**Passo 1: Construção dos Scripts**

**1.1 Cenários de Teste (10 cenários)**

**Login com credenciais válidas:**

**Dado: O usuário está na tela de login.**

**Quando: Ele insere um e-mail e senha válidos.**

**Então: Ele é autenticado com sucesso e redirecionado para a tela inicial.**

**Login com credenciais inválidas:**

**Dado: O usuário está na tela de login.**

**Quando: Ele insere um e-mail ou senha incorreta.**

**Então: Uma mensagem de erro "Credenciais inválidas" é exibida.**

**Cadastro de novo usuário com e-mail já cadastrado:**

**Dado: O usuário tenta se cadastrar com um e-mail já existente.**

**Quando: Ele completa o formulário de cadastro.**

**Então: Uma mensagem de erro "E-mail já cadastrado" é exibida.**

**Cadastro de novo usuário com sucesso:**

**Dado: O usuário preenche os campos corretamente.**

**Quando: Ele envia o formulário.**

**Então: O novo usuário é cadastrado e redirecionado para a tela inicial.**

**Navegação da tela inicial para a tela de perfil:**

**Dado: O usuário está na tela inicial.**

**Quando: Ele clica no ícone de perfil.**

**Então: Ele é redirecionado para a tela de perfil.**

**Erro ao tentar navegar para a tela de perfil sem login:**

**Dado: O usuário não está autenticado.**

**Quando: Ele tenta acessar a tela de perfil.**

**Então: Uma mensagem de erro "Necessário estar logado" é exibida.**

**Preenchimento de formulário com campos obrigatórios faltando:**

**Dado: O usuário está preenchendo um formulário.**

**Quando: Ele deixa de preencher um campo obrigatório.**

**Então: Uma mensagem de erro "Preencha todos os campos" é exibida.**

**Validação de e-mail inválido no formulário de cadastro:**

**Dado: O usuário está preenchendo o campo de e-mail.**

**Quando: Ele insere um e-mail inválido.**

**Então: Uma mensagem de erro "E-mail inválido" é exibida.**

**Logout com sucesso:**

**Dado: O usuário está autenticado.**

**Quando: Ele clica no botão de logout.**

**Então: Ele é deslogado e redirecionado para a tela de login.**

**Timeout na resposta da API:**

**Dado: O usuário executa uma ação que chama a API.**

**Quando: A resposta demora mais de 10 segundos.**

**Então: Uma mensagem "Erro de conexão" é exibida.**

**1.2 Padrão Page Object**

**Crie classes de Page Objects para cada tela e as interações:**

**(javascript)**

**// loginPage.js**

**class LoginPage {**

**get inputEmail() { return $('#email'); }**

**get inputPassword() { return $('#password'); }**

**get btnLogin() { return $('#login'); }**

**login(email, password) {**

**this.inputEmail.setValue(email);**

**this.inputPassword.setValue(password);**

**this.btnLogin.click();**

**}**

**}**

**module.exports = new LoginPage();**

**1.3 Data-Driven com JSON**

**Exemplo de arquivo JSON para parametrizar testes de login:**

**[**

**{**

**"email": "usuario@valido.com",**

**"password": "senhaValida",**

**"expected": "Login com sucesso"**

**},**

**{**

**"email": "usuario@invalido.com",**

**"password": "senhaInvalida",**

**"expected": "Credenciais inválidas"**

**}**

**]**

**No script de teste:**

**const loginData = require('./data/loginData.json');**

**loginData.forEach((data) => {**

**it(`Deve validar o login para ${data.email}`, () => {**

**LoginPage.login(data.email, data.password);**

**expect(browser.getAlertText()).toEqual(data.expected);**

**});**

**});**

**Passo 2: Execução em Ambientes Diferentes**

**2.1 Configurar o WebdriverIO para emuladores Android/iOS:**

**No arquivo wdio.conf.js:**

**capabilities: [{**

**platformName: 'Android',**

**platformVersion: '11.0',**

**deviceName: 'Pixel\_3',**

**app: '/path/to/app.apk',**

**automationName: 'UiAutomator2'**

**},**

**{**

**platformName: 'iOS',**

**platformVersion: '14.4',**

**deviceName: 'iPhone 12',**

**app: '/path/to/app.app',**

**automationName: 'XCUITest'**

**}**

**],**

**2.2 Integração com BrowserStack**

**Configure as capabilities para conectar ao BrowserStack:**

**capabilities: [{**

**'browserstack.user': 'YOUR\_USERNAME',**

**'browserstack.key': 'YOUR\_ACCESS\_KEY',**

**'device': 'Samsung Galaxy S21',**

**'os\_version': '11.0',**

**'app': 'bs://YOUR\_APP\_ID'**

**}],**

**Passo 3: Geração de Evidências**

**3.1 Captura de Screenshots**

**No WebdriverIO, ative a captura de screenshots ao falhar:**

**(javascript)**

**afterTest: function (test, context, { error }) {**

**if (error) {**

**browser.takeScreenshot();**

**}**

**},**

**3.2 Relatórios com Allure Report**

**Instale e configure o Allure Report:**

**(bash)**

**npm install @wdio/allure-reporter --save-dev**

**No arquivo wdio.conf.js:**

**(javascript)**

**reporters: [**

**['allure', {**

**outputDir: 'allure-results',**

**disableWebdriverStepsReporting: true,**

**disableWebdriverScreenshotsReporting: false,**

**}],**

**],**

**Passo 4: Integração CI/CD com GitLab**

**No .gitlab-ci.yml:**

**(yaml)**

**stages:**

**- test**

**test:**

**image: node:12**

**script:**

**- npm install**

**- npm run test**

**artifacts:**

**when: always**

**paths:**

**- allure-results/**

**only:**

**- master**