

**Pontifícia Universidade Católica do Rio Grande do Sul**  
**Verificação e Validação de Software**  
**Engenharia de Software**

**Carolina Ferreira, Felipe Freitas, Mateus Caçabuena e Murilo  
Kasperbauer**

***Relatório do Trabalho 2***

**Porto Alegre**  
**2024**

# Introdução

Neste relatório, abordamos a aplicação de testes em um sistema de complexidade média, um Jogo da Velha com análise de tabuleiro por modelo de Inteligência Artificial. Este sistema consiste no *frontend* de um projeto desenvolvido em TypeScript, utilizando o *framework* Angular, e uma *API* desenvolvida em Python com o framework FastAPI para treinar os algoritmos que analisam o estado do jogo. Não há banco de dados além dos dados necessários para o treinamento das Inteligências Artificiais, que não são o objetivo dos testes.

Para obter o código-fonte do sistema, você pode acessar os seguintes repositórios:

- *Frontend*: [github.com/felipefreitassilvalearning/TicTacToePWA](https://github.com/felipefreitassilvalearning/TicTacToePWA)
- *API*: [github.com/EngenhariaSoftwarePUCRS/Inteligencia\\_Artificial/Trabalho01](https://github.com/EngenhariaSoftwarePUCRS/Inteligencia_Artificial/Trabalho01)

## Instruções de Compilação e Execução

### Requisitos

1. **NodeJS** ^16 (Recomendado: 20 LTS)
2. **Python** ^3.9 (Recomendado: 3.10.4)

### Instalação do NodeJS

1. Verifique a versão instalada do Node:

```
node --version
```

2. Caso necessário, instale a versão correta do NodeJS a partir do [site oficial](#).

### Instalação do Python

1. Verifique a versão instalada do Python:

```
python --version
```

2. Caso necessário, instale a versão correta do Python a partir do [site oficial](#).

### Configuração do Ambiente de Desenvolvimento

1. Acesse o repositório do projeto *frontend* (com os testes) e siga o passo-a-passo presente no arquivo `README.md`.
  - Link: [github.com/felipefreitassilvalearning/TicTacToePWA](https://github.com/felipefreitassilvalearning/TicTacToePWA)

2. Para a *API*, recomendamos extrair o arquivo `API.zip` e seguir as instruções no arquivo `README.md`. Se preferir, o código original está disponível no link abaixo:
  - o Link: [github.com/EngenhariaSoftwarePUCRS/Inteligencia\\_Artificial/Trabalho01](https://github.com/EngenhariaSoftwarePUCRS/Inteligencia_Artificial/Trabalho01)

## Acessando o Ambiente

1. O *frontend* está disponível no seguinte endereço:

```
localhost:4200
```

2. Não é necessário o acesso na *API* diretamente, mas está disponível localmente no seguinte endereço:

```
localhost:8080
```

## Como Rodar os Testes

1. Para rodar os testes na mesma pasta do *frontend*, execute o seguinte comando:

```
bash
Copiar código
npm run tests
```

2. Alternativamente, para rodar cada teste separadamente, execute:

```
bash
Copiar código
npm run test:unit
npm run test:integration
npm run test:e2e
```

Com estes passos, o ambiente estará preparado para a execução dos testes e a análise crítica da qualidade dos mesmos, conforme os objetivos estabelecidos para este trabalho.

## Objetivos

Os objetivos dos testes incluem validar as funcionalidades principais do Jogo da Velha, assegurando que o sistema funciona conforme esperado. Nossa jornada de usuário selecionada é:

“Eu, como usuário do sistema de análise do tabuleiro de jogo da velha, quero conseguir interagir com um tabuleiro de jogo da velha e, principalmente, que o modelo analise corretamente meus movimentos e, se houver, os de meu adversário, para que eu consiga validar manualmente a eficácia do modelo que estou treinando.”

Neste programa, foi testado:

- Funcionalidades básicas dos botões da aplicação (marcar X, marcar O, começar um novo jogo)
- Casos de vitória do X, vitória do O e “Velha” (empate)
- Integração com o serviço de avaliação (verificação da disponibilidade do serviço e obtenção dos resultados)
- Apresentação dos resultados na tela após cada movimento do jogo

### Testes já disponibilizados no código

Testes padrão do Angular apenas para o caso de “Component Did Mount”, ou seja, o componente é criado. Apesar de já existirem, dos 5 testes, apenas 4 funcionam e não possuem qualidade pois não estão testando nada que agregue valor ao negócio.

# Casos de Teste

## Testes Unitários

1. Componente Célula (Quadrado – “Square”)
  - a. Cria componente
  - b. Cria botão sem valor algum
  - c. Cria botão com valor X
  - d. Cria botão com valor O
2. Componente Tabuleiro (“Board”)
  - a. Cria componente
  - b. Começa novo jogo
  - c. É inicializado com valores padrão
  - d. É capaz de fazer um movimento que troca o jogador ativo
  - e. É capaz de mudar a quantidade de jogadores (Computador ou Amigo)
  - f. Mostra vencedor corretamente
  - g. Não pode sobreescrever uma célula
  - h. Apresenta predição dos modelos de IA na tela
3. Componente Principal (“App”)
  - a. Deve criar a página
  - b. Deve possuir título correto da aplicação

## Testes de Integração

Os testes de integração foram realizados para verificar o comportamento do endpoint da API do jogo TicTacToe desenvolvido. O objetivo foi validar se o servidor FastAPI retorna os resultados esperados para diferentes configurações do tabuleiro.

### Detalhes dos Casos de Teste

1. Tabuleiro para ‘X\_GANHOU’
  - **Descrição:** Enviar um tabuleiro que resulta na vitória do jogador ‘X’;
  - **Resultado esperado:** O servidor retorna “X GANHOU” como resultado.

## 2. Tabuleiro para 'TEM\_JOGO'

- **Descrição:** Enviar um tabuleiro que ainda possua um jogo em andamento dentro do sistema;
- **Resultado esperado:** O servidor deve retornar "TEM JOGO";

## 3. Tabuleiro para 'VELHA'

- **Descrição:** Enviar um tabuleiro que resulta no empate de uma partida no sistema;
- **Resultado esperado:** O servidor deve retornar "VELHA";

## 4. Tabuleiro inválido

- **Descrição:** enviar um tamanho de tabuleiro inválido (que não possua 9 células como exigido pelo sistema);
- **Resultado esperado:** o servidor deve retornar um erro indicando o tamanho inválido do tabuleiro;

# Testes de Sistema

## 1. Testar vitória para o jogador X

- **Descrição:** este teste verifica se o sistema corretamente identifica uma vitória do jogador X. Para isso, diferentes cenários de vitória (linha, coluna, diagonal) serão testados;
- **Resultado esperado:** mensagem de vitória para X (ex.: "Jogador X venceu!").

## 2. Testar vitória para o jogador O

- **Descrição:** este teste verifica se o sistema corretamente identifica uma vitória do jogador O. Para isso, diferentes cenários de vitória (linha, coluna, diagonal) serão testados;
- **Resultado esperado:** mensagem de vitória para O (ex.: "Jogador O venceu!").

## 3. Testar condição de empate (Velha)

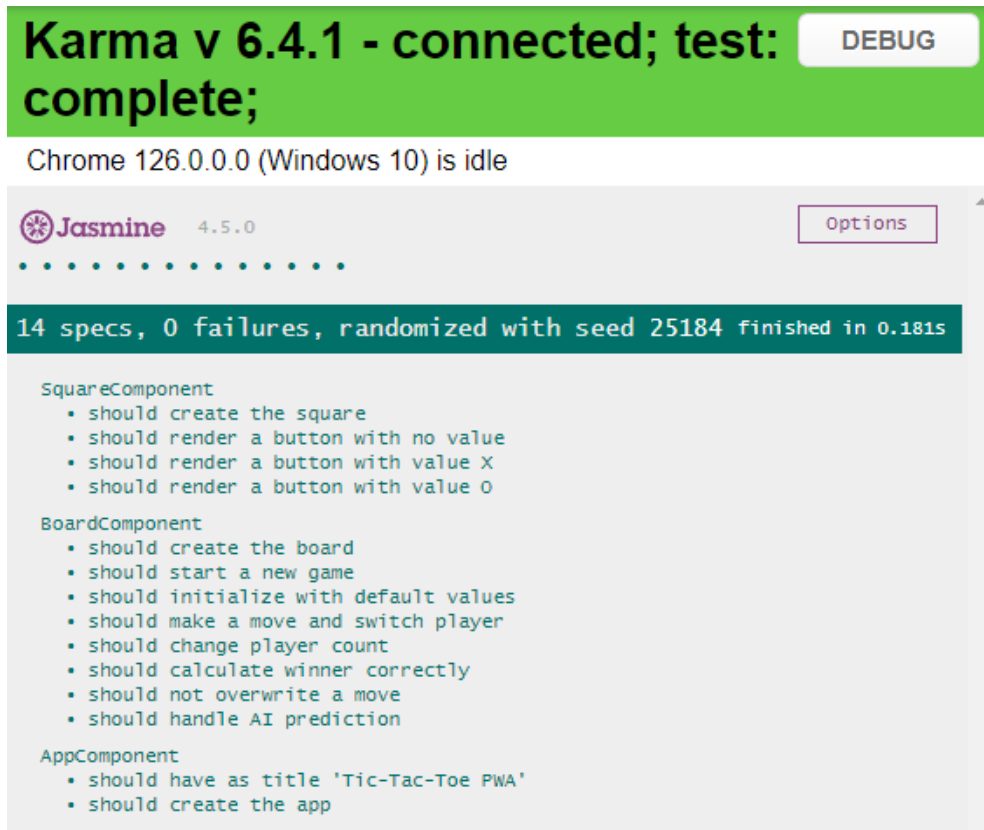
- **Descrição:** este teste verifica se o sistema corretamente identifica um empate na partida. Todas as células devem ser preenchidas sem vencedor;
- **Resultado esperado:** não haver mensagem de vitória de nenhum dos jogadores;

## Resultados dos Testes

Os resultados dos testes realizados são apresentados a seguir, demonstrando a eficácia e a precisão do sistema em identificar corretamente os diferentes cenários de um jogo de Tic-Tac-Toe (Jogo da Velha). Os testes cobriram as verificações unitárias, de integração e de sistema, com evidências visuais para os testes unitários e de sistema, e uma tabela detalhada para os testes de integração.

### Teste Unitário

Os testes unitários verificaram individualmente cada componente do sistema para garantir que eles funcionem corretamente de maneira isolada. Os testes incluíram a criação de componentes, inicialização de botões e tabuleiros, e a execução de ações específicas. As imagens anexadas comprovam que todos os componentes foram criados e funcionaram conforme o esperado:



The screenshot displays the Karma test runner interface. At the top, a green banner indicates 'Karma v 6.4.1 - connected; test: complete;' with a 'DEBUG' button. Below this, it states 'Chrome 126.0.0.0 (Windows 10) is idle'. The main area shows the Jasmine logo and version '4.5.0' with an 'Options' button. A summary bar reports '14 specs, 0 failures, randomized with seed 25184 finished in 0.181s'. The test results are listed under three categories: SquareComponent, BoardComponent, and AppComponent, each with a list of specific test cases.

```
Karma v 6.4.1 - connected; test: complete; DEBUG
Chrome 126.0.0.0 (Windows 10) is idle

Jasmine 4.5.0 Options
14 specs, 0 failures, randomized with seed 25184 finished in 0.181s

SquareComponent
  • should create the square
  • should render a button with no value
  • should render a button with value X
  • should render a button with value O

BoardComponent
  • should create the board
  • should start a new game
  • should initialize with default values
  • should make a move and switch player
  • should change player count
  • should calculate winner correctly
  • should not overwrite a move
  • should handle AI prediction

AppComponent
  • should have as title 'Tic-Tac-Toe PWA'
  • should create the app
```

## Teste de Integração

Os testes de integração foram realizados para validar o comportamento do endpoint da API do jogo Tic-Tac-Toe desenvolvido com FastAPI. O objetivo foi garantir que o servidor retornasse os resultados esperados para diferentes configurações do tabuleiro. A tabela abaixo resume os casos de teste e seus resultados:

Caso de teste	Descrição	Resultado esperado	Resultado obtido	Status
1	Tabuleiro para 'X_GANHOU'	Vitória para 'X'	Vitória do jogador 'X'	Passou
2	Tabuleiro para 'TEM_JOGO'	Jogo em andamento	Jogo continua em andamento	Passou
3	Tabuleiro para 'VELHA'	Empate	Empate entre os jogadores	Passou
4	Tabuleiro inválido	Erro do tamanho do tabuleiro inválido	Error: Invalid board size	Passou

## Teste de Sistema

Os testes de sistema verificaram a funcionalidade completa do jogo, garantindo que todas as interações e resultados finais fossem exibidos corretamente. As imagens anexadas comprovam que o sistema identificou corretamente as vitórias de cada jogador e a condição de empate, apresentando as mensagens esperadas:

```
TicTacToe
Winners
✓ should show that X has won (2469ms)
✓ should show that O has won (799ms)
✓ should show no winner (for tie) (1071ms)
```

```
(Run Finished)

Spec                                Tests  Passing  Failing  Pending  Skipped
✓ gameStates.spec.cy.ts            00:06    3        3        -        -        -
✓ All specs passed!                00:06    3        3        -        -        -
```