Projeto de Testes Automatizados

Tecnologias utilizadas:

- **Projeto** – Maven 3.8.4.
- **IDE** – VS CODE.
- **Linguagem de programação** - JAVA 11.0.14.
- **FrameWork de automação** - Selenium WebDriver 3.14.
- **FrameWork de testes** – Junit 4.12.
- **Navegador** – Google Chrome.

Dependências Utilizadas no projeto:

- **Junit 4.12**

```
<dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
        <scope>test</scope>
</dependency>
```

- **Selenium 3.14**

```
<dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>3.14.0</version>
</dependency>
```

Estrutura do projeto:

O projeto foi estruturado da seguinte maneira respectivamente:

1. **Src** : Pacote principal.
2. **Ressource**: pacote onde serão armazenados todos os recursos externos.
3. **Test**: Pacote onde será desenvolvido o projeto.
4. **Java**: Pacote refenciando a linguagem.
5. **Br.com.sicred**: Pacote onde será armazenado o projeto.
6. **Builder**: Pacote onde serão armazenado as classes buidler.
7. **Page**: Pacote onde serão armazenadas as classes de PageObject.
8. **Test**: Pacote onde serão armazenados as classes de teste.
9. **Util**: Pacote onde serão armazenados as classes de utilidades comum.

Padrões Utilizados:

- **Builder**: Utilizado para criação de objetos complexos.
- **PageObject**: Padrão organizacional para melhor mapeamento das páginas web.
- **PageFactory**: Padrão para criação de objetos web de forma otimizada.
- **Boas práticas em Java**: Projeto seguindo as boas práticas da linguagem de acordo com a convenção de Java.
- **Desenvolvimento funcional**: Preferencia para criação de métodos visando baixar a responsábilidade dos testes e neutralizar o uso de códigos “hard code”.

Organização do pacote Builder:

- **Sufixo “Builder”**: Toda classe builder deve ter o sufixo “Builder” para facilitar a manutenção do código.
- **Distribuição dos Builders**: Os Builders devem ser distribuídos na raiz do pacote builder em caso de classe única. Caso seja necessário quebrar o Builder em múltiplas classes, deve ser criado um subpacote que levará o nome de referência do Builder principal.
- **Estrutura do Builder**: Os atributos do builder devem ser "setados” com seus valores vazios, permitindo a criação de um objeto completamente vazio caso nenhum atributo seja modificado em sua chamada.

Organização do pacote Page:

- **Sufixo “PO”**: Toda classe PO deve ter o sufixo “PO” para facilitar a manutenção do código. A abreviação PO remete-se ao padrão PageObject.
- **Distribuição das PO’s**: As PO’s devem ser distribuídas na raiz do pacote Page em caso de classe única. Caso seja necessário quebrar a PO em múltiplas classes, deve ser criado um subpacote que levará o nome de referência da PO principal.
- **BasePO**: Classe base onde devem ser alocados todos os blocos de código em comum a mais de uma PO, essa classe deve ser herdada por todas as demais PO’s de modo que seus filhos tenham acesso aos blocos de código armazenados nessa classe. A classe base deve ser criada como abstrata, impedindo assim que ela seja instanciada em qualquer ponto do sistema.

Organização do pacote Test:

- **Sufixo “Test”**: Toda classe de teste deve ter o sufixo “Test” para facilitar a manutenção do código.
- **Distribuição dos Testes**:

  Os testes devem ser distribuídos de acordo com seu tipo:

  1. **Testes de componente:** São os testes destinados a validar componentes de forma unitária, esses devem ser armazenados dentro do pacote “component” criado na raiz do pacote “test”.

  2. **Testes de pequenos fluxos:** São os testes destinados a validar pequenos fluxos do sistema, esses devem ser armazenados dentro do pacote “flow” criado na raiz do pacote “test”.

  3. **Testes de grandes fluxos**: São os testes destinados a validar todo o fluxo do sistema, esses devem ser armazenados dentro do pacote “endToEnd” criado na raiz do pacote “test”.

- **BaseTest**: Classe base onde devem ser alocados todos os blocos de código em comum a mais de um teste, essa classe deve ser herdada por todos os demais testes de modo que seus filhos tenham acesso aos blocos de código armazenados nessa classe. A classe base deve ser criada na raiz do pacote “test” como classe abstrata, impedindo assim que ela seja instanciada em qualquer ponto do sistema.

Organização do pacote Util:

- **Sufixo “Util”**: Toda classe util deve ter o sufixo “Util” para facilitar a manutenção do código.
- **Distribuição dos Uteis**: Os Uteis devem ser distribuídos na raiz do pacote util.
- **Propósito da classe**: Esta classe tem como propósito armazenar os métodos que possam ser reutilizados em todo o sistema de forma simples e otimizada.