

**ENGENHEIRO DE QUALIDADE DE SOFTWARE**

Lucas Di Franco Linden

Análise de Qualidade

Taquara

2023

# RESUMO

Aqui você deve fazer um breve resumo do seu projeto. Aborde um pouco de tudo, mas não entre profundamente em nada. O “resumo” em um trabalho acadêmico “serve” para mostrar ao leitor se o conteúdo é de seu interesse ou não. Mas é um resumo, um breve relato de, no máximo 200 palavras.

# SUMÁRIO

[1. RESUMO 2](#_Toc99483086)

[2. SUMÁRIO 3](#_Toc99483087)

[3. INTRODUÇÃO 4](#_Toc99483088)

[4. O PROJETO 5](#_Toc99483089)

[4.1 Estratégia de teste 6](#_Toc99483090)

[4.2 Critérios de aceitação 6](#_Toc99483091)

[4.3 Casos de testes 6](#_Toc99483092)

[4.4 Repositório no Github 7](#_Toc99483093)

[4.5 Testes automatizados 7](#_Toc99483094)

[4.6 Integração contínua 8](#_Toc99483095)

[4.7 Testes de performance 8](#_Toc99483096)

[5. CONCLUSÃO 9](#_Toc99483097)

[6. REFERÊNCIAS BIBLIOGRÁFICAS 9](#_Toc99483098)

# INTRODUÇÃO

Na introdução você deve fazer um apanhado geral do seu cenário para o leitor. Escreva a prévia do que teríamos no trabalho, o que irá tratar, o que espera do projeto, etc.

# 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*](http://localhost:80/)

* Como este trabalho complementa o que criou em seu Trabalho de Consolidação (Módulo 19), você pode utilizá-lo como base para o seu Trabalho de Conclusão.

## 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:

[Imagem: Mapa mental – Estratégia de teste]

## Critérios de aceitação

[US-0001] – Adicionar item ao carrinho:

https://github.com/LLinden/TCC-EBAC/blob/main/Crit%C3%A9rio%20de%20aceita%C3%A7%C3%A3o%20USs/%5BUS-0001%5D%20%E2%80%93%20Adicionar%20item%20ao%20carrinho.feature

[US-0002] – Login na plataforma

https://github.com/LLinden/TCC-EBAC/blob/main/Crit%C3%A9rio%20de%20aceita%C3%A7%C3%A3o%20USs/%5BUS-0002%5D%20%E2%80%93%20Login%20na%20plataforma.feature

[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

## 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

## 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 os código fontes das automações que criar.
* Referência: Módulo 10
* Link do repositório: <cole o link aqui>

## Testes automatizados

* + 1. Automação de UI
* Crie um projeto de automação WEB com o framework e a linguagem que preferir
* Justifique a sua escolha através de um comparativo entre ao menos 3 opções de ferramentas e linguagem.
* Crie uma pasta chamada UI para os testes WEB dos casos de teste que forem automatizados
* Utilize ao menos um *Testing Pattern* (à sua escolha) na implementação dos testes.
  + 1. 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!
  + 1. 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

## Integração contínua

* Execute os testes automatizados em integração contínua utilizando o Github Actions
* Referência: Módulo 26

## 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

# 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