Test Smells

Thiago Andrade Ramalho

808904

02.11.2025

Teste de Software

Análise de Smells

Há Conditional Test Logic no teste de desativação, por conta do uso de for/if com expect condicional, pois transforma o teste em um mini-programa, o risco é cobertura ilusória e falhas pouco claras.

Ocorre Mystery Guest ao limpar o banco via _clearDB no beforeEach, preparando estado de forma implícita, fora do escopo explícito do caso de teste e acoplando a detalhe interno. O risco é de testes frágeis ou não determinísticos e de quebras em refactors..

Possui Fragile Test no caso de teste que junta criação e busca de usuário no mesmo teste, pois mistura comportamentos e depende da ordem dos passos. Isso ocasiona em diagnóstico confuso e quebras com mudanças pequenas.

Processo de Refatoração

Esse teste foi escolhido pois apresenta o maior número de smells do projeto, sendo eles identificados o uso do for e if na parte de dentro do teste, ocorrendo em expects condicionais, ou seja, é um caso de smell Conditional Test Logic, pois a execução das asserções depende de condições internas, o que torna o teste mais complexo e difícil de compreender. Nele também foram feitos cenários diferentes em um mesmo teste, ou seja, um sell Eager Test, já que há a verificação de comportamentos distintos, usuário comum e administrador, dentro do mesmo caso de teste. O teste além de não aplicar e mostra ser contra o padrão AAA, pois mistura preparação, execução e verificação.

```
33  test('deve desativar usuários se eles não forem administradores', () => {
34    const usuarioComum = userService.createUser('Comum', 'comum@teste.com', 30);
35    const usuarioAdmin = userService.createUser('Admin', 'admin@teste.com', 40, true);
36
37    const todosOsUsuarios = [usuarioComum, usuarioAdmin];
38
39    // O teste tem um loop e um if, tornando-o complexo e menos claro.
40    for (const user of todosOsUsuarios) {
41      const resultado = userService.deactivateUser(user.id);
42      if (!user.isAdmin) {
43        // Este expect só roda para o usuário comum.
44        expect(resultado).toBe(true);
45        const usuarioAtualizado = userService.getUserById(user.id);
46        expect(usuarioAtualizado.status).toBe('inativo');
47      } else {
48        // E este só roda para o admin.
49        expect(resultado).toBe(false);
50      }
51    }
52  });
```

Com a refatoração, separei em cenários diferentes, tendo um teste responsável por desativar usuário que não é administrador e outro para garantir que o usuário não seja desativado caso for um administrador. Foi retirado o loop e as condições na parte de dentro do teste, com isso, o código fica mais legível e limpo A refatoração também acabou com a smell Conditional Test Logic, dessa forma cada teste possui expects incondicionais, e também resolveu o problema do Eager Test, já que cada comportamento é validado isoladamente. Além disso, apliquei o padrão AAA, facilitando a leitura e a compreensão da estrutura do teste. Isso torna o código mais organizado, os testes mais descritivos e reduz a chance de erros futuros, além de remover os avisos do eslint relacionados ao uso condicional de expects.

```
43  test('desativa usuário se não for administrador', () => {
44    const comum = userService.createUser('Comum', 'c@t.com', 30, false);
45
46    const ok = userService.deactivateUser(comum.id);
47
48    expect(ok).toBe(true);
49    expect(userService.getUserById(comum.id).status).toBe('inativo');
50  });
51
52  test('não desativa administrador', () => {
53    const admin = userService.createUser('Admin', 'a@t.com', 40, true);
54
55    const ok = userService.deactivateUser(admin.id);
```

Relatório da Ferramenta

```
found 0 vulnerabilities
PS C:\Users\Thiago Andrade\Documents\GitHub\test-smelly> npx eslint .

C:\Users\Thiago Andrade\Documents\GitHub\test-smelly\test\userService.smelly.test.js
  44:9  error    Avoid calling `expect` conditionally`  jest/no-conditional-expect
  46:9  error    Avoid calling `expect` conditionally`  jest/no-conditional-expect
  49:9  error    Avoid calling `expect` conditionally`  jest/no-conditional-expect
  73:7  error    Avoid calling `expect` conditionally`  jest/no-conditional-expect
  77:3  warning  Tests should not be skipped            jest/no-disabled-tests
  77:3  warning  Test has no assertions                 jest/expect-expect

✖ 6 problems (4 errors, 2 warnings)

PS C:\Users\Thiago Andrade\Documents\GitHub\test-smelly>
```

O eslint conseguiu automatizar e detectar as smells, como por exemplo para Conditional Test Logic, pois informou do uso de asserções condicionais dentro de if/for. Também foi retornado avisos de testes ignorados e casos sem verificação real. Com a automatização das verificações de testes, é economizado tempo, pois não precisa rodar os testes para descobrir os problemas

além dos erros e avisos já sugerirem onde e como corrigir.

Conclusão

Testes limpos, escritos em AAA, sem lógica condicional no corpo e focados em um comportamento por vez, deixam o feedback objetivo e a manutenção mais tranquila, assim quando algo quebra, fica claro o que falhou e por quê. Também tem a análise estática, que automatiza a identificação de padrões ruins antes mesmo de rodar a suíte, padroniza prática e evita erros manuais bobos. Dessa forma, a qualidade e a sustentabilidade do projeto só aumenta, pois se tem refactor com segurança, evolução mais previsível e uma base de testes que realmente protege o sistema em vez de atrapalhar.