

**QUALIDADE DE SOFTWARE**

Ana Paula Medeiros

Análise de Qualidade

Osasco/SP

2023

# RESUMO

O projeto de engenharia de qualidade de software foca na otimização e garantia da qualidade em todas as etapas do ciclo de vida do desenvolvimento de software. Envolve a implementação de metodologias como testes automatizados, revisões de código, análise estática, entre outros, para identificar e corrigir defeitos precocemente. A abordagem busca não apenas corrigir problemas, mas preveni-los, visando melhorar a confiabilidade, segurança e desempenho do software. Além disso, inclui o gerenciamento de processos e métricas para monitorar e aprimorar continuamente a qualidade do produto, alinhando-se com as expectativas do usuário e objetivos do negócio. Este projeto busca não só a excelência técnica, mas também a entrega de produtos que atendam às necessidades do cliente de forma eficiente e eficaz.

# 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

A engenharia de qualidade de software desempenha um papel vital na entrega de produtos confiáveis e eficientes. Este campo aborda a criação, implementação e aprimoramento de processos que garantem a qualidade em todas as fases do ciclo de vida do software. Desde a concepção até a manutenção, o engenheiro de qualidade de software atua na identificação proativa de defeitos, na otimização de desempenho e na garantia de que os requisitos do usuário sejam atendidos. Este projeto explorará métodos como testes automatizados, análise estática de código, revisões sistemáticas e técnicas ágeis para melhorar continuamente a qualidade do software. Além disso, será abordado o gerenciamento de métricas e processos para medir e aprimorar a eficácia dos processos de desenvolvimento. Espera-se que esse trabalho evidencie a importância estratégica da engenharia de qualidade de software na entrega de produtos confiáveis, seguros e que atendam às demandas do mercado em constante evolução.

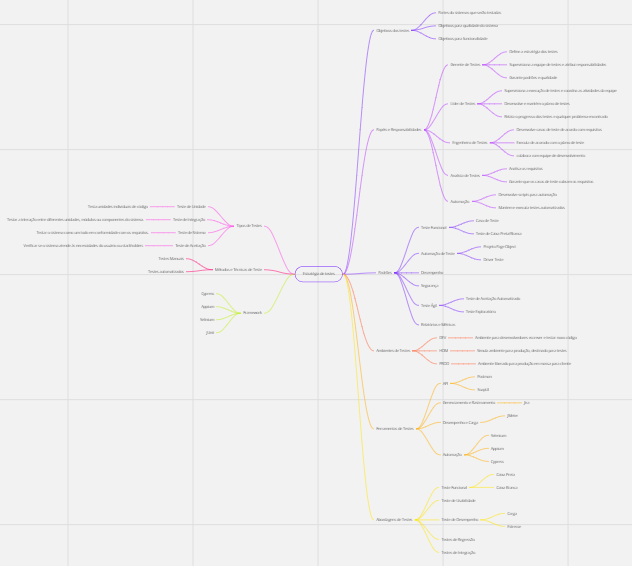
# O PROJETO

O Projeto do trabalho sobre Engenheiro de Qualidade, deve-se considerar as histórias de usuário refinadas e atuação em um time ágil. As funcionalidades seguem de trabalho de um QA, com planejamento e etapas até a entrega. Boas práticas de documentação, escrita e desenvolvimento, são consideradas.

## Estratégia de teste

A estratégia de teste é um plano detalhado que descreve como os testes serão conduzidos durante o ciclo de vida do desenvolvimento de software. Define a abordagem, os métodos, os recursos e as ferramentas a serem utilizados para garantir que o produto atenda aos requisitos de qualidade.

Uma estratégia de teste bem elaborada é fundamental para garantir a qualidade do software, fornecendo diretrizes claras para toda a equipe envolvida no desenvolvimento. É adaptável e pode ser ajustada conforme o progresso do projeto, permitindo a identificação precoce de problemas e a entrega de um produto de alta qualidade.



[Imagem: Mapa mental – Estratégia de teste]

## Critérios de aceitação

Critérios de aceitação de teste são condições ou requisitos específicos que um sistema ou funcionalidade deve atender para ser considerado aprovado após a realização de testes. Esses critérios são estabelecidos com base nos requisitos funcionais e não funcionais do software e servem como base para determinar se uma funcionalidade atende às expectativas do usuário e aos objetivos do projeto.

Devem ser escritos com linguagem clara e compreensível para todos os envolvidos no projeto e são usados para definir o escopo dos testes, definindo o que deve ser validado durante os testes e o que é esperado do sistema, garantindo a qualidade que será entregue.

Nos cenários a seguir será utilizada a Linguagem **Gherkin.**

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

Critérios de aceitação:  
  
Cenário 1: Adicionar um item válido ao carrinho  
**Dado** que o usuário está na página do produto

**Quando** o usuário clica no botão “Adicionar ao carrinho”

**Então** o item deve ser adicionado ao carrinho com sucesso

Cenário 2: Adicionar vários itens ao carrinho

**Dado** que o usuário está na página do produto

**Quando** o usuário adiciona itens ao carrinho

| produto | Quantidade|

| Camisa | 2 |

| Calça | 1 |

| Sapato | 4 |

**Então** todos os itens devem ser adicionados ao carrinho corretamente

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

Critérios de aceitação:  
  
Cenário 1: Login com credenciais válidas

**Dado** que o usuário está na página de login

**Quando** o usuário insere suas credenciais válidas (usuário e senha)

**Então** o usuário deve ser redirecionado para a página principal/logada

Cenário 2: Login sem inserir senha

**Dado** que o usuário está na página de login

**Quando** o usuário insere seu usuário sem inserir a senha

**Então** uma mensagem de erro deve ser exibida informando sobre a senha ausente

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

Critérios de aceitação:  
  
Cenário 1: Aplicar um cupom válido ao carrinho

**Dado** que o usuário tem um cupom válido

**Quando** o usuário aplica o cupom ao carrinho

**Então** o desconto do cupom deve ser refletido no total do carrinho

Cenário 2: Aplicar um cupom inválido ao carrinho

**Dado** que o usuário tem um cupom inválido

**Quando** o usuário tenta aplicar o cupom ao carrinho

**Então** uma mensagem de erro deve ser exibida informando sobre a invalidade do cupom

## Casos de testes

Os casos de testes descrevem como deve ser o comportamento esperado do sistema em determinadas condições ou ações, conseguindo assim, identificar se a funcionalidade está de acordo com o esperado.

## História de usuário 1:

CT01:Adicionar um item válido ao carrinho.

Item é adicionado ao carrinho com sucesso.

CT02: Tentar adicionar um item inexistente

Aparece mensagem informando que item não existe

CT03: Adicionar quantidade inválida(zero)

Deve aparecer mensagem que a quantidade informada é inválida

## História de usuário 2:

CT01:Login com credenciais válidas

É feito o redirecionamento para a página logada

CT02: Login com usuário válido, sem inserir a senha

Aparece mensagem de erro informando que a senha está ausente/não informada

CT03: Login com credenciais inválidas

Deve aparece mensagem informando que as credencias são inválidas

## História de usuário 3: API de Cupom

CT01: Aplicar um cupom válido ao carrinho

O Desconto do cupom deve ser considerado no valor total do carrinho

CT02: Aplicar um cupom inexistente

Deve aparecer uma mensagem informando que o cupom não existe

CT03: Aplicar um cupom expirado

Deve aparecer uma mensagem informando que o cupom está expirado

## Repositório no Github

GitHub é uma plataforma de hospedagem de código-fonte na web, utilizada para controle de versão e colaboração em projetos de desenvolvimento de Software e oferece recursos para equipes de DEV´s trabalharem juntos e compartilharem seus projetos.

Nesse trabalho usarei o seguinte link:

<https://github.com/anapmedeiros/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