

## 3ADSs – React Bootcamp – Aula 03 – Tela de Músicas

### ► Objetivo

Objetivo: Continuar desenvolvendo a página de **Músicas**, usar o **HTML-CSS-TEMPLATE** para modelar as páginas que iremos criar, junto com o card de mais informações.

### Requisitos:

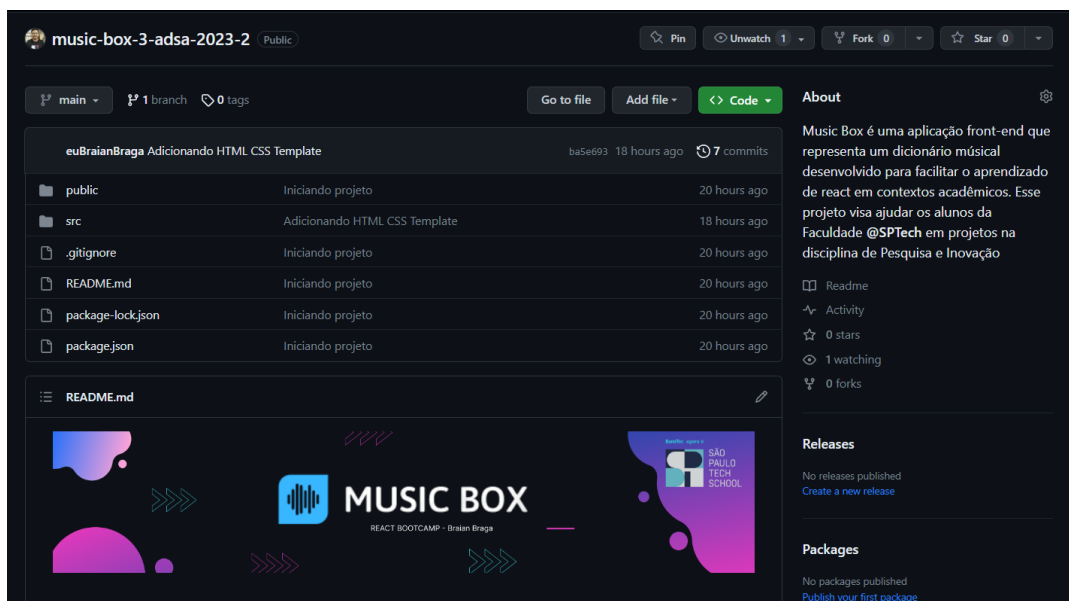
- ✓ Realizar o **setup inicial** do projeto para configuração da sua máquina;

## ► Setup Inicial

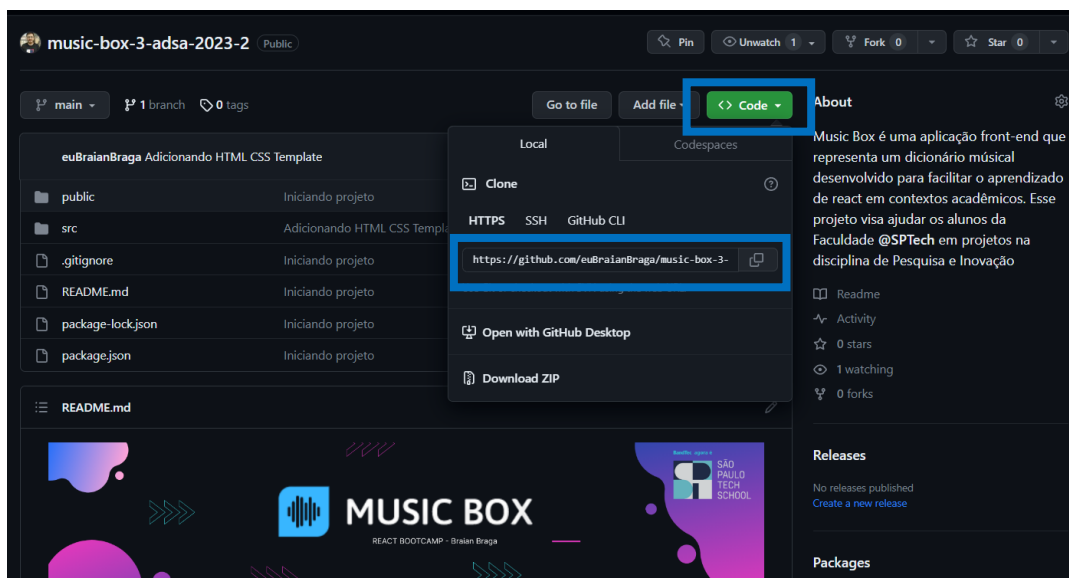
Para realizar a configuração do projeto **Music-Box** em sua máquina:

**Passo 1º :** Acesse o repositório “<https://github.com/euBraianBraga/music-box-3-adssuaturma-2023-2>”.

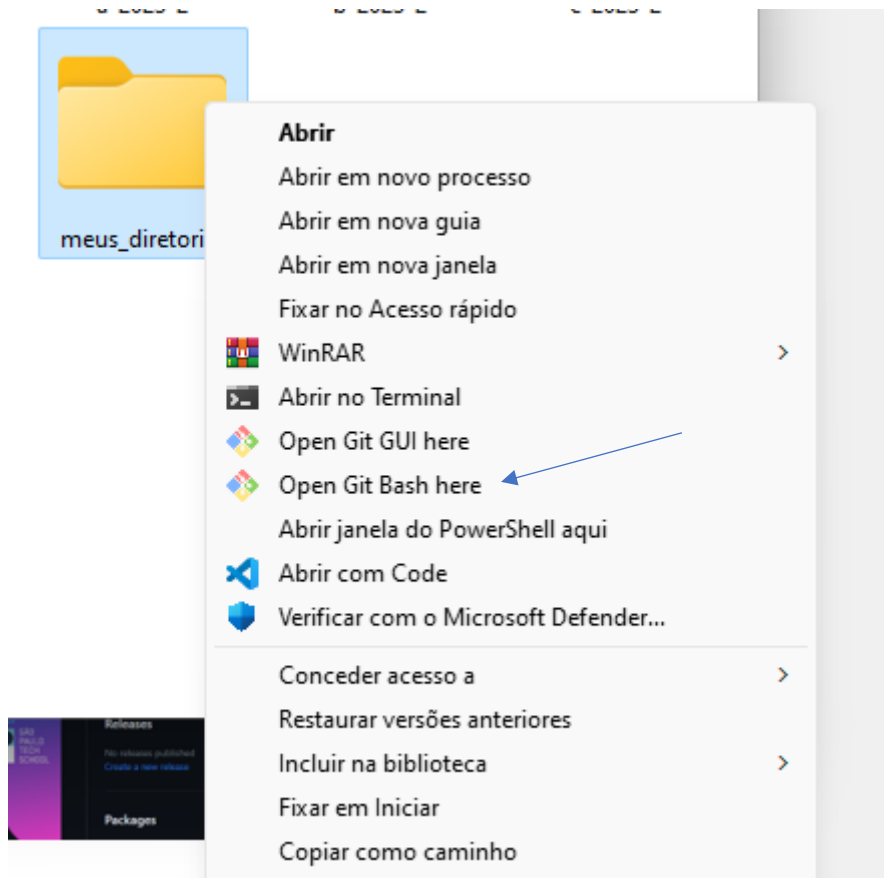
**Exemplo:** Se sua turma for **3ADSA**, o link vai ser o seguinte  
<https://github.com/euBraianBraga/music-box-3-adsa-2023-2>



**Passo 2º :** Clique no botão “**Code**” e copie o endereço HTTPS que aparecer no popup:



**Passo 3° :** Selecione um diretório em sua máquina e abra o terminal de comando “**Git Bash**” dentro do diretório para realizar os comandos para clonar o projeto:



**Passo 4° :** Com o Git Bash aberto, execute o comando “**git clone**”, colando o endereço HTTPS do projeto **Music-Box** que foi mostrado no passo anterior:

### 3ADSA

***git clone <https://github.com/euBraianBraga/music-box-3-adsa-2023-2.git>***

### 3ADSB

***git clone <https://github.com/euBraianBraga/music-box-3-adsb-2023-2.git>***

### 3ADSC

***git clone <https://github.com/euBraianBraga/music-box-3-adsc-2023-2.git>***

Depois da execução do comando acima, o resultado precisa ser parecido com algo assim:

```
MINGW64/c/Users/brain/OneDrive/Área de Trabalho/meus_diretorios
brain@Jade MINGW64 ~/OneDrive/Área de Trabalho/meus_diretorios
$ git clone https://github.com/euBraianBraga/music-box-3-adsc-2023-2.git
Cloning into 'music-box-3-adsc-2023-2'...
remote: Enumerating objects: 62, done.
remote: Counting objects: 100% (62/62), done.
remote: Compressing objects: 100% (54/54), done.
remote: Total 62 (delta 8), reused 54 (delta 5), pack-reused 0
Receiving objects: 100% (62/62), 748.82 KiB | 8.61 MiB/s, done.
Resolving deltas: 100% (8/8), done.
```

Ainda no terminal, execute o comando “**cd music-box-3-adsc-2023-2**” , para conseguir acessar o repositório clonado.

**Passo 4° :** Com o terminal aberto na raiz do repositório, instale as dependências listadas no arquivo “**package.json**” do projeto, utilizando o seguinte comando:

**npm install || npm i**

```
brain@Jade MINGW64 ~/OneDrive/Área de Trabalho/meus_diretorios/music-box-3-adsc-2023-2 (main)
$ npm i

up to date, audited 1502 packages in 2s

241 packages are looking for funding
  run 'npm fund' for details

6 high severity vulnerabilities

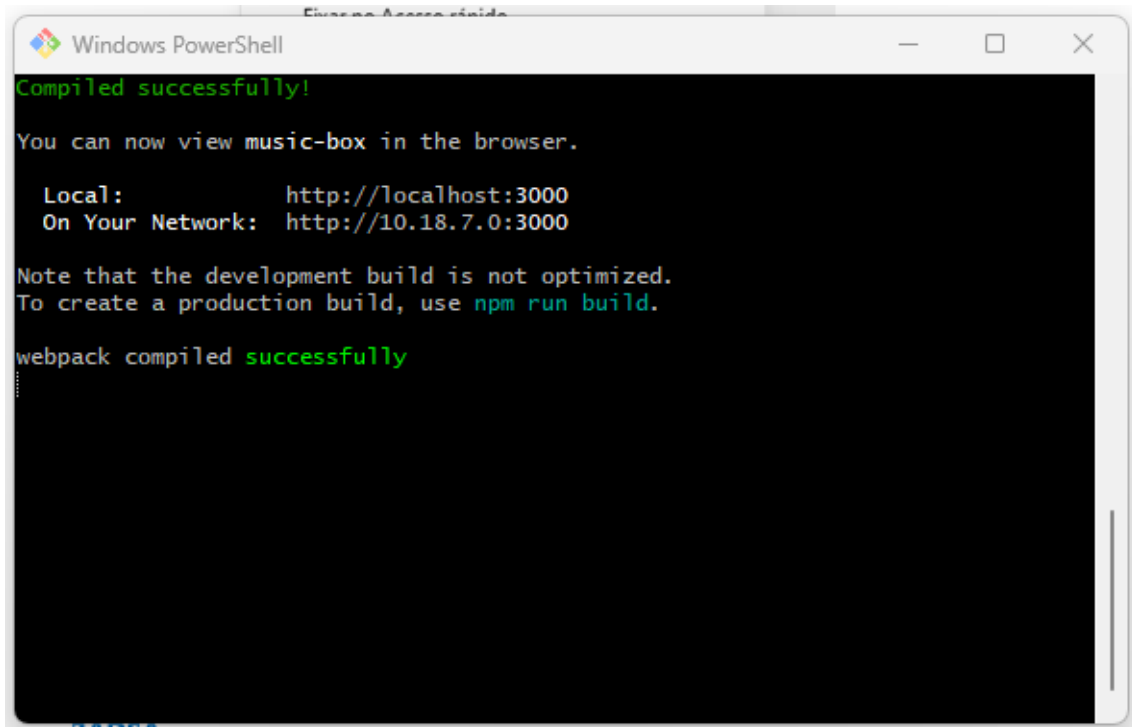
To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.

brain@Jade MINGW64 ~/OneDrive/Área de Trabalho/meus_diretorios/music-box-3-adsc-2023-2 (main)
$ |
```

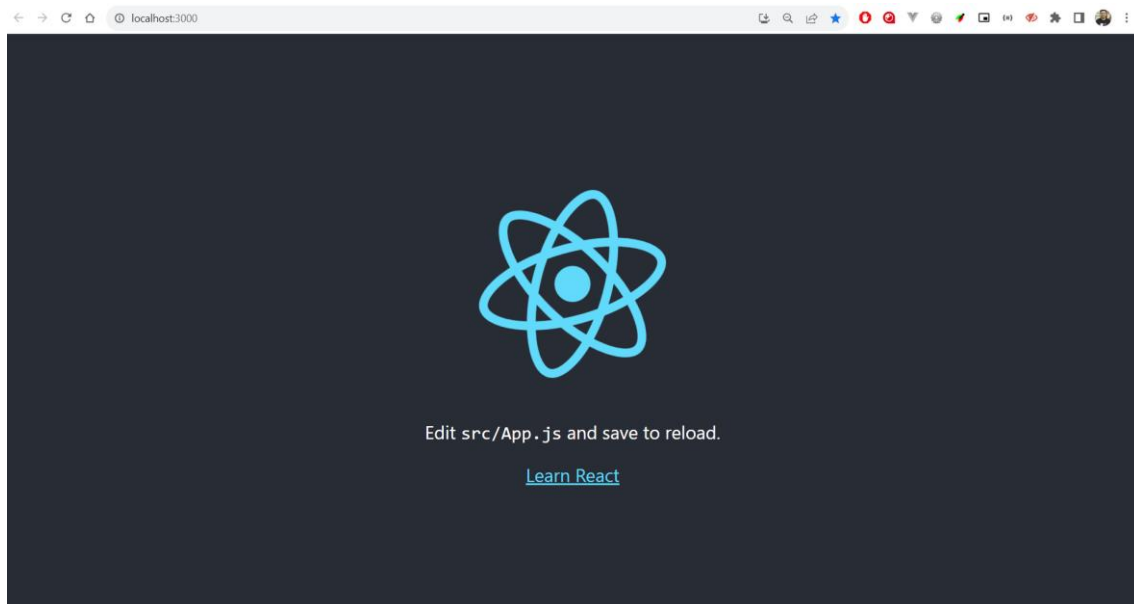
**Passo 5° :** Com o terminal aberto na raiz do repositório, com as dependências instaladas, execute o seguinte comando para executar o projeto:

## npm start



```
Compiled successfully!  
  
You can now view music-box in the browser.  
  
Local:      http://localhost:3000  
On Your Network: http://10.18.7.0:3000  
  
Note that the development build is not optimized.  
To create a production build, use npm run build.  
  
webpack compiled successfully
```

A mensagem de **successfully** indica que o projeto foi iniciado com sucesso, e uma aba em seu navegador principal, será aberta com o logo do react girando no “<http://localhost:3000/>”



**Passo 6° :** Com o projeto rodando, abra o mesmo na sua **IDE** de desenvolvimento, navegue até o arquivo “**api.js**”, altere a **URL** base da instância do axios para o endereço do MockAPI:

```
import axios from "axios";

const api = axios.create({
  |   baseURL: "SUA URL DO MOCKAPI"
  })

export default api;
```

Ao finalizar este passo do “**Setup do Projeto**”, o Music-Box estará configurado em sua **maquina local**, pronto para dar continuidade em seu desenvolvimento.

## ► Estilização

Objetivo: Usar o arquivo **HTML-CSS-TEMPLATE** para modelar as páginas.

### Requisitos:

- ✓ Realizar o **setup inicial** do projeto para configuração da sua máquina;

**Passo 1°** : Vá para o arquivo “**App.js**” e faça o importe do “**style.css**” e “**reset.css**” que são nossos arquivos de estilização do projeto já pronto:

```
import './html-css-template/css/style.css';  
import './html-css-template/css/reset.css';
```

Assim que o importe for feito, sua tela estará com um background preto:



**Passo 2° :** Como não estamos conseguindo ver as músicas que estão vindo da api, vamos apagar todo o código que estiver dentro do return no arquivo “**Musicas.jsx**”:

```
return (  
  <>  
  </>  
);
```

**Obs.:** A função listar não precisa ser apagada.

**Passo 3° :** Vá para o arquivo “**musicas.html**” dentro de **HTML-CSS-TEMPLATE** e copie o conteúdo de dentro da tag **<body>** menos a tag **<nav>** para dentro da tag return de **musicas.jsx**.


```
<div class="container">  
  <div class="filter">  
    <button class="btn">Adicionar</button>  
  </div>  
</div>  
  
<div class="container">  
  <div class="music-boxes">  
    <div class="card-music">  
      <div class="icons">  
          
          
      </div>  
      <div class="info-music">  
        <p>  
          <strong class="card-title">música: </strong>  
          <input class="input-music-enable" type="text" value="Musica" />  
        </p>  
        <p>
```



```
<strong class="card-title">artista: </strong>
<input class="input-music-enable" type="text" value="Teste" />
</p>
<p>
  <strong class="card-title">categoria: </strong>
  <input class="input-music-enable" type="text" value="Teste" />
</p>
<p>
  <strong class="card-title">ano: </strong>
  <input class="input-music-enable" type="text" value="Teste" />
</p>
<button class="btn-salvar-enable">Salvar</button>
</div>
</div>
</div>
</div>
```

O resultado precisa ser parecido com algo assim:

adicionar



música:

artista:

categoria:

ano:

Musica

Teste

Teste

Teste

Salvar

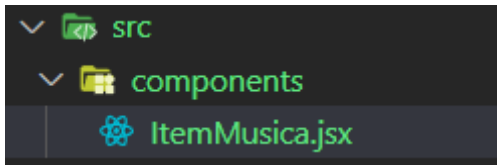
## ► ItemMusica

Objetivo: Desenvolver a página **ItemMusica**.

### Requisitos:

- ✓ Realizar o **setup inicial** do projeto para configuração da sua máquina;

**Passo 1° :** Dentro de “src”, crie uma pasta chamada **components** e dentro de components crie um arquivo chamado **ItemMusica.jsx**:



**Passo 2° :** Crie uma estrutura padrão de pagina no react, como no exemplo abaixo:

```
import React from "react";

function ItemMusica() {
  return (
    <>

    </>
  )
}

export default ItemMusica;
```

**Passo 3° :** Recorte a div “card-music” da página **Musicas.jsx** e cole dentro do **ItemMusica.jsx**:

```
import React from "react";

function ItemMusica() {
  return (
    <>
      <div class="card-music">
        <div class="icons">
          
        </div>
      </div>
    </>
  )
}
```

```

    
  </div>
  <div class="info-music">
    <p>
      <strong class="card-title">música: </strong>
      <input class="input-music-enable" type="text" value="Musica" />
    </p>
    <p>
      <strong class="card-title">artista: </strong>
      <input class="input-music-enable" type="text" value="Teste" />
    </p>
    <p>
      <strong class="card-title">categoria: </strong>
      <input class="input-music-enable" type="text" value="Teste" />
    </p>
    <p>
      <strong class="card-title">ano: </strong>
      <input class="input-music-enable" type="text" value="Teste" />
    </p>
    <button class="btn-salvar-enable">Salvar</button>
  </div>
</div>
</>
);
}

export default ItemMusica;

```

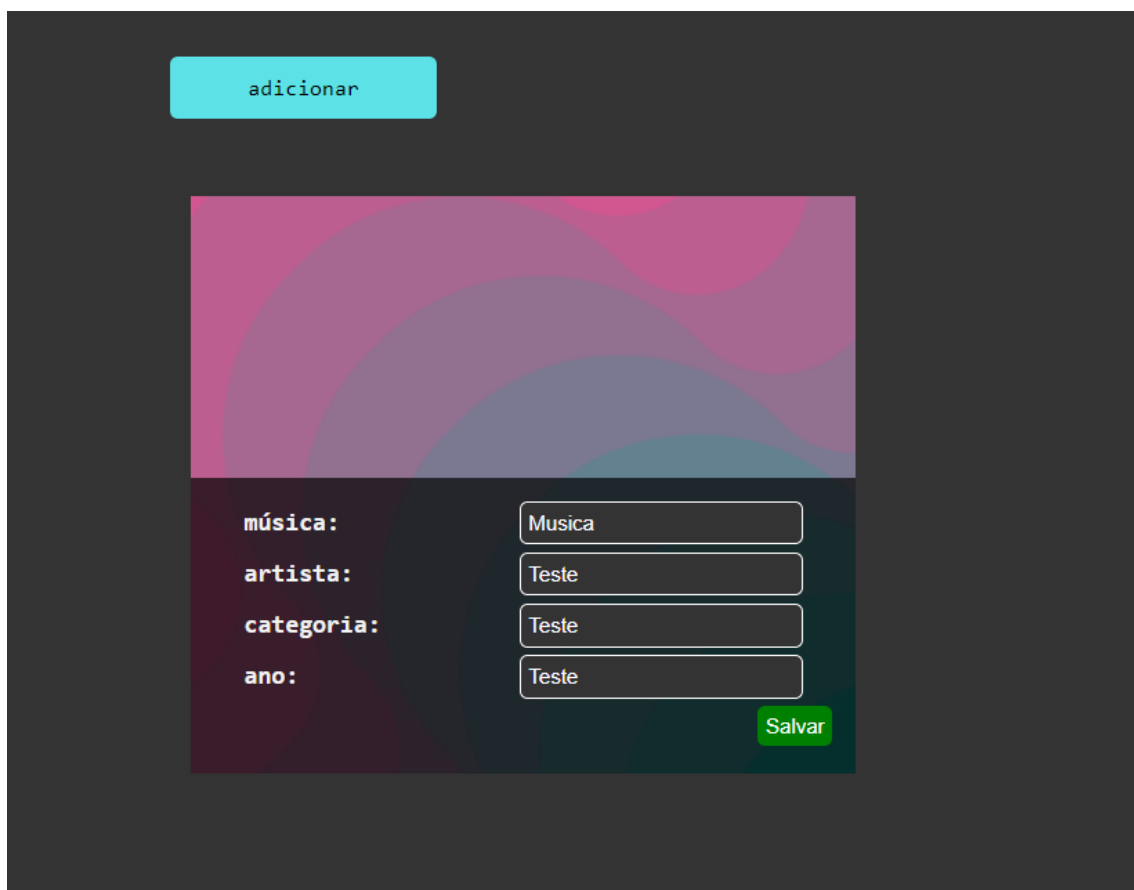
**Passo 3° :** Na página “**Musicas.jsx**” import o **componente** que acabou de ser criado:

```
import ItemMusica from "../components/ItemMusica";
```

**Passo 4° :** No lugar que foi recortado a div “**card-music**”, coloque o importe que foi feito como no exemplo abaixo:

```
<div class="container">
  <div class="music-boxes">
    <ItemMusica />
  </div>
</div>
```

**Obs.:** A quantidade de vezes que você colocar o componente dentro da div “**music-boxes**”, será o número de vezes que o card vai aparecer:



The screenshot shows a web interface with a dark background. At the top, there is a light blue button labeled "adicionar". Below it is a large rectangular area with a pink and purple wavy background. At the bottom of this area is a form with four labels on the left and four input fields on the right. The labels are "música:", "artista:", "categoria:", and "ano:". The input fields contain the text "Musica", "Teste", "Teste", and "Teste" respectively. A green button labeled "Salvar" is located at the bottom right of the form.

## ► Props

Objetivo: Alterar os valores dos inputs de forma **dinâmica** a partir de valores que são passados por props.

### Requisitos:

✓ Realizar a etapa anterior do **ItemMusica**;

**Passo 1° :** No arquivo “**Musicas.jsx**” adicione novos “**atributos || parâmetro**” que vamos utilizar nos inputs:

**Obs.:** Alterem os valores para músicas que vocês gostem.

```
<div class="container">
  <div class="music-boxes">
    <ItemMusica
      nome="Como passar de semestre"
      artista="SprintBreakers"
      genero="Música da boa"
      ano="2023"
    />
    <ItemMusica
      nome="Estudei para a prova"
      artista="SprintBreakers"
      genero="Música da boa"
      ano="2023"
    />
    <ItemMusica
```

```
    nome="Passei de semestre"  
    artista="SprintBreakers"  
    genero="Música da boa"  
    ano="2023"  
  />  
</div>  
</div>
```

**Passo 2° :** No arquivo “**ItemMusica.jsx**” vamos receber esses “**parâmetros**” para conseguir usar nos inputs no lugar do “**value**”:

Existem duas formas principais de receber os **parâmetros**:

**1° :** Usando um único objeto de **props**:

```
function ItemMusica(props) {  
  return (  
    <div>  
      <p>Nome: {props.nome}</p>  
      <p>Artista: {props.artista}</p>  
      <p>Gênero: {props.genero}</p>  
      <p>Ano: {props.ano}</p>  
    </div>  
  );  
}
```

Nesse caso, você está passando um **único objeto de props** para o componente **ItemMusica**. As vantagens dessa abordagem são:

**Organização:** Todas as informações estão **encapsuladas** em um único objeto, o que pode ser mais organizado e fácil de gerenciar.

**Extensibilidade:** Se você precisar adicionar mais propriedades no futuro, pode fazer isso facilmente sem precisar **alterar** a assinatura da função.

**2° :** Usando parâmetros individuais:

```
function ItemMusica(nome, artista, genero, ano) {  
  return (  
    <div>  
      <p>Nome: {nome}</p>  
      <p>Artista: {artista}</p>  
      <p>Gênero: {genero}</p>  
      <p>Ano: {ano}</p>  
    </div>  
  );  
}
```

Nesse caso, você está passando as propriedades individualmente como **parâmetros** para o componente **ItemMusica**. Algumas considerações aqui são:

- **Simplicidade:** Não há necessidade de desestruturar um objeto de props; os valores são diretamente acessíveis como parâmetros.
- **Limitações:** Se você precisar adicionar mais propriedades no futuro, terá que alterar a assinatura da função e atualizar todos os lugares onde ela é chamada.

**Obs.:** Ambas as abordagens têm seus usos, e a escolha depende das **necessidades** do seu projeto. Geralmente, a abordagem usando um único objeto de props é mais **escalável** e mais **organizada**, especialmente em componentes maiores. Ela também facilita a passagem de muitas propriedades.

Por outro lado, a abordagem com **parâmetros individuais** pode ser mais conveniente para **componentes pequenos** ou quando você tem poucas propriedades a serem passadas.



No bootcamp vamos utilizar a **primeira versão**:

```
function ItemMusica(props) {
```

E vamos utilizar nos inputs respectivamente

```
function ItemMusica(props) {  
  return (  
    <>  
    <div class="card-music">  
      <div class="icons">  
          
          
      </div>  
      <div class="info-music">  
        <p>  
          <strong class="card-title">música: </strong>  
          <input class="input-music-enable" type="text" value={props.nome} />  
        </p>  
        <p>  
          <strong class="card-title">artista: </strong>  
          <input class="input-music-enable" type="text" value={props.artista} />  
        </p>  
        <p>  
          <strong class="card-title">categoria: </strong>  
          <input class="input-music-enable" type="text" value={props.genero} />  
        </p>  
        <p>  
          <strong class="card-title">ano: </strong>  
          <input class="input-music-enable" type="text" value={props.ano} />  
        </p>  
      </div>  
    </div>  
  )  
}
```

```
</p>
<button class="btn-salvar-enable">Salvar</button>
</div>
</div>
</>
);
}
```

### ▶ Menu (DESAFIO)

Objetivo: Criar um componente para ser usado como nosso “Menu”.

#### Requisitos:

- ✓ Realizar **todas** as etapas anteriores;

Utilize o html abaixo para criar um componente e utilizar no arquivo “**Musicas.html**” em cima de todas as tags pré-existentis:

```
<nav>
  <div class="container">
    
    
  </div>
</nav>
```

**Obs.:** Caso a imagem não apareça, procure na internet como solucionar esse problema.