



ENGENHEIRO DE QUALIDADE DE SOFTWARE

LUIZ GUILHERME MENON

ANÁLISE DE QUALIDADE DE SOFTWARE

IRATI

2024

## 1. RESUMO

Este trabalho teórico e prático tem como objetivo contemplar diversos assuntos chave abordados no decorrer do curso profissionalizante “Engenheiro de Qualidade *Software*” fornecido pela EBAC (Escola Britânica de Artes Criativas & Tecnologia).

Com o curso obtive como pretendia uma visão mais ampla da área de Qualidade na qual atuo, aprendendo diversas técnicas como a elaboração eficaz de casos de teste, o BDD (*Behavior Driven Development*) e a integração contínua, além de diversos tipos de teste, tais como os de performance, os de segurança e os automatizados de *back-end* e *front-end*.

Neste trabalho serão descritas importantes etapas que podem fazer parte da rotina de um profissional de Qualidade durante o ciclo de desenvolvimento, contribuindo assim para a entrega de produtos confiáveis e com boa experiência de utilização, fatores estes cada vez mais desejáveis pelo mercado de TI (Tecnologia da Informação).

## 2. SUMÁRIO

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

## 2. INTRODUÇÃO

A área de Qualidade de *Software* vem ganhando cada vez mais relevância no mercado de TI, em razão do impacto positivo que pode trazer para diversos projetos, evitando falhas e consequentemente, prejuízos financeiros ou relacionados à reputação (LENILDO, 2010). Desta forma, contribui para a entrega de produtos confiáveis e com uma boa experiência de utilização por parte dos usuários.

O profissional desta área pode atuar desde o processo da análise de desenvolvimentos, conforme o princípio do *Shift Lefting Testing*, levantando pontos de atenção para elucidar nas documentações envolvidas, tais como escopo, histórias de usuário, critérios de aceitação, entre outras. Deste modo, a partir de uma documentação precisa, as próximas etapas de desenvolvimento e teste ocorrem com uma fluidez melhor, visto que por exemplo, os casos de teste poderão ser planejados antecipadamente com um maior conhecimento do que será preciso validar em determinado contexto. Após a execução dos casos de teste, também é possível realizar testes exploratórios a partir das evidências coletadas de forma a tentar encontrar algum comportamento diferente do esperado. À medida que o processo de teste avança, padrões de desenvolvimento também tendem a evoluir.

Somado a isso, os testes automatizados podem ser úteis para validar se novos desenvolvimentos não interferiram em alguma funcionalidade em operação na versão anterior, caracterizando um teste de regressão neste caso. Vale notar que estes testes automatizados podem ser executados através da integração contínua presente no ambiente *DevOps*, a qual possibilita configurar a execução automática destes testes a partir de um evento pré-definido.

Cabe ressaltar que o processo de teste, inclusive em um espectro mais amplo referente a pirâmide de testes, não assegura que um sistema não possua inconsistências, porém reduz consideravelmente a probabilidade de

que falhas críticas ocorram, salientando que em equipes que adotam metodologias ágeis como *Scrum* e *Scrumban*, a responsabilidade na entrega de um sistema de qualidade recai sobre todos os participantes envolvidos no projeto, conforme explicado na imagem a seguir.

Figura 1 - O Manifesto do Teste Ágil



Fonte: Growing Agile (2016).

Portanto, de certo ponto de vista, o profissional de QA (*Quality Assurance*) auxilia a conciliar a análise, o desenvolvimento e o usuário final, propondo soluções viáveis para as três partes através de um pensamento crítico e questionador com um olhar focado nos detalhes e nas possíveis situações que possam causar inconsistências, inclusive em funcionalidades já existentes (CTFL, 2023). A ISO/IEC (*International Organization for Standardization/International Electrotechnical Commission*) 25010 categoriza alguns pilares básicos para a avaliação de um sistema, sendo eles: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade (VOLPATO, 2017).

Este trabalho visa explicar nas próximas seções, cenários práticos de aplicação de um QA, englobando estratégias de teste, testes automatizados *web*, *mobile* e de API (*Application Programming Interface*), contando inclusive com integração contínua e testes de performance, consolidando assim boa parte dos conhecimentos adquiridos durante o curso “Engenheiro de Qualidade de Software” da EBAC.

### 3. 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.ebaonline.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:
  - o Banco de Dados: [ernestosbarbosa/lojaebacdb](https://hub.docker.com/r/ernestosbarbosa/lojaebacdb)
  - o 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
```

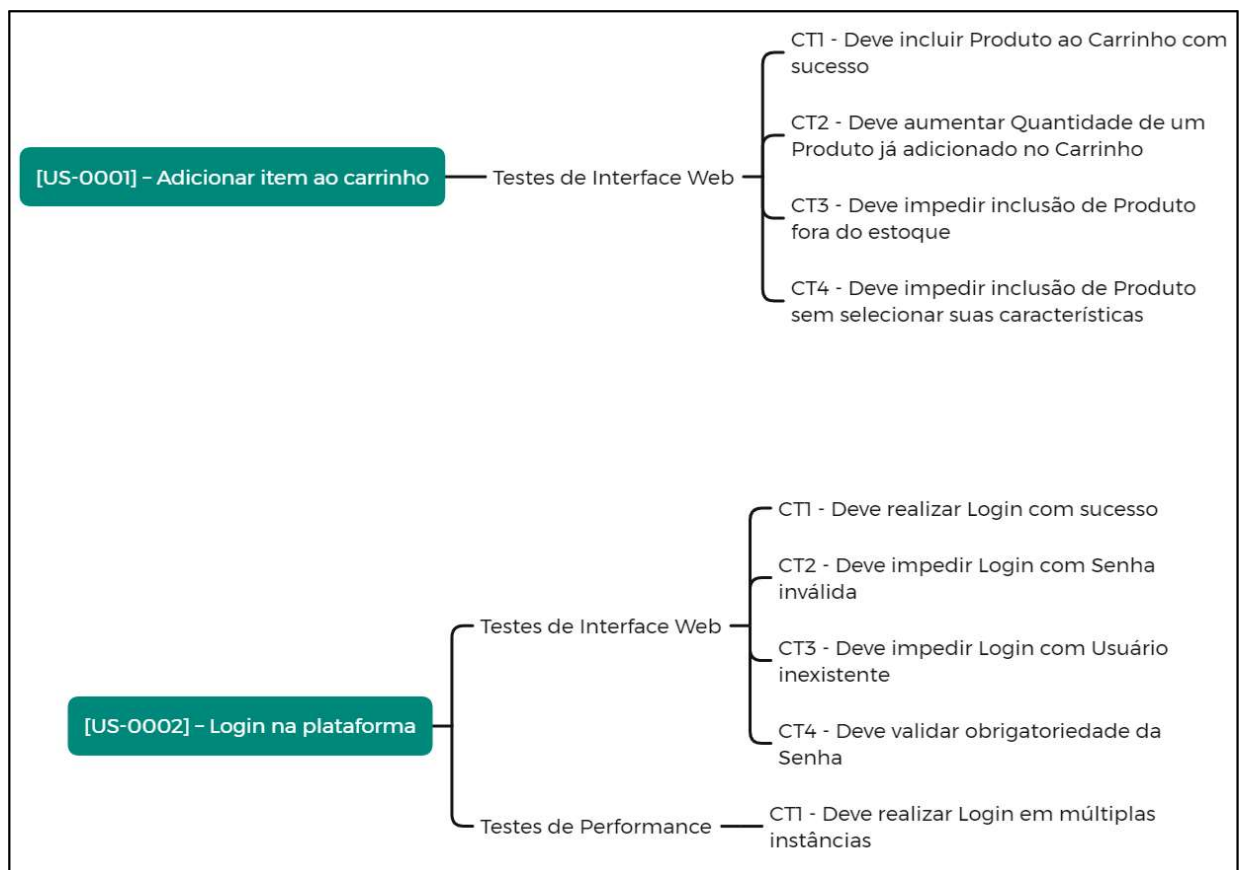
Após subir os containers a loja estará em <http://localhost:80>. Localmente o bd funcionou na porta 3316.

- 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.

### 3.1 Estratégia de teste

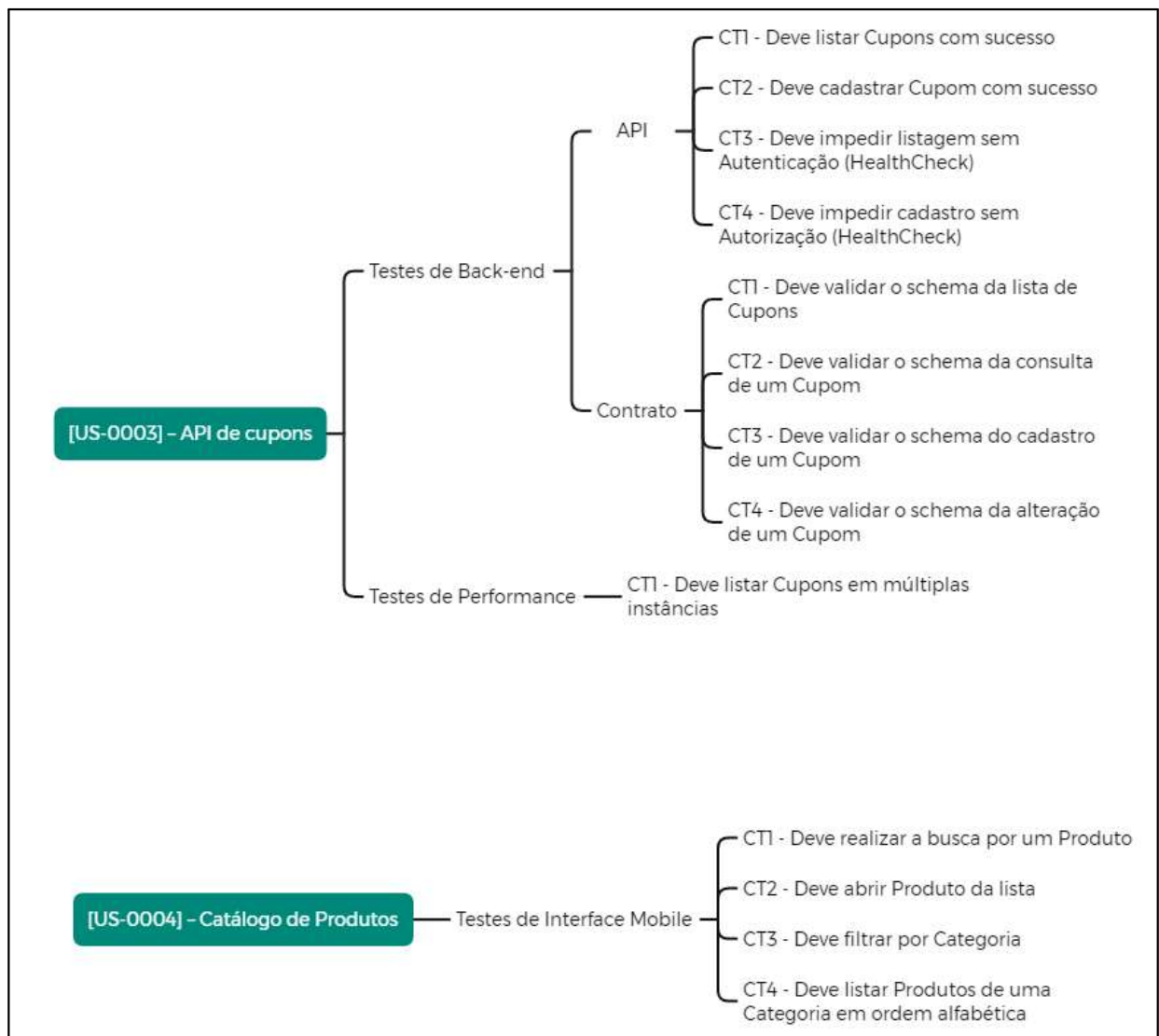
- Faça uma estratégia de testes em um mapa mental, seguindo algumas diretrizes como objetivos, papéis e responsabilidades, fases de testes, padrões, tipos de testes, técnicas de testes, ambientes, ferramentas, abordagem (manual ou automatizado), framework ou ferramenta usados, plataformas (web, api, mobile), etc.;
- Referência: Módulo 5
- Após fazer sua estratégia de teste, tire um print e cole aqui (mapa elaborado utilizando o *software XMind*):

Figura 2 - Mapa mental dos testes implementados



Fonte: o autor (2024).

Figura 3 - Mapa mental dos testes implementados



Fonte: o autor (2024).

### 3.2 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.



o [US-0001] – Adicionar item ao carrinho

o CT1 - Deve incluir Produtos ao carrinho com sucesso

Dado

Que se está na tela ou no modal de um Produto

Quando

Selecionar as variações do Produto, se disponíveis

E preencher a Quantidade de Itens

E clicar no botão Comprar

Então

Espero que seja possível adicionar o Produto ao carrinho se houver Estoque para a Quantidade selecionada

E espero que menu superior do carrinho seja atualizado trazendo todos os produtos presentes no mesmo

o CT2 - Deve aumentar Quantidade de um Produto já adicionado no Carrinho

Dado

Que se está na tela Carrinho

E que existem um ou mais Produtos vinculados ao mesmo

Quando

Aumentar a Quantidade de itens de um Produto

Então

Espero que os valores do Carrinho sejam atualizados com sucesso

o CT3 - Deve impedir inclusão de Produto fora do estoque

Dado

Que se está na tela ou modal de um Produto que possui o alerta Fora de Estoque

Quando

Clicar no botão já desabilitado Comprar

Então

Espero que seja retornada uma mensagem de alerta

- o CT4 - Deve impedir inclusão de Produto com Quantidade inválida

Dado

Que se está na tela ou modal de um Produto ou no Carrinho com itens

Quando

Preencher uma Quantidade maior que a disponível em Estoque ou preencher uma Quantidade inválida

Então

Espero que seja retornada a mensagem explicando cada caso ao usuário

E espero que não seja possível adicionar ao carrinho ou seguir para o *Checkout*

- o [US-0002] – *Login* na plataforma

- o CT1 - Deve realizar *Login* com sucesso

Dado

Que se está na tela de *Login*

Quando

Preencher um *E-mail* e Senha válidos  
E clicar no botão *Login*

Então

Espero que seja redirecionado para a tela Minha Conta

- o CT2 - Deve impedir *Login* com Usuário inativo

Dado

Que se está na tela de *Login*

Quando

Preencher um *E-mail* e Senha de um Usuário com Situação Inativa  
E clicar no botão *Login*

Então

Espero que seja exibida uma mensagem de alerta  
E que não seja redirecionado a área logada

- o CT3 - Deve impedir *Login* com credenciais inválidas

Dado

Que se está na tela de *Login*

Quando

Preencher um *E-mail* ou Senha inválidos ou inexistentes

Então

Espero que seja exibida uma mensagem de alerta

E que não seja redirecionado a área logada

- o CT4 - Deve impedir *Login* por 15 minutos após 3 tentativas de acesso

Dado

Que foram realizadas 3 tentativas inválidas de *Login* com um *E-mail*

E que se está na tela de *Login*

Quando

Preencher o *E-mail* citado

E preencher a Senha válida

E clicar no botão *Login*

Então

Espero que seja exibida uma mensagem de alerta

explicando que o acesso foi bloqueado por 15 minutos por questões de segurança

E que não seja redirecionado a área logada neste período

- o [US-0003] – API de cupons

- o CT1 - Deve listar um ou mais Cupons com sucesso

Dado

Que a requisição *Get* para listar Cupons foi configurada corretamente em um *API Client*, conforme *Swagger*

E que foi preenchido um *token* válido e com permissão de Cupons

Quando

Enviar a requisição

Então

Espero que os Cupons correspondentes aos dados de envio sejam retornados

E que seja retornado o *Status 200 Ok*

- o CT2 - Deve cadastrar Cupons com sucesso

Dado

Que a requisição *Post* para cadastrar Cupons foi configurada corretamente em um *API Client*, conforme *Swagger*

E que foi preenchido um token válido e com permissão de Cupons

Quando

Preencher todos os dados obrigatórios: Código do Cupom (não duplicado), Valor, Tipo do Desconto, Descrição

E enviar a requisição

Então

Espero que o Cupom seja cadastrado com sucesso

E que seja retornado o *Status 200 Ok*

- o CT3 - Deve impedir listagem ou cadastro sem Autenticação ou sem Autorização

Dado

Que as requisições *Get* e *Post* de Cupons foram configuradas corretamente em um *API Client*, conforme *Swagger*

E que o *header Authorization* está vazio ou com *token* inválido ou sem permissão ao recurso de Cupons

Quando

Enviar a requisição

Então

Espero que seja retornado algum dos *Status 401* (sem autorização) ou *403* (sem permissão)

- o CT4 - Deve restringir cadastro de Cupons com Códigos já existentes

Dado

Que a requisição *Post* para cadastrar Cupons foi configurada corretamente em um *API Client*, conforme *Swagger*

E que foi preenchido um token válido e com permissão de Cupons

E que existe um Cupom com Código “Teste”

Quando

Preencher o Código do Cupom com o nome “Teste”

E preencher os demais dados obrigatórios: Valor, Tipo do

Desconto, Descrição

E enviar a requisição

Então

Espero que seja retornado o *Status* 400 com uma *Response* explicando sobre a duplicidade

- Crie histórias de usuário para as funcionalidades:

- o 1. Catálogo de Produtos

Como usuário da *EBAC-SHOP*

Quero visualizar Produtos e seus valores

Para selecionar quais desejo adicionar ao Carrinho

Regras de Negócio:

- Deve listar Produtos Ativos mais relevantes por padrão
- Deve permitir ordenação por valor crescente e decrescente
- Deve permitir filtrar listagem por Categoria de Produtos

- o 2. Painel Minha Conta

Como cliente da *EBAC-SHOP*

Quero visualizar meus dados em um painel

Para conferir o histórico de pedidos ao atualizar dados cadastrais

Regras de Negócio:

- Deve exibir opção para visualizar histórico de pedidos
- Deve exibir opção para visualizar dados do cadastro do Usuário

- o 3. Meus Pedidos

Como cliente da *EBAC-SHOP*

Quero visualizar meus Pedidos

Para verificar se os dados estão corretos

Regras de Negócio:

- Deve listar o histórico de Pedidos a partir dos mais recentes
- Deve permitir cancelar o Pedido

o 4. Endereços

Como cliente da *EBAC-SHOP*

Quero visualizar meus Endereços de Cobrança e de Entrega

Para verificar se os dados estão corretos ou se precisam ser atualizados

Regras de Negócio:

- Deve listar meus Endereços de Cobrança e de Entrega
- Deve permitir atualizar dados com CEP válido

o 5. Detalhes da Conta

Como cliente da *EBAC-SHOP*

Quero visualizar meus dados cadastrais

Para verificar se os dados estão corretos ou se precisam ser atualizados, além de possibilitar a redefinição de senha

Regras de Negócio:

- Deve listar dados cadastrais e pedidos do Usuário

- Referência: Módulo 8

### 3.3 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

- o [US-0001] – Adicionar item ao carrinho
  - o 1. Deve incluir Produto ao carrinho com sucesso
  - o 2. Deve aumentar Quantidade de um Produto já adicionado no Carrinho
  - o 3. Deve impedir inclusão de Produto fora do estoque
  - o 4. Deve impedir inclusão de Produto sem selecionar suas características
  
- o [US-0002] – Login na plataforma
  - o 1. Deve realizar Login com sucesso
  - o 2. Deve impedir Login com Senha inválida
  - o 3. Deve impedir Login com Usuário inexistente
  - o 4. Deve validar obrigatoriedade da Senha
  - o 5. Deve realizar Login em múltiplas instâncias
  
- o [US-0003] – API de cupons
  - o 1. Deve listar Cupons com sucesso
  - o 2. Deve cadastrar Cupom com sucesso
  - o 3. Deve impedir listagem sem Autenticação
  - o 4. Deve impedir cadastro sem Autorização
  - o 5. Deve validar o schema da lista de Cupons
  - o 6. Deve validar o schema da consulta de um Cupom
  - o 7. Deve validar o schema do cadastro de um Cupom
  - o 8. Deve validar o schema da alteração de um Cupom
  - o 9. Deve listar Cupons em múltiplas instâncias
  
- o [US-0004] – Catálogo de Produtos
  - o 1. Deve realizar a busca por um Produto
  - o 2. Deve abrir Produto da lista
  - o 3. Deve filtrar por Categoria
  - o 4. Deve listar Produtos de uma Categoria em ordem alfabética

### **3.4 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/Lewiz-QA/TCC-EBAC-QE.git>

### 3.5 Testes automatizados

#### 3.5.1 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.
  - o Os testes foram organizados utilizando *PageObjects* e *AppActions* via comandos customizados. Foram utilizadas *fixtures* como massa de dados. Vale notar que cada teste foi desenvolvido para ser independente, seguindo a lógica do *BDD*, além de ter sido utilizada a linguagem *Gherkin*.

Foi decidido utilizar a ferramenta *Cypress* para os testes automatizados *web* de interface, fundamentando-se nos seguintes fatores:

- Possibilidade de automatização de testes de interface e de *back-end* em uma única plataforma;
- Ampla popularidade da linguagem JS (*JavaScript*), o que facilita em ajustes ou na implementação de testes avançados;
- Bom suporte da ferramenta, o qual conta com uma comunidade para esclarecer dúvidas;
- Utilização de massa de dados para testes;
- Suporte a integração contínua;
- Possibilidade de gravação dos testes;
- Possibilidade de execução em segundo plano para melhor desempenho;
- Diversas possibilidades de Relatórios. No projeto foi utilizado o *Allure*;
- Diversos *plugins* como o *Cucumber* e o *faker*.



Contudo, outras ferramentas também poderiam ser utilizadas neste contexto, tais como *Selenium*, *Playright* ou *Katalon*, embora possivelmente não entregassem a mesma praticidade do *Cypress*, além de que o mesmo possui diferenciais como os testes de componentes gráficos e a interceptação de *requests*, fatores estes que o tornam uma ferramenta atrativa pensando em prosseguir o projeto no futuro, o qual por sinal utiliza JS para os testes *mobile*, testes de performance e testes de *back-end*, facilitando assim na codificação de novos testes.

Deste modo, pode-se dizer que o *Cypress* atende bem ao projeto proposto, especialmente por possibilitar diversos tipos de teste, além de possuir uma ótima curva de aprendizado frente ao prazo de conclusão do curso.

### 3.5.2 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! ☺
- Utilizado o relatório *Jest*.

### 3.5.3 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/i-os-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.

- o Foram utilizadas *views* para mapear os elementos que interagem com os testes, além de *fixtures* como massa de dados.
- Você deve implementar testes para ao menos uma das plataformas Mobile (*Android* ou *iOS*)
  - o Testes implementados para *iOS*.
- Observações:
  - o Considere todas as boas práticas aprendidas até aqui
  - o Não esqueça de implementar a geração de relatórios
    - Utilizado relatório *Allure*.
- Referência: Módulos 11, 12, 14, 16, 17, 22, 23, 24, 29 e 30

### 3.6 Integração contínua

- Execute os testes automatizados em integração contínua utilizando o Github Actions
  - o Executados os testes *Mobile* em integração contínua com o *GitHub Actions*, de modo que ao enviar um *commit* para o *branch* “*ci*”, automaticamente os testes serão executados no *BrowserStack* a partir da chamada realizada pelo *GitHub Actions*, garantindo assim que a nova versão não ocasionou inconsistências. No *branch* “*gh-pages*”, será possível visualizar o relatório gerado pelo *Allure*. Foi gravado um vídeo no repositório apresentando este procedimento.
- Referência: Módulo 26

### 3.7 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

- Utilizado o Grafana para gerar gráficos dos testes de Performance, o qual pode ser executado via *Docker*.

#### 4. CONCLUSÃO

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

O trabalho se mostrou desafiante e intenso desde seu início, requerendo muita perseverança, no qual foi necessário aplicar na prática e discorrer sobre diversos conhecimentos adquiridos durante o curso em contextos distintos, além de diversas pesquisas relacionadas, o que contribuiu para reforçar ainda mais as bases teóricas e práticas abordadas no projeto.

O primeiro passo para a elaboração deste trabalho foi o planejamento do que precisaria ser testado, levantando casos de teste relevantes para cada contexto de acordo com as estratégias estudadas ao longo do curso. Optei por validar alguns cenários de sucesso, de falha, de segurança e de performance, dada a baixa quantidade de casos de teste proposta para este trabalho e ao prazo de conclusão do curso, já que poderiam ser desenvolvidos mais testes neste projeto. Contudo, o trabalho dispõe de um panorama geral de conceitos importantes e aplicáveis no mercado de teste de *software*, dispondo de vinte e dois casos de testes como exemplo.

Em síntese, este trabalho representa a consolidação de valiosas habilidades conquistadas durante o curso “Engenheiro de Qualidade de *Software*” da EBAC, muitas das quais impulsionaram-me a evoluir na carreira profissional e espero que prossiga neste sentido com os fundamentos que aprendi, buscando contribuir na entrega de *softwares* com cada vez mais qualidade.

## 5. REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, Maximiliano. **Testes de schema de API com Joi**. Disponível em: <<https://maximilianoalves.medium.com/testes-de-contrato-de-api-com-joi-1ce552fe2531>>. Acessado em: 1 fev. 2024.

BSTQB. **Certified Tester Foundation Level Syllabus CTFL versão 4.0**. Disponível em: <[https://bcr.bstqb.org.br/docs/syllabus\\_ctfl\\_4.0br.pdf](https://bcr.bstqb.org.br/docs/syllabus_ctfl_4.0br.pdf)>. Acessado em: 2 fev. 2024.

GREAVES, Karen. **Brazilian Portuguese Testing Manifesto**. Disponível em: <<https://www.growingagile.co/brazilian-portuguese-testing-manifesto/>>. Acessado em: 3 fev. 2024.

LENILDO. **Qualidade de Software - Engenharia de Software 29**. Disponível em: <<https://www.devmedia.com.br/qualidade-de-software-engenharia-de-software-29/18209>>. Acessado em: 10 fev. 2024.

VOLPATO, Tiago. **Arquitetura de Software de Qualidade**. Disponível em: <[https://edisciplinas.usp.br/pluginfile.php/3528559/mod\\_resource/content/1/Aula12\\_Qualidade.pdf](https://edisciplinas.usp.br/pluginfile.php/3528559/mod_resource/content/1/Aula12_Qualidade.pdf)>. Acessado em: 11 fev. 2024.