ENGENHEIRO DE QUALIDADE DE SOFTWARE

Matheus Gonçalves Dos Santos Silva

Análise de Qualidade

2024 – Rio de Janeiro

SUMÁRIO

RESUMO

Este projeto de conclusão de curso tem como objetivo principal a elaboração de uma estratégia de testes abrangente para validar o e-commerce EBAC Shop, aplicando os conhecimentos adquiridos ao longo do curso de Profissão: Engenheiro de Qualidade de Software. A proposta é simular a participação em um time ágil, seguindo todo o fluxo de trabalho de um Quality Engineer (QE), desde o planejamento até a entrega das funcionalidades.

INTRODUÇÃO

No decorrer deste trabalho, serão abordados diversos aspectos da engenharia de qualidade, incluído a definição dos critérios de aceitação para as estórias de usuários, a criação dos casos de testes detalhados, implementação dos testes automatizados Web, API e Aplicativos móveis.
A abordagem adotada combina técnicas de testes manuais e automatizadas, utilizando ferramentas modernas e padrões de mercado para garantir a eficácia dos testes.
O projeto também inclui a configuração de um pipeline de integração contínua para executar os testes automatizados e a implementação de testes de performance para assegurar que o sistema atenda aos requisitos de desempenho esperados. Assim, este trabalho não apenas reforça os conhecimentos adquiridos.

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
    - Loja EBAC: 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

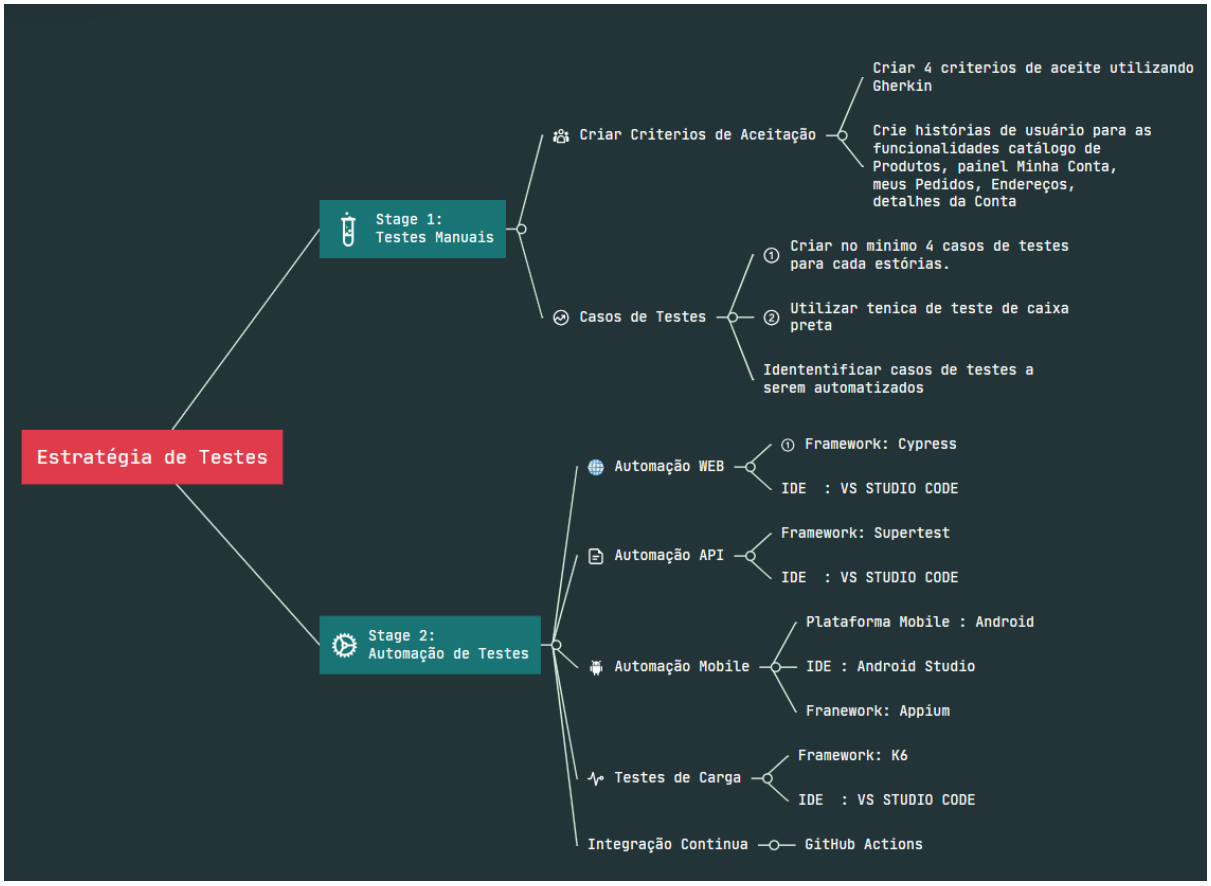
docker run -d --name wp -p 80:80 --network ebac-network ernestosbarbosa/lojaebac:latest

Após subir os containers a loja estará em http://localhost:80

- 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:



Critérios de aceitação

- Considere as histórias de usuário:
    - [US-0001] – Adicionar item ao carrinho

  Funcionalidade: Adicionar item ao carrinho

  Cenário: Validação de Seleção de Tamanho e Cor para adicionar ao carrinho

  Dado que o usuário acesse a loja da EBAC SHOP
  Quando selecionar o produto que deseja comprar
  E clicar sobre o botão "Comprar"
  E não selecionar "Size" e "Color"
  Então o sistema deve exibir uma mensagem de toast: "Selecione uma das opções do produto antes de adicioná-lo ao carrinho."

Cenário: Adicionar um produto ao carrinho

Dado que o usuário acesse a loja da EBAC SHOP
Quando selecionar o produto que deseja comprar
E selecionar "Size" e "Color"
E clicar sobre o botão "Comprar"
Então o sistema deve exibir uma mensagem de toast: "<nome do produto> foi adicionado ao seu carrinho."

Cenário: Validação de limite de carrinho

Dado que o usuário acesse a loja da EBAC SHOP
Quando selecionar o produto que deseja comprar
selecionando "Size" e "Color"
E clicar sobre o botão "Comprar"
E clicar sobre o botão "Ver Carrinho"
Então o sistema deve redirecionar o usuário para o carrinho exibindo o produto selecionado

Cenário: Quantidade de produtos limitada ao estoque

Dado O usuário acesse o site EBAC Shop e adicione um produto ao carrinho
Quando a quantidade total do produto no carrinho ultrapassar o estoque disponível
Então o sistema deve exibir uma mensagem de alerta informando a quantidade máxima permitida para compra.

- [US-0002] – Login na plataforma

Funcionalidade: Login

Cenário: Realizar o login com sucesso

Dado que o usuário acesse a loja da EBAC SHOP para realizar o login
Quando informar email e senha corretamente já cadastrado anteriormente na plataforma
E clicar sobre o botão "Login"
Então o sistema deve realizar a verificação do email e senha informados e conceder o

acesso do mesmo a plataforma

Cenário: Exibir mensagem de erro para login incorreto

Dado que o usuário acesse a loja da EBAC SHOP para realizar o login
E informe um email ou senha incorretos
Quando clicar sobre o botão "Login"
Então o sistema deve exibir a mensagem de erro "Email ou senha incorretos. Por favor, tente novamente."

Cenário: Bloquear login após 3 tentativas falhas

Dado que o usuário tente fazer login com email ou senha incorretos 3 vezes consecutivas
Quando clicar sobre o botão "Login" na terceira tentativa
Então o sistema deve exibir a mensagem de erro "Você excedeu o número máximo de tentativas de login."

Cenário: Realizar o login após período de bloqueio

Dado que o período de bloqueio tenha expirado
Quando o usuário tentar fazer login novamente
Então o sistema deve permitir novas tentativas de login resetando o contador de tentativas de login falhadas.

- [US-0003] – API de cupons

Funcionalidade: API de cupons

Cenário: Listar todos os cupons

Dado que o admin esteja autenticado na API
Quando fizer uma requisição GET para listar todos os cupons
Então o sistema deve retornar uma lista de todos os cupons cadastrados

Cenário: Listar cupons por ID

Dado que o admin esteja autenticado na API
Quando fizer uma requisição GET com um ID específico
Então o sistema deve retornar os detalhes do cupom correspondente ao ID

fornecido

Cenário: Cadastrar um novo cupom

Dado que o admin esteja autenticado na API
Quando fizer uma requisição POST com os campos obrigatórios
Então o sistema deve cadastrar o novo cupom

Cenário: Código do cupom já existente

Dado que o admin esteja autenticado na API
Quando fizer uma requisição POST com o código do cupom "Ganhe10"
E o código do cupom já exista
Então o sistema deve retornar um erro indicando que o código do cupom já está em uso

- 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

Como usuário do site EBAC-SHOP,
Quero adicionar produtos a lista de desejos “My wishlist”
Para que eu possa encontrar, comparar e selecionar facilmente os itens que desejo comprar no futuro.

CT01 – Acessar a página de compras e adicionar um produto a lista de compras
CT02 – Selecionar um produto com cor diferente e adicionar a lista de desejos
CT03 – Adicionar o mesmo produto que já foi adicionado
CT04 – Remover o produto da lista de desejos

    - Painel Minha Conta

Como um usuário registrado no EBAC-SHOP,
Quero acessar o Painel Minha Conta,
Para ver minhas compras recentes, gerenciar meus endereços de entrega e faturamento, e editar minha senha e detalhes da conta.

CT01 – Acessar o painel minha conta
CT02 – Validar se está sendo exibido os menus painel, pedidos, downloads, endereços, detalhes e sair
CT03 – Validar opção “Sair”

CT04 – Validar se o nome do usuário está sendo exibido no texto de boas vindas

- Meus Pedidos

Como um usuário registrado no EBAC-SHOP,
Quero acessar a seção "Meus Pedidos",
Para visualizar o histórico de minhas compras, incluindo detalhes como o número do pedido, data, status, total de itens e valor, além de poder visualizar os detalhes de cada pedido específico.

CT01 – Acessar os pedidos e validar se os produtos estão listados
CT02 – Acessar os pedidos e visualizar os detalhes do pedidos
CT03 – Acessar os pedidos e validar os botões “Próximo” e “Anterior”
CT04 – Acessar os pedidos e validar os status

- Endereços

Como um usuário registrado no EBAC-SHOP,
Quero acessar e editar meus endereços de cobrança e entrega,
Para garantir que minhas compras sejam enviadas para o endereço correto e que minhas faturas sejam cobradas no endereço apropriado.

CT01 – Acessar Endereços e validar o preenchimento
CT02 – Acessar Endereços e validar a atualização de endereço
CT03 – Acessar Endereços e validar os campos obrigatórios
CT04 – Validar a edição do endereço de faturamento

- Detalhes da Conta

Como um usuário registrado,
Quero acessar a seção “Detalhes da Conta”,
Para atualizar minhas informações pessoais e senha.

CT01 – Acessar Detalhes da conta, alterar os dados nome e email
CT02 – Realizar a atualização da senha
CT03 – Validar o não preenchimento de campos obrigatórios

CT04 – Ao atualizar a senha informando a senha atual incorreta

- 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 público 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: MathewsZione/TCC-EBAC-QE: TCC-EBAC-QE (github.com)

Testes automatizados

4.1.1 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.

  Para este projeto eu escolhi o Cypress por quê:
  - É de fácil configuração
  Para realizar a instalação das dependências basta dar um simples "npm install", enquanto linguagens como Selenium e Playwright requer uma instalação mais complexa, incluindo várias bibliotecas.

- Tem uma ótima integração com ecossistema JavaScript
O Cypress foi desenvolvido especificamente para aplicações JavaScript modernas (React, Angular, Vue.js), oferecendo uma API intuitiva e poderosa para desenvolvedores JavaScript. Já Selenium e Playwright, suportam várias linguagens de programação, mas a integração não é tão otimizada para o ecossistema JavaScript.

- Documentação
Cypress: Possui uma documentação clara, extensa e bem-organizada, além de uma comunidade ativa que contribui com plugins e suporte.
Já Selenium e Playwright, a documentação de ambas são vastas, mas pode ser complexa e fragmentada devido ao suporte a múltiplas linguagens e à longa história do projeta. Apesar da documentação boa e crescente, ainda são tão maduras quanto a do Cypress.

- 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.

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!

4.1.2 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

CONCLUSÃO

Em resumo, o projeto consolidou meu conhecimento teórico e prático, essencial para minha formação como Engenheiro de Qualidade de Software. A aplicação de melhores práticas de engenharia de qualidade, a integração de processos contínuos de teste e a validação de desempenho do sistema demonstraram a importância de uma abordagem completa para assegurar a excelência no desenvolvimento de software. As habilidades e conhecimentos adquiridos serão valiosos em minha

carreira futura, permitindo-me contribuir de forma significativa para a qualidade dos produtos de software.

REFERÊNCIAS BIBLIOGRÁFICAS

Seguir regras ABNT