PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA
GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

LORRAYNE MARAYZE SILVA DE OLIVEIRA - 816603

ANÁLISE DE EFICÁCIA DE TESTES COM TESTE DE MUTAÇÃO

TESTE DE SOFTWARE

BELO HORIZONTE
2025

ANÁLISE INICIAL

O objetivo principal desta etapa foi avaliar a qualidade da suíte de testes existente através de duas métricas complementares: cobertura de código e mutation score.

A Figura 1 apresenta o resultado da execução do comando npm test no terminal. Observa-se que todos os 50 testes da suíte foram aprovados com sucesso, indicando que não há falhas aparentes no código testado. A métrica Stmts% (Statement Coverage) registrou 85,41%, significando que aproximadamente 85% das linhas de código executáveis foram percorridas durante a execução dos testes.

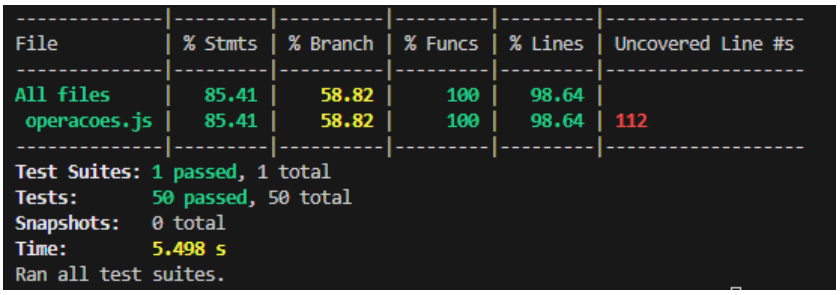


Figura 1. Resultado da execução da suíte de testes com npm test

A Figura 2 demonstra o resultado da análise de mutação realizada pelo Stryker, um framework que introduz modificações sistemáticas no código fonte (mutantes) para verificar se os testes são capazes de detectá-las. O Mutation Score Total Covered obtido foi de 73,71%, indicando que aproximadamente 74% dos mutantes gerados foram detectados, ou seja, “mortos” pelos testes, enquanto 26% sobreviveram.

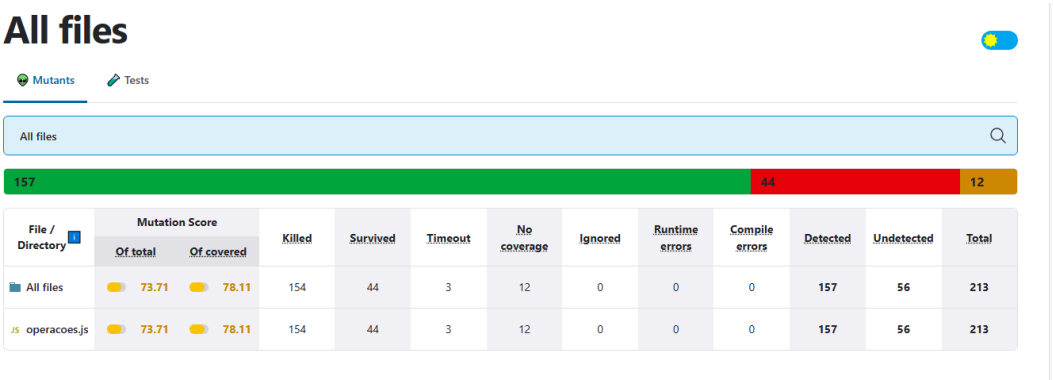


Figura 2. Relatório do Stryker com o Mutation Score inicial

Há uma diferença entre as duas métricas obtidas: a cobertura de código está em 85,41%, enquanto o mutation score ficou menor em 73,71%. Essa diferença pode se dar devido a questão de que executar uma linha de código não é o mesmo que testá-la, ou seja, quando um teste passa por um trecho de código sem verificar se o resultado está correto, ele contribui para a cobertura, mas não garante que aquele código está funcionando como deveria.

ANÁLISE DE MUTANTES CRÍTICOS

Na mutação da função clamp, o Stryker modificou as comparações que verificam se o valor está fora do intervalo definido, alterando a condição que testava se o valor era menor que o mínimo para incluir o

caso de igualdade ou para sempre retornar falso, e fez o mesmo com a verificação do valor maior que o máximo. Essas mudanças afetaram a lógica de limite, mas o teste existente apenas verificava uma situação em que o valor já estava dentro do intervalo. Dessa forma, mesmo com as mutações, o comportamento observado no teste permaneceu o mesmo, por isso, os mutantes sobreviveram: o teste não explorava os casos de borda, e acabou não exercitando as condições alteradas.

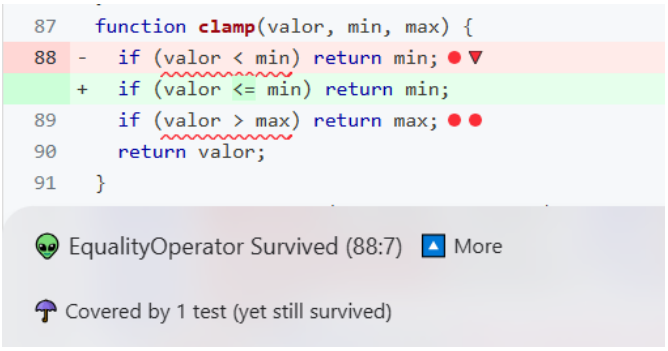


Figura 3. Mutantes na Função Clamp

Uma mutação também ocorreu na função de fatorial, onde o Stryker alterou as condições que tratam valores negativos e os casos base em que o número é zero ou um. Ele modificou o operador de comparação para incluir o zero como negativo, além de transformar a condição dos casos base em expressões incorretas ou sempre falsas. Apesar dessas alterações, o teste disponível apenas verificava o cálculo do fatorial para um número maior que um, não abordando as situações em que o argumento é zero, um ou negativo. Assim, as mutações sobreviveram porque o teste não exercitava os casos limites e as exceções da função.

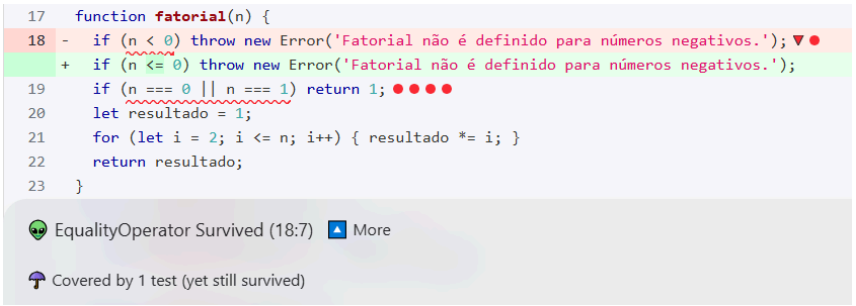


Figura 4. Mutantes na Função Fatorial

SOLUÇÃO IMPLEMENTADA

Na função clamp, foram criados vários cenários para cobrir situações que antes não eram testadas. Antes, o teste só verificava quando o valor estava dentro do intervalo, então as mutações que alteravam as comparações de limite não eram detectadas. Agora, os testes incluem valores abaixo do mínimo, acima do máximo, casos com limites invertidos e até limites negativos. Esses casos garantem que o comportamento da função seja validado em todas as condições, inclusive quando *min* é maior que *max*, o que exigiu a adição da linha que inverte os limites dentro da função.

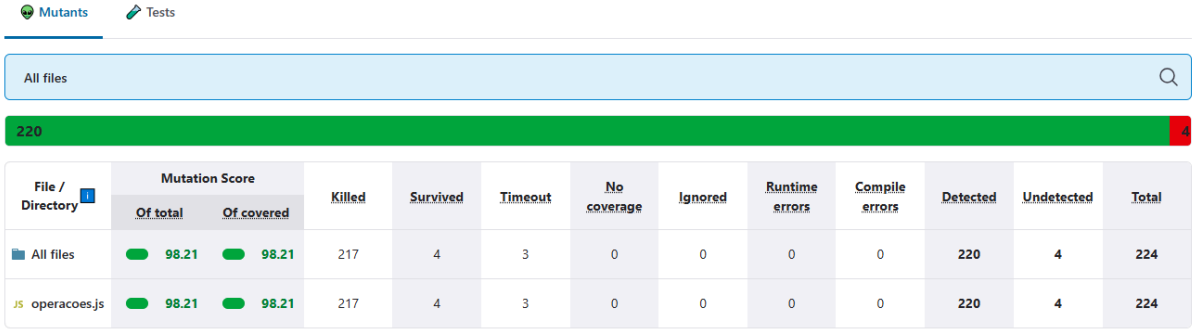
Os novos testes da função fatorial, foram adicionados cenários que cobrem todos os pontos críticos que as mutações alteravam. Além de manter os casos com números maiores que um, foram criados testes específicos para o caso base, para um número pequeno, e para entradas inválidas, como números

negativos. Esses novos testes fazem com que cada condição da função seja validada, assim se o operador de comparação for alterado ou se as condições base forem modificadas, os testes falharão, garantindo que as mutações que antes sobreviviam agora sejam detectadas corretamente.

RESULTADOS FINAIS

Após a implementação dos novos testes, houve uma melhora tanto na cobertura de código quanto no mutation score. A cobertura, que inicialmente era de 85,41%, passou para 100% em todas as métricas (statements, branches, functions e lines), indicando que todas as partes do código foram completamente exercitadas pelos testes. Além disso, o Mutation Score Total Covered atingiu 98,21%, mostrando que a maior parte dos mutantes gerados pelo Stryker foram corretamente detectados e eliminados.

Esses resultados mostram que os novos testes foram eficazes em validar o comportamento esperado das funções, especialmente nos casos de borda e exceção que antes não eram testados. O conjunto final contou com 99 testes.



File / Directory	Mutation Score		Killed	Survived	Timeout	No coverage	Ignored	Runtime errors	Compile errors	Detected	Undetected	Total
	Of total	Of covered										
All files	98.21	98.21	217	4	3	0	0	0	0	220	4	224
operacoes.js	98.21	98.21	217	4	3	0	0	0	0	220	4	224

Figura 5. Stryker Mutation Score

File / Directory	Killing	Covering	Not Covering	Total tests
All tests	72	27	0	99
operacoes.test.js	72	27	0	99

Figura 6. Stryker Tests

CONCLUSÃO

Os resultados mostram que o teste de mutação é importante para avaliar a qualidade real dos testes. A análise inicial mostrou que a cobertura de código não garante que os testes sejam bons: começamos com 85,41% de cobertura, mas apenas 73,71% de mutation score, indicando que muitos testes não validavam o código adequadamente.

Após analisar os mutantes que sobreviveram e criar novos testes focados nesses pontos fracos, foi possível chegar a 100% de cobertura e 98,21% de mutation score. Essa melhoria aconteceu porque os novos testes não verificaram apenas se o código funciona, mas também testaram casos extremos, valores limites e situações de erro.

O teste de mutação se mostrou necessário para identificar onde os testes precisam melhorar, colaborando para a criação de uma suíte de testes mais robusta.