CAHIER DES CHARGES - PALIER 1 : BASE FONCTIONNELLE DE L'APPLICATION DE GESTION DES OBJETS DÉFECTUEUX

---

1. Présentation du Projet

Dans un environnement industriel, la **gestion des objets défectueux** est essentielle pour assurer un contrôle qualité rigoureux et garantir la fiabilité de la production. Ce projet vise à développer une application web permettant aux opérateurs de **recenser, suivre et gérer les objets défectueux** de manière efficace et sécurisée.

L'application repose sur **Django** pour le backend et **PostgreSQL** pour la base de données, offrant une structure modulaire et performante. L'interface utilisateur, développée avec **Django Templates et Bootstrap/Tailwind CSS**, assurera une expérience fluide et intuitive.

1.1. Vue d'ensemble du projet

Le projet est organisé en **trois paliers** de développement progressifs :

Palier	Objectif principal	Fonctionnalités clés
Palier 1 : Base fonctionnelle	Développer un **CRUD sécurisé et optimisé** pour gérer les objets défectueux	- Gestion des objets (ajout, modification, consultation, suppression sécurisée) - Sécurisation des données et validation des entrées - Optimisation des requêtes SQL (indexation, pagination, mise en cache) - Interface utilisateur simple et ergonomique
Palier 2 : Authentification et gestion des utilisateurs	Implémenter un **système de gestion des accès et des rôles**	- Gestion des utilisateurs et rôles (opérateurs, administrateurs) - Sécurisation avancée et permissions d'accès - Suivi des actions et logs d'activité
Palier 3 : Intégration de l'IA pour la détection automatique	Automatiser la détection des objets défectueux via un **modèle de vision par ordinateur**	- Upload et analyse d'images - Intégration d'un modèle IA de classification - Affichage des résultats et recommandations

Ce **premier palier** se concentre sur la mise en place des **fonctionnalités fondamentales**, garantissant une base solide et évolutive.

---

2. Périmètre Fonctionnel

2.1. Gestion des objets défectueux

- **Ajout d’un objet** avec les informations essentielles (nom, date d’inspection, statut, description).
- **Modification** des informations d’un objet enregistré.
- **Affichage d’une liste des objets** avec pagination et tri.
- **Suppression sécurisée** avec confirmation pour éviter les erreurs, et mise en place d’une suppression "soft" permettant la restauration des objets supprimés.
- **Filtrage et recherche** selon plusieurs critères (nom, statut, date d’inspection).

2.2. Sécurisation et validation des données

- **Utilisation de Django Forms** pour la validation des entrées et éviter les erreurs utilisateur.
- **Protection contre les attaques XSS et CSRF**, avec un filtrage des données entrantes.
- **Mise en place de requêtes préparées** pour éviter les injections SQL.
- **Gestion des erreurs et messages d’alerte** en cas de saisie invalide.

2.3. Optimisation des performances

- **Indexation des colonnes critiques** (nom_produit, status, date_inspection) pour optimiser la rapidité des requêtes.
- **Pagination des résultats** afin de limiter la charge sur le serveur et améliorer l’expérience utilisateur.
- **Mise en cache des requêtes fréquentes** pour réduire le temps de chargement.
- **Lazy loading des objets** pour ne charger que les informations essentielles au départ.

2.4. Interface utilisateur

- **Affichage clair et dynamique** des objets sous forme de tableau, avec colonnes triables.
- **Formulaire intuitif et ergonomique** pour ajouter et modifier les objets.
- **Système de notifications et messages d’alerte** en cas de suppression ou d’erreur.
- **Interface responsive** pour une compatibilité avec PC, tablettes et mobiles.

---

3. Guide de Départ : Plan de Développement

Le développement du **Palier 1** suivra une **chronologie claire** pour assurer une implémentation progressive et structurée.

3.1. Installation et Configuration de l'Environnement

- Installation de **Python** et configuration de l'environnement virtuel.
- Installation de **Django** et des dépendances nécessaires.
- Installation et configuration de **PostgreSQL**.

3.2. Initialisation du Projet Django

- Création du projet Django et de l'application principale.
- Configuration de la base de données PostgreSQL dans settings.py.
- Génération et application des migrations initiales.

3.3. Développement du Modèle de Données

- Définition du modèle **ObjetDefectueux** en Django ORM.
- Ajout des contraintes et de l'indexation des colonnes critiques.
- Création des migrations et mise en place de la base de données.

3.4. Implémentation des Fonctionnalités CRUD

- Création des **vues Django** pour l'ajout, la modification et la suppression des objets.
- Développement des **formulaires sécurisés** pour la gestion des entrées utilisateur.
- Mise en place des **routes URL** pour chaque fonctionnalité.

3.5. Optimisation des Requêtes et Sécurisation

- Validation des entrées utilisateur avec **Django Forms**.
- Mise en place des **requêtes préparées** et prévention des injections SQL.
- Ajout de **l'indexation** et optimisation des requêtes ORM.
- Activation de la **pagination** pour améliorer les performances.

3.6. Développement de l'Interface Utilisateur

- Création des **templates HTML** pour afficher et gérer les objets.
- Mise en place d'un **tableau dynamique** avec tri et filtres.
- Intégration de **Bootstrap ou Tailwind CSS** pour le design.

3.7. Tests et Validation

- Tests unitaires sur le **modèle et les vues Django**.
- Vérification de la **pagination et des performances des requêtes**.

- Test des **scénarios de suppression et restauration**.
- Vérification des **protections CSRF et XSS**.

3.8. Lancement et Déploiement

- Exécution du serveur Django et validation des fonctionnalités.
- Préparation du projet pour une présentation.

---

4. Règles du Projet Bootcamp

- **Projet individuel :** Chaque participant doit développer son propre projet en suivant les spécifications définies.
- **Compréhension du code :** Vous devez comprendre chaque ligne de votre code. Il ne suffit pas de copier du code, vous devez savoir expliquer son fonctionnement.
- **Utilisation des IA Assistantes :** L'utilisation d'outils d'intelligence artificielle (ChatGPT, Copilot, etc.) est autorisée, mais leur usage doit être réfléchi et justifié. Vous devez adapter et comprendre les suggestions avant de les intégrer.
- **Bonne pratique et organisation :**
    - Code propre et structuré (respect des conventions Django).
    - Bonne gestion des fichiers et modules pour un projet modulaire et lisible.
    - Documentation minimale des fonctionnalités et du code.

5. Conclusion

Le Palier 1 constitue le socle de l'application en mettant en place une gestion robuste, optimisée et sécurisée des objets défectueux. Il assure une base fonctionnelle stable permettant l'ajout ultérieur de la gestion des utilisateurs et de l'intelligence artificielle pour l'automatisation.

Avec cette approche, l'application sera performante dès ses premières versions, tout en étant évolutive pour accueillir des fonctionnalités plus avancées dans les prochaines phases de développement.