# DETECÇÃO DE REGRESSÕES VISUAIS EM INTEGRAÇÃO CONTÍNUA: IMPLEMENTAÇÃO E ANÁLISE DE TÉCNICAS PARA DETECTAR FALHAS DE INTERFACE

Douglas Martignago Rosso[1], Ana Cláudia Garcia Barbosa[2]

**Resumo:** Ciclos curtos de entrega tornaram-se padrão em equipes de desenvolvimento web com a adoção de metodologias ágeis e processos de integração e entrega contínuas. Novas funcionalidades, correções e ajustes de interface são implementados com frequência cada vez maior, o que torna inviável a verificação visual manual, especialmente em aplicações com múltiplas telas e resoluções. Pequenas alterações em elementos como cores, espaçamentos, alinhamentos e tipografia podem gerar regressões visuais que afetam a experiência do usuário, mas que testes funcionais tradicionais não capturam. Este trabalho compara três técnicas de regressão visual automatizada: comparação pixel a pixel, similaridade estrutural e abordagem por regiões. As verificações foram automatizadas em fluxo contínuo, com captura automática de imagens de referência a partir do ramo principal e geração de relatórios para revisão dos resultados. Os experimentos cobriram cenários de alteração controlada na interface, incluindo uma situação sem modificações como controle, em diferentes tamanhos de tela, para avaliar o comportamento das técnicas em contextos de responsividade. Os resultados permitiram analisar a capacidade de detecção de regressões visuais, a ocorrência de alertas indevidos e o esforço de validação diante de variações de renderização, alterações de layout e conteúdo dinâmico. Combinar as três técnicas num fluxo único ampliou a cobertura e compensou as limitações individuais de cada abordagem, reduzindo o risco de regressões passarem sem alerta.

**Palavras-chave:** regressão visual; integração contínua; testes automatizados; qualidade de interfaces; automação de testes; comparação de imagens.

**ABSTRACT:** Short delivery cycles have become standard in web develop-

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), Criciúma - Santa Catarina - Brasil. douglasrosso10@gmail.com

[2]Orientadora. Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), Criciúma - Santa Catarina - Brasil. agb@unesc.net

ment teams with the adoption of agile methodologies and continuous integration and delivery processes. New features, fixes, and interface adjustments are implemented with increasing frequency, making manual visual verification impractical, especially in applications with multiple screens and resolutions. Small changes in elements such as colors, spacing, alignment, and typography can generate visual regressions that affect the user experience but are not captured by traditional functional tests. This work compares three automated visual regression techniques: pixel-by-pixel comparison, structural similarity, and a region-based approach. The checks were automated in a continuous workflow, with automatic capture of reference images from the main branch and report generation for result review. The experiments covered controlled interface change scenarios, including an unmodified situation as a control, across different screen sizes, to evaluate the behavior of the techniques in responsive contexts. The results enabled analysis of the visual regression detection capability, the occurrence of false alerts, and the validation effort in the presence of rendering variations, layout changes, and dynamic content. Combining the three techniques in a single workflow broadened coverage and compensated for the individual limitations of each approach, reducing the risk of regressions going undetected.

**Keywords:** visual regression; continuous integration; automated testing; interface quality; test automation; image comparison.

## 1 INTRODUÇÃO

A qualidade visual de interfaces web ganhou relevância prática à medida que equipes passaram a adotar ciclos curtos de entrega. Um ajuste de espaçamento, uma troca de fonte ou uma simples alteração de cor pode modificar a aparência da interface sem tocar na lógica funcional, e esse tipo de mudança costuma passar despercebido até que alguém aponte o problema em produção. Quando a versão atual diverge da referência aprovada, configura-se uma regressão visual: uma diferença indesejada que só aparece pela comparação de capturas de tela. Verificar manualmente cada tela em cada dispositivo consome tempo e está sujeito a erros, daí a busca por técnicas automatizadas em fluxos de Integração e Entrega Contínuas (CI/CD) (Shahin; Babar; Zhu, 2017). Em essência, o processo verifica se mudanças de código alteraram a apresentação da interface, comparando capturas de tela entre versões (Tanno et al., 2020; Figueiredo, 2018; Bordignon, 2019).

A pirâmide de testes proposta por Cohn (2009) situa os testes visuais automatizados em um nível pouco explorado pela divisão clássica entre testes de unidade, integração e *end-to-end*: eles verificam a aparência renderizada da interface, algo que nenhuma das camadas tradicionais cobre diretamente. A Figura 3 (Apêndice A) ilustra uma adaptação dessa pirâmide, situando a verificação visual como uma camada de checagem de componente que precede os testes exploratórios manuais, mais custosos e menos repetíveis.

Três grupos de abordagem cobrem essa verificação. A comparação *pixel a pixel* identifica qualquer diferença entre as imagens, mas tende a gerar alertas desnecessários quando há variações de renderização, de fonte ou de *layout* (organização dos elementos visuais na tela) (Figueiredo, 2018; Tanno et al., 2020). Métricas perceptuais como a SSIM (*Structural Similarity Index*), proposta por Wang et al. (2004), comparam luminância, contraste e estrutura, aproximando-se mais da percepção humana. Já a abordagem por regiões divide a tela em partes, compara áreas equivalentes e admite o mascaramento de conteúdo dinâmico, o que reduz o efeito de deslocamentos e facilita a triagem (Tanno et al., 2020).

Quando a interface exibe valores que mudam a cada execução, como datas, contadores e dados externos, surgem diferenças recorrentes que não correspondem a regressões reais. Para contornar isso, recomenda-se aplicar máscaras apenas nas regiões inevitavelmente variáveis e manter uma política explícita de atualização de *baseline*, de modo a preservar rastreabilidade sem normalizar defeitos (Tanno et al., 2020; Heinonen, 2020). Heinonen (2020) aponta o determinismo do ambiente de captura (versões fixas de navegador, resolução constante, animações desativadas) como pré-requisito para a confiabilidade dos resultados.

No cenário nacional, Santos, Souza e Souza (2022) verificou que ferramentas de automação de testes ainda são pouco adotadas nas empresas brasileiras avaliadas, com baixo índice de prioridade atribuída à atividade de testes, enquanto Pinheiro, Valentim e Vincenzi (2015) demonstrou que automatizar testes de aceitação em aplicações web reduz o esforço de verificação e o retrabalho com defeitos encontrados tardiamente. Ainda no contexto de integração contínua, Lima e Vergilio (2023) avaliou abordagens de ranqueamento baseadas em aprendizado de máquina para priorização de casos de teste em CI e verificou que essas abordagens reduzem o tempo de *feedback* (retorno sobre o resultado das verificações) ao selecionar os casos mais informativos para cada *build* (execução de construção do soft-

ware).

Tanno et al. (2020) propõem e avaliam a técnica de detecção por regiões em aplicações web, com foco em identificar diferenças essenciais e reduzir falsos positivos; o trabalho não tem como objetivo a comparação sistemática entre técnicas com cenários de mutação controlada, coleta de métricas por abordagem ou integração a CI/CD com artefatos rastreáveis. Heinonen (2020) investiga testes visuais em *pipelines* de CI e aponta o determinismo do ambiente como condição necessária, porém sem confrontar o desempenho das técnicas entre si.

Não foi encontrado, na literatura consultada, trabalho que compare as três abordagens (pixel a pixel, SSIM e regiões) sob a mesma calibração, com cenários de mutação controlada, coleta padronizada de métricas por técnica, artefatos rastreáveis em CI/CD e avaliação em diferentes larguras de visualização simultaneamente.

Sem uma comparação sistemática entre as abordagens em condições equivalentes, a escolha acaba guiada por preferência de ferramenta, não por evidência. Equipes que consideram adotar testes visuais em CI/CD raramente encontram dados concretos sobre detecção, falsos positivos e esforço de triagem lado a lado para orientar essa decisão.

Este trabalho avalia três técnicas de regressão visual dentro de um fluxo automatizado de CI/CD. Para viabilizar a comparação, foi desenvolvida a ferramenta PixelGuard, implementada para este estudo, que integra captura de telas via Playwright, comparação simultânea pelas três técnicas e publicação de evidências em relatório HTML e interface de revisão. Foram construídos doze cenários de mutação controlada e um de controle, todos executados no *viewport* fixo de $1366 \times 768$ px. A tela *dashboard* (painel de interface com indicadores) foi avaliada adicionalmente em três larguras de visualização, com imagens de referência capturadas automaticamente do ramo principal (*branch* `main`) via *git worktree*.

Partindo da premissa de que combinar as três técnicas resulta em cobertura mais abrangente do que qualquer abordagem isolada, o objetivo geral é avaliar essas técnicas quanto à detecção de falhas, à ocorrência de falsos positivos e ao esforço de triagem, com captura automática de imagens de referência do ramo principal em CI/CD. Para isso, foram definidos quatro objetivos específicos: (a) identificar cenários e tipos de mudança na interface com risco de regressão visual, (b) implementar um fluxo automatizado de verificação em CI/CD com geração de artefatos rastreáveis, (c) medir o desempenho de cada técnica em detecção, falsos positivos e tri-

agem e (d) elicitar requisitos funcionais e não funcionais, com critérios de aceitação rastreáveis, para a comparação entre as três técnicas, incluindo sua robustez diante de variações de renderização, *layout* e conteúdo dinâmico.

O restante deste artigo está organizado da seguinte forma: a Seção 2 descreve os materiais e métodos; a Seção 3 apresenta e discute os resultados; e a Seção 4 traz as conclusões e trabalhos futuros.

## 2 MATERIAIS E MÉTODOS

O trabalho compara, em condições controladas, três técnicas de detecção de regressão visual integradas a um fluxo de Integração e Entrega Contínuas (CI/CD). O delineamento combina procedimentos experimentais, com mutações controladas e coleta de métricas por técnica em cenários rotulados, e procedimentos descritivos, ao relatar os indicadores obtidos. A definição dos valores de corte teve caráter exploratório: a literatura não prescreve valores para esse tipo de calibração, e os adotados foram registrados como pontos de partida configuráveis, não como ótimos estabelecidos. A fundamentação recorre a métricas perceptuais de imagem, especialmente a SSIM proposta por Wang et al. (2004), e a práticas de operação em *pipelines* de software discutidas por Shahin, Babar e Zhu (2017) e Heinonen (2020).

### 2.1 AMBIENTE EXPERIMENTAL E DETERMINISMO

Os testes foram executados em ambiente padronizado para garantir resultados reprodutíveis. Utilizou-se Node.js (v20, ambiente de execução JavaScript) e Playwright, ferramenta de automação de navegadores, com Chromium em modo *headless* (sem interface gráfica visível) para captura de telas, com densidade de pixels fixa, idioma pt-BR e fuso horário America/Sao_Paulo. Para reduzir instabilidades, o relógio do sistema e a função `Math.random` foram congelados por um *script* de inicialização injetado antes do carregamento da página, mantendo constantes os valores dinâmicos (data, hora, números aleatórios) entre execuções. Após o carregamento completo da página, animações e transições CSS (instruções de estilização visual da página) foram desativadas por injeção de folha de estilo, garantindo que a captura registrasse a interface em estado estático.

Na comparação principal com a tela *dashboard*, as capturas foram feitas em página inteira, enquanto nos cenários de mutação utilizou-se captura pela área do *viewport*, sem *full-page*, para eliminar variações de di-

mensão entre execuções. As imagens de referência (*baseline*) foram capturadas automaticamente do ramo principal via *git worktree* a cada execução do *pipeline*, sem necessidade de armazená-las no repositório.

## 2.2 DELINEAMENTO EXPERIMENTAL

O experimento combina quatro elementos:

a) Cenário avaliado. O cenário central foi uma aplicação de página única (*single-page*) em React, biblioteca JavaScript para construção de interfaces (tela *dashboard*), capturada de forma determinística para reduzir variações entre execuções.

b) Técnicas. Foram comparadas três abordagens: comparação *pixel a pixel* com tolerância usando a biblioteca pixelmatch, comparação perceptual baseada em SSIM conforme Wang et al. (2004) com implementação própria em blocos não sobrepostos, e técnica por regiões com segmentação em grade e suporte a máscaras por célula (Tanno et al., 2020). Nesta implementação, o mascaramento espacial está disponível apenas na técnica por regiões, pois as outras duas avaliam a imagem inteira sem exclusão de áreas.

c) Larguras de visualização. Foram utilizados três *viewports* (larguras da área visível da tela): 360 px (*mobile*), 768 px (*tablet*) e 1366 px (*desktop*). A largura de 1366 px foi adotada como referência de *desktop* por ser uma das resoluções mais comuns tanto em monitores externos quanto em telas de *notebook*, não se restringindo a este último tipo de dispositivo. Cada página foi capturada em todas as larguras.

d) Máscaras e política de aceitação. A técnica por regiões foi configurada com grade $4 \times 6$ e suporte a mascaramento por célula. Para a tela *dashboard*, as máscaras permaneceram desativadas (lista vazia) para evidenciar o comportamento de ruído e alertas em condição padrão. No conjunto de cenários, o cenário de conteúdo dinâmico foi executado em duas variantes (sem máscara e com máscara aplicada à célula que contém o valor variável), para isolar o efeito do mascaramento. A atualização de *baseline* ocorre automaticamente a cada execução do *pipeline*, que captura o estado do ramo principal via *git worktree*; qualquer mudança integrada ao ramo principal torna-se automaticamente a nova referência na execução seguinte.

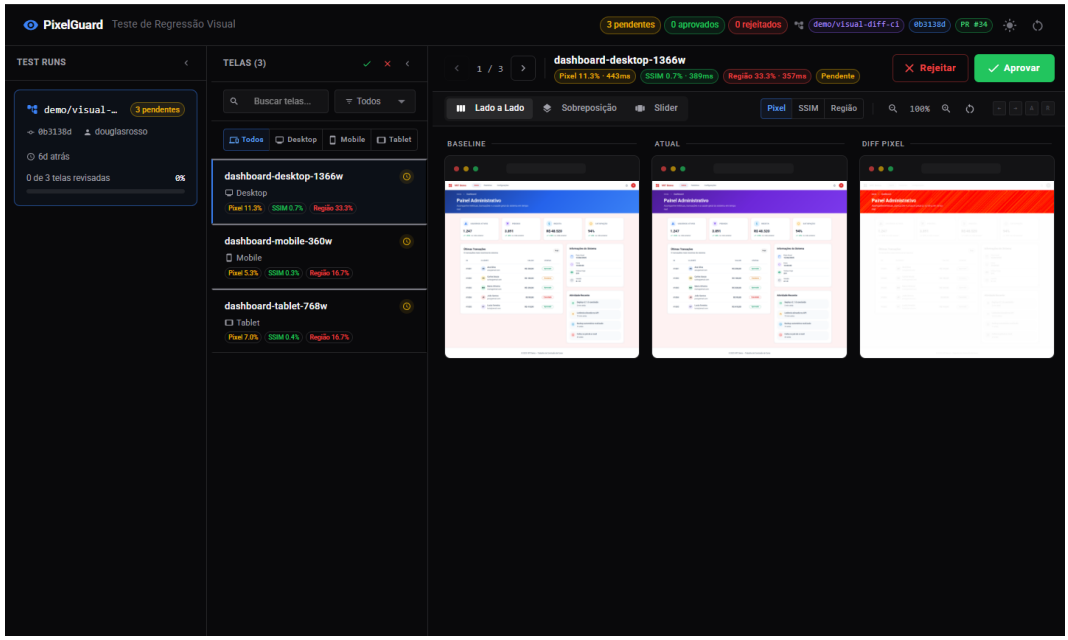
As Figuras 5, 6 e 7 (Apêndice A) resumem o princípio de cada uma das três técnicas comparadas.

## 2.3 IMPLEMENTAÇÃO DO FLUXO (PIXELGUARD)

Para a avaliação e a operação em *pipeline*, foi implementada a ferramenta PixelGuard, desenvolvida para este estudo. A ferramenta cobre captura de telas com Playwright em ambiente controlado (relógio e `Math.random` congelados, animações desativadas, idioma e fuso horário fixos), comparação simultânea pelas três técnicas com parâmetros definidos em arquivo de configuração e publicação de artefatos de diagnóstico: relatório HTML autocontido e interface de revisão em React que permite inspecionar cada comparação lado a lado, em sobreposição ou com controle deslizante.

A Figura 1 mostra a *Review UI* em uso: a lista de *test runs* e telas à esquerda, o painel central de comparação lado a lado entre *baseline* e captura atual com o mapa de diferenças por técnica (pixel, SSIM ou regiões), e as ações de aprovar ou rejeitar a alteração à direita.

Figura 1 - Interface de revisão (*Review UI*): comparação lado a lado e ações de aprovação




Fonte: Do autor.

No fluxo de CI/CD, três *workflows* distintos no GitHub Actions cobriam a verificação visual a cada solicitação de mudança, o controle do resultado por comentário e o registro de auditoria das solicitações integradas com diferenças visuais detectadas.

Em termos de fluxo, a abertura de uma solicitação de mudança (*pull request*) dispara o *workflow* de verificação visual no GitHub Actions, que captura, nas mesmas condições, a imagem de referência do ramo principal via *git worktree* e a imagem atual da *branch* em análise. As duas capturas são comparadas simultaneamente pelas três técnicas (pixel a pixel, SSIM e regiões), cada uma produzindo um veredito de PASS ou FAIL e sua métrica correspondente. O resultado é publicado como relatório HTML, na interface de revisão (*Review UI*) e como comentário na própria solicitação de mudança.

Quando nenhuma das técnicas sinaliza regressão, o fluxo segue automaticamente; quando há sinalização, a *Review UI* encaminha o caso para análise humana, que pode aprovar a diferença (registrando o motivo, nos casos em que a mudança é intencional) ou reprová-la. A interface de revisão comunica essa decisão ao GitHub via API de *Statuses*, marcando o *commit* como aprovado, reprovado ou pendente. Uma reprovação exige um novo *commit*, que reinicia todo o processo a partir da captura das imagens; uma aprovação libera a solicitação de mudança para seguir às etapas seguintes do *pipeline*. Esse encadeamento garante que toda decisão sobre uma diferença visual fique registrada e associada ao *commit* que a originou, em vez de depender de inspeção informal fora do fluxo de CI/CD.

A Figura 4 (Apêndice A) detalha esse fluxo: abertura do *pull request*, captura de *baseline* e da tela atual, comparação simultânea pelas três técnicas, geração do relatório, checagem automática de regressão e decisão de aprovação ou reprovação.

## 2.4 REQUISITOS DO FLUXO AUTOMATIZADO

Os objetivos específicos (b) e (d) preveem, respectivamente, a implementação de um fluxo automatizado com artefatos rastreáveis e a elicitação de requisitos funcionais e não funcionais para a comparação entre as três técnicas. A Tabela 1 e a Tabela 2 declaram os requisitos funcionais (RF) e não funcionais (RNF) que orientaram a implementação da PixelGuard, com critérios de aceitação verificáveis a partir do que foi descrito nesta seção e dos resultados apresentados na Seção 3.

## 2.5 AVALIAÇÃO POR CENÁRIOS DE REGRESSÃO

Além da comparação com a tela *dashboard*, foram implementadas doze páginas de cenário na aplicação, cada uma projetada para isolar um tipo de alteração e expor pontos fortes e fracos de cada técnica. A cada

Tabela 1 - Requisitos funcionais da PixelGuard

| ID | Requisito | Critério de aceitação |
|---|---|---|
| RF01 | Capturar telas via Playwright em ambiente controlado (relógio e `Math.random` congelados, animações desativadas, idioma e fuso horário fixos) | Duas execuções do cenário de controle, sem alteração de código, produzem imagens idênticas e veredito PASS nas três técnicas |
| RF02 | Comparar simultaneamente *baseline* e captura atual pelas três técnicas (pixel a pixel, SSIM e regiões) | Cada comparação gera veredito (PASS/FAIL) e métrica principal por técnica, persistidos em arquivo JSON de resultados |
| RF03 | Publicar relatório HTML autocontido e interface de revisão (*Review UI*) com evidências lado a lado, sobreposição e controle deslizante | O relatório é gerado a cada execução e permite inspecionar qualquer comparação reprovada sem acesso ao ambiente de execução |
| RF04 | Integrar o resultado da revisão ao GitHub via API de *Statuses*, registrando aprovação, reprovação ou redefinição por *commit* | O *status* do *commit* no GitHub reflete a decisão tomada na *Review UI* e é auditável no histórico da solicitação de mudança |

Fonte: Do autor.

cenário corresponde uma mutação controlada, injetada via CSS ou *script* durante a captura, e uma imagem de referência capturada nas mesmas condições, exceto pela mutação. Os cenários utilizaram *viewport* fixo de $1366 \times 768$ px (captura por área do *viewport*, sem *full-page*) para eliminar variações de dimensão entre capturas. Um cenário adicional de controle (imagem idêntica) foi incluído para verificar a ausência de falsos positivos.

Os doze cenários cobrem, respectivamente: mudança sutil de cor, deslocamento de *layout*, variação tipográfica, conteúdo dinâmico (sem e com máscara), alteração localizada de componente, opacidade e transparência, sombra e elevação, micro-deslocamento (1 px), alteração de cor de borda fina, remoção de elemento e troca de família de fonte, além do cenário de controle (imagem idêntica). A especificação completa de cada mutação (Tabela 6) está no Apêndice C.

Dois desses cenários têm uma escolha de projeto que merece destaque já no corpo do texto: no cenário de opacidade, o valor 0,92 foi calibrado deliberadamente para que a variação por canal de cor ficasse abaixo da tolerância de 0,1 adotada pelo pixelmatch, testando o limite inferior de sensibilidade das três técnicas; e no cenário de alteração de cor de borda

Tabela 2 - Requisitos não funcionais da PixelGuard

| ID | Requisito | Critério de aceitação |
|---|---|---|
| RNF01 | Determinismo: o ambiente de captura deve produzir o mesmo resultado entre execuções repetidas, dado o mesmo código | O cenário de controle (imagem idêntica) aprova nas três técnicas, com 0 % de pixels divergentes, tanto localmente quanto no *runner* do GitHub Actions |
| RNF02 | Baixo tempo de execução por comparação, compatível com a execução em *runner* de CI a cada solicitação de mudança | Tempo de execução registrado por técnica e por *viewport* inferior a 1 segundo (na prática, entre 368 ms e 470 ms no CI, conforme reportado na Seção 3) |
| RNF03 | Rastreabilidade e auditabilidade das decisões de cada execução | Cada comparação gera artefato JSON com veredito, métrica e identificação do *commit*/*branch*, e cada decisão de revisão é registrada em log de auditoria associado à solicitação de mudança |
| RNF04 | Reprodutibilidade entre ambiente local e CI (local $\equiv$ CI) | Os mesmos cenários de mutação controlada e o cenário de controle produzem vereditos equivalentes quando executados localmente e no GitHub Actions |

Fonte: Do autor.

fina, "fina"refere-se à espessura fixa de 2 px da borda, enquanto o que de fato varia entre *baseline* e captura atual é a cor.

Para cada cenário, as imagens foram comparadas pelos três métodos com os mesmos limiares da avaliação principal.

## 2.6 MÉTRICAS E COLETA

Foram coletados indicadores gerados automaticamente pelas técnicas e pelo próprio fluxo de execução. Para a comparação *pixel a pixel*, registrou-se o percentual de pixels diferentes e a contagem de pixels divergentes. Para a SSIM, registrou-se a pontuação de similaridade estrutural e a proporção de blocos com baixa similaridade. Para a abordagem por regiões, registraram-se a quantidade de células reprovadas e o total de células, além do percentual de diferença por célula. Também foram registrados erros operacionais (por exemplo, dimensões diferentes entre *baseline* e captura atual) e o tempo de execução por técnica.

## 2.7 VALIDADE E LIMITAÇÕES

Foram documentadas ameaças à validade interna, como viés na seleção de telas e sensibilidade dos limiares de aceitação, e à validade externa, como o alcance da generalização para outros tipos de aplicações e ambientes. Também foram registradas dependências do ambiente (fontes, motores de renderização e resolução) e as medidas adotadas para evitar que variações não controladas afetassem os resultados. No uso de máscaras, optou-se por restringi-las a regiões inevitavelmente variáveis e revisá-las a cada atualização de *baseline*, evitando que um ajuste inadequado escondesse defeitos.

Outras três limitações do recorte adotado merecem registro: a avaliação multi-*viewport* se restringe à tela *dashboard*, enquanto os doze cenários de mutação controlada foram executados apenas no *viewport* de $1366 \times 768$ px, para eliminar variações de dimensão entre capturas, o esforço de triagem foi avaliado de forma qualitativa, a partir das evidências geradas pelo fluxo, sem cronometragem com avaliadores, e os indicadores de taxa de detecção e taxa de alertas indevidos são reportados em relação ao conjunto rotulado de cenários construídos para este estudo, e não em relação a um histórico de alterações reais de produção.

Em particular, a ocorrência de falsos positivos foi medida conforme previsto: nenhuma das três técnicas sinalizou diferença no cenário de controle nem nos demais cenários rotulados, resultado relatado na discussão dos resultados (Seção 3) que indica ausência de tendência intrínseca a falso positivo nas condições adotadas. Essa taxa de 0 %, no entanto, vale para o conjunto de treze cenários construídos sob ambiente determinístico, e não pode ser generalizada estatisticamente (por exemplo, como intervalo de confiança) para um histórico de produção ou para ambientes com variações não controladas, dado o tamanho reduzido da amostra. Caracterizar a taxa de falsos positivos sob essas condições mais amplas exigiria um conjunto maior e mais diverso de cenários, ficando registrado como extensão natural deste trabalho.

# 3 RESULTADOS E DISCUSSÃO

A execução do fluxo gerou dois conjuntos de resultados. Primeiro, a tela *dashboard* foi comparada nas três larguras de visualização (*mobile* a 360 px, *tablet* a 768 px e *desktop* a 1366 px) dentro de uma solicitação de mudança real: a *branch* `demo/visual-diff-ci` alterou o gradiente do componente *HeroBanner* de azul para roxo, e o CI/CD comparou as

capturas recém-geradas com as imagens de referência do ramo principal. Em seguida, foram executados os doze cenários de mutação controlada e o cenário de controle no *viewport* de $1366 \times 768$ px, com o objetivo de medir a capacidade de detecção de cada técnica.

A comparação *pixel a pixel* reprovou oito dos doze cenários com mutação, a SSIM reprovou cinco e a abordagem por regiões reprovou nove. O cenário de controle passou nas três sem falso positivo. Os indicadores foram extraídos automaticamente pelo PixelGuard e salvos em arquivos JSON de resultados, com veredito (PASS ou FAIL), métrica principal de cada técnica e tempo de execução por comparação. Os limiares aplicados foram os definidos na configuração: 0,1 % para pixel, 0,99 para SSIM e 1,0 % por célula na abordagem por regiões.

## 3.1 JUSTIFICATIVA DOS LIMIARES ADOTADOS

Os parâmetros de aceitação de cada técnica (0,1 % de pixels diferentes para a comparação *pixel a pixel*, 0,99 para a SSIM e 1,0 % por célula para a abordagem por regiões) foram definidos como pontos de partida configuráveis, combinando padrões das bibliotecas utilizadas e escolhas de projeto, não como ótimos sistematicamente validados. Todos os parâmetros são expostos em `pixelguard.config.js` e podem ser substituídos por calibração específica à aplicação. A justificativa detalhada de cada valor está no Apêndice C.

## 3.2 COMPARAÇÃO NA SOLICITAÇÃO DE MUDANÇA REAL

A Tabela 3 (Apêndice B) mostra os resultados do CI/CD ao comparar a tela *dashboard* nas três larguras de visualização após a mudança de gradiente do *HeroBanner* (azul para roxo). A coluna *Pixel* indica o percentual de pixels divergentes (PASS  0,1 %), a coluna *SSIM* indica a pontuação de similaridade estrutural (PASS  0,99) e a coluna *Regiões* indica as células reprovadas em grade $4 \times 6$ (PASS = 0 células acima de 1 % de diferença interna).

Pixel e regiões reprovaram nas três larguras. A SSIM aprovou em todas elas. A mudança de gradiente cromático (azul $\rightarrow$ roxo) altera a cor percebida pelo olho humano, mas preserva a distribuição de luminância e contraste estrutural da cena, o que explica por que a SSIM ficou acima do limiar de 0,99 mesmo com uma diferença visual evidente. Pixel e regiões detectaram porque operam sobre diferenças diretas de canal de cor, sem modelar percepção humana. Esse resultado configura um falso negativo da

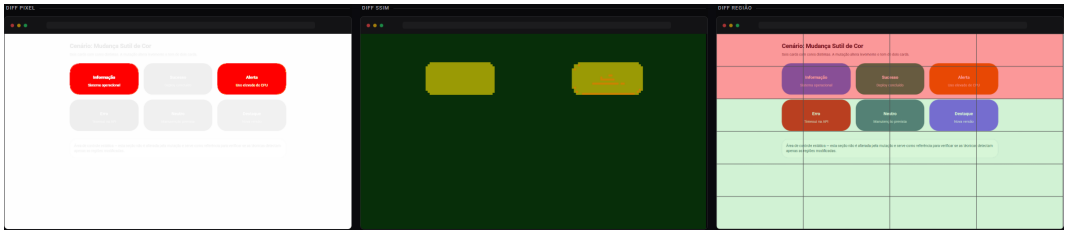
SSIM para mudanças cromáticas de luminância similar: depender apenas dessa técnica teria deixado a regressão passar sem alerta.

O CI/CD registrou tempos de execução de 470 ms (pixel), 379 ms (SSIM) e 368 ms (regiões) para o *viewport desktop*, valores mais altos que os obtidos localmente por conta do ambiente de execução do GitHub Actions. Heinonen (2020) trata o determinismo do ambiente de captura como pré-requisito para operar em *pipelines*. Sem ele, a comparação *pixel a pixel* acumula falsos positivos e perde utilidade prática. No experimento, o congelamento de `Date` e de `Math.random` antes do carregamento da página, combinado à execução em modo *headless* com densidade de pixels fixa, garantiu resultados reprodutíveis tanto localmente quanto no CI.

## 3.3 EVIDÊNCIAS VISUAIS E TRIAGEM

O fluxo gerou imagens de diferenças para as três técnicas (pixel, SSIM e regiões), além de um relatório HTML com resumo e evidências. A interface de revisão (*Review UI*) foi utilizada para suportar a triagem, permitindo selecionar uma captura e alternar entre técnicas de *diff* (comparação visual entre versões das imagens). A Figura 2 reúne, lado a lado, o painel de comparação do cenário *Mudança sutil de cor* nas três técnicas (pixel, SSIM e regiões), facilitando a leitura conjunta das evidências.

Figura 2 - Painéis de comparação lado a lado: pixel, SSIM e regiões (cenário *Mudança sutil de cor*)




Fonte: Do autor.

O painel esquerdo (pixel) marca em vermelho os dois cards alterados (5,854 % de pixels diferentes, acima do limiar de 0,1 %). O painel central (SSIM) registra pontuação 0,9976 e veredito PASS: a alteração é cromática e não perturba formas nem contornos, embora a diferença seja visível. O painel direito (regiões) reprova oito das 24 células, localizando espacialmente as áreas afetadas. A SSIM isolada não acusaria a regressão; pixel isolado acusaria, mas sem indicar onde na imagem.

## 3.4 DISCUSSÃO: SENSIBILIDADE, RUÍDO E OPERAÇÃO EM CI/CD

Entender onde cada técnica falha importa tanto quanto saber onde acerta, especialmente quando o custo de um falso positivo ou de uma regressão não detectada recai sobre a equipe de desenvolvimento.

A comparação *pixel a pixel* foi a mais reativa a alterações pequenas. Quando a interface foi modificada apenas no texto e na borda de um único componente, a técnica registrou 0,128 % de pixels diferentes, percentual baixo em termos absolutos mas suficiente para disparar o alerta, pois ultrapassa o limiar de 0,1 %. O micro-deslocamento de 1 px produziu situação semelhante: apenas uma linha de pixels deslocada, e a técnica já acusou 0,198 % de diferença. Figueiredo (2018) e Tanno et al. (2020) já haviam registrado essa característica da comparação direta de pixels: a mesma sensibilidade que impede regressões de passar despercebidas também gera alertas quando há flutuação de renderização ou variação de *anti-aliasing* entre execuções.

A SSIM teve comportamento diferente porque não conta pixels, mas avalia se a estrutura das duas imagens (formas, contornos, contraste) permanece equivalente em pequenas janelas. Além da pontuação global, a técnica registra quantos dos 16.320 blocos de $8 \times 8$ px por captura apresentaram baixa similaridade. Nos cenários em que a estrutura foi afetada, a pontuação caiu de forma marcada e a proporção de blocos com baixa similaridade subiu, com 0,8762 e 25,23 % no deslocamento de *layout* (toda a parte inferior da página foi empurrada 24 px para baixo, deformando a relação entre os elementos), 0,9269 e 9,33 % na variação tipográfica (o aumento de *letter-spacing* alterou o desenho de cada linha de texto), 0,9316 e 15,30 % no micro-deslocamento (a translação afeta a relação espacial dos blocos) e 0,8886 e 15,21 % na troca de fonte (mudou o desenho dos glifos).

Nos cenários em que apenas a aparência local mudou sem alterar estrutura, como a redução de opacidade e a adição de sombra, a SSIM permaneceu acima de 0,995 e a proporção de blocos com baixa similaridade ficou em 0 % e 2,85 %, respectivamente, dentro do limite de tolerância. Esse comportamento é coerente com o que Wang et al. (2004) descreve ao propor a SSIM: por ser perceptual, a métrica não reage a variações de baixa intensidade que não alteram a leitura estrutural da cena, mesmo quando a diferença é visível a olho nu.

A abordagem por regiões desempenhou um papel distinto das outras duas. Além do veredito, ela indica em quais células da grade $4 \times 6$ a diferença aconteceu e qual o percentual de divergência em cada uma.

Nos cenários de remoção de elemento e alteração de componente, em que a mudança é localizada, foi essa informação espacial que permitiu reduzir a inspeção a uma fração pequena da imagem (3 e 1 células, respectivamente). Nos cenários estruturais, como deslocamento de *layout* (15/24) e troca de fonte (14/24), a mesma informação revelou que o efeito se propagou por boa parte da tela. Tanno et al. (2020) apontam que localizar a mudança dentro da imagem é essencial para reduzir o esforço de triagem. Na prática, saber onde olhar importa mais do que apenas detectar se há diferença.

Para avaliar o ruído, o cenário de controle (imagem idêntica) foi incluído para verificar se alguma das técnicas sinalizaria diferença onde não havia nenhuma. As três aprovaram com 0 %, 1,0000 e 0/24, indicando que, nas condições adotadas, não há tendência intrínseca a falso positivo. Esse resultado só foi possível pelo controle ambiental adotado: sem ele, duas execuções consecutivas gerariam imagens ligeiramente diferentes por razões alheias ao código, e qualquer das técnicas dispararia alertas sem que houvesse regressão alguma.

O contraponto é o cenário de conteúdo dinâmico, que ilustra o que ocorre quando uma região da tela varia naturalmente entre execuções: data, hora e contadores. Sem máscara, a abordagem por regiões reprovou ao sinalizar 1,17 % de diferença em uma única célula, enquanto pixel e SSIM aprovaram porque a alteração era pequena demais em relação à imagem total. Com a máscara aplicada a apenas àquela célula, as três técnicas aprovaram sem esconder o restante da tela. Tanto Tanno et al. (2020) quanto Heinonen (2020) recomendam restringir as máscaras às regiões inevitavelmente variáveis, e os dados obtidos sustentam essa orientação: ampliar o mascaramento para cobrir áreas maiores arriscaria esconder regressões reais.

A Figura 8 (Apêndice B) mostra o cenário de deslocamento de *layout*: uma única regra CSS empurrou toda a página 24 px para baixo, mudança mínima no código com efeito visual expressivo na imagem. O painel da esquerda (pixel) marca em vermelho a maior parte da tela, refletindo 0,637 % de pixels diferentes. O painel central (SSIM) exibe área verde-clara extensa, correspondendo à queda de pontuação para 0,8762. O painel da direita (regiões) marca 15 das 24 células em vermelho, evidenciando a propagação espacial do efeito. As três técnicas reprovaram, cada uma expressando o problema por uma métrica diferente, o que reforça a complementaridade entre elas.

A Figura 9 (Apêndice B) ilustra o caso oposto: uma alteração real que não aciona nenhuma das três técnicas. A redução de opacidade para 0,92, valor escolhido propositalmente abaixo da tolerância por canal do pixelmatch (0,1) para testar esse limite, alterou cada pixel individualmente em magnitude inferior a essa tolerância, de forma que nenhum pixel foi contado como divergente. A SSIM ficou em 0,9996 (estrutura preservada) e nenhuma célula ultrapassou 1 %. Esse resultado exemplifica bem o equilíbrio entre sensibilidade e ruído: limiares mais permissivos reduzem alertas desnecessários, mas podem deixar passar variações de baixa amplitude que são visíveis a olho nu.

A Figura 10 (Apêndice B) expõe o limite inferior de detecção. Um deslocamento de apenas 1 px gerou 0,198 % de pixels diferentes, queda da SSIM para 0,9316 e 14 células reprovadas em 24. A queda da SSIM é mais intensa do que a magnitude sugere: ao deslocar verticalmente uma sequência de blocos, cada janela da grade passa a comparar pixels desalinhados, o que afeta luminância e contraste localmente. Translações pequenas produzem assinaturas amplas nas três técnicas, mas SSIM e regiões ofereceram leitura mais informativa sobre a natureza estrutural do efeito do que a simples contagem de pixels divergentes.

A Figura 11 (Apêndice B) representa a situação em que a tela exibe um valor que muda a cada execução, como data e contador, sem que isso caracterize uma regressão. Pixel e SSIM aprovaram a captura (0,075 % de pixels diferentes e pontuação 0,9980) porque a variação ocupa fração muito pequena da imagem total. Já a abordagem por regiões reprovou a célula que contém o valor variável, pois dentro daquele quadrante isolado o percentual de pixels modificados ultrapassou o limiar local de 1 %. Isso mostra a vantagem de avaliar a imagem por partes em vez de trabalhar com a média da imagem inteira: defeitos pequenos não ficam diluídos num valor global. Por esse mesmo raciocínio, faz mais sentido aplicar a máscara na célula correspondente do que elevar globalmente o limiar do pixel, pois essa estratégia preserva a capacidade de detecção no restante da tela e neutraliza apenas o ponto sabidamente variável.

## 3.5 COMPARAÇÃO POR CENÁRIOS DE REGRESSÃO

A Tabela 4 (Apêndice B) apresenta os resultados da avaliação por cenários. Cada linha corresponde a um tipo de mutação controlada, e as colunas indicam o veredito e a métrica principal de cada técnica.

Os treze cenários agrupam-se naturalmente em três perfis de

comportamento.

O primeiro reúne as alterações que afetam a relação espacial ou estrutural dos elementos: deslocamento de *layout*, variação tipográfica, troca de fonte, micro-deslocamento e remoção de elemento. As três técnicas reprovaram simultaneamente em todos eles. A SSIM caiu para 0,8762 (deslocamento), 0,9269 (tipografia), 0,8886 (troca de fonte), 0,9316 (micro-deslocamento) e 0,9822 (remoção), enquanto a abordagem por regiões apontou 15, 13, 14, 14 e 3 células reprovadas em 24, respectivamente. No cenário de remoção, a queda da SSIM é menos intensa do que nos demais porque o efeito estrutural se concentra em uma fração da imagem, ainda assim superando o limiar de 0,99. Mudanças que afetam a estrutura perceptual da tela, seja por deslocamento, recomposição tipográfica ou supressão de conteúdo, são detectadas pelas três abordagens. Esse comportamento é explicado por Wang et al. (2004): mudanças que perturbam a estrutura perceptual da cena se traduzem em quedas amplas ou localmente intensas nos indicadores da SSIM.

O segundo grupo abrange alterações localizadas de natureza predominantemente cromática, nas quais pixel e regiões reprovaram enquanto a SSIM aprovou: troca de cor de borda fina e alteração de texto com cor de borda em um único componente. Os percentuais de pixel são pequenos (0,273 % e 0,128 %), mas ultrapassam o limiar de 0,1 % e disparam o alerta. A SSIM ficou acima de 0,99 nas duas situações (0,9998 e 0,9924), pois a variação é cromática e não altera contornos, formas ou contraste estrutural. A abordagem por regiões reprovou as duas (4/24 e 1/24), fornecendo a informação espacial que reduz a inspeção a poucos quadrantes. Nesses casos, a SSIM isolada deixaria passar mudanças visíveis a olho nu. A combinação entre pixel e regiões corrige essa limitação, tanto detectando quanto localizando o problema. Note-se que o cenário de mudança sutil de cor, discutido anteriormente, apresenta comportamento análogo (SSIM 0,9976 acima de 0,99, aprovado, com pixel e regiões reprovados) e poderia compor este grupo, mas foi destacado nas evidências visuais por ilustrar com mais clareza a complementaridade das técnicas.

O terceiro grupo traz os cenários em que a alteração foi injetada, mas sua amplitude na imagem renderizada ficou abaixo dos limiares: opacidade 0,92 (pixel 0 %, SSIM 0,9996, 0/24) e *box-shadow* em três cards (pixel 0 %, SSIM 0,9953, 0/24). Duas razões sobrepostas explicam a ausência de detecção: a mutação produziu variação por canal inferior à tolerância do pixelmatch, e os valores de corte adotados não foram calibrados para esse

nível de amplitude. Ainda assim, a SSIM 0,9953 da sombra já aponta queda estrutural perceptível, que seria reprovada com valor de corte 0,996. Tanno et al. (2020) tratam exatamente desse ajuste fino ao discutir o equilíbrio entre sensibilidade e ruído.

### 3.6 SÍNTESE COMPARATIVA

Treze cenários (doze com mutação e um de controle) deixaram claro que os perfis de detecção são distintos e que as três técnicas se complementam. A Tabela 5 (Apêndice B) resume esse comportamento por tipo de alteração.

Não existe técnica única suficiente para todos os tipos de alteração testados. As taxas obtidas (8/12 para pixel, 5/12 para SSIM e 9/12 para regiões) evidenciam perfis distintos: pixel apresenta maior taxa de detecção em mudanças visíveis e localizadas, SSIM demonstra maior sensibilidade a alterações que afetam a estrutura perceptual, e a técnica por regiões combina detecção e apoio à triagem de forma mais abrangente (resultado consistente com a motivação de Tanno et al. (2020) para propor a abordagem baseada em regiões como alternativa à comparação direta de pixels).

## 4 CONCLUSÃO

As três técnicas de regressão visual avaliadas (comparação *pixel a pixel*, SSIM e abordagem por regiões) foram comparadas em condições equivalentes dentro de um fluxo automatizado de CI/CD, com artefatos rastreáveis gerados a cada execução. Treze cenários controlados tornaram possível o confronto direto entre as abordagens quanto a detecção, ruído e apoio à triagem.

O primeiro objetivo específico era identificar cenários com risco de regressão visual. Isso resultou em doze tipos de mutação, cobrindo desde deslocamentos de *layout* e trocas de fonte até alterações sutis de cor e conteúdo dinâmico. A seleção partiu da literatura, mas precisou ser adaptada à realidade de uma aplicação de página única em React: certos tipos de variação, como conteúdo dinâmico e animações, exigiram tratamento no próprio ambiente de captura antes de qualquer comparação ser viável.

O segundo objetivo era implementar o fluxo de CI/CD com artefatos rastreáveis. Foi atendido pela ferramenta PixelGuard, desenvolvida para este estudo. A parte mais trabalhosa não foi a implementação em si,

mas garantir que o ambiente de captura no *runner* do GitHub Actions produzisse resultados equivalentes ao ambiente local. Versões do Chromium, congelamento de `Date` e `Math.random` e desativação de animações tiveram de ser ajustados de forma iterativa até que o cenário de controle passasse com zero divergência nos dois ambientes.

O terceiro objetivo era medir detecção, falsos positivos e esforço de triagem. Os resultados respondem a essa pergunta: pixel detectou 8 dos 12 cenários com mutação, SSIM detectou 5 e regiões detectou 9. O cenário de controle passou nas três sem falso positivo. A SSIM mostrou limitação evidente em mudanças cromáticas que preservam luminância: no cenário do gradiente azul para roxo, a pontuação ficou acima de 0,99 nas três larguras (0,9934, 0,9959 e 0,9968) e a técnica aprovou a captura, mesmo com a diferença sendo visível a olho nu. A técnica por regiões foi a que mais contribuiu para a triagem, ao indicar em quais células da grade a diferença estava concentrada, o que reduz o que precisa ser inspecionado.

O quarto objetivo era elicitar requisitos funcionais e não funcionais, com critérios de aceitação rastreáveis, para a comparação entre as três técnicas, incluindo sua robustez diante de variações de renderização e conteúdo dinâmico. Esse objetivo foi atendido pelas Tabelas 1 e 2, que declaram quatro requisitos funcionais (captura controlada, comparação simultânea pelas três técnicas, publicação de relatório e *Review UI*, e integração de *status* ao GitHub) e quatro requisitos não funcionais (determinismo, baixo tempo de execução, rastreabilidade/auditabilidade e reprodutibilidade local $\equiv$ CI), cada um com critério de aceitação verificável a partir dos resultados obtidos.

Quanto à robustez especificamente, o resultado mais claro foi que o determinismo do ambiente de captura é pré-condição, não detalhe: sem controlar fontes de variação não intencional, como datas, números aleatórios e animações, qualquer técnica acumula ruído e perde utilidade prática. O mascaramento seletivo de regiões dinâmicas mostrou-se mais preciso do que elevar globalmente os limiares, porque preserva a sensibilidade no restante da tela.

Entre as limitações do trabalho, a mais relevante é o escopo de uma única aplicação de página única: os perfis de detecção observados não podem ser transferidos diretamente para sistemas com características visuais muito distintas, como painéis com grande volume de gráficos ou portais com imagens variáveis. As demais limitações do recorte adotado, como a ausência de cronometragem formal da triagem e a calibração dos limia-

res como ponto de partida configurável, já foram detalhadas na subseção *Validade e Limitações*.

Combinar as três técnicas no mesmo *pipeline* resultou na cobertura mais abrangente entre as abordagens avaliadas. Nenhuma das três, isolada, detecta todos os tipos de alteração testados: pixel identifica variações cromáticas localizadas que a SSIM não sinaliza, SSIM detecta alterações estruturais que o pixel subestima, e regiões localiza espacialmente onde o problema ocorre na imagem. A complementaridade entre elas é o argumento central do trabalho.

Para trabalhos futuros, investigar o comportamento das três técnicas em aplicações com maior densidade de conteúdo dinâmico, como painéis de monitoramento em tempo real, sistemas financeiros e portais de notícia, responderia perguntas que este estudo não alcançou. Estudar a paridade entre navegadores distintos (Chrome, Firefox, Safari) e avaliar as técnicas em dispositivos físicos, em vez de emulação, abre frentes relevantes e pouco exploradas na literatura consultada.

Uma varredura sistemática dos parâmetros de aceitação via curva ROC sobre um conjunto rotulado maior daria base quantitativa às escolhas que aqui ficaram como pontos de partida.

Por fim, retomando a limitação registrada na subseção *Validade e Limitações* sobre o tamanho reduzido da amostra: este trabalho mediu uma taxa de falsos positivos nula nos treze cenários avaliados sob ambiente determinístico, mas essa taxa não pode ser generalizada estatisticamente a partir de uma amostra tão pequena. Ampliar a caracterização para um conjunto de cenários maior e exposto a variações não controladas do ambiente (diferentes máquinas, motores de renderização e cargas de execução) permitiria estimar essa taxa com significância estatística e verificar se ela se mantém em condições menos controladas, sendo uma extensão natural deste estudo.

## DECLARAÇÃO DE USO DE INTELIGÊNCIA ARTIFICIAL

Os autores declaram que ferramentas de Inteligência Artificial generativa foram utilizadas exclusivamente como apoio auxiliar à elaboração deste trabalho, incluindo assistência na revisão linguística, estruturação textual e organização de ideias. Nenhuma ferramenta de IA foi utilizada para substituição da autoria intelectual, análise científica, interpretação dos resultados ou tomada de decisões acadêmicas. A responsabilidade integral pelo conteúdo, originalidade, veracidade das informações e conformidade

com os princípios de integridade científica permanece integralmente atribuída aos autores.

## REFERÊNCIAS


BORDIGNON, P. H. **Ferramentas visuais de teste de interfaces gráficas para checar conformidade de requisitos não funcionais**. Monografia (Trabalho de Conclusão de Curso (Graduação)) — Universidade Tecnológica Federal do Paraná (UTFPR), Dois Vizinhos, PR, Brasil, 2019. Disponível em: <http://repositorio.utfpr.edu.br/jspui/handle/1/10782>. Acesso em: 08 nov. 2025.

COHN, M. **Succeeding with Agile: Software Development Using Scrum**. Boston, MA: Addison-Wesley. 2009.

FIGUEIREDO, L. A. **Testes regressivos visuais: uma aplicação em um projeto Android**. Monografia (Trabalho de Conclusão de Curso (Graduação)) — Universidade Federal de Ouro Preto (UFOP), João Monlevade, MG, Brasil, 2018. Disponível em: <https://www.monografias.ufop.br/bitstream/35400000/862/1/MONOGRAFIA_TestesRegressivosVisuais.pdf>. Acesso em: 08 nov. 2025.

HEINONEN, J. **Design and Implementation of Automated Visual Regression Testing in a Large Software Product**. Dissertação (Master's Thesis) — LUT University, Lappeenranta, Finland, 2020. Disponível em: <https://lutpub.lut.fi/handle/10024/160665>. Acesso em: 31 out. 2025.

LIMA, J. A. d. P.; VERGILIO, S. R. **An evaluation of ranking-to-learn approaches for test case prioritization in continuous integration**. Journal of Software Engineering Research and Development, v. 11, n. 1, p. 4:1–4:20. ISSN 2195-1721. 2023, Disponível em: <https://journals-sol.sbc.org.br/index.php/jserd/article/view/2142>. Acesso em: 18 mai. 2026.

PINHEIRO, V. d. S. F.; VALENTIM, N. M. C.; VINCENZI, A. M. R. **Um comparativo na execução de testes manuais e testes de aceitação automatizados em uma aplicação Web**. In: **Anais do XIV Simpósio Brasileiro de Qualidade de Software (SBQS)**. Manaus, AM, Brasil: [s.n.], 2015. 2015. Disponível em: <https://sol.sbc.org.br/index.php/sbqs/article/view/15231>. Acesso em: 10 mai. 2026.

SANTOS, Í. d. O.; SOUZA, S. d. R. S. d.; SOUZA, P. S. L. d. **A survey on the practices of software testing: a look into Brazilian companies**. Journal of Software Engineering Research and Development, v. 10, p. 11:1–11:15. ISSN 2195-1721. 2022, Disponível em: <https://journals-sol.sbc.org.br/index.php/jserd/article/view/786>. Acesso em: 10 mai. 2026.

SHAHIN, M.; BABAR, M. A.; ZHU, L. **Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices**. IEEE Access, v. 5, p. 3909–3943. ISSN 2169-3536. 2017, Disponível em: <https://arxiv.org/pdf/1703.07019>. Acesso em: 13 nov. 2025.

TANNO, H. et al. **Region-based detection of essential differences in image-based visual regression testing**. Journal of Information Processing, v. 28, p. 268–278. ISSN 1882-6652. 2020, Disponível em: <https://www.jstage.jst.go.jp/article/ipsjjip/28/0/28_268/_pdf>. Acesso em: 31 out. 2025.

WANG, Z. et al. **Image quality assessment: From error visibility to structural similarity**. IEEE Transactions on Image Processing, v. 13, n. 4, p. 600–612. ISSN 1057-7149. 2004, Disponível em: <https://www.cns.nyu.edu/pub/lcv/wang03-preprint.pdf>. Acesso em: 31 out. 2025.

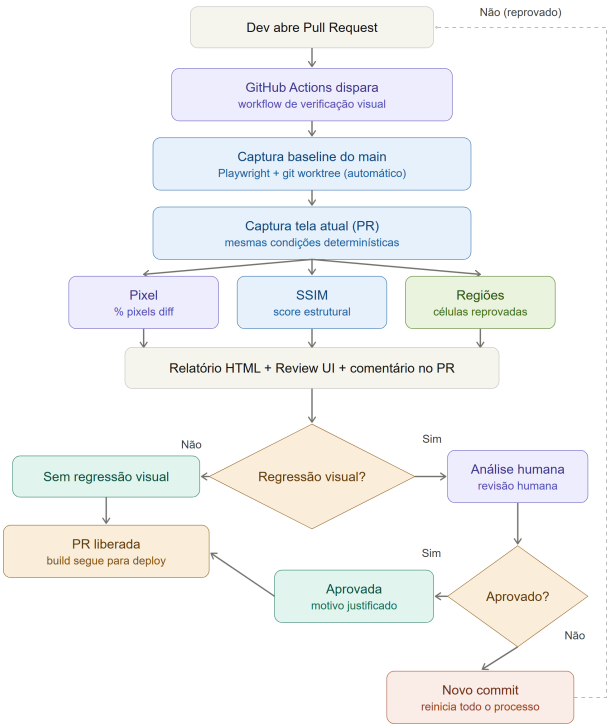
## APÊNDICE A - DIAGRAMAS COMPLEMENTARES

Este apêndice reúne os diagramas ilustrativos referenciados ao longo do artigo: a pirâmide de testes que situa a verificação visual em relação às demais camadas de teste, o fluxo completo de verificação implementado na PixelGuard e o princípio de funcionamento de cada uma das três técnicas comparadas.

Figura 3 - Pirâmide de testes.

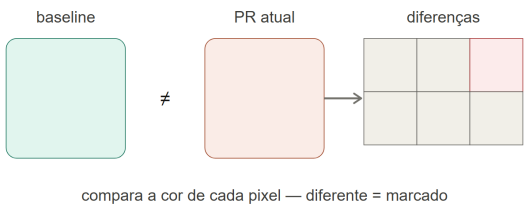

Fonte: Adaptado de Cohn (2009).

Figura 4 - Fluxo completo de verificação visual em *pull requests*



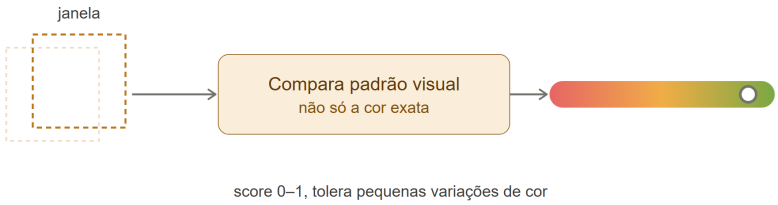

Fonte: Do autor.

Figura 5 - Princípio da técnica pixel a pixel



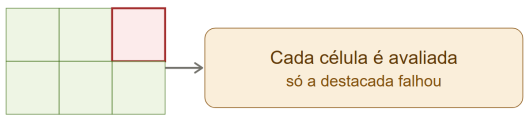

Fonte: Do autor.

Figura 6 - Princípio da técnica SSIM




Fonte: Do autor.

Figura 7 - Princípio da técnica por regiões




Fonte: Do autor.

## APÊNDICE B - EVIDÊNCIAS VISUAIS ADICIONAIS

Este apêndice reúne evidências visuais e tabelas de resultados complementares a cenários discutidos na Seção 3, cuja análise textual já cobre os achados relevantes no corpo do artigo.

Tabela 3 - Resultados do CI/CD na *branch* `demo/visual-diff-ci` (gradiente azul → roxo)

| Imagem | Pixel | SSIM | Regiões |
|---|---|---|---|
| dashboard-desktop-1366w | FAIL (11,33%) | PASS (0,9934) | FAIL (8/24) |
| dashboard-mobile-360w | FAIL (5,31%) | PASS (0,9968) | FAIL (4/24) |
| dashboard-tablet-768w | FAIL (7,05%) | PASS (0,9959) | FAIL (4/24) |

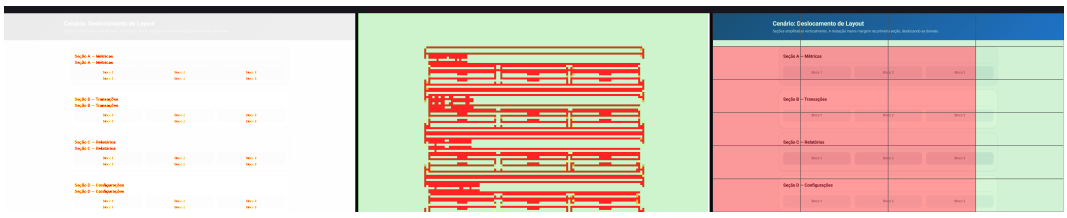
Fonte: Do autor.

Tabela 4 - Resultados por cenário de regressão

| Cenário | Pixel (%diff) | SSIM (score) | Regiões (falhas) |
|---|---|---|---|
| Mudança sutil de cor | FAIL (5,854%) | PASS (0,9976) | FAIL (8/24) |
| Deslocamento de layout | FAIL (0,637%) | FAIL (0,8762) | FAIL (15/24) |
| Variação tipográfica | FAIL (2,186%) | FAIL (0,9269) | FAIL (13/24) |
| Conteúdo dinâm. (s/ másc.) | PASS (0,075%) | PASS (0,9980) | FAIL (1/24) |
| Conteúdo dinâm. (c/ másc.) | PASS (0,075%) | PASS (0,9980) | PASS (0/24) |
| Alteração de componente | FAIL (0,128%) | PASS (0,9924) | FAIL (1/24) |
| Opacidade e transparência | PASS (0%) | PASS (0,9996) | PASS (0/24) |
| Sombra e elevação | PASS (0%) | PASS (0,9953) | PASS (0/24) |
| Micro-deslocamento (1 px) | FAIL (0,198%) | FAIL (0,9316) | FAIL (14/24) |
| Alteração de cor de borda fina | FAIL (0,273%) | PASS (0,9998) | FAIL (4/24) |
| Remoção de elemento | FAIL (0,399%) | FAIL (0,9822) | FAIL (3/24) |
| Troca de família de fonte | FAIL (2,614%) | FAIL (0,8886) | FAIL (14/24) |
| Imagem idêntica (controle) | PASS (0%) | PASS (1,0000) | PASS (0/24) |

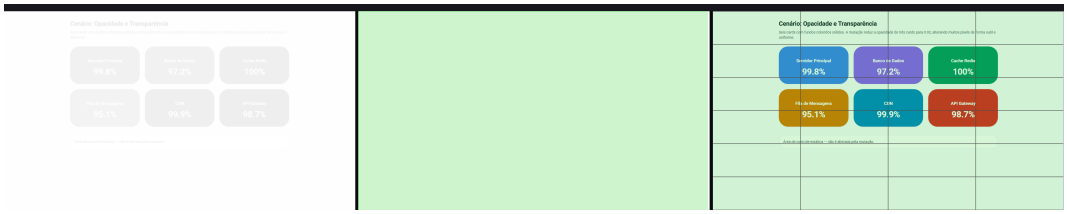
Fonte: Do autor.

Figura 8 - Diferenças geradas no cenário *Deslocamento de layout* (pixel, SSIM e regiões, da esquerda para a direita)



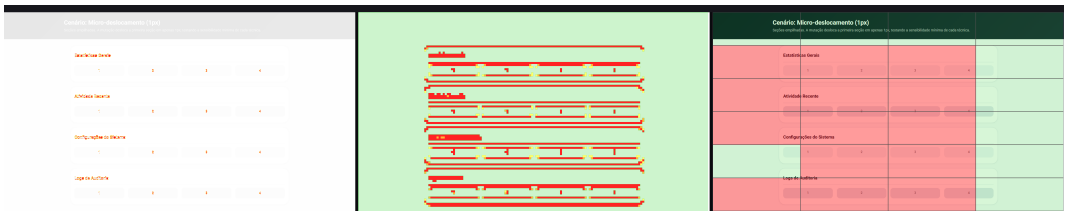
Fonte: Do autor.

Figura 9 - Cenário *Opacidade*: alteração existe, mas permanece abaixo dos limiares nas três técnicas



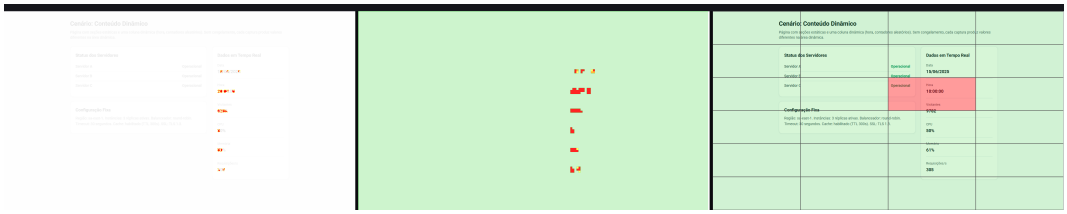

Fonte: Do autor.

Figura 10 - Cenário *Micro-deslocamento (1 px)*: as três técnicas reprovam, com perfis diferentes




Fonte: Do autor.

Figura 11 - Cenário *Conteúdo dinâmico* sem máscara: apenas a técnica por regiões isola a célula variável




Fonte: Do autor.

Tabela 5 - Síntese de detecção por técnica e tipo de alteração

| Tipo de alteração | Pixel | SSIM | Regiões |
|---|---|---|---|
| Cor sutil | Detecta | Não detecta | Detecta |
| Deslocamento | Detecta | Detecta | Detecta |
| Tipografia | Detecta | Detecta | Detecta |
| Dinâmico (s/ másc.) | Não detecta | Não detecta | Detecta |
| Dinâmico (c/ másc.) | Não detecta | Não detecta | Não detecta |
| Componente local | Detecta | Não detecta | Detecta |
| Opacidade | Não detecta | Não detecta | Não detecta |
| Sombra | Não detecta | Não detecta | Não detecta |
| Micro-deslocamento (1 px) | Detecta | Detecta | Detecta |
| Cor de borda fina | Detecta | Não detecta | Detecta |
| Remoção de elemento | Detecta | Detecta | Detecta |
| Troca de fonte | Detecta | Detecta | Detecta |
| Controle (idêntico) | Sem falso+ | Sem falso+ | Sem falso+ |

Fonte: Do autor.

## APÊNDICE C - ESPECIFICAÇÃO DOS CENÁRIOS DE MUTAÇÃO

Este apêndice detalha a mutação controlada injetada em cada um dos treze cenários (doze com mutação e um de controle) descritos na Seção 2.5, incluindo os parâmetros exatos utilizados na captura.

Tabela 6 - Especificação completa dos cenários de mutação controlada

| Cenário | Especificação da mutação |
|---|---|
| Mudança sutil de cor | Alteração de 22–37 unidades nos canais vermelho e verde em dois dos seis cards, preservando a composição estrutural |
| Deslocamento de *layout* | Inserção de 24 px de margem superior na primeira seção, deslocando as demais para baixo |
| Variação tipográfica | Aumento de *letter-spacing* em blocos de texto sem alterar a estrutura de *layout* |
| Conteúdo dinâmico | Referência congelada (`Date` e `Math.random`) versus captura sem congelamento; executado em duas condições: sem máscara e com máscara na célula variável |
| Alteração localizada de componente | Modificação do texto e da cor de borda de um único card em grade de seis |
| Opacidade e transparência | Redução de *opacity* para 0,92 (queda de 8 %) em três dos seis cards com fundo colorido, valor calibrado deliberadamente para ficar abaixo da tolerância de 0,1 do pixelmatch |
| Sombra e elevação | Adição de *box-shadow* em três dos seis cards planos, sem alterar conteúdo ou posicionamento |
| Micro-deslocamento (1 px) | Inserção de 1 px de margem superior na primeira seção, testando o limiar mínimo de detecção |
| Alteração de cor de borda fina | Troca da cor do *border* de 2 px de espessura em dois dos seis cards (a espessura é fixa; o que varia é a cor) |
| Remoção de elemento | Ocultação (`display: none`) de um card, provocando reposicionamento de elementos adjacentes |
| Troca de família de fonte | Substituição de *font-family* por fonte serifada em dois blocos de texto, alterando métricas tipográficas sem mudar o tamanho nominal |
| Imagem idêntica (controle) | Captura sem mutação em *baseline* e *current*, ambas com congelamento, para verificar ausência de falso positivo |

Fonte: Do autor.

Não foi conduzida varredura sistemática dos limiares (por exemplo, curva ROC sobre conjunto rotulado), limitação registrada na subseção *Validade e Limitações*.

Para a comparação *pixel a pixel*, adotou-se tolerância de 0,1 por canal, valor padrão da biblioteca pixelmatch, e limite global de 0,1 % de pixels diferentes. Em uma captura de $1366 \times 768$ px, esse percentual corresponde a aproximadamente 1.049 pixels divergentes, o que representa um ponto de partida, não um ótimo proposto.

Para a SSIM, adotou-se limiar mínimo de 0,99 e janela de bloco de $8 \times 8$ px. O limiar não é prescrito por Wang et al. (2004) e foi escolhido por situar-se imediatamente abaixo de 1, compatível com a expectativa de quase-igualdade entre *baseline* e captura atual. O cenário de alteração de componente obteve SSIM 0,9924, valor acima do limiar de 0,99, e rece-

beu veredito PASS conforme esperado, resultado coerente com a caracterização de Wang et al. (2004), pois a alteração é cromática e não afeta a estrutura perceptual da cena.

Para a abordagem por regiões, adotou-se grade de $4 \times 6$ (24 células) e limite de 1,0 % por célula, dez vezes mais permissivo do que o limite global de pixel, pois variações localizadas ocupam proporção maior dentro de uma célula do que sobre a imagem inteira.