ANNÉE UNIVERSITAIRE
2024–2025

ensi | ÉCOLE DES NOUVELLES SCIENCES & INGÉNIERIE
RECONNUE PAR L'ÉTAT

RAPPORT DE PROJET DE FIN DE MODULE

FRONT END

CIVIX – APPLICATION DE SIGNALEMENT ET SUIVI DES ABUS DE POUVOIR DANS LES SERVICES PUBLICS

Réalisé par :

- Asmae Shabi – Développement Front-end
- Laila Ben Ali – Responsable DevOps
- Aymane Iben Jellal – Développement Back-end

ENCADRÉ PAR :

- LOTFI EL ACHAK

I - Présentation Générale du Projet :

CIVIX est une application web citoyenne ayant pour objectif de permettre aux usagers de signaler des abus de pouvoir dans les services publics marocains : hôpitaux, police, mairies, etc. Le citoyen peut soumettre une plainte en ligne, suivre son état, joindre des preuves, et consulter des statistiques publiques. L'administration, quant à elle, peut analyser, classer, traiter, et transférer les signalements.

Objectif : Offrir une plateforme **sécurisée, centralisée et transparente** pour renforcer la relation entre citoyens et institutions publiques.

II - Contexte et Justification :

Le projet est né d'un besoin réel : de nombreux abus administratifs restent impunis ou ignorés à cause de l'absence de mécanismes de signalement accessibles et transparents. L'objectif est donc de :

- Donner la parole aux citoyens.
- Accélérer le traitement des abus signalés.
- Analyser les tendances pour améliorer les services publics.

III - Objectifs principaux :

- Créer une plateforme sécurisée pour **soumettre, traiter et suivre** des signalements.
- Fournir un espace administrateur pour **gérer les cas** et **produire des statistiques**.
- Promouvoir la **transparence et la responsabilité** dans les services publics.

IV- Architecture Technique :

Élément	Technologie
Backend	FastAPI (Python)
Frontend	React.js + Tailwind + Vite
Authentification	MYSL
Déploiement	Docker
CI/CD	GitHub Actions

Architecture Monolithique RESTful :

L'application expose une API FastAPI sécurisée par JWT. Elle communique avec une base MySQL conteneurisée par Docker. Le frontend React consomme l'API via Axios.

V - Fonctionnalités détaillées :

Côté Citoyen

- Création de compte ou signalement anonyme.
- Formulaire de signalement (type, description, localisation, gravité).
- Suivi en temps réel de l'état du dossier.
- Historique personnel des signalements.

Côté Administrateur

- Visualisation complète des signalements.
- Filtres par catégorie, gravité, ville, date.
- Réponse directe au citoyen.
- Classement et modification du statut (nouveau, en cours, résolu).
- Génération de statistiques anonymisées.

Côté Public

- Accès à des données agrégées : abus les plus fréquents, statistiques par région, taux de résolution, etc.

VI - Le dictionnaire de données :

Variable	Description	Type
admin.id	Clé primaire	INT
admin.email	Email unique de l'admin	VARCHAR(255)
admin.password_hash	Hash du mot de passe	VARCHAR(255)
citizen.id	Clé primaire	INT
citizen.email	Email unique du citoyen	VARCHAR(255)
citizen.password_hash	Hash du mot de passe	VARCHAR(255)
citizen.numero_telephone	Téléphone (optionnel)	VARCHAR(20)
signalements.id	Clé primaire	INT
signalements.citizen_id	FK → citizen.id	INT
signalements.titre	Sujet du signalement	VARCHAR(255)
signalements.localisation	Adresse ou quartier	VARCHAR(255)

signalements.ville	Ville concernée	VARCHAR(100)
signalements.description	Détail du problème	TEXT
signalements.commentaire	Observation supplémentaire	TEXT
signalements.categorie	police, hopital, admin	ENUM
signalements.gravite	mineur, majeur, urgent	ENUM
signalements.status	nouveau, en_cours, résolu	ENUM
signalements.created_at	Date de création	TIMESTAMP
signalements.updated_at	Dernière mise à jour	TIMESTAMP

Index sur ville et catégorie pour accélérer les recherches.

VII - Diagramme de cas d'utilisation UML :

Diagramme de classes :

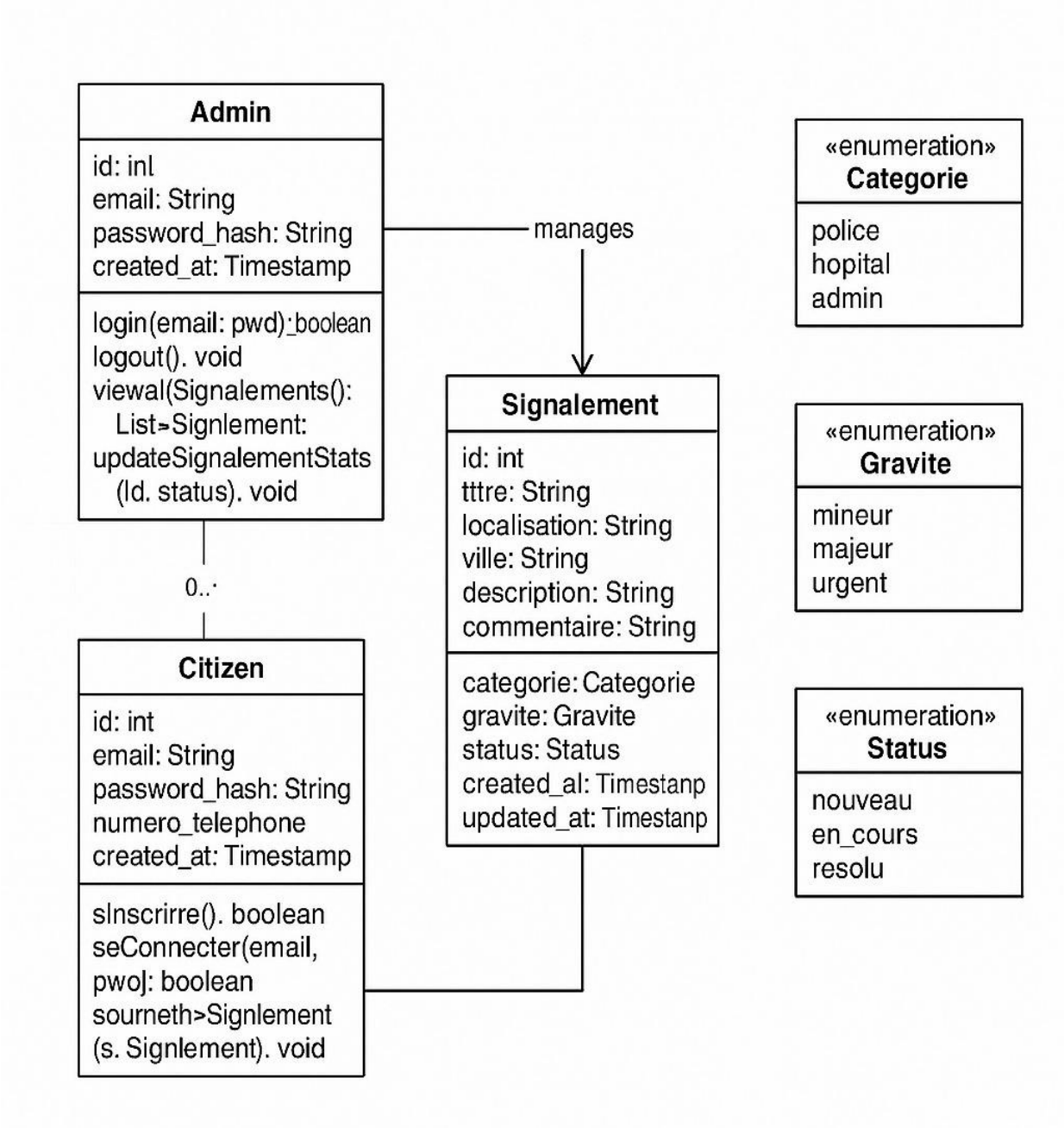
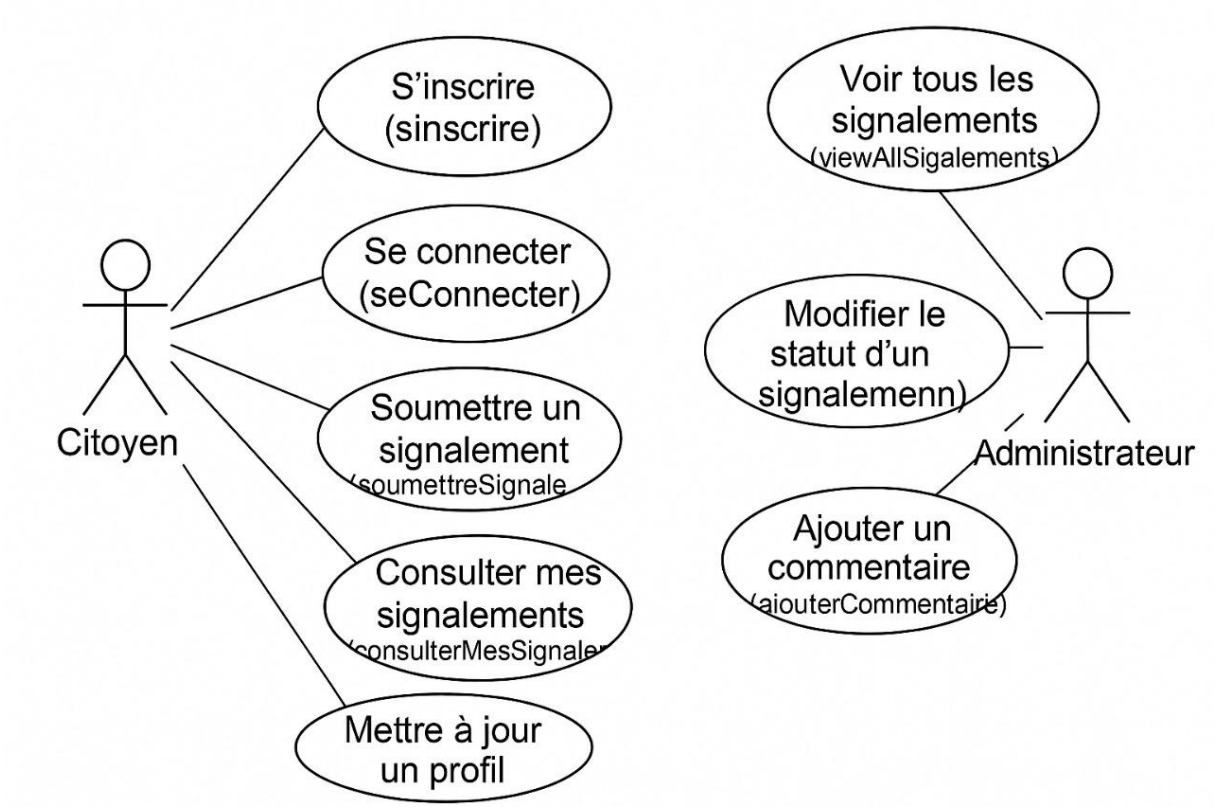


Diagramme de cas d'utilisation :



VIII - Exemple de scénario réel :

Un citoyen se connecte, sélectionne la catégorie "hôpital", remplit un formulaire décrivant une maltraitance verbale subie. Il choisit "urgent" et précise l'hôpital et la date. L'administrateur consulte ce signalement, le classe comme prioritaire, et le transfère à la direction régionale de la santé. Le citoyen est notifié via l'application que son dossier est "en cours". Finalement, le signalement est marqué "résolu".

IX - Utilisation de Docker et Conteneurisation :

Objectif :

L’usage de Docker dans ce projet vise à simplifier le déploiement, garantir un environnement homogène et assurer l'isolation de chaque service. Grâce à Docker, tous les composants de l'application (backend, frontend, base de données) sont conteneurisés et exécutés simultanément via un seul fichier docker-compose.yml.

Conteneurs utilisés :

L'application Civix s'appuie sur **trois conteneurs** :

- prj-frontend : pour l’interface utilisateur React.js (port 3000),
- prj-backend : pour l’API FastAPI (port 8000),
- mysql : pour la base de données relationnelle (port 3306).

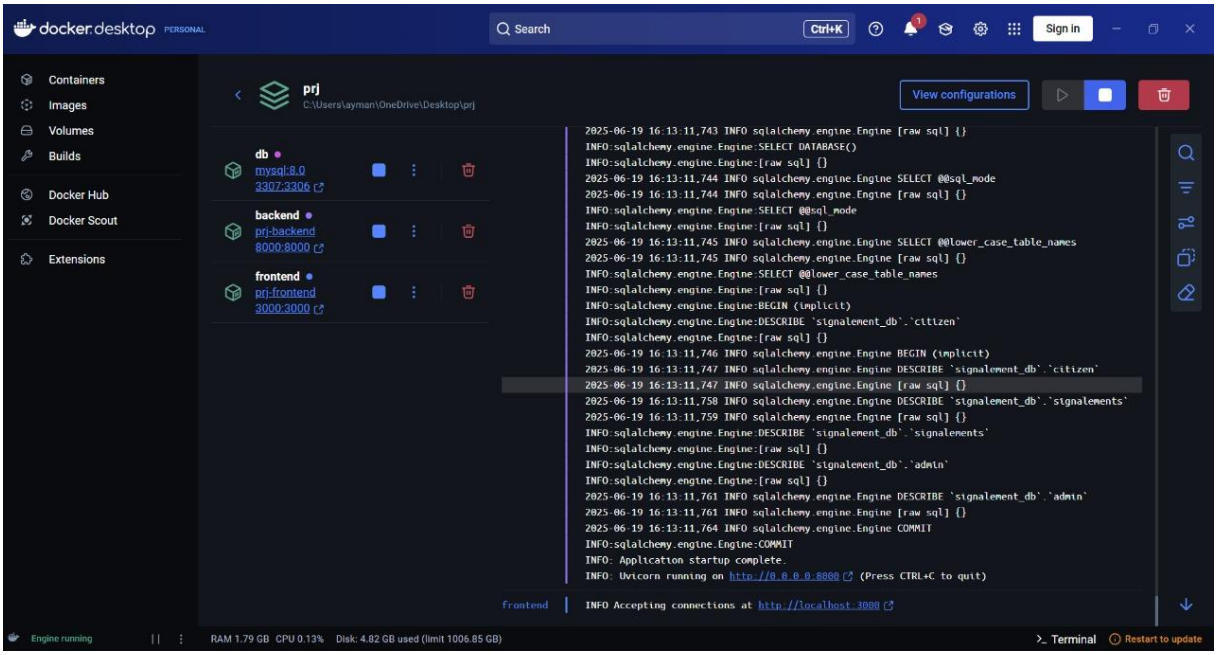


Figure 1:Vue des conteneurs actifs dans Docker Desktop

Cette interface Docker montre que tous les services sont démarrés avec succès. Le backend est lié à la base de données mysql et communique en interne avec le frontend grâce au réseau Docker partagé.

Images Docker utilisées :

Chaque conteneur est basé sur une image spécifique :

- mysql:8.0 (officielle de MySQL),
- prj-backend (image personnalisée construite depuis le Dockerfile du backend),
- prj-frontend (image personnalisée React/Vite).

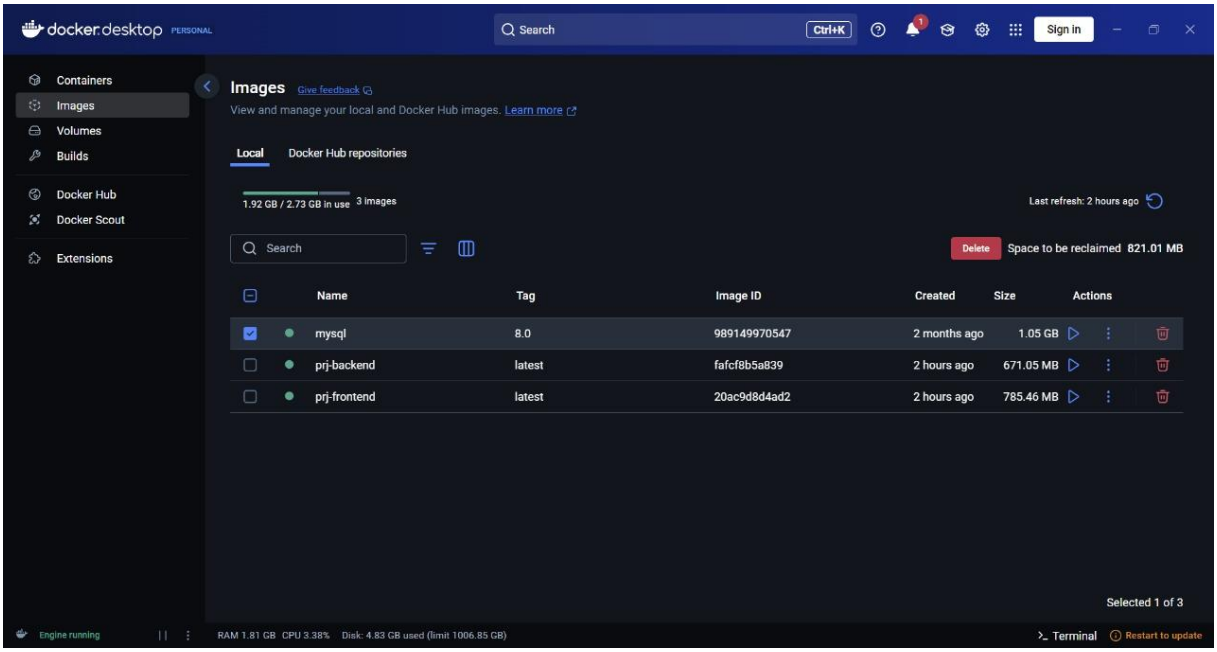


Figure 2:Liste des images Docker utilisées

Cette interface montre la taille, l'identifiant et l'état de chaque image, garantissant leur traçabilité.

Orchestration avec Docker Compose :

Tous les services sont définis dans un fichier **docker-compose.yml** placé à la racine du projet. Il permet de :

- Lancer tous les services via docker-compose up,
- Définir les ports, les noms de conteneurs, les réseaux partagés,
- Réaliser un déploiement rapide, cohérent et reproductible.

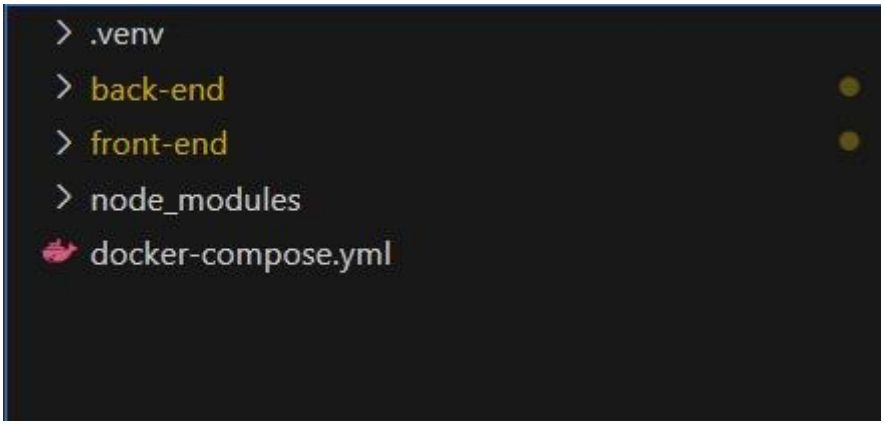


Figure 3:Arborescence du projet avec docker-compose.yml

IX-Analyse des Défis Rencontrés et Solutions Apportées :

Problème 1 – Communication entre Frontend et Backend en mode Docker

Contexte : Lors de la conteneurisation du projet, le frontend React.js ne parvenait pas à appeler l'API FastAPI. Le message d'erreur indiquait que le serveur était injoignable (fetch failed).

Analyse : Le problème venait de l'utilisation de localhost dans les appels Axios. Or, dans Docker, chaque service a sa propre IP et localhost ne fonctionne pas entre conteneurs.

Solution mise en place :

- Création d'un fichier docker-compose.yml pour connecter tous les services dans un réseau commun.
- Remplacement de http://localhost:8000 par http://backend:8000 dans les appels frontend (axios).

Problème 2 – Protéger les routes sensibles côté backend

Contexte : Au début, toutes les routes étaient accessibles à tout utilisateur, même les routes réservées aux administrateurs (suppression, mise à jour des statuts...).

Solution mise en place :

- Intégration de JWT avec un champ role (admin ou citizen).
- Création d'un décorateur @admin_required dans FastAPI pour bloquer l'accès aux routes protégées.

Résultat : séparation claire des permissions → meilleure sécurité.

Problème 3 – Structuration du backend : tout était dans main.py

Contexte : Au départ, toute la logique (routes, modèles, base de données) était regroupée dans un seul fichier main.py, ce qui rendait le projet difficile à maintenir.

Solution mise en place :

- Réorganisation du projet en modules :
  - api/ pour les routes
  - models/ pour les modèles SQLAlchemy
  - crud/ pour les opérations métier
  - core/ pour la configuration

Résultat : code plus lisible, plus modulaire, facile à faire évoluer.

Problème 4 – Absence de données initiales dans la base

Contexte : À chaque redéploiement, la base était vide. Il fallait recréer manuellement des utilisateurs et signalements pour tester.

Solution mise en place :

- Création de fichiers data.sql et seed_data.py
- Automatisation de l'initialisation de la base lors du démarrage

Résultat : base prête à l'emploi pour les démonstrations et les tests.

Problème 5 – Vérification sécurisée des mots de passe

Contexte : L'inscription acceptait des mots de passe trop simples (ex: 12345678), ce qui représentait une faille de sécurité potentielle.

Solution mise en place :

- Ajout d'un fichier test_password_verification.py
- Création d'une fonction de validation stricte (longueur, majuscule, chiffre, caractère spécial)

Résultat : mots de passe robustes, meilleure protection des comptes utilisateurs.

Problème 6 : Validations de formulaires côté frontend

Contexte : Les utilisateurs soumettaient parfois des formulaires incomplets ou mal remplis.
Cause : Absence de validation HTML ou JavaScript.

Solution :

- Mise en place de **validations React** avec gestion d'erreurs (required, minLength, regex, etc.).
- Retour visuel immédiat (champ rouge, message d'erreur) pour guider l'utilisateur.

X-Recommandations pour amélioration future :

- Intégrer un système de **notification mail/SMS**.
- Ajouter une **modération automatique** par IA (filtrage des propos offensants).
- Créer une **version mobile** React Native.
- Tableau de bord **analytique avec des graphiques** avancés pour les autorités.

XI-Conclusion :

Le projet **Civix** s'inscrit dans une démarche à la fois technique et citoyenne, visant à renforcer la transparence dans les services publics par le biais d'une solution numérique moderne. Grâce à l'intégration de technologies telles que **FastAPI**, **React.js**, **MySQL**, **Docker** et **GitHub Actions**, nous avons pu concevoir une application complète, sécurisée et structurée selon les bonnes pratiques du développement web.

Tout au long du processus, plusieurs défis ont été rencontrés, notamment liés à la communication entre services, la gestion des rôles utilisateurs et la validation des données. Ces difficultés ont été surmontées par une approche rigoureuse, une bonne organisation du code et une capacité d'adaptation progressive.

Ce projet représente une véritable expérience professionnelle, combinant réflexion métier, conception technique et esprit d'initiative. Il constitue également une base solide pour de futures évolutions, telles que l'intégration de statistiques avancées ou le déploiement sur mobile.