Le Moment Propice pour les Tests

Hier, je me suis lancé dans la création d'un outil de configuration automatique pour Shadowsocks Outline, avec l'objectif de le transformer en un projet Python que d'autres pourraient utiliser. J'ai développé un script qui met à jour un fichier `config.yaml` avec les configurations du proxy Shadowsocks en décodant les URLs Shadowsocks à partir d'un fichier `ssconfig`. De plus, j'ai créé un autre script qui utilise `gsutil` pour téléverser le fichier d'abonnement destiné aux clients sur Google Cloud Storage.

J'ai utilisé Windsurf, un éditeur de code assisté par IA, pour m'aider. Cependant, il a eu des difficultés à gérer les dépendances simulées dans les tests unitaires en Python.

En réfléchissant aux leçons de test partagées par Yin Wang, je me suis rappelé de ses expériences chez Google, où il a travaillé sur un interpréteur Python et indexé le code de l'entreprise pour la fonctionnalité de recherche. Ses collègues insistaient pour écrire des tests, ce qu'il trouvait ennuyeux. Il croyait que l'écriture de code élégant était plus importante que les tests, et que ses collègues ne comprenaient que des aspects superficiels sans saisir l'essence.

J'ai réalisé mon erreur ; l'IA ne l'a pas signalée. Je devrais m'assurer que le code principal d'une bibliothèque est solide avant de me concentrer sur les tests. Le même principe s'applique à un projet de preuve de concept. Dans mes emplois précédents, comme lors du démarrage d'un microservice, les tests devraient être écrits après que le microservice dispose de quelques API ou fonctions.

Si Windsurf avait bien géré la partie des tests, je n'aurais pas cette critique. Cependant, il y a deux problèmes distincts en jeu : le moment de la mise en œuvre des tests et la manière correcte de les écrire. Actuellement, nous nous concentrons sur le premier. Ces problèmes sont liés dans une certaine mesure. Si un éditeur de code IA ou un humain trouve facile d'écrire du code de test, le moment des tests pourrait sembler trivial. Cependant, l'effort requis pour écrire des tests est comparable à celui de l'écriture du code principal, ce qui rend le moment un aspect important à considérer.

D'un point de vue collaboratif, l'approche des tests peut varier. Pour un projet personnel, je pourrais écrire une quantité importante de code avant de créer des tests. Cependant, lorsque je travaille en équipe, il est généralement préférable d'écrire des tests pour chaque extrait ou fonctionnalité. Mais ce n'est pas toujours le cas ; cela dépend de la manière dont l'équipe collabore. Une manière plus précise de le formuler est que les tests devraient être écrits pour le code que les membres de l'équipe partagent entre eux. L'objectif est de garantir la qualité du code, donc avant de livrer le code, chaque membre de l'équipe est libre de choisir le moment où il effectue ses tests.

Lors d’une précédente expérience professionnelle, j’ai collaboré avec trois autres ingénieurs backend sur une fonctionnalité qui a pris six mois à être livrée. Du point de vue des tests, les points abordés dans cet article peuvent expliquer pourquoi le développement était lent à cette époque.

Du point de vue de la collaboration, les personnes responsables du code principal devraient également être responsables des tests associés. Les tâches devraient s’entrelacer le moins possible, avec des responsabilités claires et distinctes pour chaque membre de l’équipe.

Revenons au sujet des tests, les éditeurs de code IA manquent également de ce type d’optimisation, ce qui met en lumière un domaine à améliorer. Ce principe ne se limite pas à l’ingénierie logicielle ; il est également pertinent pour le matériel et d’autres domaines. Les tests sont une forme d’optimisation, et comme le dit le proverbe, “L’optimisation prématurée est la racine de tous les maux.”

Il est crucial de se rappeler l’objectif principal du travail. Bien que les processus et les procédures soient inévitables, nous devons garder à l’esprit ce qui est vraiment important.

Références :

- Test-Driven Development, Yin Wang
- The Logic of Testing, Yin Wang