CHAPITRE I – PRÉSENTATION GÉNÉRALE DU PROJET

I. Présentation du projet

1 Présentation générale du projet

L'Institut Universitaire Saint Jean du Cameroun (IUSJC), établissement d'enseignement supérieur disposant de trois sites (Eyang, Douala et Etokoss), fait face à un défi opérationnel majeur dans la gestion et la maintenance de ses infrastructures, particulièrement sur son site principal d'Eyang. Ce campus comprend deux bâtiments dédiés aux activités pédagogiques et administratives, un bâtiment pour la cité universitaire, et un bâtiment réservé aux pères. L'ensemble de ces infrastructures regroupe des espaces variés (chambres, salles de classe, bureaux) équipés de nombreux dispositifs (climatiseurs, réfrigérateurs, tables, chaises, téléviseurs, sanitaires, prises électriques, etc.) nécessitant un suivi rigoureux et une maintenance régulière.

Actuellement, l'équipe en charge de la gouvernance des bâtiments rencontre des difficultés considérables dans l'accomplissement de ses missions. Le système de signalement des incidents repose sur des méthodes informelles et peu structurées, conduisant à des descriptions imprécises des problèmes, des délais de traitement allongés, une traçabilité inexistante des interventions, et une impossibilité d'anticiper les besoins de maintenance. Cette situation génère non seulement une insatisfaction des occupants (étudiants de la cité universitaire, résidents du bâtiment des pères), mais également une inefficacité opérationnelle pour l'équipe de maintenance composée d'un responsable et d'agents de terrain en constante mobilité.

Le projet que nous proposons vise à concevoir et développer une **plateforme numérique intégrée de gestion de la maintenance des infrastructures de l'IUSJC**. Cette solution se composera de trois interfaces complémentaires : une **application mobile pour les occupants** (permettant de signaler facilement les incidents), une **application mobile pour les agents de terrain et le superviseur** (facilitant la prise en charge et le suivi des interventions, y compris en mode hors ligne), et une **application web pour l'administrateur** (centralisant la configuration du système, la cartographie des espaces, et la visualisation de données analytiques avancées).

L'objectif principal est de **digitaliser et optimiser l'ensemble du processus de maintenance**, depuis la détection d'un dysfonctionnement jusqu'à sa résolution, tout en fournissant des outils d'analyse et de prédiction pour une gestion proactive des équipements. Le système permettra également de générer des rapports détaillés, de suivre des indicateurs de performance clés (KPI), et d'identifier les équipements sujets à des défaillances fréquentes grâce à des mécanismes prédictifs.

2 Attentes et portée du projet

Les attentes vis-à-vis de ce projet sont multiples et concernent différents niveaux d'impact :

Sur le plan fonctionnel, le système devra offrir un processus de signalement d'incidents simplifié et intuitif pour les occupants, une gestion efficace des interventions pour les agents de terrain avec possibilité de travail hors connexion, des tableaux de bord analytiques complets pour le superviseur et l'administrateur, une cartographie interactive des bâtiments avec géolocalisation indoor, un système de gestion des équipements avec suivi de leurs états, un mécanisme de notifications push en temps réel, et des capacités d'analyse prédictive pour anticiper les pannes.

Les livrables attendus comprennent trois applications interconnectées (mobile occupant, mobile agent/superviseur, web administrateur), une documentation technique exhaustive (cahier des charges, cahier d'analyse, cahier de conception, documentation API, guide de déploiement), les fichiers APK pour les applications mobiles, une application web déployée et accessible via URL, un système de base de données robuste et sécurisé, des rapports d'analyse et de tests, et un dépôt Git structuré avec historique de versionnage.

L'impact attendu se manifeste à plusieurs niveaux : amélioration significative du temps de réponse aux incidents, réduction des coûts de maintenance grâce à une approche prédictive, augmentation de la satisfaction des occupants, optimisation de la productivité des équipes de maintenance, traçabilité complète des interventions, prise de décision éclairée basée sur des données analytiques, et meilleure allocation des ressources humaines et matérielles.

La portée du projet se limitera au site d'Eyang dans un premier temps, tout en concevant une architecture évolutive permettant une extension future aux sites de Douala et Etokoss. Le système couvrira tous les types d'espaces (chambres de la cité universitaire, bâtiment des pères, salles de classe, bureaux administratifs) et tous les types d'équipements présents dans ces espaces.

3 Enjeux et bénéfices attendus

Enjeux académiques : Ce projet s'inscrit dans le cadre de l'unité d'enseignement IS5179 (Projet Transversal ISI, Techdays) du niveau 5 de la filière Informatique et Système d'Information. Il représente une opportunité concrète de mettre en pratique l'ensemble des compétences acquises durant la formation : analyse et conception de systèmes d'information, développement d'applications mobiles et web, gestion de bases de données, architecture logicielle, travail collaboratif en équipe, gestion de projet agile, et déploiement d'applications. Le projet permet également de développer des compétences transversales essentielles telles que la résolution de problèmes complexes, la communication technique, et la gestion du temps.

Enjeux technologiques : Le projet nécessitera la maîtrise et l'intégration de technologies modernes et variées : frameworks de développement mobile cross-platform (Flutter, React

Native), technologies web front-end et back-end, systèmes de gestion de bases de données (relationnelles et/ou NoSQL), mise en œuvre de services RESTful API, intégration de systèmes de géolocalisation indoor, implémentation de mécanismes de synchronisation online/offline, utilisation de services de notifications push, mise en place de pipelines CI/CD, et application de techniques d'analyse prédictive et de machine learning. La conception d'une architecture scalable et maintenable constitue également un défi technique majeur.

Enjeux organisationnels : Pour l'IUSJC, ce projet représente un investissement stratégique dans la transformation numérique de ses processus opérationnels. Il s'agit d'un changement organisationnel qui impactera les pratiques de travail des équipes de maintenance et les habitudes des occupants. La réussite du projet nécessitera une conduite du changement appropriée, incluant la formation des utilisateurs et l'accompagnement à l'adoption de la solution.

Bénéfices attendus pour les occupants : Simplification du processus de signalement des incidents (via une simple photo et un commentaire), suivi transparent de l'état de leurs requêtes, réduction du temps de résolution des problèmes, amélioration de la qualité de vie sur le campus, et sentiment d'être écoutés et pris en charge efficacement.

Bénéfices attendus pour les agents de terrain : Réception instantanée des alertes d'incidents via notifications push, priorisation intelligente des interventions, accès aux informations détaillées sur chaque incident (photo, commentaire, localisation précise), capacité de travailler même sans connexion internet, réduction des déplacements inutiles grâce à une meilleure information, et facilitation du reporting via la génération automatique de rapports Excel.

Bénéfices attendus pour le superviseur et l'administration : Vision globale en temps réel de l'état des infrastructures, identification rapide des zones problématiques, prise de décision basée sur des données analytiques fiables, optimisation de l'allocation des ressources humaines et financières, réduction des coûts de maintenance grâce à l'approche prédictive, amélioration de la planification budgétaire, et traçabilité complète pour la responsabilité et l'audit.

Bénéfices sociétaux et environnementaux : La digitalisation du processus de maintenance contribue à une gestion plus responsable des ressources en prolongeant la durée de vie des équipements grâce à une maintenance proactive, en réduisant le gaspillage lié aux remplacements prématurés, et en optimisant les déplacements des équipes de maintenance (réduction de l'empreinte carbone). De plus, l'amélioration des conditions de vie sur le campus participe au bien-être des étudiants et du personnel, favorisant ainsi un environnement propice à l'apprentissage et au travail.

II. Contexte et environnement du projet

1 Contexte académique

Ce projet s'inscrit dans le cadre de l'unité d'enseignement **IS5179 - Projet Transversal ISI, Techdays**, dispensée au niveau 5 (cinquième année) de la filière **Informatique et Système d'Information** à l'Institut Universitaire Saint Jean. Cette UE, dotée de 4 crédits ECTS et enseignée en français et anglais, est placée sous la responsabilité de **M. Arthur PESSA** (arthur.pessa@saintjeaningenieur.org) et **M. Emmanuel MOUPOJOU** (emmanuel.moupojou@saintjeaningenieur.org).

L'objectif pédagogique de cette unité d'enseignement est de permettre aux étudiants de consolider et de mettre en pratique l'ensemble des compétences techniques et méthodologiques acquises durant leur cursus, en les confrontant à un projet réel de grande envergure. Le projet se déroule sur l'ensemble du premier semestre de l'année académique 2025-2026, avec une présentation finale prévue pour le **2, 3 ou 21 février 2026**.

Le travail collaboratif en équipe constitue un aspect fondamental de ce projet. Les étudiants sont organisés en groupes de travail et doivent mettre en œuvre des pratiques professionnelles de développement logiciel : utilisation d'un système de gestion de versions (Git via GitLab ou GitHub), respect de conventions de nommage (répertoire Git nommé ISI5_PROJET_GROUPE), production documentaire structurée, et application de méthodologies agiles.

Les livrables documentaires attendus comprennent un cahier des charges (définissant précisément les besoins et exigences), un cahier d'analyse (détaillant l'étude du système existant et les solutions proposées), un cahier de conception (présentant l'architecture et la modélisation UML du système), une documentation API (décrivant les endpoints et les contrats d'interface), un document de déploiement (expliquant les procédures de mise en production), et les fichiers APK des applications mobiles.

L'évaluation du projet repose sur quatre critères principaux : la présentation orale (5 points), l'interface utilisateur/expérience utilisateur et l'architecture du système (7 points), la qualité et la ponctualité des livrables (3 points), et la qualité du code source ainsi que le déploiement (5 points). Cette pondération reflète l'importance accordée à la fois aux aspects techniques et à la capacité de communication des étudiants.

Des contraintes temporelles strictes sont imposées, notamment la création obligatoire du répertoire Git avant le **25 octobre 2025**, sous peine de pénalités. Le dépôt de tous les documents dans le drive partagé (https://drive.google.com/drive/folders/1wvrwJoaATgE8-7LgOszkOUcy98p809Dr) est également requis avant la présentation finale.

Ce cadre académique rigoureux vise à préparer les étudiants aux exigences du monde professionnel, où la gestion de projet, le respect des délais, la qualité du code, et la capacité à travailler en équipe sont des compétences essentielles.

2 Contexte technologique et sociétal

Contexte de la transformation numérique dans l'enseignement supérieur : Le secteur de l'enseignement supérieur en Afrique subsaharienne, et particulièrement au Cameroun, connaît une accélération de sa transformation numérique. Cette dynamique, amplifiée par la crise sanitaire de la COVID-19, a mis en évidence la nécessité pour les établissements d'enseignement de digitaliser non seulement leurs activités pédagogiques, mais également l'ensemble de leurs processus opérationnels. La gestion des infrastructures et la maintenance des équipements constituent des domaines où la digitalisation peut apporter des gains d'efficacité considérables.

Tendances technologiques mobilisées : Le projet s'inscrit dans plusieurs tendances technologiques actuelles. L'**Internet des Objets (IoT) et les systèmes de gestion intelligente des bâtiments** représentent un marché en forte croissance, visant à optimiser la gestion des infrastructures. Les **applications mobiles cross-platform** permettent de développer des solutions accessibles sur différents systèmes d'exploitation (iOS et Android) avec un seul code base, réduisant les coûts et accélérant le développement. Les **Progressive Web Apps (PWA)** offrent une expérience utilisateur fluide sur navigateur tout en conservant des fonctionnalités avancées. L'**analyse de données et l'intelligence artificielle** permettent de passer d'une maintenance réactive à une maintenance prédictive, anticipant les pannes avant qu'elles ne surviennent. Les **architectures cloud et les pratiques DevOps** facilitent le déploiement, la mise à l'échelle et la maintenance des applications. Enfin, le **développement d'applications offline-first** répond aux contraintes de connectivité encore présentes dans certains contextes africains.

Adoption massive des smartphones : En Afrique subsaharienne, le smartphone est devenu le principal dispositif d'accès à Internet et aux services numériques. Cette réalité justifie le choix d'une approche mobile-first pour ce projet, garantissant une accessibilité maximale pour tous les utilisateurs (occupants et agents de terrain). La familiarité croissante des utilisateurs avec les interfaces mobiles modernes (inspirées d'applications populaires comme WhatsApp, Facebook) facilite l'adoption de nouvelles solutions.

Besoins sociétaux : Au-delà des aspects technologiques, le projet répond à plusieurs besoins sociétaux importants. L'**amélioration des conditions de vie sur les campus universitaires** constitue un enjeu de bien-être étudiant crucial pour la réussite académique. L'**optimisation des ressources** dans un contexte de contraintes budgétaires permet aux établissements d'enseignement de maximiser l'impact de leurs investissements. La **professionnalisation des services support** contribue à l'amélioration globale de la qualité de l'enseignement supérieur. Enfin, la **contribution à une gestion durable** des infrastructures, par la prolongation de la durée de vie des équipements et la réduction du gaspillage, s'inscrit dans une démarche de responsabilité environnementale.

Contexte spécifique de l'IUSJC : L'Institut Universitaire Saint Jean, avec ses trois sites et sa croissance continue, fait face à des défis de gestion opérationnelle proportionnels à son expansion. Le site d'Eyang, le plus important, comprend des infrastructures variées nécessitant une maintenance régulière et coordonnée. L'absence actuelle d'outil de gestion structuré crée

des inefficacités qui impactent directement la satisfaction des occupants et la productivité des équipes de maintenance. La mise en place de cette plateforme numérique s'inscrit dans une vision stratégique de modernisation et d'amélioration continue des services de l'établissement.

3 Contraintes générales de conception

La conception et le développement de la plateforme devront respecter plusieurs catégories de contraintes qui encadrent le projet.

Contraintes techniques :

Maintenabilité et évolutivité : Le système devra être conçu selon une architecture modulaire facilitant les modifications futures et l'ajout de nouvelles fonctionnalités. Le code devra respecter les bonnes pratiques de développement (clean code, SOLID principles) et être documenté de manière exhaustive.

Scalabilité : L'architecture devra supporter une montée en charge progressive, permettant d'étendre la solution aux autres sites de l'IUSJC (Douala, Etokoss) sans refonte majeure. La base de données et les API devront être conçues pour gérer efficacement un volume croissant d'utilisateurs, d'espaces et d'incidents.

Performance : Les temps de réponse devront rester acceptables même avec une connexion internet limitée. L'application mobile devra fonctionner de manière fluide sur des smartphones d'entrée et de milieu de gamme, représentatifs du parc de dispositifs utilisés au Cameroun.

Compatibilité : Les applications mobiles devront supporter les versions récentes d'Android et iOS. L'application web devra être compatible avec les principaux navigateurs (Chrome, Firefox, Safari, Edge).

Contraintes de sécurité et réglementaires :

Protection des données personnelles : Le système devra respecter les principes de protection des données à caractère personnel conformément aux réglementations applicables (RGPD et loi camerounaise sur la protection des données). Les données sensibles (mots de passe, informations personnelles) devront être chiffrées. L'accès aux données devra être contrôlé selon le principe du moindre privilège.

Authentification et autorisation : Un système robuste d'authentification devra être mis en place pour chaque catégorie d'utilisateurs (occupants, agents, superviseur, administrateur). Les sessions devront être sécurisées et les mots de passe stockés de manière sécurisée (hachage avec algorithmes robustes comme bcrypt).

Traçabilité : Toutes les actions critiques (création d'incidents, modifications de statut d'équipements, interventions) devront être tracées avec horodatage et identification de l'auteur, permettant un audit complet du système.

Contraintes d'expérience utilisateur (UX/UI) :

Simplicité et intuitivité : Les interfaces devront être extrêmement simples et intuitives, tenant compte de la diversité des profils utilisateurs. Les parcours utilisateurs principaux devront nécessiter un minimum d'étapes.

Design moderne et cohérent : L'interface devra suivre les tendances actuelles de design (Material Design pour Android, Human Interface Guidelines pour iOS) tout en conservant une identité visuelle propre à l'IUSJC.

Accessibilité : Les applications devront respecter les principes d'accessibilité pour permettre leur utilisation par des personnes en situation de handicap (contraste des couleurs, taille des textes, support des lecteurs d'écran).

Feedback utilisateur : Chaque action devra donner un feedback immédiat à l'utilisateur (confirmation visuelle, messages de succès/erreur clairs, indicateurs de chargement).

Contraintes de déploiement et d'intégration :

Automatisation du déploiement : Un pipeline CI/CD devra être mis en place pour automatiser les tests, le build et le déploiement de l'application web. Cette approche garantit la qualité du code et accélère les cycles de mise en production.

Documentation complète : Une documentation technique exhaustive devra accompagner le projet, facilitant la maintenance future et le transfert de connaissances.

Tests : Des tests unitaires, d'intégration et fonctionnels devront être mis en œuvre pour garantir la fiabilité du système.

Contraintes de connectivité :

Fonctionnement offline : L'application mobile des agents de terrain devra pouvoir fonctionner sans connexion internet, avec synchronisation automatique des données lorsque la connexion est rétablie. Ce mode offline devra couvrir les fonctionnalités critiques (consultation des incidents, prise en charge, marquage de résolution).

Optimisation de la bande passante : Les échanges de données devront être optimisés (compression, pagination) pour fonctionner efficacement même avec une connexion 3G limitée.

Contraintes budgétaires et temporelles :

Respect du calendrier académique : Le projet devra être livré avant les dates de présentation fixées (2, 3 ou 21 février 2026), imposant une gestion rigoureuse du planning et des priorités.

- **Utilisation de technologies open-source :** Pour minimiser les coûts, le projet privilégiera les frameworks et outils open-source, tout en assurant leur pérennité et leur support communautaire.

III Interfaces et interactions du système

1 Interfaces externes du système

Le système de gestion de maintenance de l'IUSJC interagira avec plusieurs services et systèmes externes pour assurer son fonctionnement complet.

Services de messagerie et notifications :

Firebase Cloud Messaging (FCM) / Apple Push Notification Service (APNS) : Ces services permettront l'envoi de notifications push en temps réel aux applications mobiles des occupants et des agents. Les notifications seront déclenchées lors d'événements critiques (nouvel incident, prise en charge, résolution, rappels).

Service d'email (SMTP) : Un serveur de messagerie sera utilisé pour l'envoi d'emails automatiques, notamment lors de la création de comptes agents (envoi des identifiants de connexion) et pour les notifications importantes aux administrateurs.

Services de géolocalisation et cartographie :

API de géolocalisation indoor : Une solution de géolocalisation intérieure (indoor positioning) devra être intégrée pour permettre la localisation précise des espaces au sein des bâtiments. Des technologies comme le Wi-Fi positioning, les beacons Bluetooth, ou des solutions propriétaires pourront être envisagées.

Bibliothèques de cartographie interactive : Des bibliothèques comme Leaflet, Mapbox ou Google Maps API seront utilisées pour afficher les plans des bâtiments de manière interactive, permettant à l'administrateur de visualiser la disposition des espaces et de naviguer facilement.

Services de stockage de fichiers :

Stockage cloud (AWS S3, Google Cloud Storage, Azure Blob Storage ou similaire) : Les photos prises par les occupants et agents lors de la déclaration d'incidents devront être stockées de manière sécurisée et accessible. Un service de stockage cloud garantira la disponibilité, la durabilité et l'évolutivité du stockage des médias.

Service de compression et optimisation d'images : Un service automatisé (comme Cloudinary, ImageKit ou une solution open-source) pourra être intégré pour optimiser automatiquement les images téléchargées, réduisant ainsi la consommation de bande passante et l'espace de stockage nécessaire.

Services d'analyse et de prédiction :

API de machine learning : Pour la fonctionnalité de prédiction des équipements susceptibles de tomber en panne, des services de machine learning (TensorFlow, PyTorch, scikit-learn, ou des services cloud comme AWS SageMaker, Google AI Platform) pourront être utilisés. Ces services analyseront l'historique des incidents pour identifier des patterns et prédire les futures défaillances.

Services d'authentification externe (optionnel) :

OAuth 2.0 / Single Sign-On (SSO) : Si l'IUSJC dispose déjà d'un système d'authentification centralisé (par exemple basé sur Active Directory ou un fournisseur d'identité), le système pourra s'intégrer via des protocoles standards (OAuth 2.0, SAML) pour permettre une authentification unique.

Services de génération de rapports :

Bibliothèques d'export Excel : Des bibliothèques comme Apache POI (Java), OpenPyXL (Python), ou ExcelJS (JavaScript) seront utilisées pour générer les rapports Excel demandés par les agents et le superviseur.

Services de génération de PDF : Des bibliothèques comme PDFKit, jsPDF, ou des services comme Puppeteer pourront être utilisées pour générer des rapports au format PDF si nécessaire.

2 Interfaces internes

Le système sera structuré en plusieurs modules communiquant entre eux via des interfaces bien définies.

API RESTful Backend :

L'API backend constituera le cœur du système, exposant des endpoints REST pour toutes les opérations métier. Elle sera organisée en modules fonctionnels (authentification, gestion des utilisateurs, gestion des espaces, gestion des équipements, gestion des incidents, reporting, analyses prédictives).

Chaque module exposera des routes HTTP (GET, POST, PUT, DELETE) suivant les conventions REST, retournant des réponses au format JSON.

L'API implémentera un système de middleware pour la gestion de l'authentification (vérification des tokens JWT), l'autorisation (vérification des permissions), la validation des données entrantes, et la gestion des erreurs.

Base de données :

Le backend communiquera avec la base de données via un ORM (Object-Relational Mapping) comme Sequelize (pour bases relationnelles) ou Mongoose (pour MongoDB), abstraisant les requêtes SQL/NoSQL et facilitant la manipulation des données.

Un système de migrations sera mis en place pour gérer l'évolution du schéma de base de données de manière contrôlée et réversible.

Applications mobiles :

Les applications mobiles (occupants et agents) communiqueront exclusivement avec le backend via l'API REST, envoyant des requêtes HTTP/HTTPS et recevant des réponses JSON.

Pour la messagerie temps réel et les notifications, une connexion WebSocket pourra être établie entre les applications mobiles et le backend, permettant une communication bidirectionnelle instantanée.

Les applications mobiles intégreront un système de cache local (SQLite, Hive, ou SharedPreferences) pour stocker temporairement les données et permettre le fonctionnement offline.

Un module de synchronisation sera implémenté pour gérer les conflits et assurer la cohérence des données entre le cache local et le backend lors de la reconnexion.

Application web administrateur :

L'application web communiquera également avec le backend via l'API REST, utilisant des requêtes asynchrones (fetch API, Axios) pour charger et mettre à jour les données.

Le front-end web sera structuré en composants réutilisables (React, Vue.js, Angular), suivant une architecture de type SPA (Single Page Application) pour une expérience fluide.

Module de géolocalisation indoor :

Un module spécifique sera développé pour gérer la cartographie des bâtiments et la localisation des espaces. Ce module exposera des fonctions permettant de charger les plans, d'afficher les espaces sur la carte, et de naviguer entre les différents étages et bâtiments.

L'interface de ce module sera utilisée à la fois par l'application web (pour la configuration et la visualisation) et potentiellement par les applications mobiles (pour aider les agents à localiser précisément un espace).

Module d'analyse prédictive :

Un module d'analyse et de prédiction sera développé, exposant des fonctions permettant d'entraîner des modèles de machine learning sur l'historique des incidents et de générer des prédictions sur les équipements à risque.

Ce module pourra être exécuté de manière asynchrone (via des jobs planifiés) pour ne pas impacter les performances du backend principal.

Les résultats des prédictions seront stockés en base de données et exposés via l'API REST pour consultation par l'administrateur.

3 Conditions d'intégration

La réussite du projet nécessite le respect de conditions d'intégration strictes pour garantir l'interopérabilité, la maintenabilité et l'évolutivité du système.

Architecture modulaire et découplage :

Le système sera conçu selon une architecture en couches (présentation, logique métier, accès aux données) et en modules fonctionnels indépendants.

Chaque module devra exposer des interfaces claires (contrats) définissant les inputs attendus et les outputs produits, permettant de modifier l'implémentation interne d'un module sans impacter les autres.

Le principe d'injection de dépendances sera appliqué pour faciliter les tests et permettre le remplacement de composants (par exemple, remplacer un service de stockage par un autre).

Standards et conventions :

Des conventions de nommage cohérentes seront adoptées pour les routes API, les noms de variables, les noms de fichiers, etc.

Le format de réponse des API sera standardisé (par exemple : *{ **success**: boolean, **data**: object, **message**: string, **error**: object }*).

Les codes de statut HTTP seront utilisés correctement (200 pour succès, 201 pour création, 400 pour erreur client, 401 pour non authentifié, 403 pour non autorisé, 404 pour non trouvé, 500 pour erreur serveur).

Gestion des versions :

Un système de versionnage sémantique (Semantic Versioning) sera appliqué à l'API (par exemple : v1.0.0, v1.1.0, v2.0.0).

Les routes API pourront inclure le numéro de version dans l'URL (par exemple : /api/v1/incidents) pour permettre l'évolution de l'API sans casser la compatibilité avec les clients existants.

Documentation des interfaces :

Une documentation exhaustive de l'API sera produite, décrivant chaque endpoint, les paramètres attendus, les réponses possibles, et des exemples d'utilisation.

Des outils comme Swagger/OpenAPI, Postman, ou des générateurs de documentation automatiques pourront être utilisés pour maintenir la documentation à jour.

Gestion des erreurs et des exceptions :

Un système centralisé de gestion des erreurs sera mis en place, capturant toutes les exceptions et retournant des messages d'erreur cohérents et informatifs aux clients.

Les erreurs seront loggées de manière structurée pour faciliter le débogage et le monitoring.

Tests d'intégration :

Des tests d'intégration automatisés seront développés pour vérifier que les différents modules communiquent correctement entre eux.

Ces tests couvriront les scénarios principaux (création d'incident, prise en charge par un agent, résolution, génération de rapport) et vérifieront la cohérence des données à travers les différentes couches du système.

Environnements de développement, test et production :

Plusieurs environnements distincts seront mis en place (développement, test/staging, production) pour permettre le développement et les tests sans impacter le système en production.

Les configurations spécifiques à chaque environnement (URLs des services, clés API, paramètres de base de données) seront externalisées et gérées via des variables d'environnement.

Stratégie de déploiement et de mise à jour :

Un pipeline CI/CD automatisera le processus de build, test et déploiement, garantissant que seul du code testé et validé est déployé en production.

Des stratégies de déploiement progressif (blue-green deployment, canary deployment) pourront être envisagées pour minimiser les risques lors des mises à jour.

Monitoring et observabilité :

Des outils de monitoring seront mis en place pour surveiller en temps réel les performances du système, détecter les anomalies, et alerter en cas de problème (temps de réponse élevés, taux d'erreur anormal, indisponibilité).

Des logs structurés seront collectés et centralisés dans une solution de log management (comme ELK Stack, Graylog, ou des solutions cloud) pour faciliter le débogage et l'analyse des incidents.

Des métriques d'utilisation seront collectées (nombre d'incidents créés, temps moyen de résolution, taux d'utilisation des différentes fonctionnalités) pour permettre l'amélioration continue du système.

Sécurité et conformité :

- Toutes les communications entre les clients et le backend devront être chiffrées via HTTPS/TLS.
- Les données sensibles en base de données (mots de passe, informations personnelles) devront être chiffrées au repos.
- Un système de gestion des secrets (comme HashiCorp Vault, AWS Secrets Manager) sera utilisé pour stocker de manière sécurisée les clés API, les tokens, et autres informations sensibles, évitant leur inclusion dans le code source.
- Des audits de sécurité réguliers seront effectués, incluant des tests de pénétration et des revues de code focalisées sur la sécurité.

Dépendances et bibliothèques tierces :

- Les dépendances externes (frameworks, bibliothèques) devront être soigneusement sélectionnées en privilégiant les solutions matures, bien maintenues et largement adoptées.
- Un fichier de gestion des dépendances (package.json pour Node.js, requirements.txt pour Python, pubspec.yaml pour Flutter) listera explicitement toutes les dépendances et leurs versions.
- Des outils d'analyse de vulnérabilités (comme npm audit, Snyk, Dependabot) seront utilisés pour détecter et corriger les failles de sécurité dans les dépendances.

Interopérabilité et standards ouverts :

- Le système respectera les standards ouverts et largement adoptés (REST, JSON, OAuth 2.0, JWT) pour faciliter l'intégration future avec d'autres systèmes.

- Les formats de données exportés (Excel, PDF) suivront les standards de l'industrie pour garantir leur compatibilité avec les outils externes.

Gestion du cache et de la synchronisation :

- Pour le fonctionnement offline des applications mobiles, une stratégie claire de gestion du cache sera définie, spécifiant quelles données sont mises en cache, leur durée de validité, et les conditions de synchronisation.
- Un mécanisme de résolution de conflits sera implémenté pour gérer les cas où des modifications concurrentes sont effectuées sur les mêmes données (par exemple, un incident modifié offline par un agent pendant qu'un autre agent le modifie online).
- La synchronisation sera conçue pour être efficace (synchronisation incrémentale plutôt que complète) et résiliente (reprise automatique en cas d'interruption).

Extensibilité et évolutivité géographique :

- Bien que le système soit initialement déployé pour le site d'Eyang, l'architecture devra permettre l'ajout futur des sites de Douala et Etokoss sans modification majeure.
- Un système de multi-tenancy (si pertinent) ou de configuration par site sera prévu pour gérer les spécificités de chaque campus (différentes structures de bâtiments, différentes équipes de maintenance).

Formation et accompagnement :

- Des supports de formation (guides utilisateurs, tutoriels vidéo) seront produits pour chaque catégorie d'utilisateurs (occupants, agents, administrateur).
- Un accompagnement au changement sera prévu pour faciliter l'adoption du système, incluant des sessions de formation en présentiel pour les agents et le personnel administratif.
- Un système de support utilisateur sera mis en place (par exemple, une adresse email dédiée, un canal de communication) pour recueillir les retours, répondre aux questions, et résoudre les problèmes rencontrés par les utilisateurs.

Métriques de succès et amélioration continue :

- Des indicateurs clés de performance (KPI) seront définis pour mesurer le succès du système : temps moyen de résolution des incidents, taux de satisfaction des occupants, nombre d'incidents déclarés par période, taux d'utilisation du système par les agents, précision des prédictions, etc.
- Ces métriques seront suivies régulièrement et analysées pour identifier les axes d'amélioration et prioriser les évolutions futures du système.

Ce premier chapitre établit les fondations du projet en présentant de manière exhaustive le contexte, les objectifs, les enjeux et les contraintes qui encadrent le développement de la plateforme de gestion de maintenance de l'IUSJC. Il permet de comprendre le "pourquoi" du projet et définit clairement son périmètre et ses ambitions. Les chapitres suivants détailleront respectivement l'analyse approfondie des besoins et l'identification précise des problèmes à

résoudre (Chapitre II), puis la conception technique et la mise en œuvre de la solution (Chapitre III).

CHAPITRE II : ANALYSE ET MODÉLISATION DU SYSTÈME

I. Étude des besoins

1 Besoins fonctionnels

Le système de gouvernance des infrastructures de l'IUSJC devra répondre aux besoins métier de trois catégories d'utilisateurs distincts : les occupants (clients), les agents de terrain/superviseurs et les administrateurs. Cette section détaille les fonctionnalités attendues pour chaque profil.

Gestion de l'authentification et des accès

Le système devra permettre l'authentification sécurisée des utilisateurs selon leur profil. Les occupants de chambres se connecteront via leur numéro de chambre et un mot de passe fournis lors de la réservation. Les agents de terrain et superviseurs s'authentifieront via un login et un mot de passe. Les administrateurs accéderont à la plateforme web via des identifiants dédiés. Le système devra gérer les sessions utilisateurs avec expiration automatique pour les occupants dont la période d'occupation est terminée. Un mécanisme de gestion des comptes inactifs devra empêcher la connexion des agents désactivés.

Gestion des incidents et requêtes

Les occupants devront pouvoir déclarer un incident en capturant une photo de l'équipement défectueux et en ajoutant un commentaire descriptif. Ils pourront consulter l'historique de leurs requêtes envoyées et recevoir des notifications push lors de la résolution des incidents.

Les agents de terrain recevront des notifications push pour chaque nouvel incident déclaré. Ils pourront prendre en charge un incident en modifiant le statut de l'équipement concerné. La consultation de l'historique des incidents sera possible avec filtrage par statut (Bon état, À réparer, À remplacer, En maintenance). Les agents pourront marquer un incident comme résolu ou signaler qu'un équipement nécessite un remplacement. Ils auront également la capacité d'enregistrer des incidents concernant les salles de classe ou bureaux, sans obligation de photo. Le système devra supporter le mode hors ligne pour permettre le travail sur le terrain sans connexion internet constante.

Les superviseurs bénéficieront des mêmes fonctionnalités que les agents avec des privilèges supplémentaires : activation/désactivation de comptes agents et génération de rapports Excel transmissibles aux décideurs.

Gestion des espaces et équipements

L'administrateur devra pouvoir configurer et cartographier l'ensemble des bâtiments, étages et espaces (chambres, salles, bureaux) via l'importation de fichiers Excel. Une géolocalisation indoor via carte interactive devra être proposée pour faciliter le repérage des espaces.

La gestion des équipements comprendra leur création, leur assignation aux espaces et le suivi de leur état. Chaque équipement sera associé à un statut parmi : Bon état, À réparer, À remplacer, En maintenance. L'administrateur pourra consulter les informations détaillées sur chaque espace : bâtiment d'appartenance, étage, numéro, liste des équipements avec leurs états respectifs.

Le système devra étiqueter visuellement les espaces comportant au moins un équipement non fonctionnel pour faciliter l'identification des zones nécessitant une intervention.

Gestion des occupants

L'administrateur configurera la liste des occupants de la Cité Universitaire et du bâtiment des pères via l'importation de fichiers Excel. Chaque occupant disposera d'une session temporelle correspondant à sa durée d'occupation. L'expiration automatique de la session empêchera l'accès à l'application mobile au-delà de la période autorisée. Pour tout nouvel occupant, une nouvelle session sera créée avec génération automatique des identifiants de connexion.

Gestion des agents de terrain

L'administrateur pourra enregistrer de nouveaux agents, activer ou désactiver leurs comptes. La création d'un agent déclenchera l'envoi automatique des paramètres d'accès (login et mot de passe) par email. Un agent au statut inactif ne pourra pas se connecter à l'application mobile.

Tableau de bord et statistiques

Les agents et superviseurs accéderont à un tableau de bord synthétisant la situation globale des équipements : nombre d'équipements en bon état, à réparer, à remplacer, en maintenance, ainsi que la liste des espaces ayant subi le plus de dégâts. La génération de rapports Excel sera possible pour transmission aux responsables.

L'administrateur bénéficiera de visualisations statistiques avancées comprenant :

- Le nombre d'incidents déclarés par étage et par bâtiment
- Les chambres les plus défectueuses par bâtiment et par étage
- Les chambres n'ayant enregistré aucun incident
- La répartition des équipements par statut et par bâtiment
- Des périodes d'analyse configurables (journalière, hebdomadaire, mensuelle, semestrielle, annuelle)

Fonctionnalités analytiques et prédictives

Le système devra intégrer des capacités prédictives permettant à l'administrateur de visualiser les équipements susceptibles de rencontrer un incident à court terme, avec identification précise de l'espace concerné (bureau, chambre, salle). Cette fonctionnalité s'appuiera sur l'historique des incidents et l'analyse des patterns de dysfonctionnement.

Système de notifications

Le système implémentera un mécanisme de notifications push permettant :

- D'alerter les occupants de la résolution de leurs incidents
- D'informer les agents de terrain et superviseurs des nouveaux incidents déclarés
- De notifier les agents des incidents nécessitant une intervention urgente
- D'alerter les administrateurs des tendances critiques identifiées

2 Besoins non fonctionnels

Au-delà des fonctionnalités métier, le système devra répondre à des exigences de qualité, de performance et de sécurité essentielles à son exploitation dans le contexte universitaire.

Performance et scalabilité

Le système devra supporter une charge croissante d'utilisateurs sans dégradation des performances. Les temps de réponse ne devront pas excéder 3 secondes pour les opérations courantes (consultation d'incidents, affichage de listes). Le chargement des tableaux de bord statistiques devra s'effectuer en moins de 5 secondes même avec un volume important de données historiques.

L'architecture devra être conçue pour la scalabilité horizontale, permettant l'ajout de serveurs supplémentaires en cas d'augmentation de la charge. Le système devra supporter simultanément au minimum 200 utilisateurs actifs (agents, occupants, administrateurs) sans ralentissement perceptible.

Les opérations de chargement de données via fichiers Excel devront être optimisées pour traiter efficacement des fichiers contenant jusqu'à 1000 lignes en moins de 30 secondes.

Fiabilité et disponibilité

Le système devra garantir un taux de disponibilité minimal de 99%, soit moins de 7 heures d'indisponibilité par mois. Cette exigence implique la mise en place de mécanismes de redondance et de basculement automatique en cas de défaillance.

Le mode hors ligne de l'application mobile agents devra garantir la synchronisation automatique des données dès le rétablissement de la connexion, sans perte d'information. Les données saisies hors ligne devront être stockées localement et transmises au serveur dès que possible.

Un système de sauvegarde automatique quotidien devra être implémenté, avec conservation des sauvegardes sur une période minimale de 30 jours. La restauration complète du système devra être possible en moins de 4 heures en cas de sinistre.

Sécurité et confidentialité

Toutes les communications entre les applications clientes (web et mobile) et le serveur devront être chiffrées via HTTPS/TLS. Les données sensibles stockées en base de données (mots de passe, informations personnelles des occupants) devront être chiffrées au repos.

L'authentification devra être robuste avec utilisation de tokens sécurisés (JWT) et politique de renouvellement. Les mots de passe devront être hachés avec un algorithme sécurisé (bcrypt ou Argon2) avant stockage, avec un salt unique par utilisateur.

Le système devra implémenter une politique de gestion des droits d'accès stricte, garantissant que chaque utilisateur n'accède qu'aux fonctionnalités et données correspondant à son rôle. Les sessions inactives devront expirer automatiquement après 30 minutes d'inactivité pour les administrateurs et agents, et après 2 heures pour les occupants.

Un journal d'audit devra tracer toutes les opérations sensibles (modification de statut d'équipement, création/suppression d'utilisateurs, génération de rapports) avec enregistrement de l'identifiant de l'utilisateur, de l'action effectuée et de l'horodatage.

Ergonomie et expérience utilisateur

L'interface utilisateur devra être intuitive et ne nécessiter aucune formation préalable pour les occupants. Les applications mobiles devront respecter les guidelines Material Design (Android) et Human Interface Guidelines (iOS) pour garantir une expérience familière aux utilisateurs.

Les parcours utilisateurs principaux (déclaration d'incident pour les occupants, prise en charge d'incident pour les agents) devront être réalisables en moins de 5 interactions. Les formulaires devront intégrer une validation en temps réel avec messages d'erreur explicites.

L'application web administrateur devra être responsive et fonctionnelle sur différentes tailles d'écran (desktop, tablette) tout en restant optimisée pour une utilisation desktop en priorité.

Les temps de chargement des écrans ne devront pas excéder 2 secondes, avec affichage d'indicateurs de progression pour les opérations plus longues. Le système devra fournir un feedback immédiat pour toute action utilisateur (changement visuel des boutons, messages de confirmation).

Compatibilité et portabilité

Les applications mobiles devront fonctionner sur Android 8.0+ (API level 26+) et iOS 12+, couvrant ainsi plus de 95% des dispositifs en circulation. L'application web devra être compatible avec les navigateurs modernes : Chrome 90+, Firefox 88+, Safari 14+, Edge 90+.

Le système devra fonctionner de manière optimale avec des connexions 3G/4G, fréquentes dans le contexte camerounais. Les médias (photos d'incidents) devront être compressés automatiquement pour limiter la consommation de bande passante tout en conservant une qualité suffisante pour l'identification des problèmes.

Maintenabilité et évolutivité

Le code source devra respecter les conventions et bonnes pratiques de développement pour faciliter la maintenance et l'ajout de nouvelles fonctionnalités. Une architecture modulaire devra permettre l'évolution du système sans nécessiter de refonte complète.

Une documentation technique exhaustive devra être produite, couvrant l'architecture du système, les choix de conception, les procédures d'installation et de déploiement, ainsi que la documentation des API. Des commentaires pertinents devront être intégrés dans le code pour faciliter la compréhension par de futurs développeurs.

Des tests automatisés (unitaires pour la logique métier critique, d'intégration pour les API) devront être mis en place pour détecter les régressions lors de modifications futures. La couverture de code par les tests devra atteindre au minimum 70% pour les composants critiques.

Le système de gestion de versions (Git via GitLab ou GitHub) devra être utilisé rigoureusement avec une stratégie de branches claire (main/develop/feature) et des procédures de merge avec revue de code.

Déploiement et intégration continue

Un processus CI/CD (Continuous Integration/Continuous Deployment) devra être mis en place pour automatiser les tests et le déploiement de l'application web. Chaque commit sur la branche principale devra déclencher automatiquement l'exécution des tests et le déploiement en environnement de production si les tests passent.

L'application web devra être déployée sur un serveur accessible publiquement avec une URL stable communiquée aux utilisateurs. Les fichiers APK pour les applications mobiles (occupants et agents) devront être générés et mis à disposition pour distribution.

Un système de logs centralisé devra permettre le monitoring du système en production et la détection rapide d'anomalies ou d'erreurs. Les métriques de performance et d'utilisation devront être collectées pour permettre l'optimisation continue du système.

Accessibilité

Le système devra respecter les standards d'accessibilité WCAG 2.1 niveau AA au minimum, garantissant l'utilisabilité par des personnes en situation de handicap. Les contrastes de couleurs devront être suffisants, les textes alternatifs devront être présents pour les images, et la navigation au clavier devra être possible dans l'application web.

Les messages d'erreur et notifications devront être clairs et formulés dans un français accessible, évitant le jargon technique pour les utilisateurs non spécialistes.

3 Contraintes d’exploitation

Le système devra opérer dans le contexte spécifique de l'IUSJC site d'Eyang, avec ses caractéristiques et contraintes particulières.

Contraintes environnementales

Le système devra fonctionner dans un environnement multi-sites (Eyang, Douala, Etokoss) avec une architecture permettant une extension future aux autres sites. La priorité actuelle est le site d'Eyang comportant 4 bâtiments principaux : 2 bâtiments pédagogiques et administratifs, 1 bâtiment cité universitaire, 1 bâtiment des pères.

La connectivité internet peut être variable selon les zones du campus. L'application mobile agents devra impérativement supporter le mode hors ligne pour permettre le travail sur le terrain. L'infrastructure réseau existante devra être prise en compte, avec possibilité de fonctionnement sur WiFi institutionnel et données mobiles 3G/4G.

Contraintes organisationnelles

Le système devra s'intégrer dans les processus existants de gestion des infrastructures sans nécessiter de réorganisation majeure du service. Une période de transition avec coexistence des méthodes traditionnelles et du système numérique devra être prévue.

La formation des utilisateurs devra être minimale : maximum 1 heure pour les agents de terrain, 2 heures pour les administrateurs. La documentation utilisateur devra être claire et accessible en français.

Contraintes utilisateurs

Le système devra s'adapter à des utilisateurs ayant des niveaux d'alphabétisation numérique variables. Les occupants peuvent être des étudiants technophiles mais aussi des personnes moins familières avec les technologies mobiles. L'interface devra donc être simple et intuitive pour tous.

Les agents de terrain sont en mobilité constante dans les bâtiments. L'application mobile devra être optimisée pour une utilisation rapide avec une main, des écrans de saisie simplifiés et un minimum de texte à saisir.

Contraintes de charge

Le système devra supporter une charge utilisateur variable selon les périodes :

- Pics d'utilisation en début d'année académique lors de l'installation des nouveaux occupants
- Augmentation des déclarations d'incidents en périodes d'examens (stress accru sur les équipements)
- Utilisation plus réduite pendant les périodes de vacances

Le nombre estimé d'utilisateurs simultanés :

- Occupants : jusqu'à 500 étudiants et résidents
- Agents de terrain : 5 à 10 agents actifs simultanément
- Superviseurs : 2 à 3 superviseurs
- Administrateurs : 1 à 2 administrateurs

Le volume de données devra être géré de manière optimale :

- Environ 200 à 500 espaces (chambres, salles, bureaux) à gérer
- Plusieurs milliers d'équipements répartis dans ces espaces
- Plusieurs centaines d'incidents déclarés mensuellement en période académique
- Conservation de l'historique sur plusieurs années pour les analyses prédictives

Contraintes temporelles

Le projet doit être livré pour le 2, 3 ou 21 février 2026 selon le calendrier de présentation. Cette contrainte temporelle impose un développement agile avec priorisation des fonctionnalités critiques et livraisons incrémentales.

La mise en production devra idéalement coïncider avec une période de faible activité (vacances ou début de semestre) pour faciliter la transition et permettre la formation des utilisateurs sans perturber le fonctionnement normal.

Contraintes réglementaires

Le système devra respecter la réglementation camerounaise sur la protection des données personnelles. Les données des occupants (identité, numéro de chambre, période d'occupation) sont considérées comme sensibles et devront être protégées conformément aux standards de sécurité.

Un consentement explicite des occupants pour le traitement de leurs données devra être obtenu lors de la première connexion. Une politique de confidentialité claire devra être accessible et acceptée par tous les utilisateurs.

II. Acteurs et cas d'utilisation

1 Identification des acteurs internes

Le système de gouvernance des infrastructures implique plusieurs acteurs internes à l'organisation IUSJC, chacun ayant des responsabilités et des interactions spécifiques avec la plateforme.

Administrateur système

L'administrateur système représente le rôle de plus haut niveau dans l'organisation. Il est responsable de la configuration globale du système et de la supervision de son fonctionnement. Ses responsabilités incluent la gestion complète de la cartographie des bâtiments (création, modification des bâtiments, étages et espaces), la configuration et le suivi

des équipements affectés aux différents espaces, la gestion du cycle de vie des agents de terrain (création de comptes, activation, désactivation), la gestion des occupants et de leurs sessions d'accès, l'exploitation des données statistiques et la génération de rapports analytiques pour la direction, ainsi que la visualisation des prédictions sur les équipements à risque de dysfonctionnement.

L'administrateur accède au système via une application web sécurisée. Il doit posséder une expertise technique suffisante pour comprendre l'architecture du système et effectuer des opérations de configuration complexes. Sa principale interaction est avec l'interface web administrative, et il collabore indirectement avec les superviseurs à qui il transmet les rapports générés.

Superviseur

Le superviseur occupe un rôle intermédiaire de coordination et de surveillance des opérations de maintenance. Il supervise le travail des agents de terrain, vérifie la qualité des interventions et gère l'allocation des ressources humaines selon les incidents déclarés. Ses responsabilités comprennent la consultation du tableau de bord synthétique sur l'état des équipements, l'activation et la désactivation des comptes agents de terrain, la consultation de l'historique complet des incidents tous bâtiments confondus avec possibilité de filtrage, la génération et la transmission de rapports Excel aux décideurs, le suivi des délais de résolution des incidents, ainsi que la validation de la clôture des incidents complexes nécessitant un remplacement d'équipement.

Le superviseur accède au système via une application mobile dédiée, lui permettant de suivre les opérations en temps réel même lors de ses déplacements sur le campus. Il doit posséder une bonne connaissance du fonctionnement des installations et des procédures de maintenance. Ses principales interactions sont avec les agents de terrain qu'il supervise et avec l'administrateur système à qui il remonte les problématiques structurelles.

Agent de terrain

Les agents de terrain constituent l'effectif opérationnel chargé de l'entretien et de la maintenance des équipements. Ils interviennent physiquement dans les espaces pour résoudre les incidents signalés ou détectés. Leurs responsabilités incluent la réception et la consultation des notifications d'incidents, la prise en charge des incidents en modifiant le statut des équipements concernés, l'enregistrement de nouveaux incidents détectés lors de leurs rondes (salles de classe, bureaux), la mise à jour du statut des incidents (en cours, résolu, nécessite remplacement), la consultation du tableau de bord pour prioriser leurs interventions, ainsi que la synchronisation des données en mode hors ligne.

Les agents accèdent au système via une application mobile optimisée pour une utilisation sur le terrain. Ils doivent posséder des compétences techniques en maintenance des équipements courants (électricité, plomberie, mobilier) et une capacité à diagnostiquer rapidement les problèmes. Leurs principales interactions sont avec les occupants dont ils résolvent les problèmes, avec leur superviseur à qui ils rendent compte, et avec l'application mobile qui guide leurs interventions.

2 Identification des acteurs externes

Le système interagit également avec des acteurs externes à l'équipe de maintenance, qui utilisent la plateforme pour signaler des problèmes ou reçoivent des informations du système.

Occupant Cité Universitaire

Les occupants de la Cité Universitaire sont principalement des étudiants résidant dans les chambres du bâtiment dédié. Ils utilisent le système pour signaler les dysfonctionnements d'équipements dans leur espace de vie. Leurs interactions avec le système comprennent l'authentification via leur numéro de chambre et mot de passe fournis lors de l'installation, la déclaration d'incidents en capturant une photo de l'équipement défectueux et en ajoutant un commentaire descriptif, la consultation de l'historique de leurs requêtes avec visualisation du statut de traitement, ainsi que la réception de notifications push lors de la résolution de leurs incidents.

Les occupants accèdent au système via une application mobile simple d'utilisation. Ils n'ont pas besoin de compétences techniques particulières, l'interface devant être intuitive et accessible. Leur session d'accès est temporelle, limitée à leur période d'occupation de la chambre. Leurs principales interactions sont avec les agents de terrain qui interviendront suite à leurs déclarations.

Occupant bâtiment des pères

Les occupants du bâtiment des pères représentent le personnel religieux résidant de manière permanente ou semi-permanente sur le campus. Leur utilisation du système est similaire à celle des occupants de la Cité Universitaire, avec les mêmes fonctionnalités de déclaration d'incidents et de suivi. La différence réside dans la durée potentiellement plus longue de leur session d'accès, correspondant à leur période de résidence qui peut s'étendre sur plusieurs années.

Système de notification push (Firebase Cloud Messaging)

Firebase Cloud Messaging (FCM) constitue un acteur externe de type système, fournissant le service de notifications push aux applications mobiles. Il interagit avec le backend du système pour transmettre les notifications aux utilisateurs concernés : alertes aux agents de terrain lors de nouveaux incidents, confirmations aux occupants lors de résolutions d'incidents, ainsi que rappels et alertes diverses selon la configuration du système.

FCM n'a pas d'interface utilisateur directe mais constitue un composant technique essentiel à l'expérience utilisateur en temps réel.

Service d'envoi d'emails

Le service d'envoi d'emails (SMTP ou service tiers comme SendGrid) constitue un autre acteur externe système. Il est sollicité pour transmettre les identifiants de connexion aux nouveaux agents de terrain lors de la création de leur compte, envoyer des notifications importantes aux administrateurs et superviseurs (rapports périodiques, alertes critiques), ainsi

que pour communiquer les informations d'accès aux occupants lors de leur installation si cette procédure est automatisée.

3 Diagramme et description des cas d'utilisation

3.1 Diagramme UML des cas d'utilisation

Le système de gouvernance des infrastructures de l'IUSJC s'organise autour de quatre modules principaux correspondant aux différents acteurs.

Module Occupant

- Cas d'utilisation : S'authentifier, Déclarer un incident, Consulter historique des incidents, Recevoir notification de résolution
- Acteur principal : Occupant (Cité U ou bâtiment des pères)
- Relations : Le cas "Déclarer un incident" inclut "Capturer photo équipement" et "Ajouter commentaire"

Module Agent de terrain

- Cas d'utilisation : S'authentifier, Recevoir notification nouvel incident, Consulter liste incidents, Filtrer incidents par statut, Prendre en charge incident, Modifier statut équipement, Marquer incident résolu, Signaler besoin remplacement, Enregistrer incident salle/bureau, Consulter tableau de bord, Synchroniser données hors ligne
- Acteur principal : Agent de terrain
- Relations : "Prendre en charge incident" inclut "Modifier statut équipement", "Consulter liste incidents" étend "Filtrer incidents par statut"

Module Superviseur

- Cas d'utilisation : Tous les cas d'utilisation de l'Agent de terrain, plus Activer/Désactiver compte agent, Générer rapport Excel, Transmettre rapport, Valider résolution incident complexe
- Acteur principal : Superviseur
- Relations : Superviseur hérite de tous les cas d'utilisation de l'Agent de terrain (relation de généralisation)

Module Administrateur

- Cas d'utilisation : S'authentifier, Gérer configuration bâtiments, Gérer configuration étages, Gérer configuration espaces, Gérer équipements, Assigner équipements aux espaces, Créer compte agent, Activer/Désactiver agent, Configurer occupants, Gérer sessions occupants, Consulter informations espace, Visualiser liste espaces, Étiqueter espaces défectueux, Visualiser statistiques incidents, Visualiser chambres défectueuses, Visualiser chambres sans incident, Visualiser statistiques équipements, Visualiser prédictions dysfonctionnements, Importer données Excel, Configurer géolocalisation indoor
- Acteur principal : Administrateur système

- Relations : "Gérer configuration bâtiments" inclut "Importer données Excel" et "Configurer géolocalisation indoor", "Créer compte agent" inclut l'acteur externe "Système envoi emails"

Acteurs externes systèmes

- Firebase Cloud Messaging : Intervient dans les cas d'utilisation de notifications (Recevoir notification nouvel incident, Recevoir notification de résolution)
- Système envoi emails : Intervient dans "Créer compte agent"

3.2 Description textuelle des cas d'utilisation principaux

CU01 - Déclarer un incident (Occupant)

Acteur principal : Occupant (Cité U ou bâtiment des pères)

Objectif : Signaler un dysfonctionnement d'équipement dans l'espace occupé pour obtenir une intervention de maintenance

Préconditions :

- L'occupant doit être authentifié dans l'application mobile
- La session de l'occupant doit être active (période d'occupation en cours)
- L'occupant doit avoir une connexion internet (même limitée)

Scénario nominal :

1. L'occupant accède à la fonctionnalité "Déclarer un incident" depuis l'écran d'accueil de l'application
2. Le système affiche un formulaire de déclaration avec les champs : équipement concerné (liste déroulante pré-remplie des équipements de la chambre), zone de capture photo, zone de texte pour commentaire
3. L'occupant sélectionne l'équipement défectueux dans la liste
4. L'occupant clique sur le bouton "Prendre une photo"
5. Le système active l'appareil photo du smartphone
6. L'occupant capture une photo de l'équipement défectueux
7. Le système affiche un aperçu de la photo avec options "Revalider" ou "Conserver"
8. L'occupant confirme la photo en cliquant "Conserver"
9. L'occupant saisit un commentaire explicatif dans la zone de texte (minimum 10 caractères)
10. L'occupant clique sur le bouton "Soumettre"
11. Le système valide les données saisies (présence de photo, longueur minimale du commentaire)
12. Le système compresse la photo pour optimiser la bande passante
13. Le système envoie les données au serveur backend

14. Le système affiche un message de confirmation "Votre incident a été enregistré. Vous serez notifié de sa résolution."
15. Le système envoie une notification push aux agents de terrain via FCM
16. Le cas d'utilisation se termine avec succès

Scénarios alternatifs :

A1 : Photo non conforme

- Au point 7, si la photo est floue ou inadaptée
- Le système détecte que la photo ne respecte pas les critères minimaux de qualité
- Le système affiche un message "Photo de qualité insuffisante. Veuillez réessayer avec un meilleur éclairage."
- Retour au point 4

A2 : Commentaire trop court

- Au point 11, si le commentaire contient moins de 10 caractères
- Le système affiche un message d'erreur "Veuillez décrire le problème plus en détail (minimum 10 caractères)."
- Le focus est placé dans la zone de texte commentaire
- Retour au point 9

A3 : Pas de connexion internet

- Au point 13, si aucune connexion internet n'est disponible
- Le système affiche un message "Connexion internet non disponible. Votre déclaration sera envoyée dès rétablissement de la connexion."
- Le système stocke temporairement l'incident en local
- L'incident est transmis automatiquement lors de la prochaine connexion internet
- Le cas d'utilisation se termine

Exceptions :

E1 : Session expirée

- À tout moment, si la session de l'occupant a expiré
- Le système affiche un message "Votre session a expiré. Votre période d'occupation est terminée. Merci de contacter l'administration si nécessaire."
- Le système déconnecte automatiquement l'utilisateur
- Le cas d'utilisation échoue

E2 : Erreur serveur

- Au point 13, si le serveur ne répond pas ou retourne une erreur

- Le système affiche un message "Une erreur technique est survenue. Veuillez réessayer dans quelques instants."
- Le système propose un bouton "Réessayer"
- Si l'utilisateur clique "Réessayer", retour au point 13
- Sinon, le cas d'utilisation échoue

Postconditions :

- Un nouvel incident est créé dans la base de données avec le statut "En attente de prise en charge"
- L'incident est associé à l'espace (chambre) de l'occupant
- L'équipement concerné passe au statut "À réparer" ou "En maintenance"
- Une notification push est envoyée à tous les agents de terrain actifs
- L'historique des incidents de l'occupant est mis à jour

L'occupant peut consulter son incident dans la liste avec statut "En attente"

CU02 - Prendre en charge un incident (Agent de terrain)

Acteur principal : Agent de terrain

Objectif : Accepter la responsabilité de résoudre un incident signalé et mettre à jour le statut de l'équipement concerné

Préconditions :

- L'agent doit être authentifié dans l'application mobile
- Le compte de l'agent doit avoir le statut "Actif"
- Au moins un incident doit être en statut "En attente de prise en charge"
- L'agent doit avoir accès à la liste des incidents (connexion internet ou données synchronisées en mode hors ligne)

Scénario nominal :

1. L'agent reçoit une notification push "Nouvel incident déclaré : [Type équipement] - Chambre [Numéro]"
2. L'agent clique sur la notification ou accède manuellement à la liste des incidents
3. Le système affiche la liste des incidents avec filtrage par défaut sur "En attente de prise en charge"
4. Pour chaque incident, le système affiche : photo miniature, type d'équipement, localisation (bâtiment/étage/numéro espace), date de déclaration, priorité visuelle (code couleur selon ancienneté)
5. L'agent sélectionne un incident dans la liste

6. Le système affiche les détails complets de l'incident : photo en taille réelle avec zoom possible, commentaire de l'occupant, type et localisation précise de l'équipement, date et heure de déclaration, informations de contact de l'occupant (numéro de chambre)
7. L'agent clique sur le bouton "Prendre en charge"
8. Le système affiche une boîte de dialogue de confirmation "Confirmer la prise en charge de cet incident ?"
9. L'agent confirme en cliquant "Oui"
10. Le système met à jour le statut de l'incident de "En attente de prise en charge" à "En cours de traitement"
11. Le système enregistre l'identifiant de l'agent et l'horodatage de la prise en charge
12. Le système modifie le statut de l'équipement concerné en "En maintenance"
13. Le système affiche un message de confirmation "Incident pris en charge avec succès. Bon courage !"
14. Le système synchronise immédiatement les données avec le serveur (si connexion disponible)
15. L'incident disparaît de la liste "En attente" et apparaît dans la liste "En cours" de l'agent
16. Le cas d'utilisation se termine avec succès

Scénarios alternatifs :

A1 : Mode hors ligne

- Au point 14, si aucune connexion internet n'est disponible
- Le système stocke la modification localement
- Le système affiche une icône indiquant "Synchronisation en attente"
- Les modifications sont transmises automatiquement lors du rétablissement de la connexion
- Le cas d'utilisation se poursuit normalement

A2 : Incident déjà pris en charge par un autre agent

- Au point 10, si un autre agent a pris en charge l'incident entre-temps
- Le système détecte que le statut de l'incident a changé
- Le système affiche un message "Cet incident a déjà été pris en charge par [Nom agent]"
- Le système retire l'incident de la liste de l'agent
- Le cas d'utilisation se termine sans modification

A3 : Agent souhaite ajouter un commentaire initial

- Après le point 9, l'agent peut cliquer sur "Ajouter un commentaire"
- Le système affiche une zone de texte
- L'agent saisit un commentaire (ex: "Intervention prévue demain matin", "Pièce de rechange nécessaire")
- Le commentaire est associé à l'incident et visible par l'occupant
- Retour au point 10

Exceptions :

E1 : Compte agent désactivé

- À tout moment, si le compte de l'agent est désactivé par un superviseur
- Le système détecte le statut "Inactif" lors de la tentative de synchronisation
- Le système affiche un message "Votre compte a été désactivé. Veuillez contacter votre superviseur."
- Le système déconnecte automatiquement l'agent
- Le cas d'utilisation échoue

E2 : Incident supprimé ou annulé

- Au point 10, si l'incident a été supprimé par un administrateur
- Le système retourne une erreur "Incident introuvable"
- Le système affiche un message "Cet incident n'est plus disponible"
- Le système retire l'incident de la liste de l'agent
- Le cas d'utilisation échoue

E3 : Erreur de validation des données

- Au point 11, si une erreur de cohérence est détectée (équipement inexistant, espace invalide)
- Le système affiche un message "Une erreur technique empêche la prise en charge. Veuillez contacter l'administrateur système."
- Le système génère un log d'erreur pour investigation
- Le cas d'utilisation échoue

Postconditions :

- Le statut de l'incident passe de "En attente de prise en charge" à "En cours de traitement"
- L'agent responsable est identifié et associé à l'incident
- La date et l'heure de prise en charge sont enregistrées
- Le statut de l'équipement concerné passe à "En maintenance"
- L'incident apparaît dans la liste des incidents en cours de l'agent
- Les statistiques du tableau de bord sont mises à jour
- Si connexion disponible, une notification peut être envoyée à l'occupant l'informant que son incident est en cours de traitement

CU03 - Marquer un incident comme résolu (Agent de terrain)

Acteur principal : Agent de terrain

Objectif : Clôturer un incident après intervention réussie et restaurer le statut fonctionnel de l'équipement

Préconditions :

- L'agent doit être authentifié dans l'application mobile
- L'incident doit être au statut "En cours de traitement" et pris en charge par cet agent
- L'agent doit avoir physiquement résolu le problème
- L'agent doit avoir accès à l'incident (connexion internet ou données synchronisées)

Scénario nominal :

1. L'agent consulte sa liste d'incidents en cours de traitement
2. Le système affiche la liste des incidents dont l'agent est responsable
3. L'agent sélectionne l'incident qu'il vient de résoudre
4. Le système affiche les détails de l'incident avec boutons d'action : "Marquer comme résolu", "Marquer à remplacer", "Ajouter un commentaire"
5. L'agent clique sur "Marquer comme résolu"
6. Le système affiche une boîte de dialogue de confirmation avec zone de commentaire optionnelle : "Confirmez-vous que l'incident est résolu ? Vous pouvez ajouter un commentaire sur l'intervention réalisée."
7. L'agent peut saisir un commentaire décrivant l'intervention effectuée (optionnel)
8. L'agent clique sur "Confirmer la résolution"
9. Le système valide l'action
10. Le système met à jour le statut de l'incident de "En cours de traitement" à "Résolu"
11. Le système enregistre la date et l'heure de résolution
12. Le système enregistre le commentaire de l'agent si fourni
13. Le système modifie le statut de l'équipement concerné en "Bon état"
14. Le système synchronise les modifications avec le serveur
15. Le système envoie une notification push à l'occupant via FCM : "Votre incident concernant [Type équipement] a été résolu. Merci de votre patience."
16. Le système affiche un message de confirmation "Incident marqué comme résolu avec succès"
17. L'incident disparaît de la liste "En cours" et apparaît dans l'historique avec statut "Résolu"
18. Les statistiques du tableau de bord sont automatiquement mises à jour
19. Le cas d'utilisation se termine avec succès

Scénarios alternatifs :

A1 : Équipement nécessite un remplacement

- Au point 5, l'agent constate que l'équipement ne peut être réparé
- L'agent clique sur "Marquer à remplacer"
- Le système affiche une boîte de dialogue "Confirmez-vous que l'équipement doit être remplacé ? Précisez la raison."

- L'agent saisit obligatoirement un commentaire expliquant pourquoi le remplacement est nécessaire
- L'agent confirme
- Le système met à jour le statut de l'incident à "En attente de remplacement"
- Le système modifie le statut de l'équipement en "À remplacer"
- Le système notifie le superviseur via notification push
- L'incident reste dans la liste de l'agent mais avec un marqueur spécial
- Le cas d'utilisation se termine avec un statut différent

A2 : Agent souhaite ajouter des photos après intervention

- Après le point 4, l'agent peut cliquer sur "Ajouter des photos"
- Le système permet la capture de jusqu'à 3 photos
- Les photos sont attachées à l'incident comme preuve de l'intervention
- Retour au point 5

A3 : Mode hors ligne

- Au point 14, si aucune connexion internet n'est disponible
- Le système stocke les modifications localement
- Le système affiche "Modifications enregistrées localement. Synchronisation en attente."
- La notification à l'occupant sera envoyée lors de la synchronisation
- Le cas d'utilisation se poursuit mais la notification est différée

A4 : Validation superviseur requise

- Pour certains types d'équipements critiques ou coûteux, après le point 9
- Le système détecte que l'équipement nécessite une validation superviseur
- Le système affiche "Cet incident nécessite une validation superviseur avant clôture définitive"
- Le statut passe à "En attente de validation superviseur" au lieu de "Résolu"
- Une notification est envoyée au superviseur
- L'incident reste en attente jusqu'à validation
- Le cas d'utilisation se termine avec un statut intermédiaire

Exceptions :

E1 : Délai de résolution anormalement court

- Au point 10, si le délai entre prise en charge et résolution est inférieur à 5 minutes
- Le système détecte une résolution suspecte (possiblement frauduleuse)
- Le système affiche un message "Le délai de résolution semble anormalement court. Êtes-vous certain d'avoir effectué l'intervention ?"
- Si l'agent confirme, le système enregistre un flag pour audit
- Le superviseur reçoit une notification pour vérification

- Le cas d'utilisation se poursuit avec audit

E2 : Incident modifié par un autre utilisateur

- Au point 10, si l'incident a été modifié par un superviseur entre-temps
- Le système détecte un conflit de version
- Le système affiche "Cet incident a été modifié récemment. Veuillez rafraîchir et réessayer."
- Le système recharge les détails actualisés de l'incident
- Retour au point 3

E3 : Erreur d'envoi de notification

- Au point 15, si le service FCM est indisponible
- Le système tente d'envoyer la notification
- L'envoi échoue mais le système stocke la notification pour réessai ultérieur
- Le système log l'erreur mais poursuit la clôture de l'incident
- Le cas d'utilisation se termine avec succès malgré l'échec de notification

Postconditions :

- Le statut de l'incident passe à "Résolu" ou "En attente de remplacement" selon le cas
- La date et l'heure de résolution sont enregistrées
- Le commentaire de l'agent sur l'intervention est sauvegardé
- Le statut de l'équipement passe à "Bon état" ou "À remplacer" selon le cas
- L'incident disparaît de la liste des incidents actifs de l'agent
- Une notification push est envoyée à l'occupant (immédiatement ou après synchronisation)
- Les statistiques sont mises à jour : nombre d'incidents résolus, temps moyen de résolution, etc.
- L'historique complet de l'incident (déclaration, prise en charge, résolution) est conservé
- Si remplacement nécessaire, le superviseur est notifié pour action appropriée

CU04 - Configurer les espaces et équipements (Administrateur)

Acteur principal : Administrateur système

Objectif : Créer et maintenir la cartographie complète des bâtiments, étages, espaces et équipements du campus avec leurs assignations

Préconditions :

- L'administrateur doit être authentifié sur l'application web
- L'administrateur doit disposer des droits d'administration complets

- Pour l'importation, l'administrateur doit disposer de fichiers Excel correctement formatés
- La connexion internet doit être stable pour les opérations de chargement de fichiers

Scénario nominal :

1. L'administrateur accède au menu "Gestion des infrastructures" depuis le tableau de bord
2. Le système affiche un sous-menu avec options : "Bâtiments", "Étages", "Espaces", "Équipements", "Importation Excel"
3. L'administrateur clique sur "Importation Excel" pour une configuration initiale ou mise à jour massive
4. Le système affiche une interface d'importation avec zones de dépôt pour 4 types de fichiers : Fichier bâtiments, Fichier espaces (chambres/salles/bureaux), Fichier équipements, Fichier assignations équipements-espaces
5. Le système affiche les modèles de fichiers Excel téléchargeables avec format attendu et exemples
6. L'administrateur télécharge les modèles, les remplit avec les données réelles
7. L'administrateur dépose ou sélectionne le fichier "Batiments.xlsx"
8. Le système valide le format du fichier (extension, colonnes requises)
9. Le système affiche un aperçu des 10 premières lignes avec nombre total de lignes détectées
10. L'administrateur vérifie l'aperçu et clique sur "Importer"
11. Le système parse le fichier et valide les données : nom unique pour chaque bâtiment, coordonnées GPS valides si fournies, type de bâtiment parmi les valeurs autorisées
12. Le système affiche une barre de progression pendant le traitement
13. Le système crée les enregistrements en base de données
14. Le système affiche un rapport d'importation : "[X] bâtiments importés avec succès, [Y] erreurs détectées"
15. Si erreurs, le système affiche la liste des lignes en erreur avec description du problème
16. L'administrateur corrige les erreurs dans le fichier et réimporte si nécessaire
17. L'administrateur répète les étapes 7-16 pour les fichiers "Espaces.xlsx", "Equipements.xlsx" et "Assignations.xlsx"
18. Une fois tous les fichiers importés, l'administrateur clique sur "Configurer géolocalisation indoor"
19. Le système affiche une interface de carte interactive
20. Pour chaque bâtiment, l'administrateur peut placer des marqueurs sur un plan ou une carte
21. L'administrateur sauvegarde la configuration géographique
22. Le système confirme "Configuration des infrastructures terminée avec succès. [Total] espaces et [Total] équipements configurés."
23. Le cas d'utilisation se termine avec succès

Scénarios alternatifs :

A1 : Configuration manuelle d'un espace individuel

- Au point 3, au lieu de l'importation Excel, l'administrateur choisit "Espaces" puis "Ajouter un espace"
- Le système affiche un formulaire : Sélection du bâtiment (liste déroulante), Sélection de l'étage (liste déroulante dépendante du bâtiment), Numéro/Nom de l'espace (texte), Type d'espace (Chambre, Salle de classe, Bureau, Autre), Capacité d'occupation (nombre), Description (texte optionnel)
- L'administrateur remplit le formulaire
- L'administrateur clique sur "Enregistrer"
- Le système valide l'unicité de l'espace (pas de doublon bâtiment+étage+numéro)
- Le système crée l'espace et affiche un message de confirmation
- L'espace apparaît dans la liste des espaces du bâtiment/étage sélectionné
- Retour au point 2 pour poursuivre la configuration

A2 : Configuration manuelle d'un équipement

- Au point 3, l'administrateur choisit "Équipements" puis "Ajouter un équipement"
- Le système affiche un formulaire : Type d'équipement (liste déroulante : Lit, Table, Chaise, Climatiseur, etc.), Marque (texte optionnel), Modèle (texte optionnel), Numéro de série (texte optionnel), Date d'acquisition (date), État initial (liste déroulante : Bon état, À réparer, À remplacer), Description (texte optionnel)
- L'administrateur remplit le formulaire
- L'administrateur clique sur "Enregistrer"
- Le système crée l'équipement avec un identifiant unique généré automatiquement
- Le système affiche "Équipement créé avec succès. ID: [ID_EQUIPEMENT]"
- L'équipement est maintenant disponible pour assignation à un espace
- Retour au point 2

A3 : Assignation manuelle d'équipements à un espace

- Au point 3, l'administrateur choisit "Espaces" puis sélectionne un espace existant
- Le système affiche les détails de l'espace avec section "Équipements assignés" (liste actuelle)
- L'administrateur clique sur "Assigner un équipement"
- Le système affiche la liste des équipements non encore assignés ou disponibles
- L'administrateur peut filtrer par type d'équipement
- L'administrateur sélectionne un ou plusieurs équipements
- L'administrateur clique sur "Assigner à cet espace"
- Le système crée les liens équipement-espace dans la base de données
- Le système met à jour l'affichage avec les équipements nouvellement assignés
- L'administrateur peut répéter l'opération pour assigner d'autres équipements
- Retour au point 2

A4 : Modification d'un espace existant

- Au point 3, l'administrateur choisit "Espaces" puis clique sur un espace dans la liste
- Le système affiche les détails complets de l'espace avec bouton "Modifier"
- L'administrateur clique sur "Modifier"
- Le système affiche le formulaire pré-rempli avec les données actuelles
- L'administrateur modifie les champs souhaités
- L'administrateur clique sur "Sauvegarder les modifications"
- Le système valide les données et enregistre les modifications
- Le système affiche "Modifications enregistrées avec succès"
- Retour à la vue détail de l'espace

Exceptions :

E1 : Fichier Excel mal formaté

- Au point 11, si le fichier ne respecte pas le format attendu
- Le système détecte des colonnes manquantes ou des types de données incorrects
- Le système affiche un message d'erreur détaillé : "Erreur de format : colonne '[NOM_COLONNE]' manquante ou invalide. Veuillez utiliser le modèle fourni."
- Le système propose de télécharger à nouveau le modèle
- Retour au point 7

E2 : Doublons détectés

- Au point 11 ou 13, si des données en doublon sont détectées
- Le système identifie les lignes contenant des doublons (ex: même numéro de chambre dans même bâtiment/étage)
- Le système affiche "Erreur : [X] doublons détectés. Lignes : [liste des numéros de lignes]"
- Le système n'importe que les lignes valides et liste les lignes ignorées
- L'administrateur doit corriger le fichier pour les doublons
- Le cas d'utilisation se poursuit partiellement

E3 : Références invalides

- Au point 17, lors de l'importation des assignations, si un équipement ou espace référencé n'existe pas
- Le système détecte que l'ID d'équipement ou d'espace n'est pas trouvé en base
- Le système affiche "Erreur de référence ligne [X] : équipement ID [ID] ou espace ID [ID] introuvable"
- Les assignations invalides sont ignorées, les valides sont importées
- L'administrateur doit vérifier les IDs dans le fichier ou créer les entités manquantes
- Le cas d'utilisation se poursuit partiellement

E4 : Fichier trop volumineux

- Au point 8, si le fichier Excel dépasse la limite autorisée (ex: 5 MB)
- Le système refuse le chargement
- Le système affiche "Fichier trop volumineux ([taille] MB). Limite autorisée : 5 MB. Veuillez scinder les données en plusieurs fichiers."
- Retour au point 7

E5 : Timeout de traitement

- Au point 13, si le traitement d'un fichier très volumineux dépasse le temps maximal autorisé
- Le système interrompt le traitement après 60 secondes
- Le système affiche "Temps de traitement dépassé. [X] lignes traitées sur [Total]. Veuillez réduire la taille du fichier."
- Les lignes traitées avant le timeout sont conservées
- L'administrateur doit importer le reste des données séparément
- Le cas d'utilisation échoue partiellement

Postconditions :

- Les bâtiments, étages et espaces sont créés et structurés hiérarchiquement dans la base de données
- Les équipements sont créés avec leurs caractéristiques et états initiaux
- Les assignations équipements-espaces sont établies et fonctionnelles
- La géolocalisation indoor est configurée et accessible depuis l'interface
- Le système est prêt à recevoir des déclarations d'incidents sur les espaces et équipements configurés
- Un rapport d'importation complet est disponible pour audit
- Les données sont immédiatement accessibles dans l'application mobile des agents
- Les occupants peuvent voir la liste des équipements de leur espace lors de déclarations d'incidents

III Modélisation conceptuelle

1 Identification des entités et attributs

La modélisation conceptuelle du système de gouvernance des infrastructures repose sur l'identification précise des entités métier, de leurs attributs et de leurs comportements. Cette section détaille les classes principales du système selon les principes de l'orienté objet.

1.1 Entités liées à la gestion des utilisateurs

Classe Utilisateur (abstraite)

La classe Utilisateur constitue la classe mère abstraite pour tous les profils d'utilisateurs du système. Elle encapsule les attributs et comportements communs à tous les types d'utilisateurs.

Attributs :

- `idUtilisateur` : String (identifiant unique, UUID)
- `email` : String (adresse email, unique)
- `motDePasseHash` : String (hash sécurisé du mot de passe, bcrypt)
- `typeUtilisateur` : TypeUtilisateur (énumération)
- `statut` : StatutCompte (énumération)
- `dateCreation` : DateTime
- `dateDerniereConnexion` : DateTime (nullable)

- `tokenAuthentification` : String (JWT token, nullable)
- `dateExpirationToken` : DateTime (nullable)

Méthodes :

- `authentifier(email: String, motDePasse: String) : Boolean` - Vérifie les identifiants et génère un token d'authentification
- `deconnecter() : void` - Invalide le token et met à jour la date de dernière connexion
- `verifierToken(token: String) : Boolean` - Valide la validité et l'intégrité du token JWT
- `changerMotDePasse(ancienMotDePasse: String, nouveauMotDePasse: String) : Boolean` - Modifie le mot de passe après vérification
- `reinitialiserMotDePasse(email: String) : void` - Génère un lien de réinitialisation et l'envoie par email
- `activerCompte() : void` - Change le statut à ACTIF
- `desactiverCompte() : void` - Change le statut à INACTIF
- `estActif() : Boolean` - Vérifie si le compte est actif
- `genererToken() : String` - Génère un nouveau JWT token

Classe Administrateur (hérite de Utilisateur)

La classe Administrateur représente les utilisateurs ayant les privilèges les plus élevés pour la configuration et la supervision globale du système.

Attributs spécifiques :

- `nom` : String
- `prenom` : String

- `telephone` : String (nullable)
- `niveauAcces` : NiveauAccesAdmin (énumération : SUPER_ADMIN, ADMIN_STANDARD)

Méthodes spécifiques :

- `creerBatiment(batiment: Batiment) : Batiment` - Crée un nouveau bâtiment dans le système
- `modifierBatiment(idBatiment: String, batiment: Batiment) : Batiment` - Met à jour les informations d'un bâtiment
- `supprimerBatiment(idBatiment: String) : Boolean` - Supprime un bâtiment et ses dépendances
- `creerEspace(espace: Espace) : Espace` - Crée un nouvel espace
- `modifierEspace(idEspace: String, espace: Espace) : Espace` - Met à jour un espace
- `supprimerEspace(idEspace: String) : Boolean` - Supprime un espace
- `creerEquipement(equipement: Equipement) : Equipement` - Crée un nouvel équipement
- `assignerEquipement(idEquipement: String, idEspace: String) : Boolean` - Associe un équipement à un espace
- `desassignerEquipement(idEquipement: String) : Boolean` - Retire l'assignation d'un équipement
- `creerAgent(agent: AgentTerrain) : AgentTerrain` - Crée un compte agent et envoie les identifiants par email
- `activerDesactiverAgent(idAgent: String, activer: Boolean) : Boolean` - Change le statut d'un agent
- `creerOccupant(occupant: Occupant) : Occupant` - Crée un compte occupant avec session temporelle
- `importerDonneesExcel(fichier: File, typeImport: TypeImport) : RapportImportation` - Importe des données massives depuis Excel
- `genererRapportStatistiques(periode: Periode, typeRapport: TypeRapport) : Rapport` - Génère des rapports analytiques
- `consulterStatistiquesGlobales(dateDebut: DateTime, dateFin: DateTime) : StatistiquesGlobales` - Récupère les statistiques sur une période
- `visualiserPredictions() : List<PredictionDysfonctionnement>` - Obtient les prédictions d'incidents
- `configurerGeolocalisation(idBatiment: String, coordonnees: Coordonnees) : Boolean` - Configure la position géographique d'un bâtiment
- `consulterHistoriqueAudit(dateDebut: DateTime, dateFin: DateTime) : List<EntreeAudit>` - Consulte le journal d'audit

Classe AgentTerrain (hérite de Utilisateur)

La classe AgentTerrain représente les agents responsables des interventions physiques de maintenance.

Attributs spécifiques :

- `nom` : String
- `prenom` : String

- `pseudo` : String (unique, affiché aux utilisateurs)
- `telephone` : String
- `specialite` : SpecialiteAgent (énumération : ELECTRICITE, PLOMBERIE, MENUISERIE, GENERALE)
- `estSuperviseur` : Boolean (indicateur si l'agent a des privilèges de supervision)
- `dateDerniereIntervention` : DateTime (nullable)
- `nombreIncidentsTraites` : Integer (compteur)
- `tempsReponseManyen` : Duration (statistique)

Méthodes spécifiques :

- `consulterIncidentsEnAttente(filtres: FiltresIncident) : List<Incident>` - Récupère les incidents disponibles
- `consulterMesIncidents(statut: StatutIncident) : List<Incident>` - Récupère les incidents dont l'agent est responsable
- `prendreEnChargeIncident(idIncident: String) : Boolean` - Prend la responsabilité d'un incident
- `marquerIncidentResolu(idIncident: String, commentaire: String) : Boolean` - Clôture un incident avec succès
- `marquerEquipementARemplacer(idIncident: String, raisonRemplacement: String) : Boolean` - Signale qu'un remplacement est nécessaire
- `ajouterCommentaireIncident(idIncident: String, commentaire: String) : Commentaire` - Ajoute un commentaire sur un incident
- `enregistrerIncidentSalle(incident: Incident) : Incident` - Crée un incident pour salle/bureau sans occupant
- `consulterTableauDeBord() : TableauDeBordAgent` - Récupère les statistiques personnelles et globales
- `genererRapportExcel(periode: Periode) : File` - Génère un rapport Excel (privilège superviseur)
- `activerDesactiverAgent(idAgent: String, activer: Boolean) : Boolean` - Gère les comptes agents (privilège superviseur uniquement)
- `synchroniserDonneesHorsLigne() : ResultatSynchronisation` - Synchronise les modifications effectuées hors ligne
- `obtenirIncidentsNonSynchronises() : List<Incident>` - Récupère les incidents modifiés localement en attente de sync

Classe Occupant (hérite de Utilisateur)

La classe Occupant représente les résidents (étudiants, pères) qui signalent les dysfonctionnements dans leurs espaces.

Attributs spécifiques :

- `numeroChambre` : String (utilisé comme login)
- `motDePasseTemporaire` : String (fourni lors de l'installation)
- `premierConnexion` : Boolean (indicateur si changement de mot de passe requis)
- `dateDebutOccupation` : DateTime
- `dateFinOccupation` : DateTime
- `typeOccupant` : TypeOccupant (énumération : ETUDIANT_CITE_U, RESIDENT_BATIMENT_PERES)

- `espacesOccupes` : List<String> (IDs des espaces occupés, généralement un seul)
- `nombreIncidentsDeclares` : Integer

Méthodes spécifiques :

- `declarerIncident(incident: Incident, photo: File, commentaire: String) : Incident` - Crée une déclaration d'incident avec photo
- `consulterMesIncidents() : List<Incident>` - Récupère l'historique des incidents déclarés
- `consulterDetailIncident(idIncident: String) : Incident` - Obtient les détails complets d'un incident
- `obtenirEquipementsChambre() : List<Equipement>` - Liste les équipements de l'espace occupé
- `sessionEstValide() : Boolean` - Vérifie si la période d'occupation est en cours
- `changerMotDePasseInitial(nouveauMotDePasse: String) : Boolean` - Change le mot de passe temporaire lors de la première connexion
- `recevoirNotificationResolution(idIncident: String) : void` - Reçoit une notification push de résolution

1.2 Énumérations liées aux utilisateurs

TypeUtilisateur

- `ADMINISTRATEUR`
- `AGENT_TERRAIN`
- `SUPERVISEUR`
- `OCCUPANT`

StatutCompte

- `ACTIF` - Compte opérationnel
- `INACTIF` - Compte désactivé temporairement ou définitivement
- `EN_ATTENTE` - Compte créé mais pas encore activé
- `EXPIRE` - Session expirée (pour occupants)
- `BLOQUE` - Compte bloqué pour raisons de sécurité

NiveauAccesAdmin

- `SUPER_ADMIN` - Accès complet à toutes les fonctionnalités incluant gestion des autres administrateurs
- `ADMIN_STANDARD` - Accès aux fonctionnalités de configuration mais pas de gestion d'autres admins

SpecialiteAgent

- `ELECTRICITE` - Spécialiste des installations électriques, climatiseurs, etc.
- `PLOMBERIE` - Spécialiste des installations sanitaires, WC, lavabos
- `MENUISERIE` - Spécialiste du mobilier, portes, fenêtres
- `GENERALE` - Agent polyvalent pour interventions diverses

TypeOccupant

- `ETUDIANT_CITE_U` - Étudiant résidant à la cité universitaire
- `RESIDENT_BATIMENT_PERES` - Résident du bâtiment des pères

1.3Entités liées à l'infrastructure

Classe Batiment

La classe Batiment représente les bâtiments physiques du campus.

Attributs :

- `idBatiment` : String (identifiant unique, UUID)
- `nom` : String (ex: "Bâtiment Pédagogique A", "Cité Universitaire")
- `code` : String (code court unique, ex: "BP-A", "CITE-U")
- `typeBatiment` : TypeBatiment (énumération)
- `adresse` : String (nullable)
- `coordonneesGPS` : Coordonnees (nullable, pour géolocalisation)
- `nombreEtages` : Integer
- `superficie` : Float (en m², nullable)
- `dateConstruction` : DateTime (nullable)
- `description` : String (nullable)
- `planBatiment` : String (URL ou chemin vers fichier plan, nullable)
- `dateCreation` : DateTime
- `dateModification` : DateTime

Méthodes :

- `ajouterEtage(etage: Etage) : Etage` - Ajoute un étage au bâtiment
- `supprimerEtage(numeroEtage: Integer) : Boolean` - Supprime un étage et ses dépendances
- `obtenirEtages() : List<Etage>` - Récupère tous les étages du bâtiment
- `obtenirEspaces() : List<Espace>` - Récupère tous les espaces de tous les étages
- `obtenirStatistiquesIncidents(periode: Periode) : StatistiquesBatiment` - Statistiques d'incidents du bâtiment
- `obtenirNombreEquipements() : Integer` - Compte total des équipements dans le bâtiment
- `obtenirNombreEquipementsParStatut(statut: StatutEquipement) : Integer` - Compte les équipements par statut
- `verifierDisponibiliteCode(code: String) : Boolean` - Vérifie l'unicité du code bâtiment

TypeBatiment

- `PEDAGOGIQUE` - Bâtiment contenant salles de classe, amphithéâtres
- `ADMINISTRATIF` - Bâtiment des bureaux administratifs
- `CITE_UNIVERSITAIRE` - Bâtiment de résidence étudiante
- `RESIDENCE_PERSONNEL` - Bâtiment des pères ou autre personnel
- `MIXTE` - Bâtiment à usage multiple

Classe Etage

La classe Etage représente un niveau d'un bâtiment.

Attributs :

- `idEtage` : String (identifiant unique, UUID)
- `idBatiment` : String (référence vers Batiment)
- `numeroEtage` : Integer (0 pour rez-de-chaussée, négatif pour sous-sol)
- `designation` : String (ex: "Rez-de-chaussée", "1er étage", "Sous-sol")
- `superficie` : Float (en m², nullable)
- `nombreEspaces` : Integer (compteur calculé)
- `planEtage` : String (URL ou chemin vers fichier plan, nullable)
- `dateCreation` : DateTime

Méthodes :

- `ajouterEspace(espace: Espace) : Espace` - Ajoute un espace à l'étage
- `supprimerEspace(idEspace: String) : Boolean` - Supprime un espace de l'étage
- `obtenirEspaces() : List<Espace>` - Récupère tous les espaces de l'étage
- `obtenirStatistiquesIncidents(periode: Periode) : StatistiquesEtage` - Statistiques d'incidents de l'étage
- `obtenirNombreEquipements() : Integer` - Compte total des équipements de l'étage
- `estPlein() : Boolean` - Vérifie si tous les espaces sont occupés (pour chambres)

Classe Espace

La classe Espace représente une chambre, salle de classe, bureau ou autre local.

Attributs :

- `idEspace` : String (identifiant unique, UUID)
- `idEtage` : String (référence vers Etage)
- `numero` : String (numéro ou nom de l'espace, ex: "C101", "Amphi A")
- `typeEspace` : TypeEspace (énumération)
- `superficie` : Float (en m², nullable)
- `capaciteOccupation` : Integer (nombre de personnes, nullable)
- `description` : String (nullable)
- `estOccupe` : Boolean (pour les chambres)
- `aEquipementDefectueux` : Boolean (flag calculé, true si au moins un équipement pas en bon état)
- `dateCreation` : DateTime
- `dateModification` : DateTime

Méthodes :

- `assignerEquipement(equipement: Equipement) : Boolean` - Ajoute un équipement à l'espace

- `retirerEquipement(idEquipement: String) : Boolean` - Retire un équipement de l'espace
- `obtenirEquipements() : List<Equipement>` - Récupère tous les équipements de l'espace
- `obtenirEquipementsParStatut(statut: StatutEquipement) : List<Equipement>` - Filtre les équipements par statut
- `verifierEquipementsDefectueux() : Boolean` - Vérifie et met à jour le flag aEquipementDefectueux
- `obtenirIncidents(actifs: Boolean) : List<Incident>` - Récupère les incidents de l'espace
- `obtenirOccupantActuel() : Occupant` - Récupère l'occupant actuel (nullable)
- `marquerOccupe(idOccupant: String) : Boolean` - Marque l'espace comme occupé
- `marquerLibre() : Boolean` - Marque l'espace comme libre
- `obtenirLocalisation() : LocalisationComplete` - Retourne bâtiment/étage/numéro formaté
- `obtenirStatistiquesIncidents(periode: Periode) : StatistiquesEspace` - Statistiques d'incidents

TypeEspace

- `CHAMBRE_SIMPLE` - Chambre pour une personne
- `CHAMBRE_DOUBLE` - Chambre pour deux personnes
- `CHAMBRE_TRIPLE` - Chambre pour trois personnes ou plus
- `SALLE_CLASSE` - Salle de cours
- `AMPHITHEATRE` - Amphithéâtre
- `BUREAU_INDIVIDUEL` - Bureau pour une personne
- `BUREAU_PARTAGE` - Bureau partagé
- `LABORATOIRE` - Laboratoire technique
- `SALLE_REUNION` - Salle de réunion
- `AUTRE` - Autre type d'espace

Classe Coordonnees

Classe pour la géolocalisation des bâtiments.

Attributs :

- `latitude` : Float
- `longitude` : Float
- `altitude` : Float (nullable)

Méthodes :

- `calculerDistance(autresCoordonnees: Coordonnees) : Float` - Calcule la distance en mètres
- `valider() : Boolean` - Vérifie la validité des coordonnées GPS

1.4 Entités liées aux équipements

Classe Equipement

La classe Equipement représente un équipement physique installé dans un espace.

Attributs :

- `idEquipement` : String (identifiant unique, UUID)
- `typeEquipement` : TypeEquipement (énumération)
- `marque` : String (nullable)
- `modele` : String (nullable)
- `numeroSerie` : String (nullable, unique si fourni)
- `statut` : StatutEquipement (énumération)
- `idEspaceActuel` : String (référence vers Espace, nullable si non assigné)
- `dateAcquisition` : DateTime (nullable)
- `valeurAchat` : Float (nullable, en FCFA)
- `dureeVieEstimee` : Integer (en mois, nullable)
- `description` : String (nullable)
- `dateCreation` : DateTime
- `dateModification` : DateTime
- `historiquePannes` : Integer (compteur de pannes)
- `dernieredatePanne` : DateTime (nullable)

Méthodes :

- `changerStatut(nouveauStatut: StatutEquipement, motif: String) : Boolean` - Met à jour le statut
- `assignerAEspace(idEspace: String) : Boolean` - Assigne l'équipement à un espace
- `retirerDeEspace() : Boolean` - Retire l'équipement de son espace actuel
- `obtenirIncidents() : List<Incident>` - Récupère tous les incidents liés à cet équipement
- `obtenirEspaceActuel() : Espace` - Récupère l'espace où est installé l'équipement (nullable)
- `calculerAgeEnMois() : Integer` - Calcule l'âge de l'équipement depuis l'acquisition
- `estSousGarantie() : Boolean` - Vérifie si l'équipement est encore sous garantie
- `incrementerHistoriquePannes() : void` - Augmente le compteur de pannes
- `obtenirTauxPanne() : Float` - Calcule le taux de pannes par an
- `necessiteRemplacement() : Boolean` - Détermine si un remplacement est recommandé

TypeEquipement

- `LIT` - Lit simple ou double
- `TABLE` - Table de travail
- `CHAISE` - Chaise
- `ARMOIRE` - Armoire de rangement
- `CLIMATISEUR` - Climatiseur ou ventilateur

- `REFRIGERATEUR` - Réfrigérateur
- `TELEVISEUR` - Téléviseur
- `BUREAU` - Bureau de travail
- `ETAGERE` - Étagère de rangement
- `LAVABO` - Lavabo
- `WC` - WC/Toilettes
- `DOUCHE` - Douche
- `PORTE` - Porte
- `FENETRE` - Fenêtre
- `PRISE_ELECTRIQUE` - Prise électrique
- `LAMPE` - Lampe/Luminaire
- `TABLEAU_BLANC` - Tableau blanc (salle de classe)
- `PROJECTEUR` - Projecteur vidéo
- `AUTRE` - Autre type d'équipement

StatutEquipement

- `BON_ETAT` - Équipement fonctionnel
- `A_REPARER` - Équipement défectueux mais réparable
- `A_REMPLACER` - Équipement irréparable nécessitant remplacement
- `EN_MAINTENANCE` - Équipement en cours de réparation
- `HORS_SERVICE` - Équipement définitivement hors service
- `EN_ATTENTE_PIECE` - Réparation suspendue en attente de pièce détachée

1.5 Entités liées aux incidents

Classe Incident

La classe Incident représente une déclaration de dysfonctionnement d'équipement.

Attributs :

- `idIncident` : String (identifiant unique, UUID)
- `idEquipement` : String (référence vers Equipement)
- `idEspace` : String (référence vers Espace)
- `idDeclarant` : String (référence vers Occupant ou AgentTerrain, selon origine)
- `typeDeclarant` : TypeDeclarant (énumération)
- `idAgentResponsable` : String (référence vers AgentTerrain, nullable avant prise en charge)
- `statut` : StatutIncident (énumération)
- `priorite` : PrioriteIncident (énumération, calculée automatiquement)
- `titre` : String (généré automatiquement ou saisi)
- `description` : String
- `photoPrincipale` : String (URL ou chemin vers photo)
- `photosSupplementaires` : List<String> (URLs, nullable)
- `dateDeclaration` : DateTime
- `datePriseEnCharge` : DateTime (nullable)
- `dateResolution` : DateTime (nullable)
- `dureeResolution` : Duration (calculée, nullable)
- `commentaireAgent` : String (nullable)

- `raisonRemplacement` : String (nullable, si équipement à remplacer)
- `necessiteValidationSuperviseur` : Boolean
- `dateValidationSuperviseur` : DateTime (nullable)
- `noteOccupant` : Integer (évaluation de 1 à 5, nullable)
- `commentaireOccupant` : String (nullable)

Méthodes :

- `prendreEnCharge(idAgent: String) : Boolean` - Assigne l'incident à un agent
- `marquerResolu(commentaire: String, photos: List<File>) : Boolean` - Clôture l'incident avec succès
- `marquerARemplacer(raison: String) : Boolean` - Signale que l'équipement doit être remplacé
- `ajouterCommentaire(auteur: String, contenu: String) : Commentaire` - Ajoute un commentaire
- `obtenirCommentaires() : List<Commentaire>` - Récupère tous les commentaires
- `calculerPriorite() : PrioriteIncident` - Détermine la priorité selon critères (type équipement, ancienneté, etc.)
- `obtenirDelaiTraitement() : Duration` - Calcule le temps écoulé depuis déclaration
- `estEnRetard() : Boolean` - Vérifie si le délai standard de traitement est dépassé
- `necessiteEscalade() : Boolean` - Détermine si une escalade au superviseur est nécessaire
- `evaluerParOccupant(note: Integer, commentaire: String) : Boolean` - Permet à l'occupant d'évaluer l'intervention
- `obtenirHistoriqueStatuts() : List<HistoriqueStatut>` - Retourne l'historique des changements de statut

TypeDeclarant

- `OCCUPANT` - Incident déclaré par un occupant
- `AGENT_TERRAIN` - Incident créé par un agent lors d'une ronde

StatutIncident

- `EN_ATTENTE_PRISE_EN_CHARGE` - Incident déclaré mais pas encore assigné
- `EN_COURS_TRAITEMENT` - Incident pris en charge par un agent
- `EN_ATTENTE_PIECE` - Traitement suspendu en attente de pièce/équipement
- `EN_ATTENTE_VALIDATION` - Résolution en attente de validation superviseur
- `RESOLU` - Incident clôturé avec succès
- `EN_ATTENTE_REMPLACEMENT` - Équipement marqué à remplacer
- `ANNULE` - Incident annulé (doublon, erreur, etc.)
- `ESCALADE` - Incident escaladé au superviseur pour traitement spécial

PrioriteIncident

- `CRITIQUE` - Nécessite intervention immédiate (sécurité, électricité, eau)
- `HAUTE` - Doit être traité dans les 24h (équipements essentiels)
- `MOYENNE` - Traitement sous 48-72h (équipements de confort)

- `BASSE` - Peut être traité sous une semaine (équipements non essentiels)

Classe Commentaire

La classe Commentaire permet les échanges sur un incident.

Attributs :

- `idCommentaire` : String (UUID)
- `idIncident` : String (référence vers Incident)
- `idAuteur` : String (référence vers Utilisateur)
- `typeAuteur` : TypeUtilisateur
- `contenu` : String
- `dateCreation` : DateTime
- `pieceJointe` : String (URL, nullable)

Méthodes :

- `modifier(nouveauContenu: String) : Boolean` - Modifie le contenu (dans délai limité)
- `supprimer() : Boolean` - Supprime le commentaire
- `obtenirAuteur() : Utilisateur` - Récupère les informations de l'auteur

Classe HistoriqueStatut

La classe HistoriqueStatut trace tous les changements de statut d'un incident.

Attributs :

- `idHistorique` : String (UUID)
- `idIncident` : String (référence vers Incident)
- `ancienStatut` : StatutIncident (nullable pour création initiale)
- `nouveauStatut` : StatutIncident
- `idUtilisateur` : String (qui a effectué le changement)
- `motif` : String (raison du changement, nullable)
- `dateChangement` : DateTime

Méthodes :

- `obtenirDuree() : Duration` - Calcule la durée dans ce statut (jusqu'au prochain changement ou maintenant)

1.6 Entités liées aux notifications

Classe Notification

La classe Notification gère les notifications push envoyées aux utilisateurs.

Attributs :

- `idNotification` : String (UUID)
- `idDestinataire` : String (référence vers Utilisateur)
- `typeNotification` : TypeNotification (énumération)
- `titre` : String
- `message` : String
- `donneesSupplementaires` : Map<String, String> (données JSON pour deep linking, nullable)
- `estLue` : Boolean
- `dateEnvoi` : DateTime
- `dateLecture` : DateTime (nullable)
- `tokenFCM` : String (token Firebase du dispositif)
- `statutEnvoi` : StatutEnvoiNotification (énumération)
- `messageErreur` : String (nullable)

Méthodes :

- `envoyer() : Boolean` - Envoie la notification via FCM
- `marquerCommeLue() : void` - Marque la notification comme lue
- `reessayerEnvoi() : Boolean` - Réessaye l'envoi en cas d'échec
- `obtenirLienProfond() : String` - Génère le lien deep link vers l'écran approprié dans l'app

TypeNotification

- `NOUVEL_INCIDENT` - Nouvel incident déclaré (pour agents)
- `INCIDENT_PRIS_EN_CHARGE` - Incident pris en charge (pour occupant)
- `INCIDENT_RESOLU` - Incident résolu (pour occupant)
- `INCIDENT_EN_RETARD` - Incident dépassant délai standard (pour superviseur)
- `COMPTE_CREE` - Nouveau compte créé (pour agent)
- `COMPTE_DESACTIVE` - Compte désactivé (pour agent)
- `SESSION_EXPIREE` - Session occupant expirée (pour occupant)
- `RAPPEL_INCIDENT` - Rappel sur incident non traité
- `VALIDATION_REQUISE` - Validation superviseur nécessaire
- `AUTRE` - Autre type de notification

StatutEnvoiNotification

- `EN_ATTENTE` - Notification créée mais pas encore envoyée
- `ENVOYE` - Notification envoyée avec succès
- `ECHEC` - Échec d'envoi
- `EXPIRE` - Notification expirée (trop ancienne)

1.7 Entités liées aux statistiques et rapports

Classe StatistiquesGlobales

La classe StatistiquesGlobales agrège les données pour le tableau de bord administrateur.

Attributs :

- `idStatistiques` : String (UUID)
- `dateDebut` : DateTime
- `dateFin` : DateTime
- `nombreTotalIncidents` : Integer
- `nombreIncidentsResolus` : Integer
- `nombreIncidentsEnCours` : Integer
- `nombreIncidentsEnAttente` : Integer
- `tempsResolutionMoyen` : Duration
- `nombreEquipementsBonEtat` : Integer
- `nombreEquipementsAReparer` : Integer
- `nombreEquipementsARemplacer` : Integer
- `nombreEquipementsEnMaintenance` : Integer
- `tauxResolution` : Float (pourcentage)
- `tauxSatisfactionOccupants` : Float (moyenne des notes, nullable)
- `nombreAgentsActifs` : Integer
- `espacesPlusDefectueux` : List<EspaceStatistique>
- `equipementsPlusDefectueux` : List<EquipementStatistique>
- `batimentsPlusIncidents` : List<BatimentStatistique>

Méthodes :

- `calculerStatistiques(dateDebut: DateTime, dateFin: DateTime) : StatistiquesGlobales` - Calcule les statistiques sur la période
- `exporter FormatExcel() : File` - Génère un fichier Excel avec les statistiques
- `exporterFormatPDF() : File` - Génère un rapport PDF
- `comparerAvecPeriodePrecedente() : ComparaisonStatistiques` - Compare avec période précédente

Classe TableauDeBordAgent

La classe TableauDeBordAgent fournit les statistiques pour un agent spécifique.

Attributs :

- `idAgent` : String
- `periode` : Periode
- `nombreIncidentsTraites` : Integer
- `nombreIncidentsEnCours` : Integer
- `tempsResolutionMoyen` : Duration
- `tauxResolution` : Float
- `noteEvaluationMoyenne` : Float (nullable)
- `listeIncidentsEnCours` : List<Incident>
- `listeIncidentsEnRetard` : List<Incident>
- `statistiquesParTypeEquipement` : Map<TypeEquipement, Integer>

Méthodes :

- `actualiser() : void` - Recalcule les statistiques en temps réel
- `exporterRapport() : File` - Génère un rapport Excel

Classe PredictionDysfonctionnement

La classe PredictionDysfonctionnement contient les prédictions d'incidents futurs basées sur l'analyse des données historiques.

Attributs :

- `idPrediction` : String (UUID)
- `idEquipement` : String
- `idEspace` : String
- `probabilitePanne` : Float (entre 0 et 1)
- `dateEstimeePanne` : DateTime (estimation)
- `facteursPredictifs` : List<String> (raisons de la prédiction : fréquence pannes, âge, type équipement, etc.)
- `niveauConfiance` : NiveauConfiance (énumération)
- `recommandationAction` : String (action préventive suggérée)
- `dateCalcul` : DateTime
- `baseDonnees` : SourceDonnees (informations sur les données utilisées pour la prédiction)

Méthodes :

- `calculerPredictions(dateDebut: DateTime, dateFin: DateTime) : List<PredictionDysfonctionnement>` - Analyse les patterns et génère les prédictions
- `obtenirEquipement() : Equipement` - Récupère l'équipement concerné
- `obtenirEspace() : Espace` - Récupère l'espace concerné
- `genererRapportPredictif() : Rapport` - Crée un rapport détaillé avec recommandations
- `marquerTraitee(action: String) : void` - Enregistre qu'une action préventive a été prise

NiveauConfiance

- `TRES_ELEVE` - Confiance > 85% basée sur données robustes
- `ELEVE` - Confiance 70-85%
- `MOYEN` - Confiance 50-70%
- `FAIBLE` - Confiance < 50%, à prendre avec précaution

Classe Rapport

La classe Rapport représente les rapports générés par le système.

Attributs :

- `idRapport` : String (UUID)
- `typeRapport` : TypeRapport (énumération)
- `titre` : String
- `dateDebut` : DateTime

- `dateFin` : DateTime
- `idGenerateur` : String (référence vers Utilisateur ayant généré le rapport)
- `dateGeneration` : DateTime
- `contenuJSON` : String (données structurées du rapport)
- `fichierExcel` : String (URL ou chemin vers fichier Excel généré, nullable)
- `fichierPDF` : String (URL ou chemin vers fichier PDF généré, nullable)
- `parametres` : Map<String, String> (paramètres utilisés pour générer le rapport)

Méthodes :

- `generer(parametres: Map<String, String>) : Rapport` - Génère le rapport selon les paramètres
- `exporterExcel() : File` - Exporte le rapport en format Excel
- `exporterPDF() : File` - Exporte le rapport en format PDF
- `envoyer ParEmail(destinataire: String) : Boolean` - Envoie le rapport par email
- `telecharger() : File` - Télécharge le fichier du rapport
- `archiver() : void` - Archive le rapport pour conservation

TypeRapport

- `INCIDENTS_PERIODE` - Rapport des incidents sur une période
- `PERFORMANCE_AGENTS` - Rapport de performance des agents
- `ETAT_EQUIPEMENTS` - Rapport sur l'état des équipements
- `ESPACES_DEFECTUEUX` - Rapport des espaces problématiques
- `PREDICTIONS` - Rapport des prédictions de dysfonctionnements
- `SYNTHESE_GLOBALE` - Rapport de synthèse complet
- `CUSTOM` - Rapport personnalisé

1.8 Entités liées aux importations

Classe RapportImportation

La classe RapportImportation contient les résultats d'une importation de données via fichier Excel.

Attributs :

- `idRapport` : String (UUID)
- `typeImport` : TypeImport (énumération)
- `nomFichier` : String
- `dateImportation` : DateTime
- `idImportateur` : String (référence vers Administrateur)
- `nombreLignesTotal` : Integer
- `nombreLignesValides` : Integer
- `nombreLignesErreur` : Integer
- `lignesErreur` : List<LigneErreur>
- `dureeTraitement` : Duration
- `statutImportation` : StatutImportation (énumération)
- `messageGlobal` : String

Méthodes :

- `ajouterErreur(numeroLigne: Integer, message: String) : void` - Ajoute une erreur détectée
- `calculerTauxSucces() : Float` - Calcule le pourcentage de lignes importées avec succès
- `exporterRapportErreurs() : File` - Génère un fichier Excel avec uniquement les lignes en erreur
- `relancerImportation() : RapportImportation` - Réessaye l'importation (après corrections)

TypeImport

- `BATIMENTS` - Importation de bâtiments
- `ESPACES` - Importation d'espaces
- `EQUIPEMENTS` - Importation d'équipements
- `ASSIGNATIONS` - Importation des assignations équipements-espaces
- `OCCUPANTS` - Importation des occupants
- `AGENTS` - Importation des agents de terrain

StatutImportation

- `EN_COURS` - Importation en cours de traitement
- `TERMINE_SUCCES` - Importation réussie (100% des lignes)
- `TERMINE_PARTIEL` - Importation partiellement réussie (certaines lignes en erreur)
- `ECHEC` - Importation échouée complètement
- `ANNULE` - Importation annulée par l'utilisateur

Classe LigneErreur

La classe LigneErreur détaille une erreur d'importation.

Attributs :

- `numeroLigne` : Integer
- `donneesLigne` : Map<String, String> (contenu de la ligne)
- `typeErreur` : TypeErreur (énumération)
- `messageErreur` : String
- `champsConcernes` : List<String> (colonnes posant problème)

TypeErreur

- `DONNEE_MANQUANTE` - Champ obligatoire vide
- `FORMAT_INVALIDE` - Données dans un format incorrect
- `DOUBLON` - Violation de contrainte d'unicité
- `REFERENCE_INVALIDE` - Référence vers entité inexistante
- `VALEUR_HORS_LIMITE` - Valeur en dehors des limites acceptables
- `CONTRAINTE_METIER` - Violation d'une règle métier

1.9 Entités liées à l'audit et la traçabilité

Classe EntreeAudit

La classe EntreeAudit trace toutes les opérations sensibles effectuées dans le système.

Attributs :

- `idEntree` : String (UUID)
- `idUtilisateur` : String (référence vers Utilisateur)
- `typeUtilisateur` : TypeUtilisateur
- `action` : TypeAction (énumération)
- `entiteCible` : String (type d'entité concernée : Incident, Equipement, etc.)
- `idEntiteCible` : String (identifiant de l'entité modifiée)
- `detailsAvant` : String (état avant modification, JSON, nullable)
- `detailsApres` : String (état après modification, JSON, nullable)
- `adresseIP` : String
- `userAgent` : String (navigateur ou app mobile)
- `dateAction` : DateTime
- `reussie` : Boolean
- `messageErreur` : String (nullable)

Méthodes :

- `enregistrer(utilisateur: Utilisateur, action: TypeAction, entite: String, id: String, details: Map) : void` - Crée une entrée d'audit
- `rechercher(criteres: CriteresRecherche) : List<EntreeAudit>` - Recherche dans les logs
- `exporterJournal(dateDebut: DateTime, dateFin: DateTime) : File` - Exporte le journal d'audit
- `analyserTendances() : AnalyseAudit` - Analyse les patterns d'utilisation

TypeAction

- `CREATION` - Création d'une entité
- `MODIFICATION` - Modification d'une entité
- `SUPPRESSION` - Suppression d'une entité
- `CONSULTATION` - Consultation de données sensibles
- `CONNEXION` - Connexion d'un utilisateur
- `DECONNEXION` - Déconnexion d'un utilisateur
- `ACTIVATION_COMPTE` - Activation d'un compte
- `DESACTIVATION_COMPTE` - Désactivation d'un compte
- `EXPORTATION_DONNEES` - Export de données (rapport, Excel, etc.)
- `IMPORTATION_DONNEES` - Import de données massives
- `CHANGEMENT_MOT_PASSE` - Changement de mot de passe
- `TENTATIVE_ACCES_NON_AUTORISE` - Tentative d'accès à ressource non autorisée

1.10 Entités liées aux sessions

Classe SessionOccupant

La classe SessionOccupant gère les sessions temporelles des occupants.

Attributs :

- `idSession` : String (UUID)
- `idOccupant` : String (référence vers Occupant)
- `idEspace` : String (référence vers Espace)
- `dateDebut` : DateTime
- `dateFin` : DateTime
- `estActive` : Boolean
- `motDePasseTemporaire` : String (hash)
- `motDePasseModifie` : Boolean
- `dateCreation` : DateTime
- `dateModification` : DateTime
- `commentaire` : String (raison de la session, nullable)

Méthodes :

- `creerSession(occupant: Occupant, espace: Espace, dateDebut: DateTime, dateFin: DateTime) : SessionOccupant` - Crée une nouvelle session
- `verifierValidite() : Boolean` - Vérifie si la session est encore valide (date actuelle entre début et fin)
- `prolonger(nouvelleDateFin: DateTime) : Boolean` - Prolonge la durée de la session
- `terminerPrematurement() : void` - Termine la session avant la date prévue
- `renouveler(nouvelleDateDebut: DateTime, nouvelleDateFin: DateTime) : SessionOccupant` - Crée une nouvelle session pour le même occupant
- `envoyerNotificationExpiration(joursAvant: Integer) : void` - Envoie une notification X jours avant expiration
- `verifierExpirationProche() : Boolean` - Vérifie si expiration dans moins de 7 jours
- `genererCredentiels() : Map<String, String>` - Génère les identifiants de connexion

1.11 Entités utilitaires

Classe Periode

Classe utilitaire pour définir des périodes temporelles.

Attributs :

- `dateDebut` : DateTime
- `dateFin` : DateTime
- `typePeriode` : TypePeriode (énumération, nullable)

Méthodes :

- `estValide() : Boolean` - Vérifie que dateDebut < dateFin
- `contient(date: DateTime) : Boolean` - Vérifie si une date est dans la période
- `chevauche(autrePeriode: Periode) : Boolean` - Vérifie si deux périodes se chevauchent
- `duree() : Duration` - Calcule la durée de la période
- `creerPeriodeJour(date: DateTime) : Periode` - Crée une période d'un jour
- `creerPeriodeSemaine(date: DateTime) : Periode` - Crée une période d'une semaine
- `creerPeriodeMois(date: DateTime) : Periode` - Crée une période d'un mois
- `creerPeriodeSemestre(date: DateTime) : Periode` - Crée une période de 6 mois
- `creerPeriodeAnnee(date: DateTime) : Periode` - Crée une période d'un an

TypePeriode

- `JOURNALIERE` - Période d'un jour
- `HEBDOMADAIRE` - Période d'une semaine
- `MENSUELLE` - Période d'un mois
- `TRIMESTRIELLE` - Période de 3 mois
- `SEMESTRIELLE` - Période de 6 mois
- `ANNUELLE` - Période d'un an
- `PERSONNALISEE` - Période personnalisée

Classe FiltresIncident

Classe utilitaire pour filtrer les incidents.

Attributs :

- `statuts` : List<StatutIncident> (nullable)
- `priorites` : List<PrioriteIncident> (nullable)
- `typesEquipement` : List<TypeEquipement> (nullable)
- `idBatiment` : String (nullable)
- `idEtage` : String (nullable)
- `idEspace` : String (nullable)
- `idAgent` : String (nullable, pour filtrer par agent responsable)
- `dateDebutPeriode` : DateTime (nullable)
- `dateFinPeriode` : DateTime (nullable)
- `motsCles` : String (recherche textuelle, nullable)

Méthodes :

- `appliquer(incidents: List<Incident>) : List<Incident>` - Applique les filtres à une liste
- `estVide() : Boolean` - Vérifie si aucun filtre n'est défini
- `reinitialiser() : void` - Réinitialise tous les filtres

Classe LocalisationComplete

Classe utilitaire pour représenter une localisation complète.

Attributs :

- `nomBatiment` : String
- `codeBatiment` : String
- `numeroEtage` : Integer
- `designationEtage` : String
- `numeroEspace` : String
- `typeEspace` : TypeEspace

Méthodes :

- `formaterCourt() : String` - Retourne format court : "BP-A/2/C201"
- `formaterComplet() : String` - Retourne format détaillé : "Bâtiment Pédagogique A, 2ème étage, Chambre 201"
- `obtenirCheminArborescence() : String` - Retourne chemin : "Bâtiment > Étage > Espace"

2 Diagramme de classes (analyse)

Le diagramme de classes d'analyse représente la structure conceptuelle du système en illustrant les entités identifiées, leurs attributs principaux et les relations entre elles. Cette représentation graphique met en évidence l'architecture objet du système de gouvernance des infrastructures.

2.1 Relations entre les classes principales

Hiérarchie d'héritage des utilisateurs

- La classe abstraite `Utilisateur` est la classe parente
- `Administrateur`, `AgentTerrain` et `Occupant` héritent de `Utilisateur`
- `AgentTerrain` peut avoir un attribut `estSuperviseur` pour distinguer les superviseurs (plutôt qu'une sous-classe séparée pour simplifier)

Relations structurelles des infrastructures

- `Batiment` contient plusieurs (1..*) `Etage` - Relation de composition (agrégation forte)
- `Etage` appartient à exactement un (1) `Batiment`
- `Etage` contient plusieurs (0..*) `Espace` - Relation de composition
- `Espace` appartient à exactement un (1) `Etage`

- `Batiment` possède éventuellement des (0..1) `Coordonnees` - Relation d'association

Relations des équipements

- `Espace` contient plusieurs (0..*) `Equipement` - Relation d'association (pas composition car équipement peut exister sans être assigné)
- `Equipement` est assigné à au plus un (0..1) `Espace`

Relations des incidents

- `Incident` concerne exactement un (1) `Equipement` - Relation d'association
- `Incident` se produit dans exactement un (1) `Espace` - Relation d'association
- `Incident` est déclaré par exactement un (1) utilisateur (`Occupant` ou `AgentTerrain`) - Relation d'association
- `Incident` est traité par au plus un (0..1) `AgentTerrain` - Relation d'association
- `Incident` possède plusieurs (0..*) `Commentaire` - Relation de composition
- `Incident` possède plusieurs (0..*) `HistoriqueStatut` - Relation de composition

Relations des sessions

- `SessionOccupant` associe un (1) `Occupant` à un (1) `Espace` - Relation d'association ternaire
- `Occupant` peut avoir plusieurs (0..*) `SessionOccupant` (historique)
- `Espace` peut avoir plusieurs (0..*) `SessionOccupant` (succession d'occupants)

Relations des notifications

- `Notification` est destinée à exactement un (1) `Utilisateur` - Relation d'association
- `Utilisateur` reçoit plusieurs (0..*) `Notification`
- `Notification` peut référencer optionnellement un (0..1) `Incident` via `donneesSupplementaires`

Relations des statistiques et rapports

- `StatistiquesGlobales` agrège des informations de `Incident`, `Equipement`, `Espace`, `Batiment` - Relations de dépendance
- `TableauDeBordAgent` agrège des informations pour un (1) `AgentTerrain` spécifique - Relation d'association
- `PredictionDysfonctionnement` concerne un (1) `Equipement` et un (1) `Espace` - Relations d'association
- `Rapport` est généré par un (1) utilisateur (`Administrateur` ou `AgentTerrain` superviseur) - Relation d'association

Relations d'audit

- `EntreeAudit` trace une action effectuée par un (1) `Utilisateur` - Relation d'association
- `EntreeAudit` concerne potentiellement une entité quelconque (polymorphisme via attributs `entiteCible` et `idEntiteCible`)

Relations d'importation

- `RapportImportation` est créé par un (1) `Administrateur` - Relation d'association
- `RapportImportation` contient plusieurs (0..*) `LigneErreur` - Relation de composition

2.2 Multiplicités des associations clés

- Batiment (1) ←→ (1..*) Etage
- Etage (1) ←→ (0..*) Espace
- Espace (0..1) ←→ (0..*) Equipement
- Equipement (1) ←→ (0..*) Incident
- Espace (1) ←→ (0..*) Incident
- Utilisateur (1) ←→ (0..*) Incident [déclarant]
- AgentTerrain (0..1) ←→ (0..*) Incident [responsable]
- Incident (1) ←→ (0..*) Commentaire
- Incident (1) ←→ (1..*) HistoriqueStatut
- Occupant (1) ←→ (0..*) SessionOccupant
- Espace (1) ←→ (0..*) SessionOccupant
- Utilisateur (1) ←→ (0..*) Notification
- AgentTerrain (1) ←→ (0..1) TableauDeBordAgent
- Equipement (1) ←→ (0..*) PredictionDysfonctionnement
- Utilisateur (1) ←→ (0..*) Rapport [générateur]
- Utilisateur (1) ←→ (0..*) EntreeAudit [acteur]
- Administrateur (1) ←→ (0..*) RapportImportation
- RapportImportation (1) ←→ (0..*) LigneErreur

2.3 Contraintes et règles métier exprimées dans le modèle

a. **Contrainte d'unicité** : Le couple (idBatiment, numeroEtage) est unique pour Etage. Le triplet (idEtage, numero) est unique pour Espace.
b. **Contrainte de validité temporelle** : Pour SessionOccupant, dateDebut < dateFin. Pour Incident, dateDeclaration ≤ datePriseEnCharge ≤ dateResolution.
c. **Contrainte de statut** : Un Equipement ne peut être assigné à un Espace que si son statut n'est pas HORS_SERVICE.
d. **Contrainte de session** : Un Occupant ne peut avoir qu'une seule SessionOccupant active (estActive = true) à la fois.
e. **Contrainte de prise en charge** : Un Incident ne peut être pris en charge que par un AgentTerrain dont le statut est ACTIF.

f. **Contrainte de résolution** : Un Incident ne peut être marqué résolu que s'il est au statut EN_COURS_TRAITEMENT et a un agent responsable assigné.
g. **Contrainte d'escalade** : Certains types d'Equipement (valeur élevée, sécurité critique) nécessitent validation superviseur (necessiteValidationSuperviseur = true).

3 Diagramme d'objets

Le diagramme d'objets illustre un état concret du système à un instant donné, montrant des instances réelles des classes et leurs liens. Cette représentation permet de valider la cohérence du modèle de classes et de mieux comprendre le fonctionnement du système à travers un exemple pratique.

Liens entre les objets

- `batimentCiteU` contient `etage2CiteU`
- `etage2CiteU` contient `chambre205`
- `chambre205` contient `climatiseur205` et `lit1Chambre205`
- `incident001` concerne `climatiseur205`
- `incident001` se produit dans `chambre205`
- `incident001` est déclaré par `etudiantMartin`
- `incident001` est traité par `agentPierre`
- `incident001` possède `historique001a` et `historique001b`
- `incident001` possède `commentaire001`
- `etudiantMartin` a la session `sessionMartin`
- `sessionMartin` associe `etudiantMartin` à `chambre205`
- `notif001` est destinée à `agentPierre`
- `notif002` est destinée à `etudiantMartin`

Cet exemple concret illustre le parcours complet d'un incident depuis sa déclaration par un occupant jusqu'à sa prise en charge par un agent de terrain, avec tous les objets intermédiaires (bâtiment, étage, espace, équipement, notifications, historique de statuts). Il démontre la

Figure 1 - Diagramme de classe Utilisateur

cohérence du modèle de classes et la manière dont les différentes entités collaborent pour réaliser les fonctionnalités métier du système.

CHAPITRE III : CONCEPTION ET MISE EN ŒUVRE DU SYSTÈME

I. Méthodologie, normes et outils

1 Méthodes de conception

Démarche de modélisation adoptée : Méthode Agile Scrum

La méthode Agile Scrum a été retenue pour le développement de ce système de gouvernance des bâtiments. Ce choix se justifie par plusieurs facteurs inhérents à la nature du projet et au contexte universitaire.

Premièrement, la méthode Scrum permet une **flexibilité et une adaptabilité** essentielles face aux évolutions potentielles des besoins. Dans un environnement universitaire dynamique comme l'IUSJC, les exigences peuvent évoluer en fonction des retours des utilisateurs (étudiants, agents de terrain, administrateurs). L'approche itérative de Scrum permet d'intégrer ces changements de manière progressive sans remettre en cause l'ensemble du projet.

Deuxièmement, le **découpage du travail en sprints** (itérations de 2 à 4 semaines) facilite la gestion du projet dans le cadre académique imposé. Chaque sprint aboutit à un livrable fonctionnel qui peut être testé et validé, permettant ainsi une validation progressive du système.

Troisièmement, la méthode Scrum favorise la **collaboration et la communication** au sein de l'équipe de développement. Les cérémonies Scrum (daily meetings, sprint planning, sprint review, sprint retrospective) assurent une synchronisation régulière de l'équipe et permettent d'identifier rapidement les obstacles.

Enfin, l'approche Scrum privilégie la **livraison de valeur** en priorisant les fonctionnalités selon leur importance métier. Cela garantit que, même avec des contraintes de temps, les fonctionnalités critiques seront développées en priorité.

Langage de modélisation : UML (Unified Modeling Language)

L'UML a été choisi comme langage de modélisation pour sa capacité à représenter de manière standardisée et claire les différents aspects du système. L'UML offre un ensemble riche de diagrammes permettant de modéliser :

- La **vue statique** du système (diagrammes de classes, de composants, de déploiement) qui décrit la structure et l'organisation des éléments constitutifs
- La **vue dynamique** du système (diagrammes de séquence, d'activités, d'états) qui illustre les comportements et les interactions
- La **vue fonctionnelle** du système (diagrammes de cas d'utilisation) qui capture les besoins des utilisateurs

L'utilisation d'UML facilite également la communication entre les différentes parties prenantes du projet (développeurs, administrateurs, utilisateurs) grâce à un vocabulaire visuel commun et normalisé.

2 Environnement et outils de développement

Choix technologiques et justifications

L'architecture du système repose sur une approche moderne et éprouvée, combinant une application web pour l'administration et deux applications mobiles pour les utilisateurs finaux.

Frontend Web (Administration)

Pour l'interface d'administration web, le choix s'est porté sur **React.js** associé à **TypeScript**. React.js est une bibliothèque JavaScript mature et largement adoptée qui permet de construire des interfaces utilisateur réactives et performantes. L'utilisation de TypeScript apporte un typage statique qui améliore la maintenabilité du code et réduit les erreurs de développement.

Le framework **Material-UI (MUI)** sera utilisé pour les composants d'interface, garantissant une cohérence visuelle et une expérience utilisateur professionnelle. MUI offre un ensemble complet de composants prêts à l'emploi, respectant les principes du Material Design de Google.

Applications Mobiles

Deux applications mobiles distinctes seront développées avec **React Native** :

1. Une application pour les **clients/occupants** des chambres
2. Une application pour les **agents de terrain et superviseurs**

React Native a été choisi pour sa capacité à développer des applications **cross-platform** (iOS et Android) avec une base de code unique, réduisant ainsi les coûts et le temps de développement. Cette technologie permet également de partager une partie de la logique métier avec l'application web grâce à l'utilisation de JavaScript/TypeScript.

La fonctionnalité de **travail hors ligne** requise pour les agents de terrain sera implémentée grâce à des bibliothèques comme **Redux Persist** et **AsyncStorage**, permettant la synchronisation automatique des données lorsque la connexion est rétablie.

Backend

Le backend sera développé avec **Node.js** et le framework **NestJS**. NestJS a été retenu pour :

- Son **architecture modulaire** inspirée d'Angular, facilitant l'organisation du code
- Son support natif de **TypeScript**, assurant la cohérence technologique avec le frontend
- Son système d'**injection de dépendances** qui améliore la testabilité
- Ses **décorateurs** qui simplifient la création d'API RESTful
- Son intégration native avec **TypeORM** pour l'accès aux données

Base de données

PostgreSQL a été sélectionné comme système de gestion de base de données relationnelle pour :

- Sa **robustesse** et sa fiabilité éprouvées
- Sa conformité **ACID** garantissant l'intégrité des données
- Son support des **relations complexes** nécessaires à la modélisation des bâtiments, étages, espaces et équipements
- Ses capacités de **requêtes avancées** et d'optimisation
- Sa **licence open-source** réduisant les coûts

Notifications Push

L'envoi de notifications push sera assuré par **Firebase Cloud Messaging (FCM)**, solution gratuite et fiable de Google supportant iOS et Android.

Stockage de fichiers

Le stockage des images (photos des équipements défectueux) sera géré par **AWS S3** ou **Cloudinary**, solutions offrant scalabilité, fiabilité et optimisation automatique des images.

Géolocalisation Indoor

Pour la cartographie interactive des bâtiments, la bibliothèque **Leaflet.js** sera utilisée avec des **SVG personnalisés** représentant les plans des bâtiments. Cette approche permet une visualisation précise des espaces sans dépendre de solutions cartographiques externes coûteuses.

Environnement de travail

- **IDE Principal** : Visual Studio Code avec extensions pour React, TypeScript, et Node.js
- **Gestion de versions** : Git avec hébergement sur GitLab (ISI5_PROJET_GROUPE)
- **CI/CD** : GitLab CI/CD pour l'automatisation des tests et du déploiement
- **Tests** : Jest pour les tests unitaires, Cypress pour les tests end-to-end
- **Documentation API** : Swagger/OpenAPI intégré à NestJS
- **Conteneurisation** : Docker pour assurer la portabilité et faciliter le déploiement
- **Orchestration** : Docker Compose pour l'environnement de développement

3 Normes et plan de qualité

Standards de programmation

Le projet respectera des conventions de codage strictes pour garantir la lisibilité, la maintenabilité et la cohérence du code :

- **Conventions de nommage** :
  - PascalCase pour les noms de classes et de composants React
  - camelCase pour les variables et fonctions
  - UPPER_SNAKE_CASE pour les constantes
  - kebab-case pour les noms de fichiers
- **Structure du code** :
  - Organisation modulaire par fonctionnalité (feature-based)
  - Séparation claire entre logique métier, présentation et accès aux données
  - Principe de responsabilité unique (Single Responsibility Principle)
  - Injection de dépendances pour faciliter les tests
- **Documentation** :
  - Commentaires JSDoc pour toutes les fonctions publiques
  - README détaillés pour chaque module
  - Documentation Swagger automatique pour l'API

Assurance qualité logicielle

Le projet s'inspirera des normes reconnues pour garantir la qualité :

- **ISO/IEC 25010** (qualité du produit logiciel) pour évaluer :
  - La performance et l'efficacité
  - La compatibilité cross-platform
  - La facilité d'utilisation
  - La fiabilité et la disponibilité
  - La sécurité
  - La maintenabilité
  - La portabilité
- **Tests systématiques** :
  - Couverture de code minimale de 80% pour les tests unitaires
  - Tests d'intégration pour les API critiques
  - Tests end-to-end pour les parcours utilisateurs principaux
  - Tests de régression automatisés via CI/CD
- **Revues de code** :
  - Peer reviews systématiques via merge requests GitLab
  - Utilisation d'ESLint et Prettier pour l'uniformité du code
  - Analyse statique avec SonarQube

Gestion des configurations et des risques

La gestion des configurations sera assurée par :

- **Versioning sémantique** (Semantic Versioning) pour toutes les releases
- **Branches Git** structurées :
  - `main` : code en production
  - `develop` : code en développement
  - `feature/*` : développement de nouvelles fonctionnalités
  - `hotfix/*` : corrections urgentes
- **Environnements distincts** :
  - Développement local
  - Staging (pré-production)
  - Production

La gestion des risques comprendra :

- **Identification des risques** : techniques, organisationnels, sécurité
- **Évaluation** : probabilité et impact
- **Mitigation** : stratégies de réduction des risques
- **Monitoring continu** : surveillance des indicateurs clés

II. Conception générale du système

1 Identification et description des modules

Le système de gestion de la gouvernance des bâtiments de l'IUSJC est structuré en plusieurs modules fonctionnels distincts, chacun répondant à des besoins spécifiques des utilisateurs.

Module d'Authentification et Gestion des Sessions

Ce module central gère l'identification et l'autorisation de tous les utilisateurs du système. Il prend en charge trois types d'authentification distincts :

- Pour les **administrateurs** : authentification par pseudo et mot de passe avec sessions sécurisées par JWT
- Pour les **occupants** : authentification par numéro de chambre et mot de passe temporaire, avec gestion automatique des sessions liées à la durée d'occupation
- Pour les **agents/superviseurs** : authentification par login et mot de passe, avec vérification du statut actif du compte

Le module assure également la gestion du cycle de vie des sessions, notamment l'expiration automatique des sessions des occupants en fin de période d'occupation.

Module de Gestion des Incidents (Core Business)

Ce module représente le cœur fonctionnel du système. Il permet :

- La **déclaration d'incidents** par les occupants (avec photo et commentaire) ou par les agents (sans photo obligatoire)
- Le **suivi du cycle de vie** des incidents à travers différents états (Nouveau, En cours, Résolu, À remplacer)
- La **prise en charge** des incidents par les agents avec assignation automatique ou manuelle
- La **résolution** des incidents avec mise à jour du statut des équipements
- L'**historique complet** des incidents par chambre/espace

Module de Gestion des Équipements

Ce module assure le suivi de l'ensemble du patrimoine mobilier de l'institution :

- **Inventaire** de tous les équipements (frigo, table, chaise, climatiseur, etc.)
- **Affectation** des équipements aux espaces (chambres, salles, bureaux)
- **Suivi des statuts** : Bon état, À réparer, À remplacer, En maintenance
- **Historique** des changements de statut
- **Import/Export** des données d'équipements via fichiers Excel

Module de Cartographie et Gestion des Espaces

Ce module permet une représentation structurée et visuelle de l'infrastructure :

- **Hiérarchie complète** : Bâtiments → Étages → Espaces (chambres, salles, bureaux)
- **Géolocalisation indoor** avec carte interactive
- **Import de configuration** via fichiers Excel
- **Visualisation de l'état** des espaces (étiquetage si équipements défectueux)
- **Recherche et filtrage** des espaces selon différents critères

Module de Notifications

Ce module assure la communication en temps réel entre les différents acteurs :

- **Push notifications** pour les agents lors de nouveaux incidents
- **Notifications** pour les occupants lors de résolution d'incidents
- **Système de préférences** de notifications configurables
- **Historique** des notifications envoyées

Module de Reporting et Analytics

Module décisionnel offrant une vue analytique sur l'activité de maintenance :

- **Tableaux de bord** avec KPI (nombre d'incidents, temps de résolution, etc.)
- **Statistiques** par bâtiment, étage, type d'équipement
- **Identification** des espaces les plus problématiques
- **Génération de rapports** Excel exportables
- **Prédiction** des équipements à risque (machine learning)

Module d'Administration

Module réservé aux administrateurs pour la configuration et la supervision du système :

- **Gestion des utilisateurs** (agents, occupants)
- **Configuration du système** (bâtiments, équipements)
- **Modération** et supervision des incidents
- **Accès aux statistiques globales**
- **Gestion des droits et permissions**

Module de Synchronisation Hors Ligne

Module technique essentiel pour les agents mobiles :

- **Stockage local** des données critiques
- **Détection** de la connectivité
- **Synchronisation automatique** lors du retour en ligne
- **Gestion des conflits** de synchronisation
- **File d'attente** des actions en attente

2 Algorithmes clés du système

Algorithme 1 : Prédiction des équipements à risque

Cet algorithme utilise un modèle de machine learning pour identifier les équipements susceptibles de tomber en panne dans un futur proche.

```
Algorithme : predictEquipmentFailure()
Entrée : equipment (Équipement), historicalData (Historique des incidents)
Sortie : riskScore (Score de risque entre 0 et 1)

Début
    // Extraction des caractéristiques
    features = []

    // 1. Ancienneté de l'équipement
    ageInMonths = calculerDifferenceEnMois(equipment.purchaseDate, dateActuelle)
    features.add(ageInMonths)
```

```
    // 2. Nombre d'incidents passés
    totalIncidents = equipment.incidents.count()
    features.add(totalIncidents)

    // 3. Fréquence des incidents (incidents par mois)
    incidentFrequency = totalIncidents / ageInMonths
    features.add(incidentFrequency)

    // 4. Temps moyen de résolution
    avgResolutionTime = calculerMoyenne(
        equipment.incidents.map(i => i.getProcessingTime())
    )
    features.add(avgResolutionTime)

    // 5. Nombre de changements de statut "À réparer" -> "Bon état"
    repairCount = equipment.statusHistory.count(
        h => h.previousStatus == TO_REPAIR AND h.newStatus == GOOD
    )
    features.add(repairCount)

    // 6. Dernier statut de l'équipement
    currentStatusScore =
        equipment.status == GOOD ? 0 :
        equipment.status == TO_REPAIR ? 0.5 :
        equipment.status == TO_REPLACE ? 1 :
        equipment.status == IN_MAINTENANCE ? 0.3
    features.add(currentStatusScore)

    // 7. Catégorie d'équipement (encodage one-hot)
    categoryVector = encoderOneHot(equipment.category)
    features.addAll(categoryVector)

    // 8. Taux de dégradation (incidents récents vs anciens)
    recentIncidents = equipment.incidents.filter(
        i => i.reportedAt > dateActuelle.soustraireMois(3)
    ).count()
    oldIncidents = totalIncidents - recentIncidents
    degradationRate = recentIncidents > 0 ?
        recentIncidents / (oldIncidents + 1) : 0
    features.add(degradationRate)

    // 9. Espace associé (certains espaces sont plus exposés)
    spaceDefectRate = equipment.space.getEquipmentByStatus(TO_REPAIR).count()
                      / equipment.space.equipment.count()
    features.add(spaceDefectRate)

    // Normalisation des features
    normalizedFeatures = normaliser(features)

    // Prédiction avec modèle ML (Random Forest ou Neural Network)
    riskScore = modeleML.predict(normalizedFeatures)

    // Seuillage
    Si riskScore > 0.7 Alors
        createPredictiveNotification(equipment, riskScore)
    Fin Si

    Retourner riskScore
Fin
```

Algorithme 2 : Synchronisation hors ligne avec gestion des conflits

```
Algorithme : synchronizeOfflineActions()
Entrée : agent (Agent connecté)
Sortie : syncResult (Résultat de la synchronisation)

Début
    syncResult = {success: [], failed: [], conflicts: []}

    // Récupérer toutes les actions en attente
    pendingActions = SyncQueue.findAll(
        agent_id = agent.id AND status = PENDING
    ).orderBy(createdAt ASC)

    Pour chaque action dans pendingActions Faire
        Essayer
            // Vérifier si l'entité existe toujours
            entityExists = verifierExistenceEntite(action.payload)

            Si NON entityExists Alors
                // Conflit : l'entité a été supprimée
                conflict = {
                    action: action,
                    type: "ENTITY_DELETED",
                    resolution: "SKIP"
                }
                syncResult.conflicts.add(conflict)
                action.status = FAILED
                action.save()
                Continuer
            Fin Si

            // Récupérer la version serveur de l'entité
            serverVersion = recupererVersionServeur(action.payload.entity_id)
            localVersion = action.payload

            // Détection de conflit (modification concurrente)
            Si serverVersion.updatedAt > localVersion.updatedAt Alors
                // Stratégie de résolution : Last-Write-Wins
                // ou Merge selon le type d'action

                Si action.action == "UPDATE" Alors
                    // Tentative de merge automatique
                    mergedData = mergerDonnees(localVersion, serverVersion)

                    Si mergedData.hasConflicts Alors
                        conflict = {
                            action: action,
                            type: "CONCURRENT_MODIFICATION",
                            localVersion: localVersion,
                            serverVersion: serverVersion,
                            resolution: "MANUAL_REQUIRED"
                        }
                        syncResult.conflicts.add(conflict)
                        action.status = FAILED
                        action.save()
                        Continuer
                    Sinon
                        action.payload = mergedData
```

```
                Fin Si
            Fin Si
        Fin Si

        // Exécuter l'action
        Selon action.action Faire
            Cas "CREATE_INCIDENT":
                result = creerIncident(action.payload)

            Cas "TAKE_INCIDENT":
                result = prendreEnChargeIncident(
                    action.payload.incident_id,
                    agent.id
                )

            Cas "RESOLVE_INCIDENT":
                result = resoudreIncident(
                    action.payload.incident_id,
                    action.payload.equipment_status
                )

            Cas "UPDATE_EQUIPMENT":
                result = mettreAJourEquipement(action.payload)

            Défaut:
                LeverException("Action inconnue")
        Fin Selon

        // Marquer comme synchronisé
        action.status = SYNCED
        action.syncedAt = dateActuelle
        action.save()

        syncResult.success.add({
            action: action,
            result: result
        })

    Attraper exception Comme e
        // Échec de synchronisation
        action.status = FAILED
        action.errorMessage = e.message
        action.retryCount = action.retryCount + 1

        Si action.retryCount < 3 Alors
            // Réessayer plus tard
            action.status = PENDING
        Fin Si

        action.save()

        syncResult.failed.add({
            action: action,
            error: e.message
        })
    Fin Essayer
Fin Pour

// Récupérer les mises à jour du serveur
lastSyncTimestamp = agent.lastSyncTimestamp
serverUpdates = recupererMisesAJourServeur(agent.id, lastSyncTimestamp)
```

```
    // Appliquer les mises à jour localement
    appliquerMisesAJourLocales(serverUpdates)

    // Mettre à jour le timestamp de dernière synchronisation
    agent.lastSyncTimestamp = dateActuelle
    agent.save()

    Retourner syncResult
Fin

Fonction mergerDonnees(local, server)
Début
    merged = {}
    conflicts = []

    Pour chaque champ dans local.fields Faire
        Si local[champ] == server[champ] Alors
            // Pas de conflit
            merged[champ] = local[champ]
        Sinon
            // Conflit détecté
            Si champ EST_DANS champsCritiques Alors
                // Ne peut pas être résolu automatiquement
                conflicts.add({
                    field: champ,
                    localValue: local[champ],
                    serverValue: server[champ]
                })
            Sinon
                // Résolution automatique : prendre la valeur locale
                merged[champ] = local[champ]
            Fin Si
        Fin Si
    Fin Pour

    merged.hasConflicts = conflicts.length > 0
    merged.conflicts = conflicts

    Retourner merged
Fin
```

Algorithme 3 : Génération de statistiques et KPI

```
Algorithme : calculateStatistics(period, building, floor)
Entrée : period (Période), building (Bâtiment optionnel), floor (Étage optionnel)
Sortie : statistics (Objet Statistiques)

Début
    statistics = new Statistics()

    // Filtres de base
    filters = {
        reportedAt: BETWEEN period.startDate AND period.endDate
    }

    Si building != NULL Alors
        filters.building_id = building.id
```

```
Fin Si

Si floor != NULL Alors
    filters.floor_id = floor.id
Fin Si

// 1. Récupérer tous les incidents de la période
incidents = Incident.findAll(filters)

// 2. Calcul des métriques de base
statistics.totalIncidents = incidents.count()
statistics.resolvedIncidents = incidents.filter(
    i => i.status IN [RESOLVED, REQUIRES_REPLACEMENT]
).count()
statistics.pendingIncidents = incidents.filter(
    i => i.status IN [NEW, IN_PROGRESS]
).count()

// 3. Temps moyen de résolution
resolvedIncidents = incidents.filter(i => i.resolvedAt != NULL)
Si resolvedIncidents.count() > 0 Alors
    totalResolutionTime = resolvedIncidents.reduce(
        (sum, i) => sum + i.getProcessingTime(), 0
    )
    statistics.averageResolutionTime =
        totalResolutionTime / resolvedIncidents.count()
Sinon
    statistics.averageResolutionTime = 0
Fin Si

// 4. Répartition des équipements par statut
equipments = Equipment.findAll(filters)
statistics.equipmentByStatus = {}

Pour chaque status dans EquipmentStatus.values() Faire
    count = equipments.filter(e => e.status == status).count()
    statistics.equipmentByStatus[status] = count
Fin Pour

// 5. Espaces les plus défectueux
spaceIncidentCounts = {}

Pour chaque incident dans incidents Faire
    spaceId = incident.equipment.space.id
    Si spaceIncidentCounts[spaceId] == NULL Alors
        spaceIncidentCounts[spaceId] = 0
    Fin Si
    spaceIncidentCounts[spaceId] += 1
Fin Pour

// Trier et prendre le top 10
sortedSpaces = trierParValeurDecroissante(spaceIncidentCounts)
statistics.mostDefectiveSpaces = sortedSpaces.slice(0, 10).map(
    entry => Space.findById(entry.key)
)

// 6. Incidents par bâtiment
statistics.incidentsByBuilding = {}
buildings = Building.findAll()

Pour chaque building dans buildings Faire
```

```
        count = incidents.filter(
            i => i.equipment.space.floor.building.id == building.id
        ).count()
        statistics.incidentsByBuilding[building] = count
    Fin Pour

    // 7. Incidents par étage
    statistics.incidentsByFloor = {}
    floors = Floor.findAll()

    Pour chaque floor dans floors Faire
        count = incidents.filter(
            i => i.equipment.space.floor.id == floor.id
        ).count()
        statistics.incidentsByFloor[floor] = count
    Fin Pour

    // 8. Espaces sans incidents (chambres irréprochables)
    allSpaces = Space.findAll(filters)
    spacesWithIncidents = incidents.map(i => i.equipment.space.id).unique()
    statistics.spacesWithoutIncidents = allSpaces.filter(
        s => NON spacesWithIncidents.includes(s.id)
    ).count()

    // 9. Taux de résolution par agent
    statistics.agentPerformance = {}
    agents = Agent.findAll(isActive = TRUE)

    Pour chaque agent dans agents Faire
        agentIncidents = incidents.filter(i => i.assignedTo.id == agent.id)
        resolved = agentIncidents.filter(
            i => i.status IN [RESOLVED, REQUIRES_REPLACEMENT]
        ).count()
        total = agentIncidents.count()

        statistics.agentPerformance[agent] = {
            totalHandled: total,
            resolved: resolved,
            resolutionRate: total > 0 ? (resolved / total) * 100 : 0,
            averageTime: calculerMoyenne(
                agentIncidents.map(i => i.getProcessingTime())
            )
        }
    Fin Pour

    // 10. Tendances temporelles
    statistics.dailyTrend = calculerTendanceJournaliere(incidents, period)
    statistics.categoryDistribution = calculerRepartitionCategories(incidents)

    Retourner statistics
Fin

Fonction calculerTendanceJournaliere(incidents, period)
Début
    trend = {}
    currentDate = period.startDate

    TantQue currentDate <= period.endDate Faire
        dayIncidents = incidents.filter(
            i => i.reportedAt.date == currentDate
        )
```

```
        trend[currentDate] = dayIncidents.count()
        currentDate = currentDate.ajouterJour(1)
    Fin TantQue

    Retourner trend
Fin
```

3 Thème graphique et maquettes

Définition du thème visuel

Le système de gestion de la gouvernance des bâtiments de l'IUSJC adopte un thème visuel moderne, professionnel et intuitif, en cohérence avec l'identité de l'institution.

Logo et identité visuelle

Le logo du système reprend les couleurs institutionnelles de l'IUSJC tout en apportant une touche de modernité. Il intègre des éléments symbolisant :

- Un bâtiment stylisé représentant l'infrastructure
- Un symbole de maintenance (clé à molette ou engrenage)
- Une connexion (réseau ou signal) symbolisant la dimension numérique

Palette de couleurs

La palette de couleurs a été soigneusement sélectionnée pour assurer lisibilité, accessibilité et cohérence visuelle :

Couleurs principales :

- **Bleu primaire** (`#2563EB`) : Couleur institutionnelle, inspire confiance et professionnalisme
- **Bleu secondaire** (`#3B82F6`) : Variante plus claire pour les éléments interactifs
- **Gris foncé** (`#1F2937`) : Textes principaux et éléments structurels

Couleurs d'accentuation :

- **Vert succès** (`#10B981`) : Confirmation d'actions, équipements en bon état
- **Orange avertissement** (`#F59E0B`) : Équipements à réparer, alertes modérées
- **Rouge erreur** (`#EF4444`) : Équipements à remplacer, erreurs critiques
- **Bleu info** (`#3B82F6`) : Informations et notifications

Couleurs neutres :

- **Blanc** (`#FFFFFF`) : Arrière-plans principaux
- **Gris très clair** (`#F3F4F6`) : Arrière-plans secondaires

- **Gris clair** (`#E5E7EB`) : Bordures et séparateurs
- **Gris moyen** (`#9CA3AF`) : Textes secondaires

Typographie

- **Police principale** : Inter (sans-serif moderne et lisible)
  - Titres : Inter Bold (24-32px)
  - Sous-titres : Inter Semibold (18-20px)
  - Texte courant : Inter Regular (14-16px)
  - Petits textes : Inter Regular (12px)
- **Police secondaire** : Roboto Mono (pour codes, identifiants)

Principes de design UI/UX

1. **Clarté et simplicité** : Interfaces épurées, hiérarchie visuelle claire
2. **Accessibilité** : Contraste suffisant (WCAG AA), tailles de texte lisibles
3. **Cohérence** : Composants réutilisables, comportements prévisibles
4. **Feedback visuel** : États interactifs clairs (hover, active, disabled)
5. **Responsive design** : Adaptation fluide aux différentes tailles d'écran
6. **Mobile-first** : Optimisation prioritaire pour les appareils mobiles

Maquettes principales

En raison des limitations de formatage, je vais décrire textuellement les maquettes principales qui seront créées avec Figma :

a. Application Mobile Occupant

Écran de connexion

- Logo centré en haut
- Champ "Numéro de chambre" avec icône de porte
- Champ "Mot de passe" avec icône de cadenas et toggle visibilité
- Bouton "Se connecter" en plein largeur
- Message d'erreur en rouge si échec
- Design minimaliste sur fond blanc avec accents bleus

Écran d'accueil/Dashboard

- En-tête avec photo de profil, nom, numéro de chambre
- Statut de la session (date de fin) avec barre de progression
- Bouton d'action principal : "Déclarer un incident" (grande card bleue)
- Section "Mes incidents" avec liste des incidents déclarés
- Pour chaque incident : miniature photo, équipement, statut (badge coloré), date
- Menu de navigation en bas : Accueil, Incidents, Profil

Écran de déclaration d'incident

- Titre "Déclarer un incident"
- Sélection de l'équipement (dropdown avec recherche)
- Zone de capture photo (preview si photo prise)
- Bouton "Prendre une photo" avec icône appareil photo
- Champ texte "Commentaire" (multiligne)
- Bouton "Envoyer" en bas
- Feedback visuel lors de l'envoi (spinner + message)

Écran d'historique des incidents

- Filtres en haut : Tous, En cours, Résolus
- Liste chronologique inversée
- Cards incidents avec :
    - Photo miniature
    - Équipement et emplacement
    - Statut avec badge coloré
    - Date de déclaration
    - Agent assigné (si applicable)
    - Timeline des étapes (déclaré → pris en charge → résolu)

b. Application Mobile Agent/Superviseur

Écran de connexion

- Logo + titre "Espace Agent"
- Champs login et mot de passe
- Bouton "Se connecter"
- Indicateur de mode hors ligne si pas de connexion

Dashboard Agent

- En-tête avec photo, nom, badge superviseur (si applicable)
- Statistiques en cards :
    - Incidents en attente (nombre avec icône alerte)
    - Mes incidents en cours (nombre avec icône outils)
    - Résolus aujourd'hui (nombre avec icône check)
- Bouton "Déclarer un incident" (salle de classe/bureau)
- Liste des incidents récents avec filtres
- Menu inférieur : Accueil, Incidents, Rapports, Profil

Écran liste des incidents

- Barre de recherche en haut

- Filtres : Statut, Bâtiment, Étage, Date
- Cards incidents :
    - Photo (si disponible)
    - Numéro de chambre/salle
    - Équipement défectueux
    - Statut (badge)
    - Priorité (si système de priorisation)
    - Bouton "Prendre en charge" ou "Voir détails"
- Indicateur hors ligne avec liste des actions en attente de sync

Écran détail incident

- Photo en plein écran (zoomable)
- Informations structurées :
    - Localisation (bâtiment, étage, numéro)
    - Équipement concerné
    - Déclaré par et le (date/heure)
    - Commentaire du déclarant
    - Agent assigné (si pris en charge)
- Boutons d'action selon statut :
    - "Prendre en charge" (si nouveau)
    - "Marquer comme résolu" (si en cours)
    - Options de résolution : Bon état / À remplacer
- Champ de commentaire agent

Écran rapports (Superviseur)

- Sélection de période (dropdown prédéfinis + personnalisé)
- Sélection de filtres (bâtiment, étage)
- Visualisations :
    - Graphique en barres : incidents par statut
    - Graphique linéaire : tendance temporelle
    - Top 5 espaces défectueux
    - Performance des agents
- Bouton "Générer rapport Excel"
- Bouton "Envoyer par email"

c. Application Web Administrateur

Page de connexion

- Design centré sur fond avec image de l'université
- Card de connexion avec logo
- Champs pseudo et mot de passe
- Bouton "Se connecter"

- Footer avec liens (Contact, Aide)

Dashboard principal

- Sidebar navigation à gauche :
  - Dashboard
  - Agents
  - Occupants
  - Bâtiments & Espaces
  - Équipements
  - Incidents
  - Statistiques
  - Prédictions
  - Paramètres
- En-tête avec recherche globale, notifications, profil
- Zone principale avec :
  - KPI cards en haut (4 métriques clés)
  - Graphiques (incidents par jour, par bâtiment)
  - Tableau des derniers incidents
  - Alertes et notifications importantes

Page gestion des agents

- Tableau avec :
  - Photo, Nom, Email, Téléphone, Statut (Actif/Inactif), Rôle (Agent/Superviseur)
  - Actions : Activer/Désactiver, Modifier, Voir détails
- Bouton "Ajouter un agent" (ouvre modal)
- Filtres et recherche
- Pagination

Page configuration des espaces

- Arborescence visuelle : Bâtiments → Étages → Espaces
- Vue en grille ou liste (toggle)
- Pour chaque espace : numéro, type, occupant (si applicable), nombre d'équipements, statut
- Bouton "Importer fichier Excel"
- Carte interactive indoor (vue plan) :
  - Plans SVG des bâtiments
  - Espaces cliquables
  - Code couleur selon statut
  - Zoom et navigation

Page gestion des équipements

- Tableau avec :
    - Nom, Catégorie, Numéro de série, Emplacement, Statut
    - Indicateur visuel (pastille colorée) selon statut
    - Actions : Modifier, Voir historique, Supprimer
- Filtres avancés : Statut, Catégorie, Emplacement, Date d'achat
- Bouton "Importer équipements Excel"
- Modal détails équipement avec :
    - Toutes les infos
    - Historique des statuts (timeline)
    - Liste des incidents liés
    - Score de prédiction de panne

Page statistiques avancées

- Sélecteur de période et filtres en haut
- Onglets :
    - Vue globale
    - Par bâtiment
    - Par étage
    - Par catégorie d'équipement
    - Par agent
- Graphiques variés :
    - Diagrammes en barres
    - Courbes d'évolution
    - Camemberts (répartition)
    - Heatmap (espaces défectueux)
- Tableaux de données détaillées
- Export (PDF, Excel, CSV)

Page prédictions ML

- Liste des équipements à risque
- Pour chaque équipement :
    - Score de risque (gauge visuelle)
    - Emplacement
    - Justification (facteurs de risque)
    - Recommandation d'action
- Filtres par seuil de risque et emplacement
- Possibilité de créer une maintenance préventive

Cette conception détaillée fournit un cadre complet pour l'implémentation du système. Les diagrammes UML modélisent rigoureusement la structure et le comportement du système, les algorithmes clés assurent des fonctionnalités critiques comme la prédiction et la synchronisation, et le thème graphique garantit une expérience utilisateur cohérente et professionnelle.

CONCLUSION GÉNÉRALE

Au terme de ce projet transversal, le développement du **système de gestion intelligente de la maintenance des infrastructures de l'IUSJC** nous permettra de mobiliser et d'approfondir l'ensemble des compétences techniques, méthodologiques et organisationnelles acquises durant notre formation en **ingénierie informatique et systèmes d'information**. Ce projet représente bien plus qu'un simple exercice académique : il constitue une réponse concrète à une problématique réelle de notre institution et démontre le potentiel transformateur des technologies de l'information dans l'optimisation des processus opérationnels du secteur éducatif.

L'objectif principal de ce projet est de concevoir et de développer un **écosystème numérique intégré** facilitant la gestion de la maintenance des espaces et équipements du site d'Eyang. Pour atteindre cet objectif ambitieux, nous mettrons en place une architecture complète articulée autour de trois composantes interconnectées : une **application mobile intuitive pour les occupants** simplifiant le signalement des incidents, une **application mobile robuste pour les agents de terrain et superviseurs** optimisant la coordination des interventions et le suivi en temps réel (même en mode hors ligne), et une **plateforme web d'administration puissante** offrant des outils de configuration, de cartographie interactive, d'analyse statistique et de prédiction des besoins de maintenance.

Les choix technologiques effectués se sont révélés particulièrement pertinents pour répondre aux exigences du projet. L'utilisation de **Flutter** pour le développement mobile à travers des interfaces natives performantes et cohérentes pour Android et iOS à partir d'une base de code unique, optimisant ainsi les efforts de développement et de maintenance. Le framework **NestJS** pour le backend qui offre une structure modulaire, scalable et maintenable, facilitant l'implémentation de la logique métier complexe et l'intégration des différents services. La base de données **MongoDB** pour apporter la flexibilité nécessaire afin de gérer des structures de données hétérogènes tout en garantissant d'excellentes performances en lecture/écriture. L'intégration de **technologies d'intelligence artificielle** pour la prédiction des incidents qui rajoute une dimension anticipative précieuse au système. Enfin, la mise en place d'un **pipeline CI/CD** en vue d’assurer la qualité du code et faciliter les déploiements continus.

Ce projet nous permet également d'expérimenter concrètement toutes les phases du **cycle de vie d'un projet logiciel d'envergure** : de l'analyse approfondie des besoins utilisateurs à travers des entretiens et observations terrain, à la conception architecturale et la modélisation UML (diagrammes de classes, d'état, de cas d'utilisation, de flux), en passant par le développement itératif selon la méthodologie **Scrum**, les tests rigoureux (unitaires, d'intégration, d'acceptation), jusqu'au déploiement automatisé et à la maintenance évolutive. Cette expérience complète nous a permis de comprendre les enjeux, les défis et les bonnes pratiques de la conduite d'un projet informatique professionnel.

Sur le plan des compétences techniques, ce projet vient considérablement nous permettre renforcer notre maîtrise des technologies modernes de développement. C’est l’occasion de

consolider nos connaissances en **développement mobile cross-platform**, en **architecture API REST**, en **gestion d'états complexes**, en **synchronisation de données hors ligne**, en **géolocalisation indoor**, en **analyses de données et visualisation**, en **machine learning appliqué**, en **sécurisation des applications** (authentification JWT, chiffrement, protection HTTPS), et en **DevOps** (conteneurisation, orchestration, automatisation des déploiements). Ces compétences techniques avancées constituent un atout majeur pour notre future carrière d'ingénieur.

Au-delà des aspects purement techniques, ce projet développe également des **compétences transversales essentielles**. Le travail en équipe, avec une répartition claire des rôles et des responsabilités, nous apprend l'importance de la communication, de la coordination et de la gestion des interdépendances. La documentation rigoureuse de l'architecture, du code et des processus nous sensibilise à la nécessité de la transmission du savoir et de la pérennité des projets. La gestion des contraintes temporelles et des priorités nous formera à la prise de décision sous pression. Enfin, la présentation régulière de l'avancement du projet aux parties prenantes (enseignants, utilisateurs potentiels) renforcera nos capacités de communication orale et de vulgarisation technique.

En définitive, ce projet transversal constitue une **étape décisive et structurante** dans notre parcours de formation d'ingénieur en informatique. Il offre l'opportunité exceptionnelle de concevoir, développer et déployer une solution complète répondant à un besoin réel et concret, en mobilisant l'ensemble des connaissances théoriques et pratiques acquises durant nos années d'études. L'expérience de la transformation numérique appliquée à la gestion des infrastructures éducatives nous permet donc de comprendre concrètement comment les technologies de l'information peuvent créer de la valeur pour les organisations et améliorer significativement leur efficacité opérationnelle.