Projeto Bom Samaritano
Plano de Testes

Versão 1.0

Plano de Testes

1. Introdução

1.1 Propósito

O propósito deste documento é coletar todas as informações necessárias para planejar e controlar o esforço de teste em uma interação. Ele descreve as abordagens de teste de software.

Este plano de testes suporta os seguintes objetivos:

- Os principais alvos do teste são os dados de entrada que o usuário pode digitar (inputs).
- O objetivo deste teste é simular o usuário interagindo com os campos de texto, no qual ele entra com um dado ou informação.
- O resultado esperado é que os campos preenchidos possam ser enviados ao servidor que neste caso será explorado via console do navegador devido não existir no atual momento o backend.
- Para este teste será necessário uma linguagem de programação, que neste caso foi escolhido o Javascript, também será necessário a ferramenta Node.js para executar o script de teste, o navegador Google Chrome e a biblioteca Selenium para automatizar os testes no navegador.
- O teste será feito com o site publicado no GitHub Pages.

1.2 Escopo

O escopo dos testes previstos por este documento incluem Teste de Unidade e quanto às Funcionalidades do software.

NÃO serão incluídas neste plano de teste os de Integração e de Sistema bem como Requisitos Não-Funcionais.

1.3 Audiência pretendida

Este plano de testes se destina a auxiliar a equipe de desenvolvimento no processo de teste do site. Este documento fornece os objetivos e resultados esperados mediante a execução dos testes e explicita também as ferramentas que deverão ser utilizadas para a execução deste plano de testes.

1.4 Terminologia e Acrônimos

JS – Javascript
Script – Arquivo do tipo Javascript contendo as linhas de código-fonte que servem para dar as instruções necessárias para a execução do teste.
Inputs – Campos de entrada de dados na página no qual o usuário interage.

Browser – Navegador de Internet (Software)
JSON – Javascript Object Notation

1.5 Referências

- Projeto Bom Samaritano – Caso de Uso versão 1 (Fabricio Cruz);
- Projeto Bom Samaritano – Especificação de Requisitos versão 1 (Fabricio Cruz)
- Documentação Selenium WebDriver (Selenium)

2. Missão e Motivação dos Testes

- Avaliar se o formulário está sendo submetido corretamente
- Avaliar se os dados preenchidos estão sendo transformados em um JSON
- Encontrar bugs e falhas caso ocorram

3. Itens Alvos de Testes

Será testado neste plano de testes o seguinte caso de uso:

- Caso de Uso: Cadastro de usuário no sistema

4. Esboço dos Testes Planejados

4.1 Esboço dos Testes Incluídos

Os testes a serem executados serão realizados na página de cadastro de usuários, onde existem um formulário que este deve preencher. Para tal serão inseridos valores que seriam esperados do usuário. Ao término do preenchimento deste formulário, o teste automatizado deverá clicar no botão que finaliza o cadastro, permitindo que esses dados sejam enviados ao servidor (console do browser).

4.2 Esboço dos Testes Excluídos

Devido ao foco dos testes serem nos campos de dados e submissão do formulário não haverá necessidade de se testar elementos de estilização da página ou outras funcionalidades que não se enquadrem dentro deste foco. Alguns principais motivos:

- Elementos não pertinentes ao foco do teste.
- Funcionalidades separadas do escopo do formulário.

5. Abordagem de Testes

Para este plano de testes será utilizado o Teste de Unidade, pois o objetivo é simular os dados a serem introduzidos por um usuário.

Para executar este teste, de maneira automatizada, será acessado pelo Selenium via Javascript/Node a url do site publicado no GitHub Pages e então será acessado cada elemento correspondente a um campo que será interagido pelo usuário dentro da página com o formulário. Para ser possível este acesso aos elementos da página, estes poderão ser acessados via “id”, “class” ou pela estrutura de nós do HTML utilizando a função xpath presente no Selenium. Ao finalizar será clicado no botão que encerra o preenchimento do formulário e simula o envio dos dados ao backend (que no momento ainda é inexistente, portanto, será abordado via console do navegador).

5.1 Técnicas e Tipos de Testes

5.1.1 Testes Funcionais

Objetivo da Técnica:	Entrada e obtenção dos dados nos inputs do formulário.
Técnica:	Será acessado e atribuído valores aos campos de entrada de dados do formulário para simular o usuário. • O resultado é o esperado quando os valores entrados são válidos. • Se algum campo requerido não for preenchido uma pequena mensagem junto ao campo aparece informando a necessidade de preenchimento.
Oráculos:	Será utilizado como estratégia a geração de mensagens no console do Node.js (o que aparece também no terminal do Visual Studio Code). Mensagens referentes ao sucesso do teste. Se essas mensagens não aparecerem ou vierem juntos com outros logs do Node indica um erro na execução.
Ferramentas Necessárias:	Serão necessárias as seguintes ferramentas: • Node.js • Visual Studio Code • Browser
Critérios de Sucesso:	Esta técnica suporta os cenários válidos para o preenchimento do formulário de cadastro.
Considerações Especiais:	O teste não irá funcionar se não houver conexão com a internet e se o Selenium for de uma versão diferente da utilizada para codificar o teste. Pode falhar caso não se use a versão compatível do Chrome com a instalada na máquina que irá rodar os testes.

5.1.2 Testes de Interface de Usuário

Objetivo da Técnica:	Exercita os seguintes itens para observar e registrar a navegação e acesso das funcionalidades acessadas pelo usuário: • O campo do formulário aparece levemente destacado quando selecionado (possui um sombreamento). • Via Selenium são passados os valores que aparecem na tela em cada campo do formulário.
Técnica:	Acessa e modifica os valores de cada campo de entrada de dados.
Oráculos:	Através do próprio navegador que estará com a página do site aberta será possível visualizar os campos sendo preenchidos e selecionados.
Ferramentas Necessárias:	A técnica requer o uso de ferramentas de automação de script de testes como o Selenium e o browser, no qual estará executando o teste e onde os elementos estarão sendo acessados e interagidos.
Critérios de Sucesso:	Esta técnica suporta a tela de cadastro do usuário.
Considerações Especiais:	Somente os campos que podem ser acessados pelo usuário é que serão alvo dos testes, sendo assim quaisquer outros elementos da página não serão contados para fins deste teste.

6. Entregáveis

- Este plano de testes.
- Script utilizado no teste (Código-fonte).

7. Necessidades de Ambiente

Para realizar os testes serão necessários as ferramentas:

- Linguagem de programação - Javascript;
- Editor de código-fonte – Visual Studio Code;
- Software para rodar os arquivos de testes – Node.js;
- Browser – Google Chrome;
- Driver do browser – Chrome Driver;
- Ferramenta para automatizar os testes – Selenium WebDriver.

Antes de escrever o código de teste será necessário realizar o download da biblioteca Selenium e do driver do Google Chrome. Após o download e instalação da biblioteca do Selenium e o driver do browser, será necessário realizar a instalação do Node.js. Com tudo, até então, instalado e configurado será necessário criar um projeto na máquina via Node.js e então escrever o código que será usado para os testes. Para a execução do teste deverá ser utilizado o próprio Node via terminal do Visual Studio Code (ou no prompt do Node.js).