Documentação de Configuração e Execução de Testes - Projeto Auvo

https://github.com/FabioDiass/Auvo.git

https://github.com/FabioDiass/Auvo/blob/main/playwright-report/index.html

1. Introdução

O objetivo deste documento é fornecer um guia completo para a configuração do ambiente de testes, a execução dos testes e destacar decisões importantes relacionadas ao projeto "Auvo".

2. Pré-Requisitos

2.1 Ferramentas Necessárias

- **Node.js**: Versão 16 ou superior (para garantir compatibilidade com os pacotes e dependências).
    - Baixar Node.js.
- **NPM**: Gerenciador de pacotes Node.js, geralmente instalado junto com o Node.js.
- **Playwright**: ferramenta de testes utilizada no projeto.

2.2 Dependências do Projeto

- Para garantir que todas as dependências necessárias sejam instaladas, execute o seguinte comando:

npm install

2.3 Configuração do Ambiente

- Verifique se o projeto está configurado para rodar no ambiente desejado (desenvolvimento, teste, produção).
- Se o projeto utilizar variáveis de ambiente, crie um arquivo .env com as configurações necessárias (ex: credenciais de banco de dados, URL da API, etc.).

3. Configuração de Testes

3.1 Instalação das Ferramentas de Teste

Se o Playwright for a ferramenta de testes utilizada, siga o seguinte processo de instalação:

1. Instale o Playwright no projeto:

npm install playwright

2. Após a instalação, você pode verificar a versão do Playwright:

npx playwright –version

3.2 Configuração de Testes

- Se o projeto requer configuração específica de browsers (Chromium, Firefox, Webkit), defina as opções no arquivo de configuração do Playwright ou em arquivos de teste individuais.

const { chromium } = require('playwright');

3.3 Scripts de Execução de Testes

Configure scripts no package.json para facilitar a execução dos testes.

```
"scripts": {
  "test": "playwright test"
}
```

Execute os testes com:

```
npm run test
```

4. Execução dos Testes

4.1 Executando Testes Locais

Para executar os testes de forma local, execute o seguinte comando:

```
npx playwright test
```

Isso irá rodar todos os testes definidos no projeto.

4.2 Executando Testes em Ambiente de CI

Se o projeto estiver integrado a uma ferramenta de CI (como GitHub Actions, GitLab CI, etc.), a execução de testes pode ser automatizada. Exemplos de configuração de CI podem ser encontrados nas respectivas documentações dessas ferramentas.

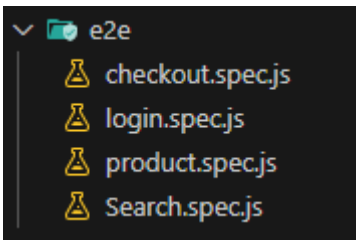
4.3 Cobertura de Testes

Garantimos que todos os cenários descritos no desafio foram cobertos pelos testes.

1. Login: Fazer login na aplicação com credenciais padrão fornecidas.
2. Navegação e Pesquisa: Navegar até a seção de produtos e realizar uma pesquisa.
3. Validação do Produto: Selecionar um produto, verificar o título, preço e descrição.
4. Adicionar ao Carrinho e Checkout: Adicionar o produto ao carrinho, acessar o carrinho e seguir até a tela de checkout.
5. Finalização de Pedido: Preencher os dados de checkout e finalizar o pedido.

5. Estrutura de Diretórios de Testes

De acordo com as boas práticas, para facilitar a manutenção e escalabilidade. Um exemplo de estrutura pode ser:



6. Relatórios de Teste HTML

https://github.com/FabioDiass/Auvo/blob/main/playwright-report/index.html

Após a conclusão do teste, um Reporter HTML foi gerado, que mostra um relatório completo de testes, permitindo que filtre o relatório por navegadores, testes aprovados, testes falhados, testes ignorados e testes "skippados". Você pode clicar em cada teste e explorar os erros do teste, bem como cada etapa do teste. Por padrão, o relatório HTML é aberto automaticamente se alguns dos testes falharem**.**

7. Decisões Importantes

- **Escolha da Ferramenta de Teste**: Optou-se pelo Playwright devido à sua capacidade de testar múltiplos browsers, facilidade de integração com CI e execução paralela de testes.
- **Estrutura de Testes**: A escolha de separar testes e2e (fim a fim) e testes unitários em diretórios diferentes visa melhorar a organização e facilitar a manutenção.
- **Execução de Testes em CI**: Decidiu-se integrar os testes ao pipeline de CI para garantir que as alterações no código não quebrem a funcionalidade existente e para realizar testes automatizados com frequência.

8. Conclusão

Esse documento fornece os passos necessários para configurar e executar os testes no projeto Auvo, além de registrar as decisões tomadas durante o processo. A execução automatizada dos testes garante maior confiabilidade e qualidade no código.