PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

MATHEUS VINICIUS MOTA RODRIGUES
814410

ANÁLISE DE TEST SMELLS

BELO HORIZONTE
2025

Introdução

Este relatório tem como objetivo demonstrar o processo de detecção e refatoração de *Test Smells* em uma suíte de testes do módulo UserService. O trabalho busca aplicar boas práticas de engenharia de software, garantindo testes mais legíveis, robustos e sustentáveis ao longo do tempo.

A metodologia empregada seguiu o padrão **Arrange, Act e Assert (AAA)** e utilizou a ferramenta **ESLint** para análise estática de código, com o intuito de identificar automaticamente problemas de estilo e padrões inadequados nos testes.

Análise de Test Smells Identificados

Durante a análise manual do arquivo *userService.smelly.test.js*, foram identificados diferentes *Test Smells* que comprometiam a clareza e a eficácia dos testes. Abaixo estão descritos três dos principais:

Conditional Test Logic

Foi identificado o uso de lógica condicional (if e for) dentro de um único teste, responsável por validar tanto usuários comuns quanto administradores. Esse tipo de smell torna o teste mais difícil de compreender e mascara falhas, já que múltiplos comportamentos são avaliados no mesmo caso de teste.

Viola o princípio de que cada teste deve validar apenas um comportamento isolado, o que dificulta a manutenção e reduz a confiabilidade da suíte.

Fragile Test

O teste que verificava a geração de relatórios de usuários dependia da formatação exata da string retornada. Essa prática cria testes frágeis, que podem falhar por mudanças inofensivas, como a adição de um espaço, quebra de linha ou mudança de ordem.
Em vez disso, o ideal é validar apenas o conteúdo essencial e não o formato literal, garantindo que o teste seja resistente a pequenas alterações.

Hidden Assertion

O teste que verificava a criação de usuários menores de idade utilizava um bloco try/catch sem assertiva fora da exceção. Isso significa que, se a exceção não fosse lançada, o teste ainda passaria gerando um falso positivo.
Esse tipo de smell é perigoso porque oculta falhas reais, dando uma falsa sensação de segurança sobre a correção do código testado.

Processo de Refatoração

O teste mais problemático era o que continha laços e condicionais para verificar usuários comuns e administradores.

```
Test

for (const user of todosOsUsuarios) {
  const resultado = userService.deactivateUser(user.id);
  if (!user.isAdmin) {
    expect(resultado).toBe(true);
  } else {
    expect(resultado).toBe(false);
  }
}
```

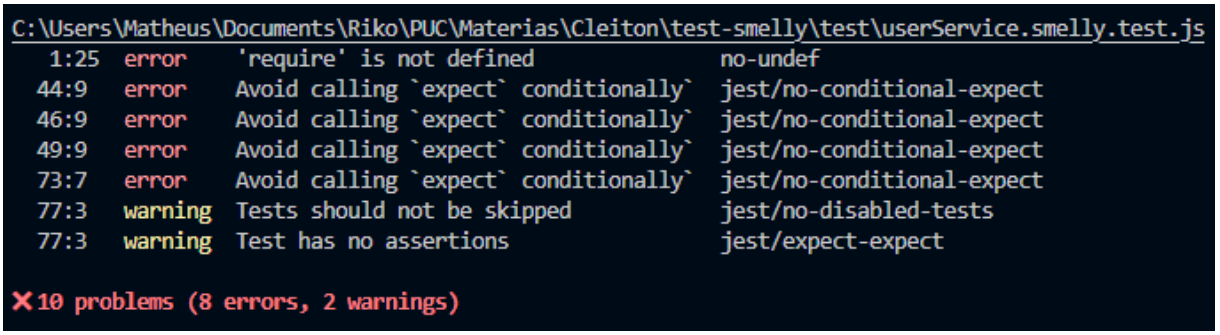
Durante a refatoração, esse teste foi dividido em dois, cada um com uma única responsabilidade, eliminando o uso de estruturas condicionais.
Além disso, o uso de blocos try/catch foi substituído por assertivas mais seguras, utilizando o método expect(...).toThrow() do Jest.

```
Test

test('deve desativar um usuário comum corretamente', () => {
  const usuarioComum = userService.createUser('Comum', 'comum@teste.com', 30);
  const resultado = userService.deactivateUser(usuarioComum.id);
  const usuarioAtualizado = userService.getUserById(usuarioComum.id);
  expect(resultado).toBe(true);
  expect(usuarioAtualizado.status).toBe('inativo');
});

test('não deve desativar um usuário administrador', () => {
  const usuarioAdmin = userService.createUser('Admin', 'admin@teste.com', 40, true);
  const resultado = userService.deactivateUser(usuarioAdmin.id);
  const usuarioAtualizado = userService.getUserById(usuarioAdmin.id);
  expect(resultado).toBe(false);
  expect(usuarioAtualizado.status).toBe('ativo');
});
```

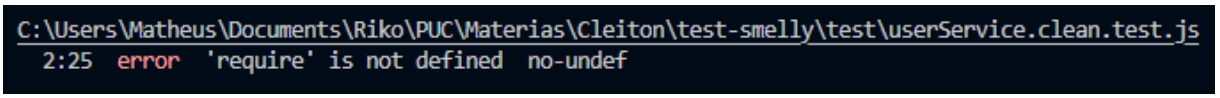
Com essa separação, os testes ficaram mais curtos, legíveis e específicos, cada um verificando um único comportamento esperado.

Relatório da Ferramenta (ESLint)

Na primeira execução do ESLint, foram detectados vários avisos e erros, incluindo o uso de testes desabilitados (.skip), condicionais em testes e blocos de código complexos.



Após a refatoração, o novo arquivo userService.clean.test.js não apresentou nenhum erro ou aviso referente aos testes, confirmando a eliminação dos *Test Smells*.



A ferramenta mostrou-se fundamental para automatizar a detecção de padrões problemáticos e reforçar boas práticas, tornando o processo de refatoração mais rápido e preciso.

Conclusão

A refatoração dos testes resultou em uma suíte mais clara, confiável e fácil de manter. A remoção dos *Test Smells* reduziu o acoplamento entre testes e implementação, enquanto a adoção do padrão AAA garantiu maior previsibilidade e legibilidade.

O uso de ferramentas de análise estática, como o ESLint, contribui diretamente para a qualidade contínua do projeto, permitindo a detecção precoce de problemas e promovendo a sustentabilidade a longo prazo do código de testes.

Em suma, testes limpos não apenas previnem regressões, mas também fortalecem a arquitetura e a confiabilidade do software como um todo.