Universidade Técnica do Atlântico
Departamento de Informática

Trabalho Prático 1- Fitch Fatch Game

Rodrigo Fortes

14-Novembro-2026

Conteúdo

1 Resumo

O projeto é uma implementação de um jogo da velha utilizando React js e Tailwindcss, organizado em componentes. O componente principal, **App**, gerencie o estado do jogo, como o tabuleiro, o jogador atual e o vencedor, utilizando **useState**. Ainteração entre o s componentes ocorre por meio de **props**, com **App** repassando a função de jogada (**handleSelectSquare**) para **GameBoard**, e exibindo **GameOver** quando o jogo termina.

A lógica do jogo verifica se h´q um vencedor após cada jogada, utilizando combinaoes pré-defenidas, e alterando os jogadores até que haja um ganhador ou empate. A estilização com **TailwindCSS** torna o design responsivo e atrativo, com destaque para o tabuleiro e o aspecto visual do componente **GameOver**.

2 Introdução

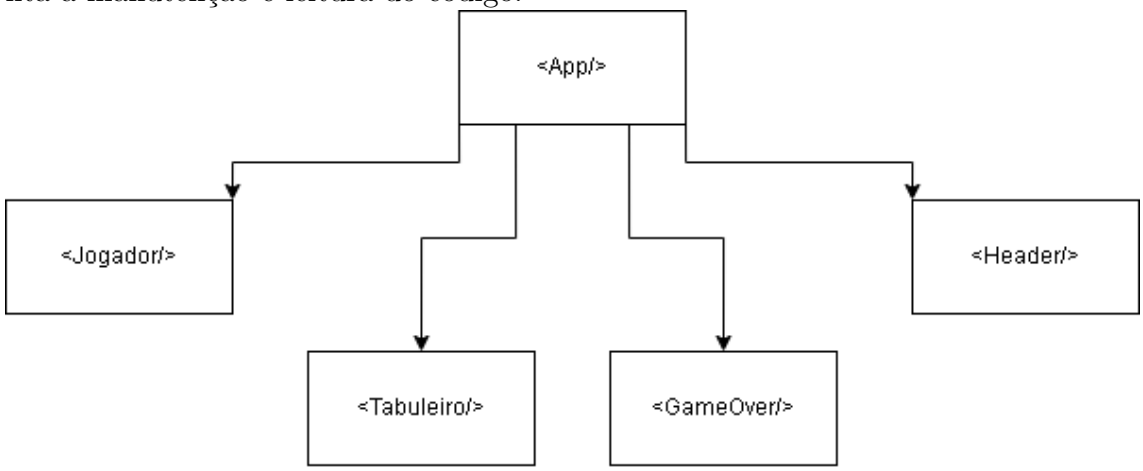
O projeto é um jogo da velha (tic-tac-toe) desenvolvido em React e estilizado com TailwindCSS, utilizando uma abordagem de componentes funcionais e hooks para gerenciar o estado e a lógica do jogo. Este relatório detalha a estrutura e os principais componentes utilizados, assim como os mecanismos de interação e alternância de estado, sem incluir trechos de código diretamente.

3 Estrutura de Componentes

A aplicação é dividida em componentes modulares, cada um responsavél por uma parte específica da interface e da lógica do jogo. Os componentes são:

1. **App**:Componente principal que contém o estado global do jogo e gerencia a lógica principal, incluindo a verificação de vitórias e a alternância de jogadores.
2. **Header**:Exibe o cabeçalho com o logotipo do jogo.
3. **GameBoard**: Representa o tabuleiro do jogo, onde as jogadas são registradas e exibidas
4. **Jogador**:Mostra os jogadores e indica visualmente o jogador atual.
5. **GameOver**: Aparece quando o jogo termina, informando o vencedor ou um empate, com opção para reiniciar.

Cada componente utiliza a abordagem declretiva do Reactjs, o que facilita a manutenção e leitura do código.



4 Hooks e Gerenciamento de Estado

A implentação faz o uso de três hooks principais do react:

- **useState**: Usado para gerenciar o estado do tabuleiro(**board**, do jogador atual (**currentPlayer**) e do vencedor(**winner**).
  - **board**:Representa o tabuleiro em uma matriz 3x3. Cada célula contém null ou o símbolo do jogador atual ("X"ou "O").

- **currentPlayer**:: Alterna entre "X"e "O"após cada jogada.
- **winner**: Armazena o vencedor do jogo ou indica empate, ativando o componente **GameOver** quando defenido.

Os hooks controlam o fluxo do jogo e garantem que as mudanças de estado sejam propagadas de forma eficiente entre os componentes. Por exemplo, **setCurrentPlayer** alterna o jogador e **setBoard** atualiza o tabuleiro para refletir as jogadas em tempo real.

5 Lógica de Jogo e Função de Verificação de Vitória

A lógica do jogo está centralizada no componente App. A função handleSelectSquare é acionada sempre que um jogador faz uma jogada, atualizando o tabuleiro e verificando a presença de um vencedor.

- **Atualização do Tabuleiro** : Após cada jogada, a função cria uma nova versão do tabuleiro com a posição atualizada, usando map para manter a imutabilidade do estado.
- **Verificação de Vitória**: A função checkWinner verifica as combinações vencedoras, definidas pela lista winningCombinations, para determinar se o jogador atual alcançou uma vitória.
  - **Lista winningCombinations** : Essa lista armazena as combinações vencedoras (linhas, colunas e diagonais) em um array bidimensional. Cada subarray representa uma sequência específica de células que, ao serem preenchidas pelo mesmo jogador, resultam na vitória.

Este ciclo continua até que o jogo termine, seja com uma vitória ou um empate (quando todas as células estão preenchidas).

6 Estilização com TailwindCSS

A interface é estilizada usando classes utilitárias do TailwindCSS, o que proporciona um design responsivo e visualmente atrativo:

- Tabuleiro e Células: Estilos como tamanhos fixos, bordas e cores foram aplicados para dar destaque às células do tabuleiro, enquanto o layout flexível se adapta a diferentes resoluções de tela.

- Componentes de Feedback: O componente GameOver utiliza animações e efeitos visuais para informar o término do jogo, criando uma experiência mais envolvente.

TailwindCSS facilita o ajuste visual e a adaptação responsiva do jogo, mantendo o código CSS limpo e centralizado.

7 Comunicação entre Componentes

A comunicação entre componentes ocorre principalmente através de props, respeitando o fluxo de dados descendente do React:

- Propagação de Eventos: A função handleSelectSquare, que altera o estado do jogo, é passada de App para GameBoard como uma prop. Cada célula chama esta função ao ser clicada, permitindo que o componente App processe a jogada e atualize o estado.

- Exibição Condicional: O componente GameOver é exibido condicionalmente com base no estado winner. Quando winner é definido (ou seja, quando há um vencedor ou um empate), GameOver exibe a mensagem correspondente e oferece um botão de reinício.

Essa arquitetura permite que os dados movam de maneira eficaz e que cada componente mantenha seu escopo funcional, o que melhora a escalabilidade do código.

8 Conclusão

A implementação do jogo da velha com React e TailwindCSS é um exemplo eficaz de como construir uma interface interativa e modular. O uso de componentes funcionais e hooks permite um controle detalhado sobre o fluxo de jogo, enquanto o TailwindCSS facilita a criação de uma interface moderna e responsiva.