Documento de Arquitetura de Software

Objetivo deste Documento

Este documento tem como objetivo descrever as funcionalidades da automação desenvolvida para o desafio de automação de testes.

Histórico de Revisão

Data	Demanda	Autor	Descrição	Versão
03/03/2022	[DESAFIO01]	Igor da Silva	Implementação do Padrão de Projeto – POM Criação do fluxo alternativo	1.0

Sumário

1. INTRODUÇÃO

1.1 Finalidade

Este documento fornece uma visão arquitetural abrangente do sistema desafio de automação de testes, com a finalidade de mostrar as configurações e execução dos testes.

1.2 Escopo

Este Documento de Arquitetura de Software se aplica ao desafio de automação de testes, que será desenvolvido por Igor da Silva.

2. REPRESENTAÇÃO ARQUITETURAL

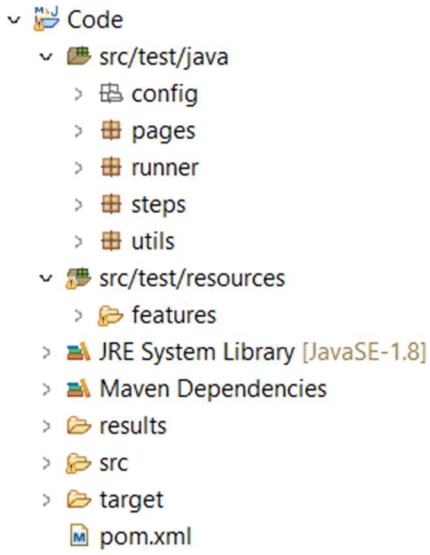
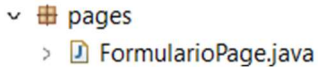


Figura 1 – Estrutura do projeto

Nesse projeto foi utilizado, o padrão POM – Page Object Model, como pode ser visualizado na figura 1.

2.1 Pacote Pages



Dentro desse pacote, iremos armazenar os elementos das páginas. Sempre que for desenvolvido uma nova página para testes, teremos que criar uma classe que vai armazenar os elementos.

2.2 Pacote Steps

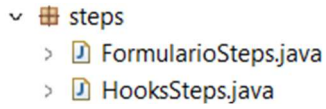


Figura 1.2 – Pacote steps

Esse pacote é responsável pelos steps, como o próprio nome diz. Dentro das classes contidas nesse pacote está a lógica para rodar os steps do cucumber. Abaixo podemos ver um método da classe FormularioSteps desenvolvido:

```
@E("preencho o campo de nome {string}")
public void preencherCampoDeNome(String nome) {
    formulario.digitarNome(nome);
}
```

Figura 1.3 – Método da classe steps

Esse método tem como objetivo digitar o valor que ele recebe do cucumber, através da string nome, dentro do elemento que foi setado na FormularioPage. Por esse motivo temos que chamar o método digitarNome.

Na Figura 1.2, podemos visualizar a classe HooksSteps. Hooks podem ser usados para executar tarefas em segundo plano que não fazem parte da funcionalidade de negócios. Tais tarefas podem ser:

Iniciando um navegador

Configurando ou limpando cookies

Conectando-se a um banco de dados

Verificando o estado do sistema

Monitoramento

```java
package steps;

import java.util.Collection;

public class HooksSteps extends CodeBase {

    @Before("@Formulario")
    public void beforeTest(Scenario scenario){
        Collection<String> sourceTagNames = scenario.getSourceTagNames();
        for (String tag : sourceTagNames) {
            CodeBase.iniciarBrowser(tag);
        }
    }

    @After("@Formulario")
        public void afterTest() {
            driver.close();
            driver.quit();
    }

}
```

Figura 1.4 – Classe HooksSteps

Em nossa classe Hooks, estamos iniciando o navegador no método beforeTest, que são funcionalidades que acontecem antes de começar a executar os tests com a tag @Formulario, também estamos finalizando o nosso driver através do método afterTest, que são funcionalidades que acontecem depois que os steps com a tag @Formulario terminam.

2.3 Pacote Utils

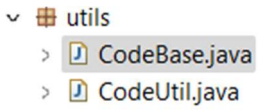
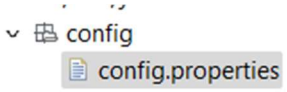


Figura 1.5 Pacote Utils

Dentro desse pacote podemos encontrar as configurações do selenium fazendo a conexão com os browsers, a classe CodeBase faz a lógica para acessar o caminho dos drivers, enquanto a CodeUtil define os tempos de carregamento e espera.

2.4 Pacote Config



Esse pacote é responsável em setar as urls e o navegador.

2.5 Features

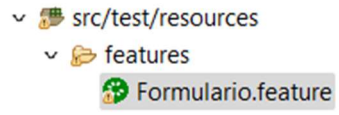


Figura 1.6 Features

Dado que utilizamos Cucumber em nosso projeto, nossos casos de testes são escritos dentro das features.

```
#language:pt
Funcionalidade: Formulario de teste
  Eu como entrevistado
  Quero automatizar o formulario
  Para ser avaliado através da minha automação

  Contexto:
    Dado que estou na pagina de formulario

  @Formulario
  Esquema do Cenário: Preencher os dados do formulario
    E preencho o campo de nome '<nome>'
    E preencho o campo de email '<email>'
    E seleciono a minha cor favorita '<cor>'
    E seleciono ou digito uma sobremesa '<sobremesa>'
    E seleciono a comida favorita '<comida>'
    E seleciono a nota referente ao meu gosto por animais '<numero>'
    E avalio os esportes
    E montar os lanches com os respectivos ingredientes
    E informar a data de hoje
    Quando informar o horario adicionando uma hora
    Então clico no botao enviar

    Exemplos:
      | nome             | email                | cor      | sobremesa | numero | comida  |
      | Maria da Silva   | igorctc226@gmail.com | Azul     | Sorvete   |      5 | Carnes  |
      | Joao da Silva    | igorctc226@gmail.com | Vermelho | Bolo      |      8 | Todas   |
      | Pedro da Silva   | igorctc226@gmail.com | Verde    | Fruta     |     10 | Massas  |
      | Matheus da Silva | igorctc226@gmail.com | Azul     | Mamão     |      1 | Legumes |
```

Figura 1.7 Cenário do formulário

2.6 Runner

Depois que tudo estiver configurado, podemos executar a nossa classe Runner.

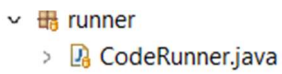


Figura 1.8 Classe Runner

Ela é responsável por rodas nossos testes, podemos selecionar as tags de cenários específicos ou rodar todos em uma só execução.