



**ECOLE MAROCAINE DES  
SCIENCES DE L'INGENIEUR**  
*Membre de*   
**HONORIS UNITED UNIVERSITIES**

---

# Conception et réalisation d'une solution intelligente pour la gestion des congés du personnel

---

Réalisé par :

M. Soufiane Moussaoui

Encadrante :

Mme. Amnay Meriem

## RAPPORT DE PROJET 4IIR

Période du projet : du 31 juillet 2025 au 20 août 2025

# Table des matières

|   |           |
|---|-----------|
| <b>Remerciement</b>                         | <b>5</b>  |
| <b>Résumé</b>                               | <b>6</b>  |
| <b>Abstract</b>                             | <b>6</b>  |
| <b>1 Introduction</b>                       | <b>7</b>  |
| 1.1 Contexte et problématique               | 7         |
| 1.2 Objectifs du projet                     | 7         |
| 1.3 Périmètre et structure du rapport       | 8         |
| 1.4 Techniques                              | 8         |
| 1.5 Méthodologie de travail                 | 9         |
| <b>2 État de l'Art</b>                      | <b>10</b> |
| 2.1 Technologies et outils de développemen  | 10        |
| 2.2 Base de Données MySQL                   | 11        |
| 2.3 Autres Outils de Développement          | 11        |
| <b>3 Analyse et Conception</b>              | <b>12</b> |
| 3.1 Analyse                                 | 12        |
| 3.1.1 Besoins Fonctionnels                  | 12        |
| 3.1.2 Besoins Non-Fonctionnels              | 12        |
| 3.2 Conception                              | 13        |
| 3.2.1 Couche Application                    | 13        |
| 3.2.2 Couche Données                        | 13        |
| 3.2.3 Considérations de Sécurité            | 14        |
| 3.2.4 Conception des Interfaces Utilisateur | 14        |
| 3.2.5 Diagrammes UML                        | 14        |
| 3.2.6 Digramme de cas d'utilisation         | 14        |
| 3.2.7 Digramme de class                     | 15        |
| 3.2.8 Digramme de sequence                  | 16        |
| <b>4 Implementation</b>                     | <b>18</b> |
| 4.1 Environnement de développement          | 18        |
| 4.1.1 Configuration du système              | 18        |
| 4.1.2 Langages et frameworks                | 18        |
| 4.1.3 Guide d'Installation                  | 18        |
| 4.2 Base de Données                         | 19        |
| 4.2.1 Modélisation des Données              | 20        |
| 4.2.2 Modèles Django Implémentés            | 20        |
| 4.2.3 Relations et Contraintes              | 21        |
| 4.3 Interface Utilisateur                   | 21        |

|   |   |           |
|---|---|-----------|
| 4.3.1                                     | Interface Employé . . . . .                 | 22        |
| 4.3.2                                     | Interface Manager/Administrateur . . . . .  | 23        |
| 4.3.3                                     | Export des rapports en PDF ou CSV . . . . . | 24        |
| 4.3.4                                     | Caractéristiques Communes . . . . .         | 25        |
| <b>Conclusion Générale . . . . .</b>      |   | <b>26</b> |
| <b>Perspectives d'Évolution . . . . .</b> |   | <b>26</b> |
| <b>Bibliographie . . . . .</b>            |   | <b>27</b> |
| <b>ANNEX . . . . .</b>                    |   | <b>28</b> |

# Table des figures

|      |  |    |
|------|--|----|
| 1.1  | Titre de la figure — MVT . . . . .                   | 8  |
| 2.1  | Titre de la figure — Project Structer . . . . .      | 10 |
| 2.2  | Titre de la figure — Mysql Workbench . . . . .       | 11 |
| 3.1  | Digramme de cas d'utilisation . . . . .              | 15 |
| 3.2  | Diagramme de class . . . . .                         | 16 |
| 3.3  | Digramme de sequence . . . . .                       | 17 |
| 4.1  | Diagramme Entité-Relations (ERD) . . . . .           | 20 |
| 4.2  | historique des congés . . . . .                      | 22 |
| 4.3  | demande congé . . . . .                              | 23 |
| 4.4  | dashboard admin . . . . .                            | 23 |
| 4.5  | Fonction de vue pour l'export des rapports . . . . . | 24 |
| 4.6  | gestion des demandes . . . . .                       | 25 |
| 4.7  | historique des congés . . . . .                      | 28 |
| 4.8  | Ajouter un type de congé . . . . .                   | 28 |
| 4.9  | gestion des demandes . . . . .                       | 29 |
| 4.10 | historique des congés . . . . .                      | 29 |
| 4.11 | profile employé . . . . .                            | 29 |
| 4.12 | formulaire demande congé . . . . .                   | 30 |

# Remerciement

Je tiens à exprimer ma profonde gratitude envers toutes les personnes qui ont contribué, de près ou de loin, à la réussite de ce projet de stage, et qui m'ont accompagné tout au long de sa réalisation.

Je remercie tout d'abord Madame Amnay Meriem, mon encadrante de stage, pour son accompagnement rigoureux, sa disponibilité constante et la qualité de ses conseils. Son implication et son sens du détail ont été essentiels pour m'orienter dans les différentes phases du projet. Grâce à son expertise et à ses remarques pertinentes, j'ai pu aborder ce travail avec méthode et progresser dans un cadre professionnel stimulant.

Je souhaite également remercier Monsieur Hamza Abouabid, enseignant de Django, dont les cours et l'approche pédagogique m'ont permis de maîtriser les outils nécessaires à la mise en œuvre du système de gestion des congés. Ses explications claires et concrètes ont été d'un grand soutien technique tout au long du développement.

J'exprime aussi ma reconnaissance à l'ensemble des enseignants et membres du personnel pédagogique pour les connaissances et les compétences qu'ils m'ont transmises au cours de ces trois années de formation. Leur engagement et leur encadrement ont constitué une base solide sur laquelle ce projet a pu se construire.

Je n'oublie pas mes camarades de promotion, avec qui j'ai partagé de nombreuses heures de travail, d'échange et de soutien mutuel. Leur esprit de collaboration et leur aide, parfois précieuse, ont largement enrichi cette expérience.

Enfin, je tiens à remercier ma famille pour son soutien moral et ses encouragements constants, qui m'ont porté tout au long de mon parcours académique et particulièrement durant cette période de stage.

## Résumé

Ce projet vise à développer une application web moderne de Gestion Intelligente des Congés du Personnel destinée à automatiser et optimiser l'ensemble du processus de gestion des demandes de congés au sein des organisations. Cette solution numérique vise à remplacer les méthodes traditionnelles manuelles par une plateforme intégrée permettant la dématérialisation complète du cycle de vie des demandes, depuis leur création par les employés jusqu'à leur validation finale par la hiérarchie.

Le système comprend quatre composantes principales : une interface de gestion sécurisée des comptes utilisateurs avec différenciation des rôles employés et managers, un système de suivi intégral des demandes avec mécanismes de validation et rejet, un tableau de bord analytique fournissant des statistiques détaillées et l'historique complet des congés, et une infrastructure spécialisée incluant l'export automatisé des rapports aux formats PDF et CSV, la visualisation du calendrier d'équipe et un système de notifications automatisées.

## Abstract

This project aims to develop a modern web application for Intelligent Personnel Leave Management designed to automate and optimize the entire leave request management process within organizations. This digital solution aims to replace traditional manual methods with an integrated platform enabling complete dematerialization of the request lifecycle, from their creation by employees to their final validation by management.

The system includes four main components : a secure user account management interface with role differentiation between employees and managers, a comprehensive request tracking system with validation and rejection mechanisms, an analytical dashboard providing detailed statistics and complete leave history, and specialized infrastructure including automated report export in PDF and CSV formats, team calendar visualization, and an automated notification system.

**Mots-clés :** Django, Python, Gestion des Congés, Application Web, Validation Hiérarchique, Suivi des Absences, UML, Base de Données, Authentification Sécurisée, Tableau de Bord, Notifications, Export PDF/CSV, Calendrier d'Équipe, Workflow d'Approbation, Interface Responsive, Contrôle d'Accès, Historique des Demandes.

# Chapitre 1

## Introduction

### 1.1 Contexte et problématique

Dans le contexte organisationnel contemporain, la gestion efficace des ressources humaines constitue un enjeu stratégique majeur pour les entreprises de toutes tailles. Parmi les processus administratifs critiques, la gestion des demandes de congés occupe une place centrale, impactant directement la planification des activités, la satisfaction des employés et l'efficacité opérationnelle.

Force est de constater que de nombreuses organisations continuent de s'appuyer sur des méthodes traditionnelles pour gérer ces processus : formulaires papier, feuilles de calcul partagées ou systèmes disparates. Cette approche fragmentée engendre des défis considérables qui compromettent l'efficacité organisationnelle. Les principales difficultés identifiées incluent la perte ou la détérioration de documents physiques, la complexité du suivi en temps réel des soldes de congés, les délais de traitement prolongés des demandes, et l'absence de traçabilité fiable des décisions prises.

Ces lacunes opérationnelles motivent le développement du système « Leave Management System », une solution technologique intégrée conçue pour moderniser et optimiser l'ensemble du processus de gestion des congés. Cette application web, développée sur la base du framework Django, vise à centraliser, automatiser et sécuriser la gestion des demandes de congés au sein de l'organisation.

### 1.2 Objectifs du projet

Ce projet s'articule autour de quatre objectifs stratégiques fondamentaux :

- Simplifier et rationaliser l'expérience utilisateur en proposant une interface à la fois intuitive et accessible, permettant aux employés de soumettre leurs demandes de congés en un minimum d'étapes et de réduire sensiblement les délais de traitement.

- Doter les responsables hiérarchiques et les équipes Ressources Humaines d'outils de gestion avancés, offrant des fonctionnalités de validation, de refus motivé et d'archivage systématique pour une prise de décision rapide et documentée.

— Garantir l'intégrité et la fiabilité des données au moyen d'un système de traçabilité exhaustif, intégrant un historique détaillé de toutes les actions effectuées ainsi qu'un mécanisme de notifications automatisées assurant une communication fluide entre tous les acteurs.

## 1.3 Périmètre et structure du rapport

Ce document technique présente une analyse complète du développement et de l'implémentation du système de gestion des congés. La structure adoptée suit une approche méthodologique rigoureuse, couvrant l'ensemble du cycle de développement logiciel.

L'analyse débute par une étude approfondie des exigences fonctionnelles et non fonctionnelles, établissant les fondements techniques et organisationnels du projet. Cette phase d'analyse est suivie par la présentation de la conception architecturale du système, incluant la modélisation détaillée de la base de données avec ses entités principales : profils employés, structures départementales, taxonomie des types de congés, et système de notifications.

## 1.4 Techniques

- Utiliser le framework **Django** pour bénéficier de sa robustesse et de ses fonctionnalités de sécurité natives.
- Implémenter une architecture **MVT (Modèle–Vue–Template)** claire et maintenable, facilitant l'évolution du code.
- Assurer la compatibilité avec tous les navigateurs modernes (Chrome, Firefox, Edge, Safari) par des tests et la mise en place de polyfills le cas échéant.

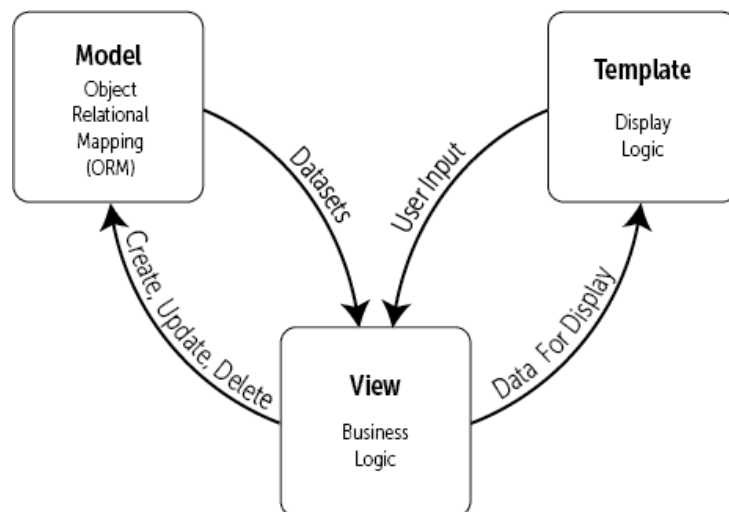


FIGURE 1.1 – Titre de la figure — MVT

**Garantir la sécurité de l'ensemble du système via :**

- Gestion granulaire des rôles et des permissions.
- Protection CSRF et XSS via les middleware Django.
- Hashage sécurisé des mots de passe (PBKDF2 / Argon2).



## 1.5 Méthodologie de travail

### Approche de développement

Le développement de ce projet a été réalisé en **mode individuel** (projet monôme) selon une démarche méthodologique rigoureuse et structurée.

### Outils et Infrastructure

**GitHub** pour la gestion du code source et le contrôle de versions

Cycle de développement étalé sur **4 semaines**

Approche itérative avec branches de fonctionnalités

### Bénéfices de cette approche

**Traçabilité complète** des modifications apportées

**Sauvegarde sécurisée** et redondante du code

**Documentation continue** via commits descriptifs

**Progression contrôlée** et mesurable

# Chapitre 2

## État de l'Art

La gestion des congés du personnel a évolué d'un processus administratif manuel vers des solutions numériques sophistiquées. Les solutions commerciales dominantes comme **BambooHR**, **Workday**, et **SAP SuccessFactors** offrent des fonctionnalités avancées mais restent coûteuses pour les PME. Les alternatives open source telles qu'**OrangeHRM** et **Odoo HR** démocratisent l'accès mais manquent souvent de personnalisation.

### 2.1 Technologies et outils de développemen

**Framework Django** s'impose comme framework de référence pour le développement d'applications de gestion d'entreprise grâce à sa philosophie "batteries included" et sa sécurité native.

L'architecture **MVT (Model-Template-View)** de Django favorise la maintenabilité et l'évolutivité du code, particulièrement adaptée aux besoins complexes de gestion des congés avec workflows d'approbation multi-niveaux.

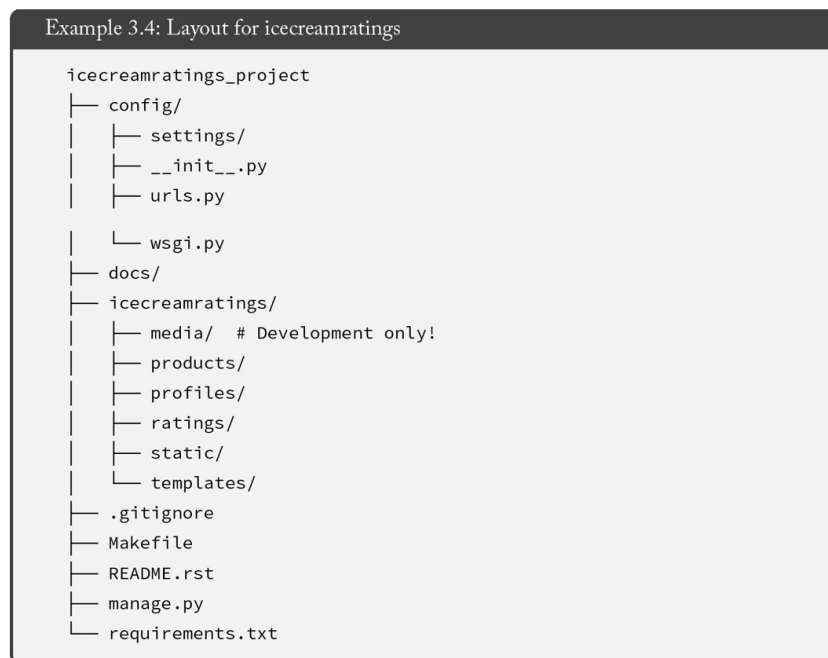


FIGURE 2.1 – Titre de la figure — Project Structer

## 2.2 Base de Données MySQL

**MySQL** constitue le choix optimal pour les systèmes de gestion des données grâce à sa fiabilité éprouvée et ses performances en environnement de production.

Sa compatibilité native avec Django via l'ORM simplifie les opérations **CRUD** et la gestion des relations complexes entre utilisateurs, demandes et validations.

Les fonctionnalités de transaction **ACID** garantissent l'intégrité des données critiques, tandis que les capacités de réplication et de sauvegarde assurent la continuité de service.

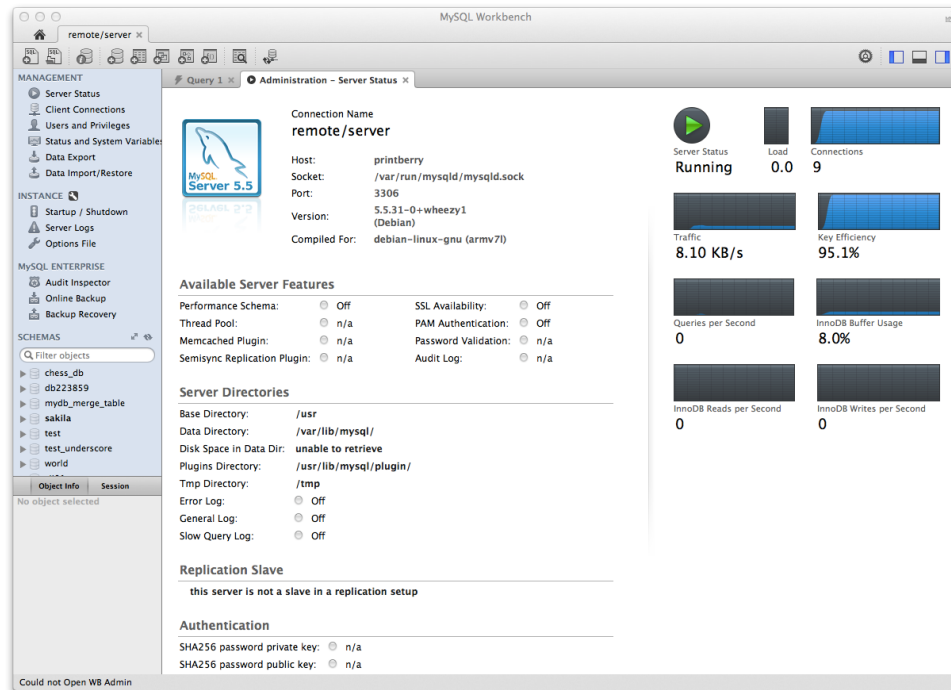


FIGURE 2.2 – Titre de la figure — Mysql Workbench

## 2.3 Autres Outils de Développement

L'écosystème de développement s'enrichit d'outils complémentaires essentiels : **HTML/CSS/JavaScript** pour les interfaces utilisateur universelles, **Bootstrap** pour le design responsive et cohérent, **Git** pour la gestion de versions, **VS Code** comme environnements de développement intégrés. Les outils de test comme **pytest** garantissent la qualité du code.

# Chapitre 3

## Analyse et Conception

### 3.1 Analyse

#### 3.1.1 Besoins Fonctionnels

Le système de gestion intelligente des congés du personnel doit répondre à des exigences fonctionnelles précises pour assurer une digitalisation complète du processus.

Les besoins primaires incluent la création et gestion sécurisée des comptes utilisateurs avec différenciation des rôles (employés, managers), permettant un contrôle d'accès granulaire selon les responsabilités hiérarchiques.

Le système doit permettre la gestion complète des profils employés (ajout, modification, suppression) et offrir un mécanisme de suivi intégral des demandes avec fonctionnalités de validation et rejet par la hiérarchie.

Les fonctionnalités avancées comprennent un tableau de bord analytique fournissant des statistiques détaillées et l'évolution des effectifs, un historique complet des congés pour la traçabilité, ainsi que des capacités d'export automatisé des **rapports aux formats PDF et CSV**.

L'interface doit intégrer une visualisation du calendrier d'équipe pour optimiser la planification des absences et un système de notifications automatisées informant les parties prenantes des décisions de validation ou refus des demandes.

#### 3.1.2 Besoins Non-Fonctionnels

**Vue d'ensemble :** Les exigences non-fonctionnelles constituent le socle de robustesse et de fiabilité du système, couvrant les aspects critiques de sécurité, performance et utilisabilité.

#### Sécurité et Protection des Données

- **Authentification robuste** avec mécanismes multi-niveaux
- **Chiffrement des données sensibles** (AES-256, TLS 1.3)
- **Contrôles d'autorisation stricts** basés sur les rôles utilisateurs
- **Protection des informations personnelles** conformément au RGPD

## Performance et Disponibilité

- **Disponibilité garantie** : 99.5% minimum (SLA)
- **Temps de réponse optimisés** : < 3 secondes pour les opérations courantes
- **Évolutivité technique** pour la montée en charge
- **Gestion de la charge** avec architecture scalable

## Expérience Utilisateur

- **Interface intuitive** avec design responsive
- **Compatibilité multi-navigateurs** (Chrome, Firefox, Safari, Edge)
- **Support mobile** pour tablettes et smartphones
- **Accessibilité** selon les standards WCAG 2.1

## Conformité et Intégrité

- **Conformité réglementaire** : RGPD, Code du travail français
- **Sauvegarde automatique** avec politique de rétention définie
- **Plan de reprise d'activité** en cas de panne système
- **Audit trail** complet des modifications de données

## Maintenabilité et Documentation

- **Code maintenable** suivant les bonnes pratiques (PEP 8, Clean Code)
- **Tests automatisés** avec couverture de code > 80%
- **Versioning sémantique** pour faciliter les évolutions

## 3.2 Conception

### 3.2.1 Couche Application

L'application est développée en utilisant le framework **Django**, qui suit une architecture **MVT (Model-View-Template)**. Elle offre une interface web sécurisée pour la gestion des congés du personnel. Les utilisateurs peuvent se connecter via un compte (employé ou administrateur), soumettre des demandes de congé, consulter leur statut et recevoir des notifications.

### 3.2.2 Couche Données

La couche de données repose sur un système de gestion de base de données relationnelle (ex : *PostgreSQL* ou *SQLite* selon l'environnement de déploiement). Django ORM (*Object Relational Mapping*) facilite les interactions avec la base sans requêtes SQL explicites.

Les principales entités modélisées sont :

- **Admin** : représente les administrateurs ayant des privilèges de gestion.

- **Employee** : contient les informations personnelles et professionnelles des employés.
- **Department** : regroupe les employés par unité administrative.
- **LeaveType** : types de congés (annuel, maternité, etc.).
- **Leave** : enregistre les demandes de congés avec leur statut.
- **Notification** : messages automatiques pour informer les utilisateurs.

### 3.2.3 Considérations de Sécurité

L'application adopte plusieurs bonnes pratiques de sécurité :

- Authentification sécurisée via le modèle intégré **User** de Django.
- Mots de passe hashés (grâce à la gestion native de Django).
- Accès restreints aux fonctions selon les rôles (employé, admin).
- Protection CSRF pour les formulaires web.
- Journalisation des actions critiques (audit trail possible).

### 3.2.4 Conception des Interfaces Utilisateur

Deux interfaces principales sont prévues :

- **Interface Employé** : tableau de bord personnel, formulaire de demande de congé, historique, notifications.
- **Interface Manager/Admin** : gestion des employés, validation des demandes, statistiques, génération de rapports.

### 3.2.5 Diagrammes UML

UML fournit un langage standard pour modéliser la structure et le comportement d'un système, améliorant la communication entre développeurs, chefs de projet et utilisateurs. Dans la gestion des congés, le diagramme de cas d'utilisation identifie acteurs et services (soumission, approbation, notification), tandis que le diagramme de classes formalise le modèle de données (Employee, Leave, Request) pour l'ORM.

### 3.2.6 Diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour une représentation du comportement fonctionnel d'un système logiciel.

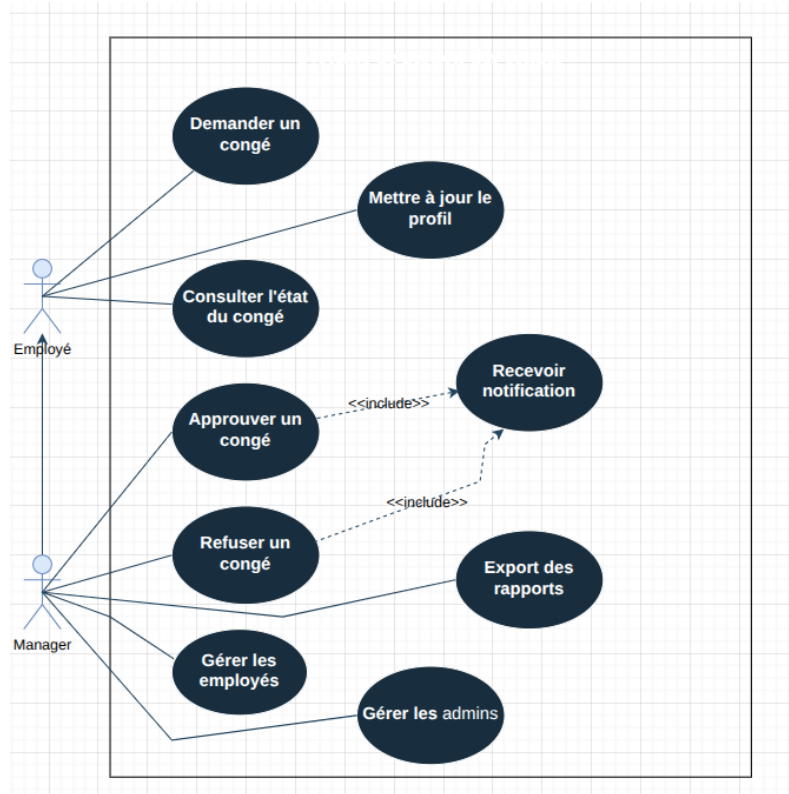


FIGURE 3.1 – Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation ci-dessus illustre les interactions entre deux acteurs : l'Employé et le Manager. L'Employé peut demander un congé, consulter son état, et mettre à jour son profil, tandis que le Manager peut gérer les employés et les admins, approuver ou refuser un congé, et exporter des rapports. Les actions d'approbation et de refus incluent l'envoi de notifications automatiques, représenté par le cas d'utilisation "Recevoir notification".

### 3.2.7 Diagramme de class

Ce diagramme de classes décrit la structure du système de gestion des congés en représentant les entités principales telles que **Employee**, **Admin**, **Conge**, et leurs relations.

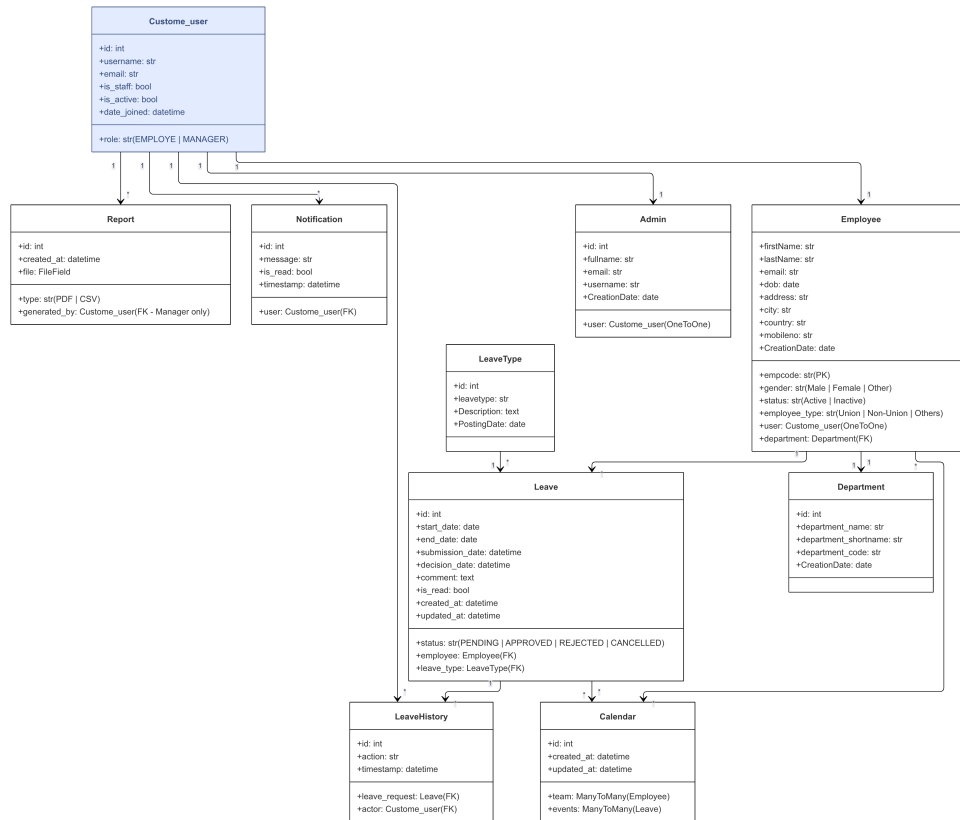


FIGURE 3.2 – Diagramme de class

Chaque entité contient des attributs pertinents, par exemple, *Employee* possède des informations personnelles, de contact et de statut, tandis que *Conge* contient les détails de la demande de congé.

### 3.2.8 Diagramme de sequence

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation Unified Modeling Language.



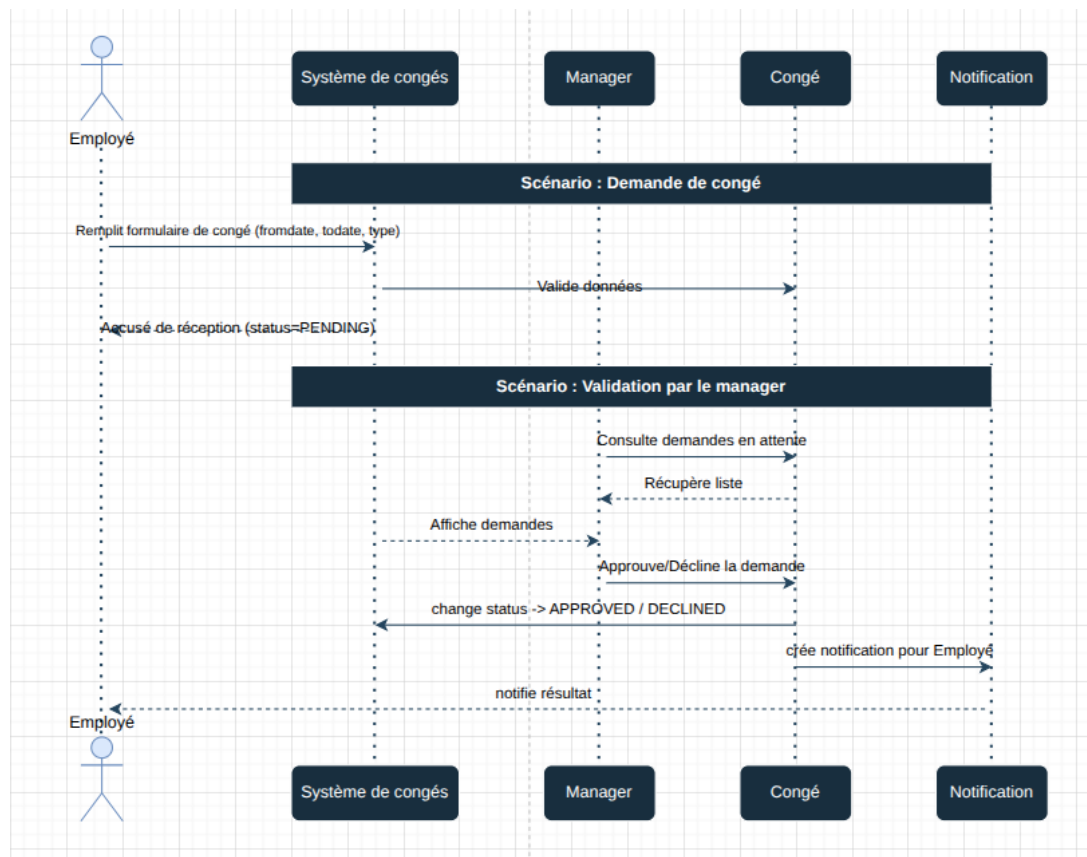


FIGURE 3.3 – Diagramme de sequence

Ce diagramme de séquence décrit l'interaction entre un employé, le système, le manager et les modules de congé et de notification lors d'une demande de congé. L'employé initie une demande via un formulaire, qui est ensuite validée et transmise au manager. Ce dernier peut approuver ou refuser la demande. Une notification est générée pour informer l'employé du résultat, après mise à jour du statut de la demande dans le système.

# Chapitre 4

## Implementation

### 4.1 Environnement de développement

La mise en place d'un environnement de développement cohérent est essentielle pour garantir la portabilité, la maintenabilité et la reproductibilité du projet. Ce projet a été réalisé par un seul développeur, ce qui simplifie la configuration et l'unification de l'environnement.

#### 4.1.1 Configuration du système

- **Système d'exploitation** : Arch Linux (rolling release, 64-bit).
- **Navigateur de test** : Mozilla Firefox (dernière version ESR), choisi pour sa conformité aux standards et ses outils de développement intégrés.
- **Éditeur de code** : Visual Studio Code avec extensions Python et Django.
- **Gestionnaire de code source** : Git (dépôt distant sur GitHub).

#### 4.1.2 Langages et frameworks

- **Back-end** : Python 3.11 + Django 4.x.
- **Front-end** :
  - *HTML5, CSS3, JavaScript* ES6+
  - **Bootstrap** 5 pour la mise en forme réactive et les composants UI.
- **Base de données** : MySQL 8.0, gérée via le connecteur `mysqlclient`.

#### 4.1.3 Guide d'Installation

**Prérequis** : Système Windows 10/11 avec droits administrateur et connexion Internet stable.

**Étape 1 : Installation des composants système** Téléchargez et installez les outils de développement essentiels :

- Python 3.11+ : <https://python.org/downloads/>
- MySQL Server : <https://dev.mysql.com/downloads/installer/>
- Git : <https://git-scm.com/download/win>
- Visual Studio Code : <https://code.visualstudio.com/>

- Mozilla Firefox : <https://firefox.com/>

**Étape 2 : Création de l'environnement virtuel** Configurez un environnement Python isolé via Command Prompt (cmd) :

```
python -m venv venv
venv\Scripts\activate
```

**Étape 3 : Installation des dépendances Python** Installez Django ainsi que les bibliothèques nécessaires avec pip :

```
pip install --upgrade pip
pip install Django mysqlclient
pip install -r requirements.txt
```

**Étape 5 : Configuration MySQL** Configurez la base de données MySQL sous Windows :

- Lancer MySQL Workbench ou Command Line Client
- Créer la base : `CREATE DATABASE nom_projet;`
- Mettre à jour les paramètres dans `settings.py`

Le développement a été réalisé sous *Arch Linux*, mais le guide ci-dessus illustre l'installation sous *Windows* afin de faciliter la reproduction pour un plus grand nombre d'utilisateurs.

## 4.2 Base de Données

La base de données du système de gestion des congés repose sur une architecture relationnelle optimisée pour Django, garantissant l'intégrité des données et les performances. Cette conception privilégie la normalisation des données tout en maintenant des relations cohérentes entre les différentes entités métier.

## 4.2.1 Modélisation des Données

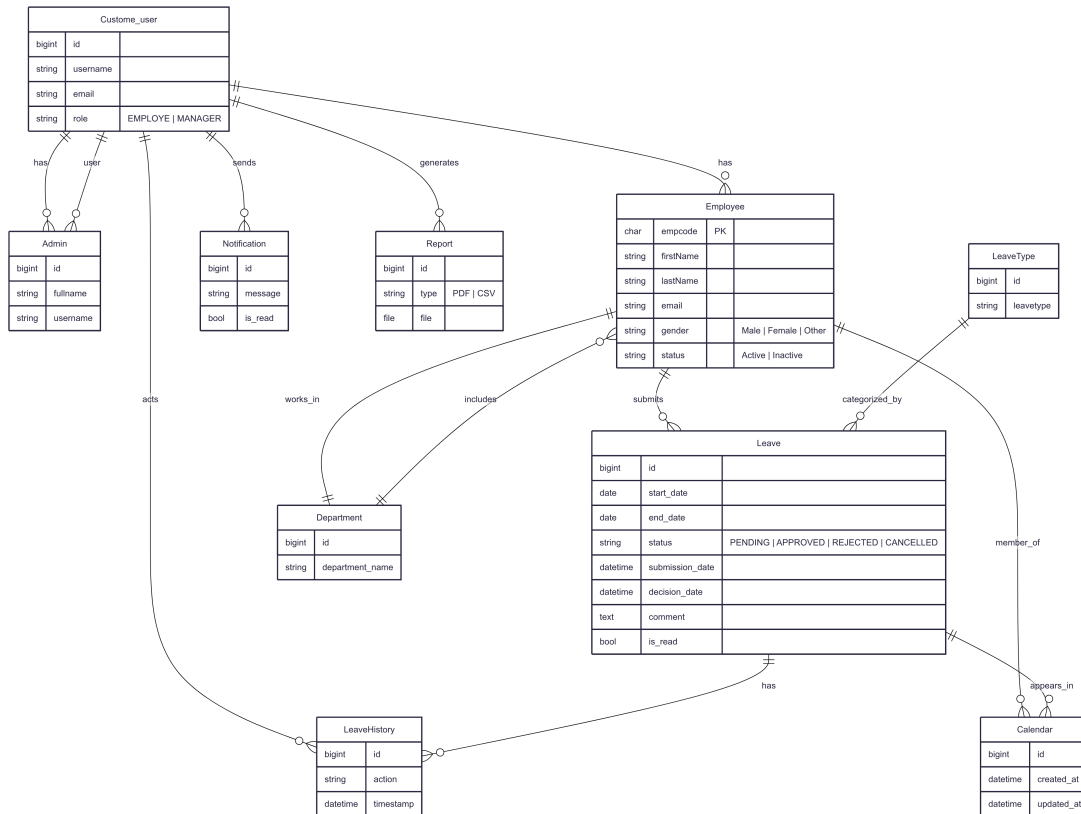


FIGURE 4.1 – Diagramme Entité-Relations (ERD)

## 4.2.2 Modèles Django Implémentés

### Modèle Admin

Le modèle *Admin* constitue l'extension du système d'authentification *Django*. Il établit une relation OneToOne avec le modèle *User* natif, permettant de conserver la robustesse du système tout en ajoutant des informations spécifiques aux administrateurs.

Les champs **fullname**, **email** et **username** stockent les informations d'identification, avec des contraintes d'unicité pour éviter les doublons.

### Modèle Department

La **structure organisationnelle** de l'entreprise est modélisée par l'entité *Department*. Chaque département possède un nom complet (**department\_name**), un nom abrégé (**department\_shortcode**) et un code unique (**department\_code**).

Cette *triple identification* permet une flexibilité dans la présentation des données selon le contexte d'utilisation.

## Modèle Employee

L'entité *Employee* représente le cœur informationnel du système, stockant les données personnelles et professionnelles. Chaque employé est identifié par un **code unique (empcode)** servant de clé primaire métier.

Le système implémente des **choix contrôlés** pour le genre, le type d'employé et le statut, garantissant la cohérence des données saisies.

## Modèle Leave

Le **processus de gestion des demandes** est centralisé dans l'entité *Leave*, qui implémente un workflow complet d'approbation. Les dates de début et fin définissent la période souhaitée.

Le *système de statut tripartite* (Pending, Approved, Declined) structure le processus d'approbation. Le champ **admin\_remark** permet d'ajouter des commentaires.

### 4.2.3 Relations et Contraintes

L'architecture relationnelle du système s'appuie sur des liens logiques entre les entités. Les modèles *Admin* et *Employee* établissent des relations 1 :1 avec le modèle *User* de Django, créant une extension cohérente du système d'authentification. Cette approche préserve l'intégrité référentielle grâce à la suppression en cascade : la suppression d'un utilisateur entraîne automatiquement la suppression de son profil *Admin* ou *Employee* associé.

La relation entre *Employee* et *Department* structure l'organisation hiérarchique de l'entreprise. Cette liaison n :1 permet à plusieurs employés d'appartenir au même département tout en garantissant qu'un employé ne peut être rattaché qu'à un seul service. Les demandes de congés sont liées aux utilisateurs via une *ForeignKey*, créant un historique personnalisé pour chaque employé. Le système de notifications utilise également cette relation pour cibler précisément les destinataires.

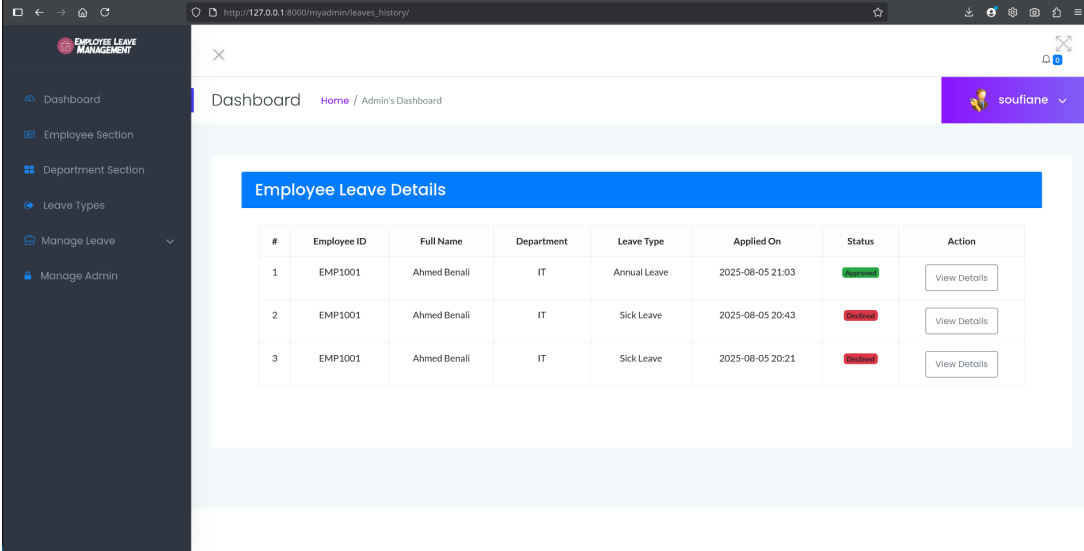
Les contraintes d'intégrité garantissent la cohérence des données. L'unicité des emails d'administrateurs, des codes d'employés et des codes de départements prévient les doublons.

## 4.3 Interface Utilisateur

Le système propose deux interfaces distinctes et isolées selon les rôles utilisateur, garantissant la sécurité et l'ergonomie adaptée à chaque profil. Cette séparation architecturale répond aux exigences de confidentialité et d'efficacité opérationnelle en proposant des fonctionnalités ciblées selon les responsabilités de chaque utilisateur.

### 4.3.1 Interface Employé

L'interface employé privilégie la simplicité et l'autonomie, offrant une vue centralisée de la situation personnelle de congés. Le tableau de bord principal présente un résumé visuel des demandes en cours, de l'historique des congés et des notifications pendantes.



The screenshot displays a web application interface for 'EMPLOYEE LEAVE MANAGEMENT'. On the left is a dark sidebar with navigation links: Dashboard, Employee Section, Department Section, Leave Types, Manage Leave, and Manage Admin. The main content area shows a 'Dashboard' header with a user profile 'soufiane' and a table titled 'Employee Leave Details'. The table lists three leave requests for employee 'Ahmed Benali' from the 'IT' department. The first is an 'Annual Leave' applied on '2025-08-05 21:03' with a 'Pending' status. The next two are 'Sick Leave' requests applied on '2025-08-05 20:43' and '2025-08-05 20:21', both with a 'Rejected' status. Each row includes a 'View Details' button.

| # | Employee ID | Full Name    | Department | Leave Type   | Applied On       | Status   | Action                        |
|---|-------------|--------------|------------|--------------|------------------|----------|-------------------------------|
| 1 | EMP1001     | Ahmed Benali | IT         | Annual Leave | 2025-08-05 21:03 | Pending  | <button>View Details</button> |
| 2 | EMP1001     | Ahmed Benali | IT         | Sick Leave   | 2025-08-05 20:43 | Rejected | <button>View Details</button> |
| 3 | EMP1001     | Ahmed Benali | IT         | Sick Leave   | 2025-08-05 20:21 | Rejected | <button>View Details</button> |

FIGURE 4.2 – historique des congés

### Fonctionnalités Employé

L'espace employé se concentre sur l'autonomie de gestion des congés personnels. La consultation du profil permet à chaque utilisateur de vérifier ses informations personnelles et professionnelles, incluant son rattachement départemental. Le processus de soumission de demandes de congés guide l'utilisateur à travers un formulaire intuitif avec sélection de dates via un calendrier interactif. Le suivi en temps réel affiche le statut de chaque demande (en attente, approuvée, refusée) avec les commentaires administratifs éventuels.

L'historique complet des congés offre une vision rétrospective des périodes d'absence, facilitant la planification personnelle. Le système de notifications informe instantanément l'employé des décisions administratives concernant ses demandes. L'interface intègre également un calendrier personnel visualisant les périodes de congés accordées et les demandes en cours, optimisant la gestion temporelle des projets personnels et professionnels.

The screenshot displays the 'Leave Application Form' within the 'EMPLOYEE LEAVE MANAGEMENT' system. The form is titled 'Leave Application Form' with the subtitle 'Submit your leave request for approval'. It contains the following fields:

- Leave Type\***: A dropdown menu with 'Days in Leave' selected.
- From Date\***: A date input field showing 'mm / dd / yyyy'.
- To Date\***: A date input field showing 'mm / dd / yyyy'.
- Description / Reason\***: A text area for providing a detailed reason for the leave request.

At the bottom of the form, there are two buttons: 'Cancel' and 'Submit Application'. Below the form, there are three tabs: 'Processing Time', 'Auto Approval', and 'Email Updates'.

FIGURE 4.3 – demande congé

### 4.3.2 Interface Manager/Administrateur

Tableau de Bord Administrateur L'interface administrative offre une vision globale et des outils de gestion avancés pour superviser l'ensemble du système. Le tableau de bord centralise les statistiques, les demandes pendantes et les alertes système dans une présentation synthétique facilitant la prise de décision.

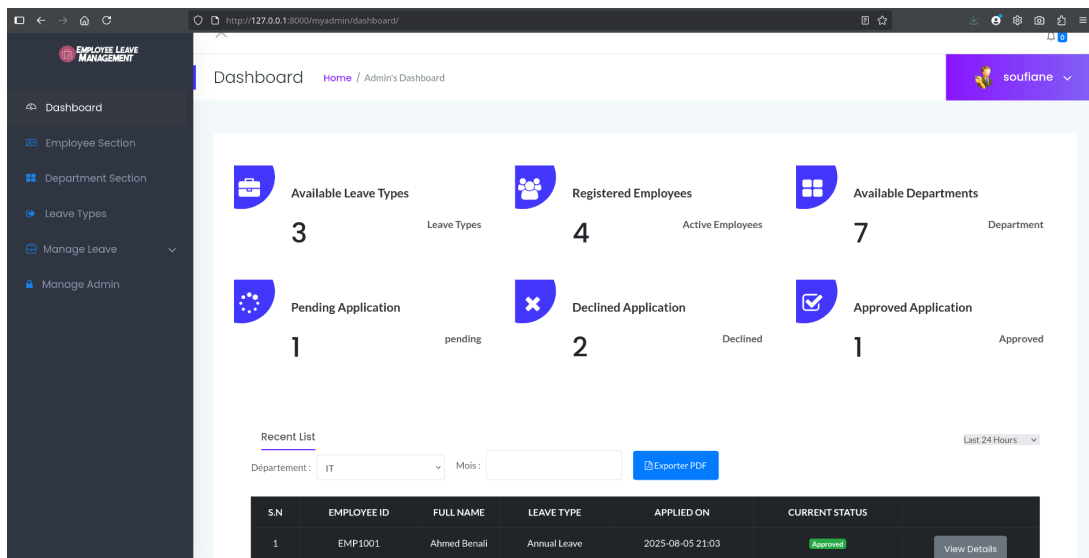


FIGURE 4.4 – dashboard admin

### 4.3.3 Export des rapports en PDF ou CSV

La section suivante illustre le fonctionnement de la fonction de vue responsable de l'export des rapports en formats PDF ou CSV.

```
@login_required
@user_passes_test(is_manager)
def monthly_dept_pdf(request, department_id):

    month = request.GET.get('month')
    if not month:
        return HttpResponse("No month provided")

    try:
        month = int(month)
    except ValueError:
        return HttpResponse("Invalid month")

    year = datetime.now().year
    department = Department.objects.get(id=department_id)

    leaves = Leave.objects.filter(
        employee__employee__department=department,
        posting_date__month=month,
        posting_date__year=year
    )

    template_path = 'admin/monthly_dept_report.html'
    context = {
        'department': department,
        'leaves': leaves,
        'month_label': f"{month}/{year}"
    }

    html = get_template(template_path).render(context)
    response = HttpResponse(content_type='application/pdf')
    pisa.CreatePDF(html, dest=response)
    return response
```

FIGURE 4.5 – Fonction de vue pour l'export des rapports

## Fonctionnalités Administrateur

L'espace administrateur propose des fonctionnalités étendues de gestion organisationnelle. La gestion complète des employés permet les opérations CRUD (Create, Read, Update, Delete) sur les profils, facilitant l'onboarding et l'évolution des carrières. L'administration des départements structure l'organisation avec la possibilité de créer, modifier ou supprimer des services selon les évolutions structurelles.

La génération de rapports et statistiques avancées offre des insights sur l'utilisation des congés par département, par type ou par période. Ces analyses facilitent la planification RH et l'identification des tendances organisationnelles. L'interface permet également la configuration des types de congés disponibles, adaptant le système aux politiques spécifiques de l'entreprise.

Le système de notifications bidirectionnel permet aux administrateurs de communiquer avec les employés, créant un canal de communication officiel tracé. L'export de données en formats multiples (PDF, Excel, CSV) facilite l'intégration avec d'autres systèmes d'information ou la production de rapports réglementaires.



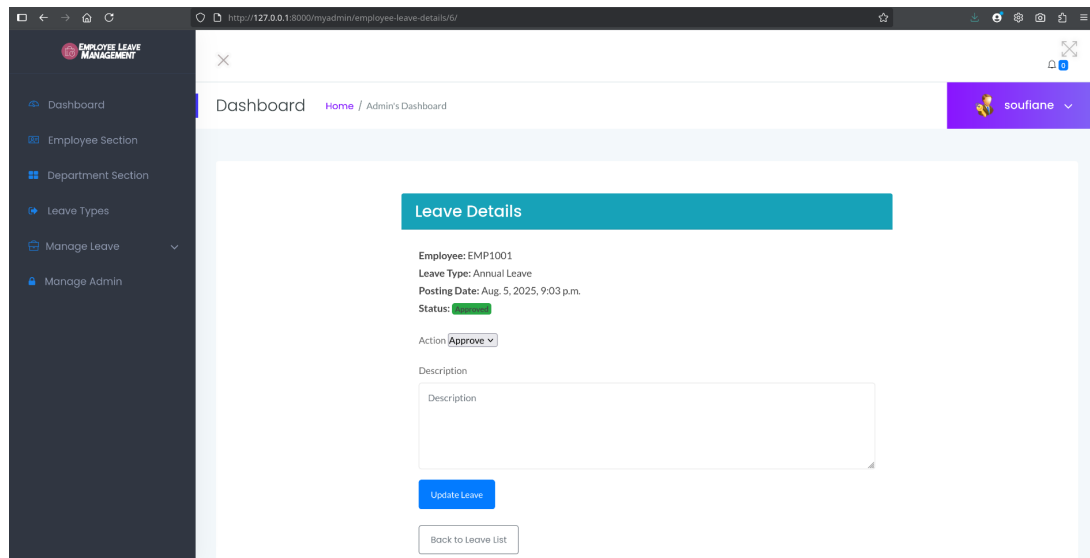


FIGURE 4.6 – gestion des demandes

### 4.3.4 Caractéristiques Communes

#### Sécurité et Isolation

L'architecture sécuritaire du système repose sur une authentification obligatoire via le système Django Auth, garantissant que seuls les utilisateurs autorisés accèdent aux fonctionnalités. L'isolation complète entre les interfaces employé et manager empêche tout accès croisé non autorisé.

#### Expérience Utilisateur

L'ergonomie générale privilégie l'intuitivité et l'efficacité opérationnelle. Le design responsive s'adapte automatiquement aux différents formats d'écran, permettant une utilisation optimale sur ordinateurs, tablettes et smartphones. La navigation contextuelle guide l'utilisateur selon son rôle et ses permissions, réduisant la complexité perçue de l'interface.

## Conclusion Générale

*Ce projet de stage, réalisé en quatrième année à l'EMSI sous l'encadrement de Mme Amnay Meriem, a permis de développer une application web sécurisée et robuste pour une gestion intelligente des congés du personnel. Toute la chaîne est désormais digitalisée, de la demande jusqu'à l'acceptation hiérarchique en passant par les listes par état, Un bon suivi et de gestion grâce aux tableaux de bord et aux notifications automatisées en temps réel. La clairvoyance architecturale basée sur la modélisation des diagrammes UML, la maquettage des interfaces garantissent la scalabilité et la maintenabilité de la solution et une interface responsive.*

Avoir une architecture clairement définie à l'avance grâce aux diagrammes UML et aux maquettes d'interface garantit une solution évolutive et maintenable, tandis que des interfaces réactives offrent une expérience utilisateur fluide, quel que soit l'écran. Ce projet m'a permis de développer mes compétences en développement full-stack, conception de bases de données et gestion de projet agile.

## Perspectives d'Évolution

Évolution possible La solution a plusieurs directions d'évolution possibles. Un module d'administration avancé qui fournirait des *tableaux de bord adaptés pour les rôles des utilisateurs (RH, direction, managers)*, et des rapports interactifs avec filtres dynamiques. Une expérience utilisateur améliorée pourrait être réalisée en ajoutant des notifications push en temps réel : WebSocket ou PWA, et en développant une application mobile native. Le projet ouvre la voie à des évolutions futures, notamment l'intégration d'analyses prédictives basées sur l'intelligence artificielle. Celles-ci permettraient d'anticiper les pics d'absences, d'optimiser la planification des ressources et d'améliorer la gestion stratégique des équipes.

L'introduction d'un workflow plus flexible permettrait des circuits de validation dynamiques et une planification prédictive basée sur l'historique des congés pour anticiper les pics d'absence. La liaison de l'API du workflow avec les services de paie et ERP facilite l'intégration et l'automatisation des échanges de données entre systèmes.

Enfin, une migration vers une architecture microservices avec une API REST complète, l'implémentation d'une authentification multi-facteurs et le respect du RGPD soutiendraient la croissance.

## Bibliographie

**Documentation Technique** Django Software Foundation. *Django Documentation*, version 4.2. Disponible en ligne : <https://docs.djangoproject.com/> (consulté en 2024).

Mozilla Developer Network. *HTML, CSS & JavaScript Documentation*. Disponible en ligne : <https://developer.mozilla.org/> (consulté en 2024).

Bootstrap Team. *Bootstrap Documentation*, version 5.3. Disponible en ligne : <https://getbootstrap.com/> (consulté en 2024).

**Ouvrages de Reference** PERCIVAL, Harry J.W. *Test-Driven Development with Python : Obey the Testing Goat*. 2ème édition, O'Reilly Media, 2017.

HOLOVATY, Adrian et KAPLAN-MOSS, Jacob. *The Definitive Guide to Django : Web Development Done Right*. 2ème édition, Apress, 2009.

MELE, Antonio. *Django 4 By Example : Build powerful and reliable Python web applications from scratch*. 4ème édition, Packt Publishing, 2022.

### RESSOURCES WEB ET TUTORIELS

**Plateformes d'Apprentissage** Real Python Team. *Django Tutorials and Best Practices*. Disponible en ligne : <https://realpython.com/> (consulté en 2024).

Django Girls Foundation. *Django Girls Tutorial*.

Disponible en ligne : <https://tutorial.djangogirls.org/> (consulté en 2024).

FreeCodeCamp. *Full Stack Web Development Course*.

Disponible en ligne : <https://www.freecodecamp.org/> (consulté en 2024).

**Outils et Technologies** Git Documentation Team. *Git Version Control System Documentation*. Disponible en ligne : <https://git-scm.com/doc>

PostgreSQL Global Development Group. *PostgreSQL Documentation*, version 15. Disponible en ligne : <https://www.postgresql.org/docs/> (consulté en 2024).

Visual Studio Code Team. *VS Code Documentation*.

Disponible en ligne : <https://code.visualstudio.com/docs>

**Articles et Publications** SMITH, John et al. "Modern Web Application Security in Django Framework". *Journal of Web Development*, vol. 15, n° 3, 2023, pp. 45-62.

# ANNEX

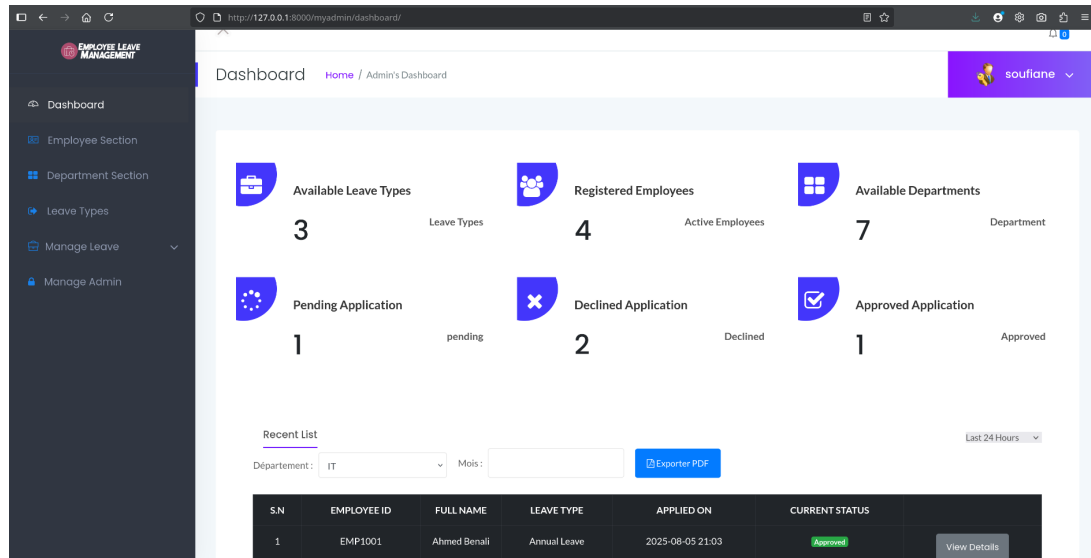


FIGURE 4.7 – historique des congés

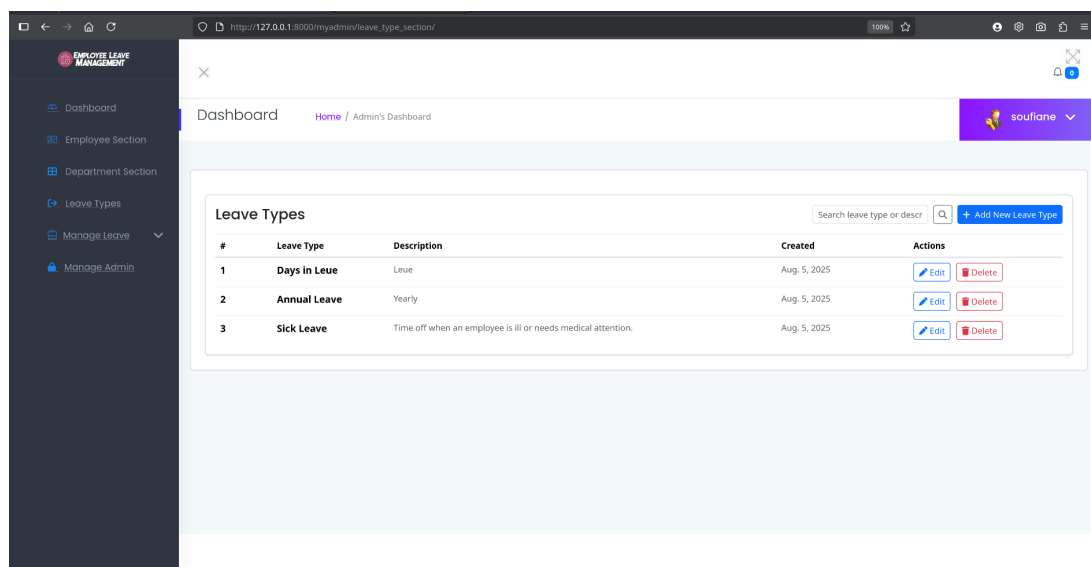


FIGURE 4.8 – Ajouter un type de congé

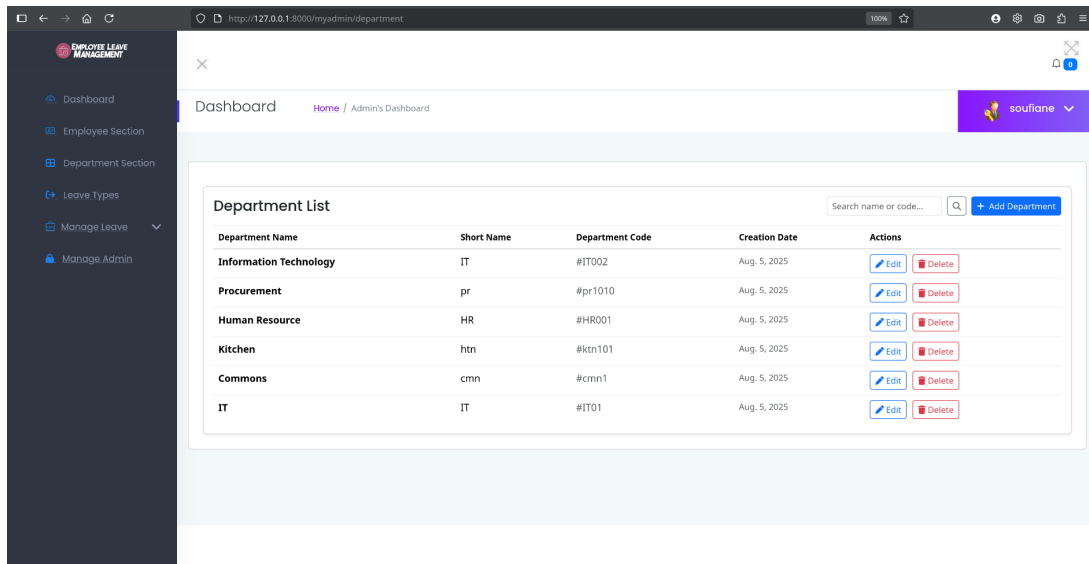


FIGURE 4.9 – gestion des demandes

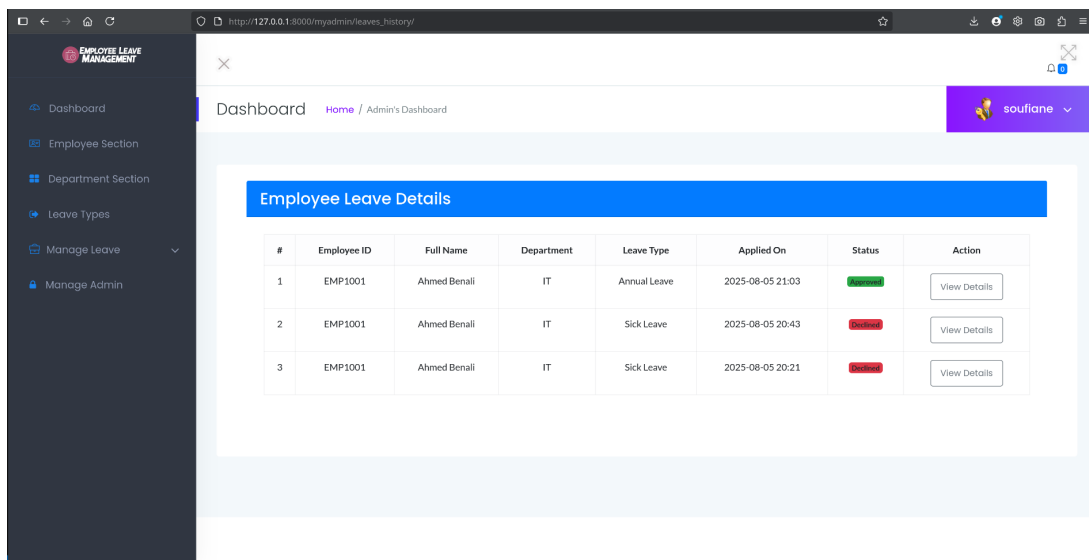


FIGURE 4.10 – historique des congés

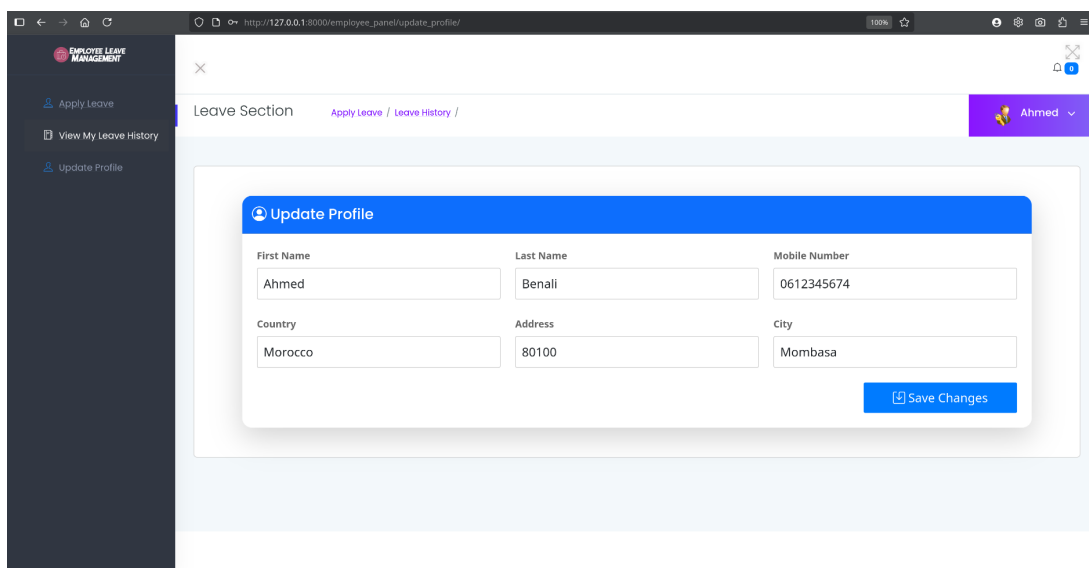


FIGURE 4.11 – profile employé

The screenshot displays a web application for 'EMPLOYEE LEAVE MANAGEMENT'. The interface includes a dark sidebar with navigation links: 'Apply Leave', 'View My Leave History', and 'Update Profile'. The main content area is titled 'Leave Section' and contains a 'Leave Application Form'. The form has a purple header with the title 'Leave Application Form' and the instruction 'Submit your leave request for approval'. The form fields include: 'Leave Type\*' (a dropdown menu currently showing 'Days in Leave'), 'From Date\*' (a date picker showing 'mm / dd / yyyy'), 'To Date\*' (a date picker showing 'mm / dd / yyyy'), and 'Description / Reason\*' (a large text area). Below the text area is a note: 'Please provide a detailed reason for your leave request.' At the bottom of the form are two buttons: 'Cancel' and 'Submit Application'. A footer bar at the bottom of the page contains three sections: 'Processing Time' with a clock icon, 'Auto Approval' with a green checkmark icon, and 'Email Updates' with an envelope icon. The user's name 'Ahmed' is visible in the top right corner.

FIGURE 4.12 – formulaire demande congé