École des Sciences de l'Information

Conception et développement d'une application pour la gestion des astreintes

Rapport de stage

Réalisé Du 15/07/2025 au 15/08/2025

Réalisé par :
EL BOUANANI Mouad

Encadré par :
Mr.JABLI Zakaria

Année Universitaire : 2025/2026

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réussite de ce stage et à la réalisation de ce projet.

Mes remerciements s'adressent en premier lieu à Monsieur **JABLI Zakaria**, mon directeur de stage au sein d'**OCP**, pour m'avoir fait confiance et m'avoir guidé tout au long de ce projet. Son expertise et ses conseils ont été précieux pour la conception et l'implémentation du système.

Je remercie également l'équipe IT d'**OCP** pour leur accueil chaleureux et leur soutien technique, ainsi que tous les utilisateurs finaux qui ont participé aux tests et à la validation du système.

Je tiens aussi à remercier notre établissement, l'ESI, pour cette opportunité qui m'a permis d'acquérir une expérience de vie inestimable. Je remercie tout particulièrement le directeur de l'école, Monsieur **Salah Eddine BAHJI**, ainsi que l'équipe administrative qui fournit des efforts infinis pour notre bien-être. Mes remerciements vont également aux professeurs, qui nous montrent chaque jour qu'il existe encore des enseignants passionnés par leurs élèves et par leur métier.

Enfin, je remercie ma famille et mes amis pour leur soutien constant et leur compréhension pendant cette période de stage.

Résumé

Le présent document décrit le développement d'une application de gestion des astreintes dédiée aux weekends et jours fériés au sein du groupe OCP. Cette initiative s'inscrit dans une démarche de modernisation des processus opérationnels et répond aux besoins spécifiques d'un environnement industriel nécessitant une surveillance continue, même en dehors des heures ouvrables traditionnelles.

L'objectif principal de ce projet est de centraliser et d'automatiser la planification des gardes, tout en tenant compte des indisponibilités des collaborateurs et en assurant un système d'escalade efficace en cas d'absence. Cette approche vise à éliminer les lacunes du processus manuel actuel, qui présente des risques d'erreurs humaines et des difficultés de coordination entre les différents services et sites du groupe OCP.

Cette solution vise à améliorer la continuité opérationnelle des différents sites OCP, en facilitant la gestion des rotations et en garantissant la transmission des alertes nécessaires. Elle permet également d'optimiser l'allocation des ressources humaines, d'assurer une traçabilité complète des interventions et de réduire significativement les délais de réaction en cas d'incident critique nécessitant une intervention immédiate.

L'application a été développée avec la MERN stack (MongoDB, Express.js, React, Node.js) et déployée via conteneurs Docker. Cette architecture moderne garantit une interface responsive, un système d'authentification robuste avec gestion différenciée des rôles, et des fonctionnalités d'export multi-format répondant aux exigences réglementaires du secteur minier et chimique.

Abstract

This document describes the development of an on-call duty management application dedicated to weekends and public holidays within the OCP Group. This initiative is part of an operational process modernization approach and addresses the specific needs of an industrial environment requiring continuous monitoring, even outside traditional working hours.

The main objective of this project is to centralize and automate shift scheduling, while taking into account employee unavailabilities and ensuring an effective escalation system in case of absence. This approach aims to eliminate the gaps in the current manual process, which presents risks of human errors and coordination difficulties between different services and sites of the OCP Group.

This solution aims to improve the operational continuity of the various OCP sites by facilitating rotation management and ensuring the transmission of necessary alerts. It also enables the optimization of human resource allocation, ensures complete traceability of interventions, and significantly reduces response times in case of critical incidents requiring immediate intervention.

The application was developed using the MERN stack (MongoDB, Express.js, React, Node.js) and deployed via Docker containers. This modern architecture guarantees a responsive interface, a robust authentication system with differentiated role management, and multi-format export functionalities meeting the regulatory requirements of the mining and chemical sector.

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Contexte et enjeux

Le projet de développement d'une application de gestion des astreintes s'inscrit dans le cadre des besoins opérationnels du Groupe OCP. Ce dernier, acteur majeur du secteur minier et chimique au niveau national et international, exploite des infrastructures critiques nécessitant un suivi permanent, y compris les weekends et jours fériés.

Actuellement, la planification et le suivi des astreintes sont réalisés de manière partiellement manuelle, ce qui entraîne des risques d'erreurs, des difficultés de coordination et un manque de visibilité sur la disponibilité des collaborateurs. Ces limitations peuvent avoir un impact direct sur la réactivité en cas d'incident, et donc sur la continuité des opérations.

Les enjeux principaux de ce projet sont donc :

— **Assurer la continuité de service** en garantissant la présence d'équipes de garde disponibles à tout moment.

— **Automatiser et fiabiliser la planification** afin de réduire les erreurs humaines et le temps consacré à l'organisation des plannings.

— **Faciliter la communication et l'escalade** des incidents pour une prise en charge rapide et efficace.

— **Offrir une meilleure visibilité** aux responsables sur les rotations et disponibilités des collaborateurs.

La mise en place d'une telle solution contribue à améliorer l'efficacité opérationnelle et à renforcer la capacité du Groupe OCP à répondre rapidement aux imprévus, tout en optimisant la gestion de ses ressources humaines.

Ce projet repose sur l'architecture **MERN stack** (MongoDB, Express.js, React, Node.js), qui permet de développer l'ensemble des couches de l'application en JavaScript, offrant ainsi une grande cohérence technique, une rapidité de développement et une forte maintenabilité.

1.2 Objectifs du projet

À partir des enjeux identifiés, l'objectif principal de ce projet est de concevoir et de déployer une application web permettant de gérer de manière centralisée et automatisée les astreintes du Groupe OCP pour les weekends et jours fériés.

Les objectifs spécifiques sont les suivants :

— **Centraliser la gestion des plannings** afin que toutes les informations relatives aux

rotations et disponibilités soient accessibles depuis une plateforme unique.
- **Intégrer un système de gestion des indisponibilités** permettant aux collaborateurs de signaler leurs absences et aux responsables de réajuster les plannings en conséquence.
- **Mettre en place un mécanisme d'escalade automatisé** pour garantir la prise en charge rapide des incidents en cas d'indisponibilité d'un agent d'astreinte.
- **Assurer la traçabilité et l'historique des gardes** pour faciliter les analyses et le suivi administratif.
- **Déployer une solution moderne et évolutive** basée sur Node.js, React et MongoDB (NoSQL database), encapsulée dans des conteneurs Docker pour un déploiement simplifié.

La réalisation de ces objectifs permettra d'optimiser la gestion des ressources humaines affectées aux astreintes, de réduire les délais d'intervention et de renforcer la fiabilité des opérations critiques au sein du Groupe OCP.

1.3 Méthodologie et démarche de réalisation

La mise en œuvre du projet a suivi une démarche structurée, inspirée des bonnes pratiques de gestion de projet logiciel, afin de garantir la qualité et la pertinence de la solution finale.

1.3.1 Analyse des besoins

Une phase d'étude initiale a été réalisée pour recueillir et formaliser les besoins fonctionnels et techniques auprès des parties prenantes (responsables d'astreinte, administrateurs systèmes et équipes opérationnelles). Cette étape a permis de :
- Identifier les fonctionnalités clés attendues (planification, gestion des indisponibilités, escalade automatique).
- Définir les contraintes techniques et organisationnelles.
- Établir un cahier des charges clair et validé.

1.3.2 Conception de la solution

Sur la base des besoins identifiés, une conception détaillée a été réalisée comprenant :
- La modélisation des données adaptée à une base NoSQL avec MongoDB.
- La définition de l'architecture logicielle en couches (front-end, back-end, base de données).
- La conception des interfaces utilisateurs avec un accent sur l'ergonomie et la simplicité.

1.3.3 Développement et intégration

Le développement du projet a été mené en suivant une approche **itérative et incrémentale**, inspirée des méthodes agiles, permettant d'assurer une validation progressive des fonctionnalités au fur et à mesure de leur implémentation. Chaque itération comprenait une phase d'analyse, de conception, de codage et de tests, avec des livrables intermédiaires validés par l'équipe encadrante.

Le choix de la **MERN stack** (MongoDB, Express.js, React, Node.js) s'est imposé pour plusieurs raisons :
- **Cohérence technologique** : l'ensemble du développement, du front-end au back-end, est réalisé en JavaScript, facilitant la maintenance et la compréhension du code.

— **Performance et scalabilité** : Node.js assure un traitement rapide des requêtes grâce à son modèle asynchrone, tandis que MongoDB permet de gérer des données volumineuses de manière flexible.
— **Modularité** : chaque couche (front-end, back-end, base de données) est indépendante et peut évoluer séparément.

Le **front-end**, développé en React, offre une interface utilisateur dynamique, réactive et responsive, intégrant des composants réutilisables et une gestion optimale de l'état via des hooks et éventuellement Redux pour les fonctionnalités complexes. Le **back-end**, basé sur Node.js et Express.js, expose une API REST sécurisée, gère la logique métier et interagit avec la base de données MongoDB. Cette API est conçue pour supporter facilement l'ajout futur de nouvelles fonctionnalités ou l'intégration avec d'autres systèmes.

La **base de données** MongoDB, de type NoSQL, a été choisie pour sa flexibilité de schéma et sa capacité à évoluer rapidement selon les besoins du projet. Les données sont structurées en collections optimisées pour les opérations de lecture/écriture fréquentes, ce qui correspond aux exigences de l'application.

Pour assurer la portabilité et simplifier le déploiement, chaque composant du projet est encapsulé dans un **conteneur Docker**, garantissant un environnement homogène entre le développement, les tests et la production. L'orchestration via `docker-compose` permet de gérer facilement les dépendances et la communication entre les services (base de données, back-end, front-end).

Enfin, l'intégration continue et la gestion de version du code source sont assurées par **Git et GitHub**, facilitant la collaboration et la traçabilité des modifications. Des tests unitaires et d'intégration sont réalisés à chaque itération pour vérifier la conformité aux spécifications fonctionnelles et techniques.

1.3.4 Tests et validation

Des tests unitaires, d'intégration et fonctionnels ont été réalisés afin d'assurer :
— Le bon fonctionnement des fonctionnalités principales.
— La robustesse et la sécurité de l'application.
— Le respect des exigences définies dans le cahier des charges.

1.3.5 Mise en production et formation

La solution a été déployée dans l'environnement cible et une formation a été dispensée aux utilisateurs finaux afin d'assurer une prise en main rapide et efficace.

Chapitre 2

Présentation de l'Organisme

2.1 Group OCP

OCP a été créé en 1920 en tant que Office Chérifien des Phosphates. Nous avons démarré notre activité avec l'exploitation d'une première mine à Khouribga. Nos activités s'étendent aujourd'hui sur cinq continents et nous travaillons tout au long de la chaîne de valeur des phosphates que ce soit dans l'extraction minière, la transformation industrielle ou encore l'éducation et le développement de communautés.

OCP a démarré sa production en mars 1921 à Khouribga et exporté ses premiers produits depuis le port de Casablanca plus tard la même année. Une deuxième mine a été ouverte à Youssoufia en 1931 ainsi qu'une troisième plus tard en 1976 à Benguerir. Le Groupe OCP s'est ensuite diversifié en investissant dans la transformation des phosphates et en implantant des sites chimiques à Safi (1965) et Jorf Lasfar (1984).

En 2008, l'Office Chérifien des Phosphates est devenu OCP Group S.A., propriété de l'Etat marocain et du Groupe Banque Populaire. Notre success-story a renforcé notre relation avec nos communautés, ancré notre engagement à réduire l'impact de nos activités sur l'environnement et motivé nos partenariats avec des entreprises locales et internationales innovantes.

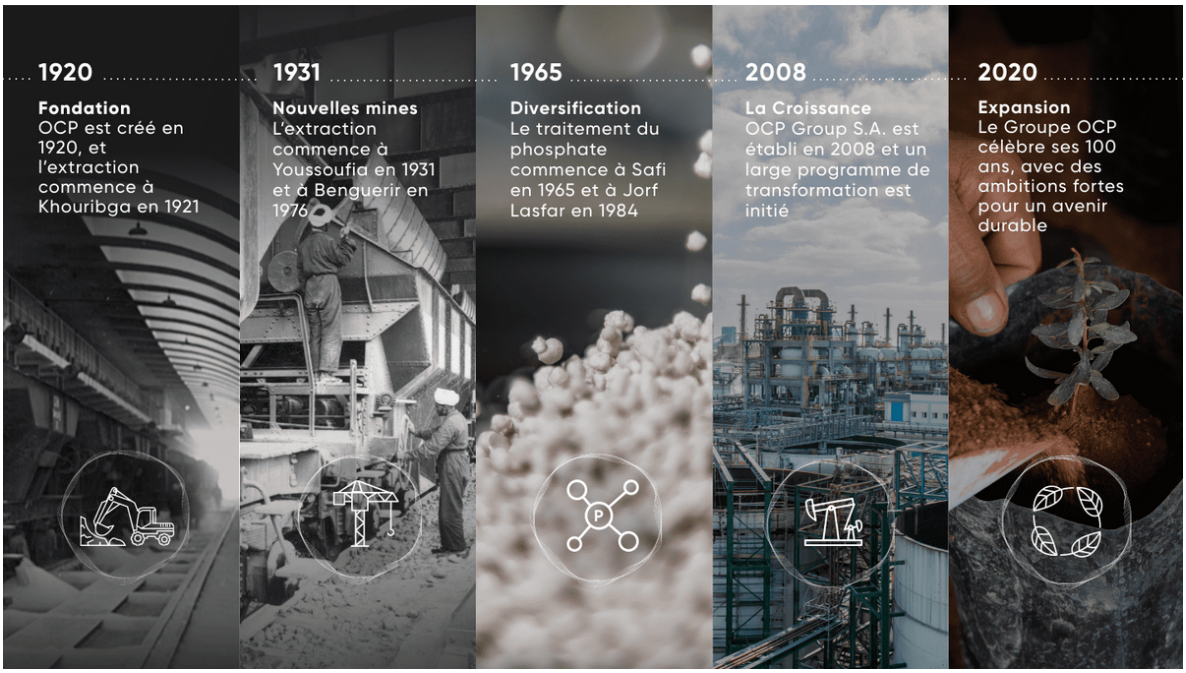

FIGURE 2.1 – Historique de GROUP OCP

2.2 Mission : Nourrir le sol pour nourrir la planète

En tant que garant de 70% des réserves mondiales en phosphates, nous jouons un rôle vital dans l'accompagnement des agriculteurs afin de produire suffisamment et répondre aux besoins alimentaires des prochaines décennies.

Notre rôle est de fournir un nombre suffisant de produits à base de phosphate en vue de faire face à la demande mondiale croissante en engrais et fertilisants. Ceci présuppose une compréhension réelle des besoins des sols et des cultures et l'accompagnement des agriculteurs pour un usage raisonné et durable des ressources. Ces objectifs doivent être atteints tout en veillant à minimiser l'impact de nos activités sur notre environnement.

Nous réalisons d'ores et déjà de grands progrès à ce niveau. Nous investissons dans l'innovation afin de rendre plus efficace l'usage actuel des phosphates. Nous cartographions les sols à travers le monde et développons une approche sur-mesure de nos produits afin de servir les besoins de tout un chacun.

Chez OCP, notre mission est de nourrir le sol pour nourrir la planète.

2.3 Vision : Une croissance durable pour tout le monde

Chez OCP nous croyons que tout est interconnecté.

Nous sommes le premier fournisseur mondial en produits phosphatés. Notre ambition, portée par notre innovation, est de nourrir la planète afin de répondre aux enjeux de sécurité alimentaire.

Nous sommes convaincus, qu'en rationnalisant nos ressources naturelles, nous pouvons soutenir plus de cultures, d'agriculteurs, de communautés et d'environnements naturels.

Au cœur de notre vision, se trouve l'humain. Nous voulons assurer une alimentation durable tout en offrant plus d'opportunités à notre écosystème et ce, en encourageant le renforcement des capacités par la création de compétences et l'éducation.

Nous sommes convaincus que cette interconnexion et cet équilibre permettront d'assurer un partage de valeur et une croissance durable pour tout le monde.

2.3.1 La valeur du phosphate

Le phosphate est une ressource précieuse. Nous considérons sa valeur non seulement en tant que minéral essentiel pour la croissance des plantes, mais aussi en tant que catalyseur de changement positif et de développement au Maroc, en Afrique et à travers le monde.

2.4 Organigramme

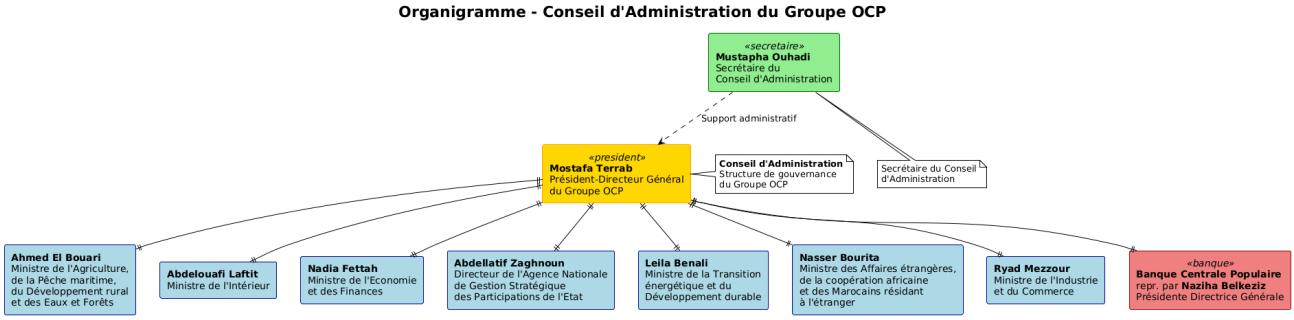

FIGURE 2.2 – Organigramme

Chapitre 3

Analyse

3.1 Analyse fonctionnelle

La gestion actuelle des astreintes au sein du Groupe OCP présente plusieurs limites. La planification est effectuée en grande partie de manière manuelle, ce qui entraîne des risques d'erreurs, un manque de visibilité sur la disponibilité des agents, et des difficultés de coordination lors des weekends et jours fériés.

Les fonctionnalités principales identifiées pour la nouvelle application sont :

— Gestion des plannings d'astreinte (création, modification, suppression).
— Gestion des indisponibilités des collaborateurs.
— Mécanisme d'escalade automatique en cas d'absence d'astreinte.
— Notifications et alertes pour informer les agents concernés.
— Consultation de l'historique des gardes pour la traçabilité.

3.2 Analyse technique

Le choix technologique adopté est la **MERN stack**, qui combine MongoDB, Express.js, React et Node.js, garantissant un développement fluide et homogène :

— **Back-end** : Node.js avec un framework adapté pour la gestion des API REST (par exemple Express.js).
— **Base de données** : MongoDB, base NoSQL, pour une gestion flexible des données.
— **Front-end** : React pour une interface utilisateur dynamique et réactive.
— **Déploiement** : conteneurs Docker pour faciliter la portabilité et la mise en production.

Les contraintes techniques à prendre en compte incluent :

— Sécurité des données et gestion des accès utilisateurs.
— Performance et réactivité de l'application.
— Compatibilité multi-navigateurs et multi-appareils.

3.3 Spécifications des besoins

Les besoins fonctionnels identifiés sont :

— Centraliser la gestion des plannings et des disponibilités.

— Permettre aux collaborateurs de déclarer leurs indisponibilités.
— Automatiser la planification en tenant compte des contraintes.
— Assurer une communication rapide via des alertes et notifications.
— Garantir la traçabilité des actions et des modifications.

Les besoins non fonctionnels comprennent :

— Une interface simple, intuitive et accessible.
— Une disponibilité élevée pour garantir la continuité du service.
— Un système sécurisé avec gestion des droits d'accès.
— Une solution évolutive pour intégrer de futures fonctionnalités.
— Une architecture scalable grâce à MongoDB et Node.js, permettant de gérer une montée en charge progressive.

Chapitre 4

Conception

4.1 Architecture générale

L'architecture de l'application de gestion des astreintes repose sur une séparation claire entre les différentes couches logicielles pour assurer modularité, maintenabilité et évolutivité.

Le système est composé de trois couches principales :

— **Couche Front-end** : développée avec React, elle offre une interface utilisateur dynamique et réactive permettant aux utilisateurs de consulter les plannings, déclarer leurs indisponibilités, et gérer les alertes.

— **Couche Back-end** : basée sur Node.js et Express.js, elle expose une API REST sécurisée pour la gestion des données, la logique métier et la communication avec la base de données.

— **Base de données** : MongoDB, une base NoSQL, stocke les informations relatives aux utilisateurs, plannings, indisponibilités et historiques, assurant flexibilité et performance.

Le déploiement s'effectue dans des conteneurs Docker, facilitant la portabilité et la mise en production.

4.2 Diagrammes UML

4.2.1 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation présente les interactions principales entre les utilisateurs et le système. Les acteurs identifiés sont :

— **Ingénieur** : responsable de la réalisation technique, il peut consulter son planning et déclarer ses indisponibilités.

— **Collaborateur** : utilisateur général du système qui consulte les plannings et reçoit les notifications.

— **Chef de services** : supervise les équipes, valide les plannings et assure la coordination.

— **Administrateur national** : gère les paramètres globaux et la configuration du système à l'échelle nationale.

— **Chef de secteur** : planifie les astreintes spécifiques à son secteur et veille à leur bonne exécution.

Les cas d'utilisation majeurs incluent :

— **Déclarer indisponibilité** : permettre aux utilisateurs de signaler leurs absences.
— **Consulter planning** : offrir un accès aux horaires de garde.
— **Envoyer notification** : alerter les utilisateurs concernés en cas d'événements importants.
— **Valider planning** : garantir la validation des plannings par les responsables.
— **Planifier astreintes secteurs** : organiser les rotations spécifiques à chaque secteur.

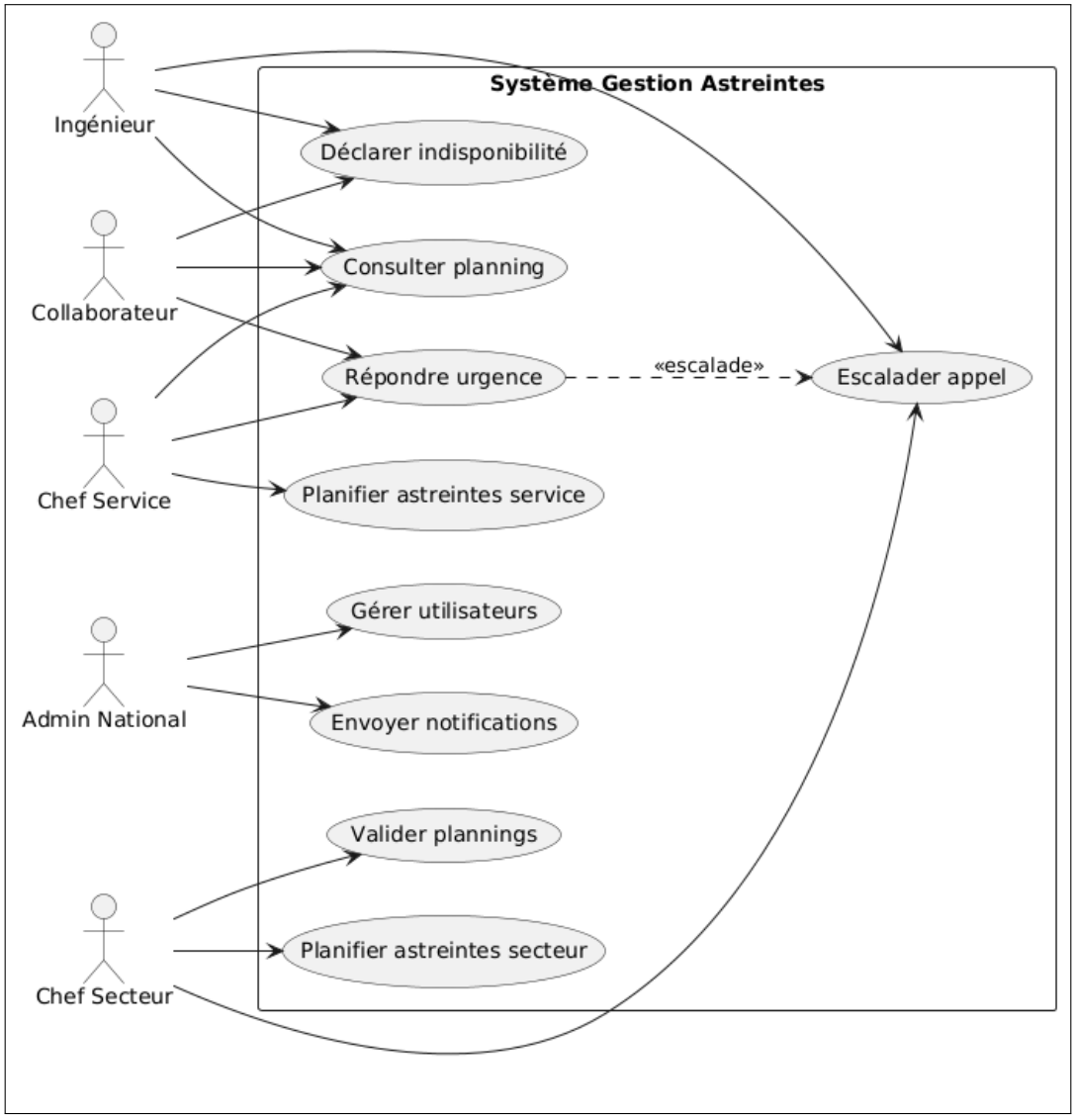

FIGURE 4.1 – Diagramme de cas d'utilisation

4.2.2 Diagramme de classes

Ce diagramme de classes illustre l'architecture technique du système de gestion des astreintes OCP. Il présente les principales entités métier et leurs relations. Parmi les classes essentielles, on trouve :

- **Site** : Représentant les sites industriels OCP (attributs : id, nom, ville, localisation).
- **Secteur** : Division organisationnelle d'un site (traitement, extraction, etc.).
- **Service** : Unité opérationnelle avec contraintes de personnel minimum (collabMin).
- **Utilisateur** : Classe parente avec différentes spécialisations (`Collaborateur`, `ChefService`, `Ingenieur`, `ChefSecteur`, `Administrateur`).
- **PlanningAstreinte** : Gérant la génération et validation des plannings.
- **AffectationGarde** : Liant les utilisateurs aux créneaux d'astreinte.
- **AppelUrgence** : Gérant le processus d'escalade des incidents.
- **Disponibilite** : Traitant les indisponibilités des collaborateurs.
- **Notification** : Système d'alertes multi-canaux.

Ces classes sont liées par des associations reflétant les dépendances fonctionnelles et la hiérarchie organisationnelle OCP, notamment :

- Un **Site** contient plusieurs `Secteurs`.
- Un **Secteur** contient plusieurs **Services**.
- Un **Service** associe plusieurs **Collaborateurs** et un **ChefService**.
- Le système d'escalade suit la hiérarchie : **Collaborateur** $\rightarrow$ **ChefService** $\rightarrow$ **Ingenieur** $\rightarrow$ **ChefSecteur**.

Ce modèle respecte les règles métier OCP avec la double planification (service + secteur) et le système d'escalade à trois niveaux.

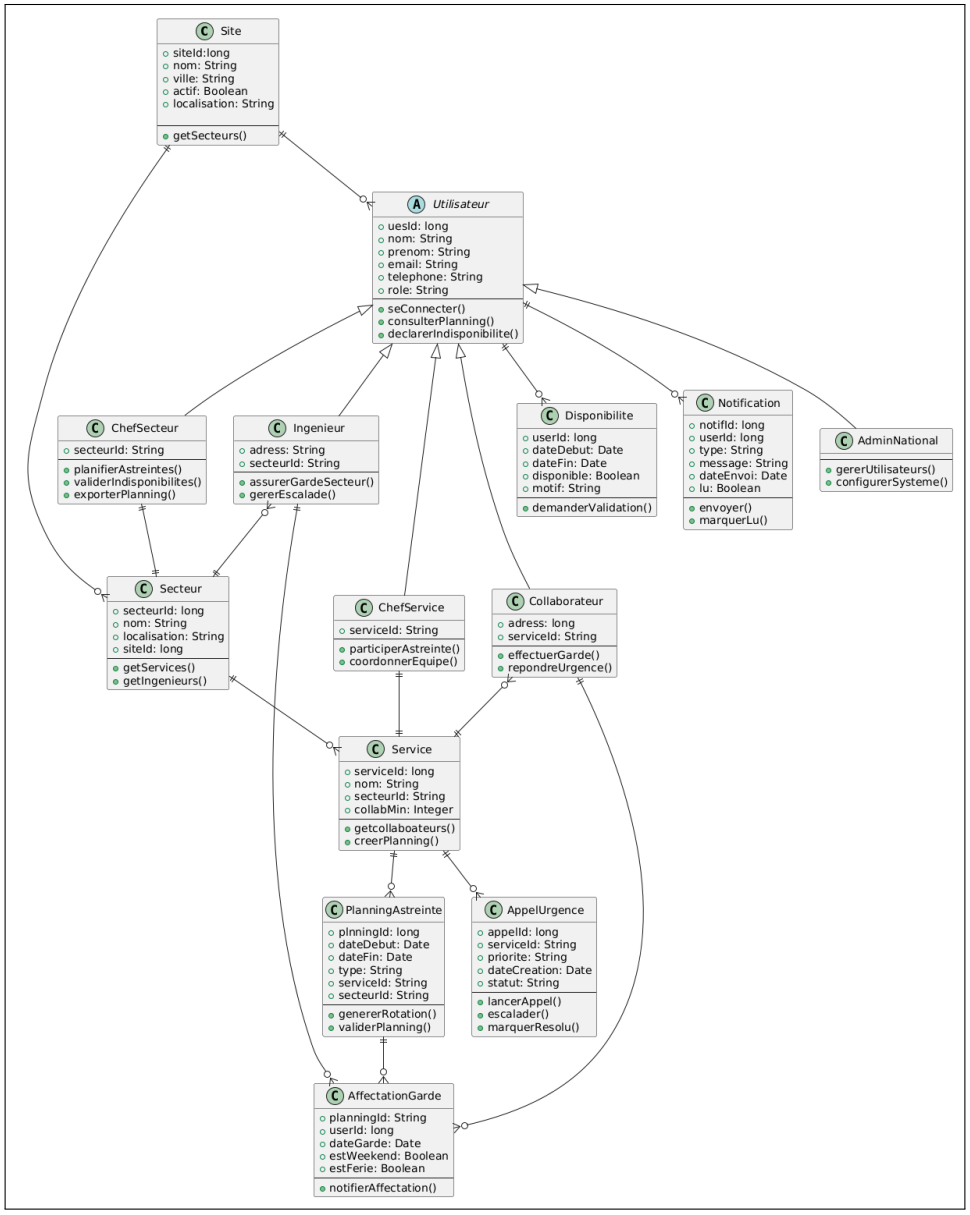

FIGURE 4.2 – Diagramme de classes

4.2.3 Diagramme d'activité

Ce diagramme d'activité illustre le processus de planification des astreintes impliquant le Chef de Secteur et le Chef de Service. Le flux opérationnel comprend les étapes suivantes :

— Le **Chef de Secteur** initie le processus en accédant au module de planification et en sélectionnant la période concernée.
— Le **Système** charge automatiquement les données du personnel et calcule les besoins minimum en personnel pour la période.
— Le **Chef de Service** intervient pour :

  - Créer la rotation des ingénieurs
  - Vérifier le respect des règles de double planification
  - Planifier au niveau du service
  - S'inclure dans la rotation comme exigé par les règles métier
- Le processus inclut une phase de résolution des conflits avant la validation finale.
- Le Chef de Secteur valide le planning final, qui est sauvegardé dans le système.
- Des notifications sont automatiquement envoyées aux personnels concernés pour les informer de leurs affectations.

Ce processus assure le respect des règles métier OCP, notamment la double planification obligatoire et la participation active du Chef de Service aux astreintes.

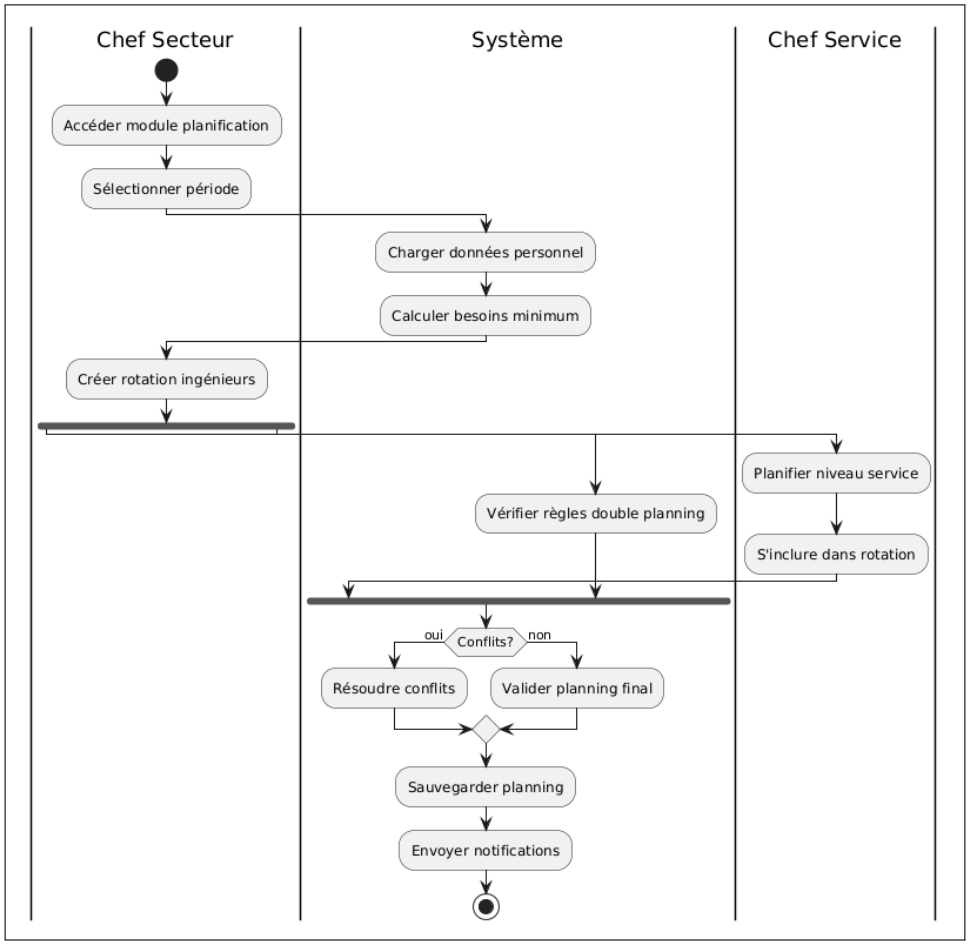

FIGURE 4.3 – Diagramme d'activité

4.2.4 Scénarios

Cette section présente un scénario clé lié à l'utilisation de l'application. Ce scénario permet de visualiser le cheminement des actions et des décisions dans différents contextes opérationnels.

Scénario 1 : Consultation et gestion des indisponibilités

Objectif : Permettre au système de détecter et gérer les indisponibilités du personnel lors d'une alerte.

Étapes :

1. Un incident ou une panne est signalé par un client.
2. Le système vérifie la disponibilité des employés affectés à la zone concernée.
3. Si l'employé prévu est disponible :
   (a) Le système envoie une notification immédiate.
   (b) L'employé intervient sur le site.
4. Si l'employé est indisponible :
   (a) Le système recherche un remplaçant dans la même équipe.
   (b) En cas d'absence de remplaçant, le chef de service est contacté.
   (c) Si aucun intervenant n'est disponible, le problème est escaladé au responsable régional.
5. L'incident est traité et le statut est mis à jour dans la base de données.

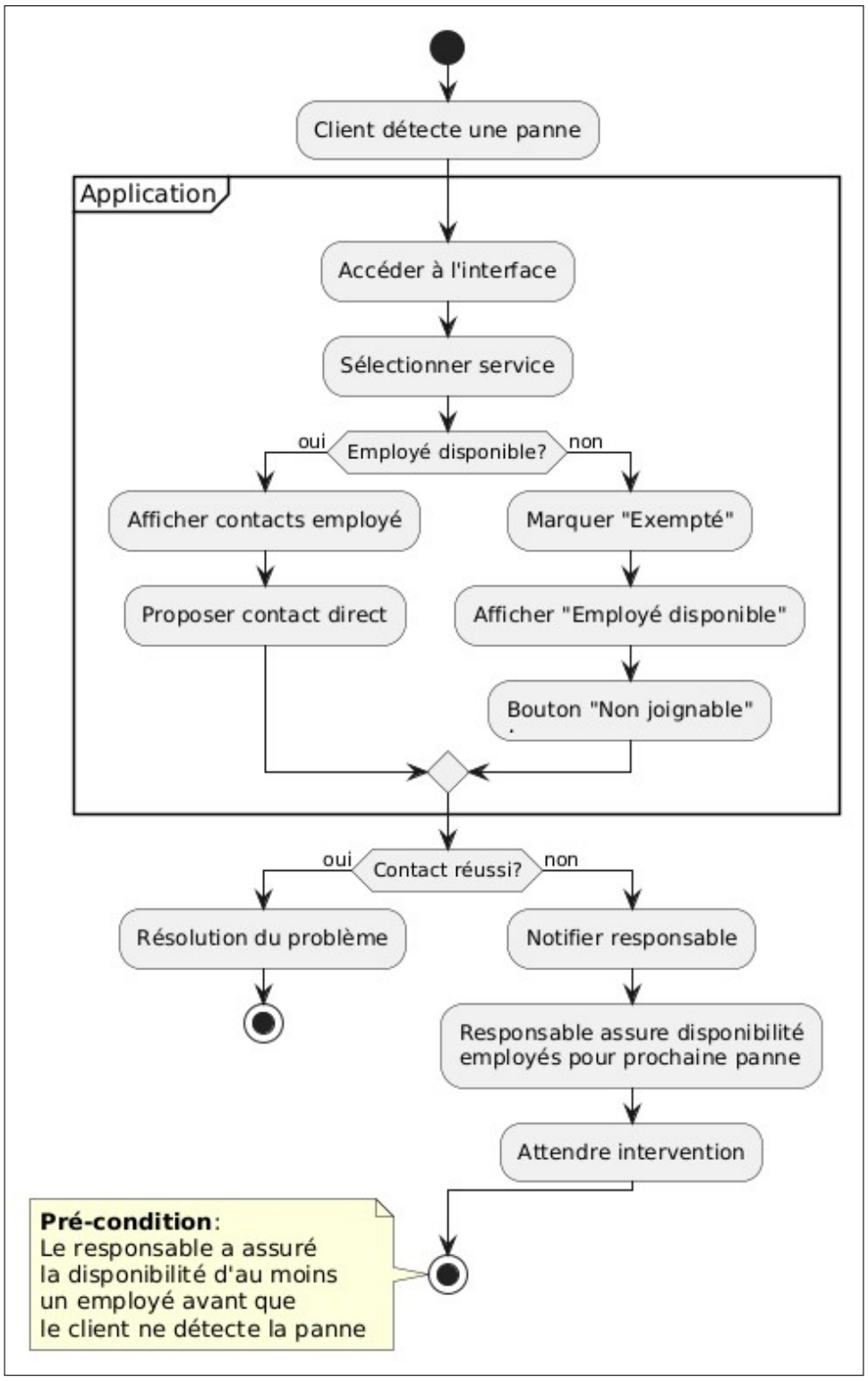

FIGURE 4.4 – Scénario 1 — Consultation et gestion des indisponibilités

Chapitre 5

Développement

5.1 Technologies utilisées

Le projet s'appuie sur la **MERN stack**, une pile technologique moderne composée de **MongoDB, Express.js, React et Node.js**, particulièrement adaptée au développement web complet (*full-stack*) en JavaScript. Cette approche permet une communication fluide entre les différentes couches de l'application, de la base de données à l'interface utilisateur. Les outils utilisés sont :

- **Node.js** avec **Express.js** pour le développement du serveur backend et la gestion des API REST.
- **React** pour la création d'une interface utilisateur dynamique et responsive.
- **MongoDB** comme base de données NoSQL, assurant le stockage des informations liées aux utilisateurs, plannings et notifications.
- **Docker** pour containeriser l'application et faciliter son déploiement et sa portabilité.

5.2 Implémentation des fonctionnalités

5.2.1 Gestion des astreintes

La gestion des astreintes comprend la création, la modification et la consultation des plannings des rotations de garde. Le système prend en compte les indisponibilités déclarées par les agents et ajuste automatiquement les affectations. Les responsables peuvent valider ou modifier les plannings via une interface dédiée.

Page d'Accueil - Application de Gestion des Astreintes OCP

Cette page d'accueil constitue le portail central de l'application de gestion des astreintes weekends et jours fériés de l'OCP, offrant une vision unifiée des plannings et un accès rapide aux fonctionnalités essentielles. Elle permet aux utilisateurs de consulter les astreintes planifiées pour les weekends (Samedi et Dimanche) et jours fériés marocains grâce à des filtres intuitifs par site, secteur et service, tout en fournissant un accès direct au module de déclaration de pannes pour initier immédiatement le processus d'escalade automatique. l'interface affiche déjà les données clés comme le nombre de

secteurs (43) et collaborateurs (17) concernés, et intègre un appel à l'authentification pour déverrouiller l'ensemble des fonctionnalités métier réservées aux différents profils (collaborateurs, chefs de service, ingénieurs et administrateurs). Conçue pour être responsive et orientée action, cette page sert de point d'entrée unique pour garantir une continuité opérationnelle 24/7 durant les périodes non-ouvrées.

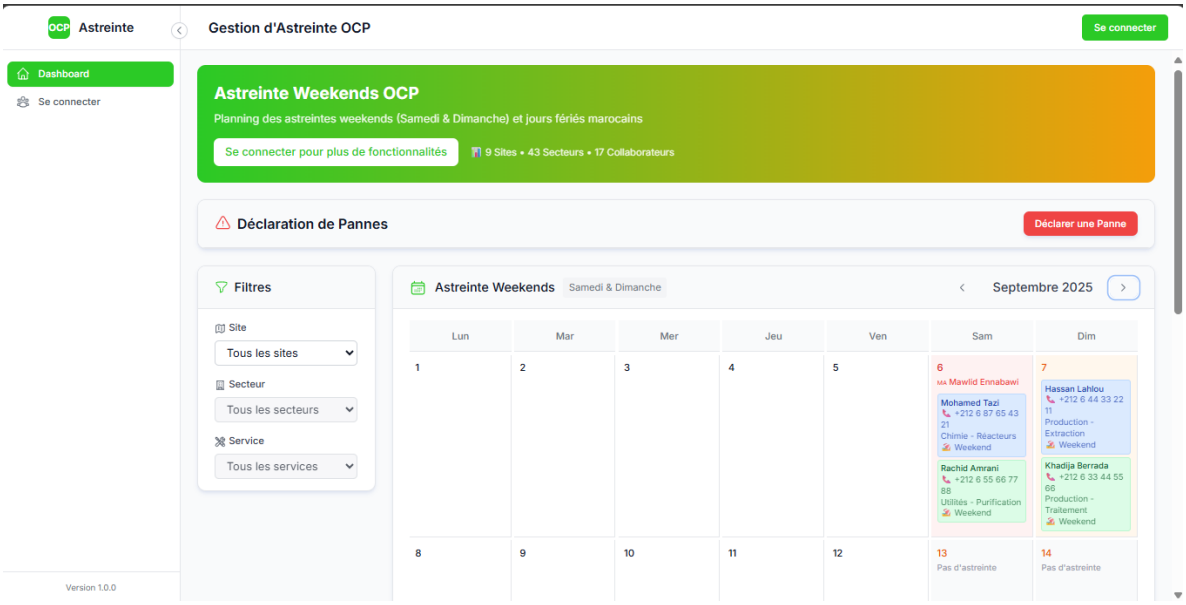

FIGURE 5.1 – Page d'accueil

Page de Déclaration de Panne - Application de Gestion des Astreintes OCP

Cette interface permet aux utilisateurs de déclarer une panne de manière structurée et efficace. Le formulaire oblige à saisir un titre et une description détaillée de l'incident, tandis que des menus déroulants permettent de préciser le type (Technique, Urgence, Moyenne) et la priorité (Normale par défaut). Deux boutons d'action ("Annuler" et "Déclarer") offrent respectivement la possibilité d'abandonner la procédure ou de soumettre le rapport, déclenchant ainsi le processus d'alerte et d'escalade vers les équipes d'astreinte concernées.

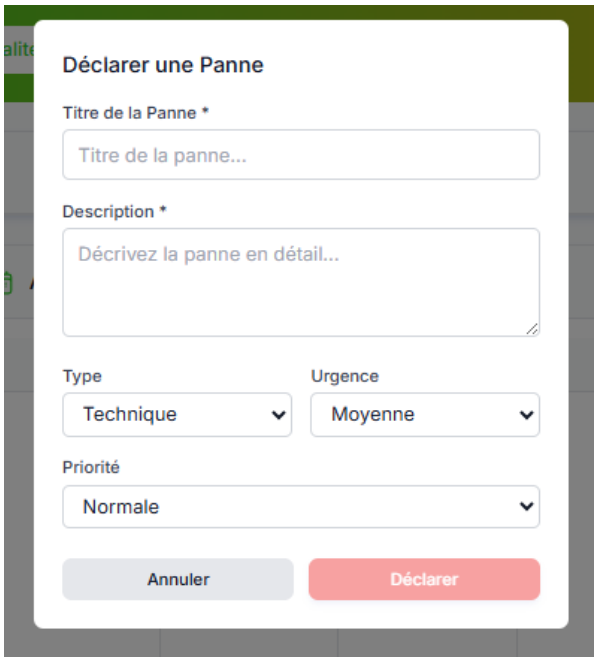

FIGURE 5.2 – Déclarer une panne

Fiche de Contact d'Astreinte - Application de Gestion des Astreintes OCP

Cette interface présente les détails complets du personnel d'astreinte, ici Khadija Berrada, collaboratrice en poste le weekend (dimanche 7 septembre 2025). La fiche affiche ses coordonnées professionnelles (email, téléphone), son adresse physique, ainsi que son affectation organisationnelle (Site : Youssoufia, Secteur : Production, Service : Traitement). Ce niveau de détail permet un contact direct et rapide en cas d'urgence, et assure une traçabilité claire des responsabilités pendant les périodes d'astreinte.

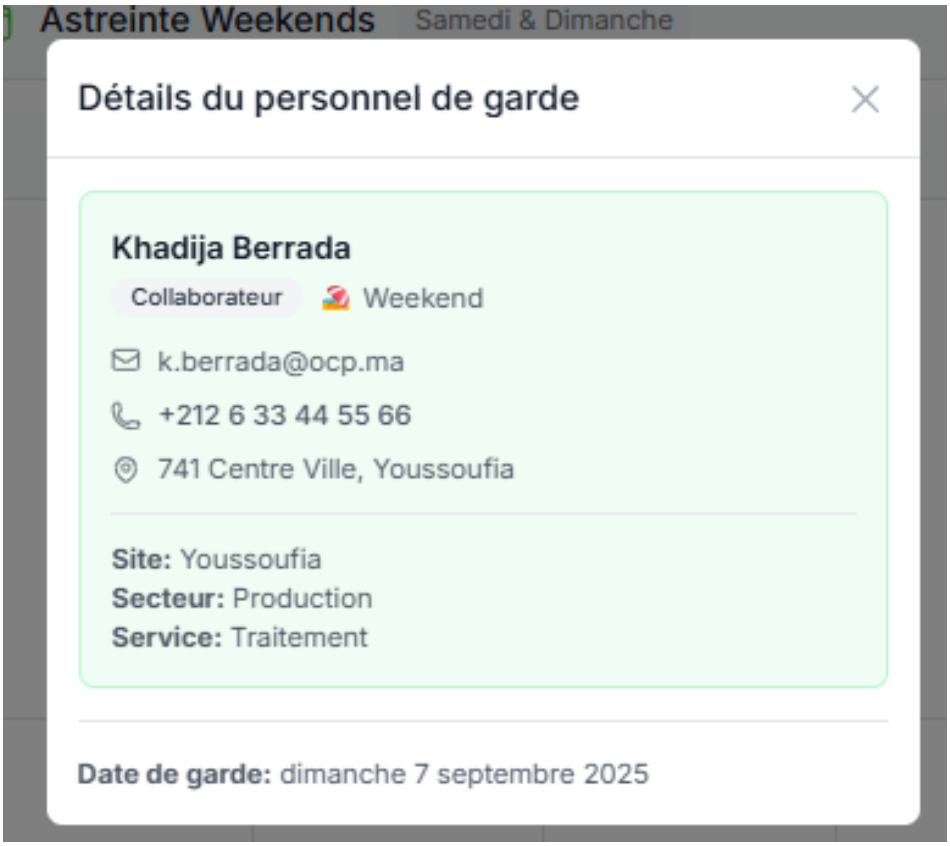

FIGURE 5.3 – Les information d'un collaborateur

Page de Connexion - Application de Gestion des Astreintes OCP

Cette interface d'authentification permet aux utilisateurs d'accéder à l'application en saisissant leurs identifiants (adresse email et mot de passe). La page propose une option "Se souvenir de moi" pour faciliter les connexions ultérieures, et le bouton "Se connecter" valide la saisie. Conçue de manière simple et professionnelle, elle sert de point d'entrée sécurisé pour accéder à toutes les fonctionnalités de gestion des astreintes weekends et jours fériés.

FIGURE 5.4 – Page de connexion

5.2.2 Gestion des utilisateurs

La gestion des utilisateurs inclut l'enregistrement, la modification des profils, ainsi que la définition des rôles (ingénieur, collaborateur, chef de service, chef de secteur, administrateur). Chaque rôle possède des permissions spécifiques, contrôlées par un système d'authentification et d'autorisation sécurisé.

Tableau de Bord Administrateur - Application de Gestion des Astreintes OCP

Cette interface constitue le tableau de bord principal personnalisé pour l'administrateur (ici Youssef, basé à Casablanca). Il offre une vision globale du système avec plusieurs sections clés : un résumé des déclarations de pannes (Nouvelles, En cours, Résolues, Total), un module de planning des astreintes weekends et jours fériés avec filtres avancés par site, secteur et service, et un accès rapide au formulaire de déclaration de pannes. Le calendrier affiche les astreintes par créneau (ex : Sam 6 - Dim 7 septembre) avec les noms des collaborateurs assignés, permettant une supervision centralisée de l'ensemble des activités d'astreinte sur le site de Casablanca.

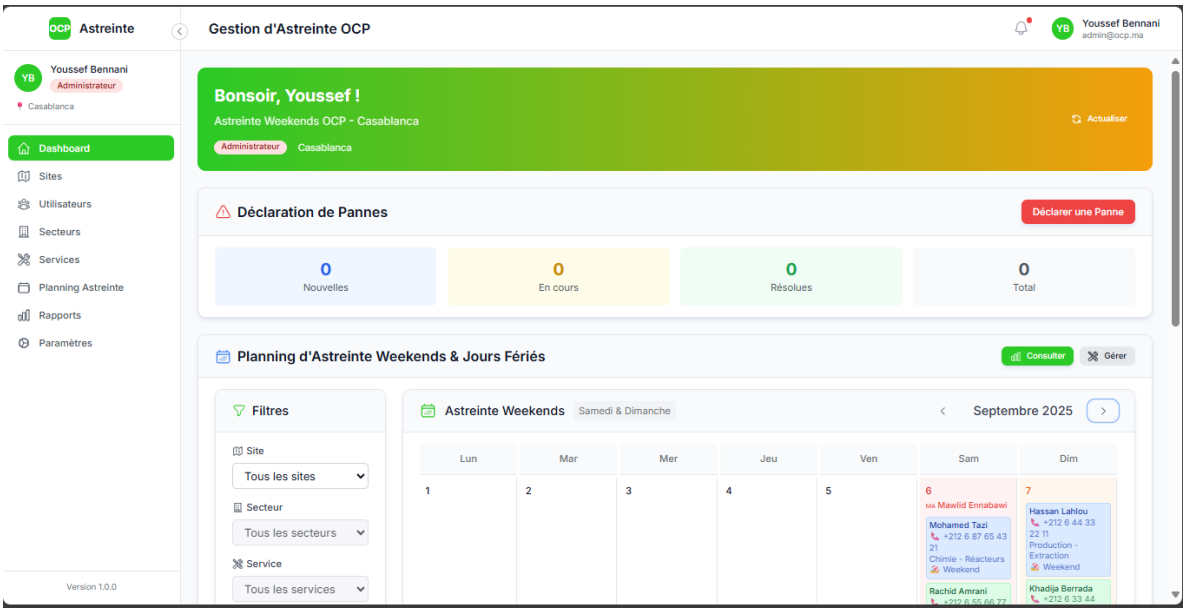

FIGURE 5.5 – Dashboard de l'administrateur

Page de Gestion des Utilisateurs - Application de Gestion des Astreintes OCP

Cette interface permet à l'administrateur de gérer tous les utilisateurs du système avec des fonctionnalités de filtrage avancé par rôle, site, secteur et service. Le tableau présente les informations des utilisateurs (nom, email, téléphone) avec leur rôle et organisation respective, et permet des actions de gestion. Un panneau latéral donne accès aux différents services de l'application (Planning Astreinte, Reports, Paramètres). La mention finale indique que les données utilisateurs sont synchronisées avec la base de données centrale de l'OCP.

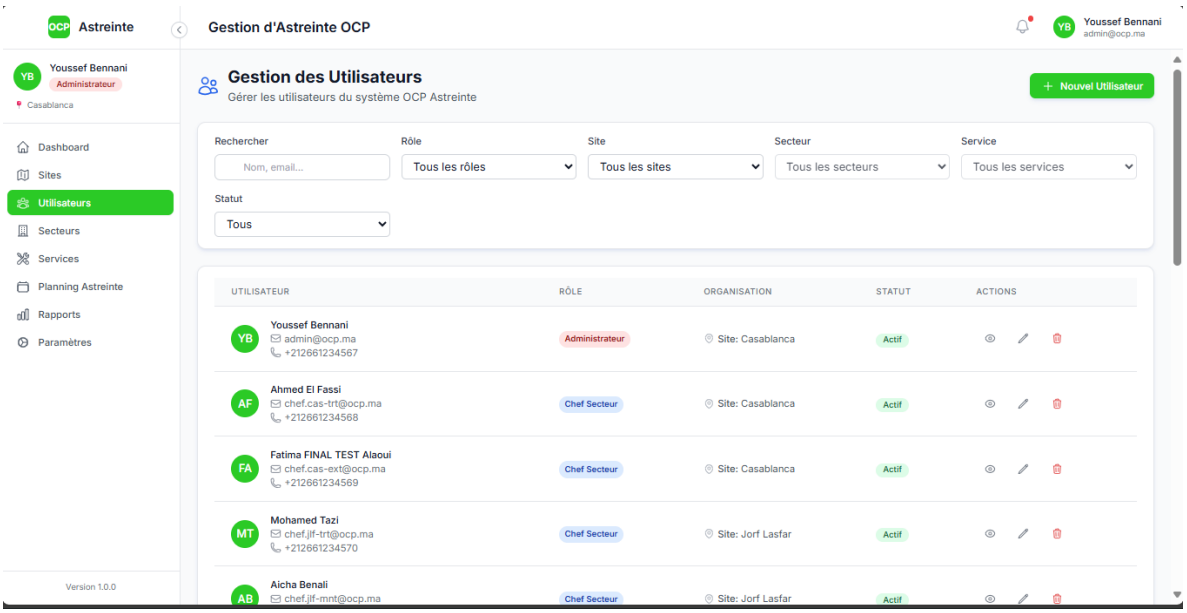

FIGURE 5.6 – Gestion des utilisateurs

Page de Gestion des Sites - Application de Gestion des Astreintes OC

Cette interface permet à l'administrateur de gérer l'ensemble des sites industriels de l'OCP. Elle liste les 9 sites avec leurs informations détaillées (localisation, nombre de secteurs et d'employés). La page offre des fonctionnalités de recherche, d'actualisation des données et d'ajout de nouveaux sites, servant ainsi de module de configuration central pour la structure organisationnelle de l'application.

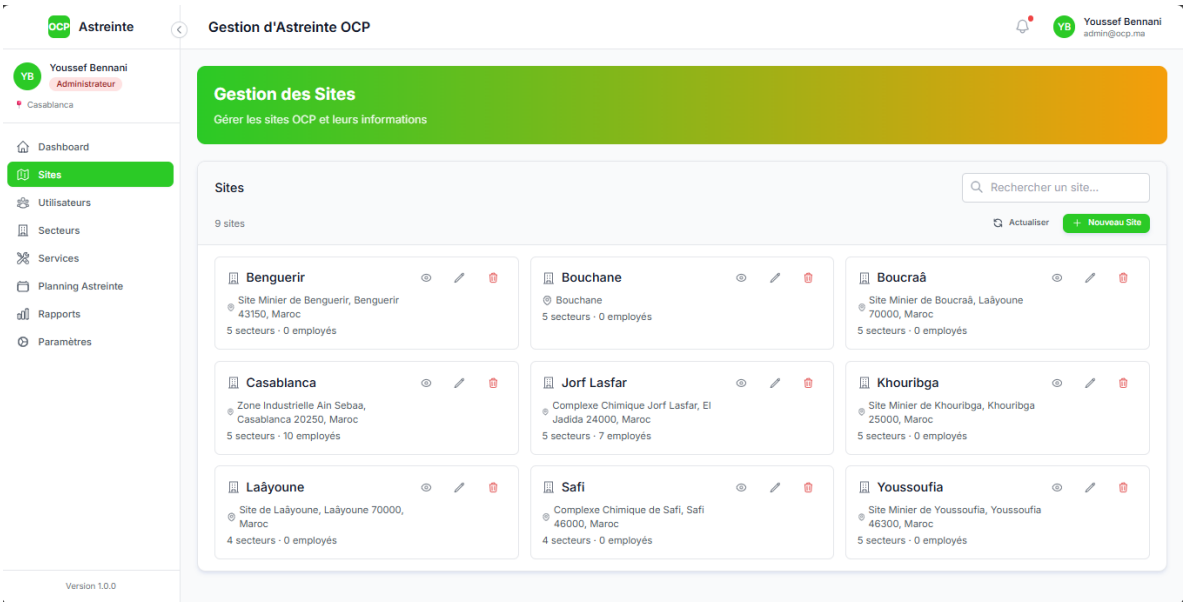

FIGURE 5.7 – Gestion des sites OCP

Page "Mon Service" - Application de Gestion des Astreintes OCP

Cette interface personnalisée permet au chef de service de gérer son unité opérationnelle (« Forage » dans cet exemple). La page présente les informations essentielles du service, incluant son code identifiant (CAS-EXT-FOR) et son responsable. Elle affiche également des indicateurs de performance clés tels que le nombre de collaborateurs (1), le nombre de gardes effectuées sur le mois (8) et le taux de couverture des astreintes (100%). La section « Mon Équipe » liste les membres avec leurs coordonnées complètes, comme Amina El Fassi. Cette vue d'ensemble permet au chef de service de superviser efficacement la disponibilité de son équipe et de garantir le respect des obligations d'astreinte.

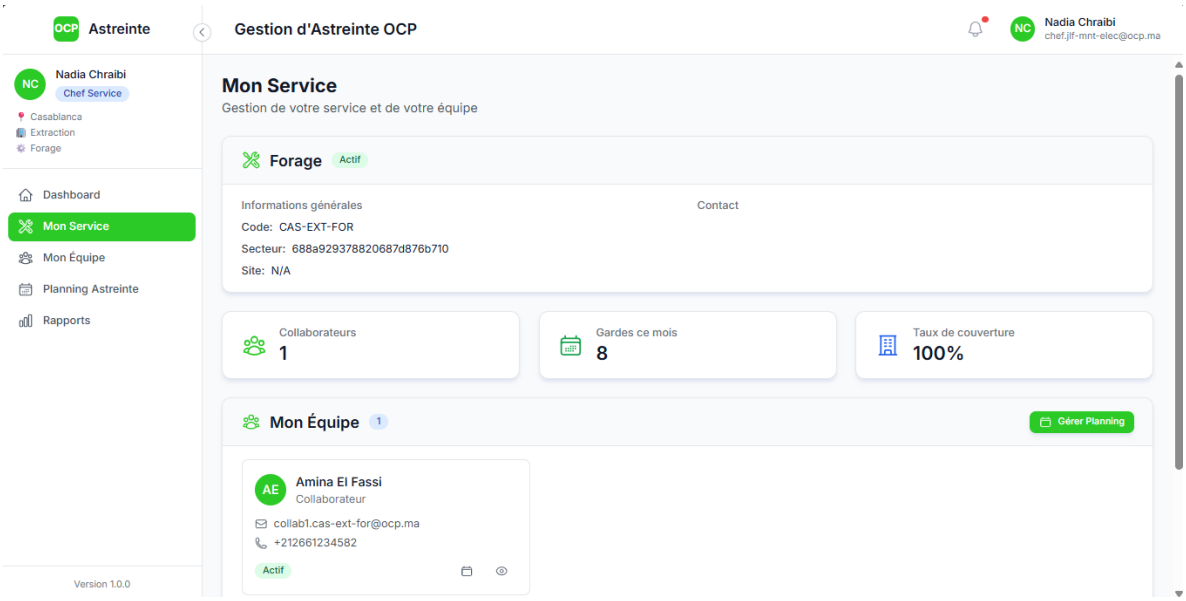

FIGURE 5.8 – Page "Mon Service"

Page "Planning des Astreintes" - Application de Gestion des Astreintes OCP

Cette interface spécialisée permet au Chef de Service de gérer les astreintes de son équipe. L'écran indique qu'aucune garde n'est actuellement assignée et présente un historique des pannes récentes avec leur niveau de priorité. La section « Assigner une Astreinte » permet d'affecter des gardes aux collaborateurs comme Hassan Zouani. Des statistiques en temps réel montrent l'état opérationnel du service (0 gardes actives, 0 pannes ouvertes, 2 personnels disponibles). Cette interface donne au Chef de Service tous les outils nécessaires pour planifier les astreintes, déclarer des pannes et superviser l'état de son équipe en un seul endroit.

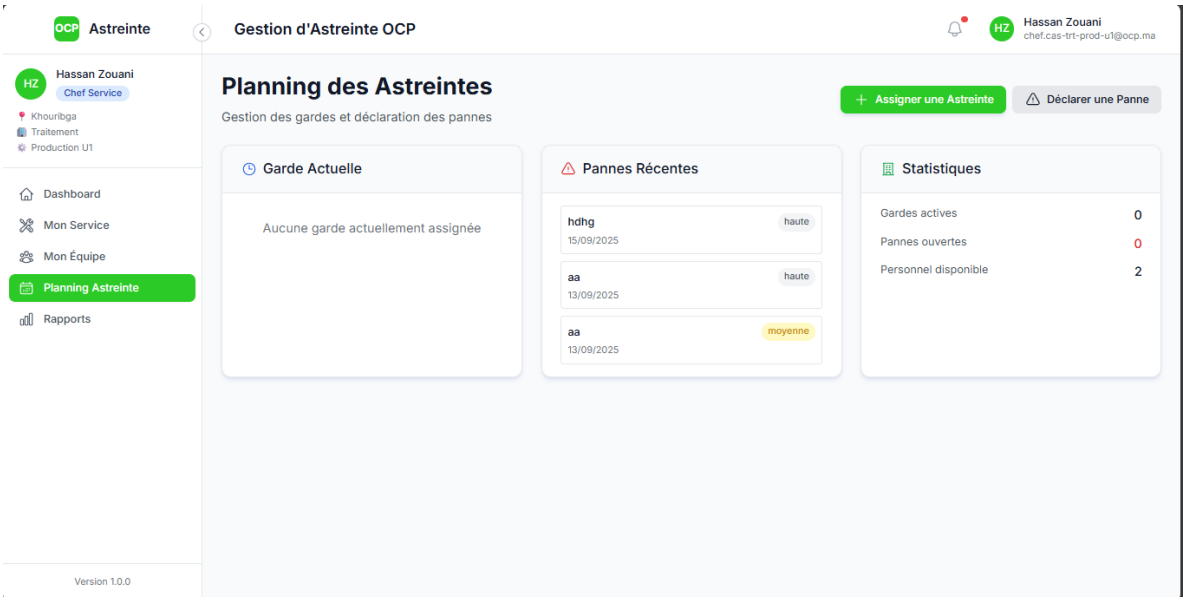

FIGURE 5.9 – Page "Planning des Astreintes"

Page de Gestion des Secteurs - Application de Gestion des Astreintes OCP

Cette interface permet à l'administrateur de gérer les secteurs d'activité pour chaque site OCP, avec un focus sur le site de Casablanca dans cet exemple. La page affiche un tableau de filtrage par code de site (BDI, BCH, BOS, etc.) et liste les secteurs avec leurs informations détaillées (nom, code, nombre de services et d'utilisateurs). Pour chaque secteur, des actions de gestion sont disponibles (Voir, Modifier, Supprimer), permettant une administration fine de la structure organisationnelle. La page montre également comment un secteur "Extraction" est répliqué across différents sites avec des codes spécifiques (ex : CAS-EXT pour Casablanca, BOU-EXT pour Boucraâ).

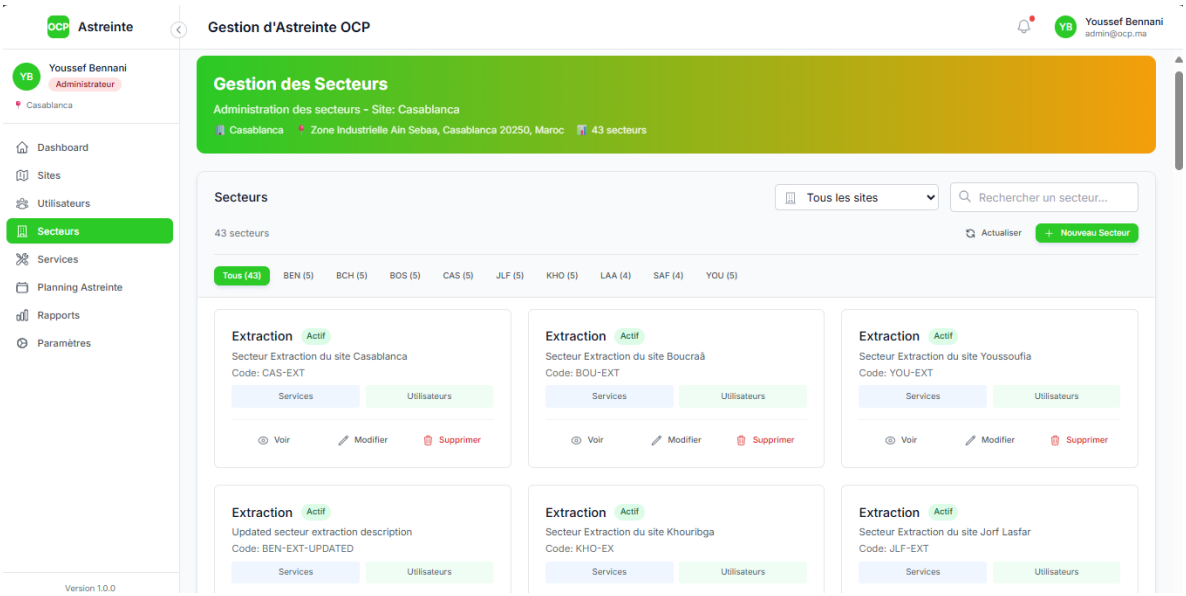

FIGURE 5.10 – Gestion des Secteurs d'OCP

Page de Gestion des Services - Application de Gestion des Astreintes OCP

Cette interface permet à l'administrateur de gérer les 104 services de l'ensemble des sites OCP. Chaque service est identifié par un code unique (ex : BDU-LOG-APPR) et rattaché à un site et un secteur spécifiques. La page affiche pour chaque service des informations cruciales comme le personnel assigné, le minimum requis d'effectifs et le taux de participation aux astreintes. Des actions de gestion (Voir, Modifier, Supprimer) sont disponibles pour chaque entrée, permettant une administration détaillée de la structure opérationnelle de l'OCP.

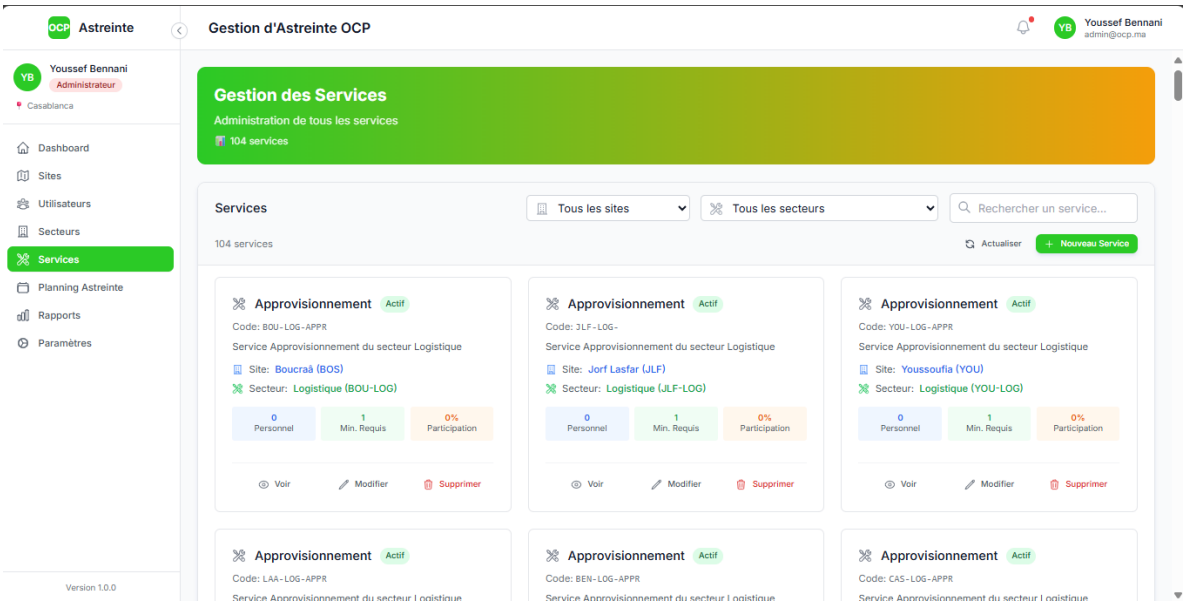

FIGURE 5.11 – Gestion des services

5.2.3 Système d'escalade

Le système d'escalade garantit que lorsqu'un agent d'astreinte est indisponible ou ne répond pas, une notification est automatiquement envoyée à un responsable ou à un autre agent désigné. Cette chaîne d'escalade permet d'assurer la continuité du service et une prise en charge rapide des incidents.

5.3 Tests et validation

Les fonctionnalités développées ont été testées à différents niveaux pour garantir la qualité et la robustesse de l'application :

- **Tests unitaires** : Vérification du bon fonctionnement individuel des composants backend (API Node.js) et frontend (composants React).
- **Tests d'intégration** : Validation des interactions entre les différents modules, notamment la communication entre le frontend et le backend via les API REST.
- **Tests fonctionnels** : Assurance de la conformité aux besoins métier définis dans le cahier des charges, incluant la gestion des astreintes, la planification et le système d'escalade.
- **Validation utilisateur** : Tests en conditions réelles avec un groupe pilote d'utilisateurs (collaborateurs, chefs de service, administrateurs) pour recueillir les retours et ajuster l'application.

Ces tests ont permis d'identifier et de corriger les anomalies, garantissant ainsi la fiabilité et la robustesse de l'application. La couverture des tests couvre les principaux cas d'usage, y compris la gestion des erreurs et les performances sous charge.

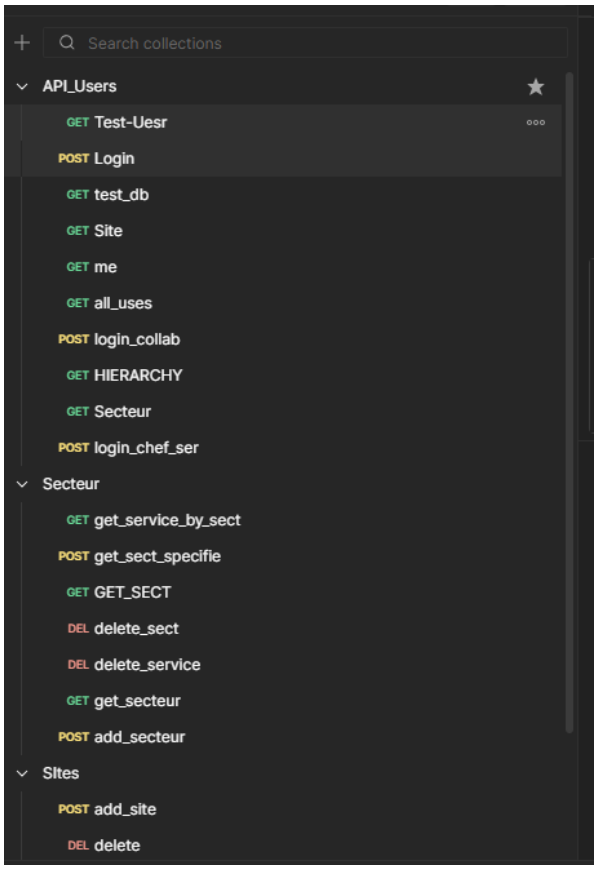

FIGURE 5.12 – Test des API avec Postman

La figure illustre les tests effectués sur les API backend à l'aide de Postman. Les endpoints testés incluent :

— **Gestion des utilisateurs** : Connexion des utilisateurs, récupération des profils, et administration des hiérarchies organisationnelles.
— **Gestion des secteurs et services** : Consultation, création et suppression des secteurs et services associés.
— **Gestion des sites** : Ajout et suppression des sites industriels dans le système.

Chapitre 6

Déploiement

6.1 Environnement de production

L'application est déployée dans un environnement de production sécurisé et scalable, garantissant une disponibilité optimale pour les utilisateurs finaux. Cet environnement comprend des serveurs dédiés ou cloud, configurés pour supporter la charge et assurer la continuité du service. La stack technique repose sur une architecture conteneurisée avec Docker, offrant flexibilité et isolation des composants.

6.2 Docker et conteneurisation

La conteneurisation avec Docker permet d'encapsuler l'ensemble de l'application (backend, frontend) dans des conteneurs isolés, facilitant ainsi la portabilité, la gestion des dépendances et la cohérence entre les environnements de développement, test et production. La base de données MongoDB est hébergée sur MongoDB Atlas (cloud), assurant scalabilité et haute disponibilité sans gestion d'infrastructure interne.

Chaque composant est défini via un fichier `Dockerfile` et orchestré avec `docker-compose` pour simplifier le lancement et la gestion multi-conteneurs.

6.2.1 Server Dockerfile

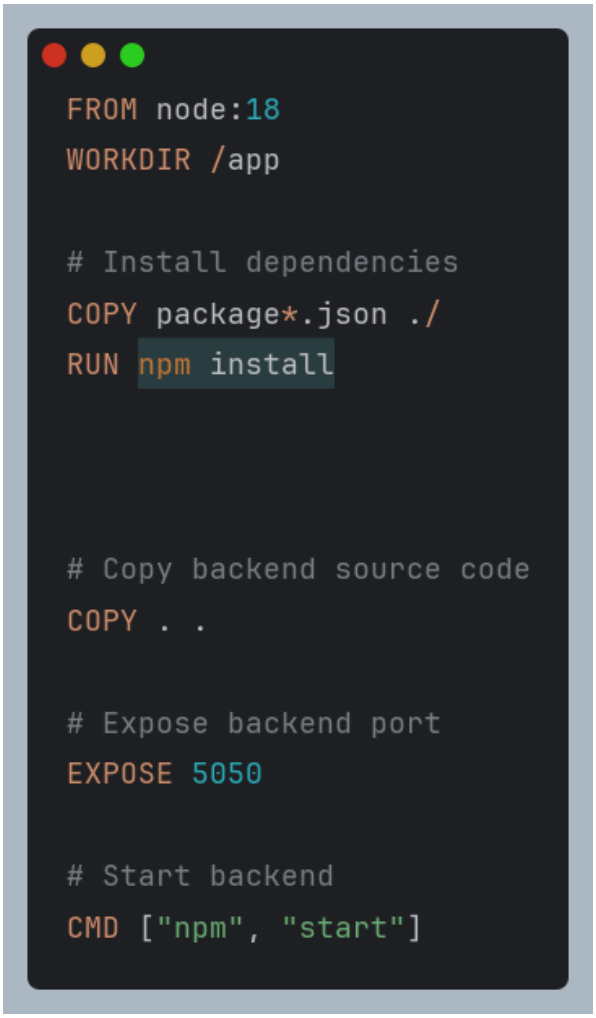

FIGURE 6.1 – ServerDockerfile - Configuration du backend Node.js

Le Dockerfile du backend utilise Node.js 18 comme image de base. Il copie les fichiers de dépendances (`package*.json`), installe les modules, copie le code source et expose le port 5050. La commande de démarrage est `npm start`.

6.2.2 Frontend Dockerfile

```dockerfile
# Step 1: Build React app
FROM node:18 AS build
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# Step 2: Serve with Nginx
FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

FIGURE 6.2 – FrontendDockerfile - Build React et service Nginx

Le Dockerfile du frontend utilise une build multi-étapes : il construit d'abord l'application React avec Node.js 18, puis utilise Nginx pour servir les fichiers static optimisés. Le port 80 est exposé pour le service web.

6.2.3 Docker Compose

```yaml
version: "3.9"
services:
  frontend:
    build: ./frontend
    ports:
      - "5173:80"   # React app
    environment:
      - REACT_APP_API_URL=http://backend:5050
    depends_on:
      - backend
  backend:
    build: ./server
    ports:
      - "5050:5050"
    environment:
      - MONGO_URI=mongodb+srv://adam:Adam37@adam.iiafxif.mongodb.net/gestion_astreinte?retryWrites=true&w=majority&appName=adam
```

FIGURE 6.3 – Docker Compose - Orchestration des services

Le fichier `docker-compose.yml` définit deux services :

— **frontend** : Construit à partir du dossier `./frontend`, exposé sur le port 5173 de l'hôte, et configuré pour communiquer avec le backend via la variable `REACT_APP_API_URL`.

— **backend** : Construit à partir du dossier `./server`, exposé sur le port 5050, et connecté à une base MongoDB Atlas via une chaîne de connexion sécurisée.

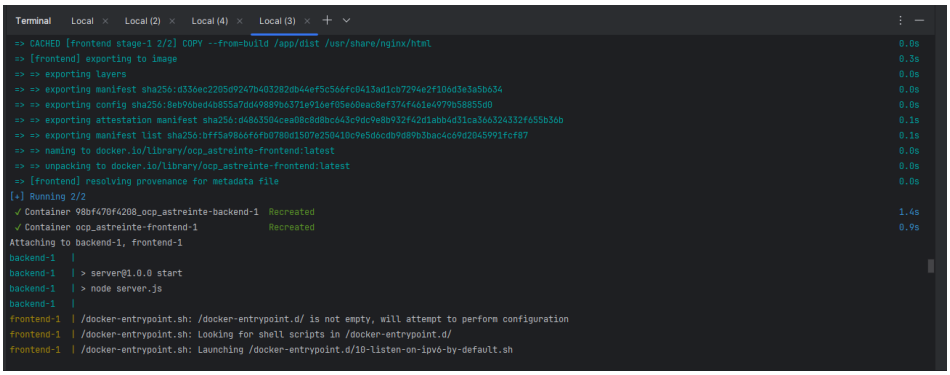

FIGURE 6.4 – Exécution de Docker Compose en environnement de production

La figure montre l'exécution réussie de la commande `docker-compose up` avec création des réseaux et démarrage des conteneurs, confirmant le bon fonctionnement de l'orchestration Docker en environnement de production.

6.3 Mise en place et configuration

Le déploiement se fait en plusieurs étapes :

— Préparation de l'infrastructure serveur ou cloud (ex : AWS, Azure, ou serveurs dédiés).
— Installation de Docker et Docker Compose sur le serveur.
— Clonage du repository et configuration des variables d'environnement (notamment la connexion MongoDB).
— Construction des images et lancement des conteneurs via `docker-compose up -build`.
— Configuration des accès sécurisés (HTTPS, firewall, authentification).
— Mise en place d'outils de monitoring (ex : Prometheus, Grafana) pour superviser la santé des services.

Chapitre 7

Conclusion et perspectives

7.1 Bilan du projet

Le développement de l'application de gestion des astreintes s'inscrit dans un contexte organisationnel où la réactivité et la bonne coordination des équipes sont des enjeux majeurs. Dès le début, le projet avait pour objectif de centraliser et automatiser le suivi des périodes d'astreinte, afin d'assurer une meilleure visibilité pour l'ensemble des acteurs impliqués et de réduire les risques liés aux retards ou à la mauvaise transmission d'informations.

La phase d'analyse a permis de mettre en évidence les besoins spécifiques de l'organisation, notamment :

— disposer d'un outil unique et fiable pour la gestion des plannings d'astreinte ;
— offrir un accès différencié selon le rôle de l'utilisateur (responsable, secrétaire, intervenant) afin de garantir la confidentialité et la sécurité des données ;
— mettre en place un système de suivi et d'escalade permettant de gérer efficacement les incidents ;
— assurer une portabilité et une facilité de déploiement sur différents environnements techniques.

La réalisation de l'application a mobilisé plusieurs technologies et bonnes pratiques de développement. L'utilisation d'une architecture **Node.js** / **Express.js** pour le backend et **React.js** pour le frontend a permis de garantir à la fois performance et modularité. La base de données **MongoDB** a été choisie pour sa flexibilité et sa capacité à gérer des données dynamiques. Enfin, l'adoption de **Docker** pour le déploiement a facilité la mise en production et l'isolation des environnements, ce qui représente un atout majeur pour la maintenance et l'évolution future du projet.

Les principales fonctionnalités livrées sont :

— **Gestion des astreintes** : création, modification et suppression des périodes d'astreinte avec une interface claire et intuitive ;
— **Gestion des utilisateurs** : attribution de rôles et gestion sécurisée des accès ;
— **Système d'escalade** : remontée automatique des alertes en cas de non-traitement d'un incident dans les délais ;
— **Tableau de bord et statistiques** : vue globale sur les astreintes en cours, à venir et passées ;
— **Déploiement via Docker** : conteneurisation et portabilité de l'application.

Sur le plan personnel, ce projet m'a permis de renforcer mes compétences techniques en développement web full-stack, gestion de bases de données, conception d'API et intégration de solutions de déploiement. J'ai également acquis une expérience significative en gestion de projet, notamment en planification des tâches, respect des délais et adaptation aux contraintes réelles du terrain.

7.2 Perspectives d'évolution

Bien que l'application réponde aux besoins exprimés initialement, plusieurs améliorations peuvent être envisagées afin de la rendre encore plus performante et adaptée aux futurs usages.

Parmi les évolutions possibles :

— **Intégration de notifications en temps réel** : envoi automatique d'alertes par email, SMS ou via une application mobile dédiée ;
— **Développement d'une application mobile** native ou hybride, permettant aux utilisateurs d'accéder au planning et de valider des interventions directement sur leur smartphone ;
— **Mise en place de statistiques avancées** : analyse des tendances, temps moyen de traitement des incidents, répartition des astreintes par service ;
— **Automatisation avancée** : intégration avec un système de gestion d'incidents pour créer automatiquement des interventions en cas de panne détectée ;
— **Renforcement de la sécurité** : ajout d'une authentification à deux facteurs (2FA) et chiffrement renforcé des données ;
— **Mode hors-ligne** : permettre à l'application de rester fonctionnelle même en l'absence de connexion internet, avec synchronisation ultérieure.

7.3 Conclusion générale

En conclusion, le développement de cette application représente une étape importante dans la modernisation et l'optimisation du processus de gestion des astreintes. Elle offre une solution centralisée, fiable et évolutive, tout en ouvrant la voie à de nouvelles perspectives d'amélioration. Au-delà de l'aspect purement technique, ce projet a également été une expérience humaine enrichissante, marquée par la collaboration, la résolution de problèmes et la recherche de solutions innovantes. Il constitue ainsi une base solide sur laquelle il sera possible de bâtir un outil encore plus complet, capable de répondre aux exigences croissantes de réactivité et d'efficacité dans le domaine de la gestion des astreintes.

Bibliographie

[1] *React Documentation*, 2023.
Disponible en ligne : `https://react.dev/`.

[2] OpenJS Foundation, *Node.js Documentation*, 2023.
Disponible en ligne : `https://nodejs.org/`.

[3] OpenJS Foundation, *Express.js Documentation*, 2023.
Disponible en ligne : `https://expressjs.com/`.

[4] MongoDB Inc., *MongoDB Documentation*, 2023.
Disponible en ligne : `https://www.mongodb.com/`.

[5] *TypeScript Documentation*, 2023.
Disponible en ligne : `https://www.typescriptlang.org/`.

[6] Tailwind Labs, *Tailwind CSS Documentation*, 2023.
Disponible en ligne : `https://tailwindcss.com/`.

[7] IETF, *JWT Specification*, 2023.
Disponible en ligne : `https://jwt.io/`.

[8] OCP Group, *Site officiel*, 2023.
Disponible en ligne : `https://www.ocpgroup.ma/`.

[9] MongoDB, *MERN Stack*, 2023.
Disponible en ligne : `https://www.mongodb.com/mern-stack`.

[10] *REST API Best Practices*, 2023.
Disponible en ligne : `https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design`.