

**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](#_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

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

* 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
* Após fazer sua estratégia de teste, tire um print e cole aqui:

[Imagem: Mapa mental – Estratégia de teste]

## 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

## 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/itnanunes/TCC-EBAC-QE

## Testes automatizados

* + 1. Automação de UI
* Crie um projeto de automação WEB com o framework e a linguagem que preferir
* Justifique a sua escolha através de um comparativo entre ao menos 3 opções de ferramentas e linguagem.
* Crie uma pasta chamada UI para os testes WEB dos casos de teste que forem automatizados
* Utilize ao menos um *Testing Pattern* (à sua escolha) na implementação dos testes.
  + 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.