

**ENGENHEIRO DE QUALIDADE DE SOFTWARE**

João Pedro Rapack Kleinakuf

Análise de Qualidade

Campo Bom

2024

# RESUMO

Este trabalho aborda a importância e as práticas de Teste de Software no desenvolvimento de aplicativos web e mobile. Foca-se na criação e execução de um plano de testes detalhado, incluindo a definição de objetivos, papéis, responsabilidades, fases de teste, critérios de aceitação, e a utilização de diferentes técnicas e ferramentas de teste. Explora a implementação de testes automatizados para interfaces de usuário (UI), APIs e aplicações mobile, além da integração contínua para assegurar a qualidade e agilidade no desenvolvimento. Também são discutidos testes de performance utilizando a ferramenta K6, com foco em garantir a estabilidade e escalabilidade do software. O trabalho é complementado por três estudos de caso práticos, que exemplificam a aplicação dos conceitos teóricos adquiridos ao longo do curso. Este estudo visa fornecer uma base sólida para profissionais de Teste de Software, destacando a relevância do planejamento e da execução disciplinada de testes para o sucesso dos projetos de software.

**2.** **SUMÁRIO**

[1. RESUMO2](#_Toc99483086)

[2. SUMÁRIO3](#_Toc99483087)

[3. INTRODUÇÃO4](#_Toc99483088)

[4. O PROJETO5](#_Toc99483089)

[4.1 Estratégia de teste6](#_Toc99483090)

[4.2 Critérios de aceitação6](#_Toc99483091)

[4.3 Casos de testes8](#_Toc99483092)

[4.4 Repositório no Github11](#_Toc99483093)

[4.5 Testes automatizados12](#_Toc99483094)

[4.6 Integração contínua14](#_Toc99483095)

[4.7 Testes de performance14](#_Toc99483096)

[5. CONCLUSÃO15](#_Toc99483097)

[6. REFERÊNCIAS BIBLIOGRÁFICAS16](#_Toc99483098)

# 3. INTRODUÇÃO

O profissional de Teste de Software é essencial para monitorar cada etapa do desenvolvimento de software, garantindo que os resultados esperados sejam alcançados. Suas responsabilidades incluem criar planos de testes, rastrear bugs, desenvolver padrões de qualidade e identificar problemas que possam impactar o usuário final. O planejamento é vital nesse contexto, atuando como uma bússola que orienta o projeto do início ao fim, assegurando que os objetivos sejam claros e alcançáveis.

Entender a importância do planejamento é crucial, pois permite monitorar adequadamente as atividades em execução e identificar riscos. Planejar envolve definir um cronograma de atividades, alocar recursos e estabelecer marcos de projeto. Essas atividades estruturam o projeto, garantindo que cada etapa seja concluída conforme os requisitos.

No desenvolvimento de software, documentos como o plano de projeto, o documento de requisitos e o plano de teste são essenciais. Este trabalho foca no plano de teste, que especifica o escopo, objetivos, requisitos, estratégias e recursos para as atividades de teste. O plano de teste é um documento crucial que orienta o processo de teste, fornecendo uma visão clara e detalhada das ações necessárias para garantir a qualidade do software.

O teste de software é uma atividade crítica que visa identificar falhas e aumentar a confiabilidade do sistema. Envolve verificação e validação do software, avaliação de atributos de qualidade e controle dos releases durante o desenvolvimento. Embora seja impossível testar completamente um sistema, um teste disciplinado e sistemático pode identificar a maioria dos erros presentes.

Realizar testes de software implica responder a perguntas importantes, como quais atributos de qualidade testar, quem será responsável pelos testes, quais recursos serão usados e como o processo e a qualidade do software serão monitorados.

Este trabalho apresenta três casos práticos que exemplificam e aprofundam os tópicos abordados, proporcionando uma visão abrangente e prática do conhecimento adquirido. Esses casos práticos demonstram a aplicação das teorias e métodos discutidos, destacando a importância e os desafios do planejamento e execução de testes de software.

# 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/repository/docker/ernestosbarbosa/lojaebacdb)
  + Loja EBAC: [ernestosbarbosa/lojaebac](https://hub.docker.com/repository/docker/ernestosbarbosa/lojaebac)
    - Comandos para subir os containers:

*docker network create --attachable ebac-network*

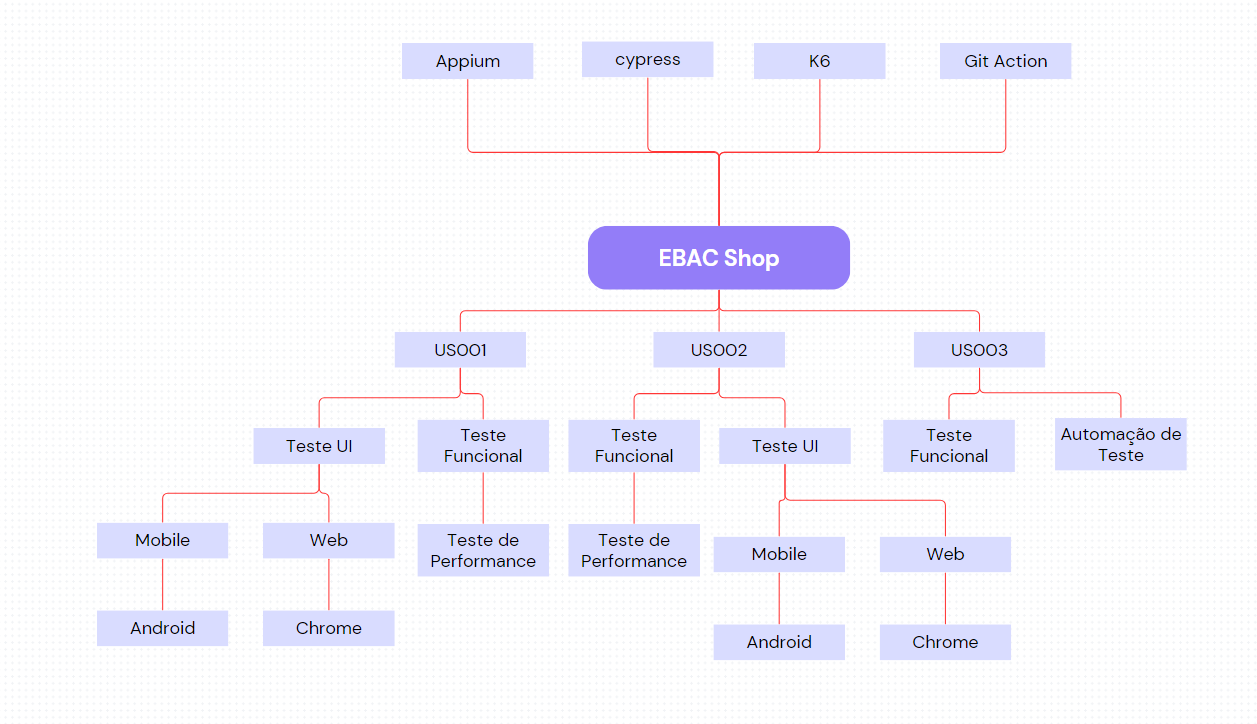
*docker run -d --name wp\_db -p 3306:3306 --network ebac-network ernestosbarbosa/lojaebacdb:latest*

*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*

## 4.1 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



## 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;
* Crie histórias de usuário para as funcionalidades:
  + Catálogo de Produtos
  + Painel Minha Conta
  + Meus Pedidos
  + Endereços
  + Detalhes da Conta
* Referência: Módulo 8

**4.2.1 Histórias de usuário - Painel Minha Conta**

**Como** cliente da EBAC-SHOP

**Quero** acessar Painel "**Minha Conta**" usando e-mail e senha

**Para** acessar recursos personalizados.

**4.2.2 Histórias de usuário – Catálogo de Produtos**

**Como** cliente da EBAC-Shop

**Quero** visualizar a lista de produtos disponíveis no catálogo

**Para** ver os itens que posso comprar.

**4.2.3 Histórias de usuário - Detalhes da Conta**

**Como** cliente da EBAC-SHOP

**Quero** visualizar informações da conta na página "Detalhes da Conta"

**Para** atualizar nome, sobrenome, e-mail e senha de acesso.

**4.2.4 Histórias de usuário - Meus Pedidos**

**Como** cliente da EBAC-SHOP

**Quero** visualizar uma lista dos meus pedidos "Meus Pedidos”

**Para** obter o status de cada pedido.

**4.2.5 Histórias de usuário - Endereços**

**Como** cliente da EBAC-SHOP

**Quero** editar as informações do meu endereço em "Meu Endereço"

**Para** atualizar nome, rua, número, CEP, cidade e estado.

## 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

**4.3.1 - Caso de teste: Adicionar item ao carrinho**

**RN01**

Não é permitido inserir mais de 10 itens de um mesmo produto no carrinho

|  |  |
| --- | --- |
| 0 até 10 | >10 |
| 0,1,4,7,9,10 | 11,22,85,92, ... |
| Dados válidos | Dados Inválidos |

**Caso de teste**

|  |  |  |
| --- | --- | --- |
|  | Entrada | Saída |
| Teste 1 | Inserir 10 itens de um mesmo produto ao carrinho | Válido |
| Teste 2 | Inserir mais de 10 itens de um mesmo produto ao carrinho | Inválido |
| Teste 3 | Inserir menos de 10 itens de um mesmo produto ao carrinho | Válido |

**RN02**

Os valores não podem ultrapassar a R$ 990,00

|  |  |
| --- | --- |
| 0 até 899,99 | > 990,00 |
| 0,10, 20 ,….., 90, 100,…., 899,99 | 990,00 1.000,…..,999.99,….. |
| Dados Válidos | Dados Inválido |

**Caso de teste**

|  |  |  |
| --- | --- | --- |
|  | Entrada | Saída |
| Teste 01 | Total do valor de R$ 899,99 | válido |
| Teste 02 | Total do valor de R$ 1.000,00 | inválido |
| Teste 03 | Total do valor de R$ 60,00 | válido |

**NR03**

Valores entre R$ 200 e R$ 600 , ganham cupom de 10%

|  |  |  |
| --- | --- | --- |
| < 199,99 | 200,00 até 600,00 | > 600 |
| 0,10,00 ,20,...,100,…, 199, | 200,00,300,000,..., 600,000 | 601,800,...900,…, |
| Dados Inválido | Dados Válidos | Dados Inválidos |

**Caso de teste**

|  |  |  |
| --- | --- | --- |
|  | Entrada | Saída |
| Teste 1 | Total do valor de R$ 200,00 | Inválido |
| Teste 2 | Total do valor de R$ 600,50 | Válido |
| Teste 3 | Total do valor de R$ 150,00 | Válido |

**RN04**

Valores acima de R$ 600 ganham cupom de 15%

|  |  |
| --- | --- |
| 0 até 599,99 | > 99,00 |
| 0,10,20,….,100,…,500,…,599,98 | 990,00,1.000,000,... |
| Dados inválidos | Dados válidos |

**Caso de teste**

|  |  |  |
| --- | --- | --- |
|  | Entrada | Saída |
| Teste 1 | Total do valor R$ 1.500,00 | Válido |
| Teste 2 | Total do valor R$ 50,00 | Inválido |
| Teste 3 | Total do valor R$599,98 | Válido |

**4.3.2 – Caso de teste: Cadastro na plataforma**

**RN01**

Cadastro de alunos deve estar entre 16 e 80 anos.

|  |  |  |  |
| --- | --- | --- | --- |
| Condições | Regra 1 | Regra 2 | Regra 3 |
| Idade | < 16 | 16 a 18 | > 80 |
| Ação |  |  |  |
| Cadastrar |  | X |  |
| Não | X |  | X |

**RN02**

Caso erre a senha por 3 vezes, deve bloquear o acesso por 15 minutos.

Caso de teste

|  |  |  |
| --- | --- | --- |
| Condição | Regra 1 | Regra 2 |
| Trava acesso por tentativa | =<2 | >=3 |
| Ação |  |  |
| Bloqueio |  | X |
| Não bloqueado | X |  |

**RN03**

Exibir mensagem de erro caso o usuário erre login e senha.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Condição | Regra 1 | Regra 2 | Regra 3 | Regra 4 |
| Usuário válido? | Sim | Sim | Não | Não |
| Senha válida? | Sim | Não | Sim | Não |
| Ação |  |  |  |  |
| Acesso | Sim | Não | Não | Não |

**Casos de testes:**

1. Login com usuário válido e senha válida, deve permitir acesso

2. Login com usuário válido e senha inválida, não deve permitir acesso

3. Login com usuário inválido e senha válida, não deve permitir acesso

4. Login com usuário inválido e senha inválida, não deve permitir acesso

## 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 o código fontes das automações que criar.
* Referência: Módulo 10
* Link do repositório: https://github.com/JoaoKleinkauf/TCC-EBAC-QE

## 4.5 Testes automatizados

4.5.1 Automação de UI

Ao considerar ferramentas de automação para nosso projeto, é fundamental selecionar uma que ofereça velocidade de execução, facilidade de manutenção de código e compatibilidade com integração contínua. Após uma análise detalhada, optei pelo Cypress como ferramenta de automação. Abaixo, descrevo os motivos dessa escolha:

API Simples e Fácil Manutenção:

A API do Cypress é intuitiva e direta, reduzindo a complexidade na criação de testes. Comparado com ferramentas como Selenium ou Playwright, o Cypress simplifica a escrita e manutenção de testes, evitando a necessidade de comandos excessivamente complexos.

Código fácil:

O Cypress se destaca por sua abordagem amigável e facilidade de uso. Esta característica acelera o desenvolvimento dos testes, permitindo que a equipe economize tempo valioso. A curva de aprendizado do Cypress é menor, o que facilita a adoção por novos membros da equipe e aumenta a eficiência geral.

Execução prática:

A estrutura nativa do Cypress proporciona uma vantagem significativa em termos de velocidade de execução. Em comparação com o Selenium, o Cypress executa testes de forma mais rápida e eficiente, permitindo realizar testes de regressão e validação com maior agilidade. Essa rapidez é crucial para manter um ciclo de desenvolvimento ágil e responsivo.

Integração Contínua:

O Cypress oferece integração nativa com várias ferramentas de CI/CD, como Jenkins, CircleCI e GitHub Actions. Isso facilita a incorporação dos testes automatizados no pipeline de entrega contínua, garantindo que a qualidade do software seja verificada continuamente em cada etapa do desenvolvimento. A integração contínua com o Cypress assegura que qualquer problema seja identificado e corrigido rapidamente, mantendo a estabilidade e a confiabilidade do produto.

Conclusão:

A escolha do Cypress foi motivada por sua simplicidade, rapidez e compatibilidade com ferramentas de integração contínua. Essas características fazem do Cypress a opção ideal para nosso projeto, permitindo-nos desenvolver e manter testes de alta qualidade de maneira eficiente e eficaz. Com o Cypress, podemos garantir que nossos testes automatizados sejam robustos e mantenham a qualidade do software durante todo o ciclo de desenvolvimento.

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.
* Você deve implementar testes para ao menos uma das plataformas Mobile (*Android* ou *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
* 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

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

# 5. CONCLUSÃO

Durante este curso, adquiri conhecimentos e habilidades fundamentais para atuar na área de Teste de Software, especialmente no desenvolvimento de aplicativos web e mobile. Compreendi a importância do planejamento e da estratégia de testes para assegurar a qualidade do software.

Aprendi a elaborar um plano de testes abrangente, considerando objetivos, papéis e responsabilidades, fases de testes, padrões, tipos de testes, técnicas, ambientes e ferramentas. A importância dos critérios de aceitação também ficou clara, auxiliando na definição e validação das funcionalidades do software.

Tive a oportunidade de praticar a criação de casos de teste, utilizando técnicas como partição de equivalência e valor limite. Identifiquei os casos que poderiam ser automatizados, explorei abordagens de automação para testes de UI, API e mobile.

Compreendi a relevância da integração contínua, que permite a execução automatizada dos testes em um ambiente de desenvolvimento colaborativo, proporcionando maior confiabilidade e agilidade na entrega do software.

Também explorei testes de performance com a ferramenta K6, avaliando o comportamento do software em condições de uso intenso, garantindo sua estabilidade e escalabilidade.

Ao concluir este curso, adquiri não só conhecimento teórico, mas também experiência prática. Agora, como profissional atuante no mercado, posso aplicar essas habilidades e boas práticas, contribuindo para a qualidade e sucesso dos projetos de software nos quais estou envolvido.

Em resumo, este curso foi crucial para meu crescimento profissional na área de Teste de Software, fornecendo as bases teóricas e práticas necessárias para desempenhar um papel eficaz no desenvolvimento de software de qualidade. Estou confiante de que as habilidades adquiridas serão valiosas ao longo da minha carreira.

# 6. REFERÊNCIAS BIBLIOGRÁFICAS

**Curso de Teste de Software online – aprenda com EBAC Online.**

Disponível em: <https://ebaconline.com.br/qualidade-de-software?utm_source=google&utm_medium=cpc&utm_campaign=ccourse_44_sowtware-testing_google_search_all_conversions_all&utm_content=c_14670107930>

**Qualidade de software: o que é e como avaliar o seu resultado?**

Disponível em: [https://www.monitoratec.com.br/blog/qualidade-de-software/#:~:text=O que é qualidade de software%3F](https://www.monitoratec.com.br/blog/qualidade-de-software/" \l ":~:text=O que é qualidade de software%3F)

# **Teste de software: o que é, importância e como fazer?**

Disponível em: <https://blog.betrybe.com/tecnologia/teste-software-guia/>

# Teste de Software: entenda o que é e a importância do procedimento

Disponível em: <https://enotas.com.br/blog/teste-de-software/>