

**ENGENHEIRO DE QUALIDADE DE SOFTWARE**

Lucas Santos Evangelista

Análise de Qualidade

Salvador

2024

# RESUMO

Este trabalho de conclusão de curso tem por objetivo reafirmar o aprendizado obtido durante o curso. Foi criado um planejamento de testes e aplicamos na prática a maior parte das automações, abrangendo testes de UI, teste de API, testes mobile e teste de performance. Utilizou-se o github actions para automatizar a pipeline do CI/CD e executar os testes automaticamente para garantir uma melhor qualidade do processo de entrega do produto.

# 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

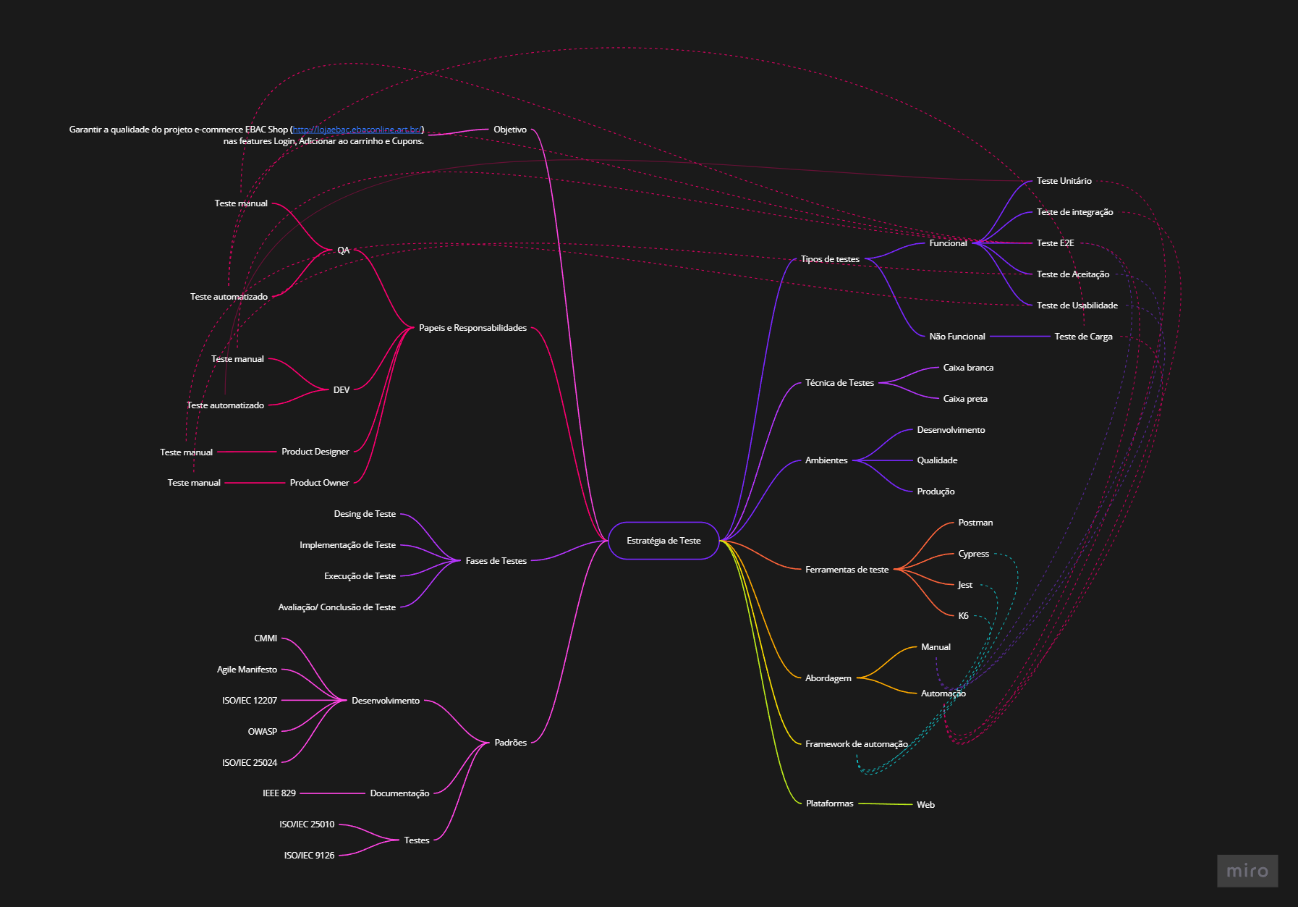
Neste projeto criaremos uma estratégia de testes que será utilizada como base para a garantia de qualidade de um E-Commerce. Para a concepção dessa estratégia de teste, utilizaremos um mapa mental que possui diretrizes que darão suporte ao ciclo de vida de desenvolvimento e teste. As histórias de usuários predefinidas e os critérios de aceitação irão guiar a criação dos casos de testes que darão suporte o desenvolvedor e o analista de qualidade durante a produção do software.

Como principal plataforma, utilizaremos o GitHub para compartilhamento do código, e, para cada automação utilizaremos diferentes framework e testing patterns. Ao final do projeto, temos como expectativa garantir que a entrega do produto seja feita dentro do prazo definido e em conformidade com as diretrizes de qualidade.

# O PROJETO

Para este trabalho de conclusão de curso, utilizamos os conhecimentos obtidos ao longo do curso para elaborar uma estratégia de testes que servirá de apoio para validar o e-commerce da EBAC Shop, hospedado no link <http://lojaebac.ebaconline.art.br/>. Utilizaremos as histórias de usuário já refinadas e iremos seguir todo o fluxo de trabalho de um Engenheiro de Qualidade, perpassando por todo o ciclo de vida do produto. Garantiremos a qualidade utilizando ferramenta de testes automatizadas como **Playwright**, Supertest, Appium com WebdriveIO e para teste de performance utilizamos o K6. Os testes foram integrados com o Allura report para melhor visualização dos resultados e foram integrados ao CI para que pudessem ser executados todas as vezes que um pull request fosse aberto para a main Branch.

## Estratégia de teste

A estratégia de teste foi criada em formato de mapa mental utilizando a ferramenta Miro. O mapa segue as seguintes diretrizes: Objetivos, Papeis e Responsabilidades, Fases de testes, Padrões, Tipos de testes, Técnica de testes, Ambientes, Ferramentas usadas e plataformas. Esse mapa dará suporte os envolvidos no processo de ideação e desenvolvimento do produto na garantia de qualidade.

## Critérios de aceitação e História de Usuário

Neste item tratamos as histórias de usuários e critérios de aceitação pré-definidos. Os mesmos foram utilizados para dar suporte a criação do item 4.3 Casos de Testes. Para as histórias pré-definidas criamos quatro critérios de aceitação usando a linguagem Gherkin. Para as funcionalidades Catálogo de Produtos, Painel Minha Conta, Meus Pedidos, Endereços, Detalhes da Conta definimos ao menos uma história de usuário.

Critérios de Aceitação:

[US-0001] – Adicionar item ao carrinho

* Adicionar mais de 10 itens de um mesmo produto ao carrinho
  + Dado que o carrinho está vazio
  + Quando adiciono 11 itens de um mesmo produto ao carrinho
  + Então a aplicação deve exibir uma mensagem de erro informando que não é possível comprar mais de 10 itens iguais
* Adicionar mais de 990 reais em itens no carrinho
  + Dado que o carrinho está vazio
  + Quando adiciono vários produtos ao carrinho
  + E o valor ultrapassa 990 reais
  + Então a aplicação deve exibir uma mensagem de erro informando que não é possível comprar mais de 990 reais em itens
* Adicionar valores maiores ou iguais 200 e menores ou iguais 600 reais em produtos no carrinho
  + Dado que o carrinho está vazio
  + Quando adiciono vários produtos ao carrinho
  + E o valor é maior ou igual a 200 e menor ou igual a 600 reais
  + Então a aplicação deve permitir comprar os itens
  + E deve inserir um cupom de desconto no valor de 10%
* Adicionar valores maior que 600 reais e menor ou igual a 990
  + Dado que o carrinho está vazio
  + Quando adiciono vários produtos ao carrinho
  + E o valor é maior que 600 e menor ou igual a 990 reais
  + Então a aplicação deve permitir comprar os itens
  + E deve inserir um cupom de desconto no valor de 15%

[US-0002] – Login na plataforma

* Fazer login com usuário inativo
  + Dado que eu esteja na tela de usuário
  + E ele esteja inativo
  + Quando faço login com esse usuário
  + Então a aplicação não deve realizar o login
* Fazer login com usuário ativo
  + Dado que eu esteja na tela de usuário
  + E ele esteja ativo
  + Quando faço login com esse usuário
  + Então a aplicação deve realizar o login com sucesso
* Fazer login com usuário ou senha incorreto até 3 vezes
  + Dado que eu esteja na tela de usuário
  + Quando faço login com usuário utilizando informações incorretas
  + Então a aplicação não deve concluir o login
  + E deve exibir um erro de dados incorretos
* Fazer login com senha incorreta por mais de 3 vezes
  + Dado que eu esteja na tela de usuário
  + E tenha tentando login por 3 vezes sem sucesso
  + Quando faço login com usuário válido e ativo utilizando informações senha incorreta
  + Então a aplicação não deve concluir o login
  + E deve exibir um erro de senha incorreta
  + E exibir uma mensagem de bloqueio de login por 15 minutos

[US-0003] – API de cupons

* Listar todos os cupons cadastrados
  + Dado que eu tenha cupons cadastrados
  + Quando faço requisição GET na api de cupons
  + E não insiro a ID do cupom
  + Então a api deve retornar uma lista com todos cupons cadastrados
* Procurar cupom por ID
  + Dado que eu tenha cupons cadastrados
  + Quando faço requisição GET na api de cupons
  + E insiro a ID do cupom
  + Então a api deve retornar o cupom referente a ID que solicitei
* Cadastrar cupom inserindo campo nome nunca cadastrado
  + Dado que eu tenha inserido todos os campos obrigatórios
  + Quando eu realizo a requisição POST na api de cupons
  + Então a aplicação deve criar um novo cupom
* Cadastrar cupom sem campos obrigatórios
  + Dado que eu não tenha inserido todos os campos obrigatórios
  + Quando eu realizo a requisição POST na api de cupons
  + Então a aplicação não deve criar um novo cupom

Histórias de Usuários

* Catálogo de Produtos
  + Como cliente da EBAC-SHOP
  + Quero visualizar os produtos para venda
  + Para realizar minhas compras
* Painel Minha Conta
  + Como cliente cadastrado na EBAC-SHOP
  + Quero visualizar o painel “Minha conta”
  + Para realizar alterações de dados pessoais
* Meus Pedidos
  + Como cliente da EBAC-SHOP
  + Quero visualizar a lista de meus pedidos
  + Para ter maior controle de quais produtos já comprei
* Endereços
  + Como cliente da EBAC-SHOP
  + Quero visualizar meus endereços cadastrados
  + Para identificar se os mesmos estão corretos
  + E poder altera-los caso necessário
* Detalhes da Conta
  + Como cliente da EBAC-SHOP
  + Quero visualizar os detalhes da minha conta
  + Para poder fazer as alterações quando necessário

## Casos de testes

Para dar suporte aos testes funcionais e automatizados, criamos ao menos quatro casos de testes para cada história de usuário que foi pré-definida no item 4.2. Os casos foram criados com base nas técnicas de testes aprendidas no curso. Obrigatoriamente o casos tem o fluxo principal e um cenário negativo. Todos os casos possuem as seguintes informações: Código do caso de teste, título, pré-condições para execução do teste, passos para serem executados, resultado esperado e a marcação se devem ou não ser automatizados.

**Adicionar item ao carrinho**

|  |  |
| --- | --- |
| CT001 – Adicionar 9 itens iguais ao carrinho | |
| Pré-condição | * + - Existir produtos cadastrados |
| Passos: | * + - 1. Acesse a página de listagem de produtos       2. Selecione algum produto para compra       3. Defina as especificações do produto como: tamanho e cor       4. Defina a quantidade para 9       5. Clique para comprar |
| Resultado esperado | O produto deve ser adicionado ao carrinho |
| Automatizar | Não |

|  |  |
| --- | --- |
| CT002 – Adicionar 10 itens iguais ao carrinho | |
| Pré-condição | * + - Existir produtos cadastrados |
| Passos: | * + - 1. Acesse a página de listagem de produtos       2. Selecione algum produto para compra       3. Defina as especificações do produto como: tamanho e cor       4. Defina a quantidade para 10       5. Clique para comprar |
| Resultado esperado | O produto deve ser adicionado ao carrinho |
| Automatizar | Não |

|  |  |
| --- | --- |
| CT003 – Adicionar 11 itens iguais ao carrinho | |
| Pré-condição | * + - Existir produtos cadastrados |
| Passos: | * + - 1. Acesse a página de listagem de produtos       2. Selecione algum produto para compra       3. Defina as especificações do produto como: tamanho e cor       4. Defina a quantidade para 11       5. Clique para comprar |
| Resultado esperado | O produto não deve ser adicionado ao carrinho e a aplicação deve exibir mensagem de erro informando que não é possível adicionar mais de 10 itens iguais ao carrinho |
| Automatizar | Não |

|  |  |
| --- | --- |
| CT004 – Adicionar R$ 990,01 em produtos no carrinho | |
| Pré-condição | * + - Existir produtos cadastrados |
| Passos: | * + - 1. Acesse a página de listagem de produtos       2. Selecione algum produto para compra       3. Defina as especificações do produto como: tamanho e cor       4. Defina a quantidade para 9       5. Clique para comprar |
| Resultado esperado | O produto deve ser adicionado ao carrinho |
| Automatizar | Não |

**Login na plataforma**

|  |  |
| --- | --- |
| CT005 – Logar com usuário válido e ativo | |
| Pré-condição | * + - Existir usuário cadastrado     - Usuário está com status ativo |
| Passos: | * + - 1. Acesse a página de login da EBAC-SHOP       2. Insira usuário existente e ativo e senha correta       3. Clique em logar |
| Resultado esperado | A aplicação deve realizar o login do usuário informado |
| Automatizar | Sim |

|  |  |
| --- | --- |
| CT006 – Logar com usuário válido e inativo | |
| Pré-condição | * + - Existir usuário cadastrado     - Usuário está com status inativo |
| Passos: | * + - 1. Acesse a página de login da EBAC-SHOP       2. Insira usuário existente porém inativo e senha correta       3. Clique em logar |
| Resultado esperado | A aplicação não deve realizar o login do usuário informado |
| Automatizar | Não |

|  |  |
| --- | --- |
| CT007 – Logar com usuário ou senha incorreto | |
| Pré-condição |  |
| Passos: | * + - 1. Acesse a página de login da EBAC-SHOP       2. Insira usuário e/ou senha incorretos       3. Clique em logar |
| Resultado esperado | A aplicação não deve realizar o login do usuário informado e exibir erro informando que as informações não estão corretas |
| Automatizar | Sim |

|  |  |
| --- | --- |
| CT008 – Logar mais de 3 vezes com senha incorreta | |
| Pré-condição | * + - Existir usuário cadastrado     - Usuário está com status ativo |
| Passos: | * + - 1. Acesse a página de login da EBAC-SHOP       2. Insira usuário existente e ativo e senha incorreta       3. Clique em logar   Repita os passos por 4 vezes |
| Resultado esperado | A aplicação não deve realizar o login do usuário informado e exibir mensagem de que o login foi bloqueado por 15 minutos |
| Automatizar | Não |

**API de cupons**

|  |  |
| --- | --- |
| CT009 – Criar cupom de desconto | |
| Pré-condição | * + - Possuir token de autorização |
| Passos: | * + - 1. Acesse a interface de execução de apis       2. Insira as informações obrigatórias da rota POST       3. Clique em executar |
| Resultado esperado | A aplicação deve criar um novo cupom e retornar as informações criadas no response e exibir código 201 |
| Automatizar | Sim |

|  |  |
| --- | --- |
| CT010 – Criar cupom de desconto com nome repetido | |
| Pré-condição | * + - Possuir token de autorização |
| Passos: | * + - 1. Acesse a interface de execução de apis       2. Insira as informações de código do cupom que já existe, defina valor, tipo de desconto e descrição no body da rota POST       3. Clique em executar |
| Resultado esperado | A aplicação não deve criar um novo cupom e retornar a informação de que já existe um cupom com esse nome e exibir código 406 |
| Automatizar | Sim |

|  |  |
| --- | --- |
| CT011 – Listar cupons de desconto | |
| Pré-condição | * + - Possuir token de autorização     - Possuir cupons já cadastrados |
| Passos: | * + - 1. Acesse a interface de execução de apis       2. Acesse a rota GET       3. Clique em executar |
| Resultado esperado | A aplicação não deve listar todos os cupons cadastrados e exibir código 200 |
| Automatizar | Sim |

|  |  |
| --- | --- |
| CT012 – Pesquisar cupom inexistente por ID | |
| Pré-condição | * + - Possuir token de autorização     - Possuir cupons já cadastrados |
| Passos: | * + - 1. Acesse a interface de execução de apis       2. Acesse a rota GET e informe a ID do cupom no path da chamada       3. Clique em executar |
| Resultado esperado | A aplicação deve exibir uma mensagem de erro de não encontrado com código 404 |
| Automatizar | Não |

## Repositório no Github

Para controle, compartilhamento e versionamento utilizaremos o GitHub e o repositório TCC-EBAC-QE.

Link do repositório: https://github.com/Lucassev05/TCC-EBAC-QE

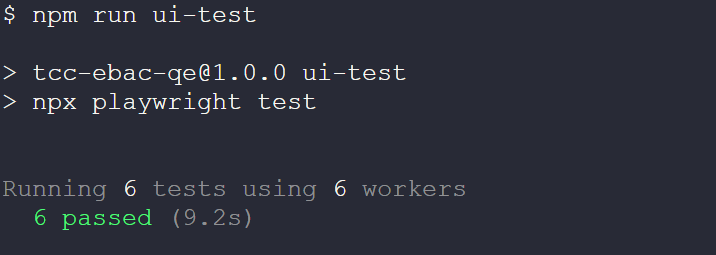
## Testes automatizados

* + 1. Automação de UI

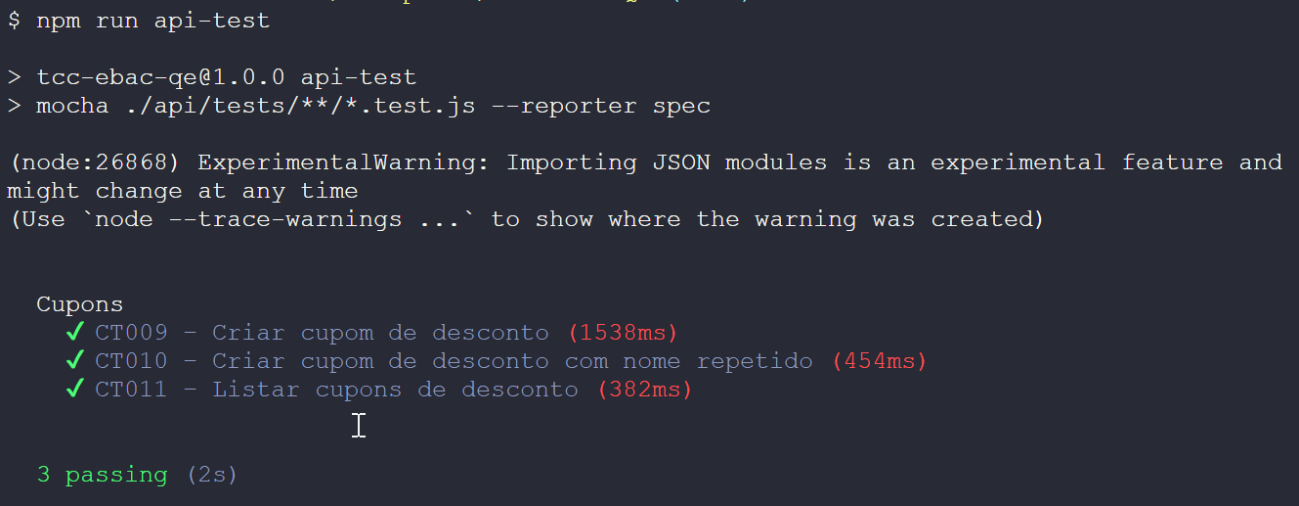
Comparativo de Ferramentas

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cypress | Playwright | Selenium |
| Facilidade de Uso/ Aprendizado | Fácil de configurar e usar | Fácil de configurar, mais atualizado. Boa documentação | Dificuldade mediana em configuração |
| Suporte a navegadores | Suporte a navegadores limitados a chromium, não permite interação com abas e janelas | Suporte a variedade de navegadores, possui suporte a abas e múltiplas janelas | Suporte abrangente de navegadores, mas para cada um precisa de uma configuração diferente. |
| Performance dos testes | Rápido | Rápido | Médio |
| Integração ecossistema | Vasto com muitos plugins | Vasto porém possui poucos plugins | Vasto |
| Comunidade e suporte | Média e em crescimento | Média e em crescimento | Grande e consolidada |

Considerando as necessidades do projeto, a escolhemos para a automação da interface do usuário (UI) o **Playwright** com **JavaScript**. Essa escolha se justifica por vários motivos importantes como abrangência de navegadores, gerenciamento de múltiplas abas e janelas e paralelismo. Com ele usaremos o Page object Pattern como testing pattern e o allura report para visualização dos resultados dos testes automatizados.

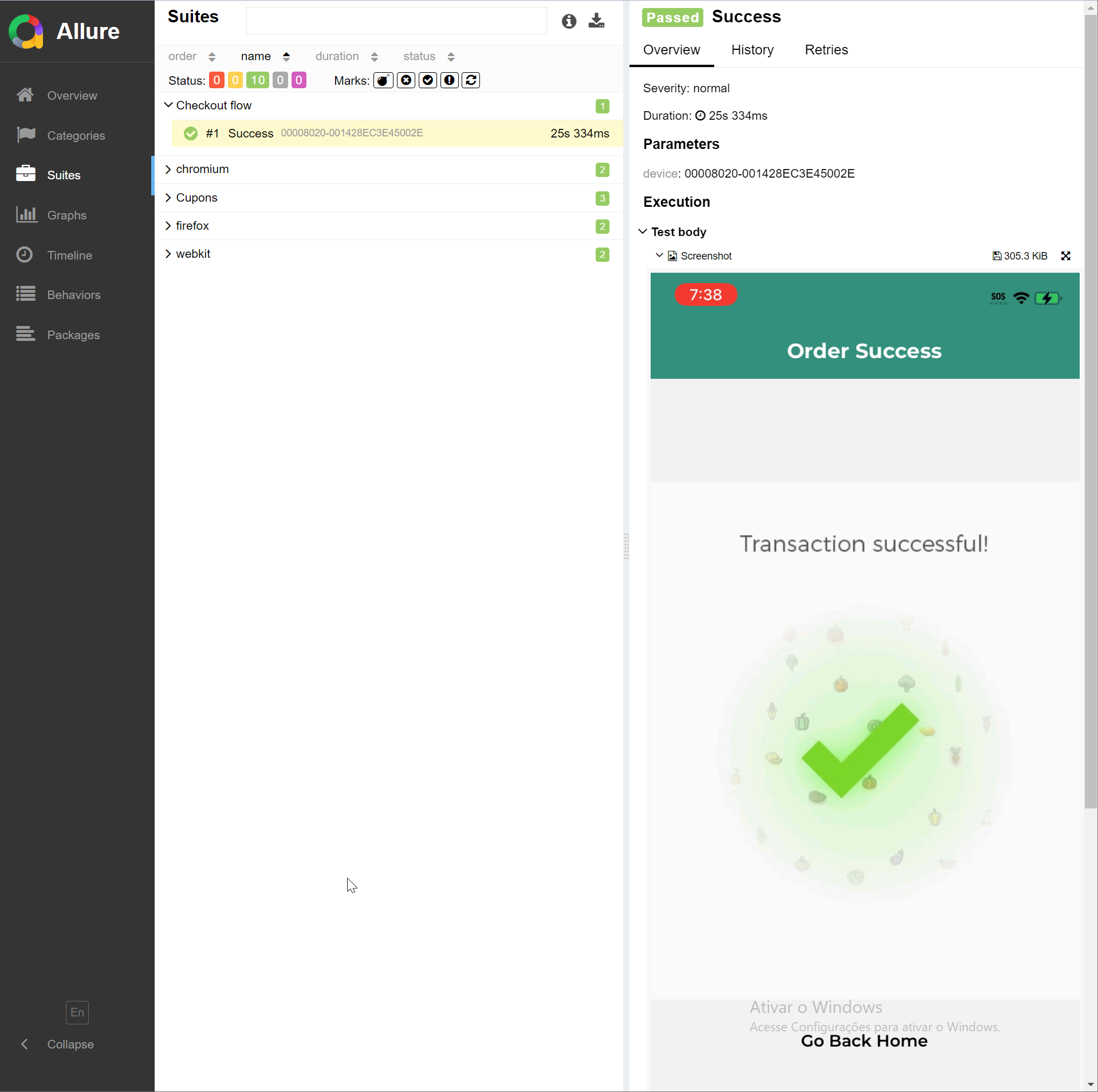


* + 1. Automação de API

Para automação da api utilizamos o supertest, mocha e chai. Para validação de contrato utilizamos o joi. Implementamos também o allura report para visualização dos resultados dos testes automatizados.

* + 1. Automação Mobile

A automação mobile foi desenvolvida para o Aplicativo iOS. Utilizamos o appium, appium inspector e webdriveio. Para execução dos testes utilizamos o saucelabs e também utilizamos o allura report para visualização dos resultados dos testes automatizados.



## Integração contínua

A integração continua foi realizada no GitHub Actions e é disparada todas as vezes que abrimos um pull request para a main Branch no github.



## Testes de performance

O teste de performance foi desenvolvido no K6, utilizando a api no endpoint de cupons. Para esse teste foi utilizado as configurações de 20 usuários virtuais, com tempo de execução de 2 minutos e RampUp de 20 segundos.



# CONCLUSÃO

Esse trabalho teve como objetivo criar uma estratégia de testes, buscando garantir a qualidade do serviço desenvolvido utilizando ferramentas de automação de testes. Ao longo do desenvolvimento do projeto, foi possível firmar os aprendizados obtidos no curso da EBAC e me promoveu a possibilidade de aprender automação em PlayWright, ferramenta que cresce exponencialmente na automação de testes UI. Apesar das limitações encontradas quanto a erros na execução dos testes e a dificuldade de gerar dois relatórios de testes ao mesmo tempo, tais limitações abriram espaço para novas pesquisas que podem abordar a configuração de múltiplos testes em um só repositório como também a criação de uma ferramenta que faça múltiplos reports.

# REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 9126-1 Engenharia de software - Qualidade de produto - Parte 1: Modelo de qualidade. 2003.