# Plano de teste Jogo de Adivinha

## 1 Introdução

O projeto do Jogo de Adivinha é um script enxuto para consolidação de conhecimento. Dessa forma, conta com funcionalidade simples de entrada de números inteiros e atribuição de números aleatórios a uma variável que simboliza o segredo a ser descoberto.

## 2 Arquitetura

As tecnologias usadas para implementação do projeto do jogo de adivinha foram o Javascript e o HTML. Sendo assim uma linguagem orientada a objetos e uma linguagem de marcação, além de do VSCODE, como editor de texto.

## 3 Funcionalidade

| **Funcionalidade** | **Comportamento esperado** | **Verificações** | **Critérios de aceite** |
|---|---|---|---|
| • Entrada de um número inteiro | • Ao adicionar um número inteiro no campo de input, o usuário deve clicar no botão "Compare com o meu segredo". E deverá receber um alerta informando se o mesmo adivinhou número secreto. | • deve ser um número inteiro<br>• Uma mensagem de confirmação caso o número de input seja igual ao número secreto.<br>• Mensagem de aviso caso o número do input seja diferente do número secreto. | • Ambas mensagens, final triste e final feliz, devem aparecer ao usuário em seus respectivos casos. |
| | | | |
| | | | |

## 4. Estratégia de Teste

### Escopo de testes

O plano de teste abrange a funcionalidade apresentada acima.

Serão executados teste em todos os níveis conforme descrito abaixo:

Testes Unitários: O código terá uma cobertura de teste unitário na funcionalidade destacada e será feita pelos desenvolvedores no momento do desenvolvimento da aplicação.

Testes Automatizados: Serão realizados testes end-to-end na funcionalidade de comparar o número aleatório com o número de entrada.

Testes Manuais: A Funcionalidade será testada manualmente pelo time de qualidade seguindo a documentação de cenários de teste e deste test plan.

## 5. Ambientes e Ferramentas

Os testes serão feitos em um ambiente de homologação, o mesmo contém as mesmas configurações do ambiente original, porém, se trata de um ambiente controlado criado previamente pelo time de qualidade para testes.

| **Ferramenta** | **Time** | **Descrição** |
|---|---|---|
| Vscode | Desenvolvimento/ Qualidade | Editor de texto utilizado para desenvolvimento do código e criação de testes. |
| Node.js | Qualidade | Ambiente de execução para criação de ambiente controlado. |
| Cypress | Qualidade | Framework para testes E2E no ambiente controlado. |

## 5.Classificação de Bugs

Os bus serão classificados com as seguintes severidades:

| **ID** | **Nível de severidade** | **Descrição** |
|---|---|---|
| 1 | Blocker | Bug que impossibilidade a execução da função, Botar de "comparar com o meu segredo" para de funcionar, impedindo o uso completo da funcionalidade, a mensagem não é entregue ao usuário após o clique no botão |
| 2 | Grave | Input  incomum retorna algum resultado inesperado. |
| 3 | Moderada | A funcionalidade não atinge certos critérios mas, de maneira geral, não é afetada. |
| 4 | Pequena | Erro ortográfico |

## 6. Definição de Pronto

Será considerado pronto a funcionalidade principal descrita nesse TestPlan, não apresentar bugs com severidade acima de Moderada e passarem por uma validação de negócio de responsabilidade do time de produto.