

# Interface Pessoa-Máquina

Licenciatura em Engenharia Informática

---

## Ficha Prática #07

---

Rafael Braga

d13414@di.uminho.pt

Daniel Murta

d6203@di.uminho.pt

José Creissac Campos

jose.campos@di.uminho.pt

(v. 2025)

## Conteúdo

<b>1</b>	<b>Objetivos</b>	<b>2</b>
<b>2</b>	<b>Componentes em Vue.js</b>	<b>2</b>
<b>3</b>	<b>Exercícios</b>	<b>2</b>
3.1	Jogo 4 em linha . . . . .	2

## 1 Objetivos

1. Praticar a definição e utilização de componentes na framework Vue.js.

## 2 Componentes em Vue.js

Um componente em Vue.js é uma instância reutilizável que encapsula dados, lógica e apresentação, permitindo a construção de interfaces de utilizador complexas através da composição.

Um componente encapsula três elementos fundamentais: o **template**, que define o *markup* HTML, incluindo o *binding* entre os dados do componente e a sua representação visual na interface; o **script**, onde se declaram, entre outros, as propriedades reativas, os métodos e os *lifecycle hooks* do componente; e a secção de **estilo (style)**, que estiliza o componente de forma isolada ou global através de CSS. Juntos, estes elementos permitem criar unidades de interface coesas e reutilizáveis, facilitando a gestão de estado e a interatividade nas aplicações web modernas.

A comunicação entre componentes é gerida por um sistema de propriedades (*props*) e eventos, otimizando a separação e a reutilização de código. A arquitetura baseada em componentes do Vue.js é essencial para o desenvolvimento eficiente e modular de aplicações web.

## 3 Exercícios

Resolva os seguintes exercícios.

### 3.1 Jogo 4 em linha

Com o conjunto de exercícios abaixo, pretende-se implementar o jogo “4 em linha”. O resultado final pretendido é o apresentado na Figura 1.

Tomando como base a implementação parcial fornecida com esta ficha, resolva então os seguintes exercícios:

1. Estude a classe de domínio representativa do jogo (`models/jogo.js`), em particular os objetos que ela utiliza (`ResultadoJogada` e `ResultadoJogo`) e as propriedades que define:

**tabuleiro** - o tabuleiro de jogo, consistindo num array de arrays (um array de colunas com as jogadas);

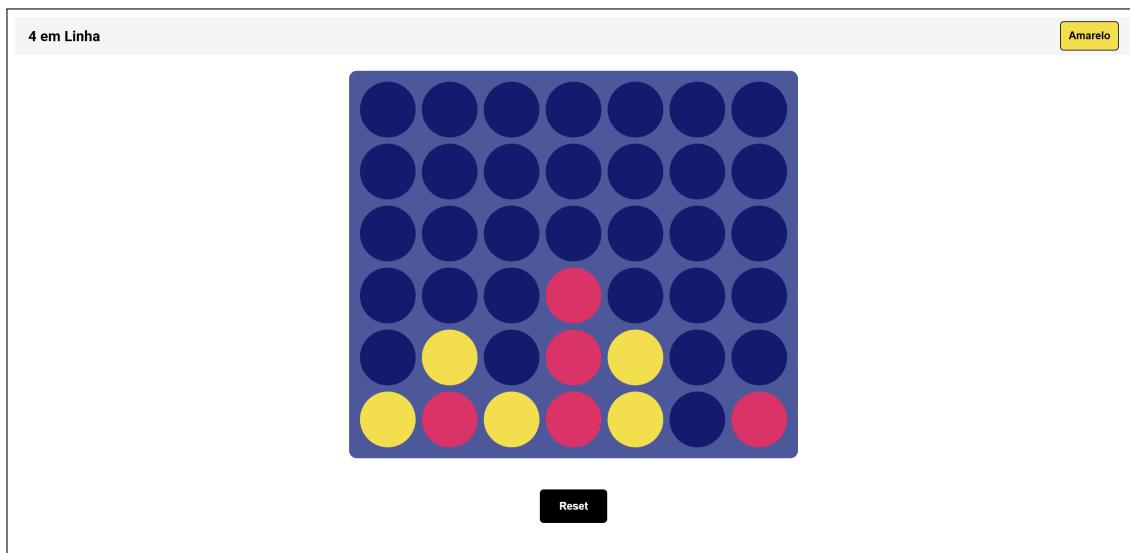


Figura 1: O jogo 4 em linha

**jogadorAJogar** - Indica se é a vez do jogador vermelho (`true`) ou do jogador amarelo (`false`) jogar;

**vencedor** - Indica se venceu o jogador vermelho ou o jogador amarelo, ou se houve empate;

**terminado** - Indica se o jogo já terminou.

2. Estude agora os métodos da classe `Jogo`, completando ou implementando os seguintes métodos:

**validarSeTerminou()** complete o método, implementando a verificação de vitória na diagonal inferior (terá de repetir o processo utilizado para os restantes casos: linha, coluna e diagonal superior);

**tabuleiroCheio()** implemente o método de acordo com a informação fornecida no comentário;

**reset()** implemente o método de acordo com a informação fornecida no comentário.

3. Altere o componente `OJogo`, de modo a que passe a ter uma propriedade `jogo` e que o seu `template` seja desenhado (de forma reativa) de acordo com esse jogo:

- o tabuleiro deverá ter tantas colunas e linhas quanto o jogo ditar, altere o `template` para usar as dimensões definidas no jogo em vez das dimensões *hard-coded*;

- o formato das células do tabuleiro é definido na classe CSS `celula`; no entanto, caso a célula já tenha uma peça, deverá ficar com a cor correspondente ao jogador que a jogou (para definir a cor da célula utilize as classes `red` e `yellow`; note que a implementação fornecida já inclui a declaração de métodos auxiliares que poderão ser úteis na determinação da cor a utilizar, terá de os implementar. Para saber que jogador jogou numa célula, use o método `getCelula` do jogo)<sup>1</sup>.
4. Implemente o evento de `click` sobre cada coluna do tabuleiro, de modo a que se possa jogar o jogo<sup>2</sup>. Um `click` numa coluna deverá introduzir uma peça amarela/vermelha na coluna clicada, dependendo do jogador que esteja a jogar. Como é óbvio, a peça deverá ser colocada na posição vazia mais baixa da coluna. Verifique se consegue fazer jogadas.
  5. Implemente o evento de `click` sobre o botão `Reset` para que seja possível fazer `reset` ao jogo. O botão deverá simplesmente invocar o método `reset`, definido na classe `Jogo`. Verifique se o tabuleiro é efetivamente limpo.
  6. Adicione ao componente `OJogo` os eventos:
    - jogadorAlterado** - que deverá ser emitido sempre que o jogador altera, indicando o jogador a fazer a próxima jogada;
    - jogoTerminou** - que deverá ser emitido quando o jogo terminar, indicado o vencedor do jogo, ou um empate;
    - reset** - que deverá ser emitido sempre que for feito `reset` ao jogo, indicando qual o jogador que deverá iniciar o novo jogo;
7. Complete o componente `Navbar` para adicionar uma barra de navegação à aplicação, que dê indicações do jogador que está a jogar e do estado final do jogo:
    - Se o jogo ainda não tiver terminado, a `navbar` deverá indicar qual o jogador que está a jogar (ver topo da Figura 1);
    - Caso contrário, deverá indicar o resultado final do jogo (ver Figura 2)<sup>3</sup>.

Este componente deverá ser utilizado no componente `App`, sendo que a informação de que ele precisa deverá ser obtida por escuta dos eventos emitidos

<sup>1</sup> Para testar o desenho do tabuleiro, pode definir um tabuleiro com jogadas na propriedade `jogo`.

<sup>2</sup> A implementação fornecida já inclui a declaração do método para funcionar como `event handler`, terá de o implementar.

<sup>3</sup> Relembre que pode utilizar “`display: flex;`” e “`justify-content: space-between;`” para colocar o conteúdo nas margens de um `container`. Para o estilo do botão, veja as definições de estilo do componente `Navbar`.

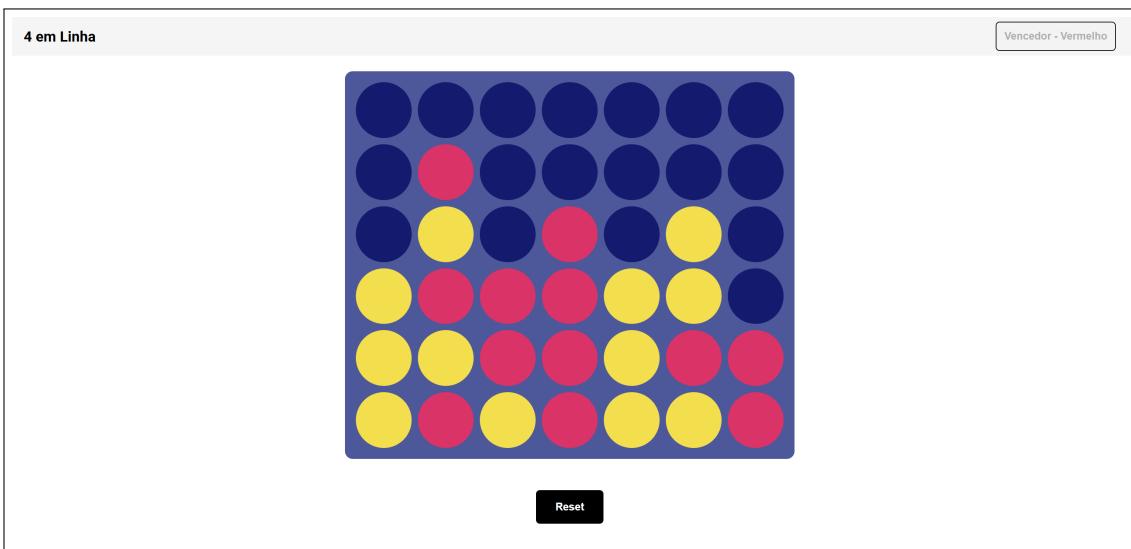


Figura 2: Jogo terminado

pelo componente `OJogo` (terá de definir em `App` as propriedades e métodos necessários a capturar essa informação, de modo a passá-la ao novo componente).

8. Avalie agora a possibilidade de adicionar mais estruturação à implementação, através da definição de novos componentes (por exemplo, um componente para representar colunas e/ou um componente para representar células). Implemente os componentes que considerar necessários.