Compte Rendu Détaillé du Projet : Développement d'une Application Web de Gestion de Parc Informatique

Date de rendu : 04/04/2025

Durée du projet :

**1. Contexte et Objectifs**

Objectif principal : Développer une application web centralisée pour gérer le parc informatique d’une organisation, en assurant le suivi des équipements, la gestion des licences logicielles, et le traitement des demandes d’intervention technique.

Livrables attendus :

- Application web fonctionnelle avec interface intuitive.

- Base de données recensant tous les équipements (PC, serveurs, imprimantes) et logiciels (licences, dates d’expiration).

- Module de gestion des interventions techniques (suivi des tickets, historique des résolutions).

- Documentation technique et guide utilisateur.

Contraintes :

- Compatibilité avec les navigateurs modernes (Chrome, Firefox, Edge).

- Sécurité des données (chiffrement, contrôle d’accès).

- Respect du budget et des délais.

**2. Méthodologie et Organisation**

Rôles de l’équipe :

- Chef de projet : Pilotage, coordination des ressources.

- Développeur full-stack : Développement backend/frontend.

- UI/UX Designer : Conception de l’interface utilisateur.

- Administrateur base de données : Modélisation et optimisation SQL.

- Responsable qualité : Tests fonctionnels et sécurité.

Outils utilisés :

- Stack technique :

- Frontend : React.js + Bootstrap.

- Backend : Node.js/Express.js.

- Base de données : PostgreSQL.

- Gestion de projet : Jira + Confluence.

- Sécurité : OWASP ZAP pour les tests de vulnérabilités.

**3. Phases du Projet**

Phase 1 : Analyse des besoins et conception (2 semaines)

- Recueil des exigences :

- Ateliers avec les utilisateurs finaux (équipe IT, gestionnaires).

- Priorisation des fonctionnalités via la méthode MoSCoW :

- Must-Have : Suivi des équipements, gestion des licences.

- Should-Have : Tableau de bord analytique.

- Maquettes et architecture :

- Wireframes (Figma) pour les pages principales (inventaire, tickets).

- Schéma ERD (Entity-Relationship Diagram) pour la base de données.

Phase 2 : Développement (6 semaines)

- Backend :

- API REST pour les opérations CRUD (créer, lire, mettre à jour, supprimer).

- Intégration de Swagger pour la documentation technique.

- Exemple de route :

```javascript

// GET /api/equipments?status=active

router.get('/equipments', (req, res) => {

const status = req.query.status;

// Requête SQL filtrée...

});

```

- Frontend :

- Tableau interactif avec filtres (statut, type d’équipement).

- Formulaire de création de ticket d’intervention (capture de screenshot possible).

- Base de données :

- Tables principales : Equipments, Software, Licenses, Tickets.

- Relations : Un équipement peut avoir plusieurs logiciels installés.

Phase 3 : Tests et validation (2 semaines)

- Tests unitaires : Couverture de 80 % avec Jest (ex: vérification des calculs de dates de licence).

- Tests d’intégration : Simulation d’utilisateurs simultanés (outil JMeter).

- Tests de sécurité :

- Injection SQL : Blocage des requêtes malveillantes.

- XSS : Sanitisation des entrées utilisateur.

Phase 4 : Déploiement et formation (1 semaine)

- Déploiement :

- Hébergement sur AWS EC2 (instance Linux).

- Configuration HTTPS via Let’s Encrypt.

- Formation utilisateur :

- Session en présentiel : Gestion des équipements et des tickets.

- Vidéos tutoriels : Ajout d’un nouvel équipement, suivi des licences.

Phase 5 : Documentation et support (1 semaine)

- Documentation :

- Guide administrateur : Procédures de sauvegarde, gestion des rôles.

- Manuel utilisateur : Captures d’écran, FAQ.

- Support post-déploiement :

- Correctifs prioritaires dans les 72h.

- Forum interne pour les retours utilisateurs.

**4. Résultats et Livrables**

Application fonctionnelle :

- Fonctionnalités clés :

- Inventaire : 500+ équipements recensés, statut mis à jour en temps réel.

- Licences : Alertes automatiques 30 jours avant expiration.

- Tickets : Réduction de 40 % du temps de résolution moyen.

- Performance : Temps de réponse < 1s pour les requêtes critiques.

Documentation :

- 100 % des processus couverts (ajout d’équipement, gestion des tickets).

- Rapport de tests de sécurité conforme aux normes OWASP.

**5. Défis Rencontrés et Solutions**

Problème 1 : Synchronisation des données en temps réel

- Solution : Implémentation de WebSockets pour les notifications instantanées (ex: changement de statut d’un équipement).

Problème 2 : Gestion des licences complexes (multi-sites)

- Solution : Ajout d’un champ "Site" dans la table Licenses et filtrage hiérarchique.

Problème 3 : Adoption par les équipes terrain

- Solution : Ateliers pratiques avec cas concrets (ex: résolution d’un ticket simulé).

**6. Bilan et Perspectives**

Points forts :

- Centralisation des données et automatisation des alertes.

- Interface utilisateur ergonomique, réduisant la courbe d’apprentissage.

Améliorations futures :

- Intégration IoT : Surveillance automatique de l’état des équipements via capteurs.

- Module de reporting avancé : Export PDF/Excel, graphiques personnalisés.

**Annexes**

Annexe 1 : Captures d’écran de l’application

1. Tableau de bord : Vue globale du parc informatique (stats par statut, type).

2. Fiche équipement : Détails techniques, historique des interventions.

3. Formulaire de ticket : Catégorisation (urgence, type de panne).

Annexe 2 : Exemple de schéma de base de données

```plaintext

Table "Equipments"

- ID (PK)

- Nom

- Type (PC, Serveur, Imprimante)

- Statut (En service, En panne, Hors service)

- Date\_achat

- Garantie\_jusquà

Table "Tickets"

- ID (PK)

- Equipment\_ID (FK)

- Date\_ouverture

- Date\_resolution

- Description\_panne

- Technicien\_assigné

```

Annexe 3 : Extrait du guide utilisateur

Procédure d’ajout d’un équipement :

1. Cliquer sur "Nouvel équipement" dans le menu.

2. Remplir les champs obligatoires (Nom, Type, Numéro de série).

3. Uploader la fiche technique (PDF).

4. Valider pour enregistrer dans l’inventaire.

Équipe Projet

- Chef de projet : [Nom]

- Développeur full-stack : [Nom]

- UI/UX Designer : [Nom]

- Administrateur base de données : [Nom]