# Cahier des charges
# Application Presence-Tracker


**UNIVERSITE POLYTECHNIQUE DE GITEGA**

**FACULTE DES TECHNOLOGIES DE LINFORMATION ET DE LA COMMUNICATION**

**DEPARTEMENT DE GENIE LOGICIEL**


**Projet académique pour lentreprise UPG**


**Préparé par : Christian MANIRABARUTA et Idris NDIKUMANA**

**Professeur : Ferdinand NIRAGIRA, MA**



**15 mai 2025**

**Version : 1.0**

# Page dapprobation

Ce cahier des charges, préparé pour lentreprise UPG, décrit le développement de lapplication Presence-Tracker. Il est soumis à lapprobation des parties prenantes suivantes :

**Représentant UPG :** À compléter
**Développeurs :** Christian Manirabaruta, Idris Ndikumana
**Date de validation :** À compléter

# Résumé exécutif

Ce cahier des charges définit les spécifications de lapplication Presence-Tracker, conçue pour lentreprise UPG afin dautomatiser la gestion des présences, absences, congés et statuts des employés. Développée par Christian Manirabaruta et Idris Ndikumana, elle inclut une authentification sécurisée, un tableau de bord administrateur, la détection dabsences consécutives, des notifications par email, la gestion des demandes de congé et le suivi des statuts des employés (au travail, empêché, ou indisponible). Basée sur Java EE (Spring Boot), elle garantit sécurité et évolutivité. Le projet, prévu pour mai 2025, sera validé par UPG avant développement.

# Table des matières

# 1 Introduction

## 1.1 Contexte

Lentreprise UPG, spécialisée dans la gestion de services professionnels, cherche à optimiser le suivi des présences des employés dans un contexte de travail en ligne. Lapplication Presence-Tracker, développée par Christian Manirabaruta et Idris Ndikumana, automatise la gestion des présences, absences, congés et statuts des employés, permettant aux employés de signaler leur arrivée, départ, demandes de congé ou justifications, et aux administrateurs de superviser les données, y compris les statuts (actif, empêché, indisponible). Elle inclut des notifications pour les absences consécutives, excluant les week-ends et jours fériés.

## 1.2 Objectifs

- Automatiser la gestion des présences, absences et demandes de congé des employés.
- Fournir un tableau de bord administrateur pour visualiser les statuts, présences et demandes.
- Détecter automatiquement deux absences consécutives, hors week-ends et jours fériés.
- Notifier les employés par email en cas dabsences consécutives.
- Gérer les demandes de congé et justifications entre employés et administrateurs.
- Suivre létat des employés (au travail, empêché, indisponible).
- Assurer une interface intuitive et sécurisée.

# 2 Description fonctionnelle

## 2.1 Fonctionnalités principales

**Authentification :**
- Connexion sécurisée via nom dutilisateur et mot de passe.
- Gestion des rôles : employé (accès limité) et administrateur (accès complet).

**Gestion des présences :**
- Enregistrement des heures de connexion et de déconnexion.
- Stockage des données avec horodatage dans une base de données.

**Tableau de bord administrateur :**
- Affichage des présences, absences, statuts et demandes de congé, mis à jour périodiquement (par exemple, quotidiennement ou à la demande).
- Historique des présences et congés, filtrable par date.

**Détection des absences :**
- Identification automatique de deux absences consécutives, hors week-ends et jours fériés.

**Notifications :**
- Envoi demails pour les absences consécutives.
- Notifications pour les statuts de demandes de congé (accepté, refusé).

**Gestion des jours fériés :**
- Interface administrateur pour gérer les jours fériés (ajout, modification, suppression).

**Gestion des demandes de congé :**
- Soumission de demandes de congé ou justifications par les employés.
- Approbation ou rejet des demandes par les administrateurs.

**Suivi des statuts des employés :**
- Affichage des statuts : au travail, empêché, ou indisponible.
- Mise à jour en temps réel via linterface administrateur.

## 2.2 Exigences non fonctionnelles

- **Sécurité :** Chiffrement des mots de passe, communication HTTPS.
- **Performance :** Temps de réponse inférieur à 2 secondes pour 100 utilisateurs simultanés.
- **Disponibilité :** 99,9 % hors maintenance.
- **Compatibilité :** Navigateurs modernes (Chrome, Firefox, Edge) et support mobile.
- **Évolutivité :** Gestion jusquà 1 000 employés.

# 3 Architecture technique

## 3.1 Environnement de développement

- **IDE :** IntelliJ IDEA.
- **Framework :** Java EE avec Spring Boot.
- **Gestion de version :** GitHub.

## 3.2 Composants techniques

**Backend :**
- **Framework :** Spring Boot pour API REST et logique métier.
- **Persistance :** Spring Data JPA avec Hibernate.
- **Sécurité :** Spring Security pour authentification et autorisation.
- **Notifications :** JavaMail pour envoi demails.
- **Planification :** Spring Scheduler pour vérification des absences.
- **Base de données :** H2 (développement), MySQL/PostgreSQL (production).

**Frontend :**
- **Interface web :** Thymeleaf ou JavaScript (React/Angular).
- **Styles :** Bootstrap ou Tailwind CSS pour design responsive.

**Infrastructure :**
- **Serveur :** Embedded Tomcat (Spring Boot).
- **Hébergement :** Cloud (AWS, Azure) ou serveur dédié.

## 3.3 Modèle de données

**employees :**
- `employeeId` : Identifiant unique (clé primaire).
- `username` : Nom dutilisateur unique.
- `password` : Mot de passe chiffré.
- `email` : Adresse email.
- `role` : Rôle (EMPLOYEE ou ADMIN).

**attendanceRecord :**
- `recordId` : Identifiant unique (clé primaire).
- `employeeId` : Référence à lemployé (clé étrangère).
- `loginTime` : Heure de connexion.
- `logoutTime` : Heure de déconnexion (optionnel).
- `date` : Date de lenregistrement.

**holiday :**
- `holidayId` : Identifiant unique (clé primaire).
- `date` : Date du jour férié.

— `description` : Description du jour férié.

**LeaveRequest :** — `requestId` : Identifiant unique (clé primaire).
— `employeeId` : Référence à lemployé (clé étrangère).
— `startDate` : Date de début du congé.
— `endDate` : Date de fin du congé.
— `reason` : Justification ou motif.
— `status` : Statut (PENDING, APPROVED, REJECTED).
— `submittedAt` : Date de soumission.

**starters :** — `statusId` : Identifiant unique (clé primaire).
— `employeeId` : Référence à lemployé (clé étrangère).
— `status` : État ($\mathrm{AT}_W ORK, PREVENTED, UNAVAILABLE$).`updatedAt` : $Datedemisejourdu$

# 4 Spécifications techniques

## 4.1 API REST

— `POST /api/attendance/login` : Enregistre la connexion dun employé.
— `POST /api/attendance/logout` : Enregistre la déconnexion dun employé.
— `GET /api/attendance/dashboard` : Affiche le tableau de bord (administrateurs).
— `POST /api/holidays` : Ajoute un jour férié (administrateurs).
— `GET /api/holidays` : Liste les jours fériés.
— `POST /api/leave/requests` : Soumet une demande de congé (employés).
— `PUT /api/leave/requests/{requestId}` : Approuve ou rejette une demande (administrateurs).
— `GET /api/leave/requests` : Liste les demandes de congé (administrateurs, employés pour leurs propres demandes).
— `GET /api/status` : Affiche les statuts des employés (administrateurs).
— `PUT /api/status/{employeeId}` : Met à jour le statut dun employé (administrateurs).

## 4.2 Configuration des emails

— **Fournisseur :** Service SMTP (ex. : Gmail).
— **Paramètres :** Hôte, port, authentification, chiffrement TLS.
— **Fréquence :** Vérification quotidienne des absences à 00h00.

## 4.3 Planification

— **Tâche :** Vérification des absences consécutives.
— **Fréquence :** Quotidienne à 00h00.
— **Logique :** Analyse des enregistrements dans `attendanceRecord` des deux derniers jours ouvrables, tenant compte des jours fériés dans `holiday`.

# 5 Contraintes et hypothèses

## 5.1 Contraintes

**Techniques :** — **Framework exclusif :** Spring Boot.

— **Environnement :** IntelliJ IDEA.

**Temporelles :** — **Développement :** Mai 2025.

**Ressources :** — **Équipe :** Christian Manirabaruta, Idris Ndikumana.

— **Budget :** À confirmer par UPG.

### 5.2 Hypothèses

— Accès Internet pour tous les employés.
— Liste initiale des jours fériés fournie par UPG.
— Service de messagerie disponible (ex. : Gmail).
— Compatibilité avec navigateurs modernes et mobiles.

## 6 Plan de développement

### 6.1 Phases du projet

**Analyse et conception (1er–7 mai 2025) :** — Finalisation du cahier des charges.
— Conception de larchitecture et du modèle de données.
— Validation par UPG.

**Développement (8–20 mai 2025) :** — Configuration de lenvironnement (Spring Boot, IntelliJ).
— Implémentation des fonctionnalités (authentification, présences, congés, statuts, notifications).
— Développement de linterface utilisateur.

**Tests (21–25 mai 2025) :** — Tests unitaires avec JUnit et Mockito.
— Tests dintégration.
— Tests utilisateurs avec un échantillon demployés.

**Déploiement (26–30 mai 2025) :** — Configuration de lenvironnement de production.
— Formation des administrateurs.
— Lancement de lapplication.

**Maintenance (à partir de juin 2025) :** — Surveillance des performances.
— Corrections de bugs et mises à jour.

### 6.2 Livrables

— Cahier des charges (ce document).
— Code source, hébergé sur un dépôt GitHub.
— Documentation technique (installation, configuration, API).
— Manuel utilisateur pour employés et administrateurs.
— Application déployée en production.

## 7 Conclusion

Lapplication Presence-Tracker, développée par Christian Manirabaruta et Idris Ndikumana, répondra aux besoins de lentreprise UPG en automatisant la gestion des présences, congés et statuts des employés, tout en améliorant la supervision. Ce cahier des charges

définit les exigences, larchitecture et le plan de développement. Les prochaines étapes incluent la validation par UPG et le lancement du projet en mai 2025.

# Glossaire

**API REST :** Interface de programmation pour la communication entre systèmes via requêtes HTTP.
**Spring Boot :** Framework Java pour applications web et microservices.
**HTTPS :** Protocole sécurisé pour la transmission de données.
**JUnit :** Framework pour tests unitaires en Java.
**MySQL/PostgreSQL :** Systèmes de gestion de bases de données relationnelles.