UNIVERSIDADE FEDERAL DO OESTE DO PARÁ
INSTITUTO DE ENGENHARIA E GEOCIÊNCIAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RAMON BARBOSA PESSOA

PLANEJAMENTO, EXECUÇÃO E ANÁLISE DE TESTES DE FUNCIONALIDADE NO SISTEMA “CONECTA CANAÃ”

SANTARÉM
2024

RAMON BARBOSA PESSOA

PLANEJAMENTO, EXECUÇÃO E ANÁLISE DE TESTES DE FUNCIONALIDADE NO SISTEMA “CONECTA CANAÃ”

Trabalho de Conclusão de Curso apresentado ao Programa de Computação para obtenção de grau de Bacharel em Ciência da Computação, Universidade Federal do Oeste do Pará, Instituto de Engenharia e Geociências.
Orientador Dr. Marcelino Silva da Silva.

SANTARÉM
2024

Dados Internacionais de Catalogação-na-Publicação (CIP)
Sistema Integrado de Bibliotecas – SIBI/UFOPA

P475p Pessoa, Ramon Barbosa

Planejamento, execução e análise de testes de funcionalidade no sistema “Conecta Canaã”. / Ramon Barbosa Pessoa. - Santarém, 2024.

40 p. : il.

Inclui bibliografias.

Orientador: Marcelino Silva da Silva.

Trabalho de Conclusão de Curso (Graduação) – Universidade Federal do Oeste do Pará, Instituto de Engenharia e Geociências, Bacharel em Ciência da Computação.

1. Testes de Software. 2. Qualidade de Software. 3. IEEE 829-2008. 4. Automação de testes. 5. Cidades Inteligentes. 6. Conecta Canaã. I. Silva, Marcelino Silva da, *orient*. II. Título.

CDD: 23 ed. 004.678

Bibliotecária - Documentalista: Cátia Alvarez – CRB/2 843

RAMON BARBOSA PESSOA

PLANEJAMENTO, EXECUÇÃO E ANÁLISE DE TESTES DE FUNCIONALIDADE NO SISTEMA “CONECTA CANAÃ”

Trabalho de Conclusão de Curso apresentado ao Programa de Computação para obtenção de grau de Bacharel em Ciência da Computação, Universidade Federal do Oeste do Pará, Instituto de Engenharia e Geociências.

Conceito: 9,8
Data de Aprovação 16/10/2024

Documento assinado digitalmente
gov.br
MARCELINO SILVA DA SILVA
Data: 21/11/2024 08:38:43-0300
Verifique em https://validar.iti.gov.br

_____
Dr. Marcelino Silva da Silva - Orientador
Universidade do Oeste do Pará

Documento assinado digitalmente
gov.br
SOCORRO VANIA LOURENCO ALVES
Data: 20/11/2024 18:12:25-0300
Verifique em https://validar.iti.gov.br

_____
Profa. Ms. Socorro Vânia Lourenço Alves
Universidade do Oeste do Pará

Documento assinado digitalmente
gov.br
CARLA MARINA COSTA PAXIUBA
Data: 20/11/2024 12:36:24-0300
Verifique em https://validar.iti.gov.br

_____
Profa. Dra. Carla Marina Costa Paxiuba
Universidade do Oeste do Pará

Dedico este trabalho à minha família, pelo apoio incondicional, e aos amigos que estiveram ao meu lado durante toda esta jornada.

AGRADECIMENTO

Agradeço primeiramente à minha família, que sempre esteve ao meu lado e me deu a oportunidade de estudar e me permitir seguir a carreira que desejava. Ao meu orientador, Prof. Dr. Marcelino Silva da Silva, por toda a orientação, paciência e conhecimento compartilhado durante a realização deste projeto. Suas orientações foram fundamentais para o sucesso deste trabalho.

Um agradecimento especial a Prof. Dra. Socorro Vânia Lourenço Alves que sempre me apoiou como docente e esteve em minha banca também para me apoiar, e a todos os demais professores da UFOPA, que contribuíram significativamente para a minha formação acadêmica e que sem eles não conseguiria alcançar o tipo de profissional que me tornei.

À minha namorada, por seu carinho, apoio e compreensão, enquanto eu estava elaborando este trabalho.

À minha amiga Sávia Almeida da Silva, pois sem suas contribuições positivas não seria possível a elaboração deste trabalho. Aos meus amigos Sávio, João, Yure e a todos os demais por estarem presentes nos momentos importantes e por tornarem essa jornada mais leve e significativa.

Por fim, agradeço a todos aqueles que, direta ou indiretamente, colaboraram para que este trabalho fosse realizado. Muito obrigado!

“O propósito da tecnologia é libertar as pessoas para que possam ser mais criativas e produtivas”
(GATES, 1995, p. 45)

RESUMO

Este trabalho aborda a aplicação de testes de funcionalidade no sistema web Conecta Canaã, utilizado pela Prefeitura Municipal de Canaã dos Carajás para gestão de ocorrências reportadas pelos cidadãos, no contexto de cidades inteligentes. O desafio consiste em implementar testes adequados seguindo a norma IEEE 829-2008, que fornece uma abordagem para a documentação de testes de software, garantindo confiabilidade e eficiência na administração pública. Revisamos a importância dos testes de software para a qualidade e identificamos a necessidade de testes adequados em aplicativos de gestão pública. Propomos a implementação de testes automatizados utilizando o framework Selenium, abrangendo diversos casos de teste e perfis de usuário. Os resultados indicam que todos os casos de teste foram aprovados, demonstrando que o Conecta Canaã opera conforme esperado e atende às necessidades operacionais. Concluímos que a adoção de práticas de teste funcional alinhadas à IEEE 829-2008 é fundamental para a qualidade em sistemas públicos, contribuindo para a eficiência administrativa e a interação eficaz entre administração e comunidade.

Palavras-Chave: Testes de Software. Qualidade de Software. IEEE 829-2008. Automação de Testes. Cidades Inteligentes. Conecta Canaã.

ABSTRACT

This paper addresses the application of functionality testing on the Conecta Canaã web system, used by the Municipality of Canaã dos Carajás for managing citizen-reported incidents within the context of smart cities. The challenge lies in implementing adequate tests following the IEEE 829-2008 standard, which provides a framework for software testing documentation, ensuring reliability and efficiency in public administration. We review the importance of software testing for quality assurance and identify the need for proper testing in public management applications. We propose the implementation of automated tests using the Selenium framework, covering various test cases and user profiles. The results indicate that all test cases passed, demonstrating that Conecta Canaã operates as expected and meets operational requirements. We conclude that adopting functional testing practices aligned with IEEE 829-2008 is essential for ensuring quality in public systems, contributing to administrative efficiency and effective interaction between administration and community.
Keywords: Software Testing. Software Quality. IEEE 829-2008. Test Automation. Smart Cities. Conecta Canaã.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
CMM	Capability Maturity Model
IEC	International Electrotechnical Commission
IoT	Internet das Coisas
ISO	International Organization for Standardisation
MPS.BR	Modelo de Processos do Software Brasileiro
NBR	Norma Brasileira
SUT	Sistema em Teste
TAA	Arquitetura de Automação de Testes
TAF	Framework de Automação de Testes
TICs	Tecnologias da Informação e Comunicação

1 INTRODUÇÃO

No âmbito da urbanização, bem como da modernização, a idealização de cidades inteligentes surgiu como uma resposta inovadora e global aos desafios crescentes na eficiência da gestão urbana. Segundo Batty et al. (2012), uma cidade é considerada inteligente quando as Tecnologias da Informação e Comunicação (TICs) são integradas às infraestruturas convencionais, de maneira coordenada e integrada, utilizando novas tecnologias digitais.

Em paralelo a isso, o autor Batty et al. (2012), destaca um crescente interesse no desenvolvimento de soluções tecnológicas voltadas para a urbanização, visando a melhoria significativa da qualidade de vida urbana. Nesse sentido, o sistema web Conecta Canaã, utilizado pela Prefeitura Municipal de Canaã dos Carajás para fornecer atualizações sobre o andamento das ocorrências reportadas, ilustra bem essa tendência.

O sistema Conecta Canaã é assistido pelo projeto de implantação e gestão de uma plataforma de cidades inteligentes baseada em Internet das Coisas (IoT) para o Município de Canaã dos Carajás, que se dedica ao desenvolvimento e à manutenção de soluções tecnológicas que respondam às necessidades da população e promovem inovação no segmento de gestão pública.

Esse projeto faz parte da colaboração entre as universidades UFPA, UFOPA e Unifesspa e a Prefeitura Municipal de Canaã dos Carajás contribui para o desenvolvimento de soluções tecnológicas voltadas para a urbanização. Nesse contexto, a equipe de desenvolvimento, da qual o autor deste trabalho faz parte, é responsável por garantir que o sistema opere de maneira eficaz, considerando os atributos de qualidade de software.

Entretanto, ainda que existam iniciativas em curso para o avanço das cidades inteligentes, surge a importância de entender se há aplicação de testes adequados de avaliação para aplicativos destinados à administração pública, isto é, se o sistema Conecta Canaã está dentro dos parâmetros de qualidade de software da IEEE 829-2008. De acordo com Andrea (2007), a qualidade do desenvolvimento de software é impulsionada por meio de teste de funcionalidades.

Rizardi et al. (2022) aponta, ao examinar a literatura contemporânea sobre administração e gestão pública, para uma convergência na necessidade de inovação no setor público. Nesse cenário, os aplicativos de gerenciamento de ocorrências, como o Conecta Canaã Web, exercem um papel importante. O impulso para o teste de funcionalidades no sistema do Conecta Canaã é sustentado pela necessidade de promover uma interação mais

eficaz entre a administração pública e a comunidade, explorando os instrumentos oferecidos pelas TICs.

Neste trabalho, será abordado a aplicação e a importância dos testes de funcionalidades para a viabilidade e confiabilidade do Conecta Canaã. Será feita uma abordagem de testes de software para assegurar que o sistema atenda as demandas operacionais, reforçando a confiança do cidadão e a eficiência administrativa.

1.1 Objetivos

Nesta seção serão apresentados os objetivos relacionados a testes de software aplicados durante o desenvolvimento do Conecta Canaã, abrangendo tanto os objetivos gerais quanto os específicos.

1.1.1 Objetivo geral

Realizar testes de funcionalidade no sistema Conecta Canaã, assegurando que ele atenda corretamente às necessidades operacionais essenciais da Prefeitura Municipal de Canaã dos Carajás. O objetivo é garantir a qualidade e a confiabilidade do software, confirmando que suas funcionalidades operam conforme os requisitos especificados e que o sistema responde adequadamente às demandas dos usuários.

Além disso, busca-se estabelecer uma abordagem para a aplicação de testes de software em sistemas de gestão pública. Ao aplicar testes alinhados à norma IEEE 829-2008, pretende-se contribuir para o aprimoramento das práticas de teste de funcionalidade em contextos semelhantes. Dessa forma, espera-se que os resultados deste trabalho beneficiem outros projetos de software na área pública, promovendo a eficiência e a confiabilidade dos sistemas utilizados.

1.1.2 Objetivos específicos

1. Planejar e projetar testes de funcionalidade para o sistema Conecta Canaã, seguindo os padrões estabelecidos pela norma IEEE 829-2008.
2. Implementar os testes de funcionalidade utilizando ferramentas de automação adequadas, como o framework Selenium, para garantir a eficiência e a abrangência dos testes.

3. Executar os testes de funcionalidade no sistema Conecta Canaã, abrangendo diferentes perfis de usuário e cenários operacionais relevantes.
4. Analisar os resultados obtidos nos testes, verificando a conformidade do sistema com os requisitos funcionais especificados e identificando possíveis falhas ou áreas de melhoria.
5. Documentar os processos e resultados dos testes realizados, proporcionando um conjunto de práticas e conhecimentos que possam servir de referência para a elaboração de testes em softwares de gestão pública similares.

1.2 Justificativa

"Desde os primeiros computadores comerciais, uma característica predominante nos softwares implantados ou lançados no mercado é a presença de um grande número de defeitos, afetando diretamente a usabilidade, a funcionalidade e a confiabilidade" (RIOS, 2006). Estes defeitos têm um impacto decisivo nos negócios, resultando frequentemente em enormes prejuízos, seja pela perda de participação no mercado ou por danos à imagem dos produtos lançados (RIOS, 2006).

Diante do contexto apresentado, compreende-se que o teste de software é uma parte fundamental para o ciclo de desenvolvimento do sistema, isto é, um software somente é viável caso ele consiga atender seus requisitos funcionais. Além disso, RIOS (2006), afirma que softwares com problemas de performance e com defeitos na execução representam custos significativos.

> Os livros e revistas relatam dezenas de casos de softwares com falhas e os seus impactos, sendo que o relatório The Economic Impact of Inadequate Infrastructure for Software Testing (NIST, May 2002), na época, estimava que o custo total dos softwares com defeitos para as organizações nos EUA correspondia, em termos aproximados, a um valor um pouco abaixo de 1% do Produto Interno Bruto (PIB). (RIOS, 2006, online)

O sistema “Conecta Canaã” representa uma iniciativa em Canaã dos Carajás, buscando estabelecer uma plataforma de comunicação direta entre a administração municipal e os cidadãos. Contudo, em decorrência de possíveis defeitos na execução do sistema Conecta Canaã, é importante avaliar se as funcionalidades do sistema atendem adequadamente à demanda, considerando os parâmetros de qualidade do software.

Ademais, sabe-se que há um crescente interesse em sistemas de informação aplicados à gestão pública, especificamente no contexto de cidades que buscam se tornarem inteligentes. Assim, surgiu o interesse em aprimorar os processos de teste de funcionalidade do “Conecta Canaã” e contribuir não apenas para o sucesso local do “Conecta Canaã”, mas também contribuir para o seu sucesso mais amplo.

1.3 Organização do trabalho

A estrutura deste trabalho está organizada da seguinte forma: a Seção 2 apresenta os fundamentos teóricos e os trabalhos relacionados, abordando conceitos sobre qualidade de software, testes de funcionalidade e normas como a IEEE 829-2008. A Seção 3 descreve a metodologia aplicada no planejamento, projeto, execução e análise dos testes de funcionalidade realizados no sistema Conecta Canaã, detalhando os casos de uso e os procedimentos adotados. Por fim, a Seção 4 traz as considerações finais, destacando a importância dos testes para a garantia da qualidade do software e sugerindo possíveis trabalhos futuros que possam contribuir para o aprimoramento de sistemas de gestão pública.

2 FUNDAMENTOS TEÓRICOS OU TRABALHOS RELACIONADOS

O referencial teórico para esta pesquisa está composto por autores que analisam e discutem questões sobre qualidade de software e teste de software, bem como a influência dessas questões nos aplicativos, afinal, ao destacar engenharia de software, tais questões se revelam como base para o estudo proposto.

Nesse sentido, a contribuição dos autores Pressman (2021) e Sommerville (2011), tornam-se indispensáveis, visto que os autores expressam clareza no entendimento de questões de avaliação funcional de software. Essa compreensão torna-se ainda mais relevante à medida que a sociedade atual, cada vez mais buscando aplicações de IoT, exige eficácia das funcionalidades dos sistemas.

2.1 Qualidade de software

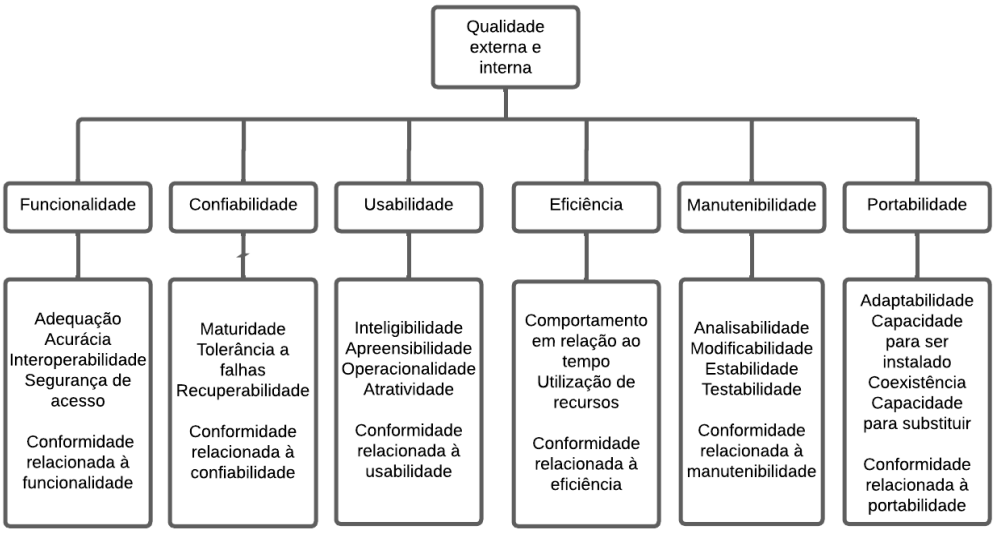
Diante do evidente aumento da demanda por sistemas complexos nas organizações, é perceptível que a qualidade de software torna-se cada vez mais importante durante o desenvolvimento de software. Segundo a definição de Bartié (2002) a qualidade de

software é garantida através de um processo sistemático que cobre todas as etapas e artefatos do desenvolvimento, visando assegurar conformidade e eliminar defeitos.

Para Pressman (2021), a qualidade de software está associada à conformidade com os requisitos funcionais e não-funcionais especificados no processo de desenvolvimento do sistema. Essa compreensão é sustentada pela NBR ISO/IEC 9126, que estabelece um modelo para a avaliação da qualidade tanto externa quanto interna do software, conforme apresentado na Figura 1.

O modelo define seis características principais e suas respectivas subcaracterísticas, tais características podem ser utilizadas para especificar os requisitos de software. (ISO/IEC 9126). Portanto, com base nos conceitos apresentados, os requisitos de software são componentes de qualidade indispensáveis para atender as necessidades dos usuários.

Figura 1 - Modelo de qualidade



Fonte: ISO/IEC 9126 (2003)

O modelo apresentado pela ISO/IEC 9126, destaca seis subcaracterísticas de qualidade de software. A primeira, funcionalidade, refere-se à capacidade e adequação das funcionalidades do software. A segunda, confiabilidade, refere-se à habilidade do software de operar de maneira consistente sob condições específicas. A terceira, usabilidade, refere-se ao quão fácil e agradável é para os usuários interagirem com o sistema.

Além disso, a eficiência refere-se ao desempenho do software, considerando o uso otimizado de recursos, como tempo de resposta e consumo de memória. A manutenibilidade refere-se ao quão fácil o software pode ser corrigido, adaptado e aprimorado ao longo do

tempo. Por fim, a portabilidade refere-se à capacidade do software de ser transferido e operado em diferentes ambientes e plataformas. Assim, este trabalho baseia-se na qualidade de software denominada funcionalidade.

Em sua obra Sommerville (2011) define os requisitos de software em requisitos funcionais e requisitos não-funcionais. O autor esclarece que os requisitos funcionais descrevem o que o sistema deve fazer, como deve reagir a entradas específicas e seu comportamento em certas situações. Em contraste, os requisitos não-funcionais referem-se às restrições gerais ao sistema, como limitações de tempo, processos de desenvolvimento e normas.

De acordo com o autor Machado (2015), essa definição é corroborada, uma vez que ele afirma que os requisitos refletem as características e restrições do produto de software do ponto de vista das necessidades dos usuários e não dependem da tecnologia utilizada para implementar a solução. Diante disso, pode-se afirmar que a qualidade do produto está relacionada à qualidade do processo de desenvolvimento.

Além disso, a importância da qualidade de software é reforçada pelos modelos de maturidade de processos, como o Capability Maturity Model (CMM) e o Modelo de Processos do Software Brasileiro (MPS.BR). No CMM, práticas de teste e validação são formalmente instituídas a partir do Nível 3, onde os processos são definidos e padronizados, incluindo atividades sistemáticas de verificação e validação do software (Paulk et al., 1993). De forma semelhante, no MPS.BR, a área de processo de Verificação e Validação é contemplada a partir do Nível B, enfatizando a necessidade de testes estruturados para assegurar a qualidade do produto (SOFTEX, 2016).

2.2 Testes de software

É indubitável que o teste de software é um processo fundamental para o ciclo de vida do sistema. Este processo pode ser entendido tanto de forma intuitiva, como um processo de verificação do software, quanto técnica. Segundo Myers et al. (2012), o teste de software baseia-se no processo de execução de um software com o objetivo de revelar falhas. Dessa forma, é possível identificar as causas das falhas e implementar as correções necessárias.

Na fase de testes de software, é comum empregar simultaneamente diferentes tipos de testes. Diante disso, os testes são classificados por Molinari (2008) em quatro dimensões, cada uma representando um aspecto específico da avaliação.

A primeira dimensão (Estado do Teste) engloba Testes de Unidade, Testes de Integração e Testes de Sistema, focando na etapa de desenvolvimento em que o teste é feito.

A segunda dimensão (Técnica de Teste), inclui Testes Operacionais, Testes de Regressão, Testes de Caixa-Preta, Testes de Caixa-Branca e Testes de Verificação, destacando as metodologias utilizadas na execução dos testes.

A terceira dimensão (Metas do Teste), engloba os Testes Funcionais, Testes de Interface, Testes de Performance, Testes de Aceitação, Testes de Stress, Testes de Volume, Testes de Configuração, Testes de Integridade, Testes de Manutenção e Testes de Segurança, que definem as áreas de foco dos testes.

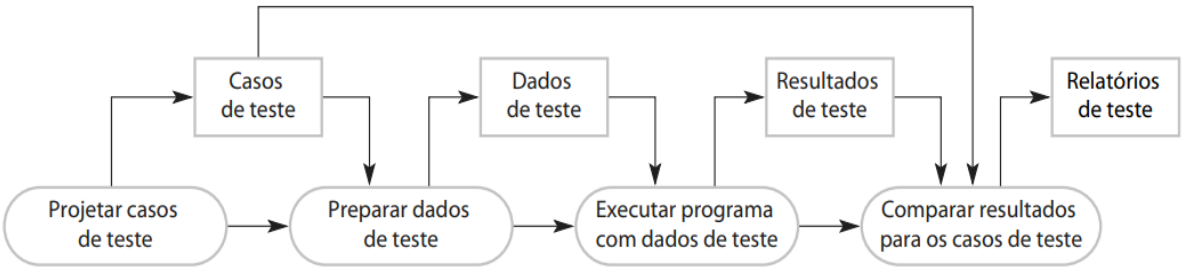
Por fim, a quarta dimensão (Ambiente de Teste), representa o contexto tecnológico específico em que os testes são realizados, incluindo Testes de Aplicações Web, Mainframe, Server, Client e Network. Portanto, diante da significativa quantidade de tipos de teste, é importante selecionar aqueles que são mais adequados para identificar as causas das falhas, considerando os diferentes propósitos e características de cada sistema.

De modo complementar, Sommerville (2011) propõe que o processo de teste visa, em primeiro lugar, evidenciar para o desenvolvedor e o cliente que o software atende a seus requisitos estabelecidos. Em segundo lugar, identificar cenários em que o software possa apresentar comportamentos incorretos, indesejáveis ou diferentes das especificações. Portanto, o teste de software é executado de forma controlada para verificar se seu comportamento está alinhado com o que foi especificado e validar sua qualidade.

Dentro desse contexto de atividades de teste, destaca-se dois conjuntos de atividades fundamentais para formalizar essas atividades: os casos de teste e os procedimentos de teste. Segundo Craig E Jaskiel (2002), um caso de teste especifica uma condição específica a ser verificada, incluindo valores de entrada, restrições para sua execução e o resultado ou comportamento esperado. Os autores definem que o procedimento de teste refere-se à descrição dos passos necessários para executar um caso ou um conjunto de casos de teste.

Conforme apresentado na figura 2, no modelo tradicional de teste de software, descrito por Sommerville (2011), é possível perceber que a definição dos casos de teste é fundamental e constitui a primeira etapa no processo de implementação dos testes. O autor explica que os casos de teste atuam como especificações detalhadas das entradas e das saídas esperadas, além de esclarecer o que está sendo avaliado.

Figura 2 - Modelo do processo de teste de software



Fonte: Sommerville (2011)

2.2.1 Testes funcionais

Durante o processo de testes de software, o teste funcional, de acordo com Myers (2004), é um conjunto de tarefas que visam identificar disparidades entre o comportamento do sistema e suas especificações, focando no comportamento do sistema do ponto de vista do usuário final. O autor afirma ainda que esse conjunto de atividades baseiam-se na análise das especificações para criar casos de teste que avaliam a funcionalidade do sistema.

Ao enfatizar o conceito dos testes funcionais e testes não funcionais, Pressman e Max (2021) afirmam que os testes de funcionalidade devem ser repetidos utilizando os casos de teste criados durante as etapas de construção dos protótipos incrementais. Nesse sentido, a execução desses testes revelam-se essenciais para o processo de teste, com a finalidade de assegurar que cada componente do sistema funciona conforme as especificações e requisitos definidos.

2.2.2 Automação de testes

Os testes automatizados, de acordo com Bernardo e Kon (2008), exercem um papel importante no processo de teste de software ao utilizar programas ou scripts para avaliar funcionalidades do sistema e realizar verificações automáticas dos efeitos colaterais obtidos. Além disso, essa abordagem permite a execução de um grande volume de testes de maneira ágil e com menor esforço, possibilitando a repetição rápida de todos os casos de teste sempre que necessário.

Essa ideia é corroborada por Dustin et al. (2009), os quais descrevem a automação de testes como um processo que utiliza ferramentas de software para criar e realizar testes. Segundo os autores, os testes realizados manualmente, como testes funcionais, desempenho, simultaneidade, estresse e outros, podem ser automatizados. A automação desses testes destacam-se em maior agilidade no processo de teste, gerando redução de custos.

Dentro desse contexto de automação de testes, o International Software Testing Qualifications Board (ISTQB, 2016) identifica vários atributos que devem ser considerados para a eficácia dos testes automatizados. Entre esses atributos, destaca-se a Arquitetura de Automação de Testes (TAA), a qual deve estar alinhada com a arquitetura do software em teste, assegurando que os requisitos do sistema estejam bem definidos.

Além disso, o Sistema em Teste (SUT), deve minimizar a dependência entre a interface gráfica e as interações e dados. Outro aspecto relevante é a estratégia de automação de testes, a qual deve buscar um equilíbrio entre custos, benefícios e riscos. Por fim, a escolha de um Framework de Automação de Testes (TAF) deve priorizar soluções que sejam intuitivas e bem documentadas, contribuindo para a eficácia geral da automação.

De modo complementar, o ISTQB (2016) evidencia que a arquitetura geral de automação de testes é estruturada em quatro camadas principais: geração de testes, definição de testes, execução de testes e adaptação de testes. Diante disso, é possível perceber que todas essas camadas desempenham um papel importante na criação e manutenção de testes automatizados eficazes.

2.2.2.1 Ferramenta de testes automatizados

De acordo com Maxim e Pressman (2016), a seleção da ferramenta de automação de testes deve considerar a compatibilidade com todas as plataformas necessárias, a capacidade de testar diferentes tipos de tela e resoluções, além de suportar variados mecanismos de entrada. Além disso, é importante que a ferramenta ofereça conectividade com sistemas externos para permitir testes completos do sistema.

Dentre as ferramentas de automação de testes disponíveis, destaca-se o framework Selenium, o qual abrange para uma variedade de conjunto de ferramentas e bibliotecas que possibilitam a automação de testes em diferentes navegadores e plataformas. A utilização do Selenium é ilustrada na Figura 3, que apresenta um exemplo de código demonstrando a aplicação do Selenium WebDriver para automatizar o processo de login em uma página Conecta Canaã.

Figura 3 - Exemplo de teste com Selenium

```python
def login(browser: Browser, email, password) -> bool:

    # Abre a página desejada
    browser.driver.get(f"{os.getenv('HOME_PAGE')}")

    # Encontra o campo de e-mail e preenche com o e-mail fornecido
    email_field = browser.driver.find_element(By.XPATH, "//div[contains(@class, 'MuiFormControl-root')][1]//input")
    email_field.clear()
    email_field.send_keys(f"{email}")

    # Encontra o campo de senha e preenche com a senha fornecida
    password_field = browser.driver.find_element(By.XPATH, "//div[contains(@class, 'MuiFormControl-root')][2]//input")
    password_field.clear()
    password_field.send_keys(f"{password}")

    # Encontra o botão de login e clica nele
    login_button = browser.driver.find_element(By.XPATH, "//button[@type='submit']")
    login_button.click()

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

Fonte: Autor (2024)

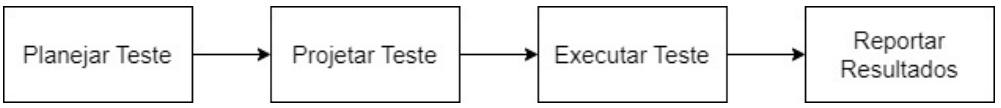
No código a função login abre a URL fornecida, localiza e preenche os campos de e-mail e senha, e submete o formulário clicando no botão de login. Após o envio, a função aguarda a presença de um elemento específico que indica um login bem-sucedido. Se o elemento é encontrado, a função retorna True, caso contrário, retorna False.

2.3 Padrão IEEE 829-2008

No que se refere à desenvolver os testes de software, Lino (2021) ressalta a importância de empregar um padrão de processos de testes. Nesse contexto, o fluxo parcial do processo padrão do IEEE 829-2008, apresentado na figura 4, destaca-se como um referência, considerando que este fluxo dispõe de etapas necessárias para realizar a documentação e execução dos testes.

A adoção da norma IEEE 829-2008 é fundamental, pois ela fornece um conjunto padronizado de diretrizes para a documentação e execução de testes de software. Essa padronização contribui para a consistência e a qualidade do processo de teste, assegurando que todas as etapas sejam conduzidas de maneira sistemática e eficaz. Conforme destaca IEEE (2008), o uso dessa norma facilita a comunicação entre os membros da equipe e demais partes interessadas, além de promover a rastreabilidade e a transparência nos resultados obtidos durante os testes.

Figura 4 - Processos de testes



Fonte: Adaptado de IEEE (2008).

De acordo com o fluxo apresentado na figura 4, observa-se que o processo de teste de software, abordando os principais pontos, conforme definido pela norma IEEE 829-2008, inclui quatro etapas: "Planejar Teste", "Projetar Teste", "Executar Teste" e "Reportar Resultados". Segundo Sommerville (2011), o processo de desenvolvimento de software inclui tarefas técnicas, colaboração e gerenciamento para criar um software funcional e de qualidade.

Evidencia-se, portanto, que ao seguir essas etapas, o processo torna-se mais ágil e adaptável às especificidades do sistema, mantendo um padrão de qualidade reconhecido. O desenvolvimento de testes, estruturado em torno dessas quatro etapas, garante que o software atenda de maneira adequada e consistente às suas exigências. A seguir, detalha-se como cada uma dessas etapas contribui para o processo, conforme a norma IEEE (2008):

Na etapa de plano de teste, descreve-se vários componentes do sistema, incluindo a definição do objetivo do teste, a abordagem adotada e a identificação dos itens a serem testados. Além disso, especifica-se os recursos e as tarefas envolvidas, assim como as responsabilidades de cada tarefa e os potenciais riscos associados. Dessa forma, é estabelecida uma visão geral do que será realizado durante o processo de teste.

Na etapa de projeto de teste, documenta-se, detalhadamente, as especificações e os procedimentos de teste, isto é, define-se as condições de teste e a criação de casos de teste. Para Craig E Jaskiel (2002), o caso de teste descreve um conjunto de valores de entrada, condições para a sua execução, e um resultado ou comportamento esperado. Além disso, elabora-se os dados e os ambientes necessários para a execução dos testes.

Na etapa de execução de teste, realiza-se a execução prática dos casos de teste, conforme especificado na etapa anterior. Durante esta etapa, registra-se todos os resultados para permitir a comparação dos resultados obtidos com os resultados esperados. Sabendo disso, a atenção aos detalhes e a aderência aos procedimentos definidos são fundamentais para assegurar a validade e a confiabilidade dos resultados obtidos, uma vez que qualquer alteração em relação ao resultado esperado é registrada e classificada como um defeito.

Por fim, na etapa de análise de resultados, os dados coletados durante a execução dos testes são cuidadosamente analisados. Esta análise busca identificar áreas de melhoria,

avaliar a eficácia do sistema e determinar se os objetivos iniciais foram atingidos. Portanto, através desta abordagem estruturada, o projeto visa assegurar que a aplicação atenda às expectativas e requisitos estabelecidos pelos clientes.

3 DESENVOLVIMENTO

Nesta seção são expostos os materiais e métodos utilizados na condução deste trabalho, cobrindo deste a análise de requisitos até a interpretação dos resultados. Além disso, destaca-se que este trabalho possui natureza quantitativa, descritiva e experimental, e está estruturado em três etapas: plano, projeto, execução e análise de resultados. A fim de elucidar a metodologia adotada, realiza-se testes de funcionalidades no âmbito de um projeto específico, utilizando uma versão simplificada do padrão IEEE 829-2008.

3.1 Plano

O planejamento dos testes de funcionalidade do sistema web Conecta Canaã abrange três fases principais: a descrição do sistema, a análise dos requisitos do sistema e a elaboração dos casos de teste. Na descrição é contextualizado as funcionalidades e os objetivos do sistema. Em seguida, a análise dos requisitos identifica as necessidades e especificações do sistema. Por sim, a elaboração de todos os casos de teste relevantes do sistema. Estes processos são detalhados a seguir.

3.1.1 Processo de desenvolvimento do Conecta Canaã

O Conecta Canaã integra o projeto Smart City, cujo objetivo é posicionar Canaã dos Carajás entre as pioneiras em cidades inteligentes no Brasil. Com este propósito, o Conecta Canaã Web funciona como um ambiente de gestão pública utilizado pela Prefeitura Municipal de Canaã dos Carajás, permitindo atualizações sobre o andamento das ocorrências reportadas pelos cidadãos da cidade por meio do aplicativo móvel.

O Conecta Canaã Web tem como objetivo, gerenciar as solicitações de serviços de manutenção urbana, relatos de infrações e questões vinculadas ao Código de Posturas e à Ouvidoria. Este sistema distingue suas funcionalidades conforme o perfil do usuário conectado. Existem três perfis de usuário disponíveis: Executor, Controlador e Administrador.

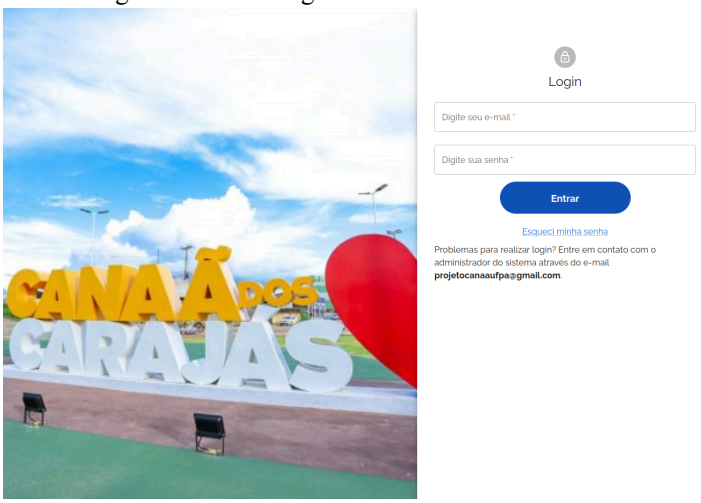
O perfil de Executor é atribuído aos usuários responsáveis por resolver um ou mais tipos específicos de ocorrências reportadas pelos cidadãos.

Os usuários com perfil Controlador têm a função de designar um Executor apropriado para cada nova ocorrência registrada. Além disso, este perfil permite interagir com os cidadãos para esclarecer dúvidas sobre as ocorrências e modificar o tipo e o estado das mesmas conforme necessário. Ou seja, esse perfil atua como um intermediário entre os cidadãos e os Executores.

O perfil de Administrador oferece acesso a algumas das funcionalidades disponíveis para os perfis de Executor e Controlador. Administradores são responsáveis pela elaboração de relatórios detalhados baseados nas ocorrências registradas e pelo cadastro de novos usuários no sistema. Isso assegura uma gestão abrangente e eficiente das operações municipais.

O Conecta Canaã foi desenvolvido na linguagem JavaScript e Typescript, utilizando a biblioteca React para desenvolver as telas do sistema. Hospedado na infraestrutura de nuvem da cidade de Canaã dos Carajás. Na figura 5, encontra-se a tela inicial do Conecta Canaã Web. Esta primeira tela apresenta o formulário de login, onde o usuário informa o seu e-mail e sua senha para acessar o sistema.

Figura 5. Tela de login do sistema Conecta Canaã



Fonte: Autor (2024)

O Conecta Canã, incluindo os códigos de bibliotecas, possui mais de 85 mil arquivos, totalizando cerca de 461 MB. Após a etapa de construção do sistema, que consiste em centralizar os códigos, remover bibliotecas não utilizadas, além de outras otimizações. É gerado quatro arquivos: um arquivo HTML, que possui um modelo padrão; dois arquivos

JavaScript que serão responsáveis por controlar os conteúdos que serão apresentados em tela; um arquivo CSS que armazenará as classes de estilização da página.

3.1.2 Análise de requisitos

A segunda fase do plano de teste tem como ponto de partida a análise dos requisitos funcionais do sistema Conecta Canaã Web, por meio da análise das necessidades e expectativas dos usuários. Para facilitar a análise e o acompanhamento dos requisitos, o Quadro 1 mostra a listagem dos requisitos levantados, que estão organizados em quatro colunas:

(1) informa o perfil do utilizador que deve ser capaz de realizar aquela tarefa;

(2) um identificador único que referencia o requisito funcional;

(3) um título para o requisito;

(4) descrição detalhada do requisito.

É importante ressaltar que, devido à elevada quantidade de requisitos funcionais presentes no sistema, fez-se necessário realizar uma seleção dos requisitos para a elaboração dos casos de testes e dos seus respectivos cenários. Para esta seleção, estabeleceu-se alguns critérios, incluindo: (1) frequência de uso da funcionalidade, (2) relevância da funcionalidade para o sistema, (3) complexidade do requisito funcional.

Quadro 1 - Lista de requisitos funcionais

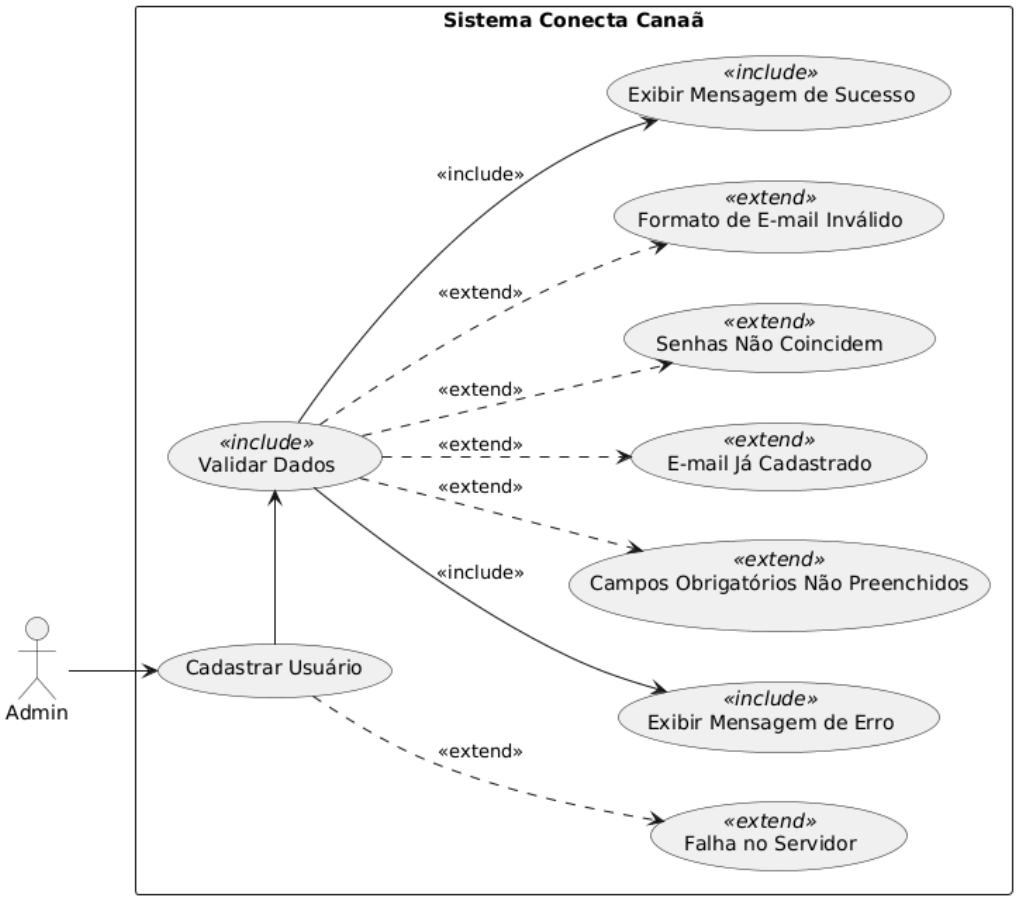
Perfil	ID RF.	Requisito Funcional	Descrição
Controlador	RF 01	Adicionar Comentário	O sistema deve permitir adicionar comentários a uma ocorrência existente.
Executor	RF 02	Visualizar Ocorrência	O sistema deve permitir visualizar os detalhes completos de uma ocorrência.
	RF 03	Visualizar Comentário	O sistema deve permitir visualizar comentários adicionados a uma ocorrência.
Administrador	RF 04	Realizar Login	O sistema deve permitir acessar o sistema por meio de credenciais válidas.
	RF 05	Cadastrar Usuário	O sistema deve permitir cadastrar um novo usuário administrador no sistema.
	RF 06	Aplicar Filtros de Busca	O sistema deve listar ocorrências de acordo com o filtro de busca.
	RF 07	Visualizar Ocorrências	O sistema deve permitir visualizar a lista de ocorrências registradas por status.

Fonte: Autor (2024)

3.1.2 Casos de uso

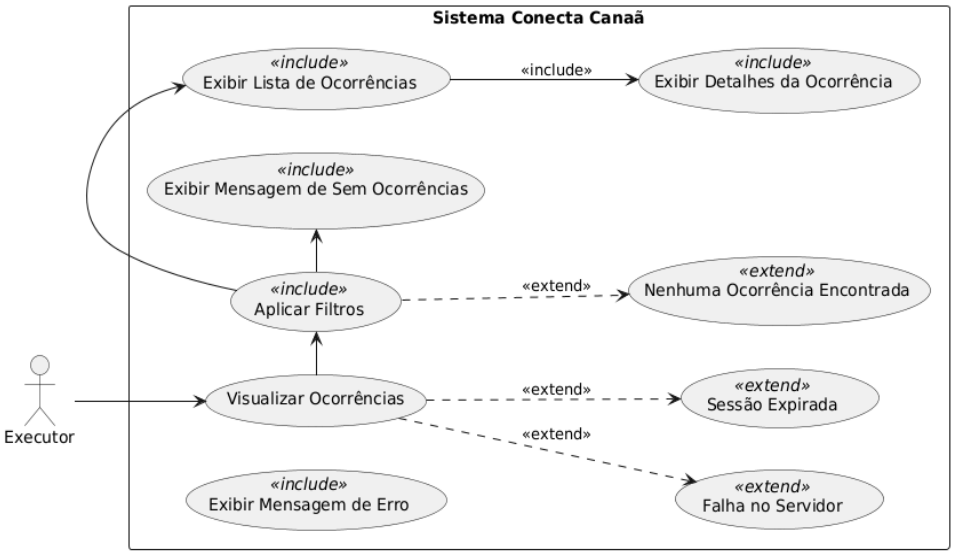
Os casos de uso são essenciais na elaboração dos casos de teste, pois fornecem uma compreensão clara das funcionalidades do sistema do ponto de vista do usuário, permitindo identificar cenários relevantes para a validação do software. De acordo com Sommerville (2011), a utilização de casos de uso facilita a derivação de casos de teste eficazes, assegurando que todos os requisitos funcionais sejam contemplados durante o processo de teste. A seguir, são apresentados os casos de uso desenvolvidos para o sistema Conecta Canaã.

Figura 6 - Caso de uso de cadastro de usuário



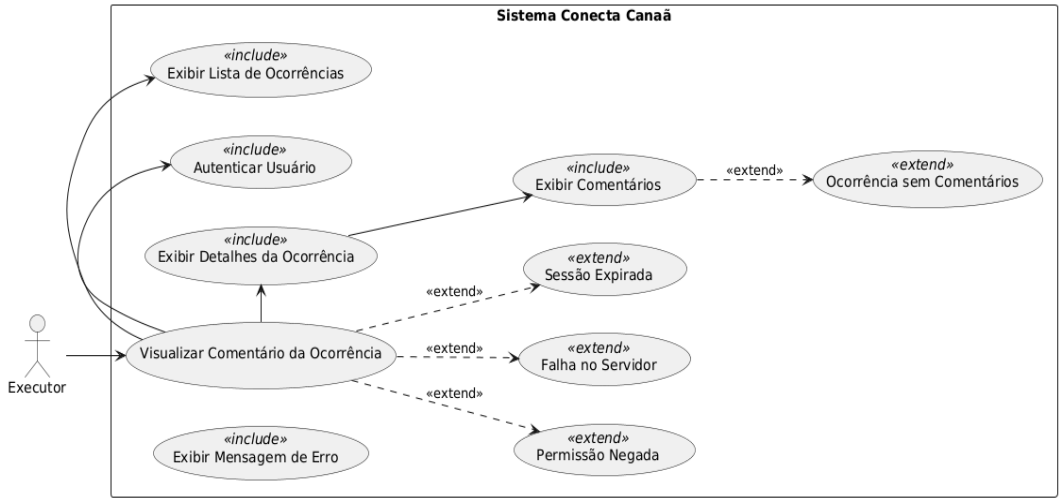
Fonte: Autor (2024)

Figura 7 - Caso de uso de listagem de ocorrências



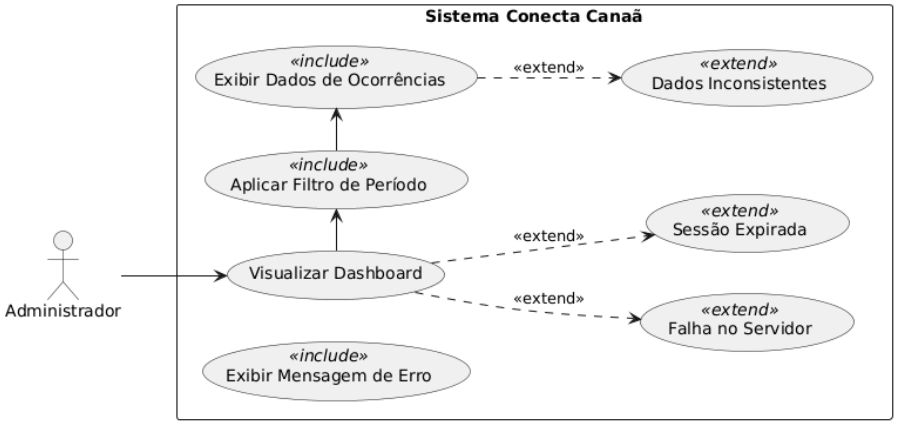
Fonte: Autor (2024)

Figura 8 - Caso de uso de visualizar comentário



Fonte: Autor (2024)

Figura 9 - Caso de uso de dados do dashboard



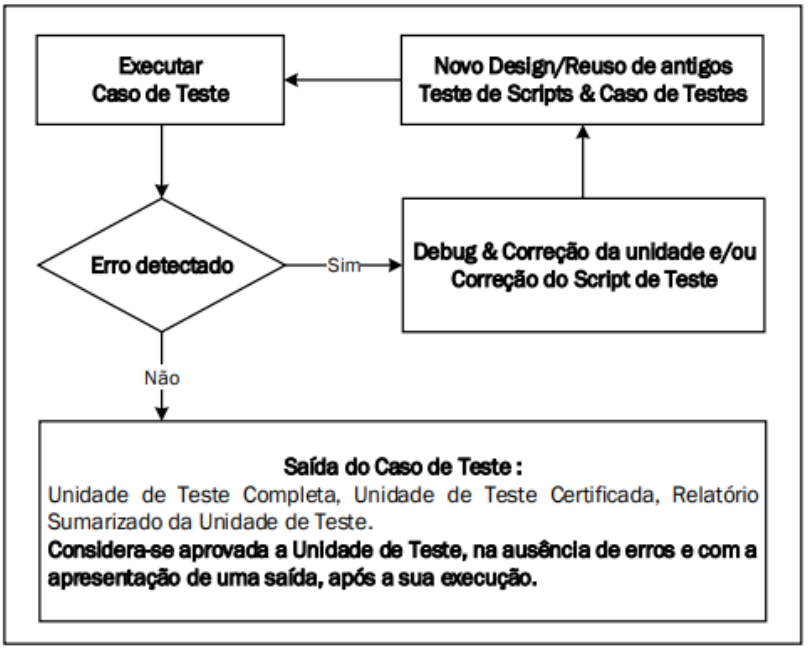
Fonte: Autor (2024)

3.1.2 Casos de teste

Após a conclusão da análise dos requisitos funcionais, a terceira fase do desenvolvimento foi iniciada com a elaboração dos casos de testes. O sistema Conecta Canaã Web emprega diversas bibliotecas que facilitam e aceleram o processo de desenvolvimento. Essas bibliotecas já possuem casos de teste específicos realizados pelos desenvolvedores responsáveis, o que dispensa a necessidade de testá-las individualmente.

No cenário de testes de funcionalidade, o sistema conduz resultados de acordo com os cenários predefinidos. Se o resultado for conciso, o teste é considerado finalizado e será revisado apenas em caso de alterações nas funcionalidades. Caso contrário, um relatório é gerado e encaminhado ao setor de desenvolvimento, identificando o cenário de teste e propondo ajustes no script ou na funcionalidade correspondente, como é observado no fluxo abaixo:

Figura 10 - Fluxo de execução dos casos de testes



Fonte: Lino (2021)

Deste modo, os requisitos funcionais selecionados serão submetidos a testes, sendo que cada caso de teste pode abranger um ou mais desses requisitos. No quadro 2 abaixo é apresentado o mapeamento dos casos de testes, estes são divididos em quatro colunas: (1) faz referência ao requisito funcional apresentado no quadro 1, (2) identificador único do teste de caso, (3) título do teste de caso.

3.1.2.1 Casos de testes do perfil controlador

Quadro 2 - Casos de testes do perfil executor.

Identificação RF	Identificação TC	Caso de Teste
RF 01	TC 01	Adicionar comentário a uma ocorrência.

Fonte: Autor (2024)

3.1.2.2 Casos de testes do perfil executor

Quadro 3 - Casos de testes do perfil controlador.

Identificação RF	Identificação TC	Caso de Teste
RF 02	TC 02	Visualizar ocorrência adicionada pelo cidadão.
RF 03	TC 03	Visualizar comentário da ocorrência pelo controlador.

Fonte: Autor (2024)

3.1.2.3 Casos de testes do perfil administrador

Quadro 4 - Casos de testes do perfil administrador.

Identificação RF	Identificação TC	Caso de Teste
RF 04	TC 04	Realizar login com credenciais válidas.
	TC 05	Tentativa de realizar login com credenciais inválidas.
RF 05	TC 06	Realizar cadastro de um novo usuário administrador no sistema.
	TC 07	Tentativa de realizar cadastro de um usuário já existente no sistema.
RF 06	TC 08	Aplicar filtro de busca em lista de ocorrências.
	TC 09	Tentativa de aplicar filtro de busca inexistente em lista de ocorrências.
RF 07	TC 10	Visualizar a quantidade de ocorrências com status em aberto
	TC 11	Visualizar a quantidade de ocorrências com status concluídas
	TC 12	Visualizar lista de ocorrências com status sem localização

Fonte: Autor (2024)

3.2 Projeto

Com base no planejamento dos casos de teste, optou-se por implementar testes de funcionalidade automatizados para o sistema Conecta Canaã Web. Essa escolha decorre da eficiência que os testes automatizados proporcionam na avaliação da execução dos testes de funcionalidade, dentro do contexto específico do sistema. Para realizar a execução dos testes, selecionou-se o framework Selenium.

A implementação do Selenium no sistema Conecta Canaã Web tem como objetivo simular as ações dos usuários ao executar os casos de testes escolhidos. Com isso, assegura-se a identificação de falhas ou a demonstração que as funcionalidades operem de acordo com o esperado. Essa abordagem automatizada não apenas torna o processo de teste mais ágil, mas também reduz a possibilidade de erros humanos.

A arquitetura do Conecta Canaã Web adota um modelo cliente-servidor, que favorece a manutenção e a escalabilidade do sistema. Essa estrutura permite a integração de novas funcionalidades sem comprometer o desempenho. A arquitetura é composta por três camadas distintas: Apresentação, Lógica e Dados. Na camada de apresentação, os usuários, incluindo executores, controladores e administradores, interagem com a aplicação via internet através do navegador Google Chrome.

Ademais, a camada lógica é estruturada em torno de um servidor de aplicação que opera com o sistema operacional Ubuntu Server e o serviço web NGINX, garantindo o funcionamento do sistema, que foi desenvolvido nas linguagens C e Perl, versão 1.26. Por fim, na camada de dados, um servidor de banco de dados armazena as informações utilizando o SGBD Mongodb na versão 7.0.

Seguindo os critérios definidos na etapa de planejamento, elaborou-se casos de teste específicos para cada requisito funcional, considerando os diferentes perfis de usuário do sistema. Diante disso, para os perfis Executor e Controlador, elaborou-se três casos de teste, enquanto para o perfil Administrador, elaborou-se doze casos de teste. A elaboração dos cenários de teste utilizou o formulário padrão descrito na pesquisa de Lino (2021), o qual se destaca por sua abordagem sucinta e eficiente na definição dos cenários de teste.

Quadro 5 - Formulário do caso de teste

Código:	Número único que identifica cada caso de teste.
Caso de Teste:	Título do teste.
Localização:	Caminho onde se localiza o objeto de teste.
Descrição:	Descrição sucinta do propósito do caso de teste.
Pré-Condição:	Uma ou mais condições que têm de se verificar para que o caso de

	teste possa começar.
Procedimento:	Listagem de todos os passos executados durante o teste.
Resultado Esperado:	Indica qual a saída (output) do sistema após a execução do teste.
Data de Teste:	Indica a data da execução do teste.
Resultado:	Indica o resultado do teste (passou ou falhou) tendo em conta o comportamento do sistema após a execução do teste.

Fonte: Lino (2021)

É importante ressaltar que para cada caso de teste será desenvolvido um script que terá o objetivo de validar a funcionalidade. O sucesso do cenário de teste está condicionado à conformidade do resultado apresentado pelo sistema com as expectativas estabelecidas. No entanto, caso haja qualquer falha em relação ao resultado esperado, o teste será categorizado como não aprovado.

Para a condução dos testes, criou-se três perfis de usuários: Administrador, Controlador e Executor. Essa estratégia permitiu a realização de testes em diferentes cenários operacionais, refletindo em uma variedade de funcionalidades e permissões no sistema. De modo complementar, a extração de dados da base de dados fez-se possível por meio do ambiente de homologação com objetivo de selecionar uma amostra desse ambiente.

Esta amostra, composta por 30 ocorrências, possibilitou a condução do processo de validação, o qual envolveu a comparação entre os dados exibidas no dashboard e os dados reais contidas no banco de dados. Diante disso, pode-se afirmar que os testes foram conduzidos em um contexto que representa de forma semelhante ao funcionamento do Conecta Canaã WEB.

3.3 Execução

Nesta fase de execução de teste, como ponto de partida, iniciou-se a elaboração de scripts de automação específico para cada caso de teste, utilizando a linguagem de programação Python e o framework Selenium. Esta abordagem possibilitou a realização de testes de funcionalidade, conforme delineado no plano previamente estabelecido, com a finalidade de garantir uma ampla análise dos aspectos funcionais do sistema.

A partir do código de teste elaborado pelo autor e hospedado na plataforma GitHub, foram realizados os seguintes testes de software que validam as funcionalidades principais do sistema, incluindo a automação do processo de login e a verificação da presença de elementos indicativos de sucesso.

Quadro 6 - Caso de teste 01 - Adicionar comentário a uma ocorrência.

Código	TC 01
Caso de Teste	Adicionar comentário a uma ocorrência.
Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/
Descrição	Permite que um usuário do tipo controlador adicione comentário a uma ocorrência.
Pré-condição	Estar na tela de login
Procedimento	1. Realizar login com credenciais de controlador. 2. Clicar no botão Ver Detalhes, após selecionar uma ocorrência na lista. 3. Preencher o campo “Digite o parecer técnico da ocorrência” 4. Clicar no botão Realizar Parecer 5. Confirmar envio de parecer
Resultado esperado	Parecer cadastrado com sucesso.
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

Quadro 7 - Caso de teste 02 - Visualizar ocorrência adicionada pelo controlador.

Código	TC 02
Caso de Teste	Visualizar ocorrência adicionada pelo cidadão.
Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/dashboard
Descrição	Permite que um usuário do tipo executor visualizar as ocorrência direcionadas por controladores.
Pré-condição	Estar na tela de login
Procedimento	1. Realizar login com credenciais de executor.
Resultado esperado	Visualizar apenas as listas de ocorrências direcionadas ao executor específico que acessou o sistema.
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

Quadro 8 - Caso de teste 03 - Visualizar comentário da ocorrência.

Código	TC 03
Caso de Teste	Visualizar comentário da ocorrência.
Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/dashboard
Descrição	Permite que um usuário do tipo executor visualize os comentários adicionados pelo controlador na ocorrência.
Pré-condição	Estar na tela de login
Procedimento	1. Realizar login com credenciais de executor. 2. Clicar no botão Ver Detalhes, após selecionar uma ocorrência na lista.
Resultado esperado	Visualizar comentário adicionado por controlador.
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

Quadro 9 - Caso de teste 05 - Realizar login com credenciais válidas.

Código	TC 04
Caso de Teste	Realizar login com credenciais válidas.
Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/
Descrição	Permite que um usuário com credenciais válidas acesse o sistema.
Pré-condição	Estar na página de login
Procedimento	1. Preencher o campo “Digite seu e-mail” 2. Preencher o campo “Digite sua senha” 3. Clicar no botão Entrar
Resultado esperado	Acessar a página de “Dashboard”
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

Quadro 10 - Caso de teste 06 - Realizar login com credenciais inválidas.

Código	TC 05
Caso de Teste	Tentativa de realizar login com credenciais inválidas.
Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/
Descrição	Não permitir que um usuário com credenciais inválidas acesse o sistema.
Pré-condição	Estar na página de login.
Procedimento	1. Preencher o campo “Digite seu e-mail” 2. Preencher o campo “Digite sua senha” 3. Clicar no botão Entrar
Resultado esperado	Apresentar uma mensagem “Houve um problema com o login, verifique suas credenciais!”
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

Quadro 11 - Caso de teste 07 - Cadastro de usuário.

Código	TC 06
Caso de Teste	Realizar cadastro de um novo usuário administrador no sistema.
Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/dashboard
Descrição	Permite que um usuário com perfil de administrador cadastre um novo administrador no sistema.
Pré-condição	Estar na tela de login
Procedimento	1. Realizar login com credenciais de administrador. 2. Clicar em Cadastros > Cadastro de Usuários 3. Preencher o campo “Nome completo” 4. Preencher o campo “E-mail” 5. Preencher o campo “Telefone” 6. Preencher o campo “Senha” 7. Preencher o campo “Confirmação de Senha” 8. Selecionar o tipo de usuário como Administrador 9. Clicar no botão Cadastrar Usuário 10. Confirmar cadastro

Resultado esperado	Apresentar uma mensagem “Usuário cadastrado com sucesso.”
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

Quadro 12 - Caso de teste 08 - Cadastro de usuário.

Código	TC 07
Caso de Teste	Tentativa de realizar cadastro de um usuário já existente no sistema.
Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/dshboard
Descrição	Não permite realizar o cadastro de um usuário existente no sistema.
Pré-condição	Estar na tela de login
Procedimento	1. Executar o script de teste 2. Realizar login com credenciais de administrador. 3. Clicar em Cadastros > Cadastro de Usuários 4. Preencher o campo “Nome completo” 5. Preencher o campo “E-mail” 6. Preencher o campo “Telefone” 7. Preencher o campo “Senha” 8. Preencher o campo “Confirmação de Senha” 9. Selecionar o tipo de usuário como Administrador 10. Clicar no botão Cadastrar Usuário 11. Confirmar cadastro
Resultado esperado	Apresentar uma mensagem de Erro “Um usuário com o e-mail “usuario@e-mail.com” já foi cadastrado no sistema. Por favor, insira um novo e-mail.”
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

Quadro 13 - Caso de teste 09 - Aplicar filtro de busca.

Código	TC 08
Caso de Teste	Aplicar filtro de busca em lista de ocorrências.
Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/dshboard
Descrição	Permite filtrar a lista de ocorrências usando o código da ocorrência.
Pré-condição	Estar na tela de login
Procedimento	1. Executar o script de teste 2. Realizar login com credenciais de administrador. 3. Preencher o campo “Pesquise uma ocorrência” 4. Clicar no botão de pesquisa
Resultado esperado	Apresentar a ocorrência correspondente ao código pesquisado.
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

Quadro 14 - Caso de teste 10 - Aplicar filtro de busca.

Código	TC 09
Caso de Teste	Tentativa de aplicar filtro de busca inexistente em lista de ocorrências.

Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/dshboard
Descrição	Não permite filtrar a lista de ocorrências usando o código inexistente da ocorrência.
Pré-condição	Estar na tela de login
Procedimento	1. Executar o script de teste 2. Realizar login com credenciais de administrador. 3. Preencher o campo “Pesquise uma ocorrência” 4. Clicar no botão de pesquisa
Resultado esperado	Não apresentar nenhuma ocorrência.
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

Quadro 15 - Caso de teste 11 - Visualizar a qualidade de ocorrências com status em aberto.

Código	TC 10
Caso de Teste	Visualizar a qualidade de ocorrências com status em aberto.
Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/dshboard
Descrição	Permite que um usuário com perfil de administrador visualize a qualidade de ocorrências em aberto.
Pré-condição	Estar na tela de login
Procedimento	1. Executar o script de teste 2. Realizar login com credenciais de administrador. 3. Clicar em Dashboard > Dashboard 1 4. Aplicar o filtro de período: 01/01/2021 a 16/08/2024. 5. Consultar os dados.
Resultado esperado	Apresentar o número de ocorrências abertas igual a 20.
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

Quadro 16 - Caso de teste 12 - Visualizar a qualidade de ocorrências com status concluídas.

Código	TC 11
Caso de Teste	Visualizar a qualidade de ocorrências com status concluídas.
Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/dshboard
Descrição	Permite que um usuário com perfil de administrador visualize a qualidade de ocorrências concluídas.
Pré-condição	Estar na tela de login
Procedimento	1. Executar o script de teste 2. Realizar login com credenciais de administrador. 3. Clicar em Dashboard > Dashboard 1 4. Aplicar o filtro de período: 01/01/2021 a 16/08/2024. 5. Consultar os dados.
Resultado esperado	Apresentar o número de ocorrências concluídas igual a 10
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

Quadro 17 - Caso de teste 13 - Visualizar lista de ocorrências com status sem localização.

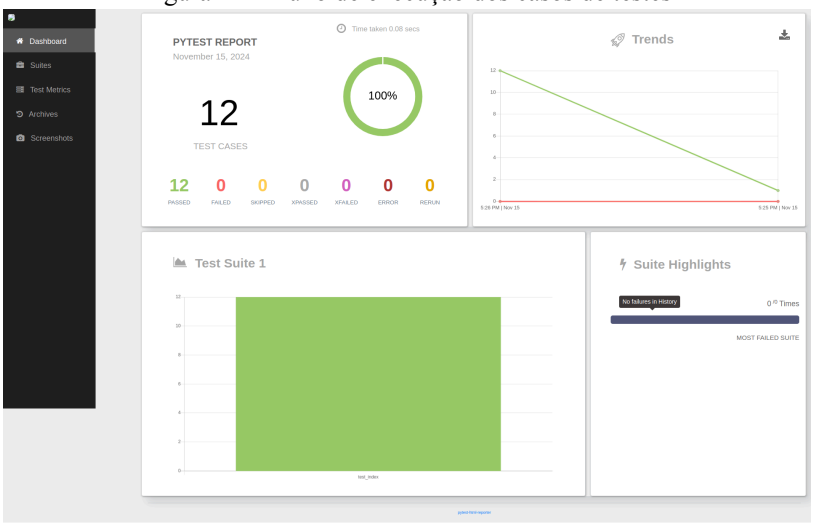
Código	TC 12
Caso de Teste	Visualizar lista de ocorrências com status sem localização.
Localização	https://conecta-hom.smartcitycanaadoscarajas.com.br/dshboard
Descrição	Permite que um usuário com perfil de administrador visualize a qualidade de ocorrências sem localização.
Pré-condição	Estar na tela de login
Procedimento	1. Executar o script de teste 2. Realizar login com credenciais de administrador. 3. Clicar em Ocorrências > Ocorrências sem localização
Resultado esperado	Apresentar lista de ocorrências com status sem localização.
Data de Teste	16/08/2024
Resultado	Ok

Fonte: Autor (2024)

3.4 Análise de resultados

Na última fase do processo de teste, conforme definido pela IEEE 829-2008, é realizada a análise dos resultados obtidos, a fim de validar a integridade do sistema testado. No contexto do sistema Conecta Canaã Web, foram conduzidos diversos casos de teste para avaliar o funcionamento do software sob diferentes perfis de usuário. Esta seção apresenta a análise detalhada dos resultados para cada um dos casos de teste executados.

Figura 11 - Fluxo de execução dos casos de testes



Fonte: Autor (2024)

Dentre os casos testados, destacam-se:

- Caso de Teste 01 - Adicionar comentário a uma ocorrência (TC 01): A execução do TC 01 mostrou que um usuário do perfil controlador pode adicionar um comentário a uma ocorrência com êxito. Este resultado demonstra que o sistema atende ao requisito RF 01. Portanto, o teste TC 01 foi aprovado.
- Caso de Teste 02 - Visualizar ocorrência adicionada pelo cidadão (TC 02): A execução do TC 02 mostrou que um executor pode visualizar as ocorrências reportadas pelo cidadão. Este resultado demonstra que o sistema atende ao requisito RF 02. Portanto, o teste TC 02 foi aprovado.
- Caso de Teste 03 - Visualizar comentário da ocorrência (TC 03): A execução do TC 03 mostrou que um executor pode visualizar os comentários adicionados por um controlador na ocorrência. Este resultado demonstra que o sistema atende ao requisito RF 03. Portanto, o teste TC 03 foi aprovado.
- Caso de Teste 04 - Realizar login com credenciais válidas (TC 04): A execução do TC 04 mostrou que o sistema permite o acesso de um usuário com credenciais válidas com êxito. Este resultado demonstra que o sistema atende ao requisito RF 04. Portanto, o teste TC 04 foi aprovado.
- Caso de Teste 05 - Realizar login com credenciais inválidas (TC 05): A execução do TC 05 mostrou que o sistema não permite o acesso de um usuário com credenciais inválidas. Este resultado demonstra que o sistema atende ao requisito RF 04. Portanto, o teste TC 05 foi aprovado.
- Caso de Teste 06 - Cadastro de usuário administrador (TC 06): A execução do TC 06 mostrou que o sistema realiza o cadastro de um novo usuário administrador com êxito. Este resultado demonstra que o sistema atende ao requisito RF 05. Portanto, o teste TC 06 foi aprovado.
- Caso de Teste 07 - Tentativa de cadastro de usuário já existente (TC 07): A execução do TC 07 mostrou que o sistema não realiza o cadastro de um usuário com um e-mail já existente. Este resultado demonstra que o sistema atende ao requisito RF 05. Portanto, o teste TC 07 foi aprovado.
- Caso de Teste 08 - Aplicar filtro de busca (TC 08): A execução do TC 08 mostrou que o filtro de busca na lista de ocorrências funciona corretamente especificando a ocorrência com seu respectivo código. Este resultado demonstra que o sistema atende ao requisito RF 06. Portanto, o teste TC 08 foi aprovado.

- Caso de Teste 09 - Tentativa de aplicar filtro de busca inexistente (TC 09): A execução do TC 09 mostrou o comportamento do sistema ao aplicar um filtro de busca com um código inexistente. O sistema não retornou nenhuma ocorrência, de acordo com o esperado. Este resultado demonstra que o sistema atende ao requisito RF 06. Portanto, o teste TC 09 foi aprovado.
- Caso de Teste 10 - Visualizar qualidade de ocorrências com status em aberto (TC 10): A execução do TC 10 mostrou que o sistema exibiu a quantidade de ocorrências abertas com êxito. O número apresentado correspondeu ao esperado. Este resultado demonstra que o sistema atende ao requisito RF 07. Portanto, o teste TC 10 foi aprovado.
- Caso de Teste 11 - Visualizar qualidade de ocorrências com status concluídas (TC 11): A execução do TC 11 mostrou que o sistema exibiu a quantidade de ocorrências concluídas com êxito. O número apresentado correspondeu ao esperado. Este resultado demonstra que o sistema atende ao requisito RF 07. Portanto, o teste TC 11 foi aprovado.
- Caso de Teste 12 - Visualizar lista de ocorrências com status sem localização (TC 12): A execução do TC 12 que o sistema listou todas as ocorrências que ainda não possuíam localização atribuída. O resultado foi conforme o esperado. Este resultado demonstra que o sistema atende ao requisito RF 07. Portanto, o teste TC 12 foi aprovado.

4 CONSIDERAÇÕES FINAIS

No contexto atual das inovações tecnológicas, é fundamental comprometer-se com os atributos de qualidade de software durante todo o desenvolvimento. Contudo, conforme a realidade apresentada neste trabalho, é indispensável avaliar se as funcionalidades do Conecta Canaã WEB atendem adequadamente à demanda, considerando a norma IEEE 829-2008. Assim, espera-se oferecer conhecimentos valiosos que podem beneficiar a elaboração de testes funcionais em softwares de gestão pública, ampliando o impacto e a relevância deste estudo.

Após a execução dos testes funcionais, pode-se afirmar que todos os casos de teste foram concluídos com êxito, indicando que o sistema Conecta Canaã WEB atende aos requisitos estabelecidos e opera conforme o esperado em diferentes cenários de casos de

testes. A análise dos resultados demonstra a eficácia na integridade do software sob os diferentes perfis de usuário testados, ou seja, os cenários de casos foram aprovados em 100% nos testes.

Desta forma, conclui-se que as vantagens da implementação dos testes, estabelecidos pela IEEE, vão além da melhoria da qualidade, permitindo também a garantia de que o Conecta Canaã WEB é uma ferramenta robusta e confiável para a administração pública. A realização bem-sucedida dos testes funcionais indica a eficácia do sistema em atender às necessidades da comunidade e garantir uma gestão eficiente e transparente das ocorrências reportadas.

Portanto, a implementação de práticas de teste funcionais e a adesão da norma IEEE 829-2008, são fundamentais não apenas para a validação técnica dos sistemas, mas também para a promoção da inovação na gestão pública. O sistema Conecta Canaã oferece uma abordagem que pode servir como referência para a melhoria contínua de sistemas semelhantes em outras localidades, contribuindo para o avanço das cidades inteligentes e a otimização da interação entre administração e comunidade.

REFERÊNCIAS

ANDREA, Jennitta. Envisioning the Next-Generation of Functional Testing Tools. **IEEE Software**, v. 24, 2007. Disponível em: https://doi.org/10.1109/MS.2007.73. Acesso em: 25 ago. 2024.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 9126**: Engenharia de software - Qualidade de software. Rio de Janeiro, 2003.

BARTIÉ, Alexandre. **Garantia da qualidade de software: adquirindo maturidade organizacional**. 1 ed. Rio de Janeiro: Elsevier, 2002.

BATTY, M. *et al.* Smart cities of the future. **The European Physical Journal Special Topics**, v. 214, n. 1, p. 481-518, 2012. Disponível em: https://doi.org/10.1140/epjst/e2012-01703-3. Acesso em: 25 ago. 2024.

Bernardo, Paulo Cheque; KON, Fabio. A Importância dos Testes Automatizados. **Engenharia de Software Magazine**, v. 1, n. 3, p. 54-57, 2008. Disponível em: https://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf. Acesso em: 25 ago. 2024.

CRAIG, Rick D.; JASKIEL, Stefan P. **Systematic Software Testing**. 1 ed. Boston: Artech House Publishers, 2002.

DUSTIN, Elfriede; GARRETT, Thom; GAUF, Bernie. **Implementing automated software testing: how to save time and lower costs while raising quality**. 1º ed. Boston: Addison-WesleyProfessional, 2009.

IEEE COMPUTER SOCIETY. 829-2008 - IEEE Standard for Software and System Test. Fredericksburg, VA, USA: **IEEE Computer Society**, 2008.

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

LINO, Adriano Del Pino. **LabDER - Laboratório Virtual de Ensino-Aprendizagem de Banco de Dados Relacionais: Uma abordagem de avaliação automática de Diagramas ER e SQL**. 2021. Tese (Doutorado) - Universidade de Coimbra. Coimbra, 2021. Disponível em: https://hdl.handle.net/10316/95440. Acesso em: 25 ago. 2024.

MACHADO, Felipe Nery Rodrigues. **Análise e Gestão de Requisitos de Software – Onde nascem os sistemas**. 3 ed. São Paulo: Editora Érica, 2015.

MOLINARI, Leonardo. **Teste de Software – Produzindo sistemas melhores e mais confiáveis**. 4 ed. São Paulo: Editora Érica, 2008.

MYERS, Glenford J.; SANDLER, Corey; BADGETT, Tom. **The art of software testing**. 3 ed. Nova Jersey: John Wiley & Sons, 2012.

Paulk, Marcos. C.; Curtis, Bill; Chrissis, Maria Beth; Weber, Carlos V. **Capability Maturity Model for Software, Version 1.1**. IEEE Software. v. 10, n. 4, p. 481-518, 1993. Disponível em: https://ieeexplore.ieee.org/document/219617. Acesso em: 25 ago. 2024.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software: uma abordagem profissional**. 9. ed. Porto Alegre: AMGH, 2021.

CÓDIGO DE TESTE. **Canaã Conecta Web Tester**. [S. l.]: GitHub, 2024. Disponível em: https://github.com/ramon141/-canaa-conecta-web-tester. Acesso em: 19 nov. 2024.

RIOS, Emerson; MOREIRA, Trayahú. **Teste de software**. 2 ed. Alta Books Editora, 2006.
RIZARDI, Bruno Martins et al. **Caminhos da Inovação no Setor Público**. Brasília: Enap, 2022. Disponível em: http://repositorio.enap.gov.br/handle/1/7420. Acesso em: 25 ago. 2024.

SELENIUM. **Selenium WebDriver**. [S. l.]: SeleniumHQ, 2024. Disponível em: https://www.selenium.dev. Acesso em: 25 ago. 2024.

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

SOFTEX. **Associação para promoção do software brasileiro. Guia Geral de Software**, 2016. Acesso em: 25 ago. 2024.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.