

**QUALIDADE DE SOFTWARE**

Paula Sena

Analista de Qualidade

São Paulo

2022

# RESUMO

Qualidade de software ainda é muito negligenciado. Embora traga benefícios indiscutíveis para qualquer produto ou serviço, ainda é visto como um capricho, e não como uma necessidade. Uma das principais responsabilidades do QA é prevenir bugs, ajudar o produtct Owner a modificar ou melhorar as historias de usuários, com o objetivo de tornar os requisitos mais próximos da realidade.

# 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

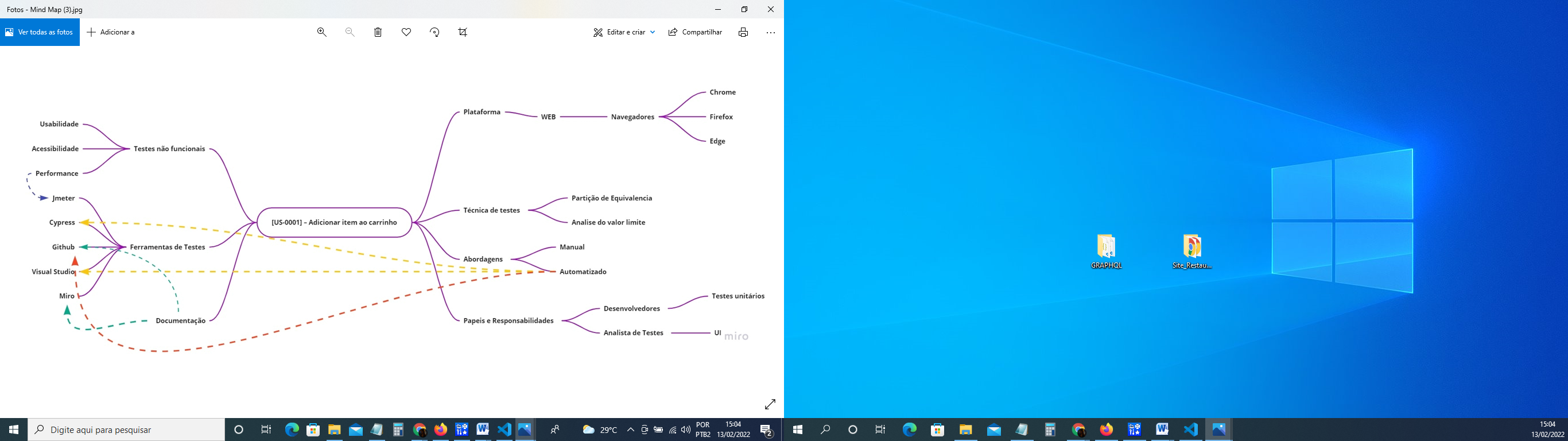
Com ciclos pequenos de sprint de duas a quatro semanas, cada integrante do time deve se manter focado na realização de suas tarefas tendo em mente cumpri-las no prazo. Os desenvolvedores devem se ocupar com o desenvolvimento das historias que lhes foram designadas. Enquanto isso, o QA deve planejar, escrever os casos de testes e automatizar os testes. Ao final de cada sprint, a equipe deve realizar uma reunião de review de sprint em que as historias do usuário finalizadas durante a sprint devem ser apresentadas ao PO e às demais partes interessadas do produto.

# O PROJETO

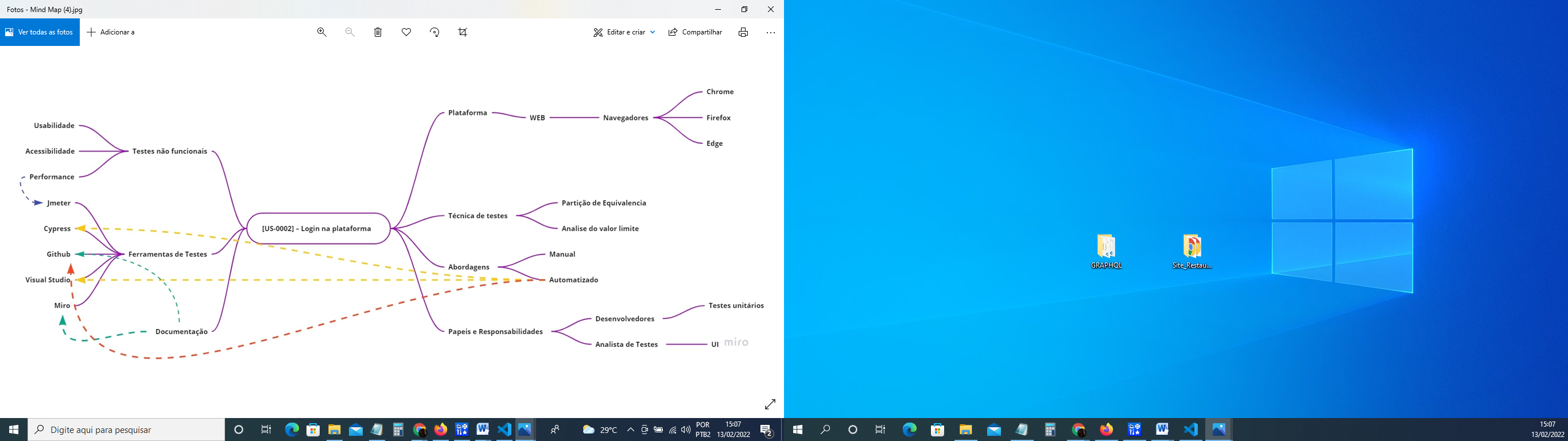
Para o Projeto mestre da primeira parte do curso Profissão: Engenheiro de Qualidade, você deve considerar as histórias de usuário já refinadas e como se você estivesse participando de um time ágil. As funcionalidades devem seguir todo o fluxo de trabalho de um QA, desde o planejamento até a entrega. 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

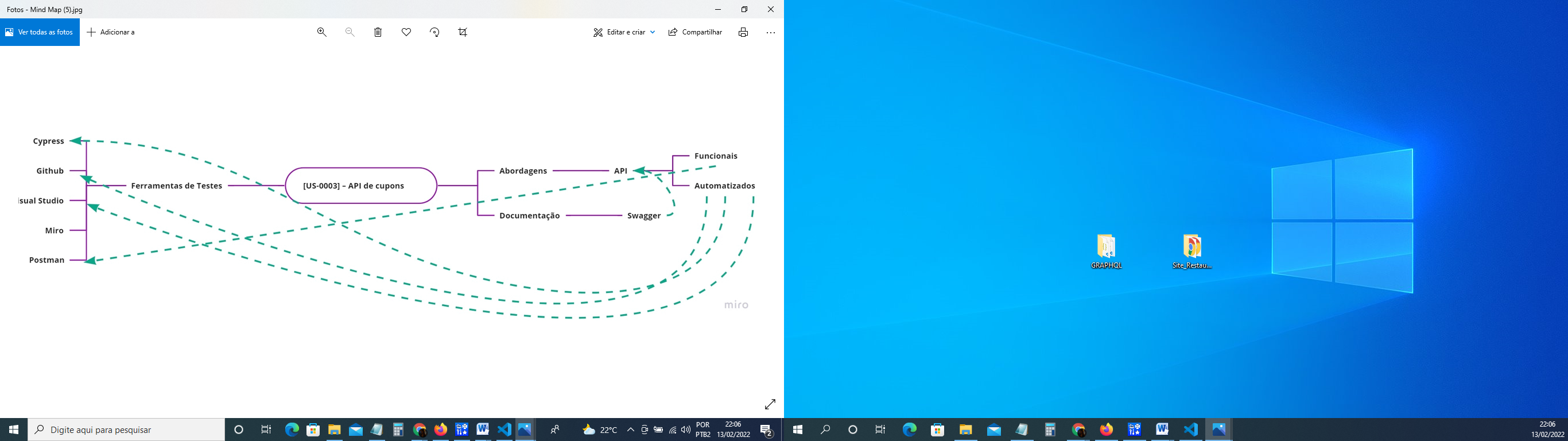
* Faça uma estratégia de testes em um mapa mental, seguindo algumas diretrizes como objetivos, papeis e responsabilidades, fases de testes, padrões, tipos de testes, técnicas de testes, ambientes, ferramentas, abordagem (manual ou automatizado), framework ou ferramenta usados, plataformas (web, api, mobile), etc.;
* Referência: Módulo 5
* Após fazer sua estratégia de teste, tire um print e cole aqui  
    
   **[US-0001] – Adicionar item ao carrinho**



**[US-0002] – Login na plataforma**



**[US-0003] – API de cupons**



## 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

Critérios de aceitação:

**Feature:** Adicionar produto no carrinho   
  
**Scenario 1:** Não deve adicionar o mesmo produto acima de dez (quantidade) ao carrinho

**Given** que estou logado na loja ebac  
**When** Seleciono o produto “15”  
**And** adiciono ao carrinho  
**Then** deve apresentar mensagem de erro ‘Atenção: Permitido até 10 itens por compra’.

**Background**:   
**Given** estou na tela de checkout

**Scenario outline 2:** Validar cupons de desconto na tela de checkout  
**When** que digito o <cupom>  
**Then** deve exibir <mensagem> do desconto

|  |  |  |
| --- | --- | --- |
| Valores | Desconto | Mensagens |
| 300 | 10% | “Você ganhou desconto de 10%” |
| 400 | 10% | “Você ganhou desconto de 10%” |
| 500 | 10% | “Você ganhou desconto de 10%” |
| 600 | 10% | “Você ganhou desconto de 10%” |
| 650 | 15% | “Você ganhou desconto de 15%” |

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

Critérios de aceitação:

**Scenario 1**: No campo login deve apresentar o “placeholder” com as mensagens “Email, nome do usuário ou CPF”

**Given** estou na loja ebac-shop  
**When** acesso a tela de login   
**Then** deve apresentar o placeholder com a descrição do tipo de acesso por exemplo “Email, nome do usuário ou CPF”

**Background**:  
**Given** que eu acesse a página de autenticação do portal EBAC  
  
**Scenario outline 2:** Autenticação valida e invalida  
**When** eu digitar <usuario>  
**And**  <senha>  
**Then** Então deve exibir <mensagem>

|  |  |  |
| --- | --- | --- |
| “nome do usuário”-“email”-“CPF” | senha | mensagem |
| “[mariaoliveira22@gmail.com](mailto:mariaoliveira22@gmail.com)” | “@D%Hdkd1m” | “Olá, Maria” |
| “21724183888” | “@D%Hdkd1m” | “Olá, Maria” |
| “mariaoliveira” | “@D%Hdkd1m” | “Olá, Maria” |
| “[mariaoliveira22@gmail.com](mailto:mariaoliveira22@gmail.com)” | “@D%Hd” | “Senha invalida tente novamente” |
| “[mariaoliveira2@gmail.com](mailto:mariaoliveira2@gmail.com)” | “@D%Hdkd1m” | “Usuário invalido tente novamente” |

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

Critérios de aceitação:

Contexto:

**Given** acesso o payload para API de cupons

**Scenario outline 1**: Gerar cupons e listar cupons

**When** realizo uma requisição com código de cupom “<codigo>”  
**And**  valor “<valor>”   
**And** tipo de desconto “<tipo>”  
**And** descrição “<descricao>”  
**Then** deve ser apresentado status code “<code>”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| código | valor | tipo | descrição | Status code |
| ganhe 10 | R$200 | fixed\_product | Cupom de teste | 200 |
| ganhe 10 | R$300 | fixed\_product | Cupom de teste | 200 |
| ganhe 10 | R$500 | fixed\_product | Cupom de teste | 200 |
| ganhe 10 | R$600 | fixed\_product | Cupom de teste | 200 |
| ganhe 15 | R$601 | fixed\_product | Cupom de teste | 200 |
| ganhe 10 | R$200 | fixed\_product | Cupom de teste | 200 |

**Scenario** 2: cupons cadastrados  
**Given** que tenho um payload para API de cupons   
**When** realizo “Get”  
**Then** deve listar cupons

## 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:

|  |  |  |
| --- | --- | --- |
| R01 | Não é permitido inserir mais de 10 itens de um mesmo produto ao carrinho | |
| CT01 | Deve permitir adicionar 8 itens do mesmo produto ao carrinho | Válido |
| CT02 | Não deve permitir adicionar 15 itens do mesmo produto ao carrinho | Invalido |

|  |  |  |
| --- | --- | --- |
| R02 | Os valores não podem ultrapassar a R$ 990,00 | |
| CT01 | Deve permitir adicionar produtos no valor de R$800,00 | Válido |
| CT02 | Deve permitir adicionar produtos no valor de R$1.500,00 | Invalido |

|  |  |  |
| --- | --- | --- |
| R03 | Valores entre R$ 200 e R$ 600 , ganham cupom de 10% | |
| CT01 | Compra no valor de R$100,00 | Sem desconto de 10% |
| CT02 | Compra no valor de R$400,00 | Com desconto de 10% |
| CT03 | Compra acima de R$700,00 | Sem desconto de 10% |

## História de usuário 2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| R01 | Somente usuários ativos podem fazer login | | | | | |
| Condições | | **Regra1** | **Regra2** | **Regra3** | **Regra 4** |  |
| Usuário valido | | V | F | V | F |  |
| Senha valida | | V | V | F | F |  |
| Ações | | | | | | |
| Permitir autenticação | | V | F | F | F |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| R02 | Se o usuário errar por 3 vezes a senha, deve travar por 15 minutos o login | | | | |
| Condições | | **Regra1** | **Regra2** | **Regra3** |  |
| Senha invalida | | 3<Tentativas | 3>Tentativas | 3>= Tentativas |
| Ações | | | | | |
| Bloqueado por 15 minutos | | N | S | S |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| R03 | Deve exibir uma mensagem de erro caso o usuário erre o login e senha | | | | | |
| Condições | | **Regra1** | **Regra2** | **Regra3** | **Regra4** | |
| Usuario valido? | | S | N | S | N |
| Senha valida? | | N | S | S | N |
| Ações | | | | | | |
| Exibir mensagem de erro? | | S | S | S | N | |

## História de usuário 3: API de Cupom

|  |  |  |  |
| --- | --- | --- | --- |
| Deve listar todos os cupons cadastrado ou listar buscando por ID do cupom GET | | | |
| CT01 | R01 | Listar cupons cadastrados | 200 |
| CT02 | R02 | Listar cupons por ID | 200 |
| CT03 | R03 | Listar cupons por email | 401 |

|  |  |  |  |
| --- | --- | --- | --- |
| Deve cadastrar cupom não pode ser repetido POST | | | |
| CT01 | R01 | Cadastrar cupons tipo de desconto “percent” | 201 |
| CT02 | R02 | Cadastrar cupons tipo de desconto “fixed\_product” | 201 |
| CT03 | R03 | Cadastrar cupons tipo de desconto “fixed\_produc” | 401 |

## Repositório no Github

* Crie um repositório no github com o nome TCC-EBAC;
* Deixe o repositório publico 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: <cole o link aqui>

## 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

Neste projeto tive a possibilidade de aplicar algumas técnicas de testes, automatizar processos repetitivos e aprender algumas ferramentas para prevenir alguns impactos negativos para o cliente. QA não pode ser apenas limitado em procurar bugs. QA deve prevenir sugerir melhorias etc. Todos sabem que não é garantia ter zero bugs, mas podemos aplicar estratégias para evita-las, e ter um cliente satisfeito com feedback rápido.

# REFERÊNCIAS BIBLIOGRÁFICAS

Seguir regras ABNT