



ENGENHEIRO DE QUALIDADE DE SOFTWARE

Rafael Morais Martins

Análise de Qualidade

São Paulo

2024

1. RESUMO

Este projeto explora a aplicação prática de testes de API, testes de UI e testes de performance, utilizando as técnicas aprendidas ao longo do curso de Engenheiro de Qualidade de Software da EBAC. O objetivo principal é integrar teoria e prática, permitindo uma compreensão mais profunda do conteúdo abordado.

Ao longo do desenvolvimento do projeto, foram seguidas etapas detalhadas que refletem o dia a dia de um profissional de QA no mercado de trabalho. Cada fase do processo de teste foi cuidadosamente planejada e executada, garantindo a aplicabilidade das metodologias aprendidas.

Além disso, o projeto possibilitou a realização de testes em um ambiente real, onde foi possível identificar e resolver problemas, validar funcionalidades e assegurar a qualidade do software. A experiência adquirida não apenas reforçou o conhecimento técnico, mas também desenvolveu habilidades essenciais, como análise crítica e resolução de problemas.

Por meio dessa abordagem prática, o projeto contribui para a formação de um engenheiro de qualidade de software mais preparado para enfrentar os desafios do mercado, capacitando-o a garantir a excelência em produtos digitais.

SUMÁRIO

2.

1. RESUMO	2
2. SUMÁRIO	3
3. INTRODUÇÃO	4
4. O PROJETO	5
4.1 Estratégia de teste	6
4.2 Critérios de aceitação	6
4.3 Casos de testes	6
4.4 Repositório no Github	7
4.5 Testes automatizados	7
4.6 Integração contínua	8
4.7 Testes de performance	8
5. CONCLUSÃO	9
6. REFERÊNCIAS BIBLIOGRÁFICAS	9

3. INTRODUÇÃO

O Quality Assurance (QA) desempenha um papel crucial na entrega de software de alta qualidade, contribuindo para a satisfação do cliente e a reputação da empresa. Por meio de uma combinação de testes manuais e automatizados, um profissional de QA assegura que o produto final seja robusto, seguro e funcional. Este projeto tem como objetivo trazer uma visão integrada da prática e da teoria na área de QA, destacando as qualidades técnicas e teóricas que um bom profissional deve possuir.

O projeto foi dividido em várias partes, começando com uma análise do software e da API testados. Nessa etapa, foram utilizados códigos em Gherkin e mindmaps para a elaboração da estratégia de teste, permitindo uma abordagem clara e organizada. A segunda parte envolve a execução de testes de API e UI, utilizando ferramentas como Supertest e Cypress, que facilitam a automação e a verificação das funcionalidades.

Na sequência, foi realizada uma análise de performance do software, aplicando testes com o K6 para avaliar a capacidade de resposta e a eficiência do sistema sob diferentes condições. Por fim, a implementação do GitHub Actions no projeto possibilitou a automação dos testes, garantindo um fluxo contínuo de integração e entrega.

Este projeto não apenas evidencia a importância do QA, mas também demonstra a aplicabilidade das técnicas e ferramentas aprendidas ao longo da formação na área.

4. 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/r/ernestosbarbosa/lojaebacdb)
 - Loja EBAC: [ernestosbarbosa/lojaebac](https://hub.docker.com/r/ernestosbarbosa/lojaebac)
- Comandos para subir os containers:

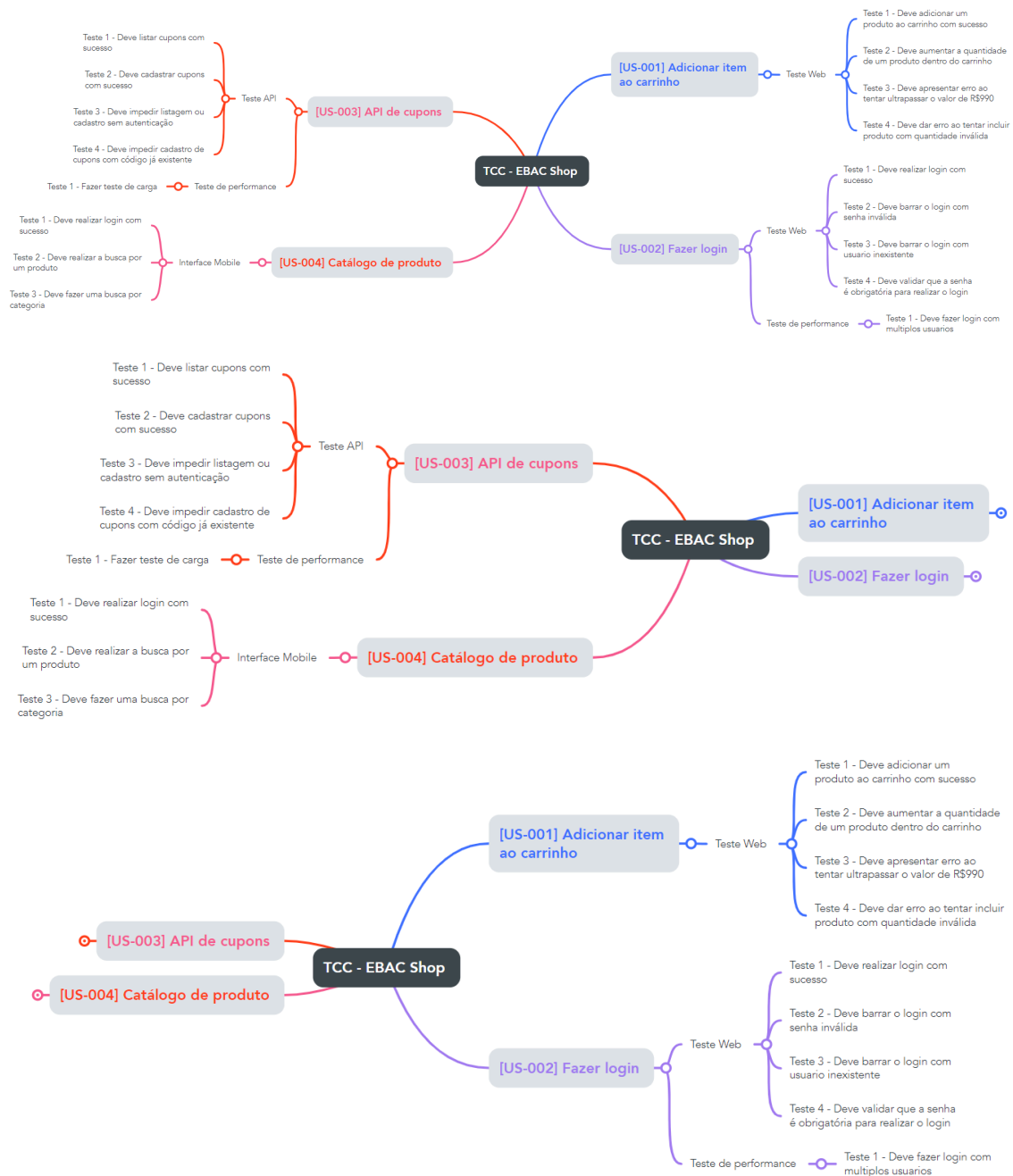
```
docker network create --attachable ebac-network  
  
docker run -d --name wp_db -p 3306:3306 --network ebac-network ernestosbarbosa/lojaebacdb:latest  
  
docker run -d --name wp -p 80:80 --network ebac-network ernestosbarbosa/lojaebac:latest
```

Após subir os containers a loja estará em <http://localhost:80>

- 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.

4.1 Estratégia de teste

- Faça uma estratégia de testes em um mapa mental, seguindo algumas diretrizes como objetivos, papéis 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:



4.2 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

4.3 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

4.4 Repositório no Github

- Crie um repositório no github com o nome TCC-EBAC-QE;
- Deixe o repositório público 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/RafaelMMorais/EBAC-TCC-QE.git>

4.5 Testes automatizados

4.5.1 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.

4.5.2 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! 😊

4.5.3 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

4.6 Integração contínua






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

4.7 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

<input type="checkbox"/> Nome de usuário	Nome	E-mail	Função
<input type="checkbox"/>  user1_ebac	—	user1_ebac@ebac.com	Assinante
<input type="checkbox"/>  user2_ebac	—	user2_ebac@ebac.com	Assinante
<input type="checkbox"/>  user3_ebac	—	user3_ebac@ebac.com	Assinante
<input type="checkbox"/>  user4_ebac	—	user4_ebac@ebac.com	Assinante
<input type="checkbox"/>  user5_ebac	—	user5_ebac@ebac.com	Assinante

5. CONCLUSÃO

Com este trabalho, obtive uma melhora significativa no meu conhecimento sobre o Cypress, uma ferramenta fundamental para a automação de testes em aplicações web. Algumas dificuldades que enfrentei ao longo do curso, especialmente em relação aos testes de performance, foram superadas durante a elaboração do TCC. O projeto apresenta de forma detalhada o uso do Cypress na automação de testes, abrangendo não apenas a interface do usuário (UI), mas também a interação com APIs, e destaca a importância de um profissional de QA no desenvolvimento de software.

Um dos aspectos mais enriquecedores dessa experiência foi a oportunidade de aplicar teorias e práticas aprendidas ao longo do curso. O projeto não apenas me permitiu explorar profundamente o Cypress, mas também me desafiou a implementar técnicas de teste que inicialmente pareciam complexas. Embora diversas dificuldades tenham permanecido ao final do TCC, acredito firmemente que meus estudos estão apenas começando. Estou confiante de que poderei superar cada um desses desafios ao longo da minha carreira, utilizando as habilidades adquiridas durante este processo.

É importante ressaltar o papel fundamental dos professores na minha formação. Eles foram essenciais na minha jornada, me guiando em escolhas assertivas de tecnologias e técnicas de teste. Graças a suas orientações, consegui desenvolver uma visão mais clara sobre a importância da qualidade no desenvolvimento de software e a necessidade de integrar práticas de QA desde as fases iniciais de um projeto.

A escolha do Cypress, por exemplo, mostrou-se a melhor opção para o projeto de UI e API. Sua principal vantagem é a capacidade de realizar testes tanto na interface quanto nas APIs, proporcionando uma ampla gama de possibilidades para testar em uma única ferramenta. Essa abordagem não apenas otimiza o processo de teste, mas também diminui o tempo e o tamanho do projeto, sem comprometer o alcance e a eficácia dos testes realizados.

Além disso, destaco a escolha do K6 como ferramenta para testes de performance. Com ele, foi possível realizar não apenas o teste de carga

solicitado, mas também testes de spike, estresse e outros tipos de avaliação de performance. Essas práticas são essenciais para garantir que o software não apenas funcione sob condições normais, mas também mantenha sua integridade e desempenho sob pressão, algo que é crucial para a experiência do usuário.

Em suma, este projeto foi de grande auxílio para o aprendizado adquirido ao longo do curso e colocou em prática uma parte significativa dos conhecimentos teóricos e práticos desenvolvidos. Estou animado para continuar explorando o campo de QA e aplicar o que aprendi em projetos futuros, contribuindo para a criação de software de alta qualidade.

6. REFERÊNCIAS BIBLIOGRÁFICAS

CYPRESS.IO. Cypress Documentation. Disponível em: <<http://docs.cypress.io>>. Acesso em: 29 out. 2024.

W3C. HTML Living Standard. Disponível em: <<http://api.jquery.com>>. Acesso em: 29 out. 2024.

NPM. npm Documentation. Disponível em: <<http://docs.npmjs.com>>. Acesso em: 30 out. 2024.

NODE.JS. Node.js Documentation. Disponível em: <<http://nodejs.org/en/docs>>. Acesso em: 30 out. 2024.