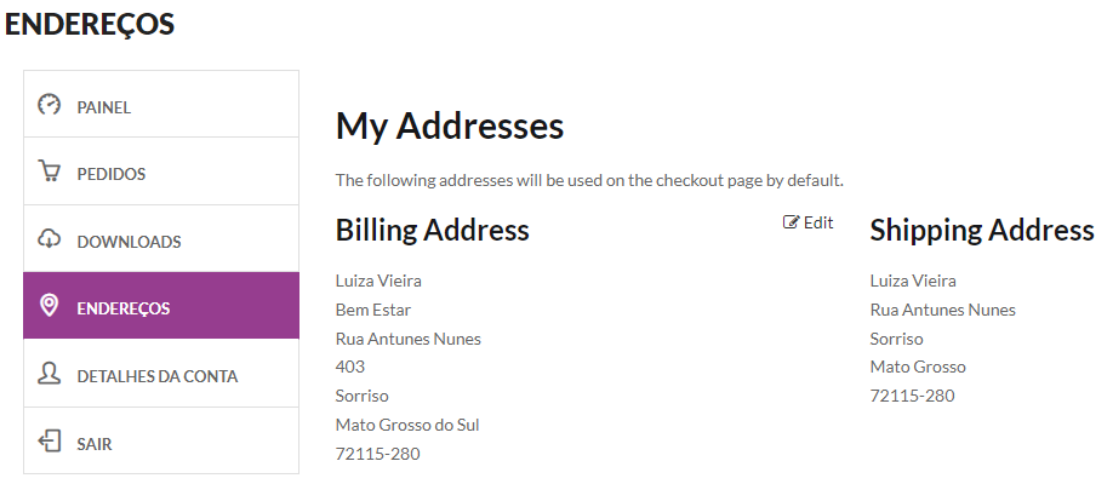
Comandos customizados.

Para iniciar na página da loja Ebac, temos que estar logados para ter acesso a algumas funcionalidades

Vamos acessar a parte de "Endereços"



São dois endereços, de faturamento e de entrega (os dois possuem a opção "Edit" clicando é possível editar).

Crie um arquivo endereço.spec.js e acrescente as primeiras linhas conforme descrito abaixo:

```
/// <reference types="cypress"/>

describe('Funcionalidade Endereços - Faturamento e Entrega', () => {
    beforeEach(()=>{
        cy.visit('Minha conta')
    });

    it('Deve fazer cadastro de faturamento com sucesso', () => {
        //Login
        //Cadastro de endereço
    });
});
```

Para fazer o login, faremos de uma forma diferente, vamos criar um comando customizado

Clique na pasta "suport" e abra o arquivo commands.js e copie a linha comentada abaixo:

```
// Cypress.Commands.add('login', (email, password) => {
... })
```

Cole no final do arquivo sem o // de comentário, troque email e password por usuário e senha, e onde estão os trêss pontinhos coloque as ações de login (basta copiar do arquivo login3.spec.js)

```
Cypress.Commands.add('login', (usuario, senha) => {
    cy.get('#username').type('aluno_ebac@teste.com')
    cy.get('#password').type('teste@teste.com')
    cy.get('.woocommerce-form > .button').click()
})
```

Mude os parâmetros "Aluno_ebac@teste" e senha "teste@teste" para usuário e senha conforme mostrado abaixo:

```
Cypress.Commands.add('login', (usuario, senha) => {
    cy.get('#username').type('usuario')
    cy.get('#password').type('senha')
    cy.get('.woocommerce-form > .button').click()
})
```

Dessa forma o usuário e senha ficam dinâmicos podendo usar o faker, dados de arquivos, dados fixos...

O código do arquivo endereço.spec.js ficara assim:

```
/// <reference types="cypress"/>

describe('Funcionalidade Endereços - Faturamento e Entrega', () => {
    beforeEach(()=>{
        cy.visit('Minha conta') // vai acessar a página de login
        cy.login('aluno_ebac@teste.com', 'teste@teste.com')// vai digitar
login e senha
    });
```

```
    it('Deve fazer cadastro de faturamento com sucesso', () => {
        //Cadastro de endereço
    });
});
```

Agora é só testaar.

Podemos melhorar esse teste, modificando o código assim:

```
/// <reference types="cypress"/>

describe('Funcionalidade Endereços - Faturamento e Entrega', () => {
    beforeEach(()=>{
        cy.visit('Minha conta') // vai acessar a página de login
        cy.fixture('perfil').then(dados =>{
            cy.login(dados.usuario, dados.senha)
        })

    });

    it('Deve fazer cadastro de faturamento com sucesso', () => {
        //Cadastro de endereço
    });
});
```

Podemos fazer vários comandos customizados, por exemplo, abra o arquivo pre-cadastro.spec.js e acrescente as linhas abaixo:

```
it('Deve efetuar o pre cadastro com comandos customizados', () => {


    });
```

Agora volte para o arquivo commands.js e acrescente as linhas abaixo:

```
cypress.Commands.add('precadastro', ()=>{
    cy.get('#reg_email').type(emailFaker)// cria um e-mail fake
    cy.get('#reg_password').type('!teste@teste$')
    cy.get(':nth-child(4) > .button').click()

    cy.get('.woocommerce-MyAccount-navigation-link--edit-account >
a').click()// link detalhes da compra
    cy.get('#account_first_name').type(nomeFaker)// gera um nome
```

```
    cy.get('#account_last_name').type(sobrenomeFaker)// gera um sobre
nome
    cy.get('.woocommerce-Button').click()
})
```

É preciso mudar os parâmetros que vem após o .type para "email, senha, nome e sobrenome respectivamente", vai ficar assim:

```
cypress.Commands.add('precadastro', ()=>{
    cy.get('#reg_email').type(email)
    cy.get('#reg_password').type(senha)
    cy.get(':nth-child(4) > .button').click()

    cy.get('.woocommerce-MyAccount-navigation-link--edit-account >
a').click()// link detalhes da compra
    cy.get('#account_first_name').type(nome)
    cy.get('#account_last_name').type(sobrenome)
    cy.get('.woocommerce-Button').click()
})
```

No arquivo command.js tem que estar assim:

```
Cypress.Commands.add('login', (usuario, senha) => {
    cy.get('#username').type(usuario)
    cy.get('#password').type(senha)
    cy.get('.woocommerce-form > .button').click()
})
Cypress.Commands.add('precadastro', (email, senha, nome, sobrenome)=>{
    cy.get('#reg_email').type(email)// cria um e-mail fake
    cy.get('#reg_password').type(senha)
    cy.get(':nth-child(4) > .button').click()

    cy.get('.woocommerce-MyAccount-navigation-link--edit-account >
a').click()// link detalhes da compra
    cy.get('#account_first_name').type(nome)// gera um nome
    cy.get('#account_last_name').type(sobrenome)// gera um sobre nome
    cy.get('.woocommerce-Button').click()
})
```

O arquivo pre-cadastro.spec.js tem que estar assim:

```
 it.only('Deve efetuar o pre cadastro com comandos customizados', () => {
        let emailFaker2 = faker.internet.email()// variavél que gera um
e-mail novo
        cy.precadastro(emailFaker2, 'senha!@#forte', 'Simone', 'Santos')
```

```
        cy.get('.woocommerce-message').should('contain', 'Detalhes da
conta modificados com sucesso')// teste que comprova a modificação dos
dados
    });
```

É apenas uma paste do código onde acrescentamos a variável emailFaker2 para gerar um e-mail fake, o cy,precadastro chama a variável emailFaker2, com uma senha, nome e sobrenome a serem preenchidos na página. O e-mail tem que ser novo a cada teste pois a página não aceita o mesmo e-mail para um novo cadastro.

Efetue o teste, testando apenas o trecho do código (é o IT que contém o .only).

Agora vamos customizar o arquivo produtos.spec.js. Copie o código referente a adição de produtos no carrinho, é esse abaixo:

```
 cy.get('[class="product-block grid"]')
            .contains('Abominable Hoodie').click()// produto
            cy.get('.button-variable-item-M').click()// tamanho
            cy.get('.button-variable-item-Green').click()// cor
            cy.get('.input-text').clear().type(quantidade)// quantidade
            cy.get('.single_add_to_cart_button').click()// botão
adicionar
```

Abra o arquivo command.js e crie um método dessa forma:

```
Cypress.Commands.add('addProdutos', (quantidade)=>{
    cy.get('[class="product-block grid"]')
    .contains('Abominable Hoodie').click()// produto
    cy.get('.button-variable-item-M').click()// tamanho
    cy.get('.button-variable-item-Green').click()// cor
    cy.get('.input-text').clear().type(quantidade)// quantidade
    cy.get('.single_add_to_cart_button').click()// botão adicionar
})
```

O método é o addProdutos que precisa do parâmetro “quantidade” os sinais => criam uma função.

Modifique a linha que adiciona o produto, criando um novo parâmetro, o código ficará assim:

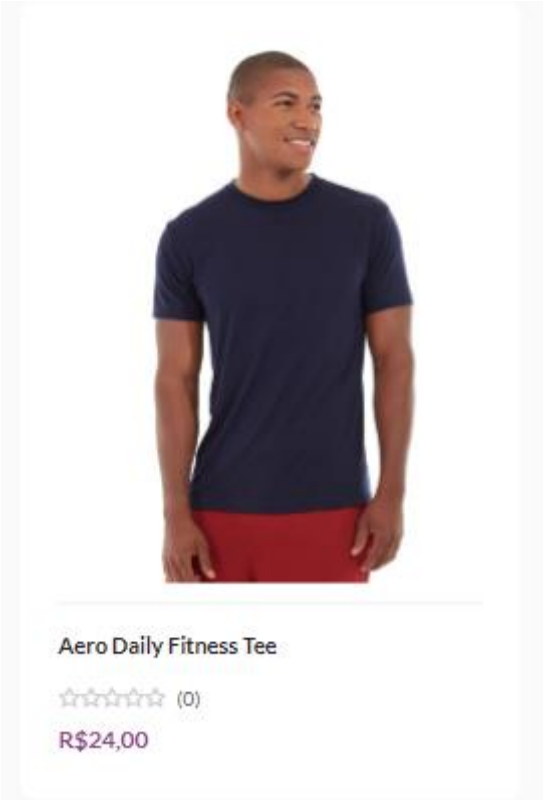
```
Cypress.Commands.add('addProdutos', (produto, quantidade)=>{
    cy.get('[class="product-block grid"]')
    .contains(produto).click()// produto
    cy.get('.button-variable-item-M').click()// tamanho
    cy.get('.button-variable-item-Green').click()// cor
    cy.get('.input-text').clear().type(quantidade)// quantidade
    cy.get('.single_add_to_cart_button').click()// botão adicionar
})
```

Não esqueça de salvar. Agora no arquivo produtos.spec.js acrescente o código abaixo:

```
it.only('Deve adicionar produtos usando comando customizado', () => {
        cy.addProdutos('Abominable Hoodie', 3)
    });
```

Esse trecho faz o teste de adicionar o produto "Abominable Hoodie" três vezes no carrinho (quantidade), tudo isso através do método criado addProdutos. Utilize o .only pra rodar somente esse IT(teste).

Agora escolha outro produto para ser adicionado no carrinho, o produto abaixo:



O código ficará assim:

```
it.only('Deve adicionar produtos usando comando customizado', () => {
        cy.addProdutos('Aero Daily Fitness Tee', 3)
    });
```

Vai dar erro!!!!

O erro está na parte em que se escolhe a cor, tem tem esse produto escolhido na cor Verde, nesse caso vamos adicionar o parâmetro cor, e aproveitar e adicionar o parâmetro tamanho no arquivo command.js, assim:

```
Cypress.Commands.add('addProdutos', (produto, tamanho, cor,
quantidade)=>{
    cy.get('[class="product-block grid"]')
    .contains(produto).click()// produto
    cy.get('.button-variable-item-'+ tamanho).click()// tamanho
    cy.get('.button-variable-item-'+ cor).click()// cor
    cy.get('.input-text').clear().type(quantidade)// quantidade
    cy.get('.single_add_to_cart_button').click()// botão adicionar
})
```

O sinal de + está concatenando (juntando) o `cy.get('.button-variable-item-'` com o parâmetro tamanho, o mesmo acontece com cor.

Agora no arquivo produtos.spec.js mude o código, assim:

```
it.only('Deve adicionar produtos usando comando customizado', () => {
        cy.addProdutos('Aero Daily Fitness Tee', 'M', 'Black', '2')
    });
```

Agora é só rodar o teste.

Atenção!!! Se o produto não tiver estoque, o mesmo não vai ser adicionado ao carrinho. No momento em que faço esse teste 01/02/2023 aconteceu isso, o produto está esgotado, mas o teste passou.

Agora copie e cole o código acima (o IT) e mude o produto o tamanho e a cor, assim:

```
it.only('Deve adicionar produtos usando comando customizado', () => {
        cy.addProdutos('Abominable Hoodie', 'L', 'Blue', '2')
    });
```

Rode o teste.

Você pode demorar pra fazer o primeiro teste mas depois basta copia-lo pra fazer o segundo terceiro, quarto, mudando apenas os nomes dos produtos, tamanho, e cor, criando cenários positivos, negativos.

Tire o .only dos testes e suba tudo para o github.