Refatoração de Testes com Smells

Autor: João Francisco Carvalho Soares de Oliveira Queiroga

Disciplina: Teste de Software

Projeto: Refatoração de Testes e Detecção de Test Smells

Análise de Smells

1. Eager Test O teste “deve criar e buscar um usuário corretamente” realizava duas validações diferentes (criação e busca) dentro de um único caso. Isso fere o princípio de foco e clareza dos testes. O risco é que uma falha em apenas uma das ações prejudique o diagnóstico da causa do erro.

2. Conditional Test Logic O teste “deve desativar usuários se eles não forem administradores” utilizava `for` e `if` para aplicar verificações diferentes dependendo do tipo de usuário. Essa lógica condicional tornava o teste menos previsível e mais difícil de manter. Além disso, violava a regra do ESLint `jest/no-conditional-expect`.

3. Fragile Test O teste “deve gerar um relatório de usuários formatado” dependia de uma string exata com formatação rígida (`\n`, ordem fixa). Qualquer mudança mínima no formato quebraria o teste, mesmo sem alterar o comportamento funcional. Isso aumenta a fragilidade e o custo de manutenção.

Processo de Refatoração

Teste escolhido: “deve desativar usuários se eles não forem administradores”

Antes:

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

Depois:

```
test("deve desativar usuário comum e retornar true", () => {
  const usuario = userService.createUser("Comum", "comum@teste.com", 30);

  const resultado = userService.deactivateUser(usuario.id);
  const usuarioAtualizado = userService.getUserById(usuario.id);

  expect(resultado).toBe(true);
  expect(usuarioAtualizado.status).toBe("inativo");
```

```
});

test("não deve desativar usuário administrador e deve retornar false", () => {
  const admin = userService.createUser("Admin", "admin@teste.com", 40, true);

  const resultado = userService.deactivateUser(admin.id);
  const usuarioAtualizado = userService.getUserById(admin.id);

  expect(resultado).toBe(false);
  expect(usuarioAtualizado.status).toBe("ativo");
});
```

Decisões de refatoração:

- Separação em dois testes menores para eliminar lógica condicional.
- Aplicação do padrão *Arrange, Act, Assert* em cada caso.
- Melhoria na clareza dos nomes dos testes.
- A validação passou a focar em comportamento (status e retorno), e não em fluxo interno.

Relatório da Ferramenta

O ESLint apontou os seguintes problemas no arquivo original:

```
error    Avoid calling `expect` conditionally  jest/no-conditional-expect
warning  Tests should not be skipped          jest/no-disabled-tests
warning  Test has no assertions               jest/expect-expect
```

Esses alertas indicam a presença de lógica condicional nos testes, uso de `test.skip()` e ausência de verificações. A ferramenta automatizou a detecção de *smells* ao aplicar regras de boas práticas para o Jest, ajudando a localizar pontos frágeis e inconsistentes no código de teste.

Conclusão

A refatoração reduziu complexidade, aumentou legibilidade e reforçou o isolamento dos testes. O uso do padrão AAA e nomes descritivos facilita o entendimento e manutenção. A utilização de ferramentas de análise estática como o ESLint contribui significativamente para identificar *smells* e manter a qualidade contínua do código, prevenindo regressões e promovendo práticas de teste mais limpas e sustentáveis.