Portal do Contribuinte

Este projecto visa atender a demanda dos testes automatizados do Portal do Contribuinte. Serão utilizadas as seguintes tecnologias: Katalon, Selenium e Cucumber (BDD) e os scripts serão desenvolvidos em linguagem Groovy.

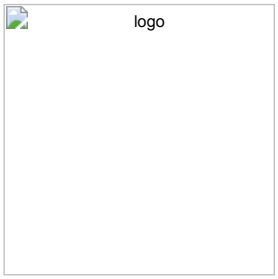
Arquitetura do projeto

Para este projeto foi utilizado o modelo de design Page Objects Model (POM) para criar os repositório de objetos para interação dos elementos com a interface do usuário (Web UI) A arquitetura de testes é um componente importante para garantir a qualidade do software e facilitar a execução eficiente e escalável dos testes automatizados. Neste documento, abordaremos os principais componentes da arquitetura e as melhores práticas para sua implementação.

Padrão de design (Page Objects Model)

O modelo de padrão de objetos (Page Objects) é um padrão de design utilizado no desenvolvimento de testes automatizados para garantir a eficiência e a reutilização de código. Ele visa criar uma estrutura organizada e modular para os testes, permitindo uma fácil manutenção e escalabilidade do conjunto de testes. Nesse padrão, os objetos são projetados para representar elementos ou componentes do sistema sendo testado. Cada objeto é responsável por encapsular as funcionalidades e propriedades relacionadas a um determinado elemento, como botões, campos de entrada, páginas web, entre outros. Essa abstração permite que os testes sejam escritos de forma mais legível e compreensível, facilitando a criação de casos de teste e a identificação de erros. Além disso, o padrão de objetos design promove a reutilização de código, pois os objetos podem ser criados uma vez e utilizados em diferentes testes. Isso reduz a duplicação de código e torna a manutenção dos testes mais eficiente, uma vez que as atualizações nos objetos são refletidas automaticamente em todos os testes que os utilizam.

A imagem abaixo representa a iteração da web com os elementos. Para cada página da Web no aplicativo, deve haver uma classe de página correspondente. Essa classe Page identificará os WebElements dessa página da Web e também contém métodos Page que executam operações nesses WebElements:



Desta forma a nível do script é ilustrado da seguinte forma:

Estrutura de Arquivos

A estrutura do Portal do Contribuinte foi definida na seguinte estruturação:

```
./
|-- portal-do-contribuinte/                    # Código-fonte do projeto
|-- Profiles/                                  # Configuração da BaseURL dos ambientes
|-- test Cases/                                # Diretório dos casos de testes
|   |-- Testes de Componentes/
|   |-- Testes de Fumaça (smoke test)/
|   |-- Testes de Regressão/
|   |-- Testes Funcionais/
|   |-- Testes de Integração/
|-- Object Repository/                         # Gerenciamento e reutilização dos elementos mapeados da UI (Interface do Usuário)
|   |-- Api/                                   # Diretório dos endpoints
|   |-- Web/                                   # Diretório dos elementos da UI
|     |-- Page/                                # Mapeamento dos elementos do Portal do Contribuinte
|     |-- Portal-Administrativo/               # Mapeamento dos elementos do Portal Administrativo
|-- Test Suites/                               # Diretório das Suites de Testes
|   |-- Testes de Componentes/
|   |-- Testes de Fumaça (smoke test)/
|   |-- Testes de Regressão/
|   |-- Testes Funcionais/
|   |-- Testes de Integração/
|-- Data Files/                                # Diretório das massas de dados
|-- Keywords/                                  # Diretório do mapeamento das páginas da UI com os artefactos do Object repository
|     |-- api/                                 # Pacote de classes dos testes de API
|     |-- page/                                # Pacote de classes do Portal do Contribuinte
|     |-- portalAdministrativo/                # Pacote de classes do Portal Administrativo
|-- Include/                                   # Diretório de integração com o Cucumber (BDD)
|   |-- features/                              # Testes documentados em escrita BDD
|       |-- Testes de Componentes/
|       |-- Testes de Fumaça (smoke test)/
|       |-- Testes de Regressão/
|       |-- Testes Funcionais/
|       |-- Testes de Integração/
|   |-- scripts/                               # Pacote de Scrips dos testes implementados em BDD
|     |-- groovy/
|     |-- portalAdministrativoSteps/           # Pacote de classes do Portal Administrativo
|     |-- steps/                               # Pacote de classes do Portal do Contribuinte
|-- reports/                                   # Relatórios das execuções dos testes
```

Ambiente de Testes

O `SIRIUS` é oficialmente o ambiente definido para criação dos scripts de testes e execução. No entanto, poderá ser utilizado em conjunto o ambiente de `POLARIS`.

em desenvolvimento..

CI/CD

Em desenvolvimento...

Monitoramento e Manutenção

O monitoramento contínuo e a manutenção dos testes automatizados são essenciais para garantir sua eficácia e relevancia do projeto, assim após a implementação do CI (Integração Contínua) o monitoramento e manutenção dos scripts de testes serão realizados conforme o resultado da execução dos testes na pipeline, serão abertos tickets específicos para o problema e criado uma branch apartir do número do JIRA. Também serão definidos semanalmente os envolvidos e responsáveis pela monitorização e manutenção dos scripts de testes.

Classes, métodos e boas práticas

Introdução

Aqui serão documentados as classes e métodos implementados durante a criação da arquitetura do projeto de automação bem como o padrão de escrita adotado para os nomes dos métodos, classes, variáveis.

Índice

Classes e Métodos

Descrição

As classes e métodos foram documentadas diretamente no script de código e encontram-se nos diretórios **/Keywords/ e **/Include/scripts/groovy/ **., assim ao abrir o arquivo `index.html` é possível verificar a documentação das classes. Por exemplo: 

Para verificar a documentação completa, utilize o seguinte documento: Groovy Docs (http://10.129.106.146/web-automation/portal-do-contribuinte/blob/b329b8deb7f8fcd6eeae34d3fe0b0fd1c822bdfe/Include/config/doc/DefaultPackage/CommonSteps.html)

Como mencionado na estrutura de arquivos estes diretórios são responsáveis por gerar os pacotes (packages) contendo a classe e os métodos definidos para cada página.

Adicione comentários de documentação ao criar um novo script de classe. Os comentários devem começar com /* e terminar com */. Use tags como @param, @return e @throws para documentar parâmetros, valores de retorno e exceções lançadas pelo código. Exemplo:

```
/**
 * Insere os dados de utilizador e de palavra-passe nos respectivos campos do elemento da página de login
 * @param username insere o NIF do utilizador
 * @param password insere a palavra-passe gerada
 */
public void inserirCredenciais(String username, String password) {
        step.inserirTexto(utilizador, username)
        step.inserirTexto(senha, password)
}
```

Convenção das nomenclaturas

A nomenclatura adotada é o padrão **PascalCase**, desta forma segue abaixo os exemplos para padronização da escrita no projeto:

`Variáveis`: Utilize pascal case para nomear variáveis, onde a primeira letra de cada palavra composta em uma variável é maiúscula. Por exemplo, **"NomeCompleto", "DataNascimento", "ElementoDePagina".**

`Métodos`: Siga a mesma convenção para nomear métodos, começando com letra minúscula e capitalizando a primeira letra de cada palavra subsequente. **Por exemplo, "clicarBotao", "preencherFormulario", "validarPagina".**

`Classes`: Utilize Pascal Case para nomear classes, começando com letra maiúscula e capitalizando a primeira letra de cada palavra subsequente. Por exemplo, **"LoginPage", "HomePage", "CommonSteps", "LoginSteps".**

`Elementos`: Para nomear os elementos extraídos da WebPage utilize o hífen para separar o tipo e nome do elemento seguido do Pascal Case, começando com letra maiúscula e capitalizando a primeira letra de cada palavra subsequente. Por exemplo, **"btn-IniciarSessao", "btn-CriarNovaDeclaracao", "btn-Submeter".**

Boas Práticas

Ao seguir a correcta criação e padronização das classes e métodos no projeto do Portal do Contribuinte é necessário também ter em atenção se além de seguir a arquitetura do projeto também é possível criar scripts de testes automatizados que possam ser reutilizados e de fácil manutenção. Aqui vão alguns pontos importantes a serem observados a considerar durante a criação e execução do código implementado:

1. Planejamento: Antes de iniciar o desenvolvimento dos testes automatizados, é importante realizar um planejamento adequado dos elementos. Defina os objetivos dos testes, identifique as funcionalidades críticas do sistema, valide se a escrita do cenário tem a cobertura adequada para implementação.
2. Mapeamento dos elementos: Ao validar a funcionalidade a ser implementada, comece mapeando os elementos no `Object Repository` e separando conforme a estrutura da página no Front-End. Nomeie os elementos sem utilização de caracteres especiais.
3. Separação clara entre código de teste e código de automação: Mantenha uma separação clara entre o código dos testes e o código de automação. Isso facilita a manutenção dos testes, permite reutilização e melhora a legibilidade e a clareza do código.
4. Criação de testes independentes: Certifique-se de que cada teste automatizado seja independente dos outros. Isso evita dependências e problemas de ordem de execução, permitindo a execução dos testes de forma isolada e em qualquer ordem desejada.
5. Utilização de dados de teste reutilizáveis: Evite codificar dados de teste diretamente nos scripts de automação. Em vez disso, utilize dados de teste reutilizáveis, que possam ser facilmente modificados e atualizados conforme necessário. Isso facilita a manutenção dos testes e reduz a duplicação de código. Utilize o javaFaker para geração de dados não pré-definidos.
6. Documentação do código: Ao criar novas classes/métodos certifique-se que crie a documentação conforme o padrão definido no índice Classes e Métodos

Configurando o ambiente para testes locais

Acesso ao projecto

Você pode Clonar via git clone (http://10.129.106.146/web-automation/portal-do-contribuinte.git) ou baixar em arquivo .zip (/web-automation/portal-do-contribuinte/repository/develop/archive.zip)

Tecnologias utilizadas

- Katalon Enterprise
- Orientação a objectos
- Groovy
- JDK 8
- Git & GitLab
- Navegadores(Chrome, Edge e Mozila) atualizados

Manual de montagem de ambiente(Windows)

1. Baixar ferramentas

- Katalon Studio (https://backend.katalon.com/download-lastest-version?platform=win_64&type_download=kse_pe)
- Git (https://github.com/git-for-windows/git/releases/download/v2.39.1.windows.1/Git-2.39.1-64-bit.exe)(Versão mais atual)

2. Instalando o Katalon

```
1. Após baixá-lo, abrir o arquivo .zip
2. Extrair os arquivos em uma pasta desejada
3. Executar o arquivo katalon.exe
```

3. Clonando o projecto

Pré condições:

- Ter instalado o Katalon Studio e GIT

- Possuir acesso(user e password) no projecto no gitLab. Obs: Caso não possua, solicitar acesso ao Líder da Automação de testes.

Configurando o ambiente local via git integrado no Katalon

```
1. Execute o arquivo katalon.exe para abrir a ferramenta
2. No ícone de GIT, selecionar a opção "Clone Project"
3. Inserir a URL do projecto(URL)
4. Informar os dados de autenticação do git (User e Password)
5. Acionar a opção "Next"
6. Selecionar a Branch que deseja(default develop)
7. Acionar a opção "Finish"
```

Configurando o ambiente local pelo git bash

```
1. No explorador de arquivo do windows, crie uma pasta onde irá armazenar o código
2. Dentro da pasta, clique com o botão direito e selecione a opção "GIT Bash Here"
3. Digite o comando: git clone http://10.129.106.146/qa-automation/Katalon-qa-automation-cadastro-de-contribuinte.git
4. Insira as credenciais do gitLab (User e Password)
5. Acione "OK" e aguarde até que seja finalizado o clone
6. Finalizado o clone, execute o katalon.exe
7. Com o Katalon aberto, no menu de ferramentas, acesse o "File > Open Project"
8. Selecione a pasta onde foi feito o clone e pronto!
```

4. Atualizando os drivers para execução

```
1. Com o Katalon aberto, no menu da ferramenta, busque por "Tools"
2. Selecione o submenu "Update Webdrivers"
3. Clique em cada driver que deseja atualizar
```

5. Plug-in's essenciais

1. Excel Keywords Clique aqui para baixar (https://store.katalon.com/product/34/Excel-Keywords)

6. Executando o primeiro teste

```
1. Acessar a pasta "Test Cases"
2. Selecionar a pasta que contenha os cenários desejados
3. Duplo clique no test case desejado
3. No menu da ferramenta, selecionar a opção "Run" e selecionar o driver desejado para a execução(default Chrome)
```

Happy Testing! :robot: :space_invader: