ENGENHEIRO DE QUALIDADE DE SOFTWARE

Gabriel Thiego Trindade Sperduto

Análise de Qualidade

Rio de Janeiro

2025

1. RESUMO

Este Trabalho de Conclusão de Curso tem como objetivo aplicar as práticas aprendidas ao longo da formação de Engenheiro de Qualidade de Software da EBAC, utilizando a plataforma EBAC Shop como base de testes. Foram criadas e refinadas histórias de usuário com critérios de aceitação e casos de teste, seguidos de automações em três camadas: interface web (UI), API e mobile. A plataforma foi executada localmente via Docker, permitindo total controle do ambiente. Os testes automatizados foram implementados com Cypress, Supertest e Appium, com integração contínua via GitHub Actions e testes de performance usando K6. O projeto evidencia o ciclo completo da atuação de um profissional de QA, desde o planejamento até a entrega de valor com foco em qualidade, eficiência e rastreabilidade.

2. sumário

3. INTRODUÇÃO

Este trabalho tem como objetivo validar a plataforma EBAC Shop por meio de uma abordagem prática baseada em histórias de usuário e testes automatizados. A proposta é aplicar o ciclo completo de atuação do Engenheiro de Qualidade de Software, desde o planejamento até a entrega dos resultados, utilizando ferramentas e técnicas aprendidas ao longo do curso. A loja foi executada em ambiente local por meio de containers Docker, garantindo autonomia e controle sobre os testes.

A estrutura do projeto inclui testes manuais e automatizados nas camadas de interface (UI), API e mobile. As ferramentas utilizadas foram Cypress, Supertest, Appium e K6, com execução em integração contínua via GitHub Actions. Todas as etapas do projeto foram documentadas e versionadas em repositório público.

Este documento apresenta a estratégia de testes, critérios de aceitação em Gherkin, casos de teste aplicando técnicas como partição de equivalência e valor limite, além da justificativa das escolhas técnicas feitas para cada tipo de automação. A experiência adquirida ao longo da execução serviu como base para consolidar o conhecimento técnico necessário para atuação na área de Qualidade de Software.

4. O PROJETO

4.1 Ambiente de Testes

A loja EBAC Shop foi executada localmente com uso de containers Docker, conforme instruções fornecidas no material do curso e tutoriais do Módulo 19. O ambiente foi configurado utilizando as imagens públicas disponibilizadas no Docker Hub:

- Banco de Dados: `ernestosbarbosa/lojaebacdb:latest`
- Loja EBAC: `ernestosbarbosa/lojaebac:latest`

Comandos utilizados:

```
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 a execução dos containers, a loja fica acessível em: **http://localhost:80**

Este ambiente serviu como base para os testes automatizados e manuais desenvolvidos neste projeto.

4.2 Estratégia de Teste

A estratégia de teste foi construída com base no ciclo completo de qualidade, considerando planejamento, execução e entrega. O objetivo principal é garantir que as funcionalidades críticas da EBAC Shop funcionem conforme o esperado, com foco em estabilidade, usabilidade, desempenho e cobertura de testes nas principais plataformas (web, API e mobile).

Os papéis e responsabilidades foram concentrados no perfil do Engenheiro de Qualidade, responsável por escrever as histórias de usuário, refinar critérios de aceitação, desenhar os casos de teste e implementar automações.

Fases de testes adotadas:

- Planejamento e refinamento de histórias;
- Elaboração de critérios e casos de teste;
- Execução de testes manuais;
- Automação dos testes nas três camadas (UI, API e mobile);
- Execução em CI com GitHub Actions;
- Testes de performance com K6.

Tipos de teste aplicados:

- Funcionais (UI e API)
- Exploratório (manual)
- Mobile (Android)
- Performance (K6)
- Regressão automatizada

Técnicas aplicadas: partição de equivalência, valor limite e tabela de decisão

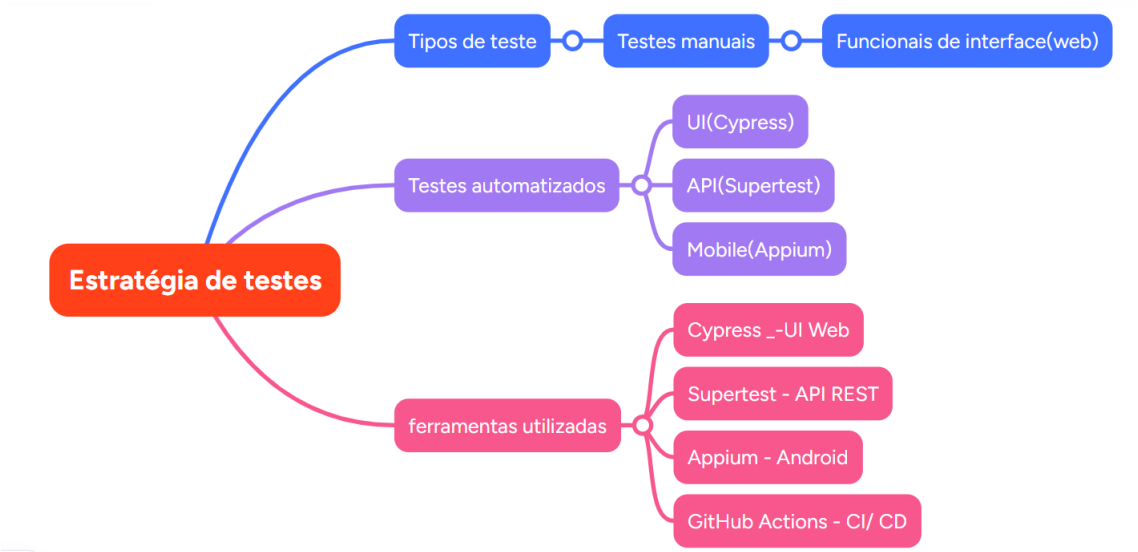
Abordagem: híbrida (testes manuais e automatizados)

Ambiente: loja EBAC local em Docker

Ferramentas: Cypress, Supertest, Appium, GitHub Actions, K6

Frameworks: Mocha/Chai (API), Cucumber (Mobile)

O mapa mental com a visão geral da estratégia de teste está apresentado a seguir:



4.3.1 Casos de Teste

Este tópico apresenta os casos de teste elaborados para as histórias de usuário do projeto, contemplando testes automatizados e manuais. Os casos utilizam técnicas como partição de equivalência e valor limite para garantir cobertura eficaz. Para cada história, pelo menos um caminho feliz e um alternativo foram considerados.

4.3.2 Casos de Teste Automatizados

História [US-0001] – Adicionar item ao carrinho (UI)

- **CT01: Adicionar item tamanho XL e cor verde**
- Técnica: Caminho feliz.
- Objetivo: Verificar se o produto com variações selecionadas é adicionado corretamente ao carrinho.
- **CT02: Adicionar dois itens iguais**

Técnica: Caminho feliz e valor limite.

Objetivo: Confirmar que a quantidade no carrinho atualiza para 2 quando adicionados itens duplicados.

- **CT03: Remover item do carrinho**

Técnica: Caminho alternativo.

Objetivo: Validar que a remoção do item atualiza o carrinho para estado vazio.

História [US-0002] – Login na plataforma (UI)

- **CT01: Login com credenciais válidas**

Técnica: Caminho feliz.

Objetivo: Confirmar autenticação bem-sucedida e redirecionamento correto.

- **CT02: Login com e-mail válido e senha incorreta**

Técnica: Caminho alternativo.

Objetivo: Verificar exibição de mensagem de erro ao tentar login com senha inválida.

História [US-0003] – API de Cupons

- **CT01: Criar cupom com valor fixo válido**

Técnica: Caminho feliz.

Objetivo: Garantir criação de cupom via API com dados válidos.

- **CT02: Atualizar valor do cupom existente**

Técnica: Caminho feliz.

Objetivo: Confirmar alteração do valor do cupom via API.

- **CT03: Consultar cupom por ID**

Técnica: Caminho feliz.

Objetivo: Validar consulta dos dados atualizados do cupom via API.

- **CT04: Consultar cupom inexistente**

Técnica: Caminho alternativo.

Objetivo: Verificar resposta adequada para cupom não encontrado.

História [US-0007] – Catálogo de Produtos (Mobile)

- **CT01: Visualizar lista de produtos**

Técnica: Caminho feliz.

Objetivo: Garantir que o catálogo carrega e exibe produtos corretamente no aplicativo.

- **CT02: Filtrar produtos por categoria**

Técnica: Caminho alternativo.

Objetivo: Validar funcionalidade de filtro para categorias específicas.

4.3.3 Casos de Teste Manuais

História [US-0004] – Catálogo de Produtos (Web)

- **CT01: Visitar página do catálogo**

Técnica: Caminho feliz.

Objetivo: Confirmar carregamento da página com todos os produtos listados.

- **CT02: Ordenar produtos por preço crescente**

Técnica: Caminho alternativo.

Objetivo: Verificar ordenação correta dos produtos do menor para o maior preço.

- **CT03: Pesquisar produto por nome**

Técnica: Valor limite.

Objetivo: Validar pesquisa por nome retornando resultados relevantes.

- **CT04: Exibir detalhes do produto**

Técnica: Caminho feliz.

Objetivo: Confirmar que a página de detalhes exibe informações corretas do produto selecionado.

História [US-0005] – Meus Pedidos

- **CT01: Visualizar pedidos após login**

Técnica: Caminho feliz.

Objetivo: Confirmar exibição da lista de pedidos do usuário autenticado.

- **CT02: Visualizar tela sem pedidos**

Técnica: Caminho alternativo.

Objetivo: Verificar mensagem adequada quando não houver pedidos.

- **CT03: Cancelar pedido**

Técnica: Caminho alternativo.

Objetivo: Validar funcionalidade de cancelamento de pedido.

- **CT04: Visualizar detalhes de pedido**

Técnica: Caminho feliz.

Objetivo: Confirmar exibição detalhada de um pedido selecionado.

História [US-0006] – Minha Conta

- **CT01: Acessar detalhes da conta após login**

Técnica: Caminho feliz.

Objetivo: Confirmar acesso às informações do usuário autenticado.

- **CT02: Acessar página Minha Conta sem login**

Técnica: Caminho alternativo.

Objetivo: Validar redirecionamento para a página de login.

- **CT03: Atualizar informações pessoais**

Técnica: Caminho feliz.

Objetivo: Confirmar alteração dos dados pessoais com sucesso.

- **CT04: Alterar senha**

Técnica: Caminho feliz.

Objetivo: Validar processo de mudança de senha.

4.4 Repositório no GitHub

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

4.5 Testes Automatizados

Os testes automatizados foram divididos em três frentes: interface web (UI), API e mobile. A escolha das ferramentas considerou maturidade, comunidade ativa e facilidade de integração com outras partes do projeto, como GitHub Actions e geração de relatórios.

3.5.1 Automação de UI

Para a automação de testes da interface web, foi utilizado o Cypress com JavaScript, por sua simplicidade, documentação clara e excelente suporte à inspeção de elementos. Outras opções analisadas foram Selenium e Playwright. O Cypress foi escolhido por permitir testes rápidos e fáceis de manter, além de integração nativa com ferramentas de CI/CD.

Os testes foram organizados na pasta **UI**, utilizando o padrão **Page Object** para maior reutilização e clareza. Foram automatizados os seguintes cenários:

- Adicionar item ao carrinho (variação XL e cor verde)
- Adicionar dois itens iguais ao carrinho
- Adicionar e remover item do carrinho
- Acesso ao catálogo de produtos
- Ordenação de produtos por preço
- Login com credenciais válidas
- Login com e-mail válido e senha inválida
- Visualização de pedidos após login
- Nenhum pedido realizado
- Acesso aos detalhes da conta após login
- Tentativa de acesso sem login

3.5.2 Automação de API

Os testes de API foram implementados com a biblioteca Supertest, integrando requisições REST ao WooCommerce. Os testes estão organizados na pasta **API** e validam os endpoints de cupom da loja, cobrindo o ciclo completo: criação, atualização e consulta de cupons. Foram automatizados os seguintes casos:

- Criar cupom com nome aleatório
- Atualizar valor do cupom
- Consultar dados de um cupom existente

Esses testes também validam os contratos (status code, campos obrigatórios) e utilizam autenticação básica conforme exigido pelo endpoint.

3.5.3 Automação Mobile

Para os testes mobile, foi utilizado o Appium com JavaScript na plataforma Android, focando exclusivamente na funcionalidade de catálogo de produtos, conforme escopo proposto. O cenário automatizado consiste em:

- Visualizar lista de produtos carregada corretamente (caminho feliz)
- Rolar a lista para carregar mais produtos (caminho alternativo)

O padrão **Page Object** também foi aplicado para os testes mobile. A estrutura de automação está organizada na pasta **Mobile**.

Todos os testes automatizados geram relatórios, e foram configurados para rodar de forma contínua através do GitHub Actions, garantindo integração ao ciclo de desenvolvimento.

4.6 Integração Contínua

A automação dos testes foi integrada ao GitHub Actions, permitindo a execução automática sempre que houver um push ou pull request na branch main. O pipeline é definido no arquivo `.github/workflows/testes.yml`.

A configuração utiliza um runner Ubuntu com Node.js versão 18 e passa por todas as etapas essenciais para garantir a execução contínua dos testes de API e UI, conforme abaixo:

- Clonagem do repositório (`actions/checkout@v3`)
- Configuração do ambiente Node.js (`actions/setup-node@v3`)
- Instalação das dependências via `npm install`
- Correção de permissões do Jest
- Início de dois containers Docker: banco de dados (lojaebacdb) e WordPress com WooCommerce pré-instalado (lojaebac)
- Aguardo de 30 segundos para garantir que os containers estejam prontos
- Execução dos testes de API com o comando `npm run test:api`
- Execução dos testes de UI com o Cypress em modo headless via `npm run test:ui`

As credenciais para autenticação da API (USERNAME e PASSWORD) são armazenadas de forma segura nos secrets do repositório, garantindo execução isolada e segura.

4.7 Testes de Performance

Para avaliar o desempenho da API do WordPress com WooCommerce, foi utilizado o K6, ferramenta de teste de carga baseada em JavaScript. O script realiza um teste de estresse em três etapas:

1. Rampa: aumento progressivo de usuários até 20 usuários virtuais em 20 segundos;
2. Manutenção: mantém 20 usuários ativos por 1 minuto e 40 segundos;
3. Desligamento: redução gradual para 0 usuários em 30 segundos.

Durante a execução, o script realiza dois tipos de requisições:

- Autenticação JWT: envio de usuário e senha para obter token via `/jwt-auth/v1/token`;
- Consulta de posts: requisição autenticada ao endpoint `/wp/v2/posts` para simular operação real.

O teste valida códigos de status 200, sucesso na autenticação e resposta correta do endpoint de posts.

O uso do K6 permitiu analisar a escalabilidade e consistência da API sob carga, assegurando desempenho aceitável próximo ao ambiente de produção.

5. CONCLUSÃO

O presente trabalho demonstrou, na prática, como aplicar técnicas de Garantia da Qualidade de Software em um ambiente de e-commerce utilizando ferramentas modernas e acessíveis. Através da criação e execução de testes manuais e automatizados (UI, API, Mobile), integração contínua com GitHub Actions e análise de performance com K6, foi possível validar a estabilidade, funcionalidade e desempenho da aplicação EBAC Shop.

A estrutura do projeto foi organizada com foco na clareza, reprodutibilidade e escalabilidade dos testes. Além disso, a separação por histórias de usuário permitiu uma cobertura mais direcionada e objetiva dos cenários reais enfrentados por usuários da loja.

Com este estudo, conclui-se que é plenamente viável implementar uma estratégia de qualidade robusta mesmo em projetos de menor porte, utilizando ferramentas gratuitas e práticas ágeis. A experiência adquirida reforça a importância da automação e da integração contínua como pilares para garantir entregas de software confiáveis e eficientes.

6. REFERÊNCIAS BIBLIOGRÁFICAS

EBAC – Escola Britânica de Artes Criativas e Tecnologia. Curso de Engenharia de Qualidade de Software. São Paulo, 2025. Disponível em: https://ebaconline.com.br/. Acesso em: 15 jul. 2025.

EBAC – Escola Britânica de Artes Criativas e Tecnologia. Módulo 8: Introdução a testes automatizados. Curso de Engenharia de Qualidade de Software. São Paulo, 2025.

EBAC – Escola Britânica de Artes Criativas e Tecnologia. Módulo 20: Testes de performance com K6. Curso de Engenharia de Qualidade de Software. São Paulo, 2025.

EBAC – Escola Britânica de Artes Criativas e Tecnologia. Módulo 22: Testes de API com Jest e Supertest. Curso de Engenharia de Qualidade de Software. São Paulo, 2025.

EBAC – Escola Britânica de Artes Criativas e Tecnologia. Módulo 24: Testes de interface com Cypress. Curso de Engenharia de Qualidade de Software. São Paulo, 2025.

EBAC – Escola Britânica de Artes Criativas e Tecnologia. Módulo 25: Testes mobile com Appium. Curso de Engenharia de Qualidade de Software. São Paulo, 2025.

EBAC – Escola Britânica de Artes Criativas e Tecnologia. Módulo 26: Integração Contínua com GitHub Actions. Curso de Engenharia de Qualidade de Software. São Paulo, 2025.

MARTIN, Robert C. *Código limpo: habilidades práticas do Agile software*. 1. ed. Rio de Janeiro: Alta Books, 2009.