

**ENGENHEIRO DE QUALIDADE DE SOFTWARE**

**Nome:** Leonardo Martins Pereira dos Reis

**Cidade:** Petrópolis – Rio de Janeiro

**Análise de Qualidade do E-commerce EBAC Shop**

# RESUMO

Este trabalho de conclusão de curso apresenta uma estratégia de testes abrangente para o e-commerce EBAC Shop. O objetivo principal foi garantir a qualidade das funcionalidades chave, como login, adição de itens ao carrinho e API de cupons, além de novas funcionalidades como catálogo de produtos e painel do cliente. Foram definidos critérios de aceitação em Gherkin, criados casos de teste manuais e automatizados para UI (Cypress), API (Supertest) e Mobile (Appium para Android/iOS), e testes de performance (K6). A integração contínua foi implementada com GitHub Actions para execução automática dos testes.

# SUMÁRIO

[1. RESUMO 2](#_Toc99483086)

[2. SUMÁRIO 3](#_Toc99483087)

[3. INTRODUÇÃO 4](#_Toc99483088)

[4. O PROJETO 5](#_Toc99483089)

[4.1 Estratégia de teste 6](#_Toc99483090)

[4.2 Critérios de aceitação 6](#_Toc99483091)

[4.3 Casos de testes 6](#_Toc99483092)

[4.4 Repositório no Github 7](#_Toc99483093)

[4.5 Testes automatizados 7](#_Toc99483094)

[4.6 Integração contínua 8](#_Toc99483095)

[4.7 Testes de performance 8](#_Toc99483096)

[5. CONCLUSÃO 9](#_Toc99483097)

[6. REFERÊNCIAS BIBLIOGRÁFICAS 9](#_Toc99483098)

# INTRODUÇÃO

A qualidade de software é um pilar fundamental para o sucesso de plataformas de e-commerce, impactando diretamente a experiência do usuário e a credibilidade da marca. Este trabalho tem como objetivo aplicar as metodologias e ferramentas de Quality Assurance aprendidas no curso Profissão: Engenheiro de Qualidade de Software da EBAC para realizar uma análise de qualidade completa da loja virtual EBAC Shop. Serão abordadas desde a criação da estratégia de testes, passando pela definição de critérios de aceitação e casos de teste, até a implementação de testes automatizados para diferentes plataformas (Web, API, Mobile), testes de performance e a configuração de um pipeline de integração contínua.

# O PROJETO

Para este trabalho de conclusão de curso **Profissão: Engenheiro de Qualidade de software**, você deve utilizar o conhecimento adquirido ao longo do curso para elaborar uma estratégia de testes adequada para validar o e-commerce EBAC Shop (<http://lojaebac.ebaconline.art.br/>). Você deve considerar as histórias de usuário já refinadas como se você estivesse participando de um time ágil. As funcionalidades devem seguir todo o fluxo de trabalho de um *Quality Engineer* (QE), desde o planejamento até a entrega. Siga as etapas dos sub-tópicos para se orientar no trabalho.

**ATENÇÃO**:

* Conforme a sua estratégia, você pode executar os testes no endereço disponibilizado ou utilizando as imagens disponíveis no Docker Hub:
  + Banco de Dados: [ernestosbarbosa/lojaebacdb](https://hub.docker.com/repository/docker/ernestosbarbosa/lojaebacdb)
  + Loja EBAC: [ernestosbarbosa/lojaebac](https://hub.docker.com/repository/docker/ernestosbarbosa/lojaebac)
    - Comandos para subir os containers:

*docker network create --attachable ebac-network*

*docker run -d --name wp\_db -p 3306:3306 --network ebac-network ernestosbarbosa/lojaebacdb:latest*

*docker run -d --name wp -p 80:80 --network ebac-network ernestosbarbosa/lojaebac:latest*

*Após subir os containers a loja estará em* [*http://localhost:80*](http://localhost:80)

* Como este trabalho complementa o que criou em seu Trabalho de Consolidação (Módulo 19), você pode utilizá-lo como base para o seu Trabalho de Conclusão.

## Estratégia de teste

* Faça uma estratégia de testes em um mapa mental, seguindo algumas diretrizes como objetivos, papeis e responsabilidades, fases de testes, padrões, tipos de testes, técnicas de testes, ambientes, ferramentas, abordagem (manual ou automatizado), framework ou ferramenta usados, plataformas (web, api, mobile), etc.;
* Referência: Módulo 5
* Diagrama

  O conteúdo gerado por IA pode estar incorreto.Após fazer sua estratégia de teste, tire um print e cole aqui:

## Critérios de aceitação

* Considere as histórias de usuário:
  + [US-0001] – Adicionar item ao carrinho
  + [US-0002] – Login na plataforma
  + [US-0003] – API de cupons
* Para cada uma delas crie pelo menos 4 critérios de aceitação usando a linguagem Gherkin;
* Crie histórias de usuário para as funcionalidades:
  + Catálogo de Produtos
  + Painel Minha Conta
  + Meus Pedidos
  + Endereços
  + Detalhes da Conta
* Referência: Módulo 8

## Casos de testes

* Crie pelo menos 4 casos de testes para cada história de usuário, sempre que possível, usando as técnicas de testes (partição de equivalência, valor limite, tabela de decisão etc.).
* Considere sempre o caminho feliz (fluxo principal) e o caminho alternativo e negativo (fluxo alternativo). Exemplo de cenário negativo: “Ao preencher com usuário e senha inválidos deve exibir uma mensagem de alerta...”
* Identifique quais os casos de teste serão automatizados, sendo ao menos 1 caminho feliz e 1 caminho alternativo.
* Referência: Módulos 4 e 5

**4.3.1 US-0001 – Adicionar item ao carrinho**

* **CT\_US0001\_01: Adicionar produto com variações (cor e tamanho) com sucesso.**
  + **Objetivo:** Validar a adição de um produto configurável ao carrinho após selecionar todas as variações obrigatórias.
  + **Resultado Chave Esperado:** Produto correto é adicionado ao carrinho, mensagem de sucesso é exibida e contador do carrinho é atualizado.
  + **Status:** Automatizado (UI) - Caminho Feliz.
* **CT\_US0001\_02: Tentar adicionar produto sem selecionar variação obrigatória.**
  + **Objetivo:** Verificar o tratamento de erro ao tentar adicionar um produto configurável sem selecionar uma variação necessária.
  + **Resultado Chave Esperado:** Produto não é adicionado, mensagem de alerta solicitando a seleção da variação é exibida.
  + **Status:** Automatizado (UI) - Caminho Negativo.
* **CT\_US0001\_03: Adicionar múltiplas unidades de um produto ao carrinho.**
  + **Objetivo:** Validar a adição de mais de uma unidade de um item, verificando a atualização da quantidade e subtotal.
  + **Resultado Chave Esperado:** Produto adicionado com a quantidade especificada, carrinho e subtotal atualizados corretamente.
  + **Status:** Automatizado (UI) - Caminho Feliz.
* **CT\_US0001\_04: Tentar adicionar produto fora de estoque.**
  + **Objetivo:** Verificar a interface e a funcionalidade para produtos indisponíveis.
  + **Resultado Chave Esperado:** Botão "Comprar" desabilitado ou mensagem de "Fora de estoque", impedindo a adição ao carrinho.
  + **Status:** Manual (ou UI) - Caminho Negativo.

**4.3.2 US-0002 – Login na plataforma**

* **CT\_US0002\_01: Login com sucesso (usuário ativo, credenciais válidas).**
  + **Objetivo:** Validar o fluxo principal de autenticação bem-sucedida.
  + **Resultado Chave Esperado:** Usuário redirecionado para o painel "Minha Conta" com mensagem de boas-vindas.
  + **Status:** Automatizado (UI) - Caminho Feliz.
* **CT\_US0002\_02: Tentativa de login com senha incorreta.**
  + **Objetivo:** Validar a mensagem de erro para senhas inválidas.
  + **Resultado Chave Esperado:** Permanência na página de login com mensagem de erro específica sobre senha incorreta.
  + **Status:** Automatizado (UI) - Caminho Negativo.
* **CT\_US0002\_03: Tentativa de login com usuário inexistente.**
  + **Objetivo:** Validar a mensagem de erro para usuários não cadastrados.
  + **Resultado Chave Esperado:** Permanência na página de login com mensagem de erro específica sobre usuário desconhecido.
  + **Status:** Automatizado (UI) - Caminho Negativo.
* **CT\_US0002\_04: Bloqueio de conta após múltiplas tentativas falhas.**
  + **Objetivo:** Validar a regra de negócio de bloqueio temporário de conta.
  + **Resultado Chave Esperado:** Mensagem de bloqueio exibida após 3 tentativas falhas, impedindo login subsequente por 15 minutos.
  + **Status:** Manual - Negativo (Regra de Negócio).

**4.3.3 US-0003 – API de cupons**

* **CT\_US0003\_01 (API Exemplo): Validar cupom existente e ativo.**
  + **Objetivo:** Verificar se a API retorna sucesso e os dados corretos para um cupom válido.
  + **Resultado Chave Esperado:** Resposta 200 OK, JSON com {"valido": true} e detalhes do desconto.
  + **Status:** Automatizado (API) - Caminho Feliz.
* **CT\_US0003\_02 (API Exemplo): Validar cupom inexistente.**
  + **Objetivo:** Verificar se a API trata corretamente cupons não encontrados.
  + **Resultado Chave Esperado:** Resposta 404 ou 400, JSON com {"valido": false, "mensagem": "Cupom não encontrado"}.
  + **Status:** Automatizado (API) - Caminho Negativo.
* **CT\_US0003\_03 (API Exemplo): Validar cupom expirado.**
  + **Objetivo:** Verificar se a API identifica e rejeita cupons expirados.
  + **Resultado Chave Esperado:** Resposta adequada (ex: 200 com {"valido": false} ou 400), JSON com {"valido": false, "mensagem": "Cupom expirado"}.
  + **Status:** Automatizado (API) - Caminho Negativo.
* **CT\_US0003\_04 (API Exemplo): Requisição de validação de cupom sem o código.**
  + **Objetivo:** Verificar a validação de entrada da API para campos obrigatórios.
  + **Resultado Chave Esperado:** Resposta 400 Bad Request, JSON com mensagem de erro sobre campo obrigatório.
  + **Status:** Automatizado (API) - Caminho Negativo.

## Repositório no Github

* Crie um repositório no github com o nome TCC-EBAC-QE;
* Deixe o repositório publico até a análise dos tutores;
* Neste repositório você deve subir este arquivo e todos os código fontes das automações que criar.
* Referência: Módulo 10
* Link do repositório: https://github.com/Ieozin/TCC-EBAC-QE

## Testes automatizados

* + 1. Automação de UI

A escolha do Cypress para automação de UI se deu por sua arquitetura moderna que executa testes diretamente no navegador, proporcionando maior velocidade e confiabilidade. Sua API intuitiva, recursos integrados como assertions e mocks, e a facilidade de debug com snapshots e vídeos são ideais para o escopo deste projeto. Além disso, sendo baseado em JavaScript, alinha-se com as outras ferramentas de automação escolhidas (Supertest, K6), promovendo consistência no stack tecnológico. Comparado ao Selenium, o Cypress oferece uma curva de aprendizado mais suave e configuração simplificada. Em relação ao Playwright, embora promissor, o Cypress possui uma comunidade e um conjunto de recursos de relatórios mais consolidados no momento, o que é benéfico para um TCC.

* + 1. Automação de API
* Crie uma pasta chamada API para os testes de API dos casos de teste que forem automatizados
* Você deve utilizar a ferramenta Supertest para criar seus testes de API
* Não esqueça de validar os contratos! ☺
  + 1. Automação Mobile
* Considere para os APPs apenas a funcionalidade de Catálogo de Produtos
* Você pode encontrar os APPs em:
  + *Android*: <https://github.com/EBAC-QE/testes-mobile-ebac-shop/tree/main/app/android>
  + *iOS*: <https://github.com/EBAC-QE/testes-mobile-ebac-shop/tree/ios-tests/app/ios>
* Crie uma pasta chamada Mobile para os testes em aplicativos dos casos de teste que forem automatizados
* Utilize ao menos um *Testing Pattern* (à sua escolha) na implementação dos testes.
* Você deve implementar testes para ao menos uma das plataformas Mobile (*Android* ou *iOS*)
* Observações:
  + Considere todas as boas práticas aprendidas até aqui
  + Não esqueça de implementar a geração de relatórios
* Referência: Módulos 11, 12, 14, 16, 17, 22, 23, 24, 29 e 30

## Integração contínua

* Execute os testes automatizados em integração contínua utilizando o Github Actions
* Referência: Módulo 26

## Testes de performance

* Usando o K6, implemente um teste de performance em ao menos 2 casos de testes
* Referência: Módulo 28
* Configurações do teste de performance:

-Usuários virtuais: 20  
-Tempo de execução: 2 minutos  
-RampUp: 20 segundos  
-Massa de dados: Usuário / senha:

user1\_ebac / psw!ebac@test  
user2\_ebac / psw!ebac@test  
user3\_ebac / psw!ebac@test  
user4\_ebac / psw!ebac@test  
user5\_ebac / psw!ebac@test

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

# CONCLUSÃO

Coloque sua experiência na realização do trabalho, o que aprendeu, quais lições pode aplicar em sua vida profissional etc.

# REFERÊNCIAS BIBLIOGRÁFICAS

Seguir regras ABNT