Jérémy Sorant

FOCUS ON CRAFT PRACTICES

TOOLBOX

Like the carpenter who works wood, we have many tools at our disposal to practice our art.

Here is a brief presentation of the tools and practices I use on a daily basis.

DOMAIN DRIVEN DESIGN

I primarily aim to solve business issues.

To achieve this, I liaise with domain experts and, if possible, with end-users of my product. I draw upon the various tools of Domain-Driven Design to thoroughly understand each problem and implement the most suitable solution.

TESTS

I write tests to quickly get feedback on what I produce.
I want to be sure that my code works and meets the expected requirements.

I strive to write robust and automated tests to ensure the quality of my code, but also to work confidently with a safety net.

BEHAVIOR DRIVEN DEVELOPMENT

BDD is a method that involves driving development through behaviors.

Developers, QA, and Product Owners collaborate to write Gherkin scenarios together, and then I use Cucumber to automate them. This way, I have tests and a living non-technical documentation on the various behaviors of my product.

TEST DRIVEN DEVELOPMENT

TDD is a method that involves driving development through tests.

This practice helps me stay focused on the requirement, break down each problem effectively, quickly write simple, testable, and quality code, bring out designs, and reduce the cognitive load required to progress daily.

DOUBLE LOOP

I develop using a double loop approach with both BDD and TDD.

I automate a BDD scenario which forms the first loop. This defines a complete use case, and its associated test remains red until the desired behavior is implemented.

Then, I go through as many small loops as necessary in TDD until my BDD test passes.

REFACTORiNG

Refactoring involves modifying the code structure without changing its behavior.

I use it regularly to keep the codebase clear, maintainable, and scalable. I rely on my IDE to maximize my efficiency.

CLEAN CODE

I take care to write clean code because I know that I spend an average of 80% of my time reading it while developing.

While it's easy to write code that a machine can understand, I strive to produce code that is easily understandable by humans.

BOY SCOUT RULE

I always deliver the code in a better state than I found it.

Through these continuous small improvements, I maintain a virtuous cycle that makes the code clearer, more maintainable, and more scalable.

PAiR / MOB

I regularly code in pair or mob.

This practice provides instant feedback on our thought process and code writing quality.

Business and technical knowledge are disseminated more effectively, and it's a good method for sharing tips related to IDE manipulation.

KATAS

I regularly repeat coding exercises with the aim of improving and refining my practice.

Like in karate, I want programming to become an extension of myself.

Sometimes, I add some constraints to help me acquiring specific skills.

PARADiGMS

I'm familiar with the main programming paradigms, and I use them judiciously.

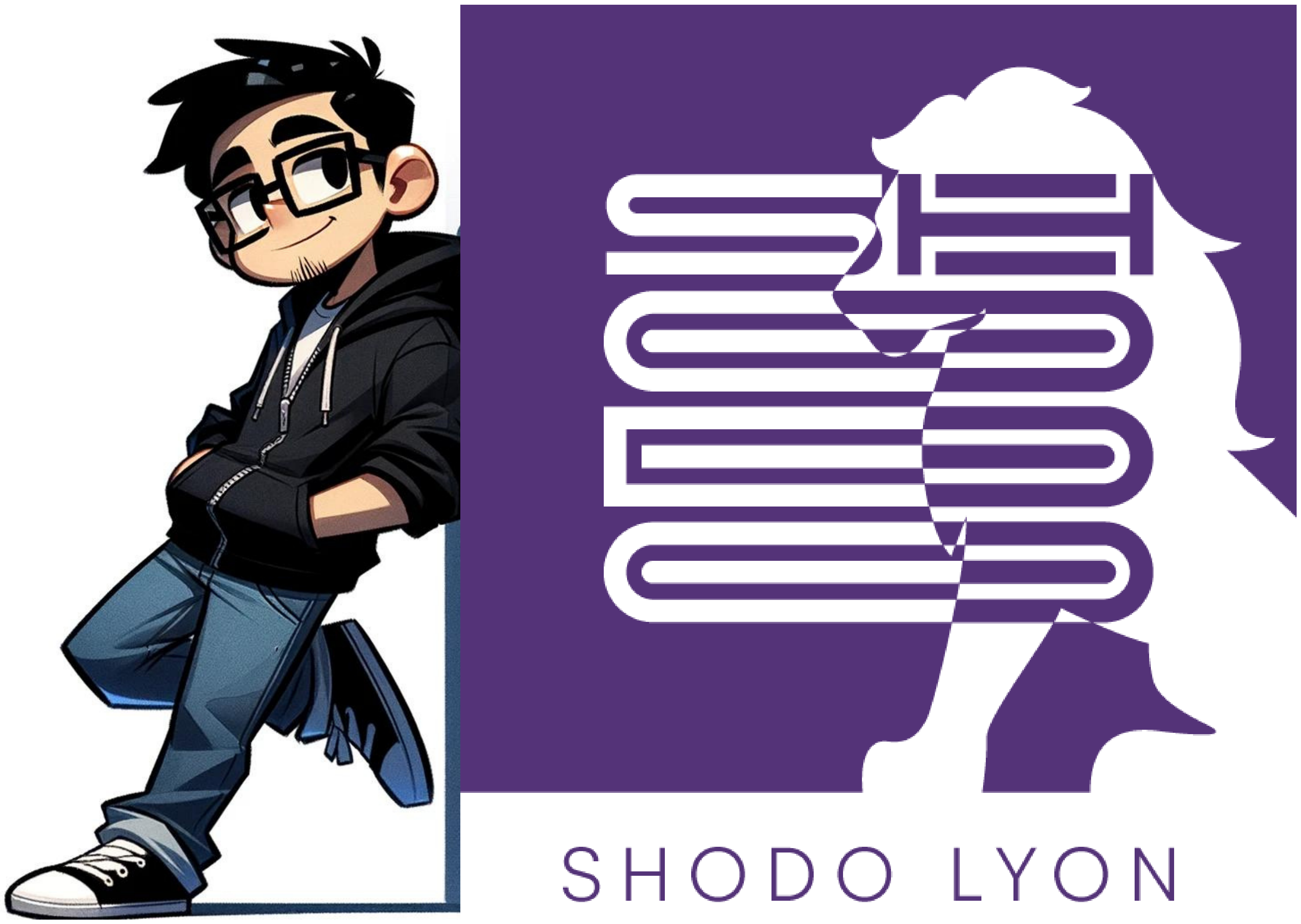
During my research, I study new practices to expand the approaches I could have to the same problem.

CONTINUOUS LEARNING

In addition to exchanging ideas with my peers, I continuously conduct my technological watch by reading books and blogs, listening to podcasts, watching webinars, and participating in conferences.

This allows me to continually enrich my toolbox of tools and practices.

Jérémy Sorant
senior software engineer
at SHODO LYON



l’ESN craft militante
#justicesociale