CAHIER DES CHARGES

I. Présentation du Projet

Nom du projet : Gestionnaire de Parc Informatique avec Maintenance Prédictive et API d'Intégration

Client : IDATECH

Version : 1.0

Date de début : 21 Novembre 2024

Date de livraison estimée : 04 Janvier 2025

Chef de projet : Aka Beyra Jeans Didier Stanislas

Technologies utilisées :

- Backend : **Laravel**
- Frontend : **Laravel Blade (HTML, CSS, JavaScript)**
- Base de données : **MySQL**
- Maintenance prédictive : **API d'analyse de données externe**
- Authentification : **JWT ou OAuth**
- Outils de suivi des performances : **Chart.js ou Highcharts ou ApexCharts**

II. Contexte et Objectifs du Projet

Le but du projet est de créer une **application web** permettant à une entreprise de gérer son parc informatique de manière optimisée, tout en offrant une fonctionnalité de maintenance prédictive. Cette solution permettra aux administrateurs et techniciens de gérer efficacement les équipements, de suivre les incidents techniques, et de recevoir des alertes de maintenance grâce à l'analyse de données.

Objectifs principaux :

- **Gestion des équipements** : Enregistrement des équipements, suivi de leur statut, et gestion de l'historique des interventions.
- **Suivi des incidents** : Création, gestion, et suivi des tickets d'incidents signalés par les utilisateurs.
- **Maintenance prédictive** : Intégration d'une API externe pour analyser les données des équipements et prévoir des maintenances préventives.
- **Tableaux de bord et rapports** : Suivi dynamique des équipements, incidents, et performances via des graphiques et rapports détaillés.
- **Sécurité et gestion des utilisateurs** : Authentification sécurisée, gestion des rôles (administrateur, technicien, utilisateur), et audit des actions des utilisateurs.

III. Fonctionnalités du Système

1. Gestion des Équipements

- **Enregistrement des équipements** :

Les équipements peuvent être enregistrés avec les informations suivantes :

  - Marque
  - Modèle
  - Numéro de série
  - Emplacement
  - Date d'achat
  - Date d'utilisation

- **Gestion des statuts des équipements** :

Chaque équipement peut avoir un statut :

  - Opérationnel
  - En maintenance
  - Hors service

- **Historique des modifications et des interventions** :

Un historique détaillé des modifications, réparations et interventions pour chaque équipement.

2. Gestion des Incidents

- **Création et suivi des tickets d'incidents** :

Les utilisateurs peuvent signaler des incidents avec une description, une catégorie et une priorité (critique, moyenne, faible).

- **Priorisation des incidents** :

Le système doit permettre de gérer les tickets selon leur priorité.

- **Notifications automatiques** :

Les techniciens recevront des notifications pour chaque ticket qui leur est assigné.

3. Maintenance Prédictive

- **API externe pour analyse des données** :

L'application devra intégrer une API externe capable de récupérer des données sur les équipements (cycles d'utilisation, température, erreurs fréquentes, etc.) pour prévoir les pannes futures.

- **Alertes automatiques pour maintenance préventive** :

Le système enverra des alertes afin de planifier des maintenances préventives basées sur l'analyse des données.

4. Tableaux de Bord et Rapports

- **Tableaux de bord dynamiques** :

Des graphiques interactifs (via **Chart.js** ou **Highcharts**) pour visualiser les informations sur les équipements, les incidents et les performances des systèmes.

- **Rapports détaillés** :

L'application permettra de générer des rapports sur les équipements, les incidents, et les performances, avec possibilité d'exportation (PDF, Excel).

5. Sécurité et Gestion des Utilisateurs

- **Authentification sécurisée** :

Utilisation de JWT ou OAuth pour une gestion sécurisée des connexions.

- **Gestion des rôles** :

Trois types d'utilisateurs avec des accès différents :

  - **Administrateur** : Accès complet au système, gestion des équipements, des utilisateurs, et des incidents.
  - **Technicien** : Accès aux tickets d'incidents, à la maintenance, et aux interventions.
  - **Utilisateur** : Accès uniquement à la création de tickets et au suivi de leurs incidents.

- **Audit des actions des utilisateurs** :

Un système de journalisation des actions effectuées par les utilisateurs dans le système.

IV. Architecture Technique

- **Backend** :
  - Framework **Laravel** pour la gestion des routes, des API, et de la logique métier.
  - **MySQL** pour la gestion des bases de données.

    - **API externe** pour la maintenance prédictive, avec des requêtes HTTP pour récupérer et analyser les données des équipements.
- **Frontend** :
    - **Laravel Blade** pour la gestion des vues et des templates, permettant d'utiliser **HTML, CSS, et JavaScript** pour une interface utilisateur dynamique.
    - Utilisation de **Chart.js** ou **Highcharts** pour afficher des graphiques dynamiques des équipements et incidents.
- **Sécurité** :
    - Authentification via **JWT** ou **OAuth** pour sécuriser l'accès aux données.
    - **HTTPS** pour sécuriser les communications entre le client et le serveur.

V. Contraintes du Projet

- **Temps de réalisation** :

Le projet doit être terminé en [temps estimé] avec des étapes claires de validation à chaque phase.

- **Technologies spécifiques** :

Le backend doit être développé en **Laravel**, le frontend en **Laravel Blade avec HTML, CSS, et JavaScript** (sans utiliser de framework JavaScript comme Vue.js ou React), avec **MySQL** pour la base de données.

- **Accessibilité et compatibilité** :

L'application doit être accessible sur tous les navigateurs modernes (Chrome, Firefox, Edge, Safari).

VI. Phases du Projet et Planning

1. **Phase 1 : Analyse et Conception (2 semaines)**
    a. Réunions de lancement et définition des spécifications détaillées.
    b. Conception des maquettes et wireframes pour l'interface utilisateur.
    c. Mise en place de l'architecture du projet (backend, frontend).
2. **Phase 2 : Développement du Backend (4 semaines)**
    a. Mise en place de Laravel, définition des routes et des contrôleurs.
    b. Création des modèles pour la gestion des équipements, des incidents et des utilisateurs.
    c. Intégration de l'API de maintenance prédictive.
3. **Phase 3 : Développement du Frontend (4 semaines)**
    a. Développement des interfaces utilisateur (dashboard administrateur, technicien, utilisateur) en **Laravel Blade** avec **HTML, CSS et JavaScript**.

    b. Intégration des graphiques et rapports dynamiques avec **Chart.js** ou **Highcharts**.
    c. Tests d'intégration entre le backend et le frontend.
4. **Phase 4 : Tests et Validation (2 semaines)**
    a. Tests fonctionnels et unitaires.
    b. Tests d'acceptation par les utilisateurs (acceptation des fonctionnalités).
    c. Tests de sécurité (authentification, gestion des rôles).
5. **Phase 5 : Mise en Production et Livraison (1 semaine)**
    a. Déploiement du projet sur le serveur de production.
    b. Formation des utilisateurs finaux.
    c. Livraison des documents techniques et manuels utilisateur.

VII. Critères de Validation

Le projet sera considéré comme terminé lorsque les critères suivants seront remplis :

1. Toutes les fonctionnalités du cahier des charges seront implémentées.
2. Le système de maintenance prédictive fonctionne avec les alertes automatiques basées sur les données de l'API externe.
3. Les utilisateurs peuvent s'enregistrer, se connecter et interagir avec les différentes interfaces.
4. Les rapports et graphiques sont générés correctement et peuvent être exportés.
5. Le système respecte les critères de sécurité et de gestion des utilisateurs.
6. Le projet est déployé sur un serveur en production et fonctionne correctement.

VIII. Livrables du Projet

- **Code source** : L'intégralité du code backend et frontend.
- **Documentation** : Documentation technique détaillée (architecture, API, base de données) et manuel utilisateur.
- **Rapports** : Rapports de tests, validations, et qualité.
- **Formation** : Formation à l'utilisation de l'application pour les utilisateurs et techniciens.

IX. Conclusion

Ce cahier des charges définit les attentes et les objectifs du projet de manière claire et détaillée. Le respect des contraintes techniques, des délais et des exigences fonctionnelles garantira le succès de la mise en œuvre du gestionnaire de parc informatique avec maintenance prédictive.