Les bonnes pratiques de la pull request Cheat Sheet

by tedprog07 via cheatography.com/203044/cs/43187/

Pourquoi ?

- Améliorer la qualité du code
- Réduire le nombre de bugs
- Progresser en lisant du code qui n'est pas le sien
- Partager la connaissance d'un sujet

Quand ?

Story => Développement sur une feature-branch => **Revue de code** => Merge => Tests => Déploiement

En tant que développeur

- Faire des petites Pull-requests
- Savoir accepter les retours de ses collègues
- Ne pas négocier systématiquement
- Remercier les bonnes idées ou accepter avec enthousiasme de meilleures solutions
- Répondre à tous les commentaires avant le merge

Respecter les principes de l'Egoless programming

- Être humble
- Accepter de faire des erreurs
- Critiquer le code, pas le developpeur
- Ne pas prendre les choses personnellement (Tu n'es pas ton code)
- Accepter les compromis pour trouver une solution rapidement, surtout quand les deadlines sont rapprochées

Que vérifier ?

- Est-ce que ça correspond à la fonctionnalité demandée ?
- Est-ce qu'il y a quelque chose d'aberrant ?
- Est-ce que le code est facile à lire ?
- Est-ce que le code est organisé correctement, est ce que les patterns utilisés sont judicieux ?
- Est-ce que le code pourrait créer un problème de performance ?

L'idée ici n'est pas de lire le code de manière exhaustive, mais plutôt de se focaliser sur quelques principes

En tant que relecteur

- Être bienveillant
- Éviter les jugements
- Préférer poser des questions aux injonctions et demandes
- Ne pas corriger le code à la place d'un collègue derrière son dos
- Commenter le code, pas la personne
- Se mettre à la place du dev et comprendre son contexte
- Éviter les rejets de Pull Request sur un outil : préférer une discussion face à face