

**QUALIDADE DE SOFTWARE**

José Carlos Fonseca da Conceição

Análise de Qualidade

Belém/PA

2024

# RESUMO

Este trabalho de conclusão apresenta uma análise abrangente dos temas abordados no Curso de Qualidade de Software da Escola Britânica de Artes Criativas & Tecnologia (EBAC). O conteúdo explora as estratégias de teste aplicadas, os critérios de aceitação, a elaboração de casos de teste, o uso de repositórios no GitHub, a automação de testes, a integração contínua e os testes de performance. A estratégia de teste adotada foi desenvolvida com base nas melhores práticas discutidas ao longo do curso, visando garantir a qualidade e a funcionalidade do software, bem como demonstrar a aplicação consolidada das técnicas adquiridas durante o aprendizado.

# SUMÁRIO

[1. RESUMO 2](#_Toc85541169)

[2. SUMÁRIO 3](#_Toc85541170)

[3. INTRODUÇÃO 4](#_Toc85541171)

[4. O PROJETO 5](#_Toc85541172)

[4.1 Estratégia de teste 5](#_Toc85541173)

[4.2 Critérios de aceitação 6](#_Toc85541174)

[4.2.1 História de usuário 1: [US-0001] – Adicionar item ao carrinho 6](#_Toc85541175)

[4.2.2 História de usuário 2: [US-0002] – Login na plataforma 8](#_Toc85541176)

[4.2.3 História de usuário 2: [US-0003] – API de cupons 9](#_Toc85541177)

[4.3 Casos de testes 11](#_Toc85541178)

[4.3.1 História de usuário 1: 11](#_Toc85541179)

[4.3.2 História de usuário 2: 12](#_Toc85541180)

[4.3.1 História de usuário 3: API de Cupom 12](#_Toc85541181)

[4.4 Repositório no Github 12](#_Toc85541182)

[4.5 Testes automatizados 12](#_Toc85541183)

[4.6 Integração contínua 13](#_Toc85541184)

[4.7 Testes de performance 13](#_Toc85541185)

[5. CONCLUSÃO 14](#_Toc85541186)

[6. REFERÊNCIAS BIBLIOGRÁFICAS 16](#_Toc85541187)

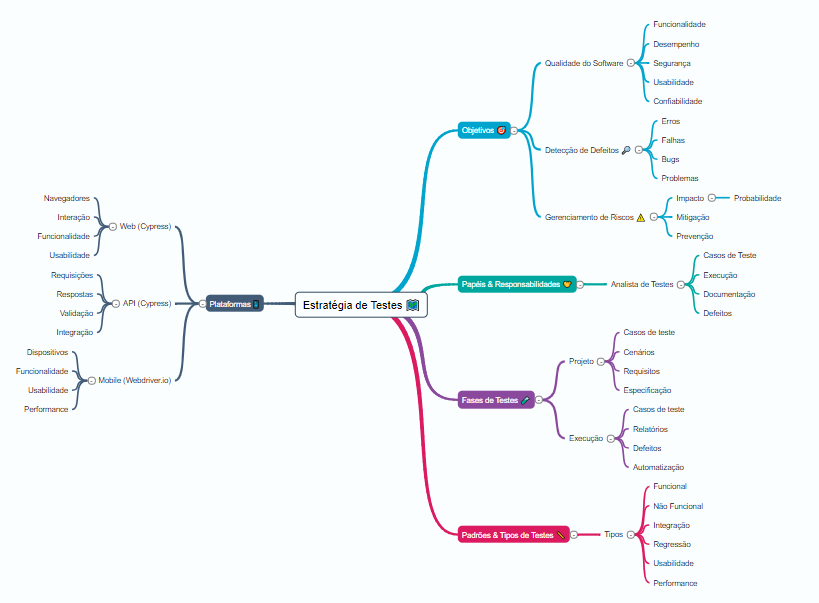
# INTRODUÇÃO

A elaboração dos casos de teste buscou cobrir de forma abrangente todos os requisitos funcionais e não funcionais do sistema, detalhando minuciosamente as etapas envolvidas, os dados de entrada, as ações executadas e os resultados esperados. Todo o código-fonte, juntamente com os casos de teste e a documentação correspondente, foi armazenado em um repositório no GitHub, o que possibilita o controle de versão, a colaboração entre desenvolvedores e o rastreamento eficaz de issues. Os testes automatizados foram implementados com o uso de ferramentas como Cypress, K6 e Visual Studio Code (VSCode), sendo executados regularmente para validar a funcionalidade do sistema após cada alteração no código. Como destacado por Pressman (2011), "a automação de testes é essencial para assegurar que modificações no software não introduzam novos defeitos" (p. 567), reafirmando a importância da implementação de um processo contínuo e eficiente de testes automatizados.

# O PROJETO

Para o Trabalho de Conclusão de Curso Qualidade de Software, você deve considerar as histórias de usuário já refinadas e como se você estivesse participando de um time ágil. As funcionalidades devem seguir todo o fluxo de trabalho de um QA, desde o planejamento até a entrega. Siga as etapas dos sub-tópicos para te orientar no trabalho. Todas as boas práticas, tanto de documentação, escrita e desenvolvimento, serão consideradas na nota. Portanto caprichem, pois além de trabalho servir como nota para o curso, vai servir como Portfólio em seu github.

# Estratégia de teste



Fonte: autor (2024)

# 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 2 critérios de aceitação usando a linguagem Gherkin;
* Em pelo menos um dos critérios, usar tabela de exemplos (Esquema do Cenário / Scenario Outline);

Referência: Módulo 8

# História de usuário 1: [US-0001] – Adicionar item ao carrinho

**Funcionalidade:** Não é permitido inserir mais de 10 itens de um mesmo produto ao carrinho;

**Cenário 1: Adicionar mais de 10 itens do mesmo produto**

**Dado** que um cliente acessa a página do produto <Produto A>  
**Quando** o cliente tenta adicionar <quantidade> unidades do "Produto A" ao carrinho  
**Então** o sistema deve exibir uma mensagem de erro <mensagem>

Exemplos:

|  |  |  |
| --- | --- | --- |
| produto | quantidade | mensagem |
| Produto A | 11 | Não é permitido adicionar mais de 10 unidades deste produto ao carrinho |
| Produto B | 12 | Não é permitido adicionar mais de 10 unidades deste produto ao carrinho |

**Cenário 2: Adicionar até 10 itens do mesmo produto**

**Dado** que um cliente acessa a página do produto <Produto A>  
**Quando** o cliente adiciona <quantidade> do <Produto B> ao carrinho  
**Então** o sistema deve permitir a adição de <quantidade> do <Produto B> ao carrinho e a compra pode ser finalizada com sucesso

Exemplos:

|  |  |
| --- | --- |
| produto | quantidade |
| Produto A | 10 |
| Produto B | 10 |

**Funcionalidade:** Os valores não podem ultrapassar a R$ 990,00;

**Cenário 1: Adicionar produtos totalizando mais de R$ 990,00**

**Dado** que um cliente adiciona produtos <Produto x e Produto y> com valores que somam R$ 995,00 ao carrinho

**Quando** o cliente tenta finalizar a compra

**Então** o sistema deve exibir uma mensagem de erro <mensagem>

Exemplos:

|  |  |  |
| --- | --- | --- |
| produto | valor | mensagem |
| Produto x | R$ 500,00 | O valor total do carrinho não pode ultrapassar R$ 990,00 |
| Produto y | R$ 495,00 |  |
| Produto z | R$ 995,00 |  |

**Cenário 2: Adicionar produtos totalizando exatamente R$ 990,00**

**Dado** que um cliente adiciona os produtos <Produto A e Produto B> com valores que somam <valor> ao carrinho

**Quando** o cliente tenta finalizar a compra

**Então** o sistema deve permitir a finalização da compra com sucesso

Exemplos:

|  |  |
| --- | --- |
| produto | valor |
| Produto A | R$ 300,00 |
| Produto B | R$ 690,00 |
| Produto C | R$ 990,00 |

**Cenário 1: Aplicar cupom de 10% para valores entre R$ 200,00 e R$ 600,00**

**Dado** que um cliente tem um total de R$ 250,00 no carrinho  
**Quando** o cliente aplica o cupom de desconto  
**Então** o sistema deve aplicar um desconto de 10% e o valor final da compra

deve ser R$ 225,00

**Cenário 2: Aplicar cupom de 10% com diferentes valores**

**Dado** que um cliente tem um total de <valor> no carrinho  
**Quando** o cliente aplica o cupom de desconto  
**Então** o sistema deve aplicar um desconto de 10% e o valor final da compra deve ser <valor\_com\_desconto>

Exemplo:

|  |  |
| --- | --- |
| valor | valor\_com\_desconto |
| 250,00 | 225,00 |
| 300,00 | 270,00 |
| 400,00 | 360,00 |
| 600,00 | 540,00 |

**Cenário 1: Aplicar cupom de 15% para valores acima de R$ 600,00**

**Dado** que um cliente tem um total de R$ 700,00 no carrinho  
**Quando** o cliente aplica o cupom de desconto  
**Então** o sistema deve aplicar um desconto de 15% e o valor final da compra deve ser R$ 595,00  
  
**Cenário 2: Aplicar cupom de 15% com diferentes valores**

**Dado** que um cliente tem um total de <valor> no carrinho  
**Quando** o cliente aplica o cupom de desconto  
**Então** o sistema deve aplicar um desconto de 15% e o valor final da compra deve ser <valor\_com\_desconto>

Exemplos:

|  |  |
| --- | --- |
| valor | Valor\_com\_desconto |
| 700,00 | 595,00 |
| 900,00 | 765,00 |

# História de usuário 2: [US-0002] – Login na plataforma

**Cenário 1: Usuário ativo faz login com sucesso**

**Dado** que um usuário ativo possui "usuário" e "senha" válidos  
**Quando** o usuário realizar o login com as credenciais  
**Então** o sistema deve permitir o login e o usuário deve ser redirecionado para a página principal

**Cenário 2: Usuário inativo tenta fazer login**

**Dado** que um usuário inativo  
**Quando** tentar fazer login usando <usuario> e <senha>  
**Então** o sistema deve exibir a mensagem de erro <mensagem> e o login deve ser recusado

Exemplos:

|  |  |  |
| --- | --- | --- |
| usuario | senha | mensagem |
| User456 | Abc123 | Usuário inativo. Por favor, contate o suporte |

**Cenário 1: Usuário erra a senha**

**Dado** que um usuário tenta fazer login com o "usuário" correto e a "senha" errada  
**Quando** o usuário submete suas credenciais  
**Então** o sistema deve exibir a mensagem de erro <mensagem>

Exemplos:

|  |  |  |
| --- | --- | --- |
| usuario | senha | mensagem |
| user\_correto | senha\_errada | Credenciais inválidas. Por favor, verifique seu e-mail ou senha. |

**Cenário 2: Usuário erra o nome de usuário**

**Dado** que um usuário tenta fazer login com o "usuário" incorreto e a "senha" correta  
**Quando** o usuário submete suas credenciais  
**Então** o sistema deve exibir a mensagem de erro <mensagem>

Exemplos:

|  |  |  |
| --- | --- | --- |
| usuario | senha | mensagem |
| user\_errado | senha\_ correta | Credenciais inválidas. Por favor, verifique seu e-mail ou senha. |

**Cenário 1: Login usando e-mail**

**Dado** que um usuário tenta fazer login com o e-mail "usuario@ebac.com" e a senha "ebac123"  
**Quando** o usuário submete suas credenciais  
**Então** o sistema deve permitir o login e o usuário deve ser redirecionado para a página principal

**Cenário 2: Login usando nome de usuário ou CPF**

**Dado** que um usuário possui <usuario> e <senha>  
**Quando** o usuário tenta fazer login usando <usuario> e <senha>  
**Então** o sistema deve permitir o login e o usuário deve ser redirecionado para a página principal

|  |  |
| --- | --- |
| usuario | senha |
| userebac | ebac123 |
| 12345678912 | ebac123 |
| user@ebac.com | ebac123 |

# História de usuário 2: [US-0003] – API de cupons

**Cenário 1: Listar todos os cupons cadastrados**

**Dado** que o administrador acessa a API de cupons  
**Quando** o administrador faz uma requisição GET para a rota "/cupons"  
**Então** o sistema deve retornar uma lista de todos os cupons cadastrados, o status code "200" e os payloads com as informações: "codigo", "valor", "tipo\_desconto", e "descrição" para cada cupom  
  
Exemplo:

|  |  |  |
| --- | --- | --- |
| rota | staus\_code | campos\_retornado |
| /cupons | 200 | codigo, valor, tipo\_desconto, descrição |

**Cenário 2: Listar cupom por ID**

**Dado** que o administrador acessa a API de cupons e um cupom com o ID "123" está cadastrado  
**Quando** o administrador faz uma requisição GET para a rota "/cupons/123"  
**Então** o sistema deve retornar os detalhes do cupom com ID "123" e a resposta deve conter os campos "codigo", "valor", "tipo\_desconto", e "descrição"

Exemplo:

|  |  |  |
| --- | --- | --- |
| rota | staus\_code | campos\_retornado |
| /cupons/123 | 200 | codigo, valor, tipo\_desconto, descrição |

**Documentação do Serviço**

**GET /cupons**

**Descrição:** Lista todos os cupons cadastrados ou um cupom específico por ID.

**Resposta de sucesso:** 200 OK com a lista de cupons.

**Parâmetros:**

id (opcional): ID do cupom para buscar um cupom específico.

**Cenário 1: Cadastrar um novo cupom**

**Dado** que o administrador acessa a API de cupons e o código do cupom "Ganhe10%" não está em uso

**Quando** o administrador faz uma requisição POST para a rota "/cupons" com o seguinte body:

*""" {*

*"code": "Ganhe10%",*

*"amount": "10.00",*

*“discount\_type":*

*"fixed\_product",*

*"description": "Cupom de desconto de teste"*

*} """*

**Então** o sistema deve cadastrar o cupom com sucesso e o sistema deve retornar uma mensagem de sucesso "Cupom cadastrado com sucesso"

**Cenário 2: Tentar cadastrar um cupom com nome** repetido

**Dado** que o administrador acessa a API de cupons e um cupom com o código "Ganhe10" já está cadastrado

**Quando** o administrador faz uma requisição POST para a rota "/cupons" com o seguinte body:

""" {

"code": "Ganhe10",

"amount": "10.00",

"discount\_type":

"fixed\_product",

"description": "Outro cupom de desconto"

} """

**Então** o sistema deve retornar uma mensagem de erro "Código do cupom já está em uso" e o cupom não deve ser cadastrado

**Documentação do Serviço**

**POST /cupons**

**Descrição:** Cadastra um novo cupom.

**Parâmetros obrigatórios no body:**

code: Código do cupom (exemplo: "Ganhe10").

amount: Valor do desconto (exemplo: "10.00").

discount\_type: Tipo do desconto (exemplo: "fixed\_product").

description: Descrição do cupom (exemplo: "Cupom de desconto de teste").

**Resposta de sucesso:** 201 Created com mensagem de sucesso.

**Resposta de erro:** 400 Bad Request se o código do cupom já existir.

# Casos de testes

* Crie pelo menos 3 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...”
* Referência: Módulo 4 e 5

# História de usuário 1: Adicionar item ao carrinho

* Deve permitir adicionar até 10 itens no carrinho;
* Deve permitir o valor de compra até R$:990,00;
* Deve aplicar cupom desconto em compras nos valores determinados;
* Ao incluir acima de 10 itens no carrinho deve exibir uma mensagem de alerta “Quantidade de itens não permitida”;
* Ao ultrapassar o limite de R$: 990,00 deve exibir uma mensagem de alerta “Não é permitido compras a cima de R$: 990,00”;
* Ao aplicar cupom desconto em compras nos valores não determinados deve exibir uma mensagem de alerta “Não é permitido aplicar cupom desconto para este valor de compra”;

# História de usuário 2: Login na plataforma

* Ao realizar login com sucesso o usuário deve ser direcionado para a tela principal;
* Ao realizar login com senha invalida deve retornar uma mensagem de erro “Login ou senha Invalida”;
* Ao realizar login com um usuário inexistente deve retornar uma mensagem de erro “Não é possível realizar Login Usuário não Cadastrado”;

# História de usuário 3: API de Cupom

* Deve listar todos cupons
* Deve Listar cupom especifico
* Ao tentar realizar busca de cupom

# Repositório no Github

* Crie um repositório no github com o nome TCC-EBAC;
* Deixe o repositório publico até a análise dos tutores;
* Neste repositório você deve subir este arquivo e todos os código fontes da automação WEB, API, Mobile, Performance e CI.
* Referência: Módulo 10
* Link do repositório: https://github.com/carlosconca/TCC-EBAC

# Testes automatizados

# Automação de UI

* Crie um projeto de automação no Cypress;
* Crie uma pasta chamada UI para os testes WEB da História de Usuário [US-0001] – Adicionar item ao carrinho;
* Na automação deve adicionar pelo menos 3 produtos diferentes e validar se os itens foram adicionados com sucesso.

# Automação de API

* Crie uma pasta chamada API para os testes de API da História de usuário **“Api de cupons”**.
* Faça a automação de **listar** os cupons e **cadastrar** cupom, seguindo as regras da História de usuário.
* Exemplo da automação de Api – GET

it('Deve listar todos os cupons cadastrados', () => {

cy.request({

method: 'GET',

url: 'coupons',

headers: {

authorization: 'código\_da\_autorização\_aqui'

}

}).should((response) => {

cy.log(response)

expect(response.status).to.equal(200)

})

});

* Obs.: Considere todas as boas práticas de otimização de cenários (Page Objects, Massa de dados, Custom Commands, elementos etc.).
* Referência: Módulo 11, 12 e 14

# Integração contínua

* Coloque os testes automatizados na integração contínua com jenkins, criando um job para execução da sua automação;
* Compartilhe o *jenkinsfile* no repositório, junto ao seu projeto.
* Referência: Módulo 15

# Testes de performance

* Usando o Apache Jmeter, faça um teste de performance com o fluxo de login da História de usuário:

[US-0002] – Login na plataforma

* Crie um template de gravação no jmeter (recording);
* Use massa de dados dinâmica em arquivo CSV;
* Referência: Módulo 18
* 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

* DICA: Em uma das requisições, após a gravação, vai aparecer os parâmetros usado. Substitua esses parâmetros pela sua massa de dados, conforme aprendido em aula:



# CONCLUSÃO

A realização deste trabalho proporcionou uma oportunidade valiosa para consolidar os conhecimentos adquiridos ao longo do Curso de Qualidade de Software da EBAC. A aplicação prática dos conceitos de estratégia de teste, critérios de aceitação, casos de teste e uso de ferramentas como Cypress, JMETER, Banco de Dados e GitHub foi fundamental para compreender como a teoria pode ser efetivamente transformada em soluções que asseguram a qualidade de um software.

Durante o desenvolvimento dos casos de teste, ficou evidente a importância de um planejamento rigoroso e detalhado para garantir que todos os requisitos funcionais e não funcionais do sistema fossem contemplados. Essa experiência reforçou a relevância de boas práticas em testes automatizados, especialmente na manutenção da integridade e funcionalidade do software frente a mudanças e atualizações.

A experiência adquirida com este projeto será aplicada em minha vida profissional, principalmente na adoção de uma abordagem sistemática e proativa em relação à qualidade de software. Aprendi a importância de definir critérios claros de aceitação e a elaborar casos de teste que reflitam fielmente as necessidades do usuário final. Além disso, a prática constante com ferramentas de automação e integração contínua me preparou para enfrentar desafios complexos em ambientes reais de trabalho.

Concluo este trabalho com uma visão mais ampla e aprofundada do papel do QA no desenvolvimento de software, bem como com a confiança necessária para implementar essas lições em minha carreira profissional, contribuindo para o sucesso dos projetos em que estiver envolvido.

# REFERÊNCIAS BIBLIOGRÁFICAS

**Referência:** PRESSMAN, R. S. *Engenharia de software: uma abordagem profissional*. 7. ed. São Paulo: McGraw-Hill, 2011.

EBAC - Escola Britânica de Artes Criativas & Tecnologia. *Curso de Qualidade de Software*. São Paulo, 2024. Disponível em: https://lms.ebaconline.com.br/lesson/33802a94-4b25-420a-82a2-e63268a86b82. Acesso em: 29 ago. 2024.