

**ENGENHEIRO DE QUALIDADE DE SOFTWARE**

Itnã Santana Nunes

Analista de Qualidade de Software

Salvador

2024

# RESUMO

Este projeto integra uma análise teórica e prática dos tópicos abordados no curso de Engenharia de Qualidade de Software oferecido pela EBAC. O objetivo é explorar as técnicas e práticas que melhoram a qualidade de software, como o desenvolvimento orientado a comportamentos (BDD), integração contínua, e a execução de diversos tipos de testes, incluindo performance, segurança, e automação para back-end e front-end.

O trabalho detalha o planejamento e a implementação de testes em diferentes plataformas e linguagens, alinhando-se à cultura DevOps e utilizando ferramentas como Selenium WebDriver para automação de testes web e mobile. Através deste estudo, pretende-se demonstrar como essas técnicas contribuem para a entrega de produtos de alta qualidade e a melhoria da experiência do usuário, atendendo às demandas crescentes do mercado de TI.

SUMÁRIO

[***1. RESUMO 2***](#_gjdgxs)

[***2. INTRODUÇÃO 3***](#_1fob9te)

[***3. O PROJETO 4***](#_3znysh7)

[**3.1 Estratégia de teste 5**](#_2et92p0)

[**3.2 Critérios de aceitação 6**](#_tyjcwt)

[**3.3 Casos de testes 11**](#_3dy6vkm)

[**3.4 Repositório no Github 11**](#_1t3h5sf)

[**3.5 Testes automatizados 11**](#_4d34og8)

[**3.6 Integração contínua 12**](#_2s8eyo1)

[**3.7 Testes de performance 12**](#_17dp8vu)

[***4. CONCLUSÃO 13***](#_3rdcrjn)

[***5. REFERÊNCIAS BIBLIOGRÁFICAS 13***](#_26in1rg)

# INTRODUÇÃO

Desde a década de 1980, quando se reconheceu a necessidade crescente de garantir a confiabilidade e a eficiência dos projetos de tecnologia da informação, a atuação de um profissional de Qualidade de Software (QA) se tornou essencial nas empresas (SOMMERVILLE, 2015). A importância do analista de testes foi reforçada pelo avanço das práticas de desenvolvimento e pela crescente demanda por produtos livres de falhas. Hoje, a presença deste profissional é fundamental em todo o ciclo de desenvolvimento, desde as fases iniciais de análise até a entrega final do projeto, assegurando a qualidade e a funcionalidade do sistema.

A abordagem contemporânea para a garantia de qualidade exige a elaboração de documentações detalhadas, como escopos e critérios de aceitação. Esses documentos são fundamentais para o planejamento e a execução dos testes, assegurando uma validação precisa dos requisitos. As práticas envolvem diversos tipos de testes que são essenciais para assegurar a qualidade do software em múltiplos aspectos.

Os testes automatizados, por exemplo, desempenham um papel crucial ao permitir a execução eficiente e repetitiva de testes complexos. Testes de regressão são fundamentais para garantir que novas alterações não comprometem funcionalidades existentes, enquanto testes de API verificam a integração e a comunicação entre diferentes sistemas. Testes de performance avaliam a robustez e a escalabilidade do sistema, garantindo que ele possa suportar a carga de usuários e transações previstas. Esses tipos de testes, realizados dentro de um ciclo de integração contínua, ajudam a identificar e corrigir problemas de forma rápida e eficiente, promovendo a entrega de produtos de alta qualidade.

Nos últimos anos, muitas organizações têm reconhecido a importância de incluir especialistas em testes em suas equipes, percebendo o impacto positivo na qualidade e na confiabilidade dos produtos. Embora os testes não possam garantir a eliminação total de erros, eles reduzem significativamente a probabilidade de falhas críticas e melhoram a experiência do usuário final.

Além disso, metodologias ágeis ressaltam a responsabilidade compartilhada pela qualidade entre todos os membros da equipe, conforme destacado no Manifesto do Teste Ágil (GROWING AGILE, 2016). Este trabalho abordará, nas seções seguintes, as práticas e estratégias para garantir a qualidade do software, incluindo a aplicação de testes automatizados em plataformas web e mobile, testes de API e o uso de integração contínua e testes de performance. O estudo baseia-se nos conhecimentos adquiridos no curso “Engenheiro de Qualidade de Software” da EBAC e visa demonstrar a eficácia dessas abordagens na entrega de produtos de alta qualidade.

# O PROJETO

Para este trabalho de conclusão de curso **Profissão: Engenheiro de Qualidade de software**, será utilizado 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/>). Será considerado as histórias de usuário já refinadas como se estivesse participando de um time ágil. As funcionalidades seguirão todo o fluxo de trabalho de um *Quality Engineer* (QE), desde o planejamento até a entrega que serão descritas nas etapas dos sub-tópicos.

**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*

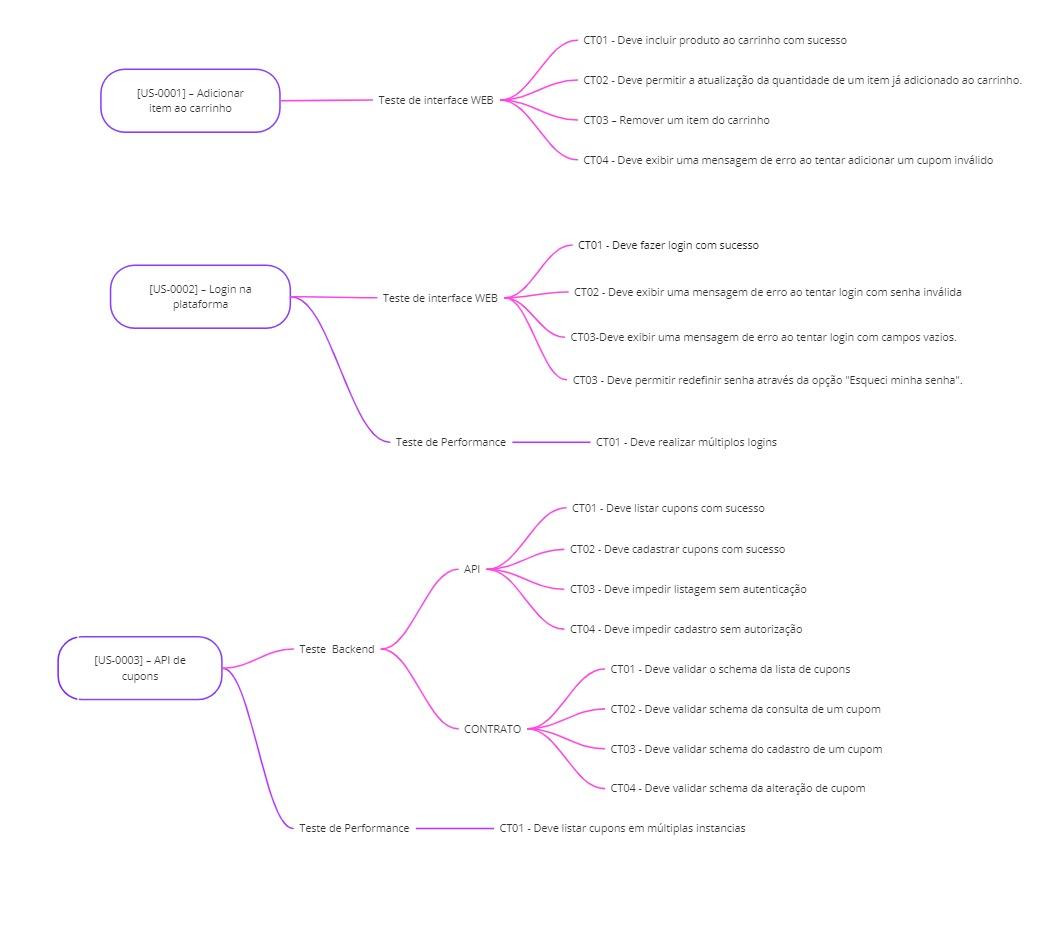
*docker run -d --name wp -p 80:80 --network ebac-network ernestosbarbosa/lojaebac:latest*

*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

A estratégia de testes é um componente crucial do planejamento de qualidade de qualquer projeto de desenvolvimento de software.



Fonte: o autor (2024)

## Critérios de aceitação

Histórias de usuário:

* + 1. [US-0001] – Adicionar item ao carrinho
    2. [US-0002] – Login na plataforma
    3. [US-0003] – API de cupons

| História de usuário | Cenário | Critério de Aceite |
| --- | --- | --- |
| US0001 | CT01 - Incluir produto ao carrinho com sucesso | **Dado** que o usuário está na página do produto  **Quando** o usuário seleciona um item disponível  **E** o usuário clica no botão "Adicionar ao carrinho"  **Então** o item deve ser adicionado ao carrinho  **E** o carrinho deve exibir o item adicionado |
| US0001 | CT02 - Atualizar a quantidade de um item já adicionado ao carrinho | **Dad**o que o usuário tem um item no carrinho  **Quando** o usuário aumenta a quantidade do item no carrinho  **Então** a quantidade do item deve ser atualizada no carrinho  **E** o preço total deve ser ajustado de acordo |
| US0001 | CT03 - Adicionar um cupom ao carrinho | **Dado** que o usuário tem itens no carrinho  **Quando** o usuário insere um código de cupom válido  **Então** o desconto correspondente ao cupom deve ser aplicado ao total do carrinho |
| US0001 | CT04 - Tentar adicionar um item ao carrinho com dados faltando | **Dado** que o usuário está na página de um produto  **Quando** o usuário tenta adicionar o produto ao carrinho sem selecionar todos os atributos necessários  **Então** uma mensagem de erro deve ser exibida indicando que todos os campos obrigatórios devem ser preenchidos |

| História de usuário | Cenário | Critério de Aceite |
| --- | --- | --- |
| US0002 | CT01 - Deve fazer login com sucesso | **Dado** que o usuário está na página de login  **Quando** o usuário insere credenciais válidas e clica no botão "Entrar"  **Então** o usuário deve ser autenticado com sucesso  **E** deve ser redirecionado para a página inicial |
| US0002 | CT02 - Exibir mensagem de erro ao tentar login com senha inválida | **Dado** que o usuário está na página de login  **Quando** o usuário insere um email válido e uma senha inválida e clica no botão "Entrar"  **Então** uma mensagem de erro deve ser exibida informando que as credenciais são inválidas |
| US0002 | CT03 - Deve exibir uma mensagem de erro ao tentar login com campos vazios. | **Dado** que o usuário está na página de login da plataforma  **Quando** o usuário deixa o campo de e-mail vazio  **E** deixa o campo de senha vazio  **E** clica no botão "Entrar"  **Então** o sistema deve exibir uma mensagem de erro informando que o e-mail e a senha são obrigatórios |
| US002 | CT05 - Fazer login com sucesso | **Dado** que o usuário está na página de login  **Quando** o usuário clica na opção "Esqueci minha senha" e insere um email válido  **Então** um email de redefinição de senha deve ser enviado para o endereço de email fornecido |

| História de usuário | Cenário | Critério de Aceite |
| --- | --- | --- |
| US0003 | CT01 - Listar cupons com sucesso | **Dado** que o usuário tem permissão para acessar a API de cupons  **Quando** o usuário faz uma requisição GET para a lista de cupons  **Então** a API deve retornar uma lista de cupons com status 200 |
| US0003 | CT02 - **Cadastrar cupons com sucesso** | **Dado** que o usuário tem permissão para cadastrar cupons  **Quando** o usuário faz uma requisição POST para cadastrar um novo cupom com dados válidos  **Então** o cupom deve ser cadastrado com sucesso  **E** a API deve retornar uma mensagem de sucesso com status 201 |
| US0003 | CT03 - Impedir listagem sem autenticação | **Dado** que o usuário não tem permissão para cadastrar cupons  **Quando** o usuário faz uma requisição POST para cadastrar um novo cupom  **Então** a API deve retornar uma mensagem de erro com status 403 |
| US0003 | CT04 - Impedir cadastro sem autorização | **Dado** que o usuário está na página de login  **Quando** o usuário insere um email válido e uma senha inválida e clica no botão "Entrar"  **Então** uma mensagem de erro deve ser exibida informando que as credenciais são inválidas |

* Crie histórias de usuário para as funcionalidades:

| Histórias de usuário | Descrição | Regra de negócios |
| --- | --- | --- |
| US0004 - Meus Pedidos | **Como** cliente da EBAC-SHOP,  **Quero** visualizar o histórico dos meus pedidos,  **Para que** eu possa acompanhar o status de minhas compras e rever pedidos anteriores. | * Somente pedidos associados ao usuário autenticado devem ser exibidos na lista de pedidos. * O status do pedido deve ser atualizado automaticamente com base em eventos de processamento e envio. * Detalhes do pedido devem ser exibidos apenas se o pedido estiver associado ao usuário autenticado para garantir a privacidade. |
| US0005 - Endereços | **Como** cliente da EBAC-SHOP,  **Quero** gerenciar meus endereços de entrega,  **Para que** eu possa adicionar, editar ou remover endereços conforme necessário. | * O sistema deve validar o formato dos endereços inseridos para garantir a entrega correta (por exemplo, formato de CEP). * Um endereço não pode ser excluído se estiver associado a um pedido em andamento ou já concluído. * Um usuário deve ter pelo menos um endereço registrado para realizar uma compra. Não é permitido excluir todos os endereços. |
| US0006 - Catálogo de Produtos | **Como** cliente da EBAC-SHOP,  **Quero** visualizar uma lista de produtos no catálogo,  **Para que** eu possa encontrar e escolher o produto que desejo comprar. | * Todos os produtos no catálogo devem estar associados a uma categoria. Produtos sem categoria não devem ser exibidos. * Os preços dos produtos devem estar sempre atualizados com base nas últimas alterações feitas pelo administrador. * Produtos com estoque esgotado devem ser marcados como “Fora de Estoque” e não devem ser comprados até que o estoque seja reposto. |
| US0007 - Painel Minha Conta | **Como** um usuário registrado,  **Quero** acessar um painel de controle da minha conta,  **Para que** eu possa gerenciar minhas informações pessoais e preferências. | * Acesso ao painel deve exigir autenticação segura; usuários não autenticados devem ser redirecionados para a página de login. * Mudanças nas informações da conta, como email e senha, devem exigir uma confirmação por email para garantir segurança. * Alterações nas configurações de conta (como preferências de comunicação) devem ser salvas e refletidas imediatamente em todas as partes do sistema. |
| US0008 - Detalhes da Conta | **Como**cliente da EBAC-SHOP,  **Quero** visualizar e editar os detalhes da minha conta,  **Para que** eu possa atualizar minhas informações pessoais e preferências de contato. | * Mudanças no email ou telefone devem ser validadas por um código de confirmação enviado ao novo email ou número. * Informações sensíveis, como senha, devem ser atualizadas através de um processo de autenticação adicional. * Alterações nas preferências de comunicação devem ser aplicadas imediatamente e refletidas em todas as comunicações futuras. |

## Casos de testes

**[US-0001] – Adicionar item ao carrinho**

1. Deve incluir produto ao carrinho com sucesso
2. Deve permitir a atualização da quantidade de um item já adicionado ao carrinho.
3. Deve permitir adicionar um cupom
4. Deve exibir uma mensagem de erro ao tentar adicionar um item ao carrinho faltando dados

**[US-0002] – Login na plataforma**

1. Deve exibir uma mensagem de erro ao tentar login com campos vazios
2. Deve permitir redefinir senha através da opção "Esqueci minha senha".
3. Deve exibir uma mensagem de erro ao tentar login com senha inválida.
4. Deve realizar múltiplos logins

**[US-0003] – API de cupons**

1. Deve listar cupons com sucesso
2. Deve cadastrar cupons com sucesso
3. Deve impedir listagem sem autenticação
4. Deve impedir cadastro sem autorização
5. Deve validar o schema da lista de cupons
6. Deve validar schema da consulta de um cupom
7. Deve validar schema do cadastro de um cupom
8. Deve validar schema da alteração de cupom
9. Teste de Performance
10. Deve listar cupons em múltiplas instâncias

## Repositório no Github

Link do repositório: https://github.com/itnanunes/TCC-EBAC-QE

## Testes automatizados

* + 1. Automação de UI

**Ferramenta Escolhida: Cypress**

Para a realização dos testes automatizados, optou-se pela ferramenta Cypress. Esta escolha foi motivada por uma série de vantagens significativas que a ferramenta oferece, que são cruciais para garantir a eficiência e a eficácia dos testes.

#### Vantagens:

* **Integração Direta com JavaScript e Node.js:** O Cypress é nativamente compatível com JavaScript e Node.js, que são amplamente utilizados no desenvolvimento web moderno. Essa integração direta facilita a implementação e manutenção dos testes, aproveitando o ambiente de desenvolvimento já estabelecido.
* **Interface Intuitiva e Visualização em Tempo Real:** A interface do Cypress é projetada para ser intuitiva, permitindo que os desenvolvedores visualizem os testes enquanto eles estão sendo executados. Isso é especialmente útil para identificar e corrigir erros de forma rápida e eficiente.
* **Monitoramento e Interceptação de Requisições HTTP:** O Cypress permite monitorar e interceptar requisições HTTP, o que é fundamental para verificar o carregamento das páginas e validar as respostas da API. Esta funcionalidade aprimora a capacidade de teste e depuração.
* **Suporte a Testes End-to-End:** Com comandos simplificados para interação com elementos da interface do usuário, o Cypress proporciona um suporte robusto para testes end-to-end, cobrindo todos os aspectos da aplicação desde a interface até a lógica de backend.
* **Testes Assíncronos e Esperas Automáticas:** A ferramenta lida eficientemente com testes assíncronos e inclui esperas automáticas, o que facilita a sincronização com o tempo de carregamento dos elementos da página e reduz a necessidade de manipulação manual de tempos de espera.

**Ferramentas Não Escolhidas:**

* **Selenium:**
  + **Desvantagens:** Embora o Selenium seja uma ferramenta amplamente reconhecida, sua configuração pode ser mais complexa, exigindo a instalação de drivers específicos para cada navegador. Além disso, os testes podem apresentar lentidão em cenários mais complexos devido à sua arquitetura.
* **Robot Framework:**
  + **Desvantagens:** O Robot Framework possui uma curva de aprendizado mais acentuada, especialmente para aqueles que não estão familiarizados com sua sintaxe baseada em palavras-chave. É mais indicado para grandes suítes de testes que envolvem múltiplos tipos de sistemas e pode não ser a melhor escolha para projetos com requisitos mais específicos ou menores.
* **Ghost Inspector:**
  + **Desvantagens:** Como uma ferramenta baseada em nuvem, o Ghost Inspector pode limitar a flexibilidade de customização e a integração com pipelines de CI/CD, o que pode ser uma desvantagem para projetos que exigem um controle mais preciso sobre o ambiente de teste e a automação contínua.

**Ferramentas e Bibliotecas Utilizadas:**

* **Linguagem:** JavaScript
* **Framework de Teste:** Cypress
* **Plugins:** Mochawesome para geração de relatórios
* **Biblioteca para Dados Fictícios:** Faker
  + 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

A conclusão deste projeto foi uma jornada desafiadora e enriquecedora, que exigiu um alto nível de empenho e a aplicação prática dos conhecimentos adquiridos ao longo do curso. Desde o início, o trabalho envolveu o uso de técnicas avançadas de testes de software, incluindo automação e práticas de integração contínua, o que proporcionou uma visão mais profunda e prática sobre o processo de garantia de qualidade.

O planejamento do projeto foi uma etapa crucial, envolvendo a definição de cenários de teste variados, como casos de sucesso, falhas, segurança e desempenho. Essa abordagem garantiu que todas as áreas importantes do software fossem avaliadas de forma abrangente, permitindo a identificação e resolução de problemas antes da entrega final.

Este trabalho não só reforçou a base teórica adquirida durante o curso de Engenharia de Qualidade de Software da EBAC, mas também ofereceu uma experiência prática valiosa. A aplicação de técnicas de automação e o uso de práticas ágeis ajudaram a aprimorar minhas habilidades e conhecimentos, fortalecendo minha capacidade de contribuir para a entrega de software de alta qualidade.

Em resumo, este projeto foi um passo importante no meu desenvolvimento profissional. As habilidades e conhecimentos adquiridos serão fundamentais para enfrentar novos desafios e aproveitar oportunidades na área de testes de software, especialmente em contextos de automação e metodologias ágeis.

# REFERÊNCIAS BIBLIOGRÁFICAS

GROWING AGILE. O Manifesto do Teste Ágil. 2016. Disponível em: <https://www.growingagile.com.br>. Acesso em: 29 ago. 2024.

ISO/IEC 25010. Sistemas e software de engenharia – Qualidade. Genebra: ISO, 2017.

SOMMERVILLE, I. Engenharia de Software. 10. ed. Porto Alegre: Bookman, 2015.

VOLPATO, A. Avaliação de Sistemas: Conceitos e Práticas. 2. ed. São Paulo: Pearson, 2017.