Sureify — Senior Engineer Challenge Approach

Cristian D. Moreno
Senior Full Stack Web Developer
Villavicencio/Colombia | +573022539479 | kyonax.corp@gmail.com | Linkedin | Github | Personal Web

For this challenge, I adopted a test-driven approach because it forces me to think through edge cases before implementation, and ensuring a more systematic type of solution, since the whole development of the function was not so long and extensive, I tried to focus mainly in the well practices, and tools that help me have a maintanable project.

TOOLS

- **Jest** for testing because is the tool that I know the most.
- **ESLint with custom rules** because consistent code style is crucial for maintainability. I enforced specific naming conventions (snake$_{case}$ for data, camelCase for functions) just for this specific case scenario to make the challenge more difficult and to show you the importance of it.

MY TDD APPROACH IN PRACTICE

I started by writing failing tests for the core pricing logic - this forced me to clearly define the expected behavior before writing any implementation.

1. **Red Phase**: I wrote tests that defined exactly how the pricing should work for two specific combinations of size, creamer, and sweetener. Based on the examples provided by the same exercise (more test could be added to it, but I try to keep it simple).
2. **Green Phase**: I implemented just enough code to make the tests pass, check the video to see more about it.
3. **Refactor Phase**: Once the logic was correct, I improved the code structure - extracting constants, improving type definitions, and ensuring the code was clean and intuitive.

What I particularly appreciate about this approach is that it creates a safety environment that allows me to refactor confidently, knowing that any breaking changes will be immediately caught by the test suites (improving the maintainability).

ARCHITECTURAL DECISIONS

I separated constants, types, and utilities because this isolation makes the system more adaptable. When pricing tables need updating, the change is centralize to one file without touching the business logic.

KEY NOTES

- Repository: Github Repo
- Every architectural decision was made with long-term maintainability in mind
- Detailed rationale for each choice is documented in CHANGELOG

CONTACT

I'm passionate about building sustainable, well-tested software. If you'd like to discuss this approach or have any questions, I'd be happy to connect via email or LinkedIn.