Criando um site de Rock em 3h com React js

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

Descrição do Projeto

Vamos criar um site que toque rock, as funcionalidades apesar de parecerem simples, acabam sendo um pouco complexas por lidar com áudio!

Vamos criar um carrossel com cinco músicas, ao clicar nas setas esquerda e direita do teclado, ele passa para a seguinte ou a anterior, também tem os botões do carrossel, além de: clicando no próximo album, ele também funciona!

Criando o projeto

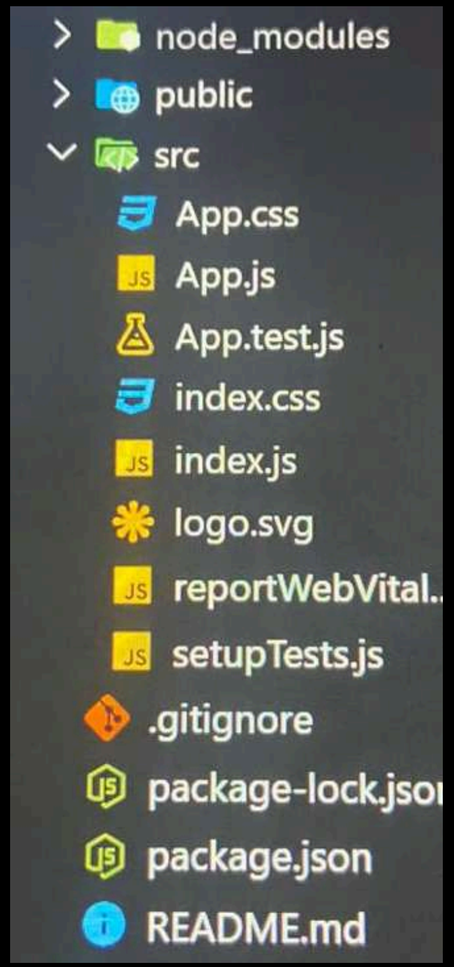
A primeira coisa que faremos é criar o projeto!

Basta abrir uma pasta FORA DA REDE, e dar o comando npx create-react-app nomedoprojeto.

Atenção: O nome do projeto não pode ter espaços, nem letras maiúsculas!

Começando

Agora que criamos nosso projeto, vamos mexer com os seguintes arquivos:



Dentro da pasta src, vamos criar uma pasta app, e colocar os arquivos:
App.css e App.js dentro dessa pasta

Também apagaremos os arquivos App.test.js, logo.svg,reportWebVitals, SetupTests.

Ficará apenas index.js, index.css, e a pasta app, com os dois arquivos dentro!

Editando arquivos

Dentro do arquivo App.css e App.js, vamos apagar tudo.

Em index.js, apagaremos as linhas:

```
import App from './App';
import reportWebVitals from './reportWebVitals';
```

```
reportWebVitals();
```

Dentro da pasta src

Dentro da pasta src, fora da pasta app, criaremos mais três pastas: Assets e Components

Em Assets, colocaremos todos os arquivos de imagem que serão utilizados,assim como os de áudio!

Componentes

No React, utilizamos HTML, CSS e Javascript juntos! Criamos nossa própria tag. Imaginem a tag a (âncora). Ela possui estrutura (<a>), tem estilo (vem subinhada e azul) e funcionalidade (o atributo href faz com que leve para outra página ou seção, o atributo target define em qual aba será aberto).

É exatamente isso que criaremos! Vamos ter uma tag (uma estrutura html), vamos atribuir a ela um estilo (css) e terá funcionalidades (javascript).

Exemplo: Teremos o álbum, e possui imagem,audio,descrição, e nome do album, além das funcionalidades de tocar, passar para a próxima, e vice versa.

Começando

Dentro do nosso arquivo App.js, vocês vão copiar e colar a estrutura que deixei disponível na rede! Aqui estamos importando as imagens, os áudios, o estilo da página e os hooks que utilizaremos!

```
//Importando os Hooks que usaremos
import React, { useState, useRef, useEffect } from 'react';
//Importando o css do projeto
import './Carrossel.css';
//Importando imagens e áudios que usaremos no projeto
import marcus from "../../Assets/marcusUrsal.jpg";
import audio from '../../Assets/Ursal - Me diga o que será preciso.mp3';
import pitty from '../../Assets/pitty2.jpeg';
import audio2 from '../../Assets/Pitty - Me Adora (Clipe Oficial).mp3';
import skank from '../../Assets/skank.webp';
import BS from '../../Assets/baianaSystem.jpeg'
import audio3 from '../../Assets/Água.mp3'
import audio4 from '../../Assets/Skank - Vamos Fugir (Lyric Video).mp3'
import legiao from '../../Assets/LegiãoUrbana.jpeg'
import audio5 from '../../Assets/Tempo Perdido.mp3'
```

Ainda dentro de App.js

```
1  //criando nosso componente App
2  const App = () => {
3      // Definindo o estado para o álbum atual,
4      //inicialmente definido como 0 (primeiro álbum)
5    const [currentAlbum, setCurrentAlbum] = useState(0);
6      // Definindo o estado para a posição de translação no
7      // eixo X, inicialmente definida como 0
8    const [translateX, setTranslateX] = useState(0);
9      // Definindo o estado para o áudio que está tocando,
10     // inicialmente definido como null (nenhum áudio tocando)
11   const [playingAudio, setPlayingAudio] = useState(null);
12     // Criando uma referência para armazenar os elementos de
13     // áudio
14   const audioRefs = useRef([]);
```

Ainda dentro de App.js

```
// Função para ir para o próximo álbum
 const handleNext = () => {
    // Atualiza o estado currentAlbum para o próximo álbum
   // (currentAlbum + 1) % albums.length garante que o
   //índice volta ao início quando chegar ao final da lista
   // de álbuns
   setCurrentAlbum((currentAlbum + 1) % albums.length);
   // Atualiza o estado translateX para transladar a posição
   // da lista de álbuns
   // Cada álbum tem 300px de largura, então diminui 300px a
   // cada avanço
   // O módulo garante que a posição de translação não
   // exceda o número total de álbuns multiplicado por 300
   setTranslateX((translateX - 300) % (albums.length * 300));
 };
```

Ainda dentro de App.js

```
1   //Função que impede o album de ir pro anterior caso esteja no primeiro
2    const handlePrev = () => {
3        // Verifica se o álbum atual é o primeiro (índice 0)
4      if (currentAlbum === 0) return; // não vá para o álbum
5      // anterior se estiver no primeiro
6  // Atualiza o estado currentAlbum para o álbum anterior
7    // (currentAlbum - 1 + albums.length) % albums.length
8    // garante que, se currentAlbum for 0, ele será configurado
9    //para o índice do último álbum
10     setCurrentAlbum((currentAlbum - 1 + albums.length) % albums.length);
11      // Atualiza o estado translateX para mover a lista de álbuns 300 pixels para a direita (para mostrar o álbum anterior)
12   // O módulo % (albums.length * 300) garante que a
13   //translação permaneça dentro dos limites da largura total
14   // dos álbuns
15     setTranslateX((translateX + 300) % (albums.length * 300));
16   };
```

Ainda dentro de App.js

```
//Função que pausa o áudio
 const handlePlayPause = () => {
   //pega como referência o álbum que está tocando atualmente
   const audio = audioRefs.current[currentAlbum];
   //Verifica se a referência tem um áudio
   if (audio) {
     //se o áudio estiver pausado
     if (audio.paused) {
       //se estiver tocando e se esse áudio é diferente
       //do audio que esta em foco
       if (playingAudio && playingAudio !== audio) {
         //se sim, ele para de tocar o audio que estava tocando
         playingAudio.pause();
       }
       //E começa a tocar o audio que esta em foco
       audio.play();
       //Atualiza a variavel que guarda qual áudio está
       // tocando!
       setPlayingAudio(audio);
       //Senão, ou seja, se o audio atual está tocando
     } else {
       //Pausa o audio
       audio.pause();
       //e guarda na variável que nenhum áudio está tocando!
       setPlayingAudio(null);
     }
   }
 };
```

Ainda dentro de App.js

```
1  const handleAlbumClick = (index) => {
2      // Atualiza o estado currentAlbum para o índice do
3      // álbum clicado
4    setCurrentAlbum(index);
5    setTranslateX(-index * 300); // Atualiza o translateX para focar no álbum clicado
6  };
```

Ainda dentro de App.js

```
1  //o useEffect é um hook que define o que é pra acontecer
2   //Assim que entrar na página
3   useEffect(() => {
4     //Nesse caso, ele fica esperando a tecla de seta
5     //para direita ser utilizada pelo usuário, se sim,
6     // chama a função que passa para o próximo álbum
7     const handleKeyDown = (event) => {
8       if (event.key === 'ArrowRight') {
9         handleNext();
10        //Se apertar a tecla de seta para esquerda,
11        //chamamos a função responsável por voltar o álbum
12      } else if (event.key === 'ArrowLeft') {
13        handlePrev();
14        //Caso pressione a tecla de espaço, ele
15        //chama a função que pausa a música
16      } else if (event.key === ' ') {
17        event.preventDefault(); // prevenir scroll da página quando espaço é pressionado
18        handlePlayPause();
19      }
20    };
```

Ainda dentro de App.js

```
1  //Ainda dentro do useEffect
2   // Adiciona um ouvinte de eventos para pressionamentos de
3   //tecla na janela
4     window.addEventListener('keydown', handleKeyDown);
5     return () => {
6         // Retorna uma função de limpeza que será chamada
7         //quando o componente for desmontado ou quando
8         //currentAlbum, translateX, ou playingAudio mudarem
9
10      window.removeEventListener('keydown', handleKeyDown);
11          // Pausa o áudio se algum estiver tocando
12      if (playingAudio) {
13        playingAudio.pause();
14      }
15    };
16  }, [currentAlbum, translateX, playingAudio]);
```

Ainda dentro de App.js

```
useEffect(() => {
  // Reproduzir a nova música ao mudar de álbum, se havia
  // uma música tocando
  if (playingAudio) {// Verifica se há algum áudio tocando
    // Obtém a referência do novo áudio do álbum atual
    const newAudio = audioRefs.current[currentAlbum];
    // Verifica se a referência do novo áudio não é nula
    if (newAudio) {
      //Se não for nulo, paramos o audio antigo
      playingAudio.pause();
      //começamos a tocar o novo áudio
      newAudio.play();
      //e avisamos a variável que controla o áudio atual,
      //que esse é o novo áudio atual
      setPlayingAudio(newAudio);
    }
  }
}, [currentAlbum]);
```

Definindo o que será retornado ao entrar em App

```
return (
   <div className="carousel-wrapper">
     <div className="carousel-container">
       <div className="carousel">
         <div
           className="albums"
           style={{
             //o estilo se transforma, ele se mexe para
             //os lados, dependendo da variável translateX
             transform: `translateX(${translateX}px)`,
           }}
         > {/*Percorrendo a lista dos álbuns e pegando
         o album e a posição dele */}
           {albums.map((album, index) => {
             //testa se o alguns dos albuns tem a posição
             //do album que está em evidência
             const isActive = index === currentAlbum;
```

Definindo o que será retornado ao entrar em App

```
        const isActive = index === currentAlbum;
        return (
          <div
            key={album.id}
          /*Adicionando o estilo diferente caso
              o album esteja ativo */
            className={`album ${isActive ? 'active' : ''}`}
            onClick={() => handleAlbumClick(index)} // Adiciona o evento de clique
          >
            <img src={album.image} alt={album.title} />
            <div className="album-info">
              <h2>{album.title}</h2>
              <p>{album.descricao}</p>
              {album.audio && (
                <audio
                  ref={(el) => (audioRefs.current[index] = el)}
                >
                  <source src={album.audio} type="audio/mp3" />
                </audio>
              )}
            </div>
          </div>
        );
      })}
    </div>
```

Definindo o que será retornado ao entrar em App

```
1
2           <button className="carousel-control prev" onClick={handlePrev}>
3             &#10094;
4           </button>
5           <button className="carousel-control next" onClick={handleNext}>
6             &#10095;
7           </button>
8         </div>
9         <button className="play-pause-btn" onClick={handlePlayPause}>
10          {playingAudio && !playingAudio.paused ? 'Pause' : 'Play'}
11        </button>
12      </div>
13    </div>
14  );
15};
16
17export default App;
```