PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

GABRIEL HENRIQUE MOTA RODRIGUES - 814411

TESTES MUTAÇÕES

BELO HORIZONTE - 2025

1. ANÁLISE INICIAL

1.1 Cobertura de Código Inicial

A execução inicial da suíte de testes apresentou as seguintes métricas de cobertura:

File	%Stmts	%Branch	%Funcs	%Lines	Uncovered
operacoes.js	85.41	58.82	100	98.64	112

Tabela 1 - Métricas de Cobertura Inicial

1.2 PONTUAÇÃO DE MUTAÇÃO INICIAL

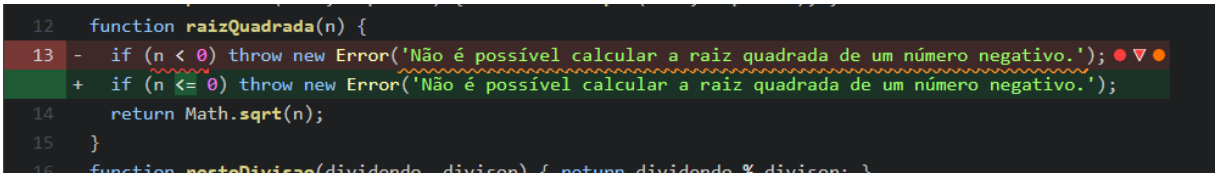
A execução inicial do teste de mutação com o Stryker apresentou os seguintes resultados:

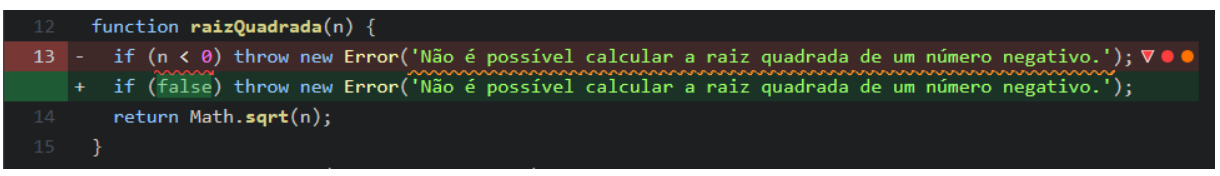
- **Mutation Score Total:** 73.71% - Apenas 3 em cada 4 mutantes foram detectados
- **Mutation Score Covered:** 78.11% - Considerando apenas código coberto
- **Mutantes Mortos**: 153 - Mutações detectadas pelos testes
- **Mutantes Sobreviventes**: 44 - Mutações NÃO detectadas (21.7% do total)
- **Mutantes sem Cobertura**: 12 - Código não executado pelos testes

A pontuação de mutação de 73.71% confirma que, apesar da cobertura de linhas de 98.64%, mais de 1 em cada 4 mutantes sobreviveu (26.29%). Isso significa que aproximadamente 26% dos bugs potenciais passariam despercebidos pela suíte de testes original.

2. ANÁLISE DE MUTANTES CRÍTICOS

2.1. raizQuadrada





Na função raizQuadrada, a mutação substituiu a verificação de n < 0 por false, ou seja,nunca lança erro, mesmo para números negativos. Outra mutação foi a substituição de < para <=. O teste original foi incapaz de matar o mutante pois ele não contemplava uma verificação específica para a entrada 0.

2.2. isPrimo

```
73    function isPrimo(n) {
74  -   if (n <= 1) return false;
    +   if (n < 1) return false;
75      for (let i = 2; i < n; i++) {
76        if (n % i === 0) return false;
77      }
78      return true;
79    }
```

```
73    function isPrimo(n) {
74      if (n <= 1) return false;
75      for (let i = 2; i < n; i++) {
76  -     if (n % i === 0) return false;
    +     if (n * i === 0) return false;
77      }
78      return true;
79    }
```

Na função isPrimo, a mutação substituiu a verificação <= 1 por < 1, ou seja, parou de contemplar o 1 como entrada inválida. Outro ponto foi a substituição do % para * invalidando completamente o propósito da função. Os testes iniciais somente validam se um número positivo > 1 é primo.

3. SOLUÇÃO IMPLEMENTADA.

3.1 raizQuadrada

O teste da função raizQuadrada foi implementado de forma a validar tanto o comportamento correto para números válidos quanto a manipulação de entradas inválidas. Em seguida, o teste confirma que a função lança corretamente uma exceção ao receber um número negativo (-2), garantindo que o erro 'Não é possível

calcular a raiz quadrada de um número negativo.' seja disparado. Além disso, foram incluídas verificações para assegurar que valores não negativos, como 16 e 0, não disparam exceção, confirmando a consistência do comportamento da função em entradas válidas e reforçando a cobertura de casos de borda.

3.2. isPrimo

O teste da função isPrimo foi projetado para verificar corretamente a classificação de números primos e não primos. Ele confirma que números primos, como 7, retornam true, enquanto números compostos, como 10, e valores especiais, incluindo 0, 1 e entradas inválidas como false, retornam false. Essa abordagem garante a validação tanto de casos típicos quanto de valores de borda, aumentando a robustez da função frente a diferentes tipos de entrada.

4. RESULTADOS

A execução do Stryker sobre operacoes.js obteve um Mutation Score de 99,52%, com 203 mutantes mortos e apenas 1 sobrevivente. A análise indica que a suíte de testes é eficaz.

File	Mutation Score		Killed	Survived	Ignored
	Total	Covered			
operacoes.js	99.52%	99.52%	203	1	6

5. CONCLUSÃO

Os resultados da análise de mutação demonstram que a suíte de testes implementada para as funções de operacoes.js apresenta alta eficácia, sendo capaz de detectar a maioria das falhas introduzidas artificialmente pelo Stryker. A detecção de 203 mutantes mortos entre 213 gerados indica robustez e boa cobertura de casos típicos e valores esperados. Entretanto, a sobrevivência de mutantes evidencia lacunas na verificação de casos de borda e condições de exceção. Portanto, embora o conjunto de testes seja amplamente confiável, a inclusão de cenários extremos e entradas inválidas poderia fortalecer ainda mais a confiabilidade das funções, garantindo que todas as situações críticas sejam devidamente testadas.