

NÍVEL BÁSICO



PROJETO 01

```
(CONTEÚDO DISPONÍVEL) {  
PEDRA;  
PAPEL;  
E TESOURA;  
(end);  
})();
```

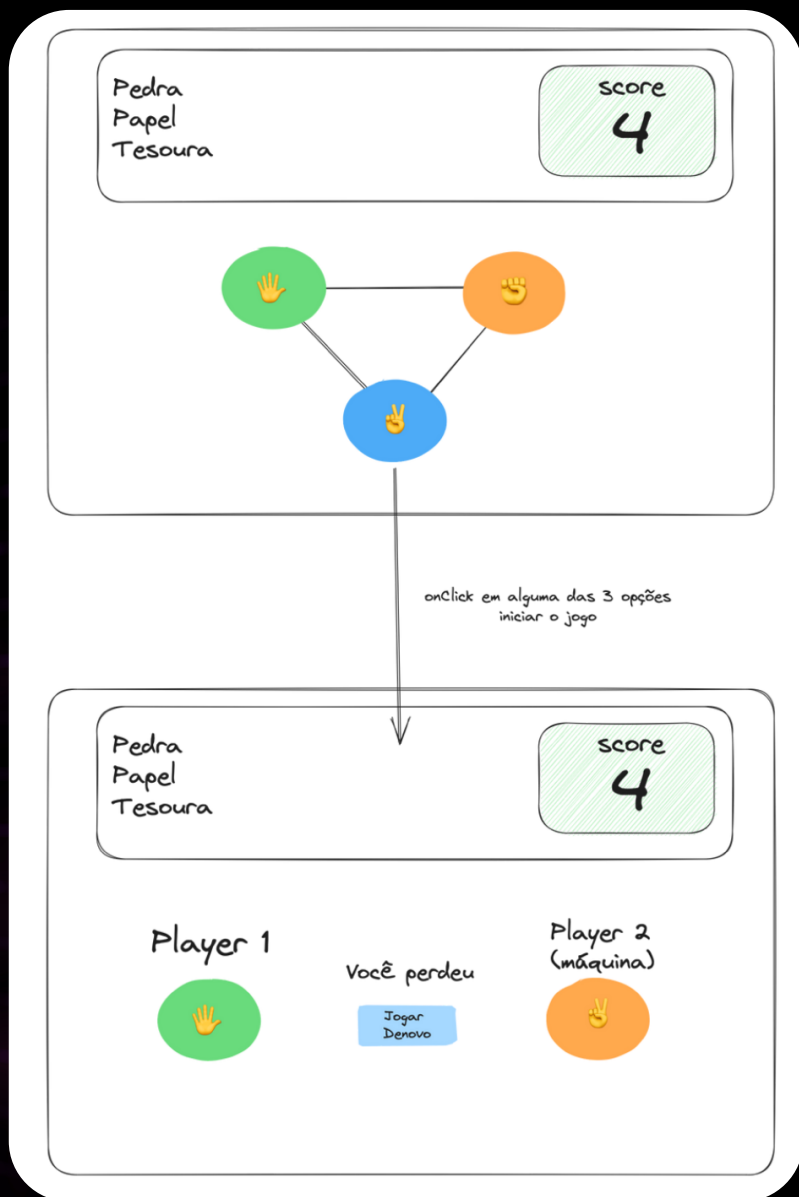
#PORTFÓLIOBOOSTPROGRAM

CONHECIMENTOS REQUIRIDOS:



FRONT-END

WIREFRAME



PEDRA, PAPEL E TESOURA

Vamos criar o clássico jogo pedra, papel e tesoura. Só que em formato de **webapps** e usando a mais recentes **Tech Stack**.



TECH STACK

- ➡ ? React
- ➡ ViteJS
- ➡ TailwindCSS



BRIEFING

Basicamente a grande sacada desse projeto é que vamos poder aplicar conceitos de **conditional rendering**, **react hooks**, e até uma **contagem regressiva**.

REGRAS:

Os jogadores devem simultaneamente esticar a mão, na qual cada um formou um símbolo (**que significa pedra, papel ou tesoura**).

Então, os jogadores comparam os símbolos para decidir quem ganhou, da seguinte forma:

- ➡ Pedra ganha da tesoura (amassando-a ou quebrando-a).
- ➡ Tesoura ganha do papel (cortando-o).
- ➡ Papel ganha da pedra (embrulhando-a).

A **pedra** é simbolizada por um punho fechado; a **tesoura**, por dois dedos esticados; e o **papel**, pela mão aberta.

Caso dois jogadores façam o mesmo gesto, ocorre um empate, e geralmente se joga de novo até desempatar.

Este jogo possui uma única regra:

- ➡ não é permitido mostrar pedra duas vezes seguidas.

NIVEL UM

Implementar uma **solução funcional** que esteja rodando em produção.

NIVEL EXTRA

Crie um botão chamado **regras**, ao clicar nesse botão um modal irá aparecer com todas as regras do jogo **pedra, papel e tesoura**.



REQUISITOS DETALHADOS

Inicie um novo projeto com `npm create vite@latest` dessa forma instalamos a última versão estável do `viteJS` já com `react`. Na hora de selecionar `framework` e `variant`, selecionamos `react` e `javascript`, respectivamente.

Caso você rode `npm run dev` o esperado é ver a tela inicial do `viteJS`.

A estrutura de componentes será da seguinte forma:

- ➡ `Header.js` (a parte do cabeçalho onde exibimos a pontuação)
- ➡ `Play.js` (Fornece 3 opções)
- ➡ `Game.js` (Exibir o resultado)
- ➡ `Footer.js` (O rodapé contém o botão Regras)
- ➡ `Modal.js` (Regras de exibição) - **Feature extra**

Você pode criar 2 rota: `"/` que exibe teu index (`play.js`) e `"/game"` que exibe o resultado do jogo.

Salve o Score utilizando um `useState Hook` e passe o valor para o `header` exibi-los:

```
e.g. Header score={score} e const [score, setScore] = useState(0);
```

O componente **Play** será o componente que iremos exibir as 3 opções disponíveis.

O componente **Game** será o seu grande desafio desse app.

Vale lembrar que para um grau maior de facilidade nesta versão será você (player 1) vs a máquina (player 2). A máquina irá escolher os valores de forma randômica entre os possíveis => "rock", "paper", "scissors"

Uma dica de como poderá ser desenvolvido a função resultado, que será gerado após os players escolherem é uma sequência de **conditional renderings**, exemplo:

```
const Result = () => {  
  if (player1 === "rock" && player2 === "scissors") {  
    setPlayerWin("win");  
    setScore(score + 1);  
  } else if (player1 === "rock" && player2 === "paper") {  
    setPlayerWin("lose");  
    ....  
  } else {  
    setPlayerWin("draw");  
  }  
};
```

A funcionalidade extra é somente um botão que explica todas as regras para quem não está familiarizado com o jogo ainda.

