

**QUALIDADE DE SOFTWARE**

Ivaneide Pires Monteiro do Nascimento

Análise de Qualidade

Belém/PA

2024

**RESUMO**

Este trabalho apresenta um projeto cujo objetivo é garantir a qualidade e a confiabilidade do sistema EBAC-SHOP através de uma estratégia de testes robusta e abrangente. O foco está na validação das funcionalidades críticas da plataforma, como a adição de itens ao carrinho, a autenticação de usuários e a implementação de um serviço de cupons, utilizando tanto testes automatizados quanto manuais.

A metodologia adotada envolve a definição de critérios de aceitação utilizando a linguagem Gherkin, o que permite especificar de forma clara os requisitos e os cenários de teste. Foram implementadas técnicas como particionamento de equivalência, análise de valor limite e tabela de decisão para assegurar uma cobertura completa dos diferentes aspectos do sistema em várias fases de teste, desde testes unitários até testes de aceitação do usuário.

Para a execução dos testes, foram utilizadas ferramentas especializadas como Cypress para testes de interface do usuário, JMeter para testes de desempenho, Postman para testes de API e Jenkins para integração contínua. Essa abordagem integrada de testes manuais e automatizados não só identifica e corrige defeitos, mas também assegura que o sistema atende aos requisitos de negócio, proporcionando uma experiência satisfatória ao usuário final.

**SUMÁRIO**

**1. RESUMO..................................................................................................1**

**2. SUMÁRIO .................................................................................................2**

**3. INTRODUÇÃO ..................................................................................................... 4**

**4. O PROJETO ...........................................................................................................6**

**4.1. Estratégia de teste ..........................................................................................7**

**4.2. Critérios de aceitação .....................................................................................8**

**4.2.1. História de usuário 1: [US-0001] – Adicionar item ao carrinho .............9**

**4.2.2. História de usuário 2: [US-0002] – Login na plataforma ......................11**

**4.2.3. História de usuário 3: [US-0003] – API de cupons ...............................13**

**4.3. Casos de testes .............................................................................................14**

**4.3.1. História de usuário 1 ...............................................................................16**

**4.3.2. História de usuário 2 ...............................................................................18**

**4.3.3. História de usuário 3: API de Cupom ....................................................19**

**4.4. Repositório no Github ...................................................................................20**

**4.5. Testes automatizados ...................................................................................21**

**4.6. Integração contínua ......................................................................................22**

**4.7. Testes de performance ................................................................................ 23**

**4.7.1. Teste de Carga .........................................................................................24**

**4.7.2. Teste de Stress ......................................................................................25**

**4.7.3. Teste de Escalabilidade ........................................................................26**

**4.7.4. Teste de Robustez .................................................................................27**

**5. CONCLUSÃO .......................................................................................................28**

**6. REFERÊNCIAS BIBLIOGRÁFICAS .....................................................................29**

**Introdução**

No contexto do desenvolvimento de aplicações contemporâneas, a qualidade de software desempenha um papel crucial, especialmente em plataformas de e-commerce onde a experiência do usuário e a confiabilidade do sistema são essenciais para o sucesso comercial. Este trabalho concentra-se na criação de uma estratégia de testes abrangente para o sistema EBAC-SHOP, com o objetivo de garantir que todas as funcionalidades operem de acordo com as expectativas e que o sistema seja capaz de lidar eficazmente com uma variedade de cenários de uso.

A estratégia de testes adotada abrange a definição de critérios de aceitação precisos, a implementação de diversas técnicas de teste e o uso de ferramentas de automação para assegurar uma cobertura completa e eficiente. Aspectos críticos como autenticação de usuários, adição de itens ao carrinho e implementação de um serviço de cupons serão minuciosamente abordados durante o processo de validação.

Espera-se que ao término deste projeto, o sistema EBAC-SHOP tenha sido submetido a um rigoroso processo de verificação, não apenas identificando e corrigindo possíveis defeitos, mas também garantindo sua conformidade com os requisitos de negócio estabelecidos, resultando em uma experiência satisfatória para os usuários finais. A utilização contínua e integrada de ferramentas como Cypress, JMeter, Postman e Jenkins será fundamental para assegurar que o produto final entregue seja de alta qualidade e atenda aos padrões esperados.

# O PROJETO

Para o Trabalho de Conclusão de Curso em Qualidade de Software, adotamos uma abordagem ágil para garantir o desenvolvimento e teste rigoroso de todas as funcionalidades do sistema EBAC-SHOP. Este projeto segue o fluxo de trabalho de um QA (Analista de Qualidade), desde o planejamento até a entrega final, com foco na aplicação das melhores práticas em documentação, escrita e desenvolvimento.

## 4.1 Estratégia de teste



## 

**4.2 Critérios de Aceitação dos Testes**A seguir estão os critérios de aceitação detalhados para cada história de usuário mencionada no projeto:

4.2.1 **História de usuário 1**: [US-0001] – Adicionar item ao carrinho

Os critérios de aceitação para esta história de usuário foram detalhados para validar diferentes cenários de adição de itens ao carrinho, incluindo restrições de quantidade e valores máximos.

**Cenário 1:** Adicionar menos de 10 itens de um produto ao carrinho

**Contexto:** Dado que estou na página de um produto

**Cenário:** Adicionar 9 itens do produto ao carrinho

**Quando:** Eu adicionar 9 itens do produto ao carrinho

**Então:** O sistema deve adicionar os itens ao carrinho

**E:** O total de itens no carrinho deve ser 9

4.2.2 **História de usuário 2**: [US-0002] – Login na plataforma

Os critérios de aceitação para o login na plataforma abrangem desde cenários de sucesso até situações de tentativas com credenciais inválidas e bloqueio de contas após múltiplas tentativas falhas.

**Cenário 1:** Login com sucesso

**Contexto:** Dado que sou um usuário ativo com o CPF "123.456.789-00" e senha "senha123"

**Cenário:** Inserir o CPF "123.456.789-00" e a senha "senha123" e clicar no botão "Login"

**Quando:** Eu inserir o CPF "123.456.789-00"

**E:** Eu inserir a senha "senha123"

**E:** Eu clicar no botão "Login"

**Então:** Devo ser redirecionado para a página de meus pedidos

**Cenário 2:** Tentativa de login com credenciais inválidas

**Contexto:** Dado que sou um usuário

**Cenário:** Inserir o e-mail "usuario@exemplo.com" e a senha "senhaErrada" e clicar no botão "Login"

**Quando:** Eu inserir o e-mail "usuario@exemplo.com"

**E:** Eu inserir a senha "senhaErrada"

**E:** Eu clicar no botão "Login"

**Então:** Devo ver uma mensagem de erro "Usuário ou senha inválidos"

**Cenário 3:** Travar login após 3 tentativas com senha errada

**Contexto:** Dado que sou um usuário ativo

**Cenário:** Inserir o e-mail ou nome de usuário ou CPF "<login>" e a senha "senhaErrada" e clicar no botão "Login"

**Quando:** Eu inserir o e-mail ou nome de usuário ou CPF "<login>"

**E:** Eu inserir a senha "senhaErrada"

**E:** Eu clicar no botão "Login"

**Então:** Devo ver uma mensagem de erro "Usuário ou senha inválidos"

**E:** Ao tentar logar novamente

**Quando:** Eu inserir o e-mail ou nome de usuário ou CPF "<login>"

**E:** Eu inserir a senha correta "senha123"

**E:** Eu clicar no botão "Login"

**Então:** Devo ver uma mensagem de erro "Conta bloqueada por 15 minutos devido a múltiplas tentativas falhas"

**Exemplos:**

login: usuario@exemplo.com, usuario123, ou 123.456.789-00

4.2.3 **História de usuário 3:** [US-0003] – API de Cupom

**CT01:** Cadastrar cupom com campos obrigatórios válidos

**Contexto:** Dado que sou um admin autenticado na EBAC-SHOP

**Cenário:** Enviar uma requisição POST para cadastrar um novo cupom com os campos obrigatórios válidos

**Quando:** Eu enviar uma requisição POST para cadastrar um novo cupom com os campos obrigatórios válidos

**Então:** O cupom deve ser cadastrado com sucesso

**E:** A resposta deve ter o status code 201

**CT02:** Cadastrar cupom com nome repetido

**Contexto:** Dado que sou um admin autenticado na EBAC-SHOP

**Cenário:** Enviar uma requisição POST para cadastrar um novo cupom com um nome já existente

**Quando:** Eu enviar uma requisição POST para cadastrar um novo cupom com um nome já existente

**Então:** Devo receber uma mensagem de erro "Nome do cupom já existe"

**E:** A resposta deve ter o status code 400

**CT03:** Listar todos os cupons com autenticação válida

**Contexto:** Dado que sou um admin autenticado na EBAC-SHOP

**Cenário:** Enviar uma requisição GET para "/wc/v3/coupons"

**Quando:** Eu enviar uma requisição GET para "/wc/v3/coupons"

**Então:** Devo receber uma lista de todos os cupons cadastrados

**E:** A resposta deve ter o status code 200

**CT04:** Listar todos os cupons com autenticação inválida

**Contexto:** Dado que sou um admin da EBAC-SHOP

**Cenário:** Enviar uma requisição GET para "/wc/v3/coupons" com autenticação inválida

**Quando:** Eu enviar uma requisição GET para "/wc/v3/coupons" com autenticação inválida

**Então:** Devo receber uma mensagem de erro de autenticação

**E:** A resposta deve ter o status code 401

**CT05:** Cadastrar cupom com valor no limite inferior

**Contexto:** Dado que sou um admin autenticado na EBAC-SHOP

**Cenário:** Enviar uma requisição POST para cadastrar um novo cupom com valor "0.01"

**Quando:** Eu enviar uma requisição POST para cadastrar um novo cupom com valor "0.01"

**Então:** O cupom deve ser cadastrado com sucesso

**E:** A resposta deve ter o status code 201

**CT06:** Listar cupons por ID com autenticação válida

**Contexto:** Dado que sou um admin autenticado na EBAC-SHOP

**Cenário:** Enviar uma requisição GET para "/wc/v3/coupons/{id}" com um ID válido

**Quando:** Eu enviar uma requisição GET para "/wc/v3/coupons/{id}" com um ID válido

**Então:** Devo receber os detalhes do cupom com o ID

**4.3 Casos de Testes**

Os casos de testes foram desenvolvidos para validar cada funcionalidade crítica do sistema EBAC-SHOP, garantindo que todos os requisitos de negócio sejam atendidos e que a plataforma ofereça uma experiência de usuário robusta e confiável.

4.3.1 **História de Usuário 1:** Adicionar Item ao Carrinho

CT01: Adicionar menos de 10 itens de um produto ao carrinho\*\*

Contexto: Dado que estou na página de um produto

Quando: Eu adicionar 9 itens do produto ao carrinho

Então: O sistema deve adicionar os itens ao carrinho

E: O total de itens no carrinho deve ser 9

CT02: Tentar adicionar mais de 10 itens de um produto ao carrinho

Contexto: Dado que estou na página de um produto

Quando: Eu tentar adicionar 11 itens do produto ao carrinho

Então: O sistema deve exibir uma mensagem de erro "Não é permitido inserir mais de 10 itens de um mesmo produto ao carrinho"

E: O total de itens no carrinho deve ser 10

CT03: Não permitir que o valor total dos itens ultrapasse R$ 990,00\*\*

Contexto: Dado que o carrinho está vazio

Quando: Eu adicionar itens ao carrinho cujo valor total exceda R$ 990,00

Então: O sistema deve exibir uma mensagem de erro "Os valores não podem ultrapassar R$ 990,00"

E: O valor total no carrinho deve permanecer em R$ 990,00

CT04: Aplicar desconto ao adicionar itens no carrinho\*\*

Contexto: Dado que o carrinho está vazio

Quando: Eu adicionar itens ao carrinho com valor total de <valorTotal>

Então: Devo receber um desconto de <percentualDesconto>%

E: O valor do desconto deve ser de R$ <valorDesconto>

CT05: Não permitir adicionar mais de 10 itens do mesmo produto ao carrinho

Contexto: Dado que o carrinho está vazio

Quando: Eu adicionar mais de 10 itens do mesmo produto ao carrinho

Então: O sistema deve exibir uma mensagem de erro "Não é permitido inserir mais de 10 itens de um mesmo produto ao carrinho"

E: A quantidade de itens desse produto no carrinho deve permanecer em 10

4.3.2 **História de Usuário 2:** Login na Plataforma

CT01: Login com sucesso

Contexto: Dado que sou um usuário ativo com o CPF "123.456.789-00" e senha "senha123"

Quando: Eu inserir o CPF "123.456.789-00" e a senha "senha123" e clicar no botão "Login"

Então: Devo ser redirecionado para a página de meus pedidos

CT02: Tentativa de login com credenciais inválidas

Contexto: Dado que sou um usuário

Quando: Eu inserir o e-mail "usuario@exemplo.com" e a senha "senhaErrada" e clicar no botão "Login"

Então: Devo ver uma mensagem de erro "Usuário ou senha inválidos"

CT03: Travar login após 3 tentativas com senha errada

Contexto: Dado que sou um usuário ativo

Quando: Eu inserir o e-mail ou nome de usuário ou CPF "<login>" e a senha "senhaErrada" e clicar no botão "Login"

Então: Devo ver uma mensagem de erro "Usuário ou senha inválidos"

E: Ao tentar logar novamente com as credenciais corretas dentro de 15 minutos, devo ver uma mensagem de erro "Conta bloqueada por 15 minutos devido a múltiplas tentativas falhas"

4.3.3 **História de Usuário 3:** API de Cupom

CT01: Cadastrar cupom com campos obrigatórios válidos

Contexto: Dado que sou um admin autenticado na EBAC-SHOP

Quando: Eu enviar uma requisição POST para cadastrar um novo cupom com os campos obrigatórios válidos

Então: O cupom deve ser cadastrado com sucesso

E: A resposta deve ter o status code 201

CT02: Cadastrar cupom com nome repetido

Contexto: Dado que sou um admin autenticado na EBAC-SHOP

Quando:Eu enviar uma requisição POST para cadastrar um novo cupom com um nome já existente

Então: Devo receber uma mensagem de erro "Nome do cupom já existe"

E: A resposta deve ter o status code 400

CT03: Listar todos os cupons com autenticação válida

Contexto Dado que sou um admin autenticado na EBAC-SHOP

Quando: Eu enviar uma requisição GET para "/wc/v3/coupons"

Então: Devo receber uma lista de todos os cupons cadastrados

E: A resposta deve ter o status code 200

CT04: Listar todos os cupons com autenticação inválida\*\*

Contexto: Dado que não estou autenticado na EBAC-SHOP

Quando: Eu enviar uma requisição GET para "/wc/v3/coupons" sem autenticação

Então: Devo receber uma mensagem de erro de autenticação

E: A resposta deve ter o status code 401

CT05: Cadastrar cupom com valor mínimo permitido\*\*

Contexto: Dado que sou um admin autenticado na EBAC-SHOP

Quando: Eu enviar uma requisição POST para cadastrar um novo cupom com valor mínimo permitido

Então: O cupom deve ser cadastrado com sucesso

E: A resposta deve ter o status code 201

CT06: Listar cupom por ID com autenticação válida

Contexto: Dado que sou um admin autenticado na EBAC-SHOP

Quando: Eu enviar uma requisição GET para "/wc/v3/coupons/{id}" com um ID válido

Então: Devo receber os detalhes do cupom com o ID especificado

E: A resposta deve ter o status code 200

4.4 **Repositório no Github**

Para garantir a organização e acessibilidade de todos os artefatos do projeto, criamos um repositório no GitHub chamado TCC-EBAC. Nele, estão armazenados os documentos do projeto, código-fonte das automações e qualquer documentação adicional necessária. O repositório é público até a avaliação dos tutores, permitindo a transparência e colaboração.

**Link do repositório:** TCC-EBAC no GitHub

**4.5 Testes Automatizados**

Para garantir a eficácia dos testes, utilizamos ferramentas de automação como Cypress para testes de UI e Postman para testes de API. Aqui estão alguns exemplos:

**Automação de UI (Cypress)**

Criamos um projeto de automação no Cypress para validar a história de usuário "Adicionar item ao carrinho" (US-0001). Os testes de UI incluem adicionar pelo menos três produtos diferentes ao carrinho e validar se os itens foram adicionados com sucesso.

**Exemplo de Automação de API (Postman)**

Para a história de usuário "API de Cupons", criamos coleções de testes no Postman. Implementamos automações para listar os cupons e cadastrar um novo cupom, seguindo os critérios estabelecidos.

**Boas Práticas Aplicadas:**

Utilização de Page Objects para melhor estruturação dos testes.

Implementação de Massa de Dados para cenários variados.

Criação de Custom Commands para reutilização de funcionalidades.

Uso de elementos dinâmicos para garantir a robustez dos testes.

**4.6 Integração Contínua**

Para garantir a continuidade e integração dos testes automatizados, utilizamos Jenkins. Configuramos um pipeline de integração contínua que executa os testes automatizados a cada commit, permitindo a detecção precoce de defeitos e garantindo a qualidade constante do software.

**Configurações do Jenkins:**

Criamos um job no Jenkins que executa os testes automatizados.

Compartilhamos o Jenkinsfile no repositório GitHub para facilitar a configuração e manutenção do pipeline.

**4.7 Testes de Performance**

Os testes de performance são essenciais para garantir que a plataforma EBAC-SHOP possa lidar com uma grande quantidade de usuários simultâneos e grandes volumes de dados sem degradação significativa no desempenho. Para isso, foram realizados diversos tipos de testes de performance:

4.7.1 **Teste de Carga**

Os testes de carga visam avaliar o comportamento do sistema sob condições normais e de pico de uso. Utilizamos ferramentas como JMeter e Locust para simular múltiplos usuários acessando a plataforma simultaneamente.

**Objetivo:** Avaliar o desempenho do sistema sob carga normal e de pico.

**Cenário:** Simulação de 100, 500 e 1000 usuários simultâneos realizando operações comuns como login, navegação de produtos, e adição de itens ao carrinho.

**Métricas Avaliadas:** Tempo de resposta, taxa de erro, throughput e utilização de recursos (CPU e memória).

4.7.2 **Teste de Stress**

Os testes de stress foram realizados para identificar o ponto de falha do sistema, ou seja, o momento em que o sistema começa a falhar ou apresentar uma degradação inaceitável no desempenho.

**Objetivo:** Identificar o ponto de falha do sistema sob condições extremas.

Cenário: Aumento gradual do número de usuários até o ponto em que o sistema não consegue mais responder de forma eficiente.

Métricas Avaliadas: Tempo de resposta, taxa de erro e comportamento do sistema em condições de stress extremo.

4.7.3 **Teste de Escalabilidade**

Os testes de escalabilidade foram conduzidos para garantir que o sistema pode escalar de forma eficiente à medida que o número de usuários e a carga aumentam.

Objetivo: Avaliar a capacidade do sistema de escalar horizontalmente e verticalmente.

Cenário: Adição gradual de recursos (como instâncias de servidor) e avaliação do impacto na performance.

Métricas Avaliadas: Tempo de resposta, throughput e utilização de recursos.

4.7.4 **Teste de Robustez**

Os testes de robustez foram realizados para verificar a capacidade do sistema de se recuperar de falhas inesperadas, como desligamento de servidores ou falhas de rede.

**Objetivo:** Garantir a resiliência e capacidade de recuperação do sistema.

**Cenário:** Simulação de falhas de rede, queda de servidores e avaliação da capacidade de recuperação.

**Métricas Avaliadas:** Tempo de recuperação, perda de dados e impacto na experiência do usuário.

Os resultados dos testes de performance foram documentados e analisados para identificar áreas de melhoria e garantir que o sistema EBAC-SHOP possa oferecer uma experiência de usuário eficiente e sem interrupções, mesmo sob condições extremas

**5. CONCLUSÃO**

O desenvolvimento e a implementação do sistema EBAC-SHOP representaram um esforço significativo para criar uma plataforma de comércio eletrônico robusta e eficiente. Através de uma abordagem meticulosa para a definição de critérios de aceitação, desenvolvimento de casos de teste e realização de testes de performance, conseguimos atingir os objetivos propostos.

Os testes de aceitação garantiram que cada funcionalidade essencial do sistema fosse validada, assegurando que os requisitos de negócio foram atendidos. Os casos de teste cobriram cenários críticos, como adição de itens ao carrinho, login de usuários e operações de cupons, garantindo que o sistema se comportasse conforme esperado em todas as situações.

Os testes de performance demonstraram que o sistema pode lidar com altos volumes de tráfego e carga, mantendo um desempenho aceitável e sendo capaz de escalar conforme necessário. As melhorias identificadas durante esses testes foram implementadas, resultando em uma plataforma mais robusta e eficiente.

O repositório no GitHub facilitou a organização e o acesso aos artefatos do projeto, promovendo a colaboração e a transparência durante todo o ciclo de desenvolvimento.

Em conclusão, o sistema EBAC-SHOP está bem posicionado para atender às demandas do mercado de comércio eletrônico, oferecendo uma experiência de usuário eficiente, segura e confiável. As práticas de teste implementadas garantem a qualidade e a performance do sistema, proporcionando uma base sólida para futuras expansões e melhorias.

**6. REFERÊNCIAS BIBLIOGRÁFICAS**

As referências bibliográficas utilizadas para embasar o desenvolvimento e as práticas de teste do sistema EBAC-SHOP incluem:

Pressman, R. S. (2014). Engenharia de Software: Uma Abordagem Profissional. McGraw-Hill.

Sommerville, I. (2011). Software Engineering. Pearson.

Cohn, M. (2004). User Stories Applied: For Agile Software Development. Addison-Wesley.

JMeter Documentation. (2023). Apache JMeter. Available at: https://jmeter.apache.org](https://jmeter.apache.org)

Locust Documentation. (2023). Locust. Available at: <https://locust.io](https://locust.io)>

Beck, K. (2002). Test Driven Development: By Example. Addison-Wesley.

Fowler, M. (2018). Continuous Integration. Martin Fowler. Available at: https://martinfowler.com/articles/continuousIntegration.html](https://martinfowler.com/articles/continuousIntegration.html)

Nielsen, J. (1994). Usability Engineering. Morgan Kaufmann.

Mapa mental: [Try markmap](https://markmap.js.org/repl)