

**QUALIDADE DE SOFTWARE**

Lucas Helvidio Piovezan Rocon

Análise de Qualidade

Vila Velha

2023

# RESUMO

Neste projeto será demonstrado algumas metodologias de teste e qualidade de

de software; tais

• Testes automatizados em UI e API;

• Integração contínua;

• Testes de performance;

• Técnicas de teste;

Para isso, documentei e utilizei uma plataforma específica como exemplo durante o projeto.

# SUMÁRIO

[1. RESUMO 2](#_Toc85541188)

[2. SUMÁRIO 3](#_Toc85541189)

[3. INTRODUÇÃO 4](#_Toc85541190)

[4. O PROJETO 5](#_Toc85541191)

[4.1 Estratégia de teste 5](#_Toc85541192)

[4.2 Critérios de aceitação 5](#_Toc85541193)

[4.2.1 História de usuário 1: [US-0001] – Adicionar item ao carrinho 5](#_Toc85541194)

[4.2.2 História de usuário 2: [US-0002] – Login na plataforma 6](#_Toc85541195)

[4.2.3 História de usuário 2: [US-0003] – API de cupons 6](#_Toc85541196)

[4.3 Casos de testes 6](#_Toc85541197)

[4.3.1 História de usuário 1: 6](#_Toc85541198)

[4.3.2 História de usuário 2: 6](#_Toc85541199)

[4.3.1 História de usuário 3: API de Cupom 7](#_Toc85541200)

[4.4 Repositório no Github 7](#_Toc85541201)

[4.5 Testes automatizados 7](#_Toc85541202)

[4.6 Integração contínua 8](#_Toc85541203)

[4.7 Testes de performance 8](#_Toc85541204)

[5. CONCLUSÃO 9](#_Toc85541205)

[6. REFERÊNCIAS BIBLIOGRÁFICAS 9](#_Toc85541206)

# INTRODUÇÃO

Esse documento aborda as estratégias de teste e o uso delas na criação de

cenários de teste, embasada nas histórias de usuário.

E tem como objetivo demonstrar os benefícios de adotar as práticas de

qualidade em um time de desenvolvimento de software e quanto isso agrega valor ao negócio.

# O PROJETO

Para o Projeto mestre da primeira parte do curso Profissão: Engenheiro de Qualidade, 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

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

Diagrama

Descrição gerada automaticamente

## Critérios de aceitação

* Considere as histórias de usuário: [US-0001] – Adicionar item ao carrinho, [US-0002] – Login na plataforma e [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

Critérios de aceitação:  
  
Cenário 1: Teste de aplicação de Cupom

**Dado** que meu carrinho de compras possui <total> de R$ em produtos  
**Quando** eu acessar a tela de pagamento

**Então** será aplicado um <cupom> de <?%> de desconto

|  |  |
| --- | --- |
| **Total** | **Cupom** |
| R$240 | 10% |
| R$700 | 15% |
| R$100 | 0% - N/A |

Cenário 2: Teste de quantidade de produtos

**Dado** que estou na página do produto  
**Quando** eu adicionar uma quantidade inválida de produtos no carrinho

**Então** será exibido um modal de alerta ao cliente

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

Critérios de aceitação:

**Usuário Ativo**: O sistema deve permitir que apenas usuários ativos façam login na plataforma. Os usuários inativos não devem ser capazes de fazer login.

**Exibir Mensagem de Erro**: Quando um usuário errar o login ou a senha, o sistema deve exibir uma mensagem de erro clara e informativa, indicando que houve um erro no processo de autenticação.

*Cenário 1*: Validação de login na plataforma

**Dado** que o usuário esteja na página de Login

**Quando** o usuário preencher o campo de login com um **<usuário>**

**E** preencher o campo de senha com uma **<senha>**

**Então** será validado se a autenticação é válida

**E** o login será sucedido

**OU** retornará uma mensagem de erro

|  |  |  |
| --- | --- | --- |
| **<USUÁRIO>** | **<SENHA>** | **RESULTADO** |
| lucas.rocon | Teste123 | Login sucedido |
| 188742rocon | @123465 | Usuário ou senha inválidos! |
| lucas.rocon | teste@3562 | Usuário ou senha inválidos! |

*Cenário 2*: Validação de travamento no login

Critério de aceitação:

**Travamento após 3 Tentativas Erradas:** Se o usuário errar a senha em três tentativas consecutivas, o sistema deve travar o login desse usuário por um período de 15 minutos. Durante esse tempo, o usuário não deve poder fazer novas tentativas de login.  
  
**Dado** que o usuário está na tela de login

**Quando** o usuário errar a senha três vezes consecutivas

**Então** o usuário será bloqueado respeitando o critério de aceitação

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

Critérios de aceitação:  
  
**Cenário 1**: Não permitir cadastrar um cupom repetido

**Dado** que o usuário seja um administrador

**Quando** o usuário requisitar um POST com o nome do cupom já existente

**Então** a mensagem de resposta deve mostrar “O código de cupom já

existe"

**Cenário 2**: Cupons cadastrados

**Dado** que eu seja um usuário admistrador

**Quando** eu requisitar um GET usando o usuário <usuario>

**E** a senha <senha>

**Então** o código de status da resposta deve ser <resposta>

|  |  |  |
| --- | --- | --- |
| <Usuário> | <Senha> | <Resposta> |
| admin\_ebac | @admin!&b@c!2022 | 200 OK |
| admin\_ebac | @admin!&b@c!2023 | 400 Bad Request |
| admin\_rocon | @testeteste | 400 Bad Request |

## 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:

**CAMINHO FELIZ:**

CT01: **Adicionar Produto ao Carrinho com Valor Dentro do Limite**

Ação: Adicionar um produto de R$ 150,00 ao carrinho.

Resultado Esperado: O produto é adicionado ao carrinho sem problemas.

**Caminhos Alternativos/Negativos:**

CT02: **Tentar Adicionar Produto com Valor Acima de R$ 990,00**

Ação: Adicionar um produto de R$ 1.000,00 ao carrinho.

Resultado Esperado: Deve exibir uma mensagem de erro informando que os valores não podem ultrapassar R$ 990,00.

CT03: **Tentar Adicionar Produto com Valor Entre R$ 200 e R$ 600 sem Cupom**

Ação: Adicionar produtos no valor total de R$ 400,00 ao carrinho sem aplicar um cupom.

Resultado Esperado: Deve exibir uma mensagem de alerta informando que valores entre R$ 200,00 e R$ 600,00 ganham um cupom de 10%. Perguntar se o usuário deseja aplicar o cupom.

## História de usuário 2:

CT01:

CT02:

CT03:

## História de usuário 3: API de Cupom

CT01:

CT02:

CT03:

## 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/lrocon18/TCC-EBAC

## Testes automatizados

* + 1. 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.
  + 1. 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

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

# REFERÊNCIAS BIBLIOGRÁFICAS

Seguir regras ABNT