



**ENGENHEIRO DE QUALIDADE DE SOFTWARE**

Victor Zamana Penna

Análise de Qualidade

São Paulo

2024

## **1. RESUMO**

Este trabalho, que combina teoria e prática, tem como propósito explorar diversos temas fundamentais discutidos ao longo do curso profissionalizante de “Engenheiro de Qualidade de Software” oferecido pela EBAC (Escola Britânica de Artes Criativas & Tecnologia).

Durante o curso, adquiri uma visão abrangente da área de Qualidade em que atuo, aprendendo técnicas como a criação eficiente de casos de teste, o desenvolvimento guiado por comportamento (BDD), e a integração contínua. Também me aprofundi em diferentes tipos de testes, incluindo testes de performance, segurança, e testes automatizados de back-end e front-end.

Neste trabalho, serão detalhadas etapas importantes que podem fazer parte da rotina de um profissional de Qualidade durante o ciclo de desenvolvimento. Isso contribui para a entrega de produtos confiáveis e com uma boa experiência de uso, características cada vez mais valorizadas no mercado de TI (Tecnologia da Informação).

## 2. SUMÁRIO

<b>1. RESUMO.....</b>	<b>2</b>
<b>2. SUMÁRIO .....</b>	<b>3</b>
<b>3. INTRODUÇÃO .....</b>	<b>4</b>
<b>4. O PROJETO .....</b>	<b>5</b>
4.1 Estratégia de teste .....	5
4.2 Critérios de aceitação.....	7
4.3 Casos de testes .....	13
4.4 Repositório no Github .....	14
4.5 Testes automatizados .....	14
4.6 Integração contínua .....	16
4.7 Testes de performance.....	17
<b>5. CONCLUSÃO .....</b>	<b>17</b>
<b>6. REFERÊNCIAS BIBLIOGRÁFICAS.....</b>	<b>18</b>

### 3. INTRODUÇÃO

A crescente complexidade dos sistemas de software e a demanda por produtos tecnológicos de alta qualidade têm evidenciado a importância crucial da área de Qualidade de Software. Neste cenário, garantir que o software não apenas atenda às especificações funcionais, mas também ofereça uma experiência de usuário consistente e segura, é fundamental para o sucesso no mercado de TI.

Este trabalho busca proporcionar uma visão abrangente sobre práticas e técnicas essenciais no campo da Qualidade de Software, adquiridas ao longo do curso de “Engenheiro de Qualidade de Software” da EBAC (Escola Britânica de Artes Criativas & Tecnologia). O objetivo principal é explorar os conceitos e metodologias que garantem a entrega de produtos de software confiáveis e de alta performance.

O trabalho será dividido em duas partes principais: uma teórica e outra prática. Na seção teórica, abordaremos as principais técnicas e metodologias aprendidas, como a elaboração eficaz de casos de teste, o desenvolvimento guiado por comportamento (BDD), e a integração contínua. Também discutiremos os diferentes tipos de testes, incluindo testes de performance, segurança, e os automatizados para back-end e front-end.

Na parte prática, descreveremos as etapas críticas do ciclo de desenvolvimento de software que um profissional de Qualidade pode enfrentar. Abordaremos como essas etapas contribuem para a entrega de produtos que não apenas atendem às especificações técnicas, mas também oferecem uma experiência de usuário satisfatória e segura.

Esperamos que este projeto forneça uma visão clara e prática de como as técnicas e ferramentas aprendidas podem ser aplicadas no dia a dia da profissão. Através deste trabalho, pretendemos demonstrar a relevância e a aplicação prática dos conceitos discutidos, e como eles podem impactar positivamente o desenvolvimento de software no atual cenário tecnológico.

## 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:

Figura 1 - Mapa mental dos testes implementados

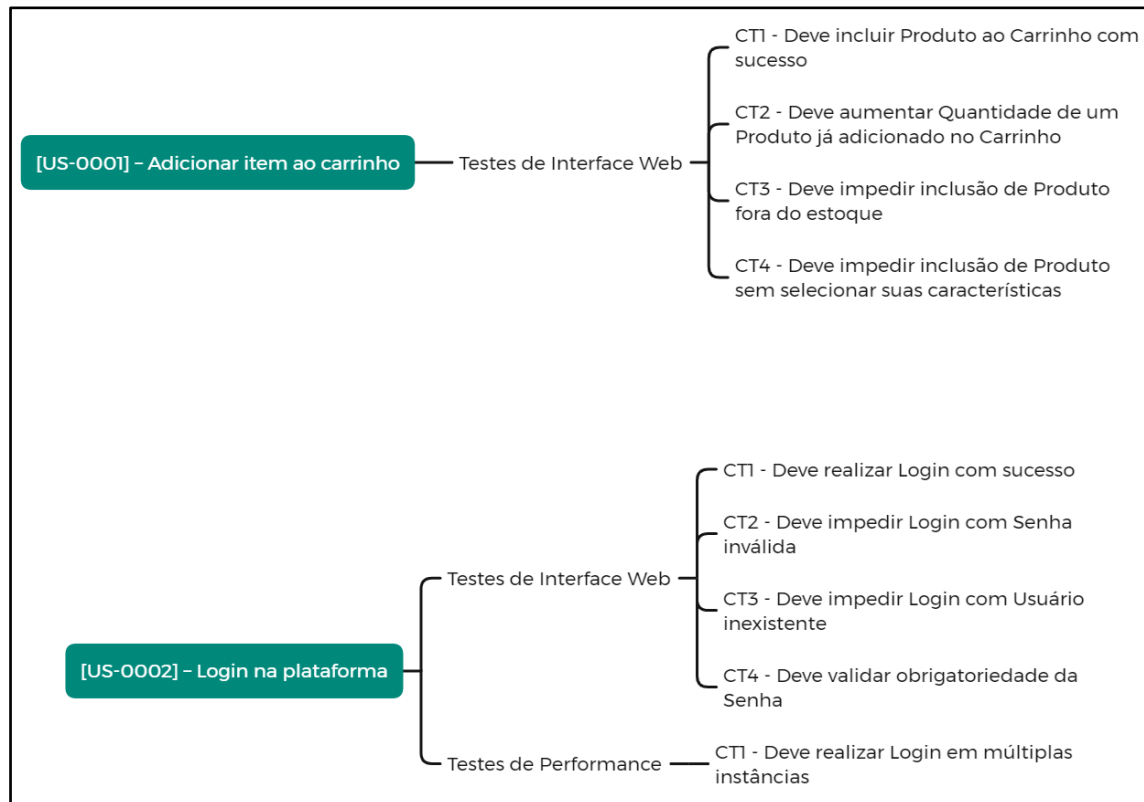
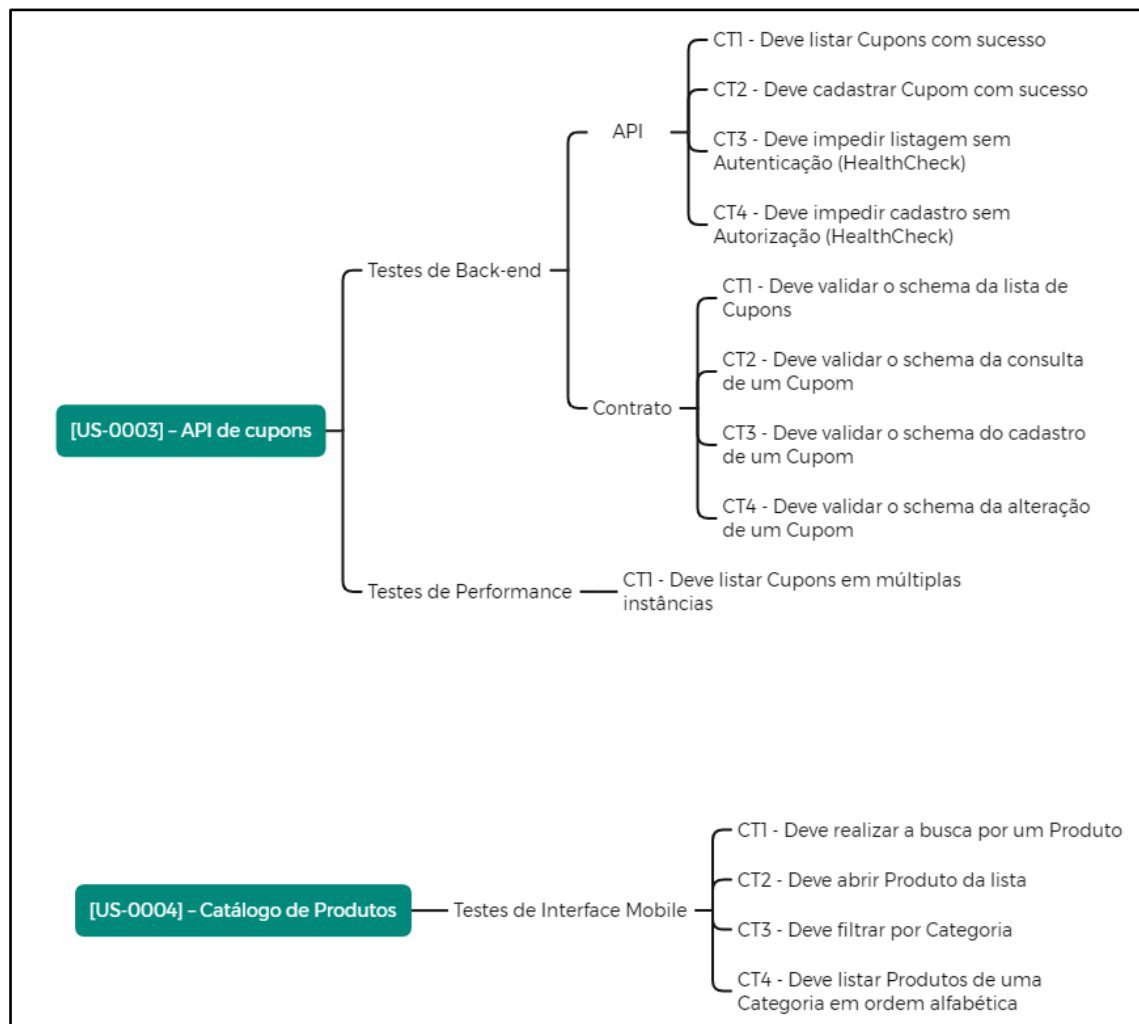


Figura 2 - Mapa mental dos testes implementados



## 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;

**CT1-**

*Cenário: Adicionar produto ao carrinho com sucesso*

Dado

que o usuário está na página do produto

Quando

o usuário seleciona as variações do produto, se disponíveis

E

o usuário preenche a quantidade de itens

E

o usuário clica no botão "Comprar"

Então

o produto deve ser adicionado ao carrinho

**CT2 –**

*Cenário: 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

**CT3 –**

*Cenário: 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

**CT4 –**

*Cenário: 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*

### – **Login na plataforma**

#### **CT1 –**

*Cenário: 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

#### **CT2 –**

*Cenário: 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

#### **CT3 –**

*Cenário: 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

#### **CT4 -**

*Cenário: 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

#### **- API de cupons**

#### **CT1 –**

*Cenário: 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*

#### **CT2 –**

*Cenário: 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*

### **CT3 –**

*Cenário: 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)

### **CT4 –**

*Cenário: 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:

- Catálogo de Produtos
- Painel Minha Conta
- Meus Pedidos
- Endereços
- Detalhes da Conta

#### 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

#### 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

#### 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

#### 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

#### 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

### 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

Adicionar item ao carrinho-

Cenário:

1. Deve incluir Produto ao carrinho com sucesso
2. Deve aumentar Quantidade de um Produto já adicionado no Carrinho
3. Deve impedir inclusão de Produto fora do estoque
4. Deve impedir inclusão de Produto sem selecionar suas características

Login na plataforma

Cenário:

1. Deve realizar Login com sucesso
2. Deve impedir Login com Senha inválida
3. Deve impedir Login com Usuário inexistente
4. Deve validar obrigatoriedade da Senha
5. Deve realizar Login em múltiplas instâncias

API de cupons

Cenário:

1. Deve listar Cupons com sucesso
2. Deve cadastrar Cupom 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 o schema da consulta de um Cupom
7. Deve validar o schema do cadastro de um Cupom
8. Deve validar o schema da alteração de um Cupom
9. Deve listar Cupons em múltiplas instâncias

#### **4.4 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/vzp777/TCC-EBAC-QE>>

#### **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.

Os testes foram organizados utilizando a abordagem de PageObjects e AppActions com comandos personalizados. Além disso, foram empregadas fixtures para a gestão de dados. Cada teste foi desenvolvido para ser autônomo, seguindo os princípios do BDD e utilizando a linguagem Gherkin.

Optou-se pelo Cypress para a automação dos testes de interface web, baseando-se em várias razões:

O Cypress permite a automação tanto de testes de interface quanto de back-end dentro da mesma plataforma. A popularidade da linguagem JavaScript (JS) facilita ajustes e a criação de testes mais avançados. A ferramenta é bem suportada, contando com uma comunidade ativa para suporte e esclarecimento de dúvidas. Utiliza massa de dados para os testes, o que é fundamental para a robustez das verificações. Oferece suporte para integração contínua, permitindo uma melhor integração com processos de desenvolvimento. Possui a capacidade de gravar os testes, o que é útil para revisões e depuração. Permite a execução em segundo plano, melhorando o desempenho dos testes.

Oferece diversas opções de relatórios, com o uso do Allure no projeto para fornecer uma visão clara dos resultados dos testes.

Inclui plugins como Cucumber e Faker, que facilitam a criação de testes e a geração de dados.

Embora outras ferramentas, como Selenium, Playwright ou Katalon, também possam ser utilizadas, o Cypress se destaca por sua praticidade e recursos específicos. A ferramenta oferece vantagens como testes de componentes gráficos e a capacidade de interceptar requests, tornando-a uma escolha atraente para projetos futuros. Além disso, o Cypress é compatível com JavaScript para testes móveis, de performance e de back-end, simplificando a codificação de novos testes.

Portanto, o Cypress demonstrou ser uma solução eficaz para o projeto, oferecendo uma variedade de tipos de testes e uma curva de aprendizado favorável dentro do prazo do curso.

#### 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.

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*)  
Testes implementados para *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

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*.






- 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

Este trabalho apresentou uma abordagem abrangente e detalhada sobre o uso do Cypress para a automação de testes de software, destacando a importância da qualidade e da eficiência no desenvolvimento de aplicações. Através da implementação de testes utilizando PageObjects e AppActions, e a adoção da linguagem Gherkin para descrever os cenários de teste, conseguimos garantir que nossos testes fossem bem organizados e alinhados com os princípios do BDD (Behavior Driven Development).

Optar pelo Cypress como ferramenta para automação de testes web se mostrou uma escolha acertada, devido às suas múltiplas vantagens. A capacidade do Cypress de realizar testes tanto de interface quanto de back-end na mesma plataforma, aliada à popularidade da linguagem JavaScript, facilitou a implementação e ajustes dos testes, além de proporcionar uma integração eficiente com a integração contínua. A ferramenta também oferece suporte robusto, com uma comunidade ativa, a possibilidade de gravação e execução em segundo plano dos testes, e a geração de relatórios detalhados,

o que contribuiu significativamente para a nossa capacidade de monitorar e melhorar a qualidade da aplicação.

Embora outras ferramentas como Selenium, Playwright ou Katalon também fossem opções viáveis, o Cypress se destacou pela sua praticidade e funcionalidades específicas, como a capacidade de realizar testes de componentes gráficos e interceptar requests. Esses recursos adicionais proporcionam uma flexibilidade e uma profundidade de testes que são essenciais para projetos complexos.

Além disso, o uso de massa de dados e plugins como Cucumber e Faker facilitou a criação e a gestão dos testes, garantindo que eles fossem robustos e representativos de cenários reais. A experiência acumulada ao longo do projeto demonstrou que o Cypress não apenas atende às necessidades atuais, mas também oferece um caminho claro para a expansão de testes em áreas adicionais, como testes móveis, de performance e de back-end, usando a mesma base de conhecimento em JavaScript.

Em suma, o trabalho evidenciou que o Cypress é uma ferramenta altamente eficaz e eficiente para automação de testes, alinhada com as necessidades do projeto e as melhores práticas da indústria. A sua adoção permitiu não apenas a implementação de testes automatizados de alta qualidade, mas também estabeleceu uma base sólida para futuros desenvolvimentos e melhorias contínuas na aplicação.

## 6. REFERÊNCIAS BIBLIOGRÁFICAS

- **CYPRESS IO.** *Cypress Documentation*. Disponível em: <https://docs.cypress.io>. Acesso em: 16 set. 2024.
- **W3C.** *HTML Living Standard*. Disponível em: <https://html.spec.whatwg.org/multipage/>. Acesso em: 16 set. 2024.
- **JQUERY FOUNDATION.** *jQuery Documentation*. Disponível em: <https://api.jquery.com>. Acesso em: 16 set. 2024.
- **MOZILLA.** *MDN Web Docs: JavaScript*. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Acesso em: 16 set. 2024.
- **NPM.** *npm Documentation*. Disponível em: <https://docs.npmjs.com>. Acesso em: 16 set. 2024.
- **NODE.JS.** *Node.js Documentation*. Disponível em: <https://nodejs.org/en/docs/>. Acesso em: 16 set. 2024.
- **EBAC – ESCOLA BRITÂNICA DE ARTES CRIATIVAS & TECNOLOGIA.** *Cursos de Formação e Certificação*. Disponível em: <https://www.ebac.com.br>. Acesso em: 16 set. 2024.