Jérémy Sorant

INTRODUCTION AUX CONCEPTS ORIENTÉS CRAFT

LE TEST-DRIVEN DEVELOPMENT

Jérémy Sorant

Avec cette série de mini-BDs, j'essaie de présenter des concepts orientés craft afin de les rendre accessibles aux non-initiés.

in/jeremy-sorant

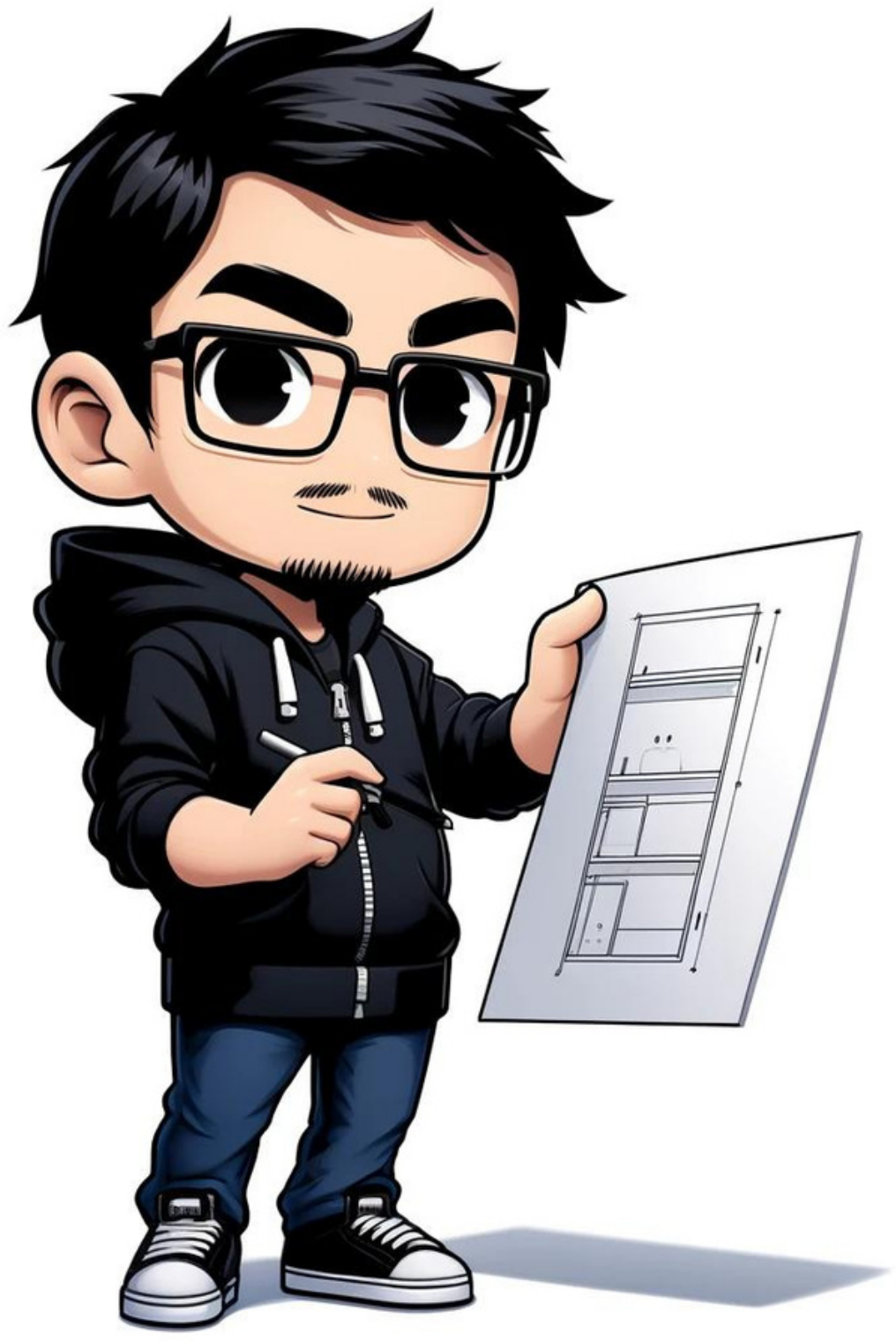
jsorant

Jérémy Sorant

DÉFINITION

Le TDD consiste à guider la conception par les tests via un processus itératif au cours duquel on applique un cycle en 3 phases : RED / GREEN / REFACTOR.

On va chercher à découper un problème en étapes (ou “steps”) et résoudre chaque étape via un cycle.



RED

Phase “RED” : j’écris un test qui défini un comportement que je souhaite implémenter. Comme il n’existe pas encore, ce test échoue : il est rouge.

Je lis avec attention le message d’erreur et je m’assure qu’il indique correctement la source du problème.

GREEN

Phase “GREEN” : je cherche à faire passer mon test le plus simplement possible.

Je ne me soucie pas de la qualité du code et je laisse volontairement le design et le nettoyage de code de côté.

REFACTOR

Phase “REFACTOR” : je retravaille le code pour le rendre plus lisible et pour supprimer la duplication de code.

Je ne modifie en aucun cas le comportement du code, mes tests doivent rester vert.

Un design plus adapté peut émerger lors de cette étape.

CHARGE COGNITIVE

En dissociant l'implémentation du design, le TDD allège ma charge cognitive et me permet d'être plus efficace.

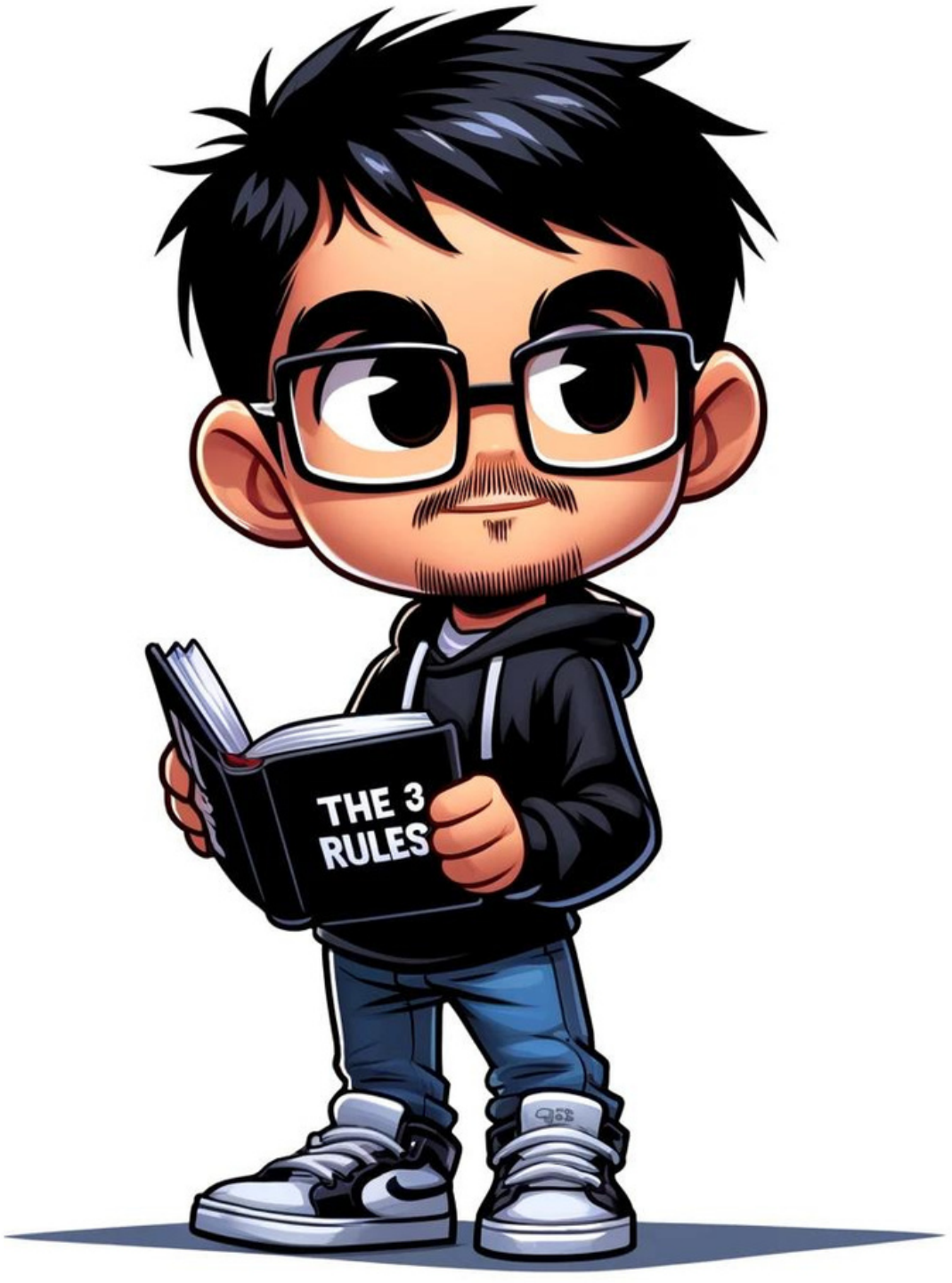
Le TDD me permet aussi d'aller plus vite vers une solution simple. Je m'évite une phase de design préliminaire qui conduit souvent à de l'over-engineering.

STEPS

Si le problème est complexe, je cherche à avancer avec de petites steps, appelées aussi “baby-steps”.

Si je suis à l’aise avec le problème, je peux augmenter la taille de mes steps, mais j’en reste conscient afin de pouvoir revenir en arrière si je me heurte à des difficultés.

Jérémy Sorant



LOiS

Uncle Bob propose 3 lois pour encadrer le TDD :

1. N’écris pas de code de production tant que tu n’as pas écrit un test unitaire qui échoue.
2. N’écris pas plus de code de test qu'il n'en faut pour qu’il échoue (ne pas compiler revient à échouer).
3. N’écris pas plus de code de production que nécessaire pour que le test unitaire réussisse.

TESTS RUNNER

Comme le TDD implique de lancer les tests (très) souvent, j'ai configuré mon tests runner en "watch mode".

Le watch mode permet au tests runner de surveiller mes fichiers et de relancer automatiquement les tests lorsque l'un d'eux est modifié. Ceci me permet d'avoir un feedback à chaque CTRL + S.

Jérémy Sorant
ingénieur logiciel sénior
chez SHODO LYON

l'ESN craft militante
#justicesociale