PLANO DE TESTE PARA

TABELA-VERDADE

Registro de Mudanças

Versão	Data de Mudança	Por	Descrição
1.0.0	06/11/2023	Brendo Costa	Versão final do documento.

1 Introdução

Este Plano de Teste visa detalhar o processo de teste para o *software* Tabela-verdade, um gerador de tabela-verdade de expressões lógicas *open source*. O Tabela-verdade é um gerador de tabela-verdade de expressões lógicas *open source* criado com *React* e *Gatsby* para *web*. A aplicação foi criada para a disciplina de Fundamentos Matemáticos para Computação, no 1° período do curso de Sistemas de Informação, da Universidade Federal Fluminense (UFF). Atualmente, esse objetivo se estendeu e agora está disponível para qualquer pessoa utilizá-lo. A aplicação é formada por quatro módulos principais: *source*, *lexer*, *parser* e *generator*. Esses módulos operam em sequência para transformar uma expressão lógica arbitrária de entrada em uma tabela-verdade na saída. Pretendemos testar cada um dos módulos que compõem a aplicação unitariamente, testar a integração entre eles e por fim testar a aplicação em seu caso de uso típico.

1.1 Escopo

1.1.1 No escopo

1.1.1.1 Módulos da aplicação a serem testadas

Módulo	Descrição
Source	Módulo responsável pelo processamento da sequência de caracteres de entrada.
Lexer	Módulo responsável por transformar a sequência de caracteres de entrada em uma sequência de *tokens*.
Parser	Módulo responsável por processar a sequência de *tokens* conforme as regras lógicas.
Generator	Módulo responsável por gerar a tabela-verdade conforme as regras processadas.

1.1.1.2 Requisitos funcionais a serem testados

Nº	Descrição
1	A aplicação deve ser capaz de receber uma sequência lógica de tamanho arbitrário como entrada pelo usuário.
2	A aplicação deve permitir que a linguagem da interface seja alterada.
3	A aplicação deve fornecer uma tabela-verdade de saída logicamente correta.
4	A aplicação deve aceitar as diferentes representações para um mesmo operador lógico.
5	A aplicação deve levar em conta a precedência dos operadores lógicos fornecidos na expressão de entrada.

1.1.1.3 Requisitos não funcionais a serem testados

Nº	Característica	Descrição
1	Eficiência de Desempenho	O processamento da sequência de entrada deve ser realizado de forma automática, sem intervenção do usuário.
2	Usabilidade	A interface de usuário deve se comportar de maneira responsiva e imediata a entradas incorretas do usuário.
3	Usabilidade	A interface de usuário deve fornecer ao usuário atalhos para formulação da expressão de entrada.
4	Portabilidade Adaptabilidade	A aplicação deve ser acessível através de dispositivos móveis sem perdas quanto a suas capacidades funcionais e não funcionais.
5	Manutenibilidade Modularidade	Os módulos da aplicação devem estar disponíveis sob a forma de componentes.
6	Manutenibilidade Analisabilidade	As dependências externas diretas da aplicação devem estar documentadas.
7	Portabilidade Instalabilidade	Todo o processo de instalação da aplicação deve ser automatizado com a mínima intervenção do usuário.
8	Eficiência de Desempenho	A aplicação não deve ocupar mais que 100 megabytes de memória de acesso dinâmico quando em execução.

9	Segurança	As dependências externas da aplicação devem ser auditáveis quanto a vulnerabilidades de segurança.
10	Portabilidade Adaptabilidade	A aplicação deve ser portável para os navegadores *web* Google Chrome, Microsoft Edge, e Mozilla Firefox sem perdas quanto a suas capacidades funcionais e não funcionais.

1.2 Objetivos de Qualidade

Os objetivos deste processo de teste são:

- Certificar que a aplicação está em conformidade com os requisitos funcionais e não funcionais especificados;
- Agregar qualidade aos usuários da aplicação, bem como aqueles que desejem estudá-la ou reutilizá-la em outros projetos;
- Diminuir a quantidade de *issues* de relatos de problemas registrados no repositório do projeto.

1.3 Papéis e Responsabilidades

Papel	Membro(s)	Responsabilidades
Desenvolvedores	Brendo Costa Daniel Verginio Lima Isaac Luiz Vieira Ferreira Mathews Vaz Reis	● Desenvolver testes unitários; ● Desenvolver testes de integração; ● Desenvolver testes de sistema.
Testador	Brendo Costa Daniel Verginio Lima Isaac Luiz Vieira Ferreira Mathews Vaz Reis	● Executar os testes de acordo com a documentação de planejamento e projeto; ● Registrar os incidentes ocorridos durante a execução.

Gerente de Teste	Brendo Costa Daniel Verginio Lima Isaac Luiz Vieira Ferreira Mathews Vaz Reis	• Especificar o Plano de Teste; • Especificar o Relatório de Resumo dos Testes; • Manter a Matriz de Rastreabilidade de Requisitos.
Projetista de Teste	Brendo Costa Daniel Verginio Lima Isaac Luiz Vieira Ferreira Mathews Vaz Reis	• Especificar os Projetos de Teste; • Especificar os Casos de Teste; • Especificar os Procedimentos de Teste.

2 Metodologia de Teste

2.1 Visão Geral

Adotaremos uma metodologia ágil para conduzir o processo de teste da aplicação. A opção por esta metodologia deve-se à pequena quantidade de membros na equipe, trazendo maior facilidade e rapidez na comunicação e alinhamento operacional.

2.2 Fases de Teste

- **Testes de unidade**, consistindo em testes individuais para cada classe utilizada em um módulo.
- **Testes de integração**, consistindo em testes que exploram a comunicação entre dois ou mais módulos da aplicação.
- **Testes de sistema**, consistindo em testes que exploram o caso de uso típico da aplicação totalmente completa e integrada e buscam verificar os requisitos especificados.

2.4 Critérios de Suspensão e Requisitos de Retomada

2.4.1 Critérios de suspensão

Nº	Descrição
1	Impossibilidade de execução da aplicação no ambiente de testes devido a erros referentes ao processo de *build*.
2	Impossibilidade de execução da aplicação no ambiente de testes devido a configuração incorreta do ambiente.
3	Impossibilidade do testador em entender um ou mais procedimentos para a execução do teste.

2.4.2 Critérios de retomada

Critério de suspensão Nº	Descrição
1	Correção dos erros referentes ao processo de *build* por parte da equipe de desenvolvimento.
2	Correção da configuração do ambiente de testes por parte da equipe de desenvolvimento.
3	Dúvidas sanadas após uma reunião de alinhamento entre o testador e o projetista de testes.

2.5 Completude do Teste

- Total cobertura dos módulos da aplicação por testes unitários.
- Todos os casos de teste manuais e automatizados devidamente executados.

2.6 Atividades do projeto, estimativas e cronograma

Atividade	Estimativa de Esforço	Início	Fim
Especificação do	24 homens/hora	30/10/2023	06/11/2023

Plano de Teste			
Especificação dos Projetos de Teste	60 homens/hora	06/11/2023	20/11/2023
Especificação dos Casos de Teste	60 homens/hora	06/11/2023	20/11/2023
Especificação dos Procedimentos de Teste	60 homens/hora	06/11/2023	20/11/2023
Executar os testes	80 homens/hora	20/11/2023	27/11/2023
Criar os relatórios de testes	40 homens/hora	20/11/2023	27/11/2023
Especificar o Relatório de Resumo dos Testes	40 homens/hora	27/11/2023	04/12/2023

3 Entregáveis de Teste

Os seguintes artefatos deverão ter sido gerados ao final do processo:

a) Antes da fase de teste:
   i) Especificação do Plano de Teste;
   ii) Especificações de Projeto de Teste;
   iii) Especificações de Procedimentos de Teste;
   iv) Especificações de Casos de Teste.

b) Durante a fase de teste:
   i) Dados de saída das ferramentas de teste;
   ii) Matriz de Rastreabilidade de Requisitos;
   iii) Logs de erros e logs de execução;

c) Após o término dos ciclos de teste:
   i) Relatórios de teste.

4 Necessidades de Recursos e Ambiente

4.1 Ferramentas de Teste

Faça uma lista de ferramentas necessárias para testar o projeto como

4.1.1 Ferramenta de rastreamento de requisitos

Ferramenta	Descrição
Google Spreadsheets	Ferramenta para manutenção da Matriz de Rastreabilidade de Requisitos.

4.1.2 Ferramenta de rastreamento de erros

Ferramenta	Descrição
ESLint	Ferramenta de análise estática para estilo e conformidade de código fonte.
SonarLint	Ferramenta de análise estática para detecção e correção de problemas de codificação.
Vitest Coverage	Ferramenta para análise da cobertura do código fonte por testes unitários.

4.1.3 Ferramentas de automação de teste

Ferramenta	Descrição
Vitest	Ferramenta para execução de testes unitários em JavaScript.

4.2 Ambiente de Teste

4.2.1 Requisitos mínimos de hardware

- CPU de 1.8 GHz;
- 8 gigabytes de RAM;
- Conexão de rede de 100 megabits;
- Armazenamento baseado em SSD.

4.2.2 Requisitos mínimos de software

- Microsoft Windows versão 10 ou Linux kernel 5;
- Node.js versão 18;
- VSCode versão 1.8;
- Git versão 2.34;
- Google Docs;
- Google Spreadsheets.

5 Termos / Acrônimos

TERMO / ACRÔNIMO	DEFINIÇÃO
Issue	Um registro de incidente ou um registro de uma necessidade para o *software*.
Open source	Projeto/aplicação de código aberto.
Web	Sistema hipertextual que opera através da *internet*.