GÜDLFT

Competitions Made Easy.

GUIDE DE DÉVELOPPEMENT

V. I.I

LE POURQUOI

En standardisant la façon dont nous créons des projets et des branches dans GitHub, il est facile pour d'autres personnes de lire et de comprendre l'avancement des projets, qu'il s'agisse de l'assurance qualité, des utilisateurs ou d'autres développeurs.

LE COMMENT

1. Lorsque vous créez votre projet, ajoutez toujours un *README*. Cela vous aidera à expliquer l'objectif de votre projet. N'oubliez pas d'inclure des éléments tels que :
    a. Des liens vers des ressources externes.
    b. Des configurations particulières.
    c. Les conventions de nommage.
    d. Tout ce qui nécessiterait que quelqu'un vienne vous dire : "J'ai une question sur X."
2. La branche principale (master) est votre source de vérité (code fini). Tout le monde devrait pouvoir cloner à partir de la branche principale et faire fonctionner le projet correctement.
3. Si vous voulez ajouter une fonctionnalité, corriger un bogue ou tout ce qui pourrait enfreindre la règle n°2, créez une branche. Le nom de la branche doit citer le type d'ajout et suivre le format <fonctionnalité/bug/amélioration>/nom-descriptif. Par exemple, si vous trouvez un bogue, vous devez créer une branche appelée bug/nom-de-bogue, et le corriger (évidemment).

4. Le Testing
    a. Il faut toujours faire des tests. Si vous ne l'avez pas testé, il est cassé.
    b. Nous sommes agnostiques au frameworks - vous pouvez utiliser le pytest, l'unittest ou le Morelia. Peu importe ce que vous utilisez, vous devriez pouvoir le lancer depuis la ligne de commande.
    c. Veuillez mettre tous les tests dans un dossier de test pour que cela soit facile à trouver.
    d. Tests unitaires >> tests d'intégration >> tests fonctionnels. Au moins, écrivez des tests unitaires.
    e. Notre objectif est de couvrir notre code avec au moins 60% de tests. Plus de tests signifie plus de sommeil.
    f. Essayez d'écrire deux fois plus de tests unitaires que de tests d'intégration ou de tests fonctionnels. Les tests unitaires nous permettent de savoir comment les choses fonctionnent et quand elles ne fonctionnent pas.
    g. Pour éviter les douleurs oculaires, veuillez regrouper les tests unitaires, d'intégration et fonctionnels dans des dossiers séparés. Vous en serez heureux.

5. La Performance
    a. Même lors de la création d'un MVP (minimum viable product), veillez à aller le plus vite possible et à écrire le meilleur code possible.
    b. Utilisez Locust pour vous assurer que le temps de chargement ne dépasse jamais 5 secondes, et que les mises à jour ne prennent pas plus de 2 secondes.
    c. Le nombre d'utilisateurs par défaut pour les tests de performance est de six utilisateurs.
6. Concentrez-vous sur un bogue/une fonction/une amélioration à la fois (un par branche) ! Travailler sur plusieurs choses peut vous faire mal à la tête. De plus, il est difficile pour le reste de l’équipe de savoir sur quoi ils peuvent travailler.
7. Ne revenez dans la branche principale que lorsque tous vos tests sont réussis. N'oubliez pas que si ça ne marche pas, ça n'a pas sa place dans la branche principale.
8. Lorsque vous êtes prêt pour la révision du code, créez une branche QA à partir de la branche principale. Cependant, contrairement aux autres branches, celle-ci ne devrait pas être fusionnée dans la branche principale.