KAMANGACODE - KAMANGA.FR

Checklist de revue de code

1. Lisibilité et maintenabilité

- [ ] Le code est-il facile à lire et à comprendre ?
- [ ] Les noms de variables, fonctions et classes sont-ils clairs et descriptifs ?
- [ ] Le code suit-il les conventions de nommage du projet/langage ?
- [ ] Les commentaires sont-ils pertinents et à jour ?
- [ ] Le code est-il bien formaté et indenté ?
- [ ] Y a-t-il du code dupliqué qui pourrait être factorisé ?

2. Architecture et conception

- [ ] Le code respecte-t-il les principes SOLID ?
- [ ] La séparation des responsabilités est-elle bien respectée ?
- [ ] Les abstractions sont-elles appropriées et cohérentes ?
- [ ] Le code s'intègre-t-il bien dans l'architecture globale du projet ?
- [ ] Les dépendances sont-elles gérées efficacement ?

3. Fonctionnalité

- [ ] Le code implémente-t-il correctement les spécifications requises ?
- [ ] Tous les cas d'utilisation sont-ils pris en compte ?
- [ ] Les cas limites sont-ils gérés ?
- [ ] Le code gère-t-il correctement les erreurs et exceptions ?

4. Performance et efficacité

- [ ] Le code est-il optimisé pour la performance ?
- [ ] Y a-t-il des algorithmes ou structures de données qui pourraient être améliorés ?
- [ ] Les requêtes de base de données sont-elles optimisées ?
- [ ] Le code évite-t-il les calculs ou opérations inutiles ?

5. Sécurité

- [ ] Le code est-il protégé contre les vulnérabilités courantes ?
- [ ] Les données sensibles sont-elles correctement protégées ?
- [ ] Les entrées utilisateur sont-elles validées et nettoyées ?
- [ ] Les autorisations et authentifications sont-elles correctement gérées ?

6. Tests

- [ ] Le code est-il accompagné de tests unitaires appropriés ?
- [ ] La couverture de tests est-elle suffisante ?
- [ ] Les tests sont-ils clairs, bien organisés et maintenables ?
- [ ] Y a-t-il des tests d'intégration ou de bout en bout si nécessaire ?

7. Documentation

- [ ] Le code est-il suffisamment documenté ?
- [ ] La documentation est-elle à jour et cohérente avec le code ?
- [ ] Les changements d'API ou de comportement sont-ils documentés ?

8. Bonnes pratiques spécifiques

- [ ] Le code respecte-t-il les guidelines spécifiques au projet ou à l'équipe ?
- [ ] Les principes de programmation propres au langage sont-ils respectés ?
- [ ] Le code utilise-t-il correctement les bibliothèques et frameworks du projet ?

9. Gestion de version

- [ ] Le commit message est-il clair et descriptif ?
- [ ] Les changements sont-ils atomiques et cohérents ?
- [ ] Le code est-il à jour avec la branche principale ?

10. Impacts plus larges

- [ ] Le code a-t-il des effets secondaires sur d'autres parties du système ?
- [ ] Les changements nécessitent-ils des mises à jour de la configuration ou du déploiement ?
- [ ] Y a-t-il des implications sur les performances ou la scalabilité du système ?

11. Amélioration continue

- [ ] Y a-t-il des opportunités d'apprentissage ou d'amélioration pour l'équipe ?
- [ ] Le code introduit-il de nouvelles techniques ou patterns intéressants ?
- [ ] Y a-t-il des suggestions pour améliorer le processus de développement ?