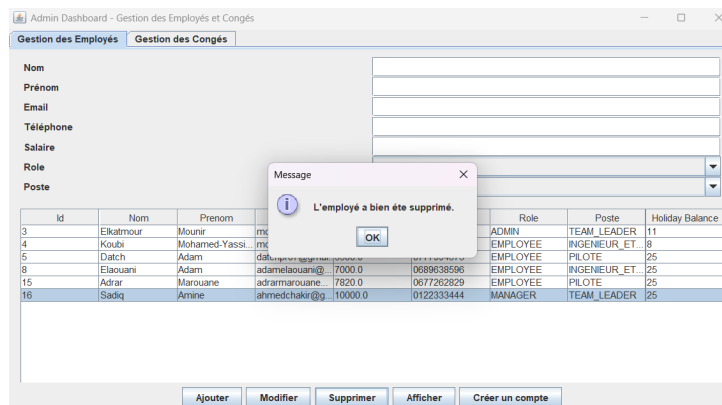


Figure 4.16: Supprimer Employee

## 4.1.2 Les fonctions implémentées

- Implémentation de l'interface générique
- Implémentation de l'interface Employé pour les filtres

Admin Dashboard - Gestion des Employés et Congés

Gestion des Employés    Gestion des Congés

Nom

Prénom

Email

Téléphone

Salaire

Rôle

Poste

Id	Nom	Prenom	Email	Salaire	Phone	Role	Poste	Holiday Balance
3	Elkalmour	Mourir	mounirelkat@gmail	93000.0	0717728208	ADMIN	TEAM LEADER	11
4	Koubi	Mohamed-Yassi	mohamedyassin	10000.22	0645593798	EMPLOYEE	INGENIEUR_ET	8
5	Datch	Adam	datchpro7@gmail	5500.0	0777994673	EMPLOYEE	PILOTE	25
8	Elaouani	Adam	adamelaouani@	7000.0	0689638596	EMPLOYEE	INGENIEUR_ET	25
15	Adrar	Marouane	adramarouane	7820.0	0677262829	EMPLOYEE	PILOTE	25
16	Ahmed	Chakr	ahmedchakr@	10000.0	0122333444	MANAGER	TEAM LEADER	25

Ajouter    Modifier    Supprimer    Afficher    Créer un compte

Figure 4.17: Afficher Employee

Nom

Prénom

Email

Téléphone

Salaire

Rôle

Poste

Id	Nom	Prenom	Email	Salaire	Phone	Role	Poste	Holiday Balance
15	Adrar	Marouane	adramarouane	7820.0	0677262829	EMPLOYEE	PILOTE	20
17	Adrar	Adam	adamadrar@gmail	12200.0	0122112012	MANAGER	PILOTE	25

Ajouter    Modifier    Supprimer    Afficher    Créer un compte

Figure 4.18: Rechercher Employee par Nom

```
public interface GeneriqueDAOI<T> {
    public List<T> afficher();
    public void ajouter(T t);
    public void modifier(T t,int id);
    public void supprimer(int id);
    public Employee findById(int EmployeeId);
}
```

Figure 4.19: Interface générique

```
public interface EmployeeDAOI {
    public List<Employee> findByFullName(String firstname,String lastname);
    public List<Employee> findByEmail(String email);
    public List<Employee> findByFirstName(String firstname);
    public List<Employee> findByLastName(String lastname);
    public List<Employee> findByPhone(String phone);
    public List<Employee> findBySalaire(double salaire);
}
```

Figure 4.20: Interface Employé

## 4.2 Gestion des congés

Cette section détaille les fonctionnalités et modules dédiés à la gestion des congés.

Id	Employé	Type	Date début	Date fin
22	Elkatmour Mounir	CONGE_PAYE	2025-01-01	2025-01-15
23	Koubi Mohamed-Yassine	CONGE_PAYE	2024-12-16	2025-01-02

Figure 4.21: Interface gestion des congés

### 4.2.1 Les fonctionnalités principales

Le module de gestion des congés offre les fonctionnalités suivantes visibles à l'utilisateur :

- **Ajouter un congé** : Permet d'enregistrer un nouveau congé pour un employé dans le système.
- **Modifier un congé** : Permet de mettre à jour les informations d'un congé existant.
- **Supprimer un congé** : Permet de supprimer un congé de la liste.
- **Afficher la liste des congés** : Affiche tous les congés enregistrés pour consultation ou modification.
- **Afficher la liste des employés** : Fournit un aperçu des employés pour la gestion des congés.

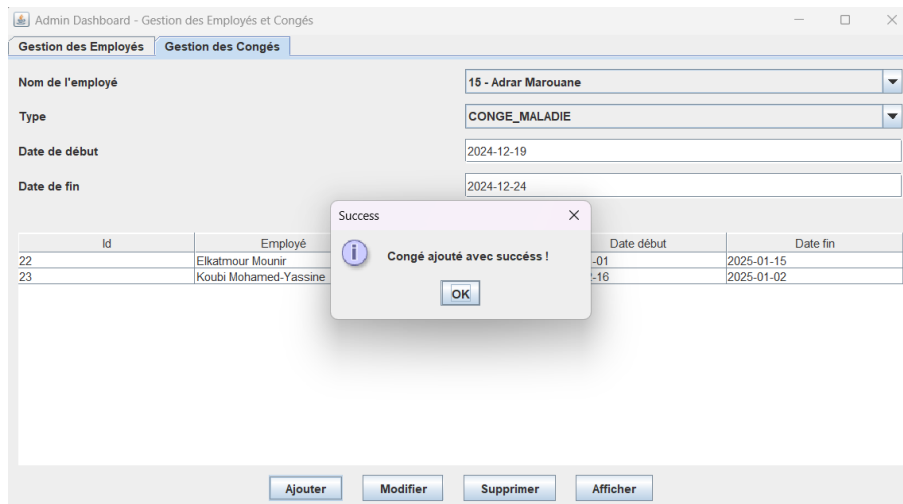


Figure 4.22: Ajouter un congé

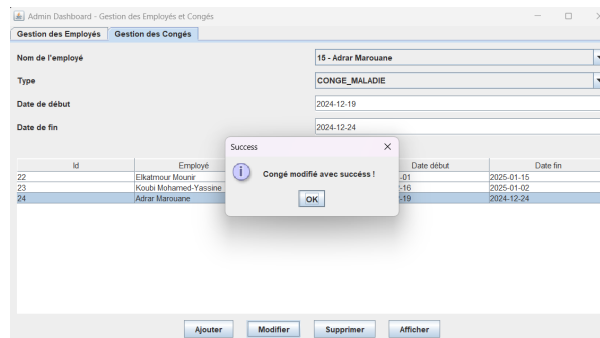


Figure 4.23: Modifier un congé

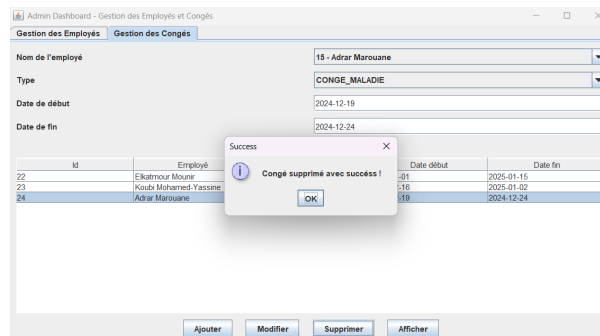


Figure 4.24: Supprimer un congé

## 4.2.2 Tests Effectués dans HolidayModel.java

Les tests effectués dans `HolidayModel.java` visent à garantir que les opérations de gestion des congés (ajout, modification et suppression) respectent les règles métiers et fonctionnent correctement. Voici un résumé des tests réalisés :

- **Test d'ajout de congé** : Ce test vérifie que l'ajout d'un congé respecte



Figure 4.25: Afficher les congés

Figure 4.26: Afficher la liste des employés dans gestion de congés

les conditions suivantes :

- La date de début doit être avant la date actuelle.
- La date de fin doit être postérieure à la date de début.
- L'employé doit disposer d'un solde suffisant de jours de congé.

Exemple :

Figure 4.27: Test de date d'ajout du congé

- **Test de modification de congé :** Lors de la modification d'un congé, ce test s'assure que :
  - La nouvelle date de début est valide (avant la date actuelle).
  - La nouvelle date de fin est bien après la date de début.
  - Le solde de congés de l'employé est mis à jour en fonction des nouvelles dates.
  - Si l'employé change, l'ancien employé récupère son solde de congés précédent et le solde du nouvel employé diminue en fonction du nombre de jours qu'il a choisis pour ses congés.
- **Test de suppression de congé :** Ce test garantit que la suppression d'un congé entraîne la récupération des jours de congé dans le solde de l'employé et que le congé est bien supprimé de la base de données.

Ces tests permettent de s'assurer que les opérations de gestion des congés sont fiables et cohérentes avec les règles définies dans le système.

### 4.2.3 Les fonctions implémentées

L'ensemble des fonctionnalités est réalisé en implémentant une interface générique, **GeneriqueDAOI**, qui standardise les opérations CRUD (ajouter, modifier, supprimer, consulter) pour différentes entités. Cela garantit une flexibilité et une réutilisabilité des méthodes dans d'autres modules.

Dans le cas spécifique des congés, une classe concrète, **HolidayDAOImpl**, implémente cette interface et gère les interactions avec la base de données pour les entités **Holiday** et **Employee**.

## 4.3 Conclusion

Pour conclure, le module de gestion des employés et des congés représente une solution intégrée et efficace pour la gestion des ressources humaines. Grâce à ses fonctionnalités principales telles que l'ajout, la modification, la suppression et l'affichage des données des employés et des congés, il offre une interface intuitive pour les utilisateurs.

L'implémentation d'une interface générique (**GeneriqueDAOI**) garantit une structure claire et réutilisable des opérations CRUD, facilitant ainsi la

gestion des différentes entités. Ce projet démontre une approche modulaire et évolutive qui peut être enrichie par de nouvelles fonctionnalités à l'avenir.

## Conclusion Générale

Dans le cadre de nos travaux pratiques, nous avons eu l'opportunité de réaliser une application de gestion d'employés en utilisant le langage de programmation Java et en adoptant une architecture bien structurée basée sur le modèle MVC (Model-View-Controller). Ce projet nous a permis d'appliquer concrètement les concepts étudiés, notamment la séparation des responsabilités, la gestion des bases de données à l'aide du DAO (Data Access Object), et la manipulation des interfaces graphiques.

Cette expérience a été particulièrement enrichissante, car elle nous a permis non seulement de renforcer nos compétences techniques, mais également de développer une meilleure compréhension des bonnes pratiques de conception logicielle. En outre, les défis rencontrés, tels que la gestion des erreurs, l'optimisation du code et l'intégration des composants, ont contribué à améliorer notre capacité à résoudre des problèmes complexes dans un contexte réel.

En conclusion, ce projet a été une excellente opportunité pour approfondir nos connaissances en développement logiciel tout en consolidant des compétences essentielles pour notre future carrière. Il nous a également sensibilisés à l'importance d'une méthodologie structurée et collaborative dans la réussite de tout projet informatique.

# Webographie

[1] - <https://github.com/mounirelkatmour/Employees-management-JAVA-MVC-DAO-CRUD>