

## 3ADSs – React Bootcamp – Aula 02 - Iniciando o Projeto

### ► Objetivo

Objetivo: Desenvolver o "**Music Box**", um projeto **React** que permite aos usuários **visualizarem** uma lista de músicas cadastradas e realizar ações como **edição**, **cadastro** e **exclusão** de músicas existentes. A aplicação será integrada com uma **API** de música para fornecer detalhes dos artistas e faixas, garantindo uma experiência interativa e completa.

### Requisitos:

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

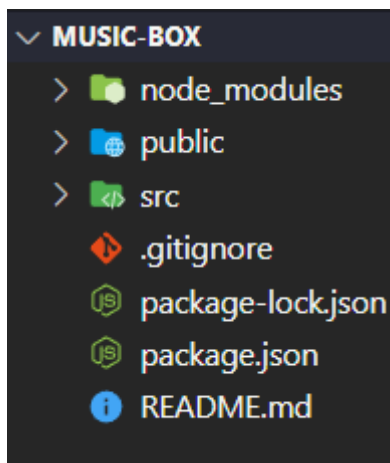
## ► Setup Inicial

**Passo 1: Criação do projeto React** Primeiro, vamos criar o projeto React usando o comando **`npx create-react-app`** || **`npm create-react-app`**. Abra o terminal ou prompt de comando e execute o seguinte comando:

**`npx create-react-app music-box`** || **`npm create-react-app music-box`**

Substitua "**nome-do-seu-projeto**" pelo nome que você deseja dar ao seu projeto, no bootcamp vamos utilizar **music-box**. O comando criará uma pasta com o nome que você escolheu para o projeto e instalará todas as dependências necessárias.

Após criar, ao abrir o projeto a estrutura dos arquivos estará assim:

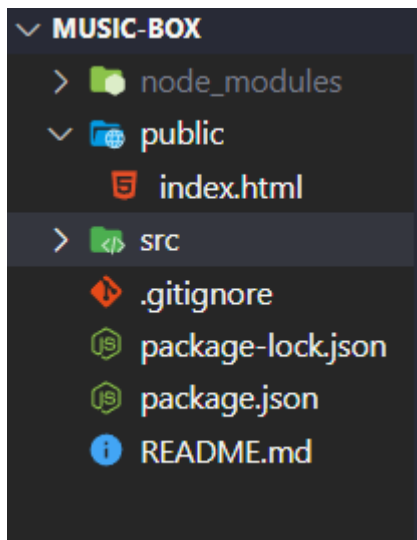


**Passo 2: Limpar o boilerplate** após criar o projeto, siga as etapas para remover os itens do boilerplate conforme descrito:

1. Remova todos os arquivos e deixando somente o **index.html**.

Dentro da pasta "**public**", remova todos os arquivos que forem diferentes do **index.html**.

Após remover, sua pasta public estará assim:



Em seguida, abra o arquivo **"index.html"** que está localizado na mesma pasta e remova algumas linhas que não vão ser usadas durante o bootcamp, deixe o conteúdo do arquivo **"index.html"** como o seguinte:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Music box criado para a materia de pesquisa e inovação da
sptech"
    />
    <title>Music Box</title>
  </head>
  <body>
    <noscript>Você precisa habilitar o javascript da página.</noscript>
    <div id="root"></div>
  </body>
</html>
```

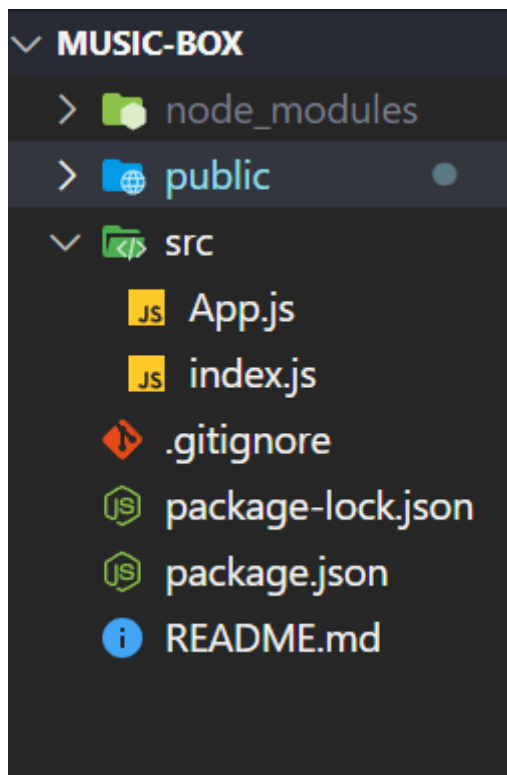
**Obs.:** As linhas que foram removidas são referentes a testes automatizados e imagens padrões que já vem com a criação do projeto.

## 2. Removendo arquivos “desnecessários” da pasta “src”

Dentro da pasta “src”, remova os seguintes arquivos:

- App.test.js
- App.css
- index.css
- logo.svg
- reportWebVitals.js
- setupTests.js

Após remover, sua pasta **src** estará assim:



## 3. Editar o arquivo App.js

Abra o arquivo **"App.js"** localizado na pasta **"src"**. Remova a importação do arquivo **"logo.svg"** e do arquivo **"App.css"**. O conteúdo do arquivo **"App.js"** deve ficar assim:

**Obs.:** Removemos o link e a imagem que não vamos mais utilizar.

```
function App() {  
  return (  
    <div className="App">  
      <h1>Hello World</h1>  
    </div>  
  );  
}  
  
export default App;
```

#### 4. Editar o arquivo index.js

Abra o arquivo **"index.js"** localizado na pasta **"src"**. Remova a importação do arquivo **"index.css"** e do **"reportWebVitals"** junto com sua chamada, a retirada dos comentários e opcional. O conteúdo do arquivo **"index.js"** deve ficar assim:

```
import App from './App';  
import React from 'react';  
import ReactDOM from 'react-dom/client';  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>  
);
```

Após seguir esses passos, você terá criado um projeto **React** e com um **boilerplate limpo**, removendo os itens que não são obrigatórios para começar a desenvolver seu aplicativo React. Lembre-se de que você pode personalizar e adicionar mais recursos ao seu projeto conforme necessário.

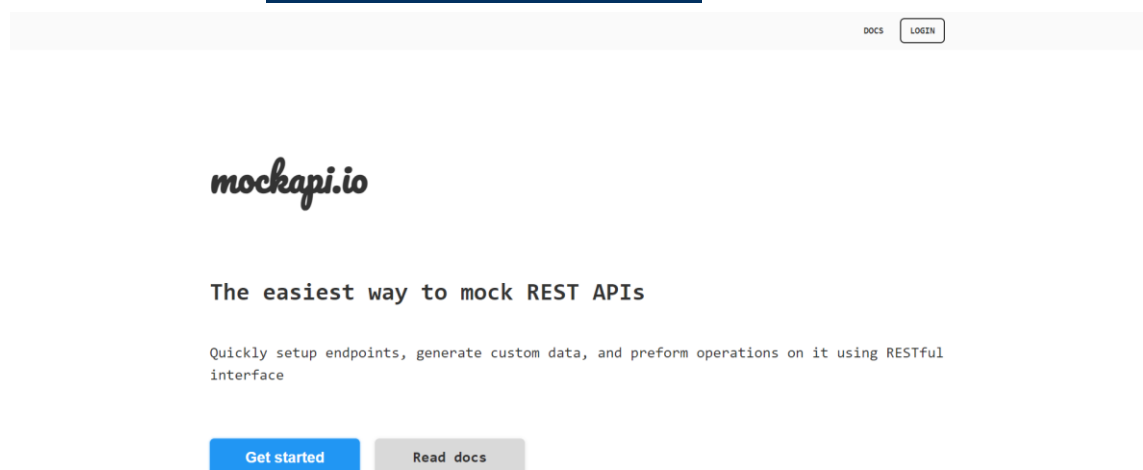
## ► Api Fake

Objetivo: Criar uma "**Api Fake**", que permite aos usuários **visualizarem** uma lista de músicas cadastradas e realizar ações como **edição**, **cadastro** e **exclusão** de músicas com valores pré-definidos pelo usuário.

### Requisitos:


✓ Acesso à internet;


#### 1. Acesse o site <https://mockapi.io/projects>



## 2. Acesse sua conta (Google, GitHub ou registre-se);

### Sign in

 SIGN IN WITH GITHUB

 SIGN IN WITH GOOGLE

---

Email

...

Password

...

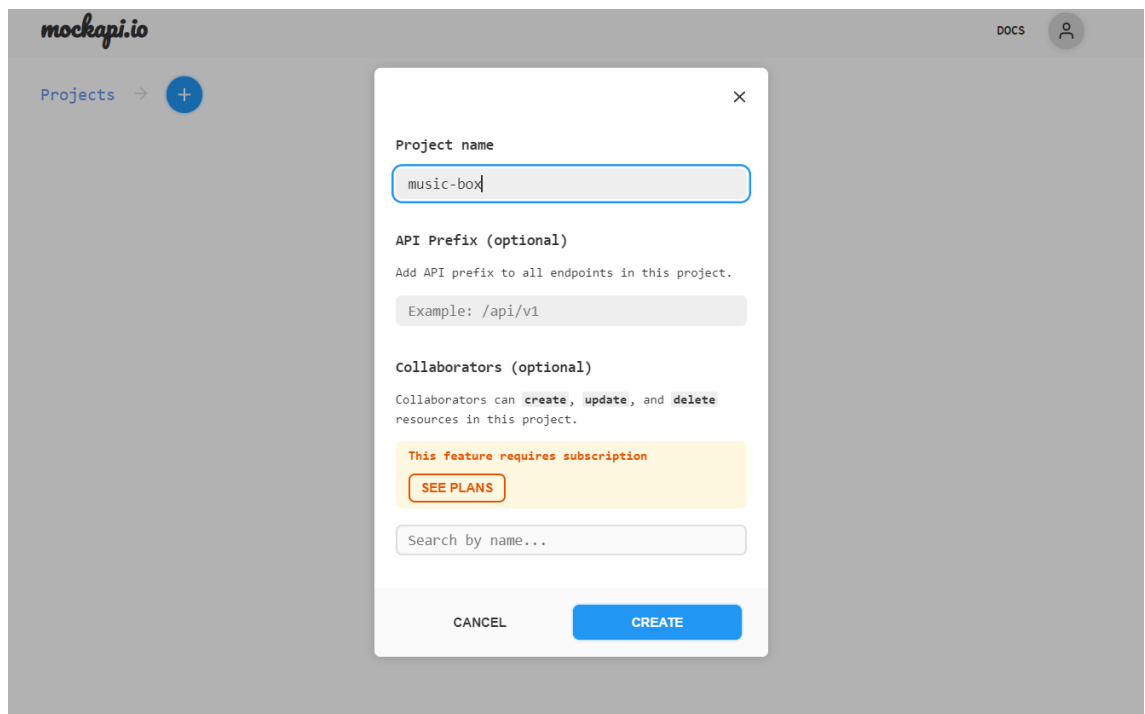
LOGIN

---

Don't have an account? Signup [here](#)

[← Back](#)

3. Nomeie seu projeto de **music-box** e confirme a criação;



mockapi.io

Projects → +

Project name

music-box

API Prefix (optional)

Add API prefix to all endpoints in this project.

Example: /api/v1

Collaborators (optional)

Collaborators can create, update, and delete resources in this project.

This feature requires subscription

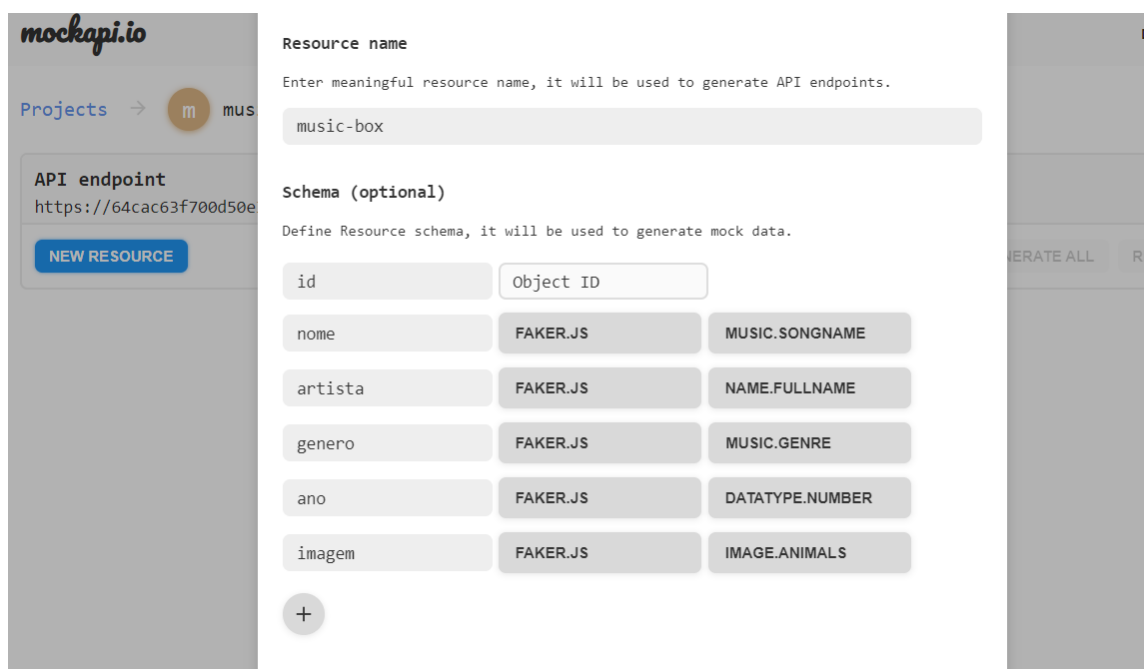
SEE PLANS

Search by name...

CANCEL CREATE

4. Clique no projeto criado e depois em **“NEW RESOURCE”**;

5. Preencha a janela com os dados abaixo:



mockapi.io

Projects → m music-box

API endpoint  
https://64cac63f700d50e...

NEW RESOURCE

Resource name

Enter meaningful resource name, it will be used to generate API endpoints.

music-box

Schema (optional)

Define Resource schema, it will be used to generate mock data.

id	Object ID
nome	FAKER.JS MUSIC.SONGNAME
artista	FAKER.JS NAME.FULLNAME
genero	FAKER.JS MUSIC.GENRE
ano	FAKER.JS DATATYPE.NUMBER
imagem	FAKER.JS IMAGE.ANIMALS

+

Após seguir esses passos, você terá criado uma **API** em minutos com os dados escolhidos para ser uma **API** de músicas.



## ► Axios

Objetivo: Instalar a **biblioteca** que vai ser responsável por fazer as requisições durante o bootcamp.

### Requisitos:

✓ Node instalado na máquina.

1. Abrir o terminal e execute o comando dentro da pasta do projeto

***npm i axios || npm install axios***

O resultado que precisa ocorrer no **terminal** e uma janela parecida com essa:

```
$ npm i axios

added 3 packages, and audited 1505 packages in 4s

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.
```

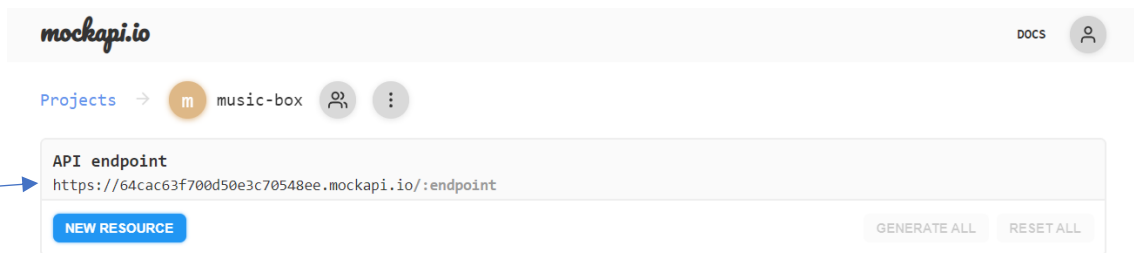
## ► Instância do Axios no React

Objetivo: Agora que o axios está instalado no projeto, precisamos fazer uma instância do mesmo para conseguir fazer a chamada da requisição.

### Requisitos:

✓ Ter feito o passo anterior da instalação do axios.

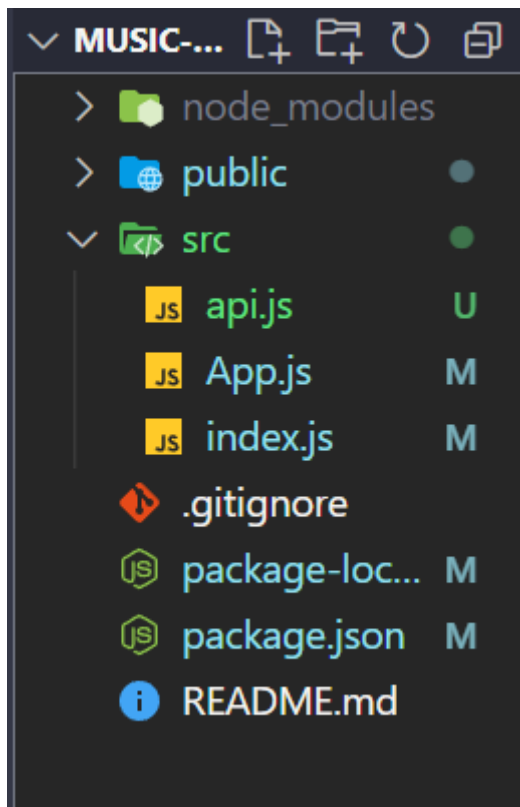
1. Abrir o projeto que foi criado em passos anteriores do **MockAPI**;
2. Copiar a **URL** https do projeto para ser usada como nossa **API**;



**Obs.:** Não usar a **URL** acima foi criada para fins didáticos.

Agora que temos a **URL** da nossa **API**, podemos ir ao código e criar a instância para conseguir fazer as requisições.

Dentro da pasta “**src**” do seu projeto, crie um arquivo chamado “**api.js**”:



Dentro do arquivo **api.js** vamos seguir os passos:

**1°.** Importando a biblioteca do Axios:

```
import axios from "axios";
```

Nesta primeira linha, o código está importando a **biblioteca** Axios. Axios é uma biblioteca popular utilizada para fazer requisições HTTP em navegadores e Node.js. Ela simplifica a comunicação com **APIs** e **recursos externos**.

2°. Criando uma instância do axios:

```
const api = axios.create({  
  baseURL: "https://6271a3e325fed8fcb5e909f1.mockapi.io/musicas"  
});
```

Aqui, o código está criando uma instância do Axios chamada "**api**" utilizando o método "**create**". Esta instância é configurada com a opção "**baseURL**", que define a URL base para todas as requisições feitas com essa instância do Axios. No caso, a **URL** base é <https://6271a3e325fed8fcb5e909f1.mockapi.io/musicas>, que criamos no MockAPI.

3°. Exportando a instância do Axios:

```
export default api;
```

Por fim, o código exporta a instância do Axios criada com o nome "**api**". Ao fazer isso, outros módulos (**arquivos**) do projeto poderão importar e utilizar essa instância para realizar requisições à API fake criada com o **MockAPI**, que está hospedada na URL base especificada.

## ► Prática

### 1°. Exemplo

```
import api from "./api";

function App () {
  return (
    <>
    <h1>Titulo</h1>
    </>
  );
}

export default App;
```

### 2°. Exemplo

```
import api from "./api";

function App () {

  function listar() {
    console.log(api.get());
  }

  return (
    <>
    <h1>Titulo</h1>
    <button onClick={listar}>Listar</button>
    </>
  );
}

export default App;
```

### 3°. Exemplo

```
import api from "./api";

function App () {

  function listar() {
    api.get()
    .then((respostaObtida) => {
      // cairá aqui se a requisição for realizada
      console.log(respostaObtida);
    })
    .catch((erroOcorrido) => {
      // cairá aqui se houver algum erro durante a requisição
      console.log(erroOcorrido);
    })
  }

  return (
    <>
    <h1>Titulo</h1>
    <button onClick={listar}>Listar</button>
    </>
  );
}

export default App;
```

#### 4°. Exemplo

```
import api from "./api";

function App () {

  function listar() {
    api.get()
      .then((respostaObtida) => {
        // cairá aqui se a requisição for realizada
        console.log(respostaObtida);
        // objeto que representa a resposta enviada pela API;
        console.log(respostaObtida.status);
        // vendo status da resposta (OK - 200)
        console.log(respostaObtida.data);
        // vendo os dados da resposta (data: [])
      })
      .catch((erroOcorrido) => {
        // cairá aqui se houver algum erro durante a requisição
        console.log(erroOcorrido);
      })
  }

  return (
    <>
    <h1>Titulo</h1>
    <button onClick={listar}>Listar</button>
    </>
  );
}
```

```
}  
  
export default App;
```

## 5°. Exemplo

```
import api from "./api";  
import { useState } from "react";  
  
function App () {  
  
  const [musicas, setMusicas] = useState([]);  
  // criando state com valor de um vetor vazio;  
  
  function listar() {  
    api.get()  
      .then((respostaObtida) => {  
        // cairá aqui se a requisição for realizada;  
        console.log(respostaObtida);  
        // objeto que representa a resposta enviada pela API;  
        console.log(respostaObtida.status);  
        // vendo status da resposta (OK - 200);  
        console.log(respostaObtida.data);  
        // vendo os dados da resposta (data: []);  
  
        setMusicas(respostaObtida.data)  
        // setando "musicas" com os mesmos dados recebidos pela resposta da  
        // requisição;  
      })  
      .catch((erroOcorrido) => { // cairá aqui se houver algum erro durante a  
        // requisição  
        console.log(erroOcorrido);  
      })  
  }  
  
  return (  
    <>  
    <h1>Titulo</h1>  
  )  
}
```

```
<button onClick={listar}>Listar</button>
</>
);
}

export default App;
```

## 6°. Bônus

```
{
  listaMusicas.map(musica => (
    <div key={musica.id}>
      <h1>{musica.nome}</h1>
    </div>
  ))
}
```