GITMEMONY

GUIDE DE RECOMMANDATIONS

TABLE DES MATIÈRES

OBJET

Ce guide de recommandations permet de s'assurer que les standards de développement sont respectés et de définir les méthodes de travail et les outils nécessaires pour l'équipe de développement afin qu'ils puissent mettre correctement en œuvre l'architecture cible.

CONTEXTE

Désirant être à la pointe des dernières avancées technologiques, GitmeMony a fait de la crypto-monnaie le cœur de ses derniers projets.

C'est ainsi que GitmeMony a entrepris la création d'une plateforme qui connecte différents sites marchands de crypto-monnaies, pour permettre aux banques d'acheter et de vendre différentes crypto-monnaies et de les utiliser par la suite.

STANDARDS ET BONNES PRATIQUES DE DÉVELOPPEMENT

Normes de codages

La convention de nommage **lower camel case** doit être appliquée à toutes les lignes de code produites.

L'IDE utilisé doit prévoir une **indentation automatique**. A défaut, l'indentation doit être faite manuellement en utilisant le style d'indentation généralement utilisé en Java (unité d'indentation : 4 espaces ; longueur de ligne < 80 caractères ; ...).

Qualité / Prévention de la dette technique

Les principes du **Clean Code** doivent régir la production du code.

Le code produit doit être **optimisé** grâce à l'utilisation de **librairies**.

Le code produit doit faire l'objet d'une **refactorisation** régulière afin de maintenir la lisibilité et la maintenabilité à un niveau élevé.

Le code produit doit être **documenté** et la documentation doit être générée.

Tout code commité doit faire l'objet de **code review** (revue du code par un pair) avant de pouvoir être mergé dans une branche de pré-production

La montée en compétence des développeurs juniors doit être faite par l'organisation de **peer programming** sur certaines tâches de développement. Le(la) développeur(se) senior sera conducteur(trice) tandis que le(la) développeur(se) junior sera l'observateur(trice).

Sécurité

Les normes de sécurité, notamment la norme **ISO-27001** et les bonnes pratiques concernant celle-ci, notamment celles éditées par la fondation **OWASP** doivent être suivies.

Tests

Le code écrit doit respecter les 3 lois du **TDD** :

- Écrire un test qui échoue avant d'écrire le code de production correspondant.
- Écrire une seule assertion à la fois, qui fait échouer le test ou qui échoue à la compilation.
- Écrire le minimum de code de production pour que l'assertion du test actuellement en échec soit satisfaite.

Contrôle de version

Il est recommandé de ne pas travailler directement sur la branche master et, par conséquent, des **branches de travail** doivent être créées. La préconisation pour ce projet est de créer une branche par **Sprint**. Chaque branche de travail sera ensuite mergée vers la branche master en fin de sprint.

Il est également recommandé de faire un **nettoyage** de branche régulier.

MÉTHODOLOGIES DE PROJET

Le framework **Scrum** de la méthodologie **Agile** doit être utilisé.

La durée des **sprints** doit être de **2 semaines** et ne doit pas être modifiée pendant toute la durée du projet.

Les **cérémonies** agiles doivent avoir lieu à un **rythme régulier** pour tous les sprints. Tous les membres de l'équipe doivent participer à toutes les cérémonies agiles (sauf congés, maladie, ...).

Liste des cérémonies :

Cérémonie	Durée	Rythme
Daily Scrum	15 minutes	Tous les jours
Planning Poker	45 minutes	Une fois par semaine
Sprint planning	4h	Une fois par sprint
Backlog refinement (affinage)	1h	Une fois par semaine
Sprint review	2h	Une fois par sprint
Rétrospective	1h30	Une fois par sprint

OUTILS

Espace de travail

Un outil de **travail collaboratif** tel que **Confluence** doit être utilisé. Cet outil doit aussi être utilisé comme logiciel de **wiki** permettant de gérer la documentation du projet.

Projet

Le projet doit être suivi sur un **outil de gestion de projet Agile**. L'outil préconisé est **Jira**. Jira permet de suivre tous les tickets du projet (Epic, User Stories, Tâches, Bugs, Spikes, ...)

Cet outil doit être associé à des outils nécessaires pour certaines **cérémonies** agiles :

- Outil de **planning poker** tel que **Agile Poker for Jira**
- Outil pour les **rétrospectives** tel que **Trello**

IDE

Les **IDE IntelliJ IDEA**, **Eclipse**, **NetBeans** ou **VS Code** peuvent être utilisés.

Sécurité

Les outils **OWASP** doivent être utilisés pour garantir la **sécurité** du code produit :

- OWASP SKF et Top Ten OWASP pour guider les développeurs
- OWASP ZAP pour contrôler le code produit

Qualité de code

Un outil de **qualimétrie** doit être utilisé pour contrôler la qualité du code, de mesurer le niveau de documentation et de connaître la couverture des tests déployée. L'outil préconisé est **SonarQube**.

Documentation

Un outil de génération automatique de la ***documentation*** doit être utilisé. L'outil préconisé est **Javadoc.**

Code Review

Un outil de code review doit être intégré à l'outil de suivi de projet, de suivi de version ou à l'IDE. L'outil préconisé est **Crucible**.

Contrôle de version

Le **versionnage** du code doit être fait grâce à **Git** et stocké sur **Github** dans un dépôt privé.

CI/CD

Un outil d'**intégration continue/déploiement continu** doit être utilisé. L'outil préconisé est **Jenkins**. Jenkins prend en charge les opérations de build, d'automatisation des tests et de déploiement.

Datas

Un outil assurant la **persistance** des données doit être utilisé. L'outil préconisé est **Hibernate**.

Conteneur

Un outil de **conteneurisation** doit être utilisé. L'outil préconisé est **Docker**.

Un outil permettant l'**orchestration** de conteneurs doit être utilisé. L'outil préconisé est **Kubernetes**.