Refatoração de Testes e Detecção de Test Smells

1. Introdução e Contexto

No trabalho anterior, utilizamos o Teste de Mutação para provar que uma suíte de testes pode ter 100% de cobertura e, ainda assim, ser ineficaz. Vimos "mutantes" sobreviverem porque nossos testes eram fracos. Agora, vamos atacar a causa raiz desse problema: os **Test Smells**.

Um "Test Smell" (mau cheiro em teste) é um sintoma no código de teste que indica um problema mais profundo, tornando-o frágil, obscuro, lento e, consequentemente, incapaz de detectar bugs.

Neste trabalho, vocês atuarão como arquitetos de software focados em qualidade, utilizando ferramentas de **análise estática (ESLint)** para detectar Test Smells e refatorar uma suíte de testes problemática, transformando-a em um código limpo, legível e robusto.

2. Objetivos de Aprendizagem

Ao final deste trabalho, o aluno deverá ser capaz de:

- Identificar manualmente diferentes tipos de Test Smells em uma suíte de testes.
- Configurar ferramentas de análise estática (ESLint + plugins) para detectar Test Smells automaticamente.
- Interpretar os relatórios e avisos gerados pelas ferramentas de linting.
- Refatorar testes "mal cheirosos" aplicando boas práticas, como o padrão **Arrange, Act, Assert (AAA)**.
- Justificar as decisões de refatoração, explicando como elas melhoram a manutenibilidade e a eficácia dos testes.
- Demonstrar a melhoria da qualidade do código de teste através da eliminação dos avisos do linter.

3. O Projeto Base

Para este trabalho, utilizaremos um projeto que simula um serviço de gerenciamento de usuários. A suíte de testes inicial foi deliberadamente escrita com vários Test Smells clássicos.

Repositório do Projeto: https://github.com/CleitonSilvaT/test-smelly

O repositório contém:

- Um diretório `src/` com a lógica de negócio (`userService.js`).
- Um diretório `__tests__/` com uma suíte de testes **deliberadamente "mal cheirosa"** (`userService.smelly.test.js`).
- Um arquivo `package.json` com as dependências básicas.

4. Etapas do Trabalho

Etapa 1: Preparação do Ambiente

1. Faça um **fork** do repositório para sua conta do GitHub.
2. Clone o seu fork para sua máquina local.

3. Instale as dependências do projeto com o comando: `npm install`.
4. Execute a suíte de testes inicial para confirmar que, apesar dos "maus cheiros", os testes estão passando: `npm test`.

Etapa 2: Análise Manual (Caça aos Smells)

1. Abra o arquivo `__tests__/userService.smelly.test.js`.
2. Leia atentamente cada teste. Com base no que aprendemos em aula, tente identificar manualmente pelo menos **3 tipos diferentes de Test Smells** (ex: Teste Frágil, Lógica Condicional, Convidado Misterioso, etc.). Anote suas observações.

Etapa 3: Configuração da Ferramenta de Detecção (ESLint)

1. Adicione as dependências do ESLint e do plugin do Jest ao projeto:

```
npm install --save-dev eslint eslint-plugin-jest
```

2. Crie o arquivo de configuração do ESLint na raiz do projeto. Crie um arquivo chamado `.eslintrc.json` e adicione o seguinte conteúdo:

```
JSON
{
  "env": {
    "es2021": true,
    "node": true,
    "jest/globals": true
  },
  "extends": [
    "eslint:recommended",
    "plugin:jest/recommended"
  ],
  "parserOptions": {
    "ecmaVersion": "latest",
    "sourceType": "module"
  },
  "plugins": [
    "jest"
  ],
  "rules": {
    "jest/no-disabled-tests": "warn",
    "jest/no-conditional-expect": "error",
    "jest/no-identical-title": "error"
  }
}
```

Etapa 4: Detecção Automática e Análise

1. Execute o ESLint no seu terminal para analisar todo o projeto:

```
npx eslint .
```

2. Analise o resultado. O ESLint deve reportar vários avisos e erros no arquivo `userService.smelly.test.js`. Compare os problemas encontrados pela ferramenta com as suas anotações da análise manual.

Etapa 5: Refatoração para Testes Limpos

1. **Não modifique o arquivo original.** Crie uma cópia dele chamada `__tests__/userService.clean.test.js`.
2. Seu objetivo é refatorar todos os testes neste novo arquivo para que eles fiquem "limpos" e sem smells.
    - Aplique rigorosamente o padrão **Arrange, Act, Assert (AAA)**.
    - Separe testes gigantes (Eager Test) em testes menores e focados.
    - Remova qualquer lógica condicional (`if`, `for`).
    - Torne os testes menos frágeis, verificando comportamentos em vez de implementações exatas.
    - Dê nomes claros e descritivos para cada teste.

Etapa 6: Validação Final

1. Execute o ESLint novamente. O linter não deve apontar **nenhum erro ou aviso** no seu novo arquivo `userService.clean.test.js`.
2. Execute a suíte de testes completa (`npm test`). Todos os testes (tanto do arquivo antigo quanto do novo) devem passar, provando que sua refatoração não quebrou a funcionalidade.
3. Faça o commit e o push das suas alterações, incluindo o novo arquivo de teste limpo, para o seu repositório no GitHub.

5. O Que Entregar

1. **Link do Repositório GitHub:** O link para o seu fork do projeto, contendo o novo arquivo `__tests__/userService.clean.test.js` com os testes refatorados.
2. **Relatório Escrito (PDF, 2-4 páginas):** Um documento contendo:
    - **Capa:** Nome da disciplina, nome do trabalho, seu nome completo e matrícula.
    - **Análise de Smells:** Descreva 3 Test Smells que você encontrou na suíte de testes original. Para cada um, explique por que ele é considerado um "mau cheiro" e qual o risco que ele representa.
    - **Processo de Refatoração:** Escolha um dos testes mais problemáticos e mostre o "Antes" (código do arquivo `smelly`) e o "Depois" (código do arquivo

clean). Explique as decisões que você tomou na refatoração e como elas corrigiram os smells identificados.
    - **Relatório da Ferramenta:** Inclua um print do resultado do ESLint na primeira execução (mostrando os erros) e comente como a ferramenta automatizou a detecção.
    - **Conclusão:** Uma breve reflexão sobre como a escrita de testes limpos e a utilização de ferramentas de análise estática contribuem para a qualidade e sustentabilidade de um projeto de software.

6. Critérios de Avaliação

- **Identificação de Smells (20%):** A capacidade de analisar e descrever corretamente os problemas na suíte de testes original.
- **Configuração da Ferramenta (15%):** O ESLint foi configurado e executado corretamente no projeto.
- **Qualidade da Refatoração (40%):** Os testes foram efetivamente corrigidos, seguindo o padrão AAA e eliminando os smells de forma clara e objetiva.
- **Qualidade do Relatório Escrito (25%):** Clareza, objetividade, estrutura e cumprimento de todos os itens solicitados na entrega.

7. Recursos Adicionais

- **Plugin ESLint para Jest (Documentação):** https://github.com/jest-community/eslint-plugin-jest
- **Artigo de Referência (Martin Fowler):** https://martinfowler.com/bliki/TestSmell.html