JORNADA FULL STACK IMPRESSIONADOR

Aula 2

Aula 2
Aula 2

Aula 2 – Instalando o react-router-dom

🧠 O que é react-router-dom?

É a biblioteca que permite que você **crie rotas (caminhos de navegação)** no seu site em React, **sem recarregar a página**. Por exemplo:

- "/" → Página Inicial
- "/artistas/23" → Página de um artista específico

📦 Como instalar o react-router-dom no seu projeto React

👉 Passo 1: Abrir o terminal

Abra o terminal do seu projeto React (onde você criou o projeto com Vite ou Create React App).
Se estiver usando o VS Code, você pode abrir o terminal com:

```
Ctrl + `
```

👉 Passo 2: Rodar o comando de instalação

No terminal, digite o seguinte comando:

```
npm install react-router-dom
```

Esse comando vai:

- Baixar a biblioteca react-router-dom.
- Adicionar ela no seu package.json, dentro de dependencies.

Aula 2 – Criando o componente SingleItem

Nesta aula, você vai aprender como criar um **componente visual reutilizável em React**, que recebe informações dinâmicas por **props** e exibe uma imagem com um ícone sobreposto, além de um texto.
Ele também possui um **link dinâmico** que usa o react-router-dom.

🧩 O que são props?

Props são as informações que você envia para um componente. Elas funcionam como **parâmetros** de uma função.

```
const SingleItem = ({ id, name, image, banner, artist, idPath }) => {
```

Esses valores são usados dentro do componente para montar o conteúdo dinamicamente.

🖼️ Exibindo Imagem e Ícone

- A imagem vem da prop image.
- O ícone é do FontAwesome e aparece por cima da imagem.
- Você pode estilizar o ícone com CSS para aparecer no canto, com position: absolute.

```
<img
  className="single-item__image"
  src={image}
  alt={`Imagem do Artista ${name}`}
/>

<FontAwesomeIcon className="single-item__icon" icon={faCirclePlay} />
```

Aula 2 – Criando o componente SingleItem

🔗 Navegação com Link

- O Link vem do react-router-dom.
- Ele cria um link dinâmico, por exemplo: /artistas/23.
- O valor de idPath é algo como "/artistas" e id é "23".

```
<Link to={`${idPath}/${id}`} className="single-item">
```

📝 Exibindo Textos

- Mostra o nome do artista ou item.
- Se artist não for passado, ele mostra "Artista" como padrão (?? é o operador de coalescência nula).

```
<p className="single-item__title">{name}</p>
<p className="single-item__type">{artist ?? "Artista"}</p>
```

Aula 2 – Criando o componente SingleItem

🧪 Exemplo de Uso (Com Renderização Condicional)
Você pode renderizar vários SingleItem com base em uma condição:

```
// Exemplo didático de uso do componente SingleItem com props
{
  items === 5 ? (
    <>
      <SingleItem id="1" name="Artista 1" image="url1" idPath="/artistas" />
      <SingleItem id="2" name="Artista 2" image="url2" idPath="/artistas" />
      <SingleItem id="3" name="Artista 3" image="url3" idPath="/artistas" />
      <SingleItem id="4" name="Artista 4" image="url4" idPath="/artistas" />
      <SingleItem id="5" name="Artista 5" image="url5" idPath="/artistas" />
    </>
  ) : (
    <>
      <SingleItem id="1" name="Artista 1" image="url1" idPath="/artistas" />
      <SingleItem id="2" name="Artista 2" image="url2" idPath="/artistas" />
      <SingleItem id="3" name="Artista 3" image="url3" idPath="/artistas" />
      <SingleItem id="4" name="Artista 4" image="url4" idPath="/artistas" />
      <SingleItem id="5" name="Artista 5" image="url5" idPath="/artistas" />
      <SingleItem id="6" name="Artista 6" image="url6" idPath="/artistas" />
      <SingleItem id="7" name="Artista 7" image="url7" idPath="/artistas" />
      <SingleItem id="8" name="Artista 8" image="url8" idPath="/artistas" />
      <SingleItem id="9" name="Artista 9" image="url9" idPath="/artistas" />
      <SingleItem id="10" name="Artista 10" image="url10" idPath="/artistas" />
    </>
```

Aula 2 – Criando o componente SingleItem

📦 **Código Completo do Componente**

```jsx
// SingleItem.jsx

import React from "react";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faCirclePlay } from "@fortawesome/free-solid-svg-icons";
import { Link } from "react-router-dom";

// Componentes em React recebem "props"
const SingleItem = ({ id, name, image, banner, artist, idPath }) => {
  return (
    <Link to={`${idPath}/${id}`} className="single-item">
      <div className="single-item__div-image-button">
        <div className="single-item__div-image">
          <img
            className="single-item__image"
            src={image}
            alt={`Imagem do Artista ${name}`}
          />
        </div>
        {/* Ícone de "Play" sobre a imagem */}
        <FontAwesomeIcon className="single-item__icon" icon={faCirclePlay} />
      </div>

      {/* Textos do item */}
      <div className="single-item__texts">
        <div className="single-item__2lines">
          <p className="single-item__title">{name}</p>
        </div>

        <p className="single-item__type">{artist ?? "Artista"}</p>
      </div>
    </Link>
  );
};
```

Aula 2 – Criando o componente ItemList

Nesta parte, vamos aprender a criar um **componente que exibe uma lista de itens dinamicamente**, com título, botão de ver tudo e uso do componente SingleItem que já criamos anteriormente.

🧠 Conceitos Didáticos Envolvidos

🧩 1. **Props no Componente ItemList**
O componente ItemList recebe essas props:

Prop	Descrição
`title`	Título da seção. Ex: "Álbuns", "Filmes", "Séries".
`items`	Quantidade de itens que devem aparecer na tela.
`itemsArray`	Um array com todos os objetos que contêm os dados para cada item.
`path`	Caminho usado para o botão "Mostrar tudo".
`idPath`	Parte da URL usada para navegação dinâmica em `SingleItem`.

🔗 2. **Link para rota “Mostrar tudo”**

Esse Link vem do react-router-dom e permite que o botão leve o usuário para outra rota da aplicação, como: /series, /albuns, etc.

```
<Link to={path} className="item-list__link">
  Mostrar tudo
</Link>
```

🎟️ 3. **Filtrando e exibindo apenas os primeiros items**

Isso é usado para limitar a quantidade de itens exibidos na tela. Exemplo:

- Se items for 5, serão mostrados apenas os **5 primeiros elementos** do array.

```
itemsArray
  .filter((_, index) => index < items)
```

Aula 2 – Criando o componente ItemList

🔁 4. **Usando .map() para criar os componentes**

```
.map((currObj, index) => (
  <SingleItem
    idPath={idPath}
    {...currObj}
    key={`${title}-${index}`}
  />
))
```

Explicação:

- O map() percorre o array e **retorna um componente para cada objeto**.
- Cada currObj é um objeto com dados de um item (ex: nome, imagem, id).
- O key ajuda o React a controlar os elementos renderizados.

✨ 5. **Usando o Spread Operator (...)**

```
<SingleItem
  idPath={idPath}
  {...currObj}
/>
```

- O operador ...currObj distribui automaticamente todas as propriedades do objeto como props.
- Por exemplo, se currObj = { id: 1, name: "Série A", image: "..." }, então:

```
<SingleItem id={1} name="Série A" image="..." />
```

Isso reduz a repetição de código e torna a renderização mais elegante.

Aula 2 – Criando o componente ItemList

🧠 **Comentário Extra: Alternativa com Array.fill()**

```
// Exemplo alternativo para criar X componentes vazios:
Array(items).fill().map((_, index) => (
  <SingleItem key={index} />
));
```

🟡 Útil para placeholders ou loading antes de carregar os dados reais.

- Você aprendeu a construir um componente que **gera uma lista de itens dinamicamente**.
- Exploramos conceitos como **props, filter(), map(), operador spread**, uso de Link e renderização por composição.
- Essa abordagem é escalável e permite montar **várias listas diferentes com um único componente genérico**.

Aula 2 – Criando o componente ItemList

📦 Estrutura do Componente

```jsx
// ItemList.jsx
import React from "react";
import SingleItem from "./SingleItem";
import { Link } from "react-router-dom";

const ItemList = ({ title, items, itemsArray, path, idPath }) => {
  return (
    <div className="item-list">
      <div className="item-list__header">
        <h2>{title} populares</h2>
        <Link to={path} className="item-list__link">
          Mostrar tudo
        </Link>
      </div>
      <div className="item-list__container">
        {itemsArray
          .filter((_, index) => index < items)
          .map((currObj, index) => (
            <SingleItem
              idPath={idPath}
              {...currObj}
              key={`${title}-${index}`}
            />
          ))}
      </div>
    </div>
  );
};

export default ItemList;
```

Aula 2 – Refatorando o Componente Main com Componentização

vamos aprender como **refatorar o componente Main** para deixá-lo mais organizado, utilizando o componente genérico ItemList. A ideia é sair de uma estrutura fixa e repetitiva para uma estrutura mais inteligente, **baseada em composição e reuso de componentes**.

💡 **Implementando a Refatoração**

```jsx
import React from "react";
import ItemList from "./ItemList";
import { artistArray } from "../assets/database/artists";
import { songsArray } from "../assets/database/songs";

const Main = () => {
  return (
    <div className="main">
      {/* Item List de Artistas */}
      <ItemList
        title="Artistas"
        items={10}
        itemsArray={artistArray}
        path="/artists"
        idPath="/artist"
      />
      {/* Item List de Músicas */}
      <ItemList
        title="Músicas"
        items={20}
        itemsArray={songsArray}
        path="/songs"
        idPath="/song"
      />
    </div>
  );
};

export default Main;
```

🧠 **O que está acontecendo aqui?**

✅ 1. **Uso do Componente ItemList**
Cada <ItemList /> representa uma seção da tela (Artistas, Músicas, etc.).

✅ 2. **Props personalizadas**
- title: Define o nome da seção, ex: "Artistas".
- items: Quantos itens serão exibidos.
- itemsArray: O array de dados que será percorrido.
- path: Caminho do botão “Mostrar tudo”.
- idPath: Caminho individual de cada item (usado para navegação com Link).

🔄 **Benefícios da Refatoração**

Antes	Depois com `ItemList`
Código duplicado	Código enxuto e reutilizável
Difícil de manter	Fácil de adicionar novas seções
Manual, sujeito a erro humano	Automático, controlado por props
Estrutura rígida	Estrutura dinâmica e flexível

💉 **Exemplo: Adicionando outra lista**
Quer adicionar uma lista de álbuns? Basta:

```
<ItemList
  title="Álbuns"
  items={8}
  itemsArray={albumsArray}
  path="/albums"
  idPath="/album"
/>
```

Você aprendeu como refatorar o componente Main utilizando o ItemList para **centralizar e simplificar a renderização de listas**, aproveitando os princípios de **componentização, reuso e escalabilidade** do React.

Aula 2 – Refatorando o Componente Header com React Router

Aprender como substituir as tags <a> tradicionais pelas Link do **React Router**, garantindo uma navegação **mais rápida e sem recarregar a página**.

🧩 Antes da Refatoração
O componente Header utilizava a tag <a> para redirecionamento:

```
<a className="header__link" href="/">
  <h1>Spotify</h1>
</a>
```

❌ Problemas com <a href="/">
- **Recarrega a página** ao clicar.
- Perde o estado atual da aplicação (ex: música tocando, item selecionado).
- Não é a forma recomendada em aplicações React com react-router-dom.

✅ Solução: Usar <Link> do React Router
Ao usar o componente Link, a navegação é feita **internamente**, sem recarregar a página.

Aula 2 – Refatorando o Componente Header com React Router

🧠 Explicação dos Conceitos

Conceito	Explicação
`Link`	Componente do `react-router-dom` para criar links internos.
`to="/"`	Caminho da rota (no caso, a página inicial).
`className`	Permite aplicar estilos personalizados (como no CSS tradicional).
`img dentro de Link`	Torna a imagem um botão clicável que navega.

💡 Código Refatorado

```jsx
import React from "react";
import logoSpotify from "../assets/logo/spotify-logo.png";
import { Link } from "react-router-dom";

const Header = () => {
  return (
    <div className="header">
      {/* Logo clicável redirecionando para a Home */}
      <Link to="/">
        <img src={logoSpotify} alt="Logo do Spotify" />
      </Link>

      {/* Título com link para a Home */}
      <Link to="/" className="header__link">
        <h1>Spotify</h1>
      </Link>
    </div>
  );
};

export default Header;
```

📌 **Comparativo Visual**

Antes (`<a href="/">`)	Depois (`<Link to="/">`)
Recarrega a página	Navegação suave, sem recarregar
Quebra estado da aplicação	Mantém tudo funcionando
Mais lento	Mais rápido e moderno

🧪 **Prática recomendada**

Você pode usar <Link> sempre que quiser navegar entre **rotas internas** da sua aplicação. Exemplo:

```
<Link to="/profile">Perfil</Link>
<Link to="/artists">Artistas</Link>
<Link to="/songs">Músicas</Link>
```

Agora você sabe como deixar o seu Header mais performático e adaptado para aplicações React, utilizando a **navegação via React Router** com o componente Link.

📁 Por que criar a pasta pages/?

Em projetos React, é comum separar os arquivos por **função** ou **propósito**.
A pasta pages/ (ou "páginas") serve para organizar os **componentes que representam as telas completas do aplicativo**, como:

- Página de início (Home)
- Página de um artista específico (Artist)
- Lista de artistas (Artists)
- Página de uma música (Song)
- Lista de músicas (Songs)

🧠 Benefícios de usar a pasta pages/:

Benefício	Explicação
Organização	Mantém o projeto limpo e dividido por responsabilidade.
Escalabilidade	Facilita a adição de novas páginas no futuro.
Reutilização	Deixa claro quais são as páginas principais e quais são componentes reutilizáveis.
Facilidade de navegação	Você sabe onde estão as rotas da aplicação.

✨ **Estrutura inicial da pasta pages/**
Agora criamos cinco arquivos simples dentro de pages/. Cada um deles representa uma rota da aplicação:

🏠 **Home.jsx**
A Home é a página principal da aplicação e está chamando o componente Main, que exibe listas de artistas e músicas.

```jsx
import React from "react";
import Main from "../components/Main";

const Home = () => {
  return <Main />;
};

export default Home;
```

🧑‍🎤 **Artist.jsx**
Esta será a página para exibir um artista específico.

```jsx
import React from "react";

const Artist = () => {
  return <div>Artist</div>;
};

export default Artist;
```

Aula 2 – Criando a pasta pages e suas páginas

Artists.jsx
Página para listar todos os artistas disponíveis.

```jsx
import React from "react";

const Artists = () => {
  return <div>Artists</div>;
};

export default Artists;
```

Song.jsx
Página individual para uma música específica.

```jsx
import React from "react";

const Song = () => {
  return <div>Song</div>;
};

export default Song;
```

Songs.jsx
Página que vai listar todas as músicas.

```jsx
import React from "react";

const Songs = () => {
  return <div>Songs</div>;
};

export default Songs;
```

Aula 2 – Refatorando o App para usar rotas com React Router

A criação da pasta pages/ é um **passo fundamental para organizar sua aplicação**, principalmente quando ela começa a crescer. Essas páginas serão usadas para configurar o sistema de rotas no React Router — que será o próximo passo!

📌 **Situação antes da refatoração**
Antes, nosso App.jsx estava assim:

Nesse caso, nossa aplicação tinha apenas uma "página principal" (Main) e tudo era carregado de uma vez só, sem possibilidade de navegação entre páginas.

```jsx
import React from "react";
import Header from "./components/Header";
import Main from "./components/Main";

const App = () => {
  return (
    <>
      <Header />
      <Main />
    </>
  );
};

export default App;
```

Aula 2 – Refatorando o App para usar rotas com React Router

🚀 **Por que refatorar?**
Estamos crescendo o projeto e começamos a criar **páginas separadas dentro da pasta pages/**.
Com isso, **precisamos de um sistema de navegação entre essas páginas**, e é aí que entra o react-router-dom.

✨ O que estamos adicionando?
🛠️ Importações necessárias:

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
```

- **BrowserRouter**: Componente que **envolve toda a aplicação** e permite o uso das rotas.
- **Routes**: Componente que **organiza todos os caminhos possíveis da aplicação**.
- **Route**: Cada rota representa uma **página acessível** com uma URL.

🔄 **Novo código do App.jsx (refatorado)**

```jsx
import React from "react";
import Header from "./components/Header";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Home from "./pages/Home";
import Artists from "./pages/Artists";
import Artist from "./pages/Artist";
import Songs from "./pages/Songs";
import Song from "./pages/Song";
```

```jsx
const App = () => {
  return (
    <BrowserRouter>
      <Header />

      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/artists" element={<Artists />} />
        <Route path="/artist/:id" element={<Artist />} />
        <Route path="/songs" element={<Songs />} />
        <Route path="/song/:id" element={<Song />} />
      </Routes>
    </BrowserRouter>
  );
};

export default App;
```

🧠 **Explicando as rotas:**

Caminho (path)	Elemento renderizado	Explicação
`/`	`<Home />`	Página inicial
`/artists`	`<Artists />`	Lista de todos os artistas
`/artist/:id`	`<Artist />`	Página de um artista específico (`:id`)
`/songs`	`<Songs />`	Lista de músicas
`/song/:id`	`<Song />`	Página de uma música específica (`:id`)

🔎 :id representa uma **rota dinâmica**, ou seja, o React Router vai pegar o valor do id na URL e permitir que você use esse valor dentro da página (como useParams()).

✅ Resultado final

- Agora temos uma **estrutura de rotas real**, como nos sites profissionais.
- Conseguimos acessar diferentes **páginas** da aplicação com URLs diferentes.
- Podemos navegar entre elas com <Link to="/rota"> sem recarregar a página.