Aula 10

1. Implementação do componente de filtro
2. Testando a aplicação

Introdução

Nesta aula, iremos adicionar a funcionalidade de filtrar os dados do gráfico por instituição bancária. Isso permitirá ao usuário selecionar uma instituição específica e visualizar apenas os dados relacionados a ela. Ao clicar em um botão correspondente a uma instituição, o gráfico será atualizado dinamicamente.

Para implementar essa funcionalidade, precisamos:

- Extrair o nome da instituição bancária a partir da propriedade name em cada registro.
- Criar botões ou um componente de seleção para permitir que o usuário escolha a instituição.
- Filtrar os dados com base na seleção do usuário.
- Atualizar o gráfico quando a seleção mudar.

1. Implementação do componente de filtro

Atualizar o conjunto de dados

Como já temos o nome da instituição bancária na propriedade name, não precisamos modificar a função `fetchData`. Porém, precisamos garantir que possamos extrair o nome do banco a partir dessa propriedade.

Assumindo que o formato do campo name é algo como "BBConsulta", podemos extrair o nome do banco da seguinte forma:

- Se o nome seguir o padrão `"BancoDoBrasilConsulta"`, poderíamos separar o nome do banco do restante da string.

Para simplificar, vamos assumir que o nome da instituição é a parte inicial da string name, antes da palavra `"Consulta"`.

Exemplo:

- Se name é "BBConsulta", o banco é "BB".
- Se name é "CaixaConsulta", o banco é "Caixa".

Vamos ajustar nosso código para extrair o nome do banco dessa maneira.

Atualizar o componente `Home`

O código completo com as alterações fica assim:

```
"use client";
import { useEffect, useState } from "react";
```

```
import { fetchData } from "../data/api";
import {
  BarChart,
  Bar,
  XAxis,
  YAxis,
  CartesianGrid,
  Tooltip,
  Legend,
  ResponsiveContainer,
  Cell,
} from "recharts";

interface DataPoint {
  createdAt: string;
  status: string;
  delay: number;
  error: string | null;
  name: string;
  bankName?: string; // Nova propriedade para armazenar o nome do banco
  aboveAverage?: boolean;
}

export default function Home() {
  const [data, setData] = useState<DataPoint[]>([]);
  const [filteredData, setFilteredData] = useState<DataPoint[]>([]);
  const [selectedBank, setSelectedBank] = useState<string | null>(null);
  const [banks, setBanks] = useState<string[]>([]);

  useEffect(() => {
    const getData = async () => {
      const result = await fetchData();

      const formattedData = result.map((item) => {
        // Extrair o nome do banco da propriedade 'name'
        // Supondo que o nome do banco está antes da palavra 'Consulta'
        const bankName = item.name.replace("Consulta", "");

        return {
          ...item,
          delay: parseInt(item.delay.replace("ms", "")),
          bankName,
        };
      });

      // Obter lista única de bancos
```

```
    const uniqueBanks = Array.from(new Set(formattedData.map((item) => item.bankName)));
    setBanks(uniqueBanks);

    setData(formattedData);
    setFilteredData(formattedData);
  };

  getData();
}, []);

useEffect(() => {
  // Filtrar dados com base no banco selecionado
  const filtered = selectedBank
    ? data.filter((item) => item.bankName === selectedBank)
    : data;

  if (filtered.length > 0) {
    const totalDelay = filtered.reduce((sum, item) => sum + item.delay, 0);
    const averageDelay = totalDelay / filtered.length;
    const threshold = averageDelay * 1.5;

    const dataWithFlags = filtered.map((item) => ({
      ...item,
      aboveAverage: item.delay > threshold,
    }));

    setFilteredData(dataWithFlags);
  } else {
    setFilteredData([]);
  }
}, [selectedBank, data]);

return (
  <main className="p-6">
    <h1 className="text-2xl font-bold mb-4">Gráfico de Desempenho</h1>

    <div className="mb-4">
      <button
        onClick={() => setSelectedBank(null)}
        className={`px-4 py-2 mr-2 ${
          selectedBank === null ? "bg-blue-500 text-white" : "bg-gray-300"
        }`}
      >
        Todos
      </button>
      {banks.map((bank) => (
```

```
            <button
              key={bank}
              onClick={() => setSelectedBank(bank)}
              className={`px-4 py-2 mr-2 ${
                selectedBank === bank ? "bg-blue-500 text-white" : "bg-gray-300"
              }`}
            >
              {bank}
            </button>
          ))}
        </div>

        <ResponsiveContainer width="100%" height={400}>
          <BarChart data={filteredData}>
            <CartesianGrid strokeDasharray="3 3" />
            <XAxis
              dataKey="createdAt"
              angle={-45}
              textAnchor="end"
              interval={0}
              height={120}
            />
            <YAxis />
            <Tooltip />
            <Legend
              layout="horizontal"
              verticalAlign="top"
              align="center"
              iconType="rect"
              iconSize={12}
            />
            <Bar dataKey="delay" name="Delay (ms)" fill="#8884d8">
              {filteredData.map((entry, index) => (
                <Cell
                  key={`cell-${index}`}
                  fill={entry.aboveAverage ? "red" : "#8884d8"}
                />
              ))}
            </Bar>
          </BarChart>
        </ResponsiveContainer>
      </main>
    );
  }
```

Explicação das Alterações

Extração do Nome do Banco: Dentro do `map` em `formattedData`, extraímos o nome do banco a partir da propriedade `name` usando `item.name.replace("Consulta", "")` e armazenamos o resultado na nova propriedade bankName.

Atualização da Interface `DataPoint`: Adicionamos a propriedade opcional `bankName?: string;`.

Obtenção da Lista de Bancos: Usamos `formattedData` para extrair uma lista única de bancos a partir de `bankName`.

Filtragem dos Dados: No `useEffect` que observa `selectedBank` e `data`, filtramos os dados com base em `bankName`.

Testando a Aplicação

Ao executar a aplicação você deverá ver:

- Um conjunto de botões no topo da página, um para cada instituição bancária extraída da propriedade `name`.
- Ao clicar em um dos botões, o gráfico será atualizado para mostrar apenas os dados relacionados à instituição selecionada.
- As barras com delays muito acima da média (calculada com base nos dados filtrados) serão destacadas em vermelho.