Universidade Federal do Ceará

TESTES ESTÁTICOS E FUNCIONAIS

Processamento Digital de Imagens

Fortaleza
13/02/2025

4.Testes

4.1 Teste Estático

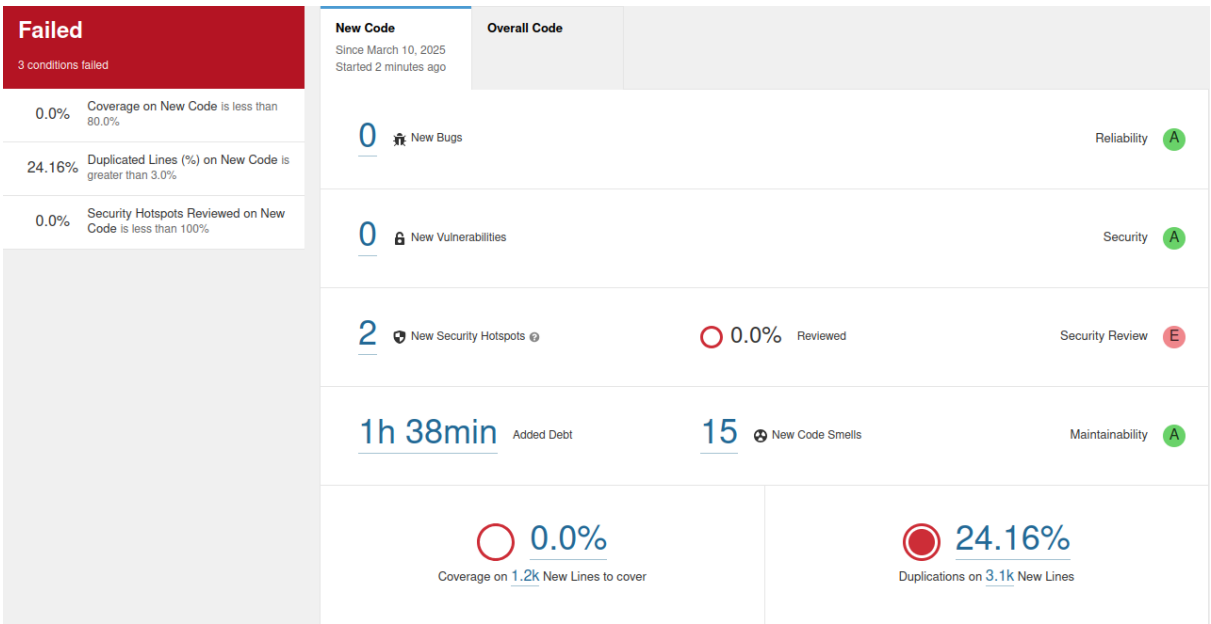
O procedimento de teste estático será realizado através da análise do código por meio da ferramenta SonarQube para identificação de possíveis bugs e vulnerabilidades com o propósito de produzir um relatório de teste estático.

4.1.1 Descrição da Ferramenta

O SonarQube é uma ferramenta de análise estática de código amplamente utilizada para identificar e resolver problemas de qualidade no software, como bugs, vulnerabilidades de segurança e más práticas de codificação. A ferramenta utiliza um conjunto de regras predefinidas para avaliar o código com base em critérios de legibilidade, segurança, desempenho e manutenção. Quando o SonarQube encontra problemas, ele os classifica em categorias como Security, Reliability, Maintainability, Coverage, Duplications entre diversas outras métricas. Cada problema identificado é classificado com um nível de severidade, o que ajuda os desenvolvedores a priorizar as correções.

4.1.2 Resultados dos testes

Backend



A análise do SonarQube identificou três condições falhas que impediram a aprovação do código. Os principais problemas detectados são:

1. Cobertura de Código

- A cobertura de testes no novo código está em **0.0%**, enquanto o mínimo exigido é **80%**.

- Isso indica que não há testes automatizados cobrindo as novas implementações, o que pode comprometer a qualidade e a confiabilidade do sistema.

2. Código Duplicado

- O percentual de duplicação no novo código é de **24.16%**, bem acima do limite aceitável de **3%**.
- A presença de código duplicado pode aumentar a complexidade e dificultar a manutenção futura.

3. Revisão de Security Hotspots

- Foram detectados **2 novos security hotspots**, porém **0%** deles foram revisados.
- A falta de revisão desses pontos pode representar riscos à segurança do sistema.

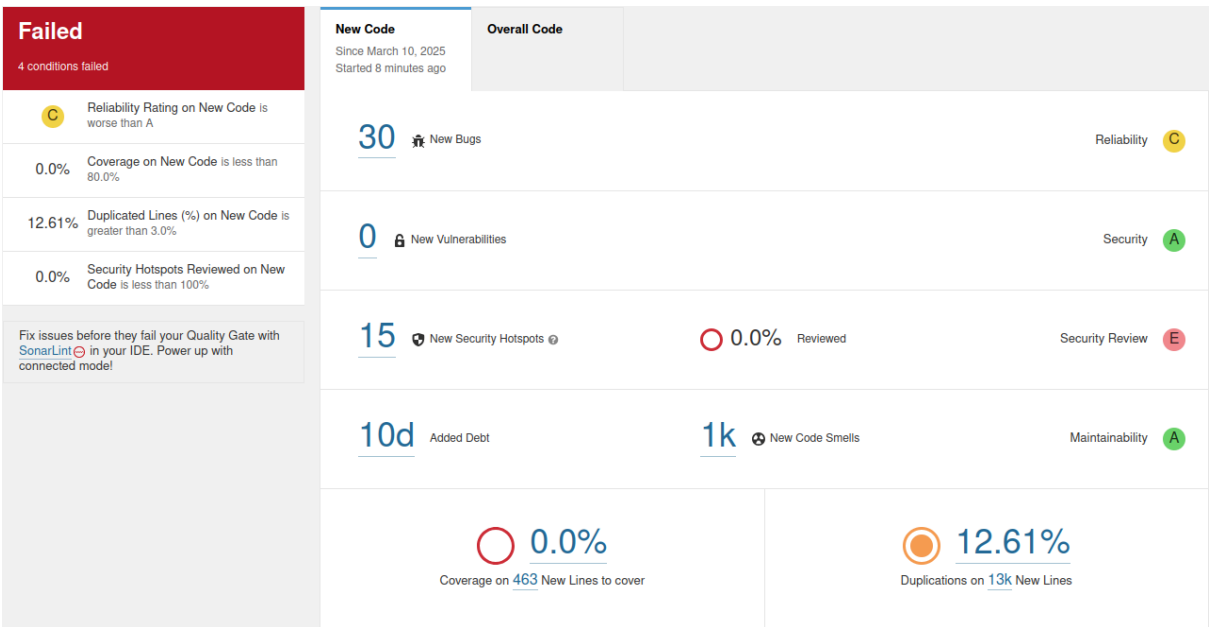
Pontos Positivos

- O código recebeu avaliação **A** em **Confiabilidade, Segurança e Manutenibilidade**, indicando que não foram identificados bugs críticos ou vulnerabilidades diretas.

Recomendações para Correção

1. **Aumentar a cobertura de testes** com testes unitários e de integração para atingir o percentual mínimo exigido.
2. **Reduzir duplicações** por meio da refatoração de trechos redundantes no código.
3. **Realizar a revisão dos security hotspots** para avaliar se representam riscos reais e, se necessário, implementar correções.

Frontend



A análise do SonarQube indicou falhas em quatro condições, resultando na reprovação do código. Os principais problemas identificados são:

1. Confiabilidade

- O código recebeu uma avaliação **C** em confiabilidade, inferior ao exigido (**A**).
- Foram detectados **30 novos bugs**, o que pode comprometer o funcionamento correto do sistema.

2. Cobertura de Código

- A cobertura de testes nas novas linhas é de **0.0%**, enquanto o mínimo exigido é **80%**.
- Isso significa que nenhuma linha do novo código foi testada, aumentando os riscos de falhas na produção.

3. Código Duplicado

- **12.61%** do novo código contém duplicações, acima do limite permitido de **3%**.
- A alta taxa de duplicação pode dificultar a manutenção e introduzir inconsistências no sistema.

4. Revisão de Security Hotspots

- Foram identificados **15 novos security hotspots**, mas **0%** foram revisados.
- Isso pode indicar possíveis riscos de segurança que ainda não foram avaliados.

Outros Indicadores

- **Dívida técnica:** Estimada em **10 dias** de esforço para correções.
- **Code Smells:** Foram identificados **1.000 code smells**, o que pode afetar a legibilidade e a manutenibilidade do código.

Recomendações para Correção

1. **Melhorar a confiabilidade** revisando e corrigindo os **30 bugs identificados**.
2. **Aumentar a cobertura de testes** incluindo testes unitários e de integração para atingir pelo menos **80% de cobertura**.
3. **Reduzir a duplicação de código**, refatorando trechos redundantes para melhorar a manutenção.
4. **Revisar os security hotspots** para garantir que não representem riscos ao sistema.

4.2 Teste Funcional

Cada teste tem uma tabela com as devidas informações conforme o modelo abaixo:

ID do RF/RNF	Identificador do requisito funcional relacionado
Caso de Teste	Identificador do caso de teste especificado.
Nome do Caso de Teste	Nome do caso de teste especificado

Precondição	Condições necessárias para realizar o teste
Dados de Entrada	Dados que o testador deve fornecer ao sistema. Ex: usuário e senha
Passo a Passo	Passo a passo para a execução do caso de teste
Resultado Esperado	O que é esperado que o sistema retorne ao final do teste
Técnica de Teste Funcional	Escolher a técnica mais adequada para testar o caso de teste
Resultado Atual	Resultado que o sistema retornou ao executar o teste.

- Os 14 testes que foram implementados passaram.

Teste: test_classificacao.py
Função: teste_calcula_ocorrencias_classes

ID do RF/RNF	RF005
Caso de Teste	CT001
Nome do Caso de Teste	Teste de cálculo de ocorrências das classes
Precondição	O sistema deve estar pronto para processar a imagem de entrada.
Dados de Entrada	Imagem de exemplo com valores de intensidade de pixel variando entre valores positivos e negativos.
Passo a Passo	1. A função `calcula_ocorrencias_classes` é chamada com uma amostra de imagem. 2. O sistema calcula as ocorrências das classes na imagem. 3. O resultado é comparado para verificar se as ocorrências são corretamente contabilizadas.
Resultado Esperado	A função retorna uma matriz de ocorrências com a forma correta e os valores esperados para as classes.

Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	A função retornou o formato correto e valores válidos.

Teste: test_classificacao.py
Função: teste_probabilidade_classes

ID do RF/RNF	RF005
Caso de Teste	CT002
Nome do Caso de Teste	Teste de cálculo da probabilidade das classes
Precondição	O sistema deve estar pronto para calcular a probabilidade a partir das ocorrências.
Dados de Entrada	Matriz de ocorrências das classes, como especificado no teste anterior.
Passo a Passo	1. A função `probabilidade_classes` é chamada com a matriz de ocorrências. 2. O sistema calcula as probabilidades de cada classe para cada pixel. 3. A soma das probabilidades por pixel deve ser 1, com exceção dos pixels sem ocorrências.
Resultado Esperado	A soma das probabilidades por pixel é igual a 1.0. Para pixels sem ocorrências, a probabilidade deve ser 0.2.
Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	O cálculo de probabilidades foi realizado corretamente, com a soma das probabilidades igual a 1.

Teste: test_curva.py
Função: test_inicializa_curva_padrao

ID do RF/RNF	RF016
Caso de Teste	CT003
Nome do Caso de Teste	Teste de inicialização de curva com ponto padrão
Precondição	O sistema deve estar pronto para inicializar uma curva
Dados de Entrada	Ponto de inicialização [50, 50].
Passo a Passo	1. A função `inicializa_curva` é chamada com o ponto [50, 50]. 2. O sistema inicializa a curva com 30 pontos e as coordenadas adequadas.
Resultado Esperado	A curva deve ter 30 pontos, com todas as coordenadas maiores que 0.
Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	A curva foi inicializada corretamente com 30 pontos.

Teste: test_curva.py
Função: test_inicializa_curva_pontos_deslocados

ID do RF/RNF	RF016
Caso de Teste	CT004
Nome do Caso de Teste	Teste de inicialização de curva com pontos deslocados
Precondição	O sistema deve estar pronto para inicializar uma curva com ponto deslocado.
Dados de Entrada	Ponto de inicialização [100, 100], raio 20, 50 pixels.

Passo a Passo	1. A função `inicializa_curva` é chamada com o ponto [100, 100], raio 20 e 50 pixels. 2. O sistema inicializa a curva de acordo com as condições fornecidas.
Resultado Esperado	A curva deve ter 50 pontos.
Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	A curva foi inicializada corretamente com 50 pontos.

Teste: test_curva.py
Função: test_inicializa_curva_pontos_negativos

ID do RF/RNF	RF016
Caso de Teste	CT005
Nome do Caso de Teste	Teste de inicialização de curva com pontos negativos
Precondição	O sistema deve estar pronto para detectar erros ao inicializar a curva.
Dados de Entrada	Ponto de inicialização [5, 5] com raio 30.
Passo a Passo	1. A função `inicializa_curva` é chamada com o ponto [5, 5] e raio 30. 2. O sistema tenta inicializar a curva.
Resultado Esperado	O sistema lança um erro, indicando que os pontos devem ser maiores que 0.
Técnica de Teste Funcional	Teste de Caixa Preta

Resultado Atual	O erro foi corretamente lançado.

Teste: test_curva.py
Função: test_inicializa_curva_qtd_pontos_negativo

ID do RF/RNF	RF016
Caso de Teste	CT006
Nome do Caso de Teste	Teste de inicialização de curva com quantidade de pontos negativa
Precondição	O sistema deve estar pronto para validar a quantidade de pontos da curva.
Dados de Entrada	Ponto de inicialização [5, 5], raio 30, quantidade de pixels -3.
Passo a Passo	1. A função `inicializa_curva` é chamada com o ponto [5, 5], raio 30 e quantidade de pixels -3. 2. O sistema valida a entrada e lança um erro.
Resultado Esperado	O sistema deve lançar um erro, indicando que a quantidade de pixels não pode ser negativa.
Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	O erro foi corretamente lançado.

Teste: test_curva.py
Função: test_inicializa_curva_quantidade_minima_pixels

ID do RF/RNF	RF016
Caso de Teste	CT007

Nome do Caso de Teste	Teste de inicialização de curva com a quantidade mínima de pontos
Precondição	O sistema deve estar pronto para inicializar uma curva com a quantidade mínima de pontos.
Dados de Entrada	Ponto de inicialização [50, 50], quantidade de pixels 2.
Passo a Passo	1. A função `inicializa_curva` é chamada com o ponto [50, 50] e quantidade de pixels 2. 2. O sistema inicializa a curva com 2 pontos.
Resultado Esperado	A curva deve ser inicializada com 2 pontos.
Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	A curva foi inicializada corretamente com 2 pontos.

Teste: test_curva.py
Função: test_inicializa_curva_raio_zero

ID do RF/RNF	RF016
Caso de Teste	CT008
Nome do Caso de Teste	Teste de inicialização de curva com raio zero
Precondição	O sistema deve estar pronto para inicializar uma curva com raio zero.
Dados de Entrada	Ponto de inicialização [30, 30], raio 0.

Passo a Passo	1. A função `inicializa_curva` é chamada com o ponto [30, 30] e raio 0. 2. O sistema inicializa a curva com todas as coordenadas iguais ao ponto de inicialização.
Resultado Esperado	A curva deve ter 30 pontos com todas as coordenadas iguais ao ponto [30, 30].
Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	A curva foi inicializada corretamente com todas as coordenadas iguais ao ponto [30, 30].

Teste: test_forca.py
Função: test_zero_distancia

ID do RF/RNF	RF007
Caso de Teste	CT009
Nome do Caso de Teste	Teste de força de continuidade com distância zero
Precondição	O sistema deve estar pronto para calcular a força de continuidade com um conjunto de pontos.
Dados de Entrada	Conjunto de pontos com coordenadas [[0, 0], [0, 0], [0, 0]].
Passo a Passo	1. A função `forca_continuidade` é chamada com os pontos [[0, 0], [0, 0], [0, 0]] e parâmetro de distância 1. 2. O sistema calcula a força de continuidade.
Resultado Esperado	O valor calculado deve ser 0.0.
Técnica de Teste Funcional	Teste de Caixa Preta

Resultado Atual	O valor calculado foi 0.0.

Teste: test_forca.py
Função: test_pontos_linear

ID do RF/RNF	RF007
Caso de Teste	CT010
Nome do Caso de Teste	Teste de força de continuidade com pontos lineares
Precondição	O sistema deve estar pronto para calcular a força de continuidade com um conjunto de pontos lineares.
Dados de Entrada	Conjunto de pontos [[0, 0], [1, 0], [2, 0], [3, 0]].
Passo a Passo	1. A função `forca_continuidade` é chamada com os pontos [[0, 0], [1, 0], [2, 0], [3, 0]] e distância 2. 2. O sistema calcula a força de continuidade.
Resultado Esperado	O valor calculado deve ser maior ou igual a 0.
Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	O valor calculado foi válido e maior que 0.

Teste: test_forca.py
Função: test_tipo_saida

ID do RF/RNF	RF007
Caso de Teste	CT011
Nome do Caso de Teste	Teste de tipo de saída da função `forca_continuidade`

Precondição	O sistema deve estar pronto para calcular a força de continuidade.
Dados de Entrada	Conjunto de pontos gerado aleatoriamente com 5 pontos.
Passo a Passo	1. A função `forca_continuidade` é chamada com os pontos aleatórios e distância 3. 2. O sistema calcula a força de continuidade.
Resultado Esperado	A saída deve ser do tipo `np.float64`.
Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	A saída foi do tipo correto.

Teste: test_forca.py
Função: test_ultimo_ponto

ID do RF/RNF	RF007
Caso de Teste	CT012
Nome do Caso de Teste	Teste de força de continuidade com o último ponto
Precondição	O sistema deve estar pronto para calcular a força de continuidade.
Dados de Entrada	Conjunto de pontos [[0, 0], [1, 1], [2, 2]].
Passo a Passo	1. A função `forca_continuidade` é chamada com os pontos [[0, 0], [1, 1], [2, 2]] e distância 2. 2. O sistema calcula a força de continuidade.

Resultado Esperado	O valor calculado deve ser maior ou igual a 0.
Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	O valor calculado foi válido e maior que 0.

Teste: test_forca.py
Função: test_triangulo

ID do RF/RNF	RF007
Caso de Teste	CT013
Nome do Caso de Teste	Teste de força de continuidade com pontos formando um triângulo
Precondição	O sistema deve estar pronto para calcular a força de continuidade.
Dados de Entrada	Conjunto de pontos formando um triângulo [[0, 0], [4, 0], [4, 3]].
Passo a Passo	1. A função `forca_continuidade` é chamada com os pontos [[0, 0], [4, 0], [4, 3]] e distância 0. 2. O sistema calcula a força de continuidade.
Resultado Esperado	O valor calculado deve ser 1.0.
Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	O valor calculado foi 1.0.

Teste: test_forca.py
Função: test_quadrado

ID do RF/RNF	RF007
Caso de Teste	CT014
Nome do Caso de Teste	Teste de força de continuidade com pontos formando um quadrado
Precondição	O sistema deve estar pronto para calcular a força de continuidade.
Dados de Entrada	Conjunto de pontos formando um quadrado [[0, 0], [4, 0], [4, 4], [0, 4]].
Passo a Passo	1. A função `forca_continuidade` é chamada com os pontos [[0, 0], [4, 0], [4, 4], [0, 4]] e distância 0. 2. O sistema calcula a força de continuidade.
Resultado Esperado	O resultado deve ser 0.0.
Técnica de Teste Funcional	Teste de Caixa Preta
Resultado Atual	O resultado foi correto.