Cahier des charges Projet UML

1. Introduction

Il s'agit d'une plateforme web collaborative dédiée aux développeurs open source, offrant une gestion avancée de dépôts Git, des outils de collaboration et un système de communication efficace.

2. Objectifs

✔ **Gestion de projets** : Création, fork, gestion des branches et historique des modifications.
✔ **Collaboration** : Revues de code, pull requests, gestion des rôles.
✔ **Sécurité** : Authentification forte, contrôle d'accès, chiffrement des données.
✔ **Automatisation** : Notifications, intégrations CI/CD.
✔ **Bonus** : Détection de vulnérabilités (IA) ou pair programming en temps réel.

---

3. Spécifications Fonctionnelles

3.1. Gestion des Utilisateurs

- **Inscription** :
  - Choix d'un identifiant, email valide et mot de passe.
  - Validation par email avant activation du compte.
- **Authentification** :
  - **Connexion sécurisée** (OAuth2, 2FA).
  - **Gestion des rôles** :
    - **Développeur** : Lecture/écriture sur les dépôts autorisés.
    - **Chef de projet** : Gestion des branches, validation des merges.
    - **Admin** : Gestion des utilisateurs et permissions globales.

3.2. Organisations

- **Création** : Un utilisateur peut créer une organisation et en devient automatiquement propriétaire.
- **Rôles** :
  - **Propriétaire** : Peut ajouter/supprimer des membres et d'autres propriétaires.
  - **Membre** : Accès aux projets de l'organisation.
- **Gestion des membres** :
  - Un propriétaire peut promouvoir un membre en propriétaire.
  - Tout propriétaire est automatiquement membre.

3.3. Gestion des Projets

- **Création** :
  - Choix entre **public** (visible par tous) ou **privé** (restreint).
  - Projet lié à un compte personnel ou une organisation.
  - Nom unique par propriétaire.
  - URL du dépôt Git associé.
- **Droits d'accès** :
  - **Projet personnel** : Seul le propriétaire peut modifier le dépôt.
  - **Projet d'organisation** : Tous les membres peuvent modifier le dépôt.
- **Gestion des branches** :
  - Création, suppression, merge.
  - **Gestion des conflits** avec outil de résolution visuel.
- **Historique des commits** :
  - Visualisation des différences (**diff visuel**).
  - Recherche par auteur, date, message.

3.4. Collaboration

- **Revues de code** :
  - Commentaires **inline** sur les modifications.
  - Mentions (@utilisateur) pour notifier des participants.
- **Pull Requests (PR)** :
  - Workflow d'approbation (nombre de validations requis).

- Blocage si conflits ou échec des tests CI.
- **Système d'Issues** :
  - Tout utilisateur peut créer une issue sur n'importe quel projet.
  - Fil de discussion chronologique.

3.5. Fork de Projets

- Un utilisateur authentifié peut forker un projet public.
- Choix du propriétaire (compte personnel ou organisation dont il est propriétaire).
- Le nouveau projet est initialisé avec le contenu du dépôt d'origine.

3.6. Automatisation

- **Notifications** :
  - Envoi d'emails/alertes pour :
    - Échec de build CI.
    - Merge réussi.
    - Nouvelle issue/PR.
- **Intégration CI/CD** :
  - Exécution automatique des tests à chaque push.

3.7. Sécurité Avancée

- **Permissions granulaires** :
  - Accès en **lecture seule** ou **écriture** par dépôt.
  - Restrictions par branche (ex : `main` protégée).
- **Chiffrement** :
  - Données sensibles (mots de passe, tokens API).
  - Code source au repos (storage chiffré).

---

4. Contraintes Techniques

4.1. Performance

- Support des **dépôts >10 Go** sans latence.
- Optimisation des opérations Git (clone, pull, push).

4.2. Interopérabilité

- **Import/Export** de dépôts depuis/sur d'autres plateformes (GitHub, GitLab).
- **API REST/GraphQL** pour intégrations externes.

4.3. Sécurité

- **Chiffrement AES-256** pour les données sensibles.
- **Audit logs** pour tracer les actions critiques.

---

5. Fonctionnalités Optionnelles

5.1. Détection de Vulnérabilités par IA

- Analyse statique du code (ex : détection de SQLi, XSS).
- Suggestions de correctifs automatisés.

5.2. Pair Programming en Temps Réel

- **Éditeur partagé** (ex : VSCode Live Share intégré).
- **Terminal collaboratif** pour débogage simultané.

---

6. Livrables Attendus

- **Application web** (frontend React/Vue + backend Node/Go).
- **Base de données** (PostgreSQL pour les métadonnées).
- **Stockage Git** (via Git LFS pour les gros fichiers).
- **Documentation technique et utilisateur**.