Aula 09

1. Criação do projeto Next.js para gráficos
2. Instalação e configuração do Recharts
3. Obtenção de dados do backend via fetch
4. Renderização do gráfico de barras com dados do JSON

1. Criação do projeto Next.js para gráficos

Primeiro, crie um novo projeto Next.js executando o seguinte comando no terminal:

```
npx create-next-app@latest frontend-graficos
```

Navegue até o diretório do projeto com o comando `cd frontend-graficos`.

2. Instalação e configuração do Recharts

Instale o Recharts no seu projeto usando o seguinte comando:

```
npm install recharts
```

3. Obtenção de dados do backend via fetch

Vamos simular uma requisição ao backend que retorna um JSON com uma lista de dados. Crie um arquivo `src/data/api.ts` com o seguinte conteúdo para simular a API:

```
export const fetchData = async () => {
    // Simulando uma chamada assíncrona ao backend
    return [
      // Para facilitar, vamos gerar dinamicamente os registros
      ...Array.from({ length: 30 }, (_, index) => {
        const minutes = (index + 6) * 5; // Incremento de 5 minutos
        const date = new Date('2024-05-09T10:00:00');
        date.setMinutes(date.getMinutes() + minutes);

        const formattedDate = date
          .toISOString()
          .replace('T', '-')
          .substring(0, 19);

        const statuses = ['success', 'error'];
        const status = statuses[Math.floor(Math.random() * statuses.length)];

        const delays = [30, 80, 180, 200, 220, 250, 300, 350, 400, 550, 700];
        const delay = `${delays[Math.floor(Math.random() * delays.length)]}ms`;
```

```
        const errors = [null, 'Timeout', 'Connection lost', 'Server error'];

        return {
          createdAt: formattedDate,
          status,
          delay,
          error: status === 'error' ? errors[Math.floor(Math.random() * errors.length)] : nu
          name: 'BBConsulta',
        };
      }),
    ];
  };
```

Explicação do Código

- Array.from: Utilizamos para gerar dinamicamente os registros adicionais necessários para totalizar aproximadamente 30 itens.
- Incremento de Tempo: Incrementamos 5 minutos a cada registro para simular dados ao longo do tempo.
- Valores Aleatórios: Randomizamos os valores de status, delay e error para simular variações nos dados.

4. Renderização do gráfico de barras com dados do JSON

Agora, vamos utilizar o Recharts para renderizar um gráfico de barras com os dados obtidos. Altere o arquivo `src/page.tsx` para incluir o seguinte código:

```
"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; // Alterado para number
  error: string | null;
```

```
  name: string;
  aboveAverage?: boolean; // Nova propriedade
}

export default function Home() {
  const [data, setData] = useState<DataPoint[]>([]);

  useEffect(() => {
    const getData = async () => {
      const result = await fetchData();
      // Convertendo 'delay' de string para número
      const formattedData = result.map((item) => ({
        ...item,
        delay: parseInt(item.delay.replace("ms", "")),
      }));

      // Calculando a média dos delays
      const totalDelay = formattedData.reduce(
        (sum, item) => sum + item.delay,
        0
      );
      const averageDelay = totalDelay / formattedData.length;

      // Definindo um limite para considerar 'muito acima da média' (por exemplo, 20% acima)
      const threshold = averageDelay * 1.5;

      // Marcando os itens que estão acima do limite
      const dataWithFlags = formattedData.map((item) => ({
        ...item,
        aboveAverage: item.delay > threshold,
      }));

      setData(dataWithFlags);
    };

    getData();
  }, []);

  return (
    <main className="p-6">
      <h1 className="text-2xl font-bold mb-4">Gráfico de Desempenho</h1>
      <ResponsiveContainer width="100%" height={400}>
        <BarChart data={data}>
          <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">
            {data.map((entry, index) => (
              <Cell
                key={`cell-${index}`}
                fill={entry.aboveAverage ? "red" : "#8884d8"}
              />
            ))}
          </Bar>
        </BarChart>
      </ResponsiveContainer>
    </main>
  );
}
```