

**QUALIDADE DE SOFTWARE**

Samuel Luis Argento Silva

Análise de Qualidade

Campinas

2025

# RESUMO

O projeto final do curso de Teste de Software da EBAC consiste em aplicar os conhecimentos adquiridos ao longo do programa em um cenário prático, simulando situações reais do mercado de trabalho. O objetivo principal é demonstrar habilidades em planejamento, execução e análise de testes em sistemas de software.

O trabalho abrange as etapas fundamentais do ciclo de testes, incluindo a elaboração de planos de teste, criação de casos de teste, execução de testes manuais e/ou automatizados, e registro de defeitos encontrados. Além disso, contempla o uso de ferramentas especializadas, como Jira para gerenciamento de tarefas, Selenium para automação de testes e Postman para testes de APIs.

O projeto também explora conceitos essenciais, como a definição de critérios de aceitação, priorização de testes e análise de resultados para garantir a qualidade do software. A proposta busca integrar as práticas de teste em diferentes tipos de sistemas, como aplicações web e mobile, oferecendo uma visão abrangente e prática sobre o papel do teste de software no desenvolvimento ágil.

Por fim, o trabalho é avaliado com base em critérios técnicos e metodológicos, permitindo que os alunos demonstrem seu domínio das técnicas e ferramentas essenciais para a atuação na área de qualidade de software.

# 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

O presente trabalho tem como objetivo explorar os processos e técnicas de teste de software em um cenário prático, consolidando os conhecimentos adquiridos ao longo do curso de Teste de Software da EBAC. O projeto aborda a importância da garantia de qualidade no desenvolvimento de sistemas, destacando como a aplicação de testes bem estruturados pode prevenir falhas e melhorar a experiência do usuário.

Ao longo do trabalho, são discutidos tópicos como a elaboração de planos de teste, a criação e execução de casos de teste, e o uso de ferramentas específicas para testes manuais e automatizados. Também são explorados aspectos relacionados à priorização de testes, critérios de aceitação e análise de resultados, buscando criar um panorama completo das práticas essenciais de teste no desenvolvimento ágil.

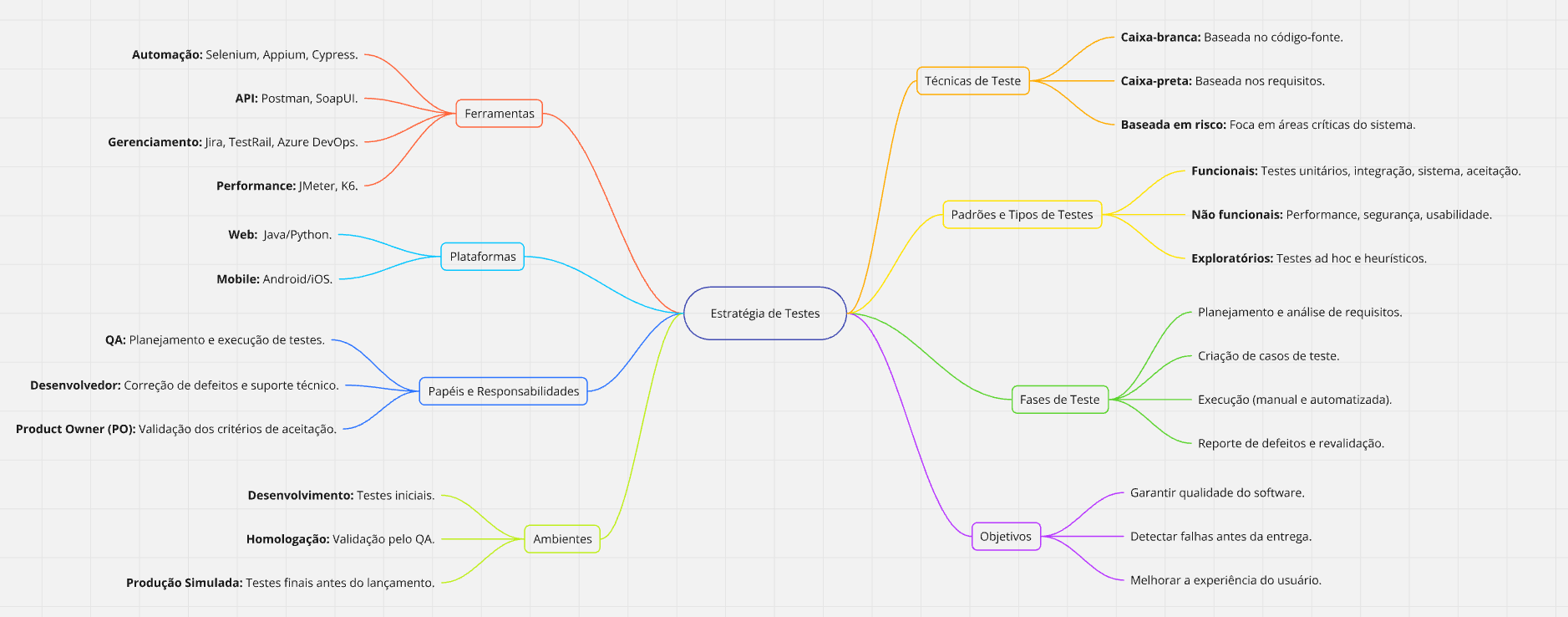
O projeto simula um cenário realista, no qual será necessário identificar requisitos, planejar as atividades de teste e registrar os resultados. A expectativa é que o trabalho demonstre como as práticas de teste podem agregar valor ao processo de desenvolvimento de software, promovendo maior confiabilidade, eficiência e satisfação do cliente.

Com isso, este projeto busca não apenas reforçar os conhecimentos técnicos adquiridos, mas também preparar o aluno para enfrentar os desafios do mercado de qualidade de software de forma prática e aplicada.

# 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



## 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

Cenário 1: Seleção realizada corretamente

Dado que eu acesse a página de um produto da EBAC SHOP

Quando eu selecionar a cor e o tamanho de 10 produtos ou menos

Então deve adicionar os produtos ao carrinho

Cenário 2: Limite máximo de itens no carrinho

Dado que eu esteja na página de um produto da EBAC SHOP

Quando eu adicionar <quantidade> unidades do produto ao carrinho

Então o sistema deve <resultado> a adição dos produtos

Exemplos:

| quantidade | resultado |

| 5 | permitir |

| 10 | permitir |

| 11 | não permitir |

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

Cenário 1: Login bem-sucedido

Dado que eu esteja na página de login da EBAC SHOP

Quando eu inserir um e-mail e senha válidos

Então devo ser redirecionado para a página inicial

Cenário 2: Tentativa de login com credenciais inválidas

Dado que eu esteja na página de login da EBAC SHOP

Quando eu tentar acessar com "<email>" e "<senha>"

Então o sistema deve exibir a mensagem "<mensagem>"

Exemplos:

| email | senha | mensagem |

| usuario@email.com | senha123 | Login realizado com sucesso |

|usuario@email.com | senhaErrada | Usuário ou senha inválidos |

|emailinvalido@email | senha123 | Usuário ou senha inválidos |

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

Cenário 1: Aplicação de cupom válido

Dado que a API receba um código de cupom válido

Quando a solicitação for enviada

Então o desconto correspondente deve ser aplicado E o sistema deve retornar um código de status 200

Cenário 2: Tentativa de uso de cupom expirado

Dado que a API receba um código de cupom "<cupom>"

Quando a solicitação for enviada

Então o sistema deve retornar a mensagem "<mensagem>" e código de status "<status>"

Exemplos:

| cupom | mensagem | status |

| DESCONTO10 | Cupom aplicado com sucesso | 200 |

| CUPOMEXPIRADO | Cupom expirado | 400 |

| CUPOMINVAL | Cupom inválido | 400 |

## 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:

**CT01:** Adicionar um produto ao carrinho com cor, tamanho e quantidade válida (≤10).

**CT02:** Tentar adicionar mais de 10 unidades e receber mensagem de erro.

**CT03:** Tentar adicionar um item sem selecionar cor ou tamanho e receber alerta.

## História de usuário 2:

**CT01:** Fazer login com e-mail e senha corretos e ser redirecionado para a home.

**CT02:** Inserir senha incorreta e receber mensagem de erro "Usuário ou senha inválidos".

**CT03:** Deixar campos vazios e receber alerta de obrigatoriedade.

## História de usuário 3: API de Cupom

**CT01:** Enviar cupom válido para API e receber status 200 com desconto aplicado.

**CT02:** Enviar cupom expirado e receber status 400 com mensagem "Cupom expirado".

**CT03:** Enviar cupom inválido e receber status 400 com mensagem "Cupom inválido".

## Repositório no Github

* Crie um repositório no github com o nome TCC-EBAC;
* Deixe o repositório público 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: Subir pro github depois.

## 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

Durante a realização deste projeto, tive a oportunidade de aprofundar meus conhecimentos em **testes de software**, tanto na parte teórica quanto na prática. Trabalhar com **automação de testes de UI e API** utilizando **Cypress** foi um grande aprendizado, pois permitiu que eu entendesse melhor como estruturar testes eficazes e reutilizáveis.

Além disso, aprendi sobre **estratégias de testes**, como definir critérios de aceitação usando **Gherkin**, aplicar **técnicas de teste (partição de equivalência, valores limite, tabela de decisão)** e utilizar boas práticas de automação, como **Page Objects, Custom Commands e uso de variáveis de ambiente**.

**Lições Aprendidas e Aplicação Profissional**

* **Melhor organização e planejamento dos testes** → Criar uma estratégia bem definida melhora a cobertura dos testes e facilita a manutenção do código.
* **Importância da automação** → Automação de testes reduz erros manuais, melhora a eficiência e permite entregas mais rápidas e confiáveis.
* **Boas práticas no código** → Aprendi a escrever testes mais limpos e organizados, facilitando a reutilização e escalabilidade.
* **Integração dos testes no processo de desenvolvimento** → A automação pode (e deve) ser integrada ao pipeline de CI/CD para garantir qualidade contínua.

Esse projeto me preparou para aplicar esses conhecimentos na minha carreira, tornando-me um profissional mais preparado para desafios na área de **QA e automação de testes**.

# REFERÊNCIAS BIBLIOGRÁFICAS

CYPRESS.IO. **Cypress - JavaScript End to End Testing Framework**. Disponível em: <https://www.cypress.io/>. Acesso em: 13 fev. 2025.