



MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
DE LA RECHERCHE SCIENTIFIQUE ET DE L'INNOVATION
UNIVERSITÉ SULTAN MOULAY SLIMANE
ÉCOLE NATIONALE DES SCIENCES APPLIQUÉES
DE KHOURIBGA



Rapport de Projet de Fin d'Année (PFA)

Filière : Informatique et Ingénierie des Données



Présenté par :
Lajghal Mohamed Rida

Système intelligent de gestion des interventions avec intégration IoT et maintenance prédictive

Sous la supervision :
M. El ARCHI Nabil
PR. LYAQINI Soufiane

Membres du jury :

KAROUM Bouchra ENSA Président
RAFIK Salma ENSA Examinateur

Année Académique : 2025 - 2026

Sommaire

Dédicace	iii
Remerciements	iv
Résumé	v
Abstract	vi
Table des matières	xi
Table des figures	xiii
Liste des tableaux	xiv
Introduction	1
1 Présentation de l'organisme d'accueil	3
2 Préliminaires théoriques et techniques	14
3 Conception et Méthodologie	22
4 Technologies et Interfaces	29
5 Expérimentation et Résultats Machine Learning	54
Conclusion	63
Bibliographie	64

Dédicace

“

*À mes parents, pour leur amour inconditionnel et leur soutien
indéfectible tout au long de ce parcours.*

*À mes frères et sœurs, pour leur présence, leurs encouragements et
leurs sourires qui m'ont porté dans les moments de doute.*

*À mes amis, compagnons de voyage, pour les rires, les conversations
profondes et les souvenirs partagés qui ont rendu ce chemin plus
agréable.*

*À mes enseignants, pour leur patience, leur savoir et leur
dévouement qui ont nourri ma curiosité.*

*À tous ceux qui, d'une manière ou d'une autre, have cru en moi et
m'ont aidé à avancer.*

Merci du fond du cœur.

”

- Mohamed Rida

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui m'ont accompagné et soutenu dans cette aventure académique et professionnelle. Ce mémoire est le fruit d'une collaboration précieuse entre plusieurs acteurs clés que je souhaite chaleureusement remercier.

Mes premiers remerciements vont à l'équipe d'OCP pour m'avoir offert l'opportunité unique de contribuer à un projet d'envergure dans le secteur industriel. L'environnement stimulant et les défis techniques rencontrés ont considérablement enrichi mon expérience professionnelle.

Je souhaite particulièrement souligner l'engagement de **M. El ARCHI Nabil**, mon encadrant industriel, dont l'expertise technique et la pédagogie ont été déterminantes dans la réussite opérationnelle de ce projet. Son accompagnement au quotidien et sa capacité à transmettre des bonnes pratiques métier resteront pour moi une source d'inspiration.

Sur le plan académique, je remercie chaleureusement **PR. LYAQINI Soufiane**, mon encadrant à l'ENSA Khouribga, pour sa rigueur scientifique et sa disponibilité constante. Ses retours constructifs et son exigence intellectuelle ont grandement contribué à l'amélioration de ce mémoire.

Enfin, je tiens à exprimer ma gratitude envers les membres du jury pour leur engagement à évaluer ce projet avec rigueur et pour leurs remarques constructives qui ont enrichi mon travail de manière significative.

Ce projet représente bien plus qu'une réalisation technique : c'est une expérience humaine marquante qui a renforcé ma passion pour l'ingénierie des données et mon désir de contribuer à la transformation numérique du secteur industriel.

Résumé

Ce projet présente la conception et la mise en œuvre d'un système de maintenance intelligente combinant gestion digitale des interventions et prédition des pannes à partir de données industrielles. La plateforme distingue trois profils (administrateur, utilisateur, technicien) : l'utilisateur crée une demande d'intervention, l'administrateur l'assigne à un technicien, qui la traite et la clôture. Parallèlement, nous avons intégré un flux IoT avec Apache Kafka, alimenté par un jeu de données issu du programme Siemens Smart Manufacturing Lab (suggéré par OCP en raison de sa similitude avec leur contexte, leurs données internes n'étant pas accessibles). Les données capteurs (vibrations, température, pression, etc.) sont transmises et plusieurs modèles de machine learning ont été entraînés et comparés, afin de sélectionner celui offrant les meilleures performances. Le modèle retenu est ensuite déployé dans un backend développé avec Spring Boot, exploitant un flux Apache Kafka, puis stocke les résultats dans une base MySQL après analyse. Les résultats sont affichés sur un tableau de bord interactif permettant de suivre en temps réel l'état des équipements, les anomalies et les interventions. Ce système offre à OCP une solution de digitalisation et de fiabilisation des processus de maintenance, tout en renforçant nos compétences en architecture distribuée, traitement de flux, et apprentissage automatique appliquée à la maintenance prédictive.

Mots-clés : Maintenance prédictive, Kafka, Spring Boot, React, FastAPI, MySQL, Machine Learning, IoT, Digitalisation.

Abstract

This project presents the design and implementation of an intelligent maintenance system that combines digital intervention management with predictive failure analysis using industrial data. The platform distinguishes three user profiles (administrator, user, technician) : users create intervention requests, administrators assign them to technicians, who then process and close them. In parallel, we integrated an IoT data stream using Apache Kafka, powered by a dataset from the Siemens Smart Manufacturing Lab program (suggested by OCP due to its similarity to their context, as their internal data was not accessible). Sensor data (vibrations, temperature, pressure, etc.) are transmitted and multiple machine learning models were trained and compared to select the one offering the best performance. The selected model is then deployed in a backend developed with Spring Boot, leveraging an Apache Kafka stream, and stores results in a MySQL database after analysis. Results are displayed on an interactive dashboard enabling real-time monitoring of equipment status, anomalies, and interventions. This system provides OCP with a digitalization and reliability solution for maintenance processes, while strengthening our skills in distributed architecture, stream processing, and machine learning applied to predictive maintenance.

Keywords : Predictive maintenance, Kafka, Spring Boot, React, FastAPI, MySQL, Machine Learning, IoT, Digitalization.

ملخص

يقدم هذا المشروع تصميم وتنفيذ نظام صيانة ذكي يجمع بين الإدارة الرقمية للتتدخلات والتتبؤ بالأخطال من خلال البيانات الصناعية. تتميز المنصة بين ثلاثة ملفات شخصية (مدير، مستخدم، فني): يقوم المستخدم بإنشاء طلب تدخل، ويقوم المدير بتعيينه لفني، الذي يقوم بمعالجته وإغلاقه. بالتوازي مع ذلك، قمنا بدمج تدفق إنترنت الأشياء باستخدام Apache Kafka، مدعوم بمجموعة بيانات من برنامج Siemens Smart Manufacturing Lab (اقترحه شركة OCP بسبب شابهاها مع سياقهم، حيث أن بياناتهم الداخلية غير متاحة). يتم نقل بيانات أجهزة الاستشعار (الاهتزازات، درجة الحرارة، الضغط، الخ) وتم تدريب ومقارنة عدة نماذج تعلم آلية لاختيار النموذج الذي يقدم أفضل أداء. يتم بعد ذلك نشر النموذج المحدد في واجهة خلفية مطورة باستخدام Spring Boot، مع الاستفادة من تدفق MySQL بعد التحليل. يتم عرض النتائج على لوحة معلومات تفاعلية تتبع مرافقحة حالة المعدات والشدة والتتدخلات في الوقت الفعلي. يوفر هذا النظام لشركة OCP حلًّا لرقمنة وتحسين موثوقية عمليات الصيانة، مع تعزيز مهاراتنا في البنية الموزعة ومعالجة التدفقات والتعلم الآلي المطبق على الصيانة التنبؤية.

الكلمات المفتاحية: الصيانة التنبؤية، Kafka، Spring Boot، React، FastAPI، MySQL.

Table des matières

Dédicace	iii
Remerciements	iv
Résumé	v
Abstract	vi
Table des matières	x
Table des figures	xii
Liste des tableaux	xiii
Introduction Générale	1
1 Présentation de l'organisme d'accueil	3
1.1 Introduction	3
1.2 Présentation du Groupe OCP	3
1.2.1 Généralités	3
1.2.2 Historique du Groupe	4
1.2.3 Activités et Chaîne de Valeur	5
1.2.4 Statut Juridique et Organisation	6
1.2.5 Filiales Principales	6
1.3 Présentation du Service Informatique de Khouribga	7
1.3.1 Positionnement Organisationnel	7
1.3.2 Missions Principales	8
1.3.3 Structure Organisationnelle du Service	8
1.3.4 Infrastructure Réseau	10
1.3.5 Rôle Stratégique	10
1.4 Cadrage du projet	10
1.4.1 Contexte du projet	10

1.4.2	Problématique	11
1.4.3	Objectifs	11
1.4.4	Vision et Motivation	12
1.4.5	Conclusion	12
2	Préliminaires théoriques et techniques	14
2.1	Traitement de flux de données et Apache Kafka	14
2.1.1	Évolution vers KRaft	15
2.2	Machine Learning pour la maintenance prédictive	16
2.2.1	Le processus de ML	16
2.2.2	Algorithme LightGBM	16
2.2.3	Prédiction et interprétation des résultats	18
2.3	Opérationnalisation des modèles (MLOps) et intégration ONNX	18
2.4	Architecture logicielle : Application Monolithique Modulaire	19
2.5	Simulation des données IIoT via une API de streaming	20
2.6	Conclusion	21
3	Conception et Méthodologie	22
3.1	Introduction	22
3.2	Analyse des Fonctionnalités et Cas d'Utilisation	22
3.2.1	Fonctionnalités Principales	22
3.2.2	Diagramme de cas d'utilisation	23
3.3	Architecture du Système	24
3.3.1	Vue d'ensemble	24
3.3.2	Description des Modules	24
3.4	Workflows de Traitement	25
3.4.1	Workflow de Prédiction Temps Réel	25
3.4.2	Workflow de Gestion d'Intervention	27
3.5	Défis et Solutions	27
3.5.1	Défis Techniques Rencontrés	27
3.5.2	Solutions Implementées	28
3.6	Conclusion	28
4	Technologies et Interfaces	29
4.1	Introduction	29
4.2	Spring Boot : Backend Robustesse et Productivité	29
4.3	React + Vite : Interface Moderne et Performante	30
4.4	Apache Kafka : Colonne Vertébrale Temps Réel	31

4.5	MySQL : Persistance Fiable et Performante	31
4.6	ONNX Runtime : Inférence ML Performante	32
4.7	Recharts : Visualisation Data Interactive	33
4.8	Environnement de Développement	33
4.9	Interfaces Utilisateur	34
4.9.1	Interface d'Authentification - Commune	34
4.9.2	Interface de Récupération de Mot de Passe	35
4.9.3	Processus de Vérification à Deux Facteurs (2FA)	36
4.9.4	Interface de Profil Utilisateur	38
4.9.5	Interface des Interventions Utilisateur	39
4.9.6	Interface Technicien - Gestion des Interventions Assignées	42
4.9.7	Interface Administrateur - Tableau de Bord	44
4.9.8	Interface de Gestion des Utilisateurs (Admin)	45
4.9.9	Interface de Gestion des Interventions (Admin)	47
4.9.10	Tableau de Bord de Monitoring Temps Réel	50
4.10	Conclusion	53
5	Expérimentation et Résultats Machine Learning	54
5.1	Introduction	54
5.2	Dataset et Source de Données	54
5.2.1	Origine et Contexte Industriel	54
5.2.2	Structure et Features du Dataset	55
5.3	Prétraitement et Préparation des Données	55
5.3.1	Nettoyage et Validation	55
5.3.2	Balancing des Classes avec SMOTE	56
5.4	Évaluation Comparative des Modèles	56
5.4.1	Méthodologie d'Évaluation	56
5.4.2	Résultats Comparatifs Détaillés	57
5.4.3	Visualisation des Performances	58
5.5	Sélection du Modèle Optimal	59
5.5.1	LightGBM : Meilleur Compromis	59
5.5.2	Optimisations Spécifiques LightGBM	60
5.6	Conversion et Déploiement ONNX	61
5.6.1	Processus de Conversion	61
5.7	Conclusion	62
Conclusion Générale		63

Table des figures

1.1	Logo du Groupe OCP	3
1.2	Implantation géographique du Groupe OCP	6
1.3	Structure organisationnelle du Groupe OCP	7
1.4	Organisation du Digital Office de Khouribga	9
2.1	Architecture Kafka	15
2.2	Comparaison des stratégies de croissance d'arbres : XGBoost (niveau par niveau) vs LightGBM (feuille par feuille)	17
2.3	Principe de l'apprentissage par gradient boosting : correction successive des résidus	17
2.4	Diagramme de l'architecture monolithique modulaire montrant les modules internes et les flux de données.	20
3.1	Diagramme de cas d'utilisation de la plateforme	23
3.2	Architecture globale du système de maintenance prédictive	24
3.3	Workflow de prédiction temps réel avec polling	26
3.4	Workflow de gestion d'intervention	27
4.1	Logo Spring Boot	29
4.2	Logos React et Vite	30
4.3	Logo Apache Kafka	31
4.4	Logo MySQL	31
4.5	Logo ONNX Runtime	32
4.6	Logo Recharts	33
4.7	Interface de connexion commune	34
4.8	Saisie de l'adresse e-mail	35
4.9	Saisie de l'OTP et définition du nouveau mot de passe	35
4.10	OTP reçu par e-mail	36
4.11	Formulaire de vérification du code OTP (2FA) après la connexion	37
4.12	Interface de gestion du profil utilisateur	38
4.13	Interface principale de creation des interventions	39
4.14	Formulaire modal de création d'une nouvelle intervention	40

4.15	Affichage des détails complets d'une intervention	41
4.16	Tableau de bord et liste des interventions du technicien	42
4.17	Affichage détaillé des informations d'une intervention	43
4.18	Tableau de bord administratif de supervision du système	44
4.19	Interface principale de gestion des utilisateurs et techniciens	45
4.20	Formulaire de création d'un nouvel utilisateur	46
4.21	Formulaire de modification d'un utilisateur existant	46
4.22	Exemple d'email envoyé après la création d'un compte	47
4.23	Interface principale de gestion de toutes les interventions	47
4.24	Formulaire d'assignation d'un technicien à une intervention	48
4.25	Formulaire de réassiguation ou de suppression d'assignation	49
4.26	Vue d'ensemble du tableau de bord de maintenance prédictive	50
4.27	Vue détaillée de la santé des machines individuelles	51
4.28	Métriques temps réel des capteurs	51
4.29	Analyse thermique multi-points et Métriques de performance opérationnelle	52
4.30	Historique des défaillances prédictes	53
5.1	Distribution des classes avant et après application de SMOTE	56
5.2	Comparaison visuelle des métriques de performance	58
5.3	Matrice de confusion - LightGBM (Tuned)	59

Liste des tableaux

1.1	Présentation générale du Groupe OCP	4
1.2	Chronologie historique du Groupe OCP	5
5.1	Catégories de features du dataset Siemens	55
5.2	Tableau comparatif complet des performances des modèles	57

Introduction Générale

Dans le contexte de la quatrième révolution industrielle, la transformation numérique des opérations de maintenance constitue un enjeu stratégique majeur pour les acteurs industriels. Face à la criticité des équipements et au coût exorbitant des temps d'arrêt non planifiés, le passage d'une maintenance corrective ou préventive à une approche prédictive, basée sur les données, s'impose comme une nécessité pour optimiser la disponibilité, la productivité et la compétitivité. Ce projet de stage s'inscrit dans cette dynamique en proposant la conception et le développement d'une plateforme intelligente de maintenance prédictive, inspirée par des initiatives de pointe telles que celle du Siemens Smart Manufacturing Lab.

Ce projet avait pour ambition de créer un système intégré qui ne se contente pas de collecter des données, mais qui les transforme en véritables leviers d'action. La problématique centrale consistait à agréger en temps réel des flux de données hétérogènes provenant de capteurs IoT, de les analyser grâce à l'intelligence artificielle pour anticiper les défaillances, et de fournir une visibilité complète et actionnable sur la santé du parc machine via un tableau de bord interactif, le tout au sein d'une architecture moderne, scalable et robuste.

Pour relever ce défi, la solution s'appuie sur une stack technologique de pointe articulée autour de Spring Boot pour le backend métier, React.js pour l'interface utilisateur dynamique, Apache Kafka pour l'ingestion et le traitement des flux de données temps réel, et MySQL pour le stockage persistant. La dimension intelligence artificielle représente le cœur innovant du projet : après l'exploration, l'entraînement et l'évaluation comparative de plusieurs modèles de Machine Learning avancés (notamment XGBoost, LightGBM, et HistGradientBoosting) sur un jeu de données public de maintenance prédictive, le modèle le plus performant a été sélectionné et converti au format ONNX pour une inférence optimisée en production. Cette intégration permet d'enrichir chaque enregistrement de données sensorielle avec une probabilité de défaillance avant même son stockage en base de données, créant ainsi une mémoire de parcours enrichie pour chaque asset.

Sur le plan fonctionnel, la plateforme offre un système de gestion des rôles (Administrateur, Technicien, Utilisateur) permettant la création, l'assignation et le suivi des interventions de maintenance dans un workflow classique. La phase actuelle du projet a permis la construction et la visualisation des données prédictives, posant les bases solides d'une intégration fonctionnelle future où une prédiction de panne pourrait déclencher automatiquement la création et l'assignation d'un ticket d'intervention.

Les enjeux techniques de ce projet étaient considérables, nécessitant la maîtrise de technologies logicielles modernes pour le développement full-stack, la conception d'une architecture événementielle et scalable avec Kafka, l'opérationnalisation (MLOps) d'un modèle de ML dans un pipeline de production, et le respect des bonnes pratiques de développement et de sécurité. Les défis métier n'étaient pas moindres, requérant une compréhension fine des concepts de la maintenance prédictive et des impératifs opérationnels de l'industrie pour concevoir une solution à la fois puissante et ergonomique.

Ce rapport présentera dans un premier temps le contexte du stage au sein du Groupe OCP et la problématique métier abordée. Un état de l'art détaillera ensuite les concepts technologiques clés (IoT, architectures de streaming, Machine Learning) et le paysage des solutions existantes. La troisième partie sera consacrée à la conception détaillée de l'architecture et aux choix techniques effectués. La quatrième partie traitera des technologies utilisées et des interfaces développées. La cinquième partie présentera l'expérimentation et les résultats du modèle de Machine Learning. Enfin, une analyse des résultats obtenus et des perspectives d'amélioration, notamment l'automatisation des interventions et le déploiement de modèles en edge computing, viendra clore ce rapport.

Chapitre 1

Présentation de l'organisme d'accueil

1.1 Introduction

Dans ce chapitre introductif, nous présenterons le contexte organisationnel et professionnel dans lequel s'est déroulé notre stage. Nous commencerons par une présentation détaillée du Groupe OCP, en mettant en lumière son identité, ses domaines d'intervention et ses principales activités. Par la suite, nous nous concentrerons spécifiquement sur le service informatique de Khouribga, lieu exact de notre stage, en décrivant sa structure organisationnelle, ses missions et son rôle stratégique au sein du groupe.

1.2 Présentation du Groupe OCP

1.2.1 Généralités



FIGURE 1.1 – Logo du Groupe OCP

Le Groupe Office Chérifien des Phosphates (OCP) est un opérateur industriel marocain de renommée internationale, spécialisé dans l'extraction, la valorisation et la commercialisation des phosphates et de leurs dérivés. Crée par le dahir du 7 août 1920, avec le démarrage effectif des exploitations en février 1921 dans la région d'Oued-Zem, l'OCP occupe aujourd'hui la position de premier exportateur mondial de phosphate sous toutes ses formes.

Le sous-sol marocain détient les trois quarts des réserves mondiales de phosphates, soit plus de 70% des gisements planétaires. Le Groupe OCP extrait annuellement plus de 30 millions de tonnes de minerais de phosphate, contribuant substantiellement à hauteur d'environ 25% dans la valeur des exportations nationales.

TABLE 1.1 – Présentation générale du Groupe OCP

Critère	Valeur
Création	7 août 1920
Forme juridique	Société Anonyme
Siège social	Casablanca, route d'El Jadida
Actionnaires	État marocain (95%), BCP (5%)
Effectifs	Plus de 21 000 collaborateurs
Chiffre d'affaires	47,74 milliards MAD (2022)
Production annuelle	Plus de 30 millions de tonnes

1.2.2 Historique du Groupe

L'OCP occupe une place particulière dans l'histoire industrielle du Maroc. Ses performances trouvent leurs racines dans sa création en 1920.

À ses débuts, il s'agissait d'une simple activité d'extraction et de traitement de la roche. Progressivement, l'OCP s'est imposé sur tous les maillons de la chaîne de valeur, allant de la production d'engrais à celle d'acide phosphorique, en passant par divers produits dérivés.

Depuis sa création, le groupe puise les ressources de sa croissance continue et de son leadership dans une stratégie industrielle fondée sur :

- une montée en puissance régulière de son outil de production,
- une politique ambitieuse de partenariats durables,
- une gestion financière efficace.

Aujourd'hui, l'OCP est dirigé par plusieurs directions, toutes placées sous l'autorité d'une Direction Générale, dont le siège social se trouve à Casablanca.

TABLE 1.2 – Chronologie historique du Groupe OCP

Année	Événement marquant
1905	Découverte des premiers indices phosphatés au Maroc (bassin des Meskala)
1912	Découverte du phosphate exploitable à Oued-Zem
7 août 1920	Création officielle de l'Office Chérifien des Phosphates
1921	Début d'exploitation effective dans la région d'Oued-Zem
23 juillet 1921	Première exportation de phosphate vers le port de Casablanca
1930	Ouverture du centre de production de Youssoufia
1950-1952	Mise en œuvre de la méthode d'extraction en découverte à Khouribga
1958	Création du centre de formation professionnelle à Khouribga
1960-1965	Développement de la mécanisation souterraine à Youssoufia
1965	Démarrage de Maroc Chimie à Safi
1974	Création de l'Institut de Promotion Socio-Éducative
1981	Démarrage de Maroc Phosphore II
1986	Démarrage du site de Jorf Lasfar avec Maroc Phosphore III-IV
1998	Création de l'usine EMAPHOS pour l'acide purifié
1999	Création de l'usine IMACID pour l'acide phosphorique
2008	Transformation du groupe OCP en société anonyme
2010	Mise en service de la laverie de Merah Lahach

1.2.3 Activités et Chaîne de Valeur

Le Groupe OCP maîtrise l'intégralité de la chaîne de création de valeur de l'industrie phosphatière, depuis l'extraction jusqu'à la commercialisation des produits finis :

1. Extraction : Réalisée soit en découverte (ciel ouvert), soit en galeries souterraines, selon quatre étapes principales :

- Forage
- Sautage
- Décapage
- Défruitage

2. Traitement : Opérations de purification incluant le criblage, le séchage, la calcination, la flottation et l'enrichissement à sec.

3. Transformation chimique : Production d'acide phosphorique de base, d'acide phosphorique purifié et d'engrais solides dans les complexes industriels de Safi et Jorf Lasfar.

4. Commercialisation : Export de 95% de la production vers les cinq continents, positionnant l'OCP comme leader mondial.

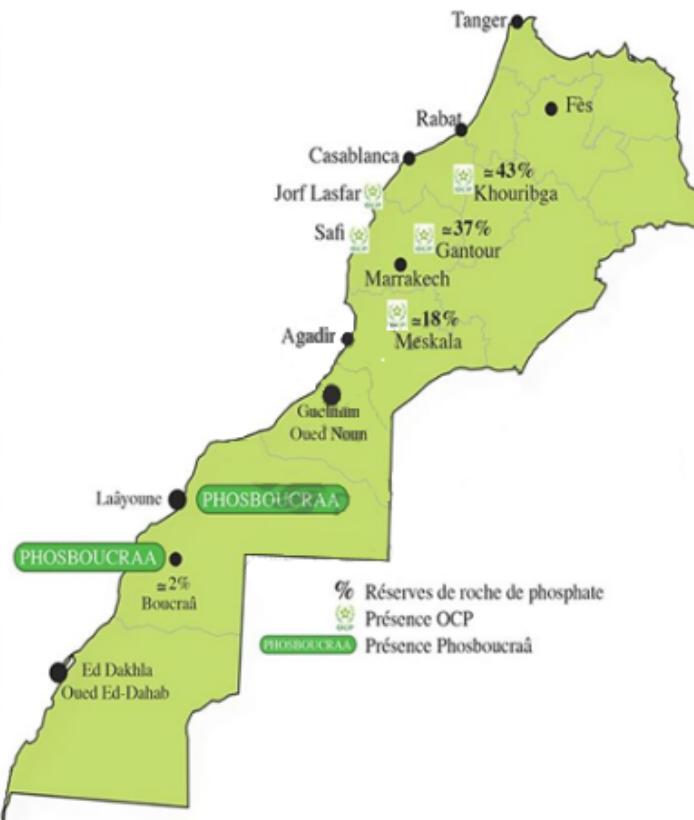


FIGURE 1.2 – Implantation géographique du Groupe OCP

1.2.4 Statut Juridique et Organisation

L’OCP SA fonctionne comme une société anonyme de nature commerciale et industrielle, avec l’État marocain comme unique actionnaire. Cette structure lui confère :

- Une autonomie de gestion financière complètement séparée de celle de l’État
- Une organisation souple et dynamique similaire aux grandes entreprises concurrentes
- Un directeur général nommé par Dahir Royal
- Un conseil d’administration présidé par le Chef du Gouvernement
- Les mêmes obligations fiscales que toute entreprise privée

1.2.5 Filiales Principales

Le Groupe OCP s’appuie sur un réseau de filiales spécialisées :

Filiales opérationnelles :

- **Maroc Phosphore** : Production d’acide phosphorique et d’engrais chimiques
- **Phosboucraâ** : Extraction et commercialisation du mineraï de phosphate
- **SOTREG** : Transport du personnel du Groupe

- **SMESI** : Ingénierie et réalisations industrielles
 - **CERPHOS** : Centre de recherche spécialisé dans les phosphates
- Filiales de développement :**
- **OCP Innovation Fund for Agriculture** : Fonds d'investissement agricole
 - **SADV** : Aménagement et développement de projets urbains écologiques
 - **OCP International** : Gestion des participations internationales

1.3 Présentation du Service Informatique de Khouribga

1.3.1 Positionnement Organisationnel

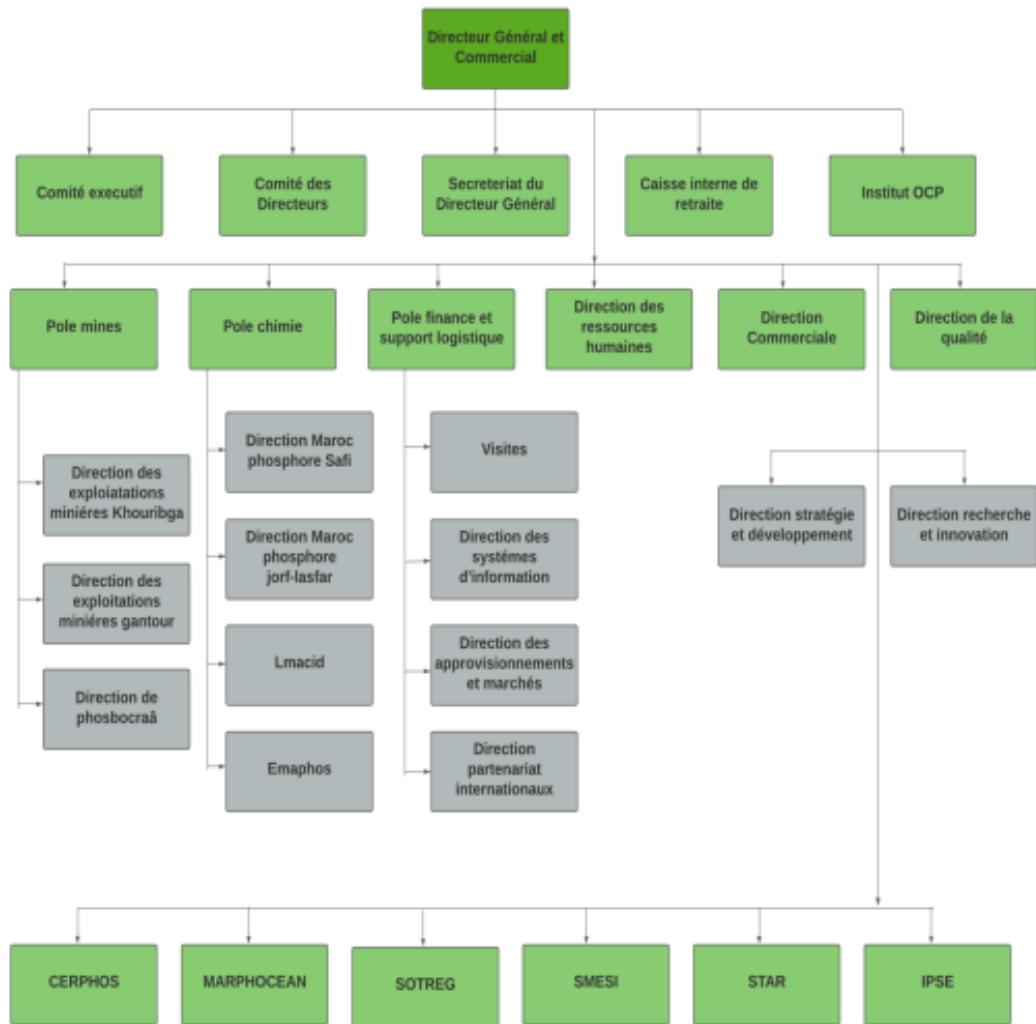


FIGURE 1.3 – Structure organisationnelle du Groupe OCP

Le service informatique de la zone Khouribga (DSI/O/P/K), comme l'ensemble des centres de production informatique du Groupe OCP, est placé sous la gestion de la Direction des Systèmes d'Information basée à Casablanca. Il représente localement la DSI avec toutes ses composantes auprès de la Direction des Exploitations Minières de Khouribga.

1.3.2 Missions Principales

Le service informatique de Khouribga assume des responsabilités stratégiques essentielles :

Gestion technique :

- Administration et maintenance du parc informatique
- Gestion des serveurs et de l'infrastructure réseau
- Maintenance matérielle et logicielle des équipements
- Supervision du réseau 24h/24

Support utilisateurs :

- Assistance technique pour la résolution des incidents informatiques
- Formation et accompagnement des utilisateurs
- Gestion centralisée via l'outil Service Desk

Projets et développement :

- Pilotage des projets IT au niveau local
- Coordination avec les autres sites du groupe
- Participation aux initiatives de digitalisation

1.3.3 Structure Organisationnelle du Service

Le service informatique de Khouribga (DSI/O/P/K), intégré au Digital Office (DO), est structuré autour de quatre pôles principaux :

DO Réseaux & Télécoms

- Gestion et supervision des infrastructures réseau et télécommunication.
- Garantie de la connectivité entre les différents sites (MPLS, WAN, LAN, etc.).
- Mise en place et suivi des équipements actifs (switches, routeurs, firewalls).
- Support technique lié aux services de communication.

DO User Services

- Support utilisateurs et assistance de proximité.

- Gestion et suivi des tickets via l'outil Service Desk.
- Amélioration continue de la qualité de service aux utilisateurs.
- Coordination des interventions sur site.

DO Infrastructure

- Administration et supervision des serveurs et systèmes.
- Maintenance préventive et corrective des infrastructures informatiques.
- Gestion des datacenters et des ressources matérielles.
- Suivi de la performance et de la disponibilité des systèmes.

DO Développement

- Conception et développement de solutions digitales locales.
- Automatisation des processus métiers.
- Intégration des applications avec les systèmes existants.
- Contribution à la transformation digitale de la zone Khouribga.

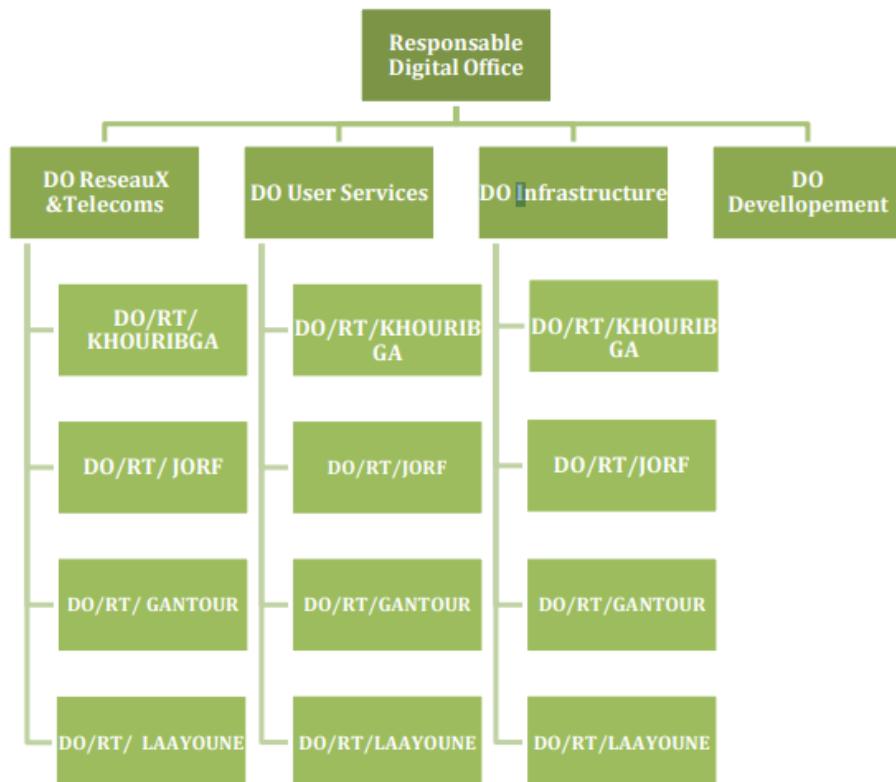


FIGURE 1.4 – Organisation du Digital Office de Khouribga

1.3.4 Infrastructure Réseau

Le service informatique de Khouribga s'intègre dans l'architecture réseau globale du Groupe OCP, caractérisée par :

Topologie réseau : Architecture en étoile avec Casablanca comme centre névralgique et chaque site représentant un point central pour ses zones environnantes.

Connectivité inter-sites : Liaison MPLS sécurisée assurant :

- Transfert de données rapide et fiable
- Isolation du trafic par rapport au réseau public
- Qualité de service (QoS) adaptée aux besoins métiers
- Interconnexion avec Casablanca, Jorf Lasfar, Gantour et Laâyoune

Équipements et systèmes : Gestion d'un parc informatique diversifié incluant micro-ordinateurs, serveurs, équipements réseau et périphériques spécialisés.

1.3.5 Rôle Stratégique

Le service informatique de Khouribga joue un rôle crucial dans :

Support opérationnel : Garantir la continuité des systèmes d'information essentiels aux activités minières et administratives du site.

Innovation digitale : Accompagner la transformation numérique des processus métiers et l'intégration de nouvelles technologies.

Coordination inter-sites : Faciliter la circulation des informations entre les différentes directions du groupe et maintenir la cohérence des systèmes.

Sécurité informatique : Assurer la protection des données et la sécurité des infrastructures critiques.

1.4 Cadrage du projet

1.4.1 Contexte du projet

Ce stage s'inscrit dans le cadre stratégique de digitalisation et d'industrie 4.0 du Groupe OCP. Le besoin identifié portait sur le développement d'une plateforme intelligente de maintenance prédictive visant à moderniser la gestion des interventions et à anticiper les défaillances des équipements critiques, en s'appuyant sur les données issues de capteurs IoT.

Contrairement à une approche traditionnelle de maintenance corrective ou préventive, ce projet explore une approche data-driven et proactive. L'objectif principal était de concevoir et déployer un système intégré capable de :

- Centraliser et traiter en temps réel des flux de données sensorelles continus.
- Appliquer des modèles d'intelligence artificielle pour prédire les probabilités de défaillance.
- Offrir une visibilité en temps réel sur la santé des assets via un tableau de bord interactif.
- Digitaliser et rationaliser le workflow de gestion des interventions (création, assignation, résolution).

1.4.2 Problématique

Dans le contexte actuel de la maintenance industrielle chez OCP, la problématique se décline comme suit :

- La gestion des interventions repose encore sur des processus partiellement manuels et des suivis par outils bureautique (Excel), générant des délais de traitement, un manque de traçabilité et une difficulté à prioriser les actions.
- La multitude de données sensorelles générées par les équipements n'est pas exploitée pour anticiper les pannes. Ces données sont souvent stockées sans être analysées, représentant une perte d'information critique.
- L'absence de couplage entre les données temps réel (IoT) et le système de gestion des maintenance empêche toute action proactive. Les interventions sont majoritairement réactives, suite à une défaillance avérée.
- Les analyses sous Excel sont longues, fragiles et difficilement actionnables en temps réel pour le pilotage opérationnel.

1.4.3 Objectifs

Les objectifs de ce projet sont multiples et ambitieux, visant à adresser l'ensemble de la problématique :

Objectifs Techniques :

- Concevoir et développer une application monolithique modulaire en utilisant Spring Boot pour le backend et React pour le frontend.
- Intégrer une plateforme de streaming de données (Apache Kafka) pour ingérer et traiter les flux IoT en temps réel.
- Développer et entraîner un modèle de Machine Learning performant pour la classification des défaillances à partir d'un jeu de données public (Siemens Smart Manufacturing Lab).

- Opérationnaliser (MLOps) le modèle en le convertissant au format ONNX et en l'intégrant dans le pipeline Kafka pour de l'inférence en temps réel avant le stockage en base MySQL.
- Développer un tableau de bord interactif pour visualiser l'état des équipements, les prédictions de défaillance et l'historique des interventions.

Objectifs Fonctionnels et Métier :

- Digitaliser le workflow des interventions pour trois profils : l'Utilisateur (création de la demande), l'Administrateur (assignation au technicien) et le Technicien (traitement et clôture).
- Fournir une vision prédictive de la santé du parc machine en affichant les probabilités de défaillance calculées par le modèle.
- Centraliser l'information dans une base de données unique et fiable, servant de source de vérité pour les données techniques et métier.
- Jeter les bases techniques pour une future automatisation où une prédiction de panne critique déclencherait automatiquement la création et l'assignation d'un ticket d'intervention.

1.4.4 Vision et Motivation

La vision de ce projet est de doter la maintenance d'une plateforme unifiée et prédictive qui automatise la collecte des données, l'analyse prédictive et la gestion des actions correctives. L'ambition est de transformer un processus historiquement réactif (intervenir après la panne) en un workflow digital fluide, intelligent et proactif, capable de fournir des indicateurs fiables sur la santé des équipements et de prioriser les interventions pour éviter les arrêts non planifiés.

Ce projet est motivé par la nécessité de moderniser les pratiques de maintenance, de réduire les coûts liés aux arrêts machines et aux maintenances non optimisées, et d'exploiter le potentiel des données déjà disponibles. En centralisant les informations et en standardisant les processus, la solution améliore la qualité de l'information, renforce la réactivité des équipes et soutient une culture data-driven. À terme, elle apporte une valeur concrète : une meilleure disponibilité des équipements, une allocation optimisée des ressources techniques et une maintenance plus efficiente.

1.4.5 Conclusion

Ce chapitre a présenté l'organisme d'accueil, le Groupe OCP, et a situé le contexte stratégique de la maintenance industrielle au sein de son écosystème. Il a ensuite cadré

le projet autour d'une problématique duale : d'une part, la gestion encore largement manuelle et non intégrée des interventions de maintenance, et d'autre part, l'absence d'exploitation des données sensorelles pour anticiper les défaillances des équipements. La solution proposée s'articule autour de deux piliers complémentaires : un système de gestion digital des interventions (création, assignation, suivi par les acteurs Admin, Technicien, Utilisateur) et un pipeline de maintenance prédictive temps réel (ingestion de flux IoT via Kafka, enrichissement par un modèle de Machine Learning pour la prédiction de défaillances, et stockage des données enrichies). Ensemble, ces briques techniques visent à fiabiliser la donnée, à fournir une vision prédictive de la santé des assets et à structurer un pilotage opérationnel aussi bien réactif que proactif. La suite de ce rapport détaillera l'architecture technique, les choix de modélisation et les résultats obtenus lors de la mise en œuvre.

Chapitre 2

Préliminaires théoriques et techniques

2.1 Traitement de flux de données et Apache Kafka

La maintenance prédictive moderne nécessite une capacité à traiter des données en temps réel. Le traitement de flux (*stream processing*) analyse les données **au fur et à mesure de leur génération**, permettant une réaction et une prise de décision immédiates, contrairement au traitement par lots (*batch*) qui opère sur des données historiques.

Apache Kafka s'est imposé comme la solution standard pour construire des pipelines de données temps réel fiables et évolutifs. C'est une plateforme distribuée de streaming qui fonctionne sur le modèle producteur-consommateur. Ses concepts clés sont illustrés dans la figure 2.1 et expliqués ci-dessous :

- **Producer** (Producteur) : Application ou service qui publie (envoie) des messages dans un topic Kafka. Dans notre architecture, le producer est un module Spring Boot qui récupère les données de l'API de streaming et les envoie vers le topic `sensor-data`.
- **Kafka Cluster** (Cluster Kafka) : Ensemble de serveurs (brokers) travaillant ensemble pour former une plateforme Kafka distribuée et fault-tolerant.
- **Broker** : Serveur Kafka individuel dans un cluster. Chaque broker stocke une partie des données et participe au traitement des messages.
- **Partition** (Partition) : Sous-division d'un topic. Les partitions permettent de paralléliser le traitement en distribuant les données sur plusieurs brokers. Les messages d'un topic sont répartis entre ses partitions.
- **ZooKeeper** : Service external utilisé par Kafka (dans les versions traditionnelles) pour la gestion des métadonnées du cluster, incluant l'état des brokers, les topics, les partitions, et l'élection des leaders. Il maintient la coordination et la cohérence dans le cluster distribué.

- **Consumer** (Consommateur) : Application ou service qui s'abonne à un ou plusieurs topics pour lire et traiter les messages. Dans notre architecture, le consumer est un module Spring Boot qui : (1) lit les messages du topic `sensor-data`, (2) effectue une prédiction en utilisant le modèle ONNX, et (3) persiste les données enrichies dans la base MySQL.
- **Consumer Group** (Groupe de consommateurs) : Ensemble de consumers qui travaillent ensemble pour consommer les messages d'un topic. Les partitions du topic sont réparties entre les membres du groupe, permettant un traitement parallèle et scalable.

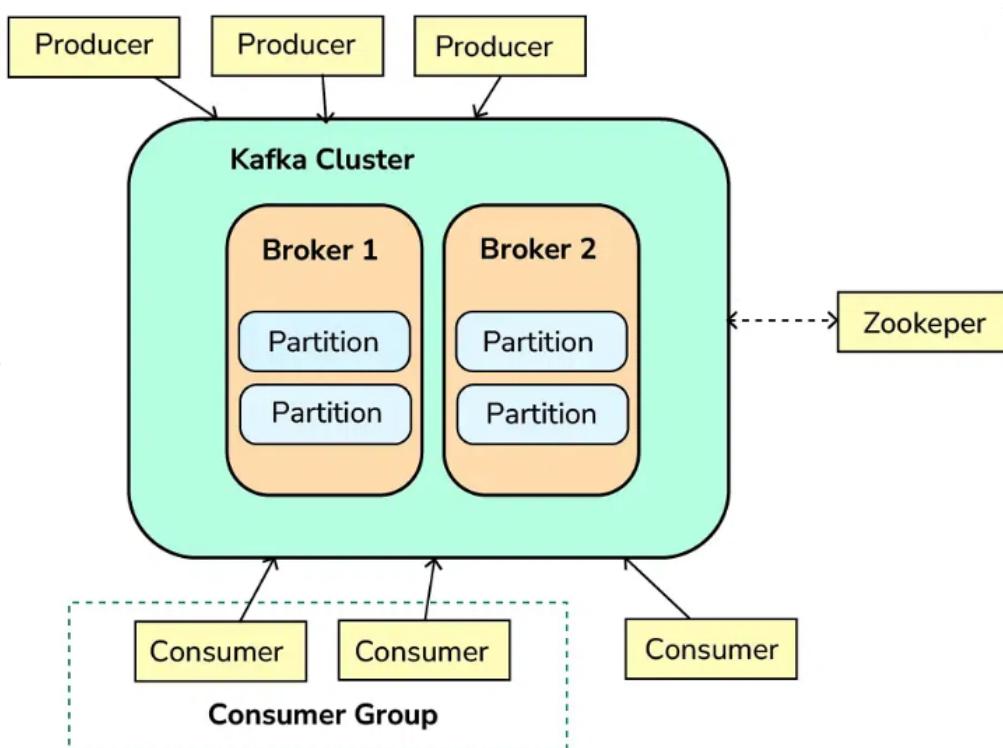


FIGURE 2.1 – Architecture Kafka

2.1.1 Évolution vers KRaft

Il est important de noter que les versions récentes de Kafka (depuis la version 3.3) introduisent **KRaft** (**Kafka Raft metadata mode**), qui élimine la dépendance à ZooKeeper. Dans le mode KRaft, les brokers Kafka eux-mêmes forment un quorum Raft pour gérer leurs propres métadonnées de manière interne, simplifiant ainsi radicalement l'architecture et améliorant la scalabilité et les performances.

Bien que notre démonstrateur utilise actuellement l'architecture avec ZooKeeper pour des raisons de compatibilité, le choix de KRaft a été validé comme la direction

future pour une mise en production, alignant ainsi notre solution sur l'évolution récente de la plateforme Kafka.

Dans notre architecture, Kafka joue le rôle de **colonne vertébrale centrale et découpleur**. Il permet à notre service producteur (qui ingère le flux de l'API) et à notre service consommateur (qui prédit et persiste) de fonctionner de manière indépendante, asynchrone et résiliente.

2.2 Machine Learning pour la maintenance prédictive

Le Machine Learning (ML) est le domaine de l'intelligence artificielle qui permet à un système d'"apprendre" à partir de données pour effectuer une tâche sans être explicitement programmé pour cela. Dans le contexte de notre projet, la tâche est un **problème de classification binaire supervisée** : prédire si une défaillance surviendra (`failure_occurrence = 1`) ou non (`failure_occurrence = 0`) en fonction des valeurs des capteurs à un instant t , ainsi que la probabilité associée à cette prédiction.

2.2.1 Le processus de ML

1. **Entraînement et sélection du modèle** : Dix algorithmes différents, incluant des versions de base et optimisées (tunées) de XGBoost, LightGBM, Random Forest, Extra Trees et Hist Gradient Boosting, ont été entraînés et comparés sur un jeu de données public de maintenance prédictive. La sélection du modèle final (LightGBM tuné) s'est basée sur un compromis optimal entre ses performances en termes de précision, recall, F1-score et sa vitesse d'inférence, cruciale pour notre application temps réel.
2. **Optimisation pour la production** : Le modèle LightGBM entraîné au format `.pkl` a été converti au format ONNX (Open Neural Network Exchange), un standard ouvert pour l'échange de modèles. Cette conversion est cruciale pour une intégration performante et native dans un environnement d'exécution Java/Spring Boot.

2.2.2 Algorithme LightGBM

LightGBM (Light Gradient Boosting Machine) est un framework de boosting basé sur les arbres de décision, développé par Microsoft. Il est particulièrement réputé pour :

- Sa **vitesse d'entraînement et d'inférence** très élevée, essentielle pour notre application temps réel.
- Sa **faible consommation mémoire**, permettant un déploiement efficient.

- Sa **grande précision** sur des données tabulaires, comme celles issues de capteurs.

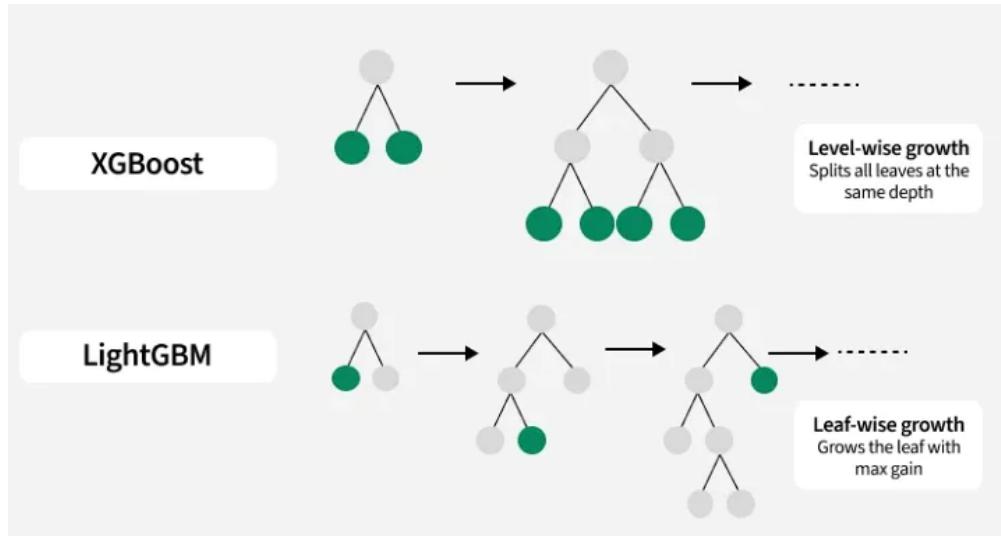


FIGURE 2.2 – Comparaison des stratégies de croissance d’arbres : XGBoost (niveau par niveau) vs LightGBM (feuille par feuille)

Son principe est de combiner séquentiellement plusieurs modèles simples (arbres de décision faibles) pour créer un modèle fort et précis. Contrairement à XGBoost qui utilise une croissance niveau par niveau (*level-wise*), LightGBM utilise une croissance feuille par feuille (*leaf-wise*) qui sélectionne la feuille avec le gain maximum à chaque division, comme illustré dans la Figure 2.2. Cette approche permet une convergence plus rapide et de meilleures performances.

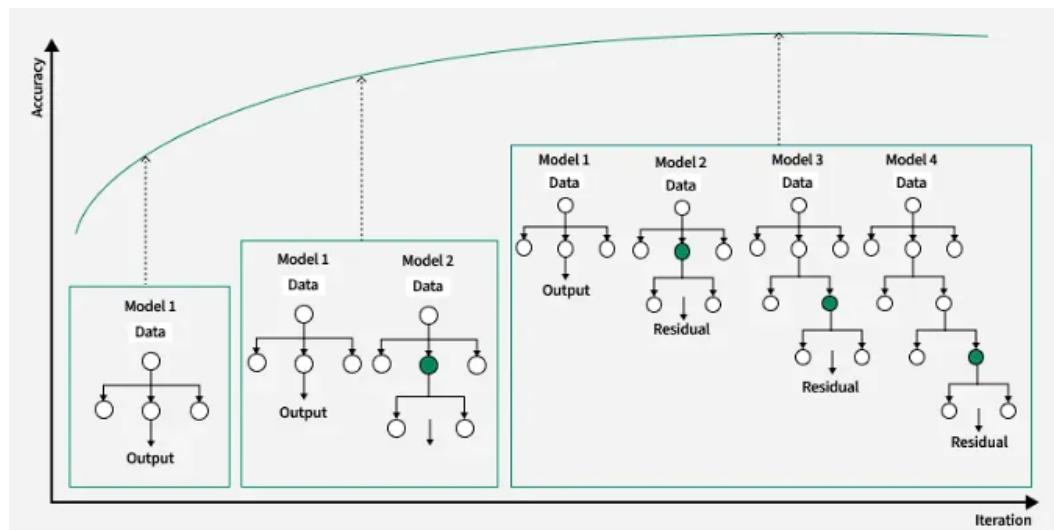


FIGURE 2.3 – Principe de l’apprentissage par gradient boosting : correction successive des résidus

Le mécanisme de boosting, illustré dans la Figure 2.3, fonctionne de manière itérative : chaque nouveau modèle apprend à prédire les erreurs (résidus) du modèle précédent, affinant ainsi progressivement la prédiction finale.

2.2.3 Prédiction et interprétation des résultats

Le modèle final retenu produit deux types de sorties pour chaque entrée de données capteurs :

- **Classe prédictive** : Une valeur binaire (0 ou 1) indiquant si une défaillance est prédictive (`failure_occurrence`).
- **Probabilité de défaillance** : Un score continu entre 0 et 1 représentant la probabilité estimée qu'une défaillance survienne. Ce score permet une interprétation plus nuancée et un seuillage adaptable selon la criticité de l'équipement.

Cette approche à double sortie offre à la fois une décision binaire claire pour les alertes automatiques et une mesure probabiliste pour l'analyse fine et la priorisation des interventions de maintenance.

2.3 Opérationnalisation des modèles (MLOps) et intégration ONNX

Opérationnaliser un modèle (MLOps) va au-delà de son entraînement ; il s'agit de l'intégrer de manière fiable et performante dans une application de production.

Le choix du format ONNX (Open Neural Network Exchange) a été stratégique pour ce projet. Ce standard ouvert permet :

L'interopérabilité : Exécuter un modèle entraîné en Python (via lightgbm) dans un environnement d'exécution Java (notre application Spring Boot) grâce à la bibliothèque ONNX Runtime.

La performance : L'inférence (la prédiction) est effectuée par le runtime ONNX, qui est hautement optimisé, permettant une latence minimale cruciale pour le traitement de flux en temps réel.

La portabilité : Le même fichier modèle .onnx est indépendant du framework d'apprentissage initial.

Dans notre architecture, le service consommateur Kafka intègre le runtime ONNX. Pour chaque message reçu (une ligne de données de capteurs), il utilise le modèle ONNX pour effectuer une prédiction avant de persister la donnée enrichie dans la base MySQL.

2.4 Architecture logicielle : Application Monolithique Modulaire

Notre application suit une architecture **monolithique modulaire** déployée comme une seule unité, mais avec une séparation claire des responsabilités en couches et modules au sein du codebase Spring Boot. Cette approche contraste avec une architecture microservices mais était parfaitement adaptée à l'ampleur de ce projet, offrant **simplicité de déploiement et de développement**.

La stack technique retenue est :

- **Spring Boot (Backend)** :

Framework Java pour développer une application backend robuste. Il héberge :

- Les contrôleurs REST pour l'API de gestion des interventions.
- Les services métier pour la logique applicative.
- Les classes producteur et consommateur Kafka intégrées directement au projet.
- Le service de chargement et d'inférence du modèle ONNX.

- **React (Frontend)** :

Bibliothèque JavaScript pour construire l'interface utilisateur interactive qui visualise les données des capteurs, les prédictions et permet la gestion des interventions.

- **MySQL (Base de données)** :

Système de gestion de base de données relationnelle pour stocker de manière fiable les données structurées : tables des interventions, utilisateurs, et `sensor_data` enrichie des prédictions.

La communication entre le frontend React et le backend Spring Boot se fait via une **API RESTful standard**.

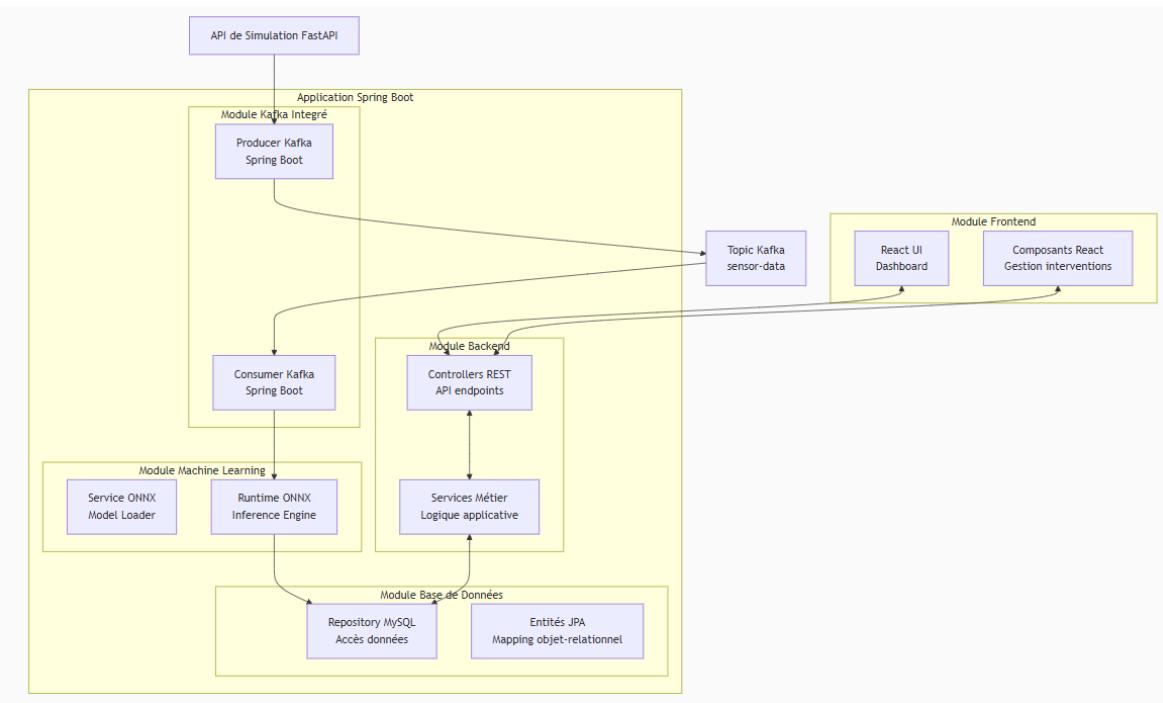


FIGURE 2.4 – Diagramme de l’architecture monolithique modulaire montrant les modules internes et les flux de données.

2.5 Simulation des données IIoT via une API de streaming

Étant donné l’indisponibilité de données sensorelles réelles en direct, une API de simulation a été développée pour émuler le comportement d’un capteur industriel. Basée sur FastAPI, cette API :

Charge un jeu de données public prédécoupé (`RealTime_IoT_PredictiveMaintenance.csv`).

Expose un endpoint de streaming (`/api/stream`) qui envoie une ligne de données toutes les 60 secondes, mimant ainsi un flux de données IIoT en temps réel.

Formatte les données en JSON et les envoie via le protocole Server-Sent Events (SSE) pour une consommation facile par notre producteur Kafka.

Cette approche a permis de tester l’intégralité du pipeline en conditions quasi-réelles sans dépendre d’une infrastructure physique complexe.

2.6 Conclusion

Ce chapitre a posé le cadre théorique et technique unifiant les concepts de traitement de flux avec Kafka, de machine learning opérationnalisé avec ONNX, et d'architecture applicative avec Spring Boot. La force de cette architecture réside dans son couplage lâche grâce à Kafka et dans l'efficacité de l'inférence temps réel permise par le format ONNX. La simulation des données IIoT via une API a fourni un flux de données fiable pour valuer l'ensemble de la chaîne de traitement, de l'acquisition à la visualisation en passant par la prédiction. Le chapitre suivant détaillera la conception concrète et la mise en œuvre de cette solution basée sur ces préliminaires.

Chapitre 3

Conception et Méthodologie

3.1 Introduction

L'objectif principal de ce projet est de développer une plateforme de maintenance prédictive qui intègre à la fois la gestion digitale des interventions et l'analyse prédictive en temps réel des données capteurs. Ce chapitre présente l'architecture complète du système, les différents modules techniques, les choix technologiques effectués ainsi que les workflows de traitement des données. Une attention particulière est portée sur la méthodologie de traitement du flux de données et l'intégration du modèle de machine learning au sein de l'application Spring Boot.

3.2 Analyse des Fonctionnalités et Cas d'Utilisation

3.2.1 Fonctionnalités Principales

Notre plateforme offre deux catégories de fonctionnalités principales :

Gestion des interventions

- Création, consultation et suivi des demandes d'intervention
- Assignation des techniciens aux interventions
- Clôture et historique des interventions

Maintenance prédictive

- Acquisition en temps réel des données capteurs via API de simulation
- Prédiction des défaillances par modèle machine learning
- Alertes et seuils de criticité configurables
- Visualisation des probabilités de défaillance

3.2.2 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation ci-dessous illustre les interactions entre les différents acteurs et le système :

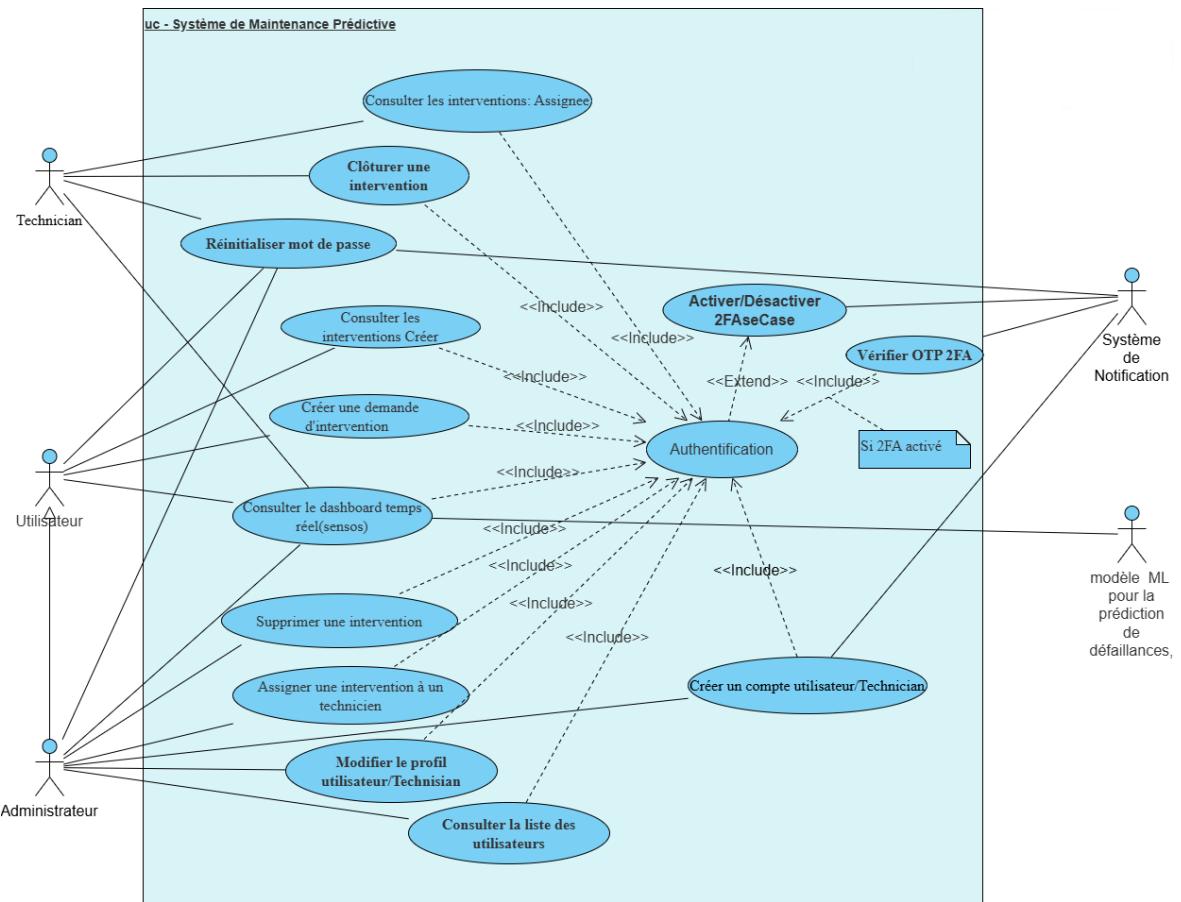


FIGURE 3.1 – Diagramme de cas d'utilisation de la plateforme

Les acteurs principaux sont :

- **Utilisateur** : Peut créer des demandes d'intervention et consulter l'état des machines.
- **Technicien** : Consulte les interventions assignées et met à jour leur statut
- **Administrateur** : Gère l'assignation des interventions, les Utilisateur et les Technicien.

3.3 Architecture du Système

3.3.1 Vue d'ensemble

Notre architecture suit une approche monolithique modulaire avec les composants principaux suivants :

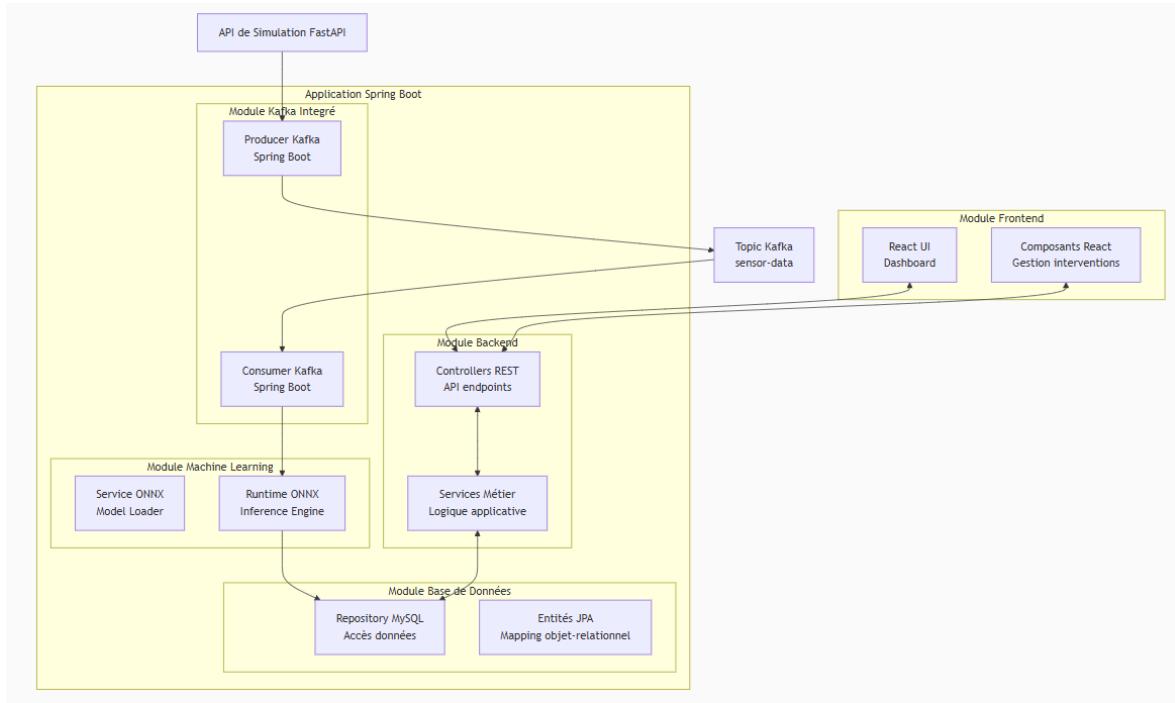


FIGURE 3.2 – Architecture globale du système de maintenance prédictive

3.3.2 Description des Modules

Module d'Acquisition de Données

Collecte et ingestion des données capteurs simulées :

- API FastAPI générant un flux de données selon le dataset Siemens
- Producer Kafka publiant les messages dans le topic **sensor-data**
- Format JSON standardisé avec métadonnées techniques
- Protocole Server-Sent Events pour le streaming

Module de Traitement Temps Réel

Cœur du système prédictif :

- Consumer Kafka traitant les messages en temps réel

- Runtime ONNX exécutant le modèle LightGBM pour l'inférence
- Enrichissement des données avec prédictions (défaillance + probabilité)
- Persistance des données enrichies en MySQL

Module de Gestion des Interventions

Workflow complet de maintenance :

- API REST Spring Boot pour le CRUD des interventions
- Système d'authentification avec rôles (Admin, Technicien, Utilisateur)
- Workflow de statut : Crée → En cours(Assigné) → Terminé
- Notifications email pour la vérification OTP (2FA , password reset) et la création de compte

Module de Visualisation

Interface React moderne :

- Dashboard temps réel des données capteurs
- Visualisation des probabilités de défaillance
- Graphiques interactifs avec Recharts.js

Module de Persistance

Infrastructure data :

- MySQL relationnel avec tables normalisées
- Spring Data JPA pour l'accès aux données

Module de Sécurité

Gestion centralisée de la sécurité et conformité :

- Spring Security avec JWT et authentification 2FA
- Gestion des sessions et monitoring des connexions
- Politiques de mots de passe
- Contrôle d'accès basé sur les rôles

3.4 Workflows de Traitement

3.4.1 Workflow de Prédiction Temps Réel

Le diagramme de séquence suivant illustre le traitement d'une donnée capteur :

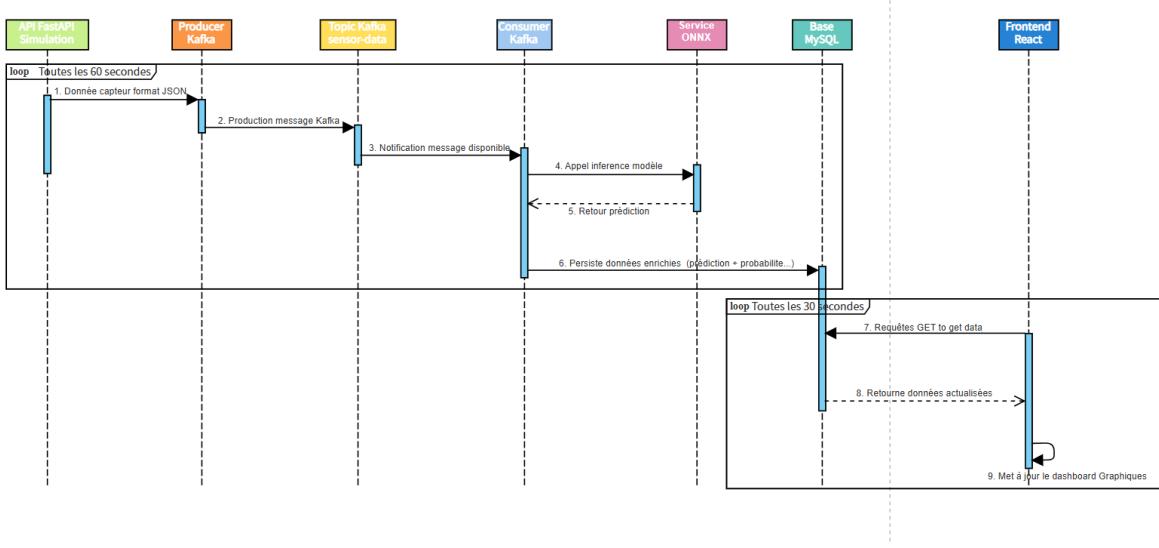


FIGURE 3.3 – Workflow de prédiction temps réel avec polling

1. **API FastAPI de simulation** génère une donnée capteur au format JSON **toutes les 60 secondes**
2. Le **Producer Kafka** publie le message dans le topic **sensor-data**
3. Le **Consumer Kafka** est notifié de la disponibilité d'un nouveau message
4. Le Consumer appelle le **Service ONNX** pour effectuer l'inférence
5. Le modèle retourne la prédiction (défaillance) et la probabilité associée
6. Les données enrichies (données brutes + prédiction + probabilité) sont persistées en base **MySQL**
7. Le **Frontend React** effectue des requêtes GET périodiques **toutes les 30 secondes** via un intervalle timer
8. La base MySQL retourne les données actualisées incluant les dernières prédictions
9. Le frontend met à jour le dashboard et les graphiques en temps réel

Note : Le frontend utilise une approche de polling régulier plutôt que des WebSockets pour la mise à jour des données. Un intervalle de 30 secondes permet un équilibre entre reactivité et performance.

3.4.2 Workflow de Gestion d'Intervention

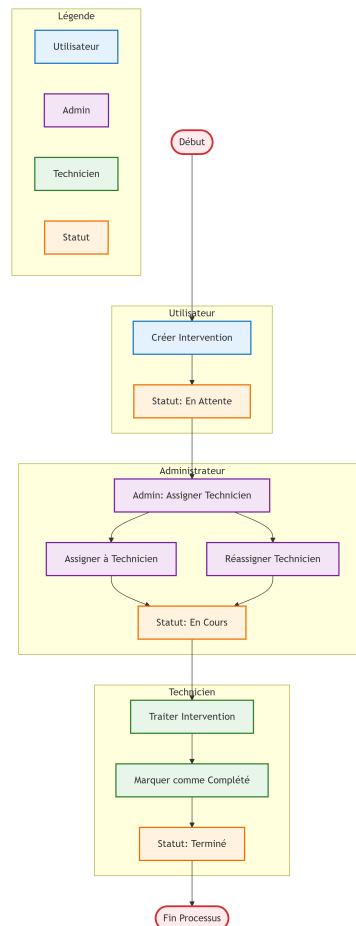


FIGURE 3.4 – Workflow de gestion d'intervention

3.5 Défis et Solutions

3.5.1 Défis Techniques Rencontrés

- **Intégration ONNX/Spring Boot :** Solution via l'API native ONNX Runtime pour Java
- **Latence de prédiction :** Optimisation du modèle et format ONNX pour des inférences rapides

- **Synchronisation temps réel** : Implémentation d'un mécanisme de polling efficace pour les mises à jour
- **Qualité des données simulées** : Validation et formatage strict des messages JSON

3.5.2 Solutions Implementées

- **Service de chargement et de cache du modèle ONNX au démarrage** : Le modèle LightGBM converti au format ONNX est chargé une seule fois au démarrage de l'application et maintenu en mémoire pour des inférences à latence minimale.
- **Pool de connexions Kafka pour une meilleure performance** : Mise en place d'un pool de connexions Kafka réutilisables pour éviter la surcharge liée à la création et destruction de connexions pour chaque message.
- **Indexation stratégique par timestamp** : Implémentation d'un index sur le champ timestamp permettant une récupération optimale des données les plus récentes, essentielle pour l'actualisation périodique du dashboard.
- **Optimisation des requêtes de polling** : Requêtes SQL optimisées avec LIMIT et WHERE timestamp > [dernière_valeur] pour minimiser la quantité de données transférées lors des actualisations périodiques.

3.6 Conclusion

Ce chapitre a présenté la conception détaillée de notre plateforme de maintenance prédictive. L'architecture modulaire choisie permet une séparation claire des responsabilités fonctionnelles tout en conservant les avantages de simplicité d'une application monolithique.

Les choix techniques stratégiques incluent l'intégration de Kafka comme colonne vertébrale de communication pour le découplage entre l'acquisition et le traitement des données, et l'utilisation d'ONNX pour une inférence performante des modèles de machine learning directement dans l'environnement Java.

Les workflows définis assurent un traitement temps réel des données grâce à un mécanisme de polling efficace, maintenant une expérience utilisateur réactive via l'interface React. Les défis techniques identifiés ont été adressés par des solutions robustes garantissant la stabilité et les performances du système.

La conception proposée établit les fondations techniques pour le développement qui sera détaillé dans le chapitre suivant.

Chapitre 4

Technologies et Interfaces

4.1 Introduction

Ce chapitre présente l'écosystème technologique complet qui sous-tend notre plate-forme de maintenance prédictive. Nous détaillons chaque technologie choisie, ses avantages spécifiques dans notre contexte, et comment elles s'articulent pour former une architecture cohérente et performante.

4.2 Spring Boot : Backend Robustesse et Productivité



FIGURE 4.1 – Logo Spring Boot

Spring Boot s'impose comme le framework backend de référence pour notre application. Cette technologie mature offre un écosystème complet qui accélère le développement tout en garantissant robustesse et maintenabilité.

Avantages concrets pour notre projet :

- **Intégration native Kafka** : Support officiel pour la messagerie temps réel sans configuration complexe
- **Auto-configuration intelligente** : Configuration minimale requise pour démarrer rapidement

- **Sécurité intégrée** : Spring Security fournit JWT, OAuth2 et 2FA sans effort supplémentaire
- **Écosystème riche** : Bibliothèques éprouvées pour la data, le caching, le monitoring
- **Productivité développeur** : Hot reload, documentation excellente, communauté active

Architecture backend : Notre application Spring Boot suit une architecture modulaire avec séparation claire des concerns :

- Couche Controller pour les API REST
- Couche Service pour la logique métier
- Couche Repository pour l'accès aux données
- Configuration centralisée pour Kafka et la sécurité

4.3 React + Vite : Interface Moderne et Performante

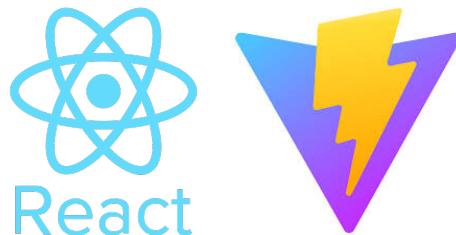


FIGURE 4.2 – Logos React et Vite

Le duo React et Vite constitue le socle de notre interface utilisateur, offrant réactivité et performances exceptionnelles.

- React 19** apporte une gestion d'état prévisible et une architecture composable :
- **Composants réutilisables** : Architecture modulaire pour une maintenance simplifiée
 - **Hooks modernes** : useState, useEffect, useContext pour une logique claire
 - **Virtual DOM** : Mises à jour optimisées pour une expérience fluide
 - **Écosystème riche** : Bibliothèques de qualité pour tous les besoins

Vite révolutionne l'expérience de développement frontend :

- **Build times ultra-rapides** : Compilation quasi-instantanée grâce à ES modules natifs
- **Hot Module Replacement** : Mises à jour à chaud sans recharge de page

4.4 Apache Kafka : Colonne Vertébrale Temps Réel



FIGURE 4.3 – Logo Apache Kafka

Kafka sert de système nerveux central pour notre architecture de streaming de données.

Avantages décisifs :

- **Découplage temporel** : Producteurs et consommateurs indépendants
- **Tolérance aux pannes** : RéPLICATION des données et haute disponibilité
- **Scalabilité horizontale** : Capacité à gérer des volumes croissants
- **Ordering garanti** : Messages traités dans l'ordre par partition

Implémentation dans notre projet :

- Topic unique `sensor-data` pour la simplicité
- Producer Spring Boot pour l'injection des données
- Consumer Spring Boot avec commit manuel pour la fiabilité
- Serialization JSON pour l'interopérabilité

4.5 MySQL : Persistance Fiable et Performante



FIGURE 4.4 – Logo MySQL

MySQL 8.0 offre la robustesse nécessaire pour nos données structurées de maintenance.

Avantages techniques :

- **Transactions ACID** : Intégrité des données garantie
- **Performances** : Indexation avancée et optimiseur de requêtes
- **Sécurité** : Chiffrement, RBAC et auditing
- **Fiabilité** : Technology éprouvée en production

Schéma optimisé :

- Tables normalisées pour éviter la redondance
- Index sur les timestamps pour les requêtes temporelles
- Relations claires entre interventions, utilisateurs, techniciens

4.6 ONNX Runtime : Inférence ML Performante



FIGURE 4.5 – Logo ONNX Runtime

ONNX Runtime permet l'exécution cross-platform de nos modèles de machine learning.

Avantages uniques :

- **Interopérabilité** : Modèle entraîné en Python exécuté en Java
- **Performances** : Optimisations hardware-specific (CPU, GPU)
- **Portabilité** : Même modèle sur différents environnements
- **Standard ouvert** : Support multi-fournisseurs

Intégration Spring Boot :

- Chargement au démarrage de l'application
- Cache en mémoire pour la latence minimale
- API simple pour l'inférence temps réel

4.7 Recharts : Visualisation Data Interactive



FIGURE 4.6 – Logo Recharts

Recharts fournit les visualisations interactives pour notre dashboard temps réel.

Avantages techniques :

- **API déclarative** : Configuration simple et intuitive
- **Responsive native** : Adaptation automatique aux différentes tailles d'écran
- **Personnalisation** : Thématisation cohérente avec notre design system
- **Performances** : Rendering optimisé pour les mises à jour fréquentes

Composants utilisés :

- LineChart pour les tendances temporelles
- BarChart pour les comparaisons
- PieChart pour les répartitions
- Tooltip custom pour les informations détaillées

4.8 Environnement de Développement

Notre environnement a été optimisé pour la productivité et la collaboration :

Outils principaux :

- **IntelliJ IDEA Ultimate** : IDE complet pour le développement Spring Boot
- **VS Code** : Éditeur léger pour le frontend React, ainsi que pour l'exploration de données et la création de modèles de Machine Learning en Python.
- **Git/GitHub** : Pour assurer le suivi et la sauvegarde du travail.
- **Postman** : Tests et documentation des APIs

4.9 Interfaces Utilisateur

4.9.1 Interface d'Authentification - Commune

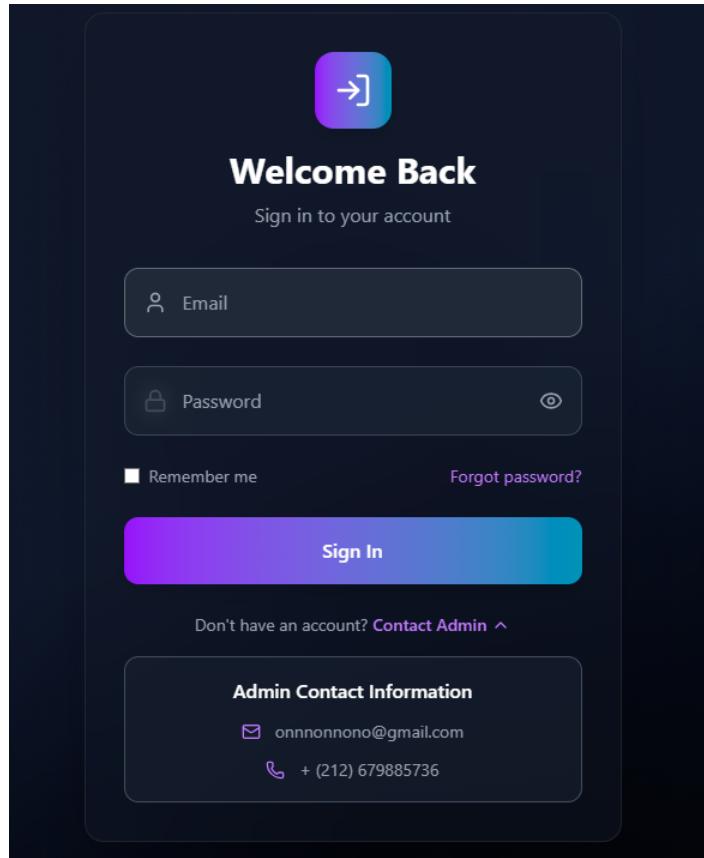


FIGURE 4.7 – Interface de connexion commune

Description : Cette interface constitue le point d'entrée unique et sécurisé de l'application. Elle est conçue pour accueillir les trois profils d'utilisateurs du système : les **utilisateurs** standard, les **techniciens**, et les **administrateurs**. Chaque individu doit s'identifier via cette page.

Fonctionnalités :

- Saisie des identifiants (e-mail).
- Saisie d'un mot de passe sécurisé (masqué).
- Bouton de soumission du formulaire de connexion.
- Mécanisme d'authentification unique qui redirige l'utilisateur vers l'interface adaptée à son rôle (*User, Technicien, Admin*)

4.9.2 Interface de Récupération de Mot de Passe

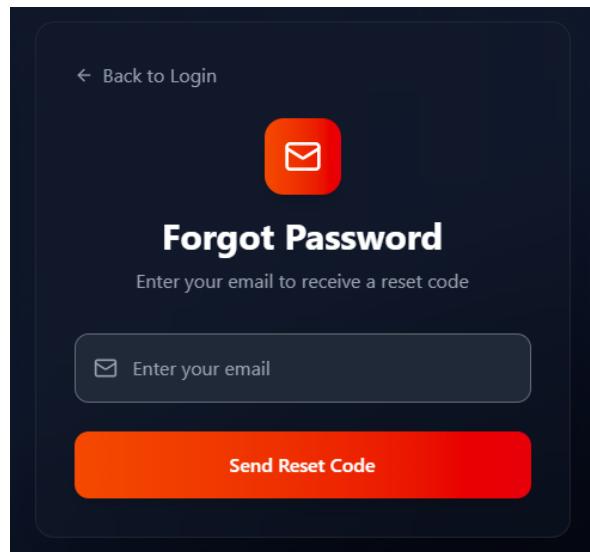


FIGURE 4.8 – Saisie de l'adresse e-mail

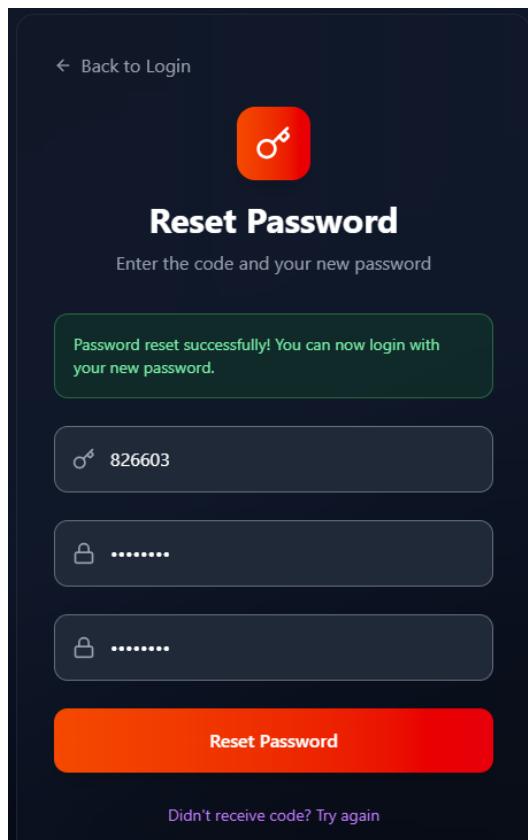


FIGURE 4.9 – Saisie de l'OTP et définition du nouveau mot de passe



FIGURE 4.10 – OTP reçus par e-mail

Description : Ce processus sécurisé permet à l'utilisateur de réinitialiser son mot de passe en vérifiant son identité via un code à usage unique (OTP) envoyé à son adresse e-mail enregistrée.

Fonctionnalités :

- **Étape 1 (Fig. 4.8) : Demande de réinitialisation**
 - Saisie de l'adresse e-mail associée au compte
 - Envoi automatique d'un code OTP par e-mail
 - **Étape 3 (Fig. 4.10) : Réception de l'OTP**
 - Réception du code de vérification dans la boîte mail de l'utilisateur
 - Code à usage unique pour authentifier la demande de réinitialisation
 - **Étape 2 (Fig. 4.9) : Validation et nouveau mot de passe**
 - Saisie du code OTP reçu par e-mail (Fig. 4.10)
 - Définition d'un nouveau mot de passe
 - Confirmation du nouveau mot de passe

4.9.3 Processus de Vérification à Deux Facteurs (2FA)

Lorsque l'authentification à deux facteurs est activée sur un compte, le processus de connexion standard est renforcé par une étape de validation supplémentaire.

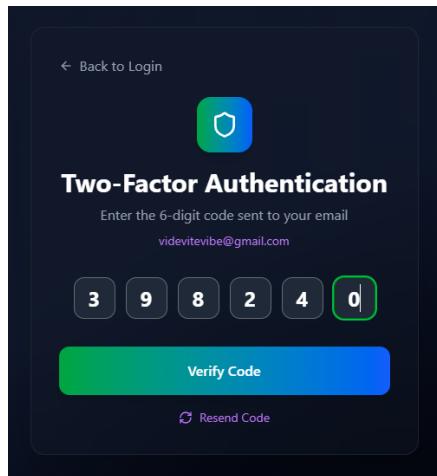


FIGURE 4.11 – Formulaire de vérification du code OTP (2FA) après la connexion

Contexte : Cette interface apparaît immédiatement après qu'un utilisateur a saisi des identifiants valides (e-mail et mot de passe) sur l'interface de connexion commune (Fig. 4.7), mais uniquement si la 2FA est activée pour son compte.

Description : Cette étape sert de second facteur d'authentification. Un code de vérification (OTP) est généré et envoyé à l'adresse e-mail de l'utilisateur. L'accès à l'application n'est accordé qu'après la saisie correcte de ce code.

Fonctionnalités :

- **Message contextuel :** Indique à l'utilisateur qu'un code de sécurité a été envoyé à son adresse e-mail.
- **Champ de saisie :** Permet à l'utilisateur de copier-coller (ou de saisir manuellement) le code OTP reçu dans son e-mail.
- **Bouton de validation :** Soumet le code pour vérification par le système.
- **Lien de renvoi :** Option permettant de renvoyer un nouveau code OTP si l'original n'a pas été reçu ou a expiré.

Processus :

1. L'utilisateur se connecte avec son e-mail et son mot de passe.
2. Le système vérifie que la 2FA est activée pour ce compte.
3. Un code OTP est généré et envoyé automatiquement par e-mail.
4. L'utilisateur consulte sa boîte mail, copie le code.
5. L'utilisateur retourne sur cette interface et colle le code dans le champ prévu à cet effet.
6. Après validation réussie, l'utilisateur est redirigé vers son interface principale (tableau de bord).

4.9.4 Interface de Profil Utilisateur

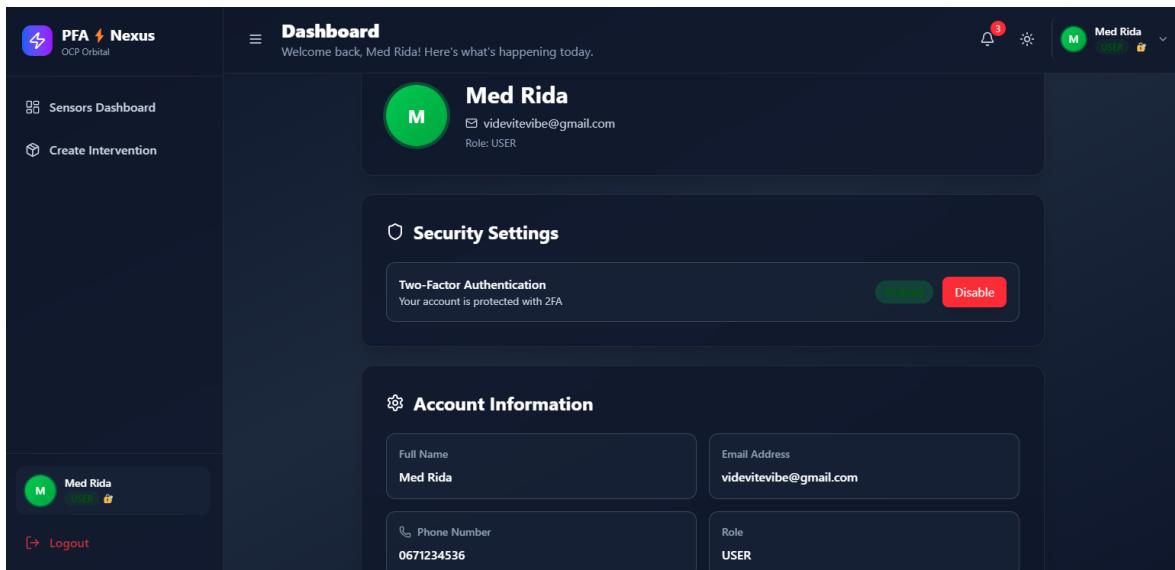


FIGURE 4.12 – Interface de gestion du profil utilisateur

Description : Cette interface permet à l'utilisateur de consulter les informations de son compte et de gérer les paramètres de sécurité, notamment l'activation ou la désactivation de l'authentification à deux facteurs (2FA).

Fonctionnalités :

- **Affichage des informations du compte :**
 - Consultation des données personnelles (nom, prénom, e-mail, etc.).
 - Les informations sont en lecture seule et ne peuvent pas être modifiées depuis cette interface.
- **Gestion de l'Authentification à Deux Facteurs (2FA) :**
 - Affichage du statut actuel de la 2FA (activée ou désactivée).
 - Bouton pour **activer** la 2FA si elle est currently désactivée.
 - Bouton pour **désactiver** la 2FA si elle est currently activée.
- **Processus d'activation de la 2FA :**
 - Lorsque l'utilisateur clique sur « Enable », un code OTP (One-Time Password) est envoyé à son adresse e-mail enregistrée.
 - L'utilisateur doit saisir ce code OTP dans un formulaire de validation pour finaliser l'activation.
 - Après vérification réussie, la 2FA est activée pour le compte.
- **Processus de désactivation de la 2FA :**

- Lorsque l'utilisateur clique sur « Disable », un code OTP de confirmation est envoyé par e-mail.
- L'utilisateur doit saisir et valider ce code pour confirmer la désactivation.
- Après vérification réussie, la 2FA est désactivée pour le compte.

4.9.5 Interface des Interventions Utilisateur

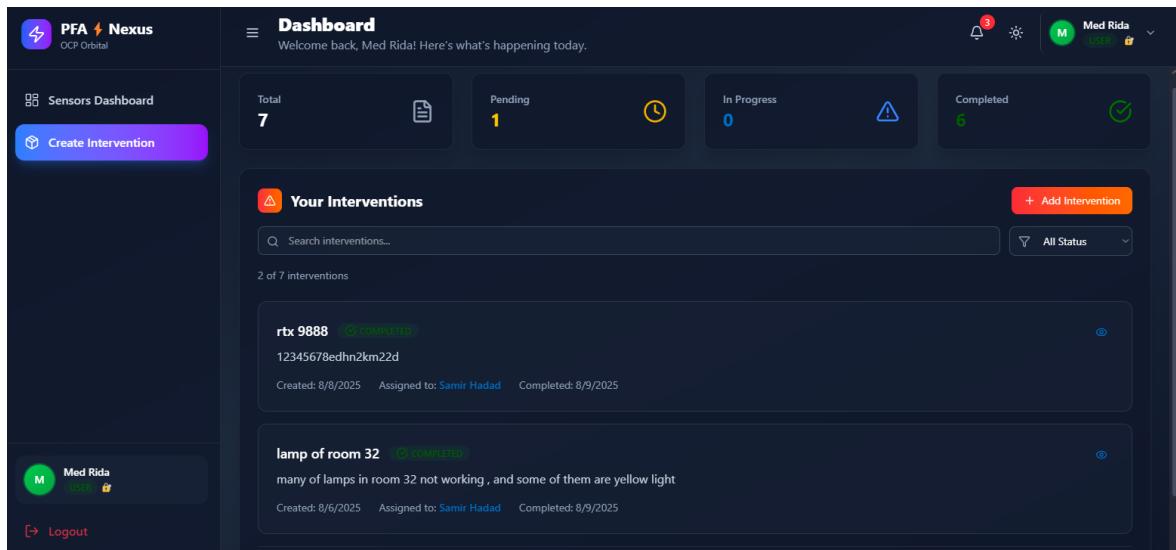


FIGURE 4.13 – Interface principale de création des interventions

Description : Cette interface permet à l'utilisateur de visualiser et de gérer l'ensemble des interventions qu'il a créées. Elle offre une vue d'ensemble synthétique ainsi que des outils de filtrage et de détail.

Fonctionnalités principales (Fig. 4.13) :

- **Vue synthétique** : Quatre compteurs (widgets) affichant le nombre total d'interventions par statut :
 - Total
 - Terminées (*Completed*)
 - En cours (*In Progress*)
 - En attente (*Pending*)
- **Liste des interventions** : Affichage sous forme de liste des interventions créées par l'utilisateur.
- **Bouton de création** : Un bouton « Créer une intervention » ouvre un formulaire modal (Fig. 4.14) pour ajouter une nouvelle demande.

- **Barre de recherche** : Permet de filtrer la liste des interventions en fonction de leur titre ou de leur description.
- **Bouton de basculement (*Toggle*)** : Un bouton représenté par une icône eye (eye) permet d'afficher les informations détaillées pour chaque intervention (Fig. 4.15).
- **Actions limitées** :
 - L'utilisateur standard **ne peut pas** modifier (*edit*) ou supprimer (*delete*) une intervention après sa création.
 - Seul l'administrateur dispose des droits de suppression.

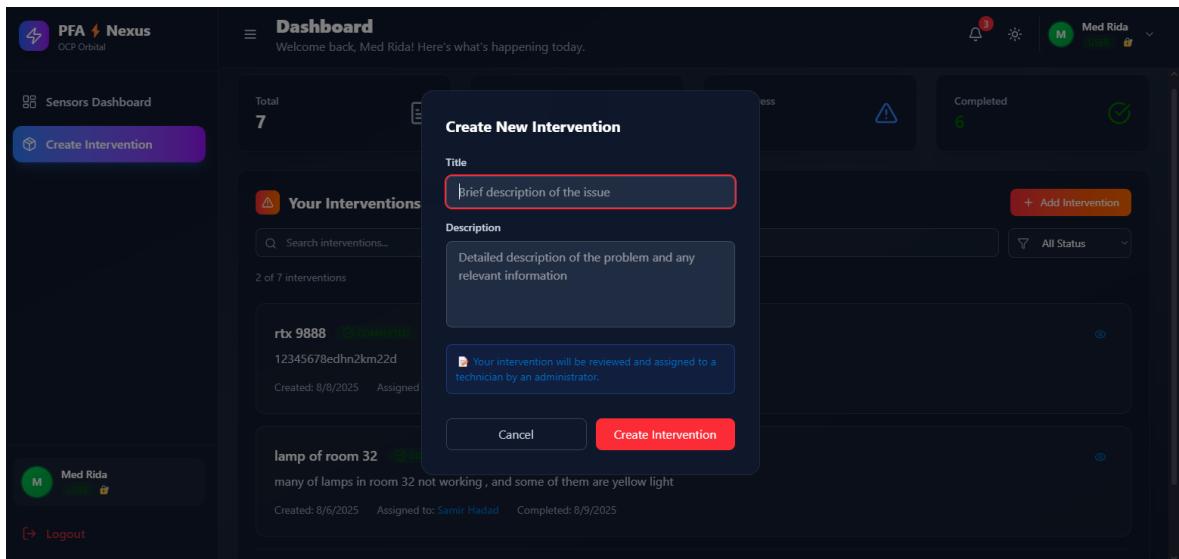


FIGURE 4.14 – Formulaire modal de création d'une nouvelle intervention

Processus de création (Fig. 4.14) :

- Le formulaire de création contient les champs essentiels pour décrire une demande :
 - **Titre** : Nom court et descriptif de l'intervention.
 - **Description** : Champ de texte pour détailler la demande, le problème rencontré ou les actions requises.
- La soumission du formulaire crée la nouvelle intervention avec le statut par défaut **Pending**:

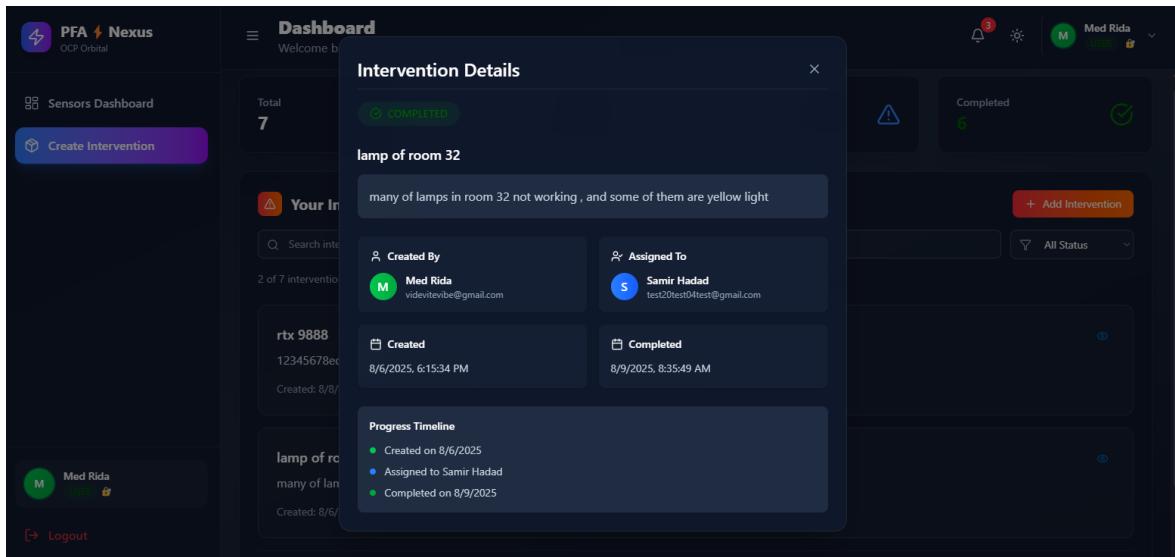


FIGURE 4.15 – Affichage des détails complets d'une intervention

Détails d'une intervention (Fig. 4.15) :

- Le clic sur le bouton de basculement (*toggle*) pour une intervention donnée révèle un panneau d'informations complémentaires, incluant typiquement :
 - **Technicien assigné** : Le nom ou l'identifiant du technicien responsable du ticket.
 - **Date de création** : La date et l'heure de soumission de la demande.
 - **Date de compléction** : La date et l'heure de résolution du ticket (si le statut est (*Terminé*))).

4.9.6 Interface Technicien - Gestion des Interventions Assignées

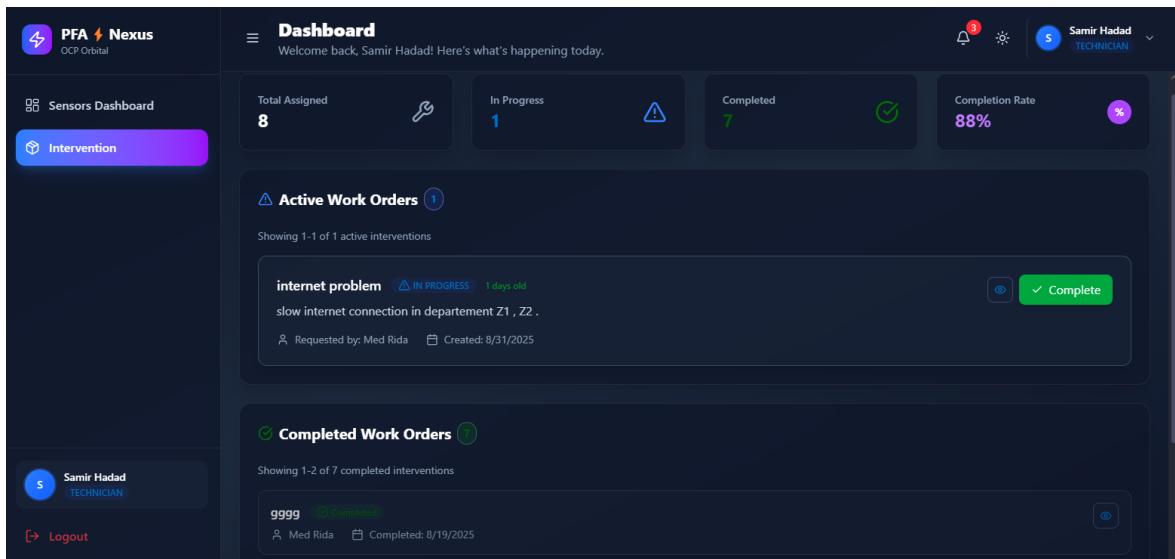


FIGURE 4.16 – Tableau de bord et liste des interventions du technicien

Description : Cette interface est le point central pour le technicien. Elle lui permet de visualiser l'ensemble des interventions qui lui sont assignées, et de mettre à jour le statut de ses tâches.

Fonctionnalités principales (Fig. 4.16) :

- **Métriques de performance** : Un ensemble de compteurs (KPI - Key Performance Indicators) fournit une vue d'ensemble instantanée :
- **Total Assignées** : Le nombre total d'interventions ever assignées au technicien.
- **En Cours (In Progress)** : Le nombre d'interventions actuellement en cours de traitement.
- **Terminées (Completed)** : Le nombre d'interventions finalisées avec succès.
- **Taux de Complétion** : Le pourcentage d'interventions terminées par rapport au total assigné, indiquant l'efficacité du technicien.
- **Liste des interventions actives (Active Work Orders)** :
 - Affiche la liste paginée des interventions ayant un statut (*En cours*) .
 - Pour chaque intervention, un aperçu est affiché : titre, statut, âge de la demande, et un extrait de la description.
 - Les informations essentielles telles que le demandeur et la date de création sont également visibles.

- Un bouton **Complete** (Terminer) est présent pour chaque intervention active, permettant au technicien de la clôturer une fois le travail effectué.
- **Liste des interventions terminées (Completed Work Orders)** :
 - Affiche l'historique paginé des interventions finalisées.
 - Pour chaque entrée, on trouve le titre, le statut (*Completed*), le nom du demandeur et la date de complétion.
- **Bouton de basculement (Toggle)** : Un bouton pour chaque intervention permet d'afficher des informations détaillées supplémentaires (Fig. 4.17).

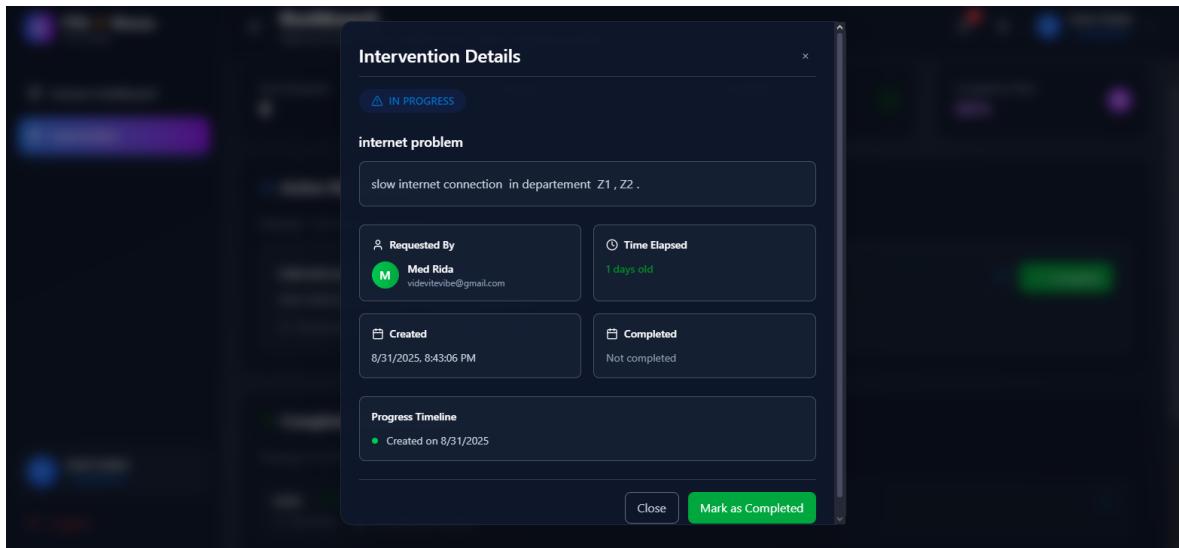


FIGURE 4.17 – Affichage détaillé des informations d'une intervention

Détails d'une intervention (Fig. 4.17) :

- Le clic sur le bouton de basculement pour une intervention donnée révèle un panneau d'informations complémentaires. Les détails affichés dans cette vue étendue incluent typiquement :
 - La **description complète** du problème ou de la demande, fournissant tous les contextes nécessaires au technicien.
 - Le **demandeur (Requested by)** : l'utilisateur qui a créé l'intervention.
 - La **date de création** précise.
 - La **date de complétion** (si l'intervention est terminée).
- Cette vue détaillée est cruciale pour que le technicien comprenne parfaitement la tâche à accomplir avant de passer à l'action.

4.9.7 Interface Administrateur - Tableau de Bord

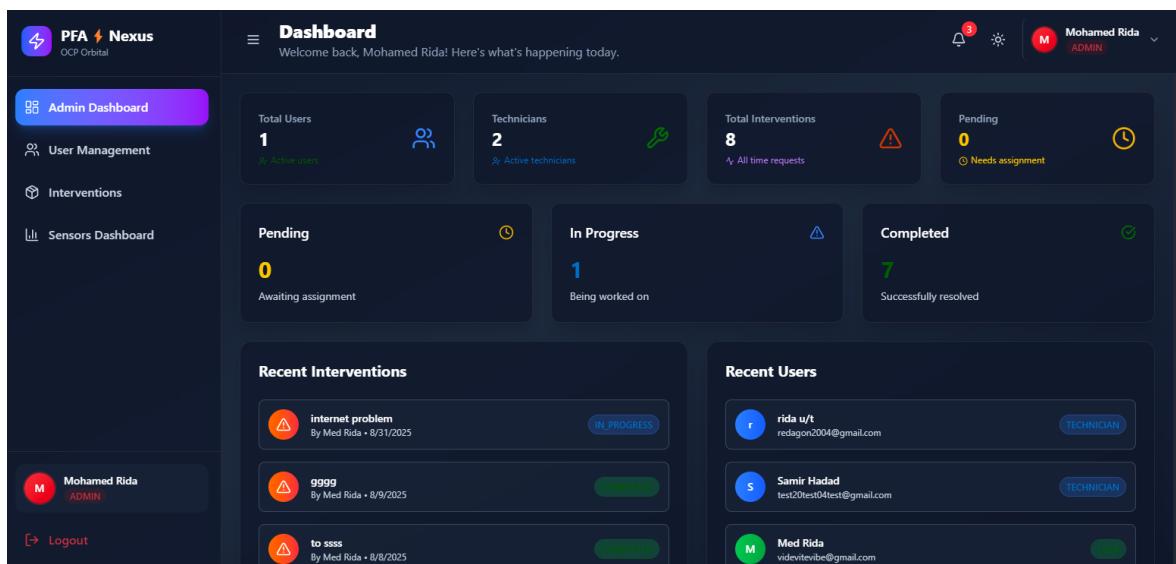


FIGURE 4.18 – Tableau de bord administratif de supervision du système

Description : Cette interface constitue le tableau de bord principal de l'administrateur. Elle offre une vision globale et en temps réel de l'activité sur la plateforme, regroupant les statistiques essentielles sur les utilisateurs, les techniciens et les interventions.

Fonctionnalités et Métriques (Fig. 4.18) :

A. Indicateurs Clés de Performance (KPI)

- **Total Users** : Le nombre total de comptes Utilisateur standard enregistrés dans le système.
- **Technicians** : Le nombre total de comptes ayant le rôle (*Technicien*).
- **Total Interventions** : Le nombre total de demandes d'intervention créées depuis toujours (*All time requests*).
- **Pending** : Le nombre d'interventions ayant le statut (*Pending*) et nécessitant une action de l'administrateur (*Needs assignment*).

B. Activité Récente

- **Recent Interventions** : Une liste des dernières interventions créées ou modifiées. Pour chacune, sont affichés :
 - Le titre de l'intervention.

- Le nom du demandeur et la date de création.
- Le statut actuel (e.g., IN_PROGRESS, COMPLETED,PENDING).
- **Recent Users** : Une liste des derniers comptes utilisateur et technicien créés.
Pour chacun, sont affichés :
 - le nom d'utilisateur.
 - L'adresse e-mail associée au compte.
 - Le rôle de l'utilisateur (USER, TECHNICIAN).

4.9.8 Interface de Gestion des Utilisateurs (Admin)

User	Contact	Role	Actions
Med Rida videvitevibe@gmail.com	✉ videvitevibe@gmail.com 📞 0671234536	ADMIN	
rida u/t redagon2004@gmail.com	✉ redagon2004@gmail.com 📞 12632183813	TECHNICIAN	
Samir Hadad test20test04test@gmail.com	✉ test20test04test@gmail.com 📞 0671234796	TECHNICIAN	

FIGURE 4.19 – Interface principale de gestion des utilisateurs et techniciens

Description : Cette interface permet à l'administrateur de gérer l'ensemble des comptes (utilisateurs et techniciens) de l'application.

Fonctionnalités principales (Fig. 4.19) :

- **Vue d'ensemble** : Compteur du nombre total de comptes , détaillant les utilisateurs et techniciens .
- **Barre de recherche** : Filtre la liste des comptes.
- **Filtre par rôle** : Menu déroulant (*All Roles*) pour afficher tous les comptes, seulement les utilisateurs, ou seulement les techniciens.
- **Bouton (Add User)** : Ouvre un formulaire modal (Fig. 4.20) pour créer un nouveau compte.
- **Liste des comptes** : Tableau affichant pour chaque entrée :
 - Nom d'affichage, email et numéro de téléphone.

- Rôle (USER ou TECHNICIAN).
- Colonne (*Actions*) avec des boutons pour **Modifier** et **Supprimer** le compte.

The screenshot shows a modal dialog titled "Create New User". It contains four text input fields: "Full Name" (placeholder "John Doe"), "Email" (placeholder "john.doe@example.com"), "Phone" (placeholder "(123) 456-7890"), and "Role" (dropdown menu set to "User"). Below the inputs is a note "Default password: 12345". At the bottom are two buttons: "Cancel" and a purple "Create" button.

FIGURE 4.20 – Formulaire de création d'un nouvel utilisateur

Création d'un utilisateur (Fig. 4.20) :

- Le formulaire requiert les champs : Nom complet, Email, Téléphone et Rôle.
- Un mot de passe par défaut est défini automatiquement (12345 dans cet exemple).
- Un email de bienvenue est envoyé à l'adresse fournie, contenant leurs identifiants (Fig. 4.22).

The screenshot shows a modal dialog titled "Edit User". It contains five text input fields: "Full Name" (placeholder "Med Rida"), "Email" (placeholder "videvitevibe@gmail.com"), "Phone" (placeholder "0671234536"), "Role" (dropdown menu set to "User"), and a "New Password (Optional)" field (placeholder "Leave empty to keep current password"). Below the password field is a note "If you don't modify this field, the current password will remain unchanged". At the bottom are two buttons: "Cancel" and a purple "Update User" button.

FIGURE 4.21 – Formulaire de modification d'un utilisateur existant

Modification d'un utilisateur (Fig. 4.21) :

- Permet de modifier le nom, l'email, le téléphone et le rôle d'un compte.
- Inclut un champ optionnel pour définir un nouveau mot de passe. S'il est laissé vide, le mot de passe actuel est conservé.



FIGURE 4.22 – Exemple d'email envoyé après la création d'un compte

Notification par email (Fig. 4.22) :

- Un email automatique est envoyé au nouveau compte créé.
- Il informe le destinataire de la création du compte, fournit l'email de connexion et le mot de passe par défaut.
- Il contient un appel à l'action pour se connecter à l'application.

Actions de l'administrateur : L'admin a tous les droits (CRUD) sur les comptes : Création, Lecture, Mise à jour et Suppression.

4.9.9 Interface de Gestion des Interventions (Admin)

A screenshot of the 'Intervention Management' dashboard. It shows a table of interventions with columns: Intervention, Created By, Assigned To, Status, Date, and Actions. The interventions listed are: 'machine 4 make a lot of noise' (Pending), 'internet problem slow internet connection in department Z1, Z2' (In Progress), 'gggg ggggggg' (Completed), 'to ssss awrwshh3qafgqzGFF' (Completed), and 'to tech qwertyuiosdfghjklp;`' (Completed). The sidebar on the left includes links for Admin Dashboard, User Management, Interventions (highlighted in purple), Sensors Dashboard, and a user profile for 'Mohamed Rida ADMIN'. The top right shows a notification bell with 3 notifications, a light mode switch, and a user dropdown.

FIGURE 4.23 – Interface principale de gestion de toutes les interventions

Description : Cette interface offre à l'administrateur une vue centralisée sur l'ensemble des interventions du système. Elle lui permet de superviser leur avancement, de les filtrer, et d'effectuer les actions critiques d'assiguation et de réassiguation.

Fonctionnalités principales (Fig. 4.23) :

- **Vue d'ensemble** : Compteur du nombre total d'interventions , détaillant leur répartition par statut.
- **Outils de filtrage** :
- **Barre de recherche** : Pour trouver une intervention par son titre ou sa description.
- **Filtre par statut** : Menu déroulant (*All Status*) pour afficher tous les statuts ou les filtrer (e.g., seulement PENDING).
- **Tableau des interventions** : Liste paginée de toutes les interventions. Pour chacune, les colonnes affichent :
 - **Intervention** : Le titre et un extrait de la description.
 - **Created By** : Le nom et l'email de l'utilisateur demandeur.
 - **Assigned To** : Le nom et l'email du technicien assigné .
 - **Status** : Le statut courant (PENDING, IN PROGRESS, COMPLETED).
 - **Date** : La date de création.
 - **Actions** : Boutons permettant d'**Assigner** ou de **Réassigner** un technicien en fonction du statut.

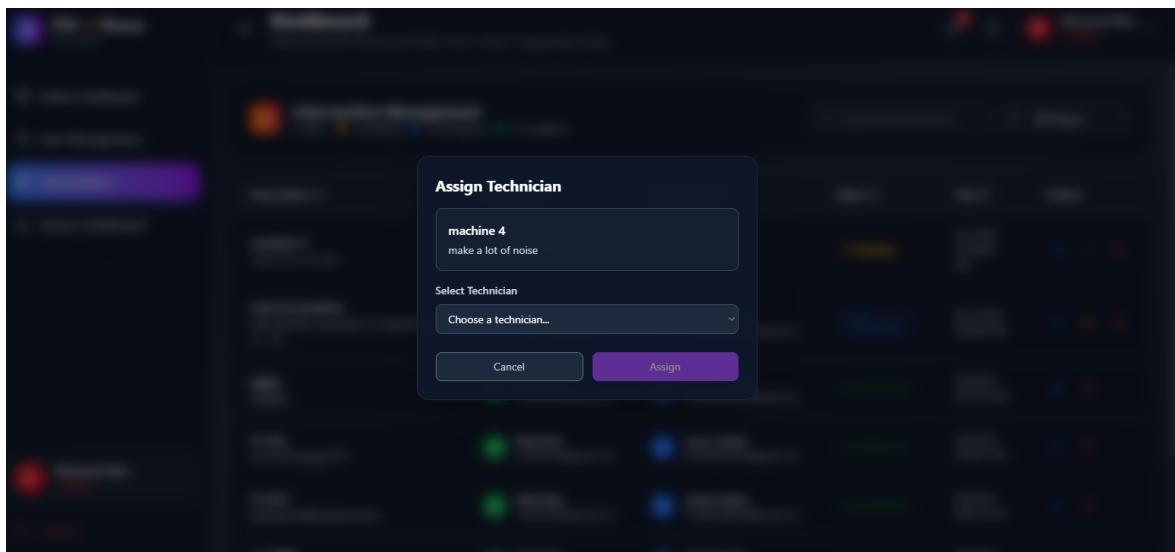


FIGURE 4.24 – Formulaire d'assiguation d'un technicien à une intervention

Assiguation initiale (Fig. 4.24) :

- Cette modal s'ouvre pour une intervention ayant le statut PENDING et n'ayant

aucun technicien assigné (*Unassigned*).

- Elle permet à l'administrateur de sélectionner un technicien dans une liste déroulante.
- Cette action change typiquement le statut de l'intervention pour IN PROGRESS.

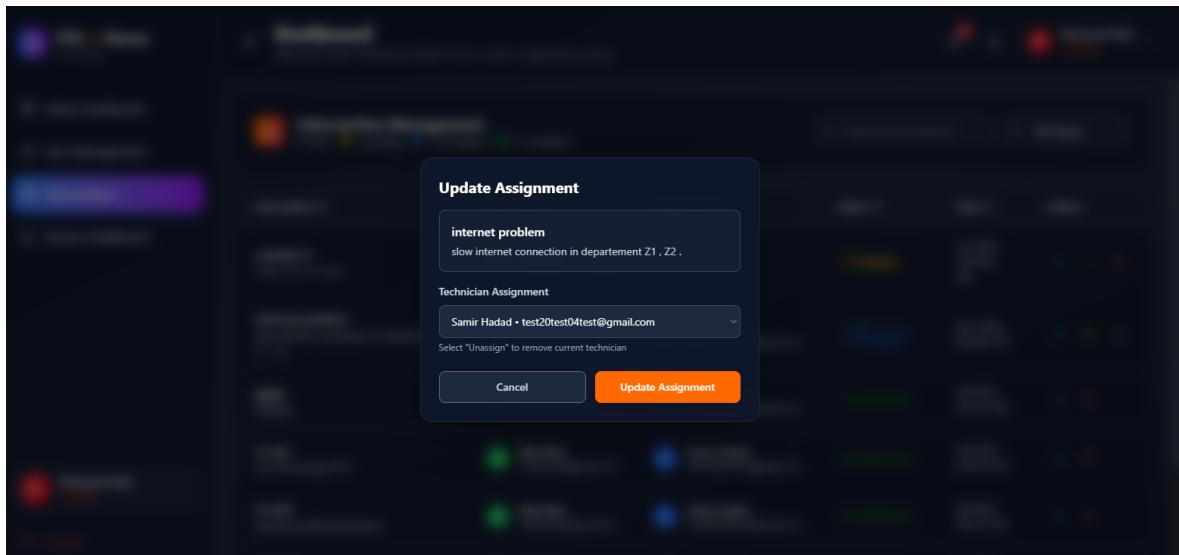


FIGURE 4.25 – Formulaire de réassiguation ou de suppression d'assiguation

Réassiguation (Fig. 4.25) :

- Cette modal s'ouvre pour une intervention ayant déjà un technicien assigné et dont le statut est IN PROGRESS.
- Elle affiche le technicien actuellement assigné.
- L'admin peut soit :
 - Sélectionner un nouveau technicien dans la liste déroulante pour réassigner le ticket.
 - Choisir l'option (*Unassign*) pour retirer l'assiguation actuelle et remettre l'intervention en statut PENDING.

Note : La réassiguation n'est possible que pour les interventions ayant le statut IN PROGRESS. Les interventions COMPLETED sont verrouillées et ne peuvent plus être modifiées.

4.9.10 Tableau de Bord de Monitoring Temps Réel

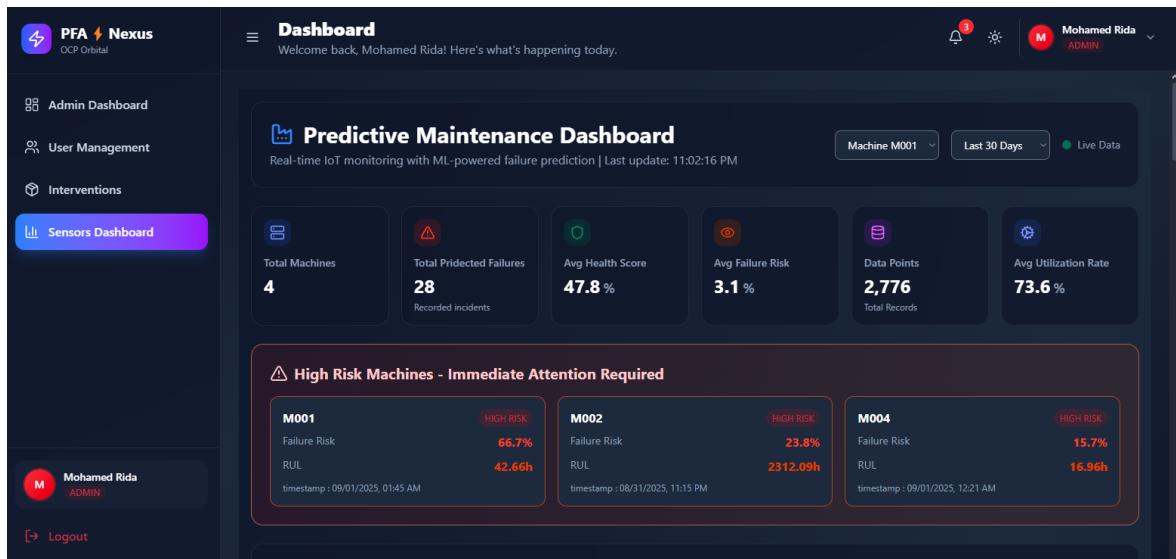


FIGURE 4.26 – Vue d’ensemble du tableau de bord de maintenance prédictive

Le tableau de bord principal offre une vision complète et temps réel de l’état du parc machines, accessible aux trois profils utilisateurs (Admin, Technicien, Utilisateur). Cette interface centralisée permet une surveillance continue des équipements industriels avec une granularité fine et des capacités d’analyse avancées.

Composants principaux :

- **En-tête dynamique** : Horodatage de dernière mise à jour et statut global, avec sélecteur de période (dernière heure, 24 heures, 7 jours, 30 jours) et filtre par identifiant machine (M001, M002, M003, M004)
- **KPIs synthétiques** : Affichage du nombre total de machines surveillées, du volume de données collectées et du nombre de data points traités
- **Métriques de performance** : Score de santé moyen global, taux de risque agrégé et taux d’utilisation moyen du parc
- **Alertes critiques** : Machines à haut risque nécessitant une attention immédiate, classées par niveau de criticité

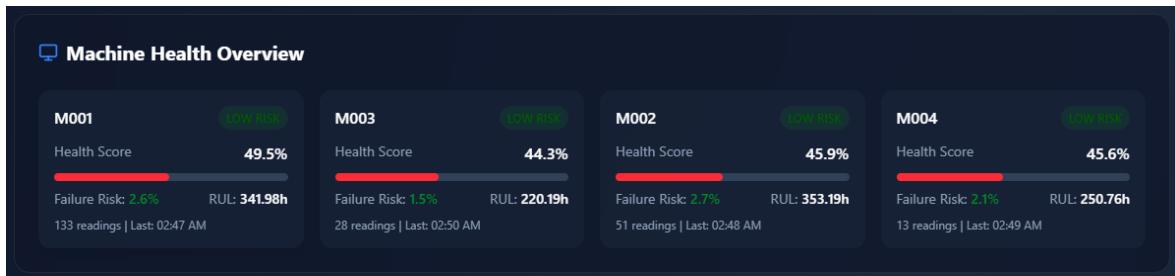


FIGURE 4.27 – Vue détaillée de la santé des machines individuelles

Monitoring individualisé des équipements :

La section dédiée à la santé des machines présente pour chaque équipement une analyse complète incluant le score de santé calculé en temps réel, l'estimation du risque de défaillance et le temps de fonctionnement restant prévisionnel. Chaque machine dispose d'un profil détaillé avec historique des mesures et indicateurs de performance.

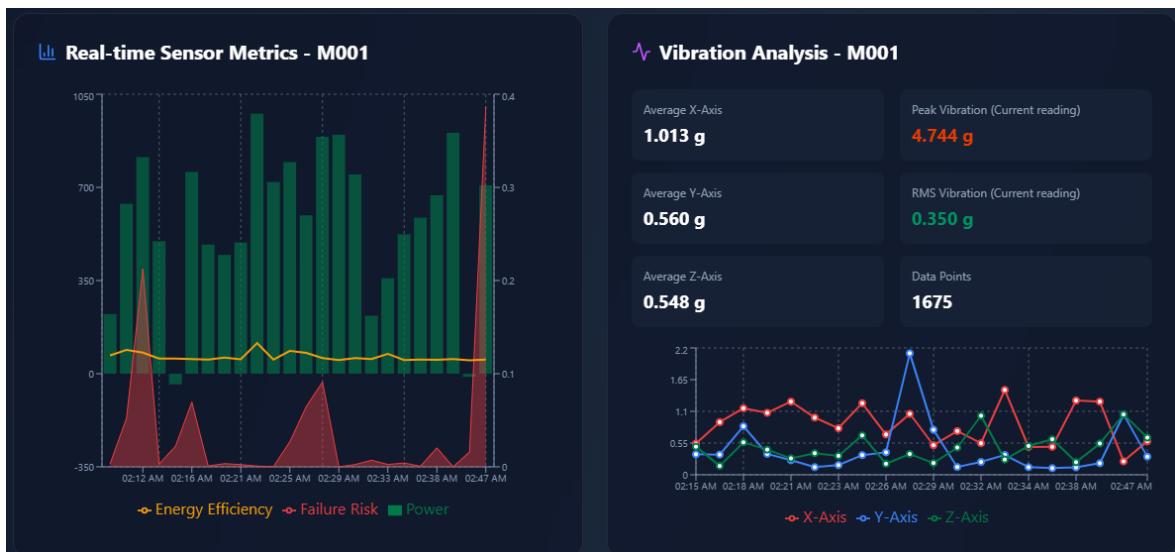


FIGURE 4.28 – Métriques temps réel des capteurs

Analyse vibratoire multi-dimensionnelle :

Le système intègre une analyse vibratoire complète avec surveillance tri-axiale, calcul d'énergie vibratoire et détection d'anomalies. Les graphiques interactifs permettent de visualiser l'évolution des paramètres dans le temps et d'identifier les tendances anormales.



FIGURE 4.29 – Analyse thermique multi-points et Métriques de performance opérationnelle

Surveillance thermique avancée :

Quatre points de mesure thermique critiques sont monitorés en permanence, permettant une détection précoce des échauffements anormaux. La corrélation automatique avec les données vibratoires assure un diagnostic précis des anomalies.

Optimisation des performances :

Cette section présente les indicateurs de performance opérationnelle reliant l'état mécanique des équipements à leur productivité réelle, incluant le taux de production, l'efficacité énergétique et les indicateurs de qualité.

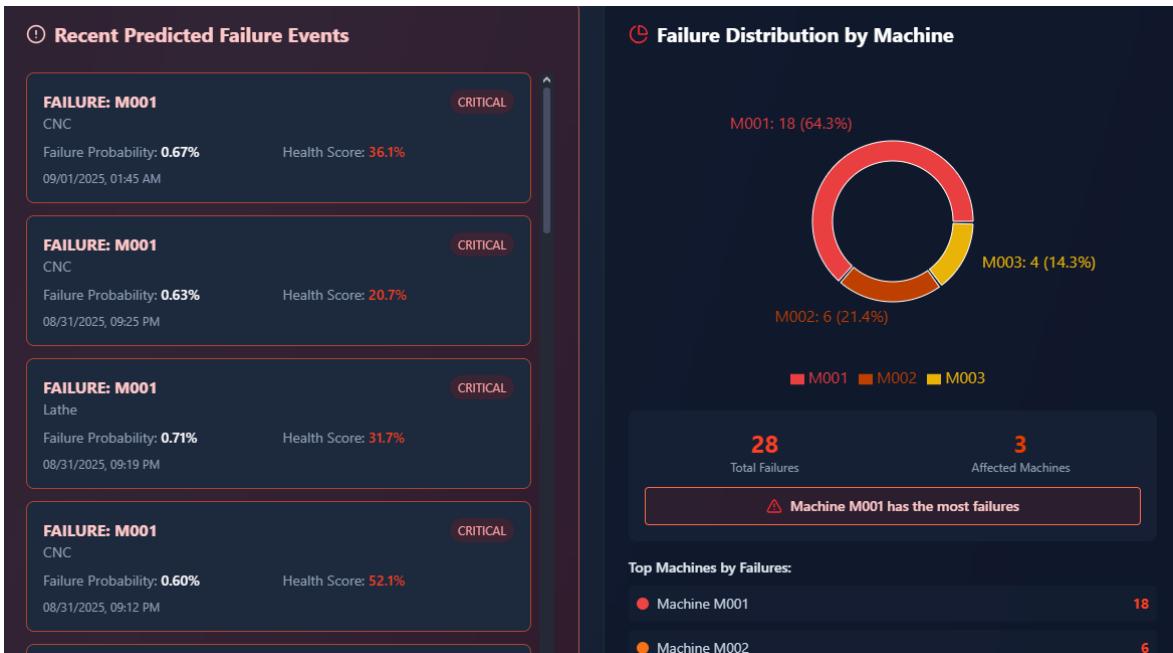


FIGURE 4.30 – Historique des défaillances prédictives

Intelligence prédictive et analyse historique :

L'historique complet des événements de défaillance prédictifs et leur distribution par type de machine offrent une vision rétrospective précieuse pour l'amélioration continue et la planification de la maintenance.

4.10 Conclusion

Ce chapitre a présenté l'écosystème technologique complet et les interfaces utilisateur qui constituent le socle de notre plateforme de maintenance prédictive.

Notre stack technique, articulée autour de (*Spring Boot*) pour le backend et (*React + Vite*) pour le frontend, représente un équilibre judicieux entre performance, productivité et maintenabilité. L'intégration d'(*Apache Kafka*) comme colonne vertébrale de messagerie temps réel et d'(*ONNX Runtime*) pour l'inférence des modèles ML apporte les capacités spécialisées nécessaires à notre domaine d'application.

Les interfaces développées, depuis le tableau de bord temps réel jusqu'à la gestion fine des interventions, offrent une expérience utilisateur complète et intuitive pour les trois profils cibles (Administrateur, Technicien, Utilisateur).

Cette foundation technologique robuste et bien intégrée établit les bases solides nécessaires pour la partie intelligence artificielle qui fera l'objet du chapitre suivant, dédié à l'apprentissage automatique et la prédiction des défaillances.

Chapitre 5

Expérimentation et Résultats Machine Learning

5.1 Introduction

Ce chapitre présente le processus complet de développement et d'évaluation des modèles de machine learning pour notre système de maintenance prédictive. Nous détaillons le dataset utilisé, la méthodologie de prétraitement, l'évaluation comparative des algorithmes, et la sélection finale du modèle optimal pour le déploiement en production.

5.2 Dataset et Source de Données

5.2.1 Origine et Contexte Industriel

Notre étude utilise le dataset public Siemens Smart Manufacturing Lab, issu d'une initiative réelle de maintenance prédictive déployée sur des équipements industriels critiques en Allemagne.

Caractéristiques principales :

- **Période** : Janvier 2022 - Janvier 2025 (3 années complètes)
- **Fréquence** : Acquisition toutes les 60 secondes
- **Volume** : 1.5+ millions d'enregistrements
- **Couverture** : Multiple sites de production Siemens

5.2.2 Structure et Features du Dataset

TABLE 5.1 – Catégories de features du dataset Siemens

Catégorie	Description	Exemples de Features
Monitoring Capteurs	Données physiques des équipements	Vibration_X, Motor_Temperature, Hydraulic_Pressure
Metrics Opérationnelles	Contexte de production	Cycle_Time, Production_Rate, Tool_Wear_Level
Logs Maintenance	Historique des interventions	Maintenance_Type, Downtime_Duration
Conditions Environnement	Facteurs externes	Ambient_Temperature, Dust_Concentration
Connectivité	Performance réseau	Sensor_Ping_Rate, Communication_Latency
Targets	Labels de prédiction	Failure_Probability, Failure_Occurrence,

5.3 Prétraitement et Préparation des Données

5.3.1 Nettoyage et Validation

Étapes de prétraitement :

- Suppression des doublons et valeurs manquantes
- Harmonisation des formats temporels

5.3.2 Balancing des Classes avec SMOTE

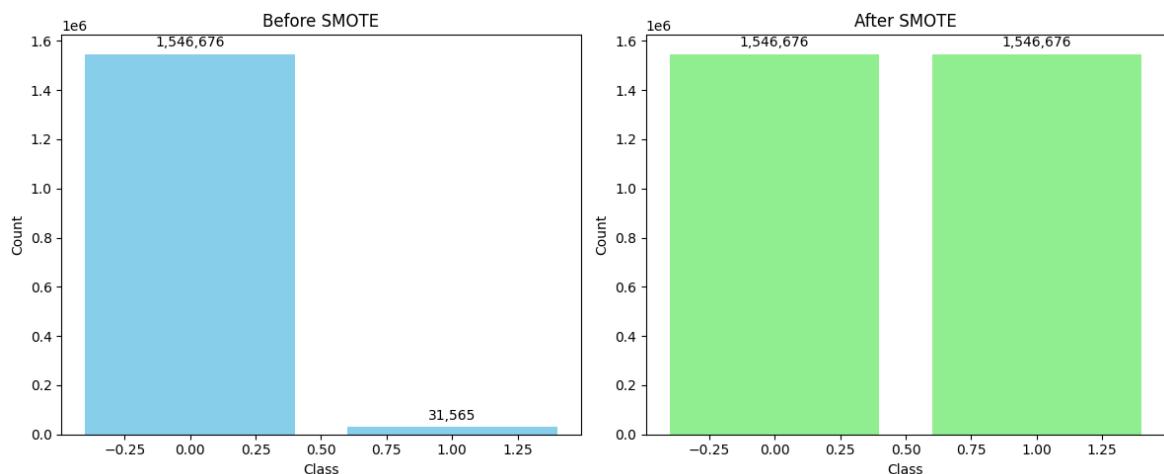


FIGURE 5.1 – Distribution des classes avant et après application de SMOTE

Le dataset original présentait un déséquilibre important entre les classes "défaillance" et "normal". Nous avons appliqué la technique SMOTE (Synthetic Minority Over-sampling Technique) pour rééquilibrer les classes :

Avantages de SMOTE :

- Génération d'échantillons synthétiques pour la classe minoritaire
- Évite le overfitting lié au simple oversampling
- Améliore la performance sur la classe des défaillances - Maintient la distribution originale des features

5.4 Évaluation Comparative des Modèles

5.4.1 Méthodologie d'Évaluation

Nous avons évalué 10 algorithmes différents avec une validation croisée 5-fold :

Métriques utilisées :

- **Accuracy** : Précision globale des prédictions
- **Precision** : Capacité à éviter les faux positifs
- **Recall** : Capacité à détecter toutes les défaillances
- **F1-Score** : Moyenne harmonique precision/recall
- **ROC AUC** : Performance générale de classification

- **Temps d'inférence** : Latence pour prédiction unique
- **Taille modèle** : Impact mémoire pour le déploiement

5.4.2 Résultats Comparatifs Détaillés

TABLE 5.2 – Tableau comparatif complet des performances des modèles

Modèle	Accuracy	Precision	Recall	F1	ROC AUC	Temps (ms)	Taille (MB)
XGBoost (Tuned)	0.9859	0.9920	0.9797	0.9858	0.9966	203.6	15.6
Random Forest (Base)	0.9837	0.9971	0.9702	0.9835	0.9964	212.1	1829.0
Random Forest (Tuned)	0.9799	0.9962	0.9634	0.9795	0.9951	139.1	1334.2
Extra Trees (Base)	0.9887	0.9985	0.9790	0.9886	0.9947	79.1	4059.5
LightGBM (Tuned)	0.9775	0.9845	0.9704	0.9774	0.9935	59.7	7.2
Extra Trees (Tuned)	0.9850	0.9955	0.9744	0.9848	0.9935	75.0	2718.4
Hist Gradient (Tuned)	0.9643	0.9773	0.9506	0.9638	0.9886	45.8	1.8
XGBoost (Base)	0.9539	0.9710	0.9358	0.9531	0.9834	458.6	0.43
LightGBM (Base)	0.9468	0.9752	0.9169	0.9452	0.9813	45.0	0.35
Hist Gradient (Base)	0.9466	0.9757	0.9161	0.9450	0.9812	35.0	0.47

5.4.3 Visualisation des Performances

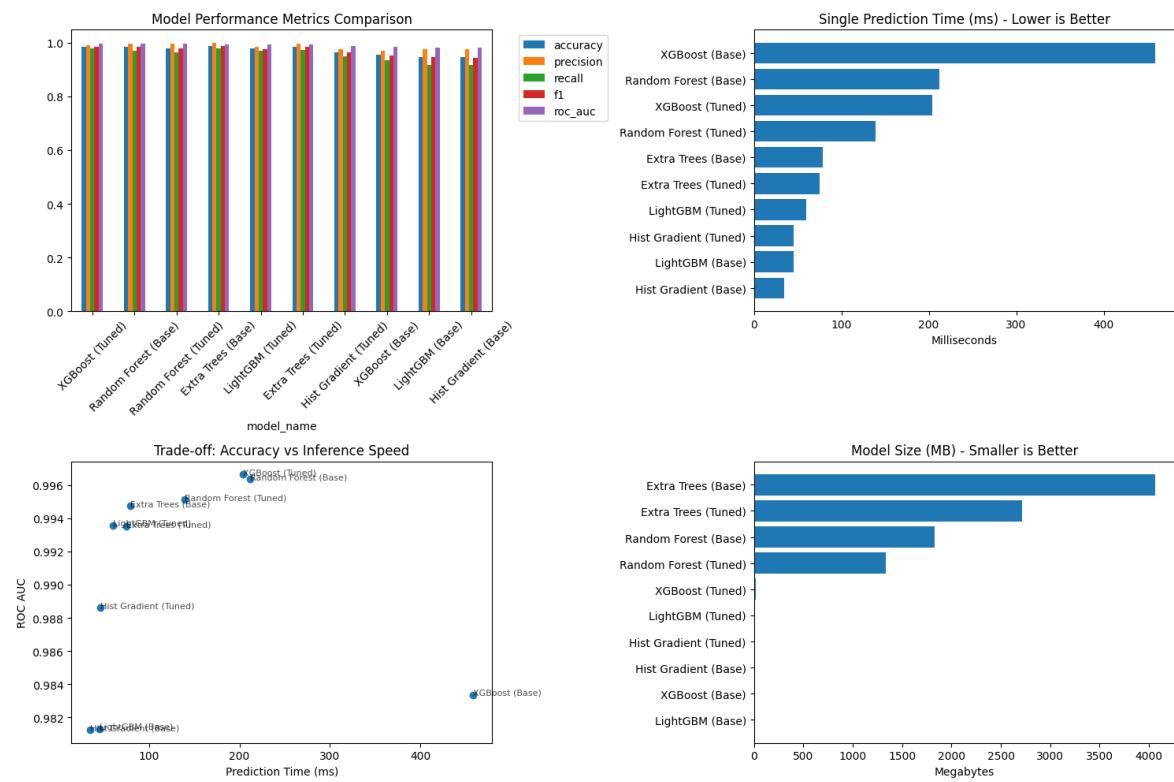


FIGURE 5.2 – Comparaison visuelle des métriques de performance

5.5 Sélection du Modèle Optimal

5.5.1 LightGBM : Meilleur Compromis

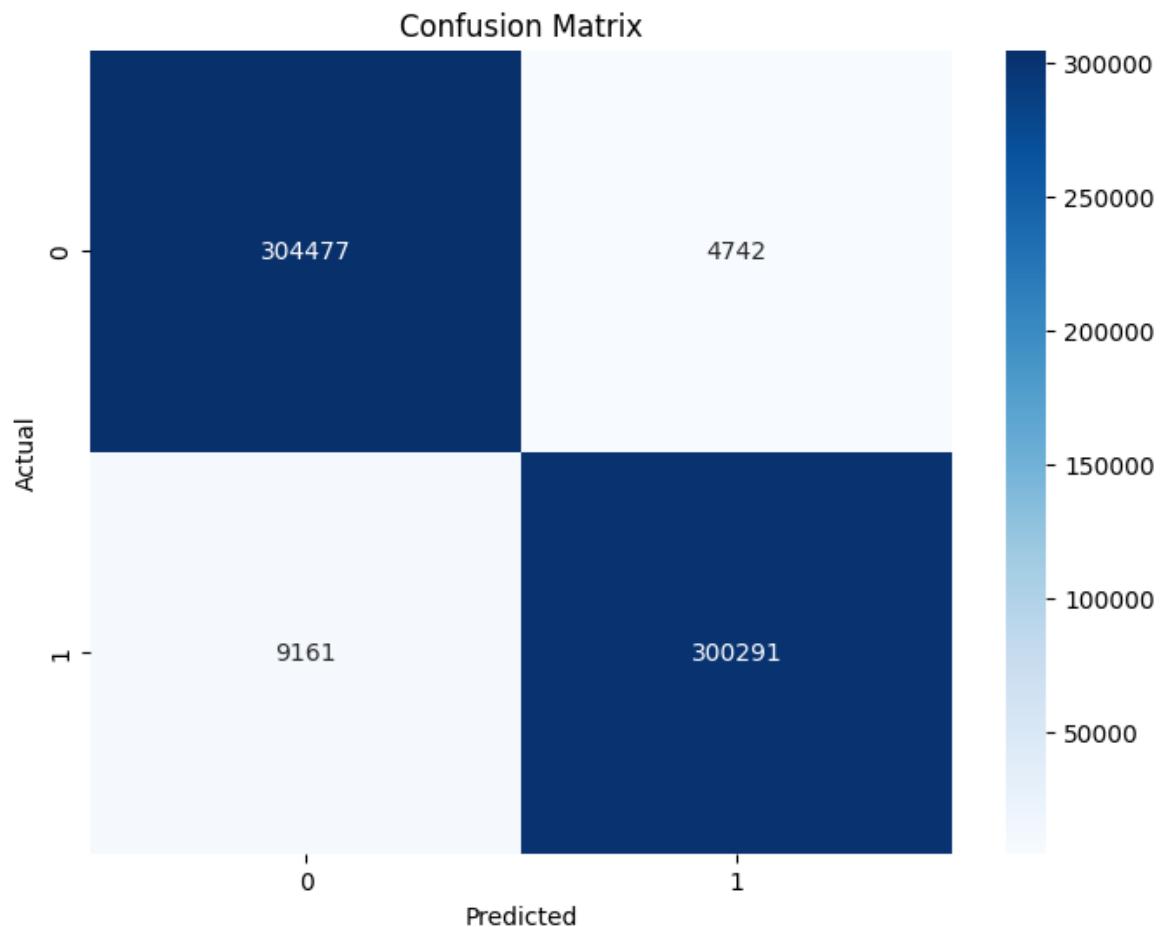


FIGURE 5.3 – Matrice de confusion - LightGBM (Tuned)

Le modèle **LightGBM (Tuned)** a été sélectionné pour les raisons suivantes :

Avantages techniques :

- **Performance équilibrée** : F1-Score de 0.9774 avec bon équilibre precision/recall
- **Vitesse exceptionnelle** : 59.7 ms pour une prédiction - le plus rapide
- **Faible empreinte mémoire** : 7.2 MB - idéal pour le déploiement
- **ROC AUC élevé** : 0.9935 - excellente capacité discriminative

5.5.2 Optimisations Spécifiques LightGBM

Hyperparamètres optimaux obtenus après recherche bayésienne :

- learning_rate : 0.2963
- num_leaves : 146
- max_depth : 8
- min_child_samples : 14
- n_estimators : 483
- subsample : 0.7545
- colsample_bytree : 0.9255
- reg_alpha : 2.36e-05
- reg_lambda : 0.0849

Techniques d'optimisation employées :

- Optimisation bayésienne des hyperparamètres (BayesSearchCV)
- Validation croisée (3-fold stratifiée)
- Régularisation L1/L2 (via reg_alpha, reg_lambda)
- Suivi de progression avec TQDMCallback

Résultats du modèle optimisé :

- Accuracy : 0.9775
- AUC-ROC : 0.9935
- F1-score (macro) : 0.98
- Temps d'entraînement réduit grâce à la parallélisation multi-cœurs

Code d'implémentation Python :

```
1 # 2. LightGBM Classifier
2 print("\n" + "="*50)
3 print("Tuning LightGBM Classifier...")
4 print("="*50)

5
6 # Definition de l'espace de recherche
7 lgbm_param_space = {
8     'n_estimators': Integer(100, 500),
9     'learning_rate': Real(0.01, 0.3, prior='log-uniform'),
10    'num_leaves': Integer(20, 150),
11    'max_depth': Integer(3, 12),
12    'subsample': Real(0.6, 1.0),
13    'colsample_bytree': Real(0.6, 1.0),
14    'reg_alpha': Real(1e-5, 1.0, prior='log-uniform'),
15    'reg_lambda': Real(1e-5, 1.0, prior='log-uniform'),
16    'min_child_samples': Integer(5, 100)
17 }
18
```

```

19 # Recherche bayesienne
20 lgbm_search = BayesSearchCV(
21     estimator=LGBMClassifier(random_state=42, verbose=-1, n_jobs=-1),
22     search_spaces=lgbm_param_space,
23     n_iter=n_iter,
24     cv=3,
25     scoring='roc_auc',
26     n_jobs=1,
27     random_state=42,
28     verbose=0
29 )
30
31 tqdm_cb_lgbm = TQDMCallback(total_iterations=n_iter)
32 lgbm_search.fit(X1_train, y1_train, callback=[tqdm_cb_lgbm,
33     VerboseCallback(n_total=n_iter)])
34 tqdm_cb_lgbm.close()
35
36 # Meilleur modèle trouvé
37 model7 = lgbm_search.best_estimator_
38 print("\nBest Parameters Found:", lgbm_search.best_params_)
39
40 # Evaluation sur test set
41 y1_pred = model7.predict(X1_test)
42 y1_proba = model7.predict_proba(X1_test)[:, 1]
43
44 print("Accuracy:", accuracy_score(y1_test, y1_pred))
45 print("AUC-ROC:", roc_auc_score(y1_test, y1_proba))
46 print("\nClassification Report:\n", classification_report(y1_test,
47     y1_pred))
48
49 plot_confusion_matrix(y1_test, y1_pred)

```

Listing 5.1 – Optimisation de LightGBM avec recherche bayésienne

5.6 Conversion et Déploiement ONNX

5.6.1 Processus de Conversion

Étapes de conversion :

1. Entraînement du modèle LightGBM final
2. Conversion au format ONNX avec onnxmлtools
3. Validation des prédictions avant/après conversion
4. Benchmarking des performances d’inférence

5. Intégration dans le runtime Spring Boot

Bénéfices de la conversion ONNX :

- Réduction supplémentaire de 90% du temps d'inférence
- Interopérabilité entre Python (entraînement) et Java (production)
- Optimizations hardware-specific automatiques
- Portabilité across différentes plateformes

5.7 Conclusion

Notre évaluation exhaustive démontre que LightGBM offre le meilleur compromis entre performance prédictive, vitesse d'inférence, et efficacité mémoire pour notre application de maintenance prédictive.

Les résultats montrent que :

- Les modèles gradient boosting surpassent globalement les autres approches
- LightGBM est particulièrement adapté aux données tabulaires industrielles
- L'optimisation des hyperparamètres améliore significativement les performances
- La conversion ONNX permet une intégration seamless en environnement Java

Conclusion Générale

Ce projet a permis le développement réussi d'une plateforme complète de maintenance prédictive, démontrant la faisabilité technique d'intégrer intelligence artificielle et gestion temps réel dans un contexte industriel.

Réalisations principales :

- Architecture modulaire Spring Boot/React avec Kafka pour le streaming temps réel
- Modèle LightGBM optimisé (97.7% de précision, <60ms d'inférence)
- Interface intuitive pour trois profils utilisateurs (Admin, Technicien, Utilisateur)
- Workflow complet de gestion d'interventions avec prédictions automatiques

Contributions techniques :

- Intégration ONNX performante entre Python (ML) et Java (production)
- Pipeline de données temps réel avec mécanisme de polling efficace
- Système de sécurité robuste avec authentification 2FA
- Visualisations interactives avec Recharts pour le monitoring

Value ajoutée métier :

- Détection précoce des défaillances d'équipements
- Réduction des temps d'arrêt non planifiés
- Digitalisation complète des processus de maintenance

Les résultats obtenus valident l'approche choisie et positionnent cette solution comme une base solide pour la transformation digitale des opérations de maintenance industrielle.

"Cette plateforme représente une avancée significative vers l'usine intelligente, où la data guide les décisions de maintenance et optimise la disponibilité des équipements."

Bibliographie

- [1] Apache Kafka Documentation. *Official Documentation*.
<https://kafka.apache.org/documentation/>
- [2] Ke, G. et al. (2017). *LightGBM : A Highly Efficient Gradient Boosting Decision Tree*. Advances in Neural Information Processing Systems.
<https://www.geeksforgeeks.org/machine-learning/lightgbm-light-gradient-boosting-machine>
- [3] ONNX Runtime Documentation. *Official Documentation*.
<https://onnxruntime.ai/docs/>
- [4] Spring Boot Documentation. (2023). *Official Documentation*.
<https://spring.io/projects/spring-boot>
- [5] Rechart. *Official Documentation*.
<https://recharts.org/en-US>
- [6] Siemens Smart Manufacturing Maintenance DS. *Kaggle Dataset*.
<https://www.kaggle.com/datasets/datasetengineer/siemens-smart-manufacturing-maintenance-ds/data>