RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

Planejamento, execução e análise de testes de funcionalidade no sistema "Conecta Canaã": uma abordagem baseada na norma IEEE 829-2008

Planning, execution and analysis of functionality tests in the "Conecta Canaã" system: an approach based on IEEE 829-2008

Planificación, ejecución y análisis de pruebas de funcionalidad en el sistema "Conecta Canaã": un enfoque basado en la norma IEEE 829-2008

Ramon Barbosa Pessoa
Universidade Federal do Oeste do Pará
Instituto de Engenharia e Geociências
Santarém
E-mail: 141barbosa@gmail.com

RESUMO

A digitalização dos serviços públicos, especialmente no cenário das cidades inteligentes, exige que a qualidade do software seja impecável. Afinal, esses sistemas afetam diretamente a interação entre a administração e os cidadãos. Neste trabalho, testamos a funcionalidade do Conecta Canaã, a plataforma de gestão de ocorrências urbanas da Prefeitura de Canaã dos Carajás, no Pará. Seguindo a norma IEEE 829-2008 e o modelo de qualidade ISO/IEC 25010:2011, planejamos, projetamos e executamos doze casos de teste automatizados usando o *framework* Selenium, em um estudo de caso técnico conduzido sobre sistema real. Cobrimos três perfis de usuário e sete requisitos funcionais priorizados. Todos os casos de teste foram aprovados, evidenciando a corretude funcional dos requisitos avaliados e a conformidade do comportamento observado com as especificações, nos limites do escopo verificado. Esses resultados documentam um protocolo de testes funcionais que pode ser replicado em outros sistemas de e-government em municípios brasileiros.

Palavras-chave: Testes de Software; Qualidade de Software; IEEE 829-2008; Automação de Testes; Cidades Inteligentes.

ABSTRACT

The digitalization of public services, especially in the smart city setting, demands that software quality be impeccable. After all, these systems directly affect the interaction between the administration and citizens. In this work, we tested the functionality of Conecta Canaã, the urban occurrence management platform of the Municipality of Canaã dos Carajás, in Pará. Following the IEEE 829-2008 standard and the ISO/IEC 25010:2011 quality model, we planned, designed and executed twelve automated test cases using the Selenium framework, in a technical case study conducted on a real system. We covered three user profiles and seven prioritized functional requirements. All test cases passed, evidencing the functional correctness of the evaluated requirements and the conformity of the observed behavior with the specifications, within the limits of the verified scope. These results document a functional testing protocol that can be replicated in other e-government systems in Brazilian municipalities.

Keywords: Software Testing; Software Quality; IEEE 829-2008; Test Automation; Smart Cities.

1 INTRODUÇÃO

O rápido crescimento da urbanização está entre os fenômenos mais determinantes do século XXI e impõe uma pressão crescente sobre a necessidade de resposta dos serviços públicos. No Brasil, a taxa de urbanização alcançou mais de 84% em 2005, e estimativas da Organização das Nações Unidas

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

sugerem que atingirá mais de 93% até 2050, o equivalente a cerca de 237 milhões de pessoas concentradas em centros urbanos (R. Machado, 2023). Tal demanda exige modelos de gestão mais eficientes e, acima de tudo, digitalizados, capazes de criar uma relação entre o poder público e um número cada vez maior de cidadãos.

É com base nesse cenário que o modelo de cidades inteligentes se firma como uma possível resposta. Uma cidade torna-se inteligente quando integra de forma coordenada as Tecnologias da Informação e Comunicação à sua infraestrutura convencional, empregando recursos digitais para ampliar a eficiência urbana (Batty et al., 2012). No Brasil, entretanto, a produção científica e a implementação de tais iniciativas ainda se concentram em poucas regiões, o que revela um campo ainda embrionário e desigualmente distribuído (Lazzaretti et al., 2019). Como expressão setorial deste paradigma, a informatização do serviço público tem sido, segundo Souza et al. (2025), vetor decisivo de modernização do Estado, alinhada à meta de instituições mais eficazes, transparentes e responsáveis preconizada pela Agenda 2030.

Essa digitalização materializa-se no conceito de *smart governance*, entendido como a integração entre administração pública, cidadãos e entidades privadas por meio de portais e plataformas digitais que ampliam o acesso aos serviços públicos (R. Machado, 2023). Ao reconhecer cada vez mais o papel da tecnologia no aperfeiçoamento dos processos internos, no fortalecimento da transparência e na eficiência da oferta de serviços, os órgãos governamentais passam a depender de softwares que integrem informações, controlem fluxos e otimizem recursos, ampliando a confiabilidade dos dados e a transparência no uso dos bens públicos (Souza et al., 2025). Tal dependência transfere para o software a responsabilidade por funções antes mediadas por servidores humanos, tornando a sua correção funcional um requisito tão crítico quanto a própria existência do serviço.

É como instância concreta desse movimento que se situa o sistema web Conecta Canaã, empregado pela Prefeitura Municipal de Canaã dos Carajás (PMCC) para gerir ocorrências urbanas relatadas por cidadãos e monitoradas por dispositivos de Internet das Coisas. O município, localizado no sudeste do Pará, registrou crescimento populacional de 188,73% entre 2010 e 2022 — de 26.716 para 77.079 habitantes —, o que equivale a uma taxa anual de 9,67% (Instituto Brasileiro de Geografia e Estatística, 2022), impondo demandas crescentes sobre serviços públicos cuja oferta não acompanhou proporcionalmente esse ritmo. O Conecta Canaã integra um projeto de cidade inteligente baseada em *Internet of Things* (IoT), desenvolvido em cooperação entre UFPA, UFOPA e Unifesspa, e acumula mais de 2.600 ocorrências atendidas pelas 8 secretarias municipais desde meados de 2022, com taxa de resolução consistentemente superior a 65% ao longo dos três anos de operação (Bacelar et al., 2025).

Para que uma aplicação de e-government cumpra esse papel, porém, a qualidade do software deixa

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

de ser atributo desejável e passa a ser condição inegociável. Entende-se qualidade de software como a conformidade sistemática do produto com os requisitos funcionais e não funcionais especificados, assegurada ao longo de todo o processo de desenvolvimento (Bartié, 2002; Pressman & Maxim, 2021). O modelo normativo de referência adotado neste trabalho para avaliá-la é a ISO/IEC 25010 em sua edição de 2011 (International Organization for Standardization, 2011), que organiza a qualidade do produto em oito características e desdobra a **adequação funcional** — a mais diretamente associada aos testes funcionais — em três sub-características: completeza, corretude e apropriabilidade (Estdale & Georgiadou, 2018). Registre-se que a norma recebeu revisão em 2023 (ISO/IEC 25010:2023), que reorganiza o modelo em nove características; a edição de 2011 foi mantida por ser a vigente à época do planejamento e da execução dos testes e por preservar inalterada a definição de adequação funcional e de suas três sub-características, eixo analítico deste estudo. A Figura 1 sintetiza o modelo, destacando a característica e as sub-características que orientam a análise conduzida neste trabalho.

A relevância dessa avaliação é simultaneamente técnica, econômica e social. Desde os primeiros sistemas comerciais, a presença de um grande número de defeitos é uma constante que afeta usabilidade, funcionalidade e confiabilidade, com reflexos diretos sobre o negócio (Rios & Moreira, 2006). O peso econômico desse fenômeno é expressivo: defeitos detectados apenas na fase de manutenção podem custar até cem vezes mais do que se corrigidos na fase de requisitos (Fernandes, 2020), e o relatório do *National Institute of Standards and Technology* estimou que o custo do software defeituoso equivalia a cerca de 1% do Produto Interno Bruto norte-americano (Rios & Moreira, 2006). À dimensão econômica soma-se a social: em aplicações de governo inteligente, a baixa qualidade compromete diretamente a confiança pública nos serviços digitais (Purbaratri & Krisnawaty, 2023). Não por acaso, a aplicação sistemática de testes de funcionalidade é reconhecida como fator determinante da qualidade do produto de software (Andrea, 2007), a ponto de os modelos de maturidade institucionalizarem essa prática — o *Capability Maturity Model* a partir do nível 3 (Paulk et al., 1993) e o MPS.BR a partir do nível B (SOFTEX, 2016).

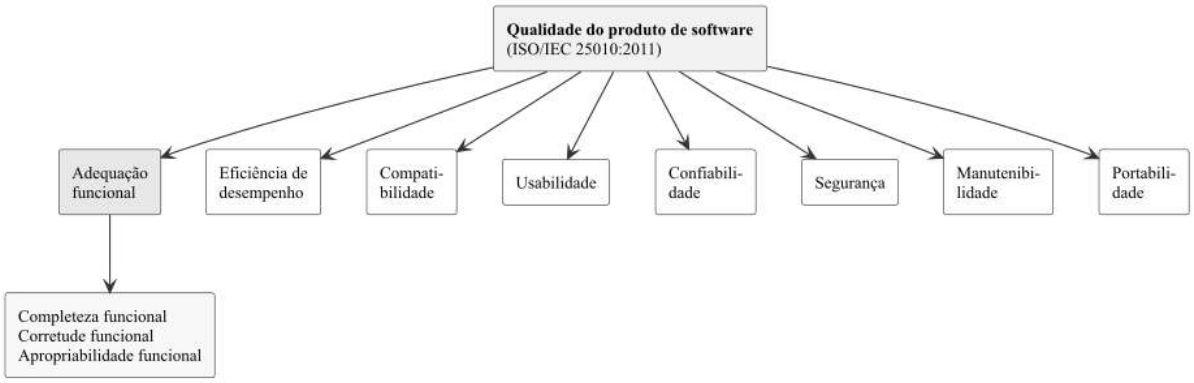
Figura 1 – Modelo de qualidade do produto de software da ISO/IEC 25010:2011, com destaque

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

para a adequação funcional



Fonte: Adaptado de International Organization for Standardization (2011).

O risco, entretanto, não reside apenas na ausência de testes, mas na sua condução sem método. Abordagens *ad hoc*, desprovidas de protocolo definido, resultam em menor cobertura, defeitos não detectados e custos adicionais de desenvolvimento e manutenção — limitação especialmente crítica em sistemas governamentais (Lima, 2023). Experiências recentes no cenário brasileiro confirmam o ganho obtido ao substituir essa informalidade por processos estruturados: Lima (2023) elaboraram casos de teste funcionais para a API de pagamentos do Governo do Rio Grande do Norte, superando uma abordagem anterior sem protocolo; Gomes (2024) construíram uma estratégia de automação para o Tribunal de Contas do Estado do Rio Grande do Norte que revelou defeitos reais, inclusive de segurança; e Fernandes (2020) demonstraram a viabilidade do Selenium WebDriver na automação de testes em sistemas institucionais. Esses trabalhos, porém, concentram-se em órgãos estaduais e federais. Configura-se, assim, a lacuna que motiva este estudo: a ausência, na literatura nacional, de relatos de aplicação de um processo de teste funcional documentado e automatizado — articulando norma de documentação, modelo de qualidade e automação — a sistemas de gestão municipal integrados a iniciativas de cidades inteligentes, contexto em que restrições de equipe, orçamento e maturidade de processo diferem substancialmente das esferas estadual e federal.

Diante desse quadro, a pergunta que orienta este trabalho é se os requisitos funcionais do Conecta Canaã são atendidos conforme especificados, de modo a garantir a confiabilidade exigida pela gestão pública digital. O objetivo geral é submeter o sistema a testes de funcionalidade que assegurem o atendimento às necessidades operacionais da Prefeitura de Canaã dos Carajás. Especificamente, pretende-se planejar e projetar os testes segundo a norma IEEE 829-2008; implementá-los com o *framework* Selenium; executá-los nos três perfis de usuário e nos cenários operacionais relevantes; analisar os resultados à luz da adequação funcional da ISO/IEC 25010; e documentar o processo como protocolo replicável. Articulam-se, assim, os três pilares que sustentam a abordagem: a IEEE 829-2008, que fornece o protocolo de documentação e execução; a ISO/IEC 25010, que define os critérios

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

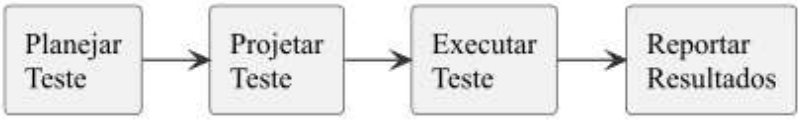
RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.
Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

de qualidade; e o Selenium, que viabiliza a automação. Cabe reconhecer que a IEEE 829-2008 foi formalmente substituída pela família ISO/IEC/IEEE 29119 (International Organization for Standardization, 2022); sua adoção justifica-se pela simplicidade e concisão de seus artefatos documentais — mais adequadas ao porte do projeto e da equipe envolvida do que a estrutura multipartite da norma sucessora —, pela ampla base de literatura e ferramentas que a referenciam e pelo fato de seus documentos serem diretamente mapeáveis aos artefatos correspondentes da ISO/IEC/IEEE 29119-3, o que preserva a validade metodológica do protocolo.

2 MATERIAIS E MÉTODOS

Esta pesquisa caracteriza-se como um estudo de caso técnico de natureza aplicada — uma avaliação empírica de software conduzida sobre um sistema real, com abordagem quantitativa e descritiva, sem manipulação de variáveis, grupos de comparação ou hipóteses experimentais —, e estrutura-se sobre as quatro etapas preconizadas pela norma IEEE 829-2008 — planejamento, projeto, execução e análise de resultados —, que constituem, simultaneamente, o referencial e o procedimento do estudo. A norma fornece um conjunto padronizado de diretrizes para documentar e executar testes de software, promovendo consistência, rastreabilidade e comunicação entre os envolvidos (IEEE Computer Society, 2008). A Figura 2 sintetiza o encadeamento dessas etapas: no *planejamento*, definem-se objetivo, abordagem, itens sob teste, recursos e riscos; no *projeto*, especificam-se casos de teste, dados e ambiente; na *execução*, aplicam-se os casos e registram-se os resultados, sendo cada divergência tratada como defeito; e na *análise*, os dados são avaliados para atestar a eficácia do sistema. Como lente interpretativa dessa última etapa, adotou-se a adequação funcional da ISO/IEC 25010, cujas sub-características — completeza, corretude e apropriabilidade — orientam o julgamento sobre o atendimento aos requisitos (Estdale & Georgiadou, 2018).

Figura 2 – Fluxo parcial do processo IEEE 829-2008



Fonte: Adaptado de IEEE Computer Society (2008).

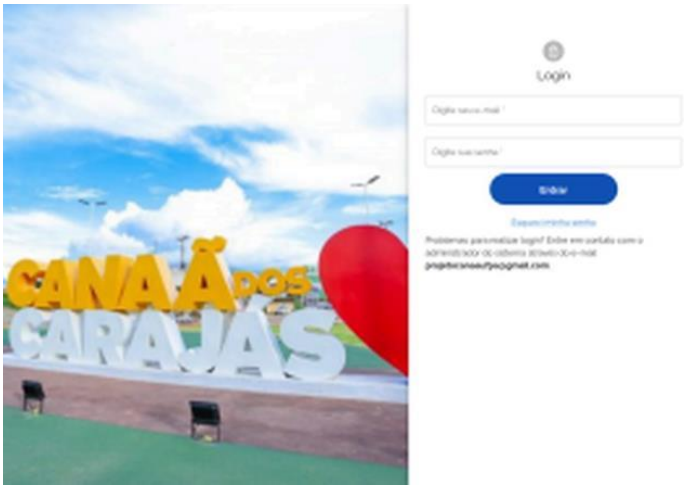
2.1 Sistema Conecta Canaã

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.
Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

O Conecta Canaã Web[1] é um ambiente de gestão pública que permite à PMCC administrar solicitações de manutenção urbana, relatos de infrações e demandas vinculadas ao Código de Posturas de Canaã dos Carajás (Lei Municipal n. 694/2015) e à Ouvidoria. Suas funcionalidades distribuem-se por três perfis de usuário: o **Executor**, responsável por resolver as ocorrências a ele direcionadas; o **Controlador**, que designa executores, interage com os cidadãos e altera o estado das ocorrências; e o **Administrador**, encarregado de elaborar relatórios e cadastrar novos usuários. Desenvolvido em JavaScript e TypeScript com a biblioteca React na camada de interface, o sistema adota arquitetura cliente-servidor em três camadas — apresentação (navegador Google Chrome), lógica (servidor Ubuntu com NGINX) e dados (MongoDB 7.0) — e está hospedado na infraestrutura de nuvem do município. A Figura 3 exibe sua tela de acesso. Os testes relatados neste artigo foram planejados e executados em agosto de 2024, sobre a versão então vigente do sistema em ambiente de homologação. O sistema recebeu atualizações posteriores àquela data, de modo que os resultados aqui relatados devem ser lidos como um retrato da versão avaliada em agosto de 2024, e não do estado atual da plataforma.

Figura 3 – Tela de login do sistema Conecta Canaã



Fonte: Autor (2024).

2.2 Planejamento dos testes

A etapa de planejamento partiu do levantamento dos requisitos funcionais do sistema, derivados das necessidades e expectativas dos usuários (F. N. R. Machado, 2015; Sommerville, 2011). O levantamento identificou trinta requisitos funcionais no conjunto do sistema, distribuídos pelos três

[1] Disponível em: https://conecta.smartcitycanaadoscarajas.com.br/

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

perfis de usuário. Dado esse volume, a seleção pautou-se por três critérios — frequência de uso, relevância para o funcionamento central e complexidade do requisito —, aplicados pelo próprio autor, com base em seu conhecimento do sistema e da operação da PMCC. A *frequência de uso* considerou a recorrência da funcionalidade na rotina dos três perfis; a *relevância* privilegiou os requisitos sem os quais o fluxo central de registro, tratamento e acompanhamento de ocorrências ficaria inviabilizado; e a *complexidade* favoreceu funcionalidades com maior número de passos de interação e regras de validação, por concentrarem maior risco de defeito. Não se adotou escala numérica, atribuição de pesos ou matriz de decisão formal, tampouco houve participação de outros avaliadores na priorização — limitação registrada na Seção 4.1. O processo resultou nos sete requisitos distribuídos pelos três perfis apresentados na Tabela 1, o que corresponde a cerca de um quarto do universo de requisitos funcionais do sistema.

Tabela 1 – Requisitos funcionais selecionados

Perfil	ID	Requisito Funcional
Controlador	RF 01	Adicionar Comentário
Executor	RF 02	Visualizar Ocorrência
Executor	RF 03	Visualizar Comentário
Administrador	RF 04	Realizar Login
Administrador	RF 05	Cadastrar Usuário
Administrador	RF 06	Aplicar Filtros de Busca
Administrador	RF 07	Visualizar Ocorrências

Fonte: Autor (2024).

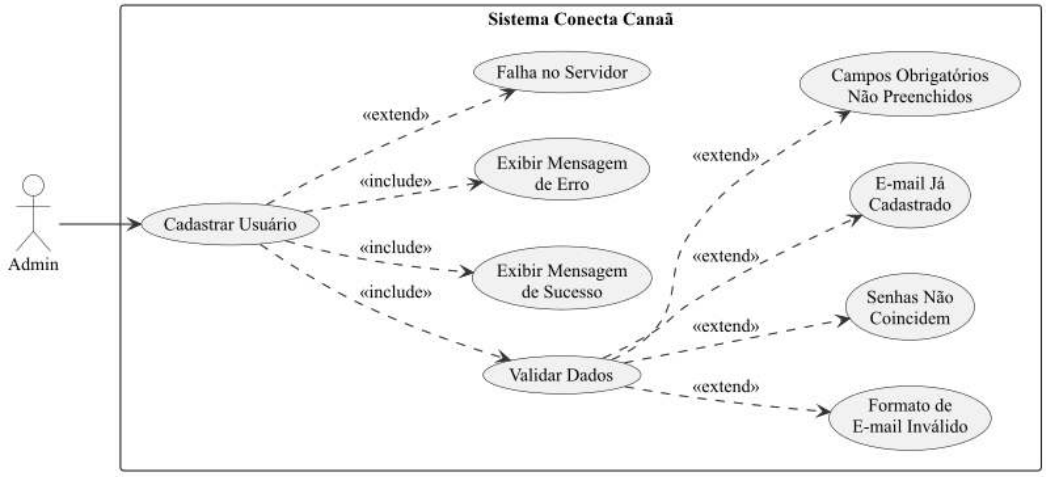
A definição dos casos de teste apoiou-se nos casos de uso correspondentes, que descrevem as funcionalidades sob a ótica do usuário e facilitam a derivação de testes capazes de contemplar todos os requisitos (Sommerville, 2011). As Figuras 4 e 5 ilustram, respectivamente, os casos de uso de cadastro de usuário e de listagem de ocorrências. Como assinalam Craig e Jaskiel (2002), um caso de teste especifica uma condição a ser verificada, com valores de entrada, restrições de execução e resultado esperado; no modelo de processo adotado (Sommerville, 2011), sintetizado na Figura 6, essa especificação constitui a etapa inicial da implementação dos testes. A partir desse mapeamento, foram elaborados três casos de teste para os perfis Executor e Controlador e nove para o Administrador, totalizando doze, cuja correspondência com os requisitos é detalhada na Tabela 2.

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.
Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

Em termos quantitativos, a suíte cobre os sete requisitos selecionados com média de 1,7 caso por requisito: RF 01 a RF 03 contam com um cenário de fluxo principal cada; RF 04 a RF 06, com um cenário válido e um inválido cada; e RF 07, com três cenários de visualização. Três dos doze casos (25%) exercitam fluxos de exceção — credenciais inválidas (TC 05), cadastro duplicado (TC 07) e busca por código inexistente (TC 09) —, tendo a totalidade dos doze casos sido executada de forma automatizada. A suíte não contempla, nesta versão, técnicas sistemáticas de teste negativo como análise de valores-limite, campos vazios, dados malformados, comprimentos máximos, caracteres especiais e combinações inválidas de entrada; essa restrição decorreu da priorização dos fluxos principais e de exceção mais frequentes na operação real do sistema, dentro do prazo disponível para o estudo, e é reconhecida como limitação na Seção 4.1.

Figura 4 – Caso de uso de cadastro de usuário



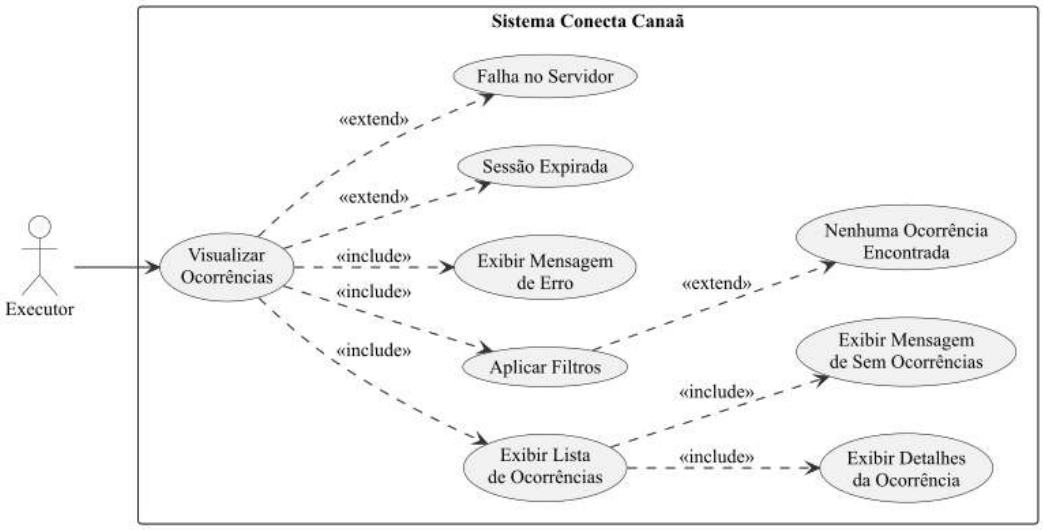
Fonte: Autor (2024).

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.

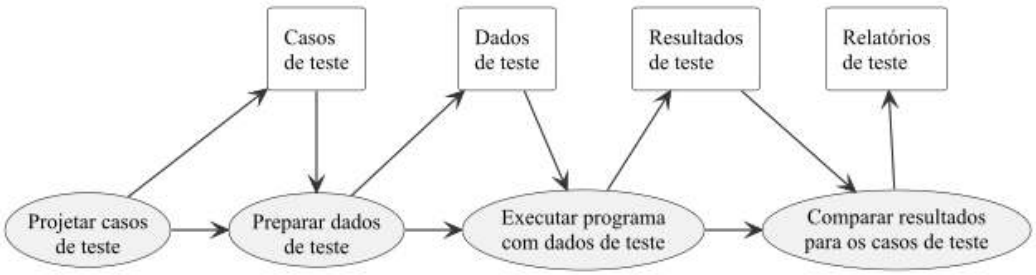
Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

Figura 5 – Caso de uso de listagem de ocorrências



Fonte: Autor (2024).

Figura 6 – Modelo do processo de teste de software



Fonte: Sommerville (2011).

2.3 Projeto dos testes

Definidos os casos, optou-se por implementá-los como testes funcionais automatizados. A atividade de teste pode ser classificada em quatro dimensões: o estado do teste (unidade, integração ou sistema), a técnica empregada (caixa-preta, caixa-branca, regressão), a meta perseguida (funcional, de interface, de desempenho, de segurança) e o ambiente em que se realiza, como as aplicações web (Molinari, 2008). O presente estudo posiciona-se na meta funcional, sob a técnica de caixa-preta, no ambiente web. O teste funcional avalia o software a partir de suas especificações, sem acesso ao código-fonte — razão pela qual também é denominado teste de caixa-preta —, buscando disparidades entre o comportamento observado e o esperado (Myers et al., 2012). Sua automação, que delega a *scripts* a execução e a verificação dos resultados, reduz o esforço humano, acelera a repetição e diminui custos (Bernardo & Kon, 2008; Dustin et al., 2009), benefício expressivo quando se considera

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

que a atividade de testes pode consumir de 30% a 35% do esforço total de um projeto (Mustofa & Fajar, 2018).

Tabela 2 – Mapeamento dos casos de teste

RF	TC	Caso de Teste
RF 01	TC 01	Adicionar comentário à ocorrência
RF 02	TC 02	Visualizar ocorrência do cidadão
RF 03	TC 03	Visualizar comentário da ocorrência
RF 04	TC 04	Login com credenciais válidas
RF 04	TC 05	Login com credenciais inválidas
RF 05	TC 06	Cadastrar novo usuário administrador
RF 05	TC 07	Cadastrar usuário já existente
RF 06	TC 08	Aplicar filtro de busca por código
RF 06	TC 09	Busca com código inexistente
RF 07	TC 10	Visualizar ocorrências em aberto
RF 07	TC 11	Visualizar ocorrências concluídas
RF 07	TC 12	Visualizar ocorrências sem localização

Fonte: Autor (2024).

A automação foi conduzida com o *framework* Selenium, referência consolidada para aplicações web. Concebido em 2004 como alternativa à ineficiência dos testes manuais (Gomes, 2024), o Selenium WebDriver acessa o DOM da página e interage diretamente com seus elementos — cliques, preenchimento de formulários e validação de retornos —, reproduzindo com fidelidade as ações reais do usuário (Fernandes, 2020; Romanini & Sotto, 2019). Sua adoção apoiou-se em três fatores: a eficiência na cobertura de múltiplos perfis e cenários, a compatibilidade com o Google Chrome — plataforma de acesso ao sistema — e a integração com a linguagem Python, utilizada no desenvolvimento dos *scripts*.

A configuração da automação observou os atributos que o ISTQB associa à sua eficácia: a arquitetura de automação de testes, alinhada à arquitetura do sistema sob teste; a minimização da dependência entre a interface gráfica e a lógica de interação; e a escolha de um *framework* intuitivo e bem documentado, que equilibre custos, benefícios e riscos (ISTQB, 2016). A combinação de Selenium e

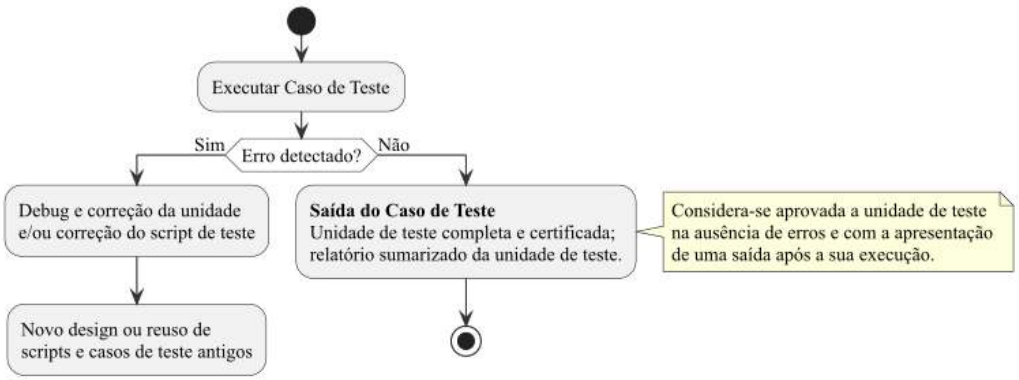
Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.
Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

Python atende a esses requisitos, além de favorecer a manutenção e a reexecução dos *scripts* a cada nova versão do sistema.

Os testes ocorreram em ambiente de homologação isolado da produção, condição que evitou qualquer interferência sobre os dados reais do município durante a verificação. Nesse ambiente, criaram-se os três perfis de usuário e extraiu-se uma amostra de 30 ocorrências do banco de dados, composta por 20 ocorrências em aberto e 10 concluídas — os registros sem localização, exigidos pelo TC 12, constituem subconjunto desses dois grupos, e não uma terceira categoria, uma vez que a ausência de coordenada geográfica é atributo independente do estado de tramitação da ocorrência. A composição contempla, assim, os cenários exigidos pelos casos de teste do *dashboard* (TC 10 a TC 12). Essa amostra serviu de referência para confrontar os valores exibidos pela interface com os registros efetivamente armazenados, tornando objetiva a verificação da apropriabilidade funcional. A descrição dos cenários seguiu o formulário padrão de Lino (2021), reconhecido pela objetividade. O fluxo de execução adotado é apresentado na Figura 7: obtido resultado conclusivo, o caso é encerrado e revisado apenas diante de alterações na funcionalidade; caso contrário, gera-se um relatório de defeito encaminhado à equipe de desenvolvimento.

Figura 7 – Fluxo de execução dos casos de teste



Fonte: Lino (2021).

2.4 Ambiente de execução e reprodutibilidade

Para viabilizar a reprodução independente dos testes, registram-se as condições de execução. Os *scripts* foram desenvolvidos em Python 3.10 com Selenium 4.21.0 e executados com o *framework* pytest 8.3.2 em sistema operacional Ubuntu 22.04, no navegador Google Chrome em sua versão estável corrente à época da execução (agosto de 2024), com o ChromeDriver provisionado automaticamente pelo pacote chromedriver-autoinstaller 0.6.4, que resolve em tempo de execução a

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

versão do *driver* compatível com o navegador instalado — razão pela qual a suíte permanece executável em versões posteriores do Chrome sem ajuste manual. As dependências estão declaradas no arquivo requirements.txt do repositório (Pessoa, 2024), e a execução dá-se pelo comando pytest, com parâmetros de ambiente (URL do sistema e credenciais dos perfis de teste) fornecidos por variáveis de ambiente. A correspondência entre *scripts* e casos está documentada no arquivo README do repositório, havendo função automatizada para cada um dos doze casos: test_controller_report_executor.py cobre TC 01 a TC 03; test_login.py, TC 04 e TC 05; test_register_user.py, TC 06 e TC 07; test_occurrence_no_location.py, TC 08, TC 09 e TC 12; e test_dashboard.py, TC 10 e TC 11. Os dados de teste — perfis de usuário e amostra de ocorrências — foram criados sinteticamente no ambiente de homologação, com apoio da biblioteca Faker para a geração dos dados de cadastro, sem uso de dados pessoais reais. Entre as execuções, o banco de homologação era restaurado a partir de um *dump* previamente salvo, o que assegurava estado inicial idêntico a cada rodada e permitia a reexecução dos casos que alteram dados, como o cadastro de usuário (TC 06). Cada caso foi executado cinco vezes, com resultado consistente em todas as repetições e registro no formulário padrão e geração de relatório HTML pelo pytest-html-reporter, versionado no diretório report do repositório; a repetição quíntupla de cada caso serviu, ela própria, como controle de testes instáveis (*flaky tests*), não tendo sido observada variação de desfecho entre as rodadas.

2.5 Execução

A execução materializou-se em *scripts* de automação desenvolvidos em Python com o Selenium e versionados em repositório público[2] (Pessoa, 2024). A Figura 8 exemplifica a rotina de autenticação: a função login abre a URL informada, localiza e preenche os campos de e-mail e senha, submete o formulário e verifica a presença de um elemento indicativo de acesso bem-sucedido, retornando True ou False. Cada caso foi documentado segundo o formulário padrão, com código, perfil, requisito associado, pré-condição, procedimento e resultado esperado. As Tabelas 3, 4 e 5 reproduzem um exemplo representativo de cada perfil — a inclusão de parecer técnico pelo Controlador, a visualização restrita de ocorrências pelo Executor e a tentativa de autenticação com credenciais inválidas pelo Administrador —, enquanto os demais casos seguem a estrutura mapeada na Tabela 2.

[2] Disponível em: https://github.com/ramon141/-canaa-conecta-web-tester

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

Figura 8 – Rotina de autenticação automatizada com Selenium

```
def login(browser, email, password) -> bool:
    browser.driver.get(os.getenv('HOME_PAGE'))

    email_field = browser.driver.find_element(
        By.XPATH, "//div[contains(@class,"
        "'MuiFormControl-root')][1]//input")
    email_field.clear()
    email_field.send_keys(email)

    password_field = browser.driver.find_element(
        By.XPATH, "//div[contains(@class,"
        "'MuiFormControl-root')][2]//input")
    password_field.clear()
    password_field.send_keys(password)

    browser.driver.find_element(
        By.XPATH, "//button[@type='submit']").click()

    try:
        browser.wait.until(EC.presence_of_element_located
        (
            (By.XPATH, "//*[contains(text(),"
             "'Ocorrencias Comuns')]")))
        return True
    except:
        return False
```

Fonte: Autor (2024).

Tabela 3 – TC 01 – Adicionar comentário à ocorrência

Campo	Descrição
Código	TC 01
Perfil	Controlador
RF	RF 01
Pré-condição	Estar na tela de login
Procedimento	1. Login com credenciais de controlador 2. Selecionar ocorrência e clicar em Ver Detalhes 3. Preencher campo de parecer técnico 4. Clicar em Realizar Parecer 5. Confirmar envio
Resultado esperado	"Parecer cadastrado com sucesso."
Data	16/08/2024
Resultado	Aprovado

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

Fonte: Autor (2024).

Tabela 4 – TC 02 – Visualizar ocorrência do cidadão

Campo	Descrição
Código	TC 02
Perfil	Executor
RF	RF 02
Pré-condição	Estar na tela de login
Procedimento	1. Realizar login com credenciais de executor
Resultado esperado	Visualizar apenas as ocorrências direcionadas ao executor específico
Data	16/08/2024
Resultado	Aprovado

Fonte: Autor (2024).

Tabela 5 – TC 05 – Login com credenciais inválidas

Campo	Descrição
Código	TC 05
Perfil	Administrador
RF	RF 04
Pré-condição	Estar na página de login
Procedimento	1. Preencher e-mail inválido 2. Preencher senha inválida 3. Clicar no botão Entrar
Resultado esperado	“Houve um problema com o login, verifique suas credenciais!”
Data	16/08/2024
Resultado	Aprovado

Fonte: Autor (2024).

3 RESULTADOS

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.
Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

A execução dos doze casos de teste seguiu integralmente o procedimento planejado, e seus desfechos foram consolidados na Tabela 6, organizada por requisito funcional e perfil de usuário. Todos os casos foram aprovados, perfazendo taxa de êxito de 100%.

No perfil Controlador, o TC 01 confirmou a inclusão de parecer técnico em uma ocorrência: após o preenchimento do campo técnico e a confirmação do envio, o sistema retornou a mensagem "Parecer cadastrado com sucesso.", atendendo ao RF 01. No perfil Executor, os TC 02 e TC 03 evidenciaram que cada usuário visualiza exclusivamente as ocorrências e os comentários a ele direcionados, sem acesso aos registros de outros executores, o que confirma a preservação do isolamento entre perfis previsto nos RF 02 e RF 03.

No perfil Administrador, concentrou-se o maior número de casos, distribuídos por quatro requisitos. O módulo de autenticação aceitou as credenciais válidas, conduzindo ao *dashboard* (TC 04), e bloqueou as inválidas com a mensagem de alerta correspon- dente (TC 05), atendendo ao RF 04. O cadastro de usuários operou corretamente ao registrar um novo administrador (TC 06) e ao recusar o registro de e-mail já existente (TC 07), preservando a unicidade exigida pelo RF 05. Os filtros de busca, por sua vez, retornaram resultados precisos quando consultado um código válido (TC 08) e sinalizaram a ausência de correspondência quando o código inexistia (TC 09), em conformidade com o RF 06.

Por fim, os casos associados ao RF 07 verificaram a fidelidade do *dashboard*: aplicado o filtro de período entre 01/01/2021 e 16/08/2024, o sistema contabilizou 20 ocorrências em aberto (TC 10) e 10 concluídas (TC 11) — totalizando as 30 da amostra —, além de exibir corretamente as ocorrências sem localização registrada (TC 12), subconjunto dos dois grupos anteriores. Em todos os casos, os valores apresentados pela interface coincidiram com os 30 registros extraídos do banco de homologação, sem discrepâncias.

Tabela 6 – Resultados consolidados dos casos de teste

TC	RF	Descrição	Res.
TC 01	RF 01	Adicionar comentário	OK
TC 02	RF 02	Visualizar ocorrência	OK
TC 03	RF 03	Visualizar comentário	OK
TC 04	RF 04	Login válido	OK
TC 05	RF 04	Login inválido	OK
TC 06	RF 05	Cadastrar usuário	OK

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

RCMOS – Revista Científica Multidisciplinar O Saber.
ISSN: 2675-9128. São Paulo-SP.
Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

TC	RF	Descrição	Res.
TC 07	RF 05	Usuário duplicado	OK
TC 08	RF 06	Filtro por código	OK
TC 09	RF 06	Filtro inexistente	OK
TC 10	RF 07	Ocorrências em aberto	OK
TC 11	RF 07	Ocorrências concluídas	OK
TC 12	RF 07	Sem localização	OK

Fonte: Autor (2024).

4 DISCUSSÃO

A aprovação integral dos casos de teste deve ser interpretada com precisão: significa que os doze casos passaram nas condições avaliadas, e é nesse escopo que se dá a leitura pelas sub-características da adequação funcional da ISO/IEC 25010 (Estdale & Georgiadou, 2018). A evidência mais robusta refere-se à *corretude funcional* — exatidão dos resultados produzidos —, sustentada tanto pelos testes de autenticação (TC 04 e TC 05) e de cadastro com verificação de unicidade (TC 06 e TC 07), que confirmaram o comportamento correto nos fluxos válidos e de exceção, quanto pela coincidência entre os valores exibidos pelo *dashboard* e os registros do banco de homologação (TC 10 a TC 12). Quanto à *completeza funcional* — abrangência das funções frente às tarefas do usuário —, os resultados oferecem apenas evidência parcial: os doze casos cobrem os sete requisitos priorizados, mas não o universo de funcionalidades do sistema, de modo que não se pode afirmar a completeza funcional do produto como um todo. De forma análoga, a *apropriabilidade funcional* — adequação das funções aos objetivos especificados — encontra indícios favoráveis no fato de os fluxos avaliados cumprirem as tarefas a que se destinam, mas sua demonstração plena exigiria avaliação junto aos usuários finais, o que extrapola o desenho deste estudo.

Esses achados ratificam a eficácia de um processo de teste estruturado em quatro etapas frente à informalidade das abordagens *ad hoc*, apontada por Lima (2023) como fonte de baixa cobertura e confiabilidade em sistemas governamentais. A automação com Selenium ampliou esse ganho ao cobrir múltiplos perfis com menor margem de erro humano e maior assertividade do que a verificação manual (Romanini & Sotto, 2019), em consonância com as diretrizes do ISTQB sobre a relação entre custo, benefício e risco (ISTQB, 2016).

O cotejo com a literatura situa a contribuição em dois eixos. Quanto à viabilidade técnica, os

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

resultados convergem com Fernandes (2020), que comprovaram a adequação do Selenium WebDriver à automação de testes em sistemas institucionais web, estendendo agora essa evidência ao domínio da gestão municipal. Quanto ao desfecho, o contraste com Gomes (2024) é elucidativo: enquanto a automação aplicada ao Tribunal de Contas do Rio Grande do Norte revelou defeitos reais, inclusive de segurança, o Conecta Canaã não acusou falhas funcionais. Tal diferença não diminui o valor do procedimento — ao contrário, um conjunto de testes que não encontra defeitos só é informativo quando deriva de um protocolo rigoroso, capaz de distinguir a ausência de falhas da ausência de verificação — e encontra explicação plausível no estado de maturidade do sistema avaliado: à época dos testes, o Conecta Canaã encontrava-se em produção havia cerca de dois anos, com mais de 2.600 ocorrências processadas (Bacelar et al., 2025), de modo que os fluxos principais aqui verificados já haviam sido extensivamente exercitados — e depurados — pelo uso real. Some-se a isso que a suíte privilegiou fluxos principais e de exceção frequentes, cenários em que defeitos remanescentes são menos prováveis em um sistema estável, o que reforça o argumento de Andrea (2007) sobre o papel dos testes sistemáticos na qualidade do produto sem autorizar generalizações sobre a ausência de defeitos.

Para além da detecção de defeitos, a adoção da IEEE 829-2008 agregou valor pela padronização documental. A formalização de cada caso em código, perfil, requisito associado, pré-condição, procedimento e resultado esperado estabeleceu uma rastreabilidade explícita entre requisitos e evidências de teste, condição que sustenta auditorias, facilita a comunicação na equipe e permite reexecutar a verificação a cada alteração do sistema — benefício inacessível às abordagens informais. Essa rastreabilidade é particularmente relevante na gestão pública, em que a prestação de contas e a transparência integram a própria finalidade do sistema.

No plano das implicações, o resultado documenta um caso de aplicação de processo estruturado de testes em sistema municipal de gestão pública e contribui para o adensamento — ainda incipiente — das iniciativas brasileiras de cidades inteligentes (Lazzaretti et al., 2019); ao assegurar verificação sistemática de funcionalidades críticas, o processo pode contribuir, indiretamente, para a confiança pública nos serviços digitais, elemento crítico de sua legitimidade (Purbaratri & Krisnawaty, 2023), embora essa relação não tenha sido objeto de medição neste estudo.

4.1 Ameaças à validade

Quanto à *validade de construto*, a taxa de aprovação de 100% mede apenas a conformidade dos doze casos nas condições avaliadas; não constitui medida de robustez, de confiabilidade global ou de ausência de defeitos, e a associação com as sub-características da ISO/IEC 25010 restringe-se ao

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

exposto no início desta seção. Quanto à *validade interna*, os testes ocorreram em ambiente de homologação, sobre amostra controlada de 30 ocorrências e dados sintéticos, condições não necessariamente equivalentes ao ambiente de produção sob carga real; ademais, os *scripts* foram elaborados e executados pelo próprio autor, sem verificação independente. A seleção dos sete requisitos também foi conduzida individualmente, sem escala numérica, pesos explícitos ou consenso entre avaliadores, o que a torna suscetível a viés de julgamento e limita a demonstração de representatividade frente ao conjunto de funcionalidades do sistema. Quanto à *validade externa*, trata-se de estudo de caso único, sobre um sistema e um município específicos: o protocolo é replicável, mas os resultados não são generalizáveis a outros sistemas sem nova aplicação. Quanto à *validade de conclusão*, embora cada caso tenha sido repetido cinco vezes com resultado estável — o que reduz o risco de desfechos fortuitos ou de testes instáveis —, o número reduzido de casos (doze) e a ausência de variação experimental entre as repetições impedem qualquer tratamento estatístico inferencial dos resultados. Registram-se, por fim, as limitações da automação adotada: os *scripts* Selenium dependem dos seletores da interface — alterações de *layout* exigem manutenção —, verificam o comportamento observável pelo navegador sem inspecionar a lógica de servidor, e a suíte não incorpora técnicas sistemáticas de teste negativo, como análise de valores-limite e dados malformados, conforme discutido na Seção 2.2.

5 CONCLUSÕES

Este trabalho aplicou, de forma sistemática, testes de funcionalidade ao sistema web Conecta Canaã, articulando a norma IEEE 829-2008 como protocolo de planejamento, projeto, execução e análise; a ISO/IEC 25010 como referência de qualidade; e o *framework* Selenium como instrumento de automação. A aprovação dos doze casos de teste nos três perfis de usuário evidencia que o sistema atende, nas condições avaliadas, aos sete requisitos funcionais selecionados, operando conforme as especificações no escopo verificado.

Três contribuições decorrem do estudo: a demonstração empírica de que a IEEE 829-2008 é aplicável e eficaz à documentação e execução de testes em sistemas municipais de e-government; a produção de evidências de corretude funcional do Conecta Canaã — com indícios parciais de completeza e apropriabilidade, restritos aos requisitos avaliados — segundo a ISO/IEC 25010; e a disponibilização de um protocolo replicável de testes automatizados para sistemas de porte semelhante, alinhado à agenda de modernização da gestão pública e aos Objetivos de Desenvolvimento Sustentável (Souza et al., 2025). Mais do que melhoria técnica, o processo estruturado garantiu rastreabilidade entre requisitos e evidências de teste, atributo que sustenta auditorias e pode contribuir para a confiança

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

dos cidadãos nos serviços públicos digitais (Purbaratri & Krisnawaty, 2023).

Reconhecem-se, todavia, limitações: os testes restringiram-se ao ambiente de homologação e a uma amostra controlada; os doze casos cobrem sete requisitos funcionais, sem esgotar as funcionalidades do sistema; e o escopo ateve-se à dimensão funcional, sem contemplar desempenho, segurança e usabilidade. Como desdobramentos, sugerem-se a incorporação de técnicas sistemáticas de teste negativo — análise de valores-limite, campos vazios, dados malformados e combinações inválidas —, a ampliação para os requisitos não funcionais das demais características da ISO/IEC 25010, a integração dos testes a um *pipeline* de integração e entrega contínuas, a exemplo da experiência de Gomes (2024), e a replicação da metodologia em outros sistemas da rede de cidades inteligentes do Pará, viabilizando estudos comparativos de qualidade entre municípios.

REFERÊNCIAS

Andrea, J. (2007). Envisioning the Next-Generation of Functional Testing Tools. *IEEE Software*, *24*. https://doi.org/10.1109/MS.2007.73

Bacelar, P., Broechl, G., Sousa, M., Kuribayashi, H., Vijaykumar, N., & Francês, C. R. (2025). Sistema Integrado Conecta Canaã: Uma Solução de Arquitetura para Mitigação de Dificuldades de Urbanização Emergente. *Anais do XXVII Congresso Ibero-Americano em Engenharia de Software (CIbSE)*.

Bartié, A. (2002). *Garantia da qualidade de software: adquirindo maturidade organizacional* (1ª ed.). Elsevier.

Batty, M., et al. (2012). Smart cities of the future. *The European Physical Journal Special Topics*, *214*(1), 481–518. https://doi.org/10.1140/epjst/e2012-01703-3

Bernardo, P. C., & Kon, F. (2008). A Importância dos Testes Automatizados. *Engenharia de Software Magazine*, *1*(3), 54–57. Obtido agosto 25, 2024, de https://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf

Craig, R. D., & Jaskiel, S. P. (2002). *Systematic Software Testing* (1ª ed.). Artech House Publishers.

Dustin, E., Garrett, T., & Gauf, B. (2009). *Implementing automated software testing: how to save time and lower costs while raising quality* (1ª ed.). Addison-Wesley Professional.

Estdale, J., & Georgiadou, E. (2018). Applying the ISO/IEC 25010 Quality Models to Software Product. *Systems, Software and Services Process Improvement. EuroSPI 2018*, *896*, 492–503. https://doi.org/10.1007/978-3-319-97925-0_42

Fernandes, G. S. (2020). *Automação de testes com Selenium WebDriver aplicada no Sistema de distribuição de disciplinas* [Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação)]. Universidade Federal de Uberlândia, Faculdade de Computação. Obtido julho 13, 2026, de https://repositorio.ufu.br/handle/123456789/31143

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

Gomes, R. L. (2024). *Elaboração e Avaliação de uma Estratégia de Automação de Testes baseada no Cypress para o TCE-RN* [Trabalho de Conclusão de Curso (Especialização em Tecnologia da Informação Aplicada)]. Universidade Federal do Rio Grande do Norte, Instituto Metrópole Digital.

IEEE Computer Society. (2008). *829-2008 — IEEE Standard for Software and System Test*. IEEE Computer Society. Fredericksburg, VA, USA.

Instituto Brasileiro de Geografia e Estatística. (2022). Censo 2022 — Canaã dos Carajás. Obtido janeiro 1, 2025, de https://www.ibge.gov.br/cidades-e-estados/pa/canaa-dos-carajas.html

International Organization for Standardization. (2011). *ISO/IEC 25010:2011 — Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. ISO/IEC. Geneva.

International Organization for Standardization. (2022). *ISO/IEC/IEEE 29119-1:2022 — Software and systems engineering — Software testing — Part 1: General concepts*. ISO/IEC/IEEE. Geneva.

ISTQB. (2016). *Advanced Level Syllabus Test Automation Engineer Version 2016*. International Software Testing Qualifications Board.

Lazzaretti, K., Sehnem, S., Bencke, F. F., & Machado, H. P. V. (2019). Cidades inteligentes: insights e contribuições das pesquisas brasileiras. *Urbe. Revista Brasileira de Gestão Urbana*, *11*, e20190118. https://doi.org/10.1590/2175-3369.011.001.e20190118

Lima, K. K. d. (2023). *Elaboração e Implementação de Casos de Testes Funcionais para a API do PagRN* [Trabalho de Conclusão de Curso (Especialização em Tecnologia da Informação Aplicada)]. Universidade Federal do Rio Grande do Norte, Instituto Metrópole Digital.

Lino, A. D. P. (2021). *LabDER — Laboratório Virtual de Ensino-Aprendizagem de Banco de Dados Relacionais: Uma abordagem de avaliação automática de Diagramas ER e SQL* [Tese de Doutorado]. Universidade de Coimbra. Obtido agosto 25, 2024, de https://hdl.handle.net/10316/95440

Machado, F. N. R. (2015). *Análise e Gestão de Requisitos de Software — Onde nascem os sistemas* (3ª ed.). Editora Érica.

Machado, R. (2023). Cidades inteligentes e gestão pública: uma relação exploratória do cenário brasileiro. *Revista Brasileira de Gestão Pública*, *2*(2), 1–9.

Molinari, L. (2008). *Teste de Software — Produzindo sistemas melhores e mais confiáveis* (4ª ed.). Editora Érica.

Mustofa, K., & Fajar, S. P. (2018). Selenium-Based Multithreading Functional Testing. *Indonesian Journal of Computing and Cybernetics Systems*, *12*(1), 63–72. https://doi.org/10.22146/ijccs.28121

Myers, G. J., Sandler, C., & Badgett, T. (2012). *The art of software testing* (3ª ed.). John Wiley & Sons.

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). Capability Maturity Model for Software, Version 1.1. *IEEE Software*, *10*(4), 18–27. https://doi.org/10.1109/52.219617

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.

Ano VII, v.2 2026 | submissão: 10/10/2026 | aceito: 15/10/2026 | publicação: 29/10/2026

Pessoa, R. B. (2024). Canaã Conecta Web Tester. Obtido novembro 19, 2024, de https://github.com/ramon141/-canaa-conecta-web-tester

Pressman, R. S., & Maxim, B. R. (2021). *Engenharia de software: uma abordagem profissional* (9ª ed.). AMGH.

Purbaratri, W., & Krisnawaty, N. (2023). Software Quality Model ISO/IEC 25010 Prioritization Smart Government Application. *Proceeding of the Perbanas International Seminar on Economics, Business, Management, Accounting and IT (PROFICIENT)*, 261–270.

Rios, E., & Moreira, T. (2006). *Teste de software* (2ª ed.). Alta Books Editora.

Romanini, I. Z., & Sotto, E. C. S. (2019). Selenium WebDriver na Evolução dos Testes Manuais. *Revista Interface Tecnológica*, *16*(2), 313–322. https://doi.org/10.31510/infa.v16i2.627

SOFTEX. (2016). *Guia Geral MPS de Software*. Associação para Promoção da Excelência do Software Brasileiro.

Sommerville, I. (2011). *Engenharia de software* (9ª ed.). Pearson Prentice Hall.

Souza, M. J. C. d., Galhardo, C. X., Teixeira, K., & Moreira, A. (2025). Desenvolvimento e Validação de um Software para Automação de Processos de Gestão Tecnológica no Serviço Público. *Revista Multidisciplinar do Nordeste Mineiro*, *21*(3), 1–29. https://doi.org/10.61164/s4608v39

Este é um artigo publicado em acesso aberto (Open Access) sob a licença CreativeCommons Attribution, que permite uso, distribuição e reprodução em qualquer meio, sem restrições desde que o trabalho original seja corretamente citado.