Exemplos de Consumo dos JSONs no NextJS

📦 Estrutura de Dados

Todos os arquivos JSON estão em `/home/ubuntu/dashboard_acidentes_mg/data/`

🔧 Como Consumir no NextJS

1. Copiar arquivos JSON para o projeto NextJS

```
# Copiar para a pasta public do NextJS
cp /home/ubuntu/dashboard_acidentes_mg/data/*.json /seu-projeto-nextjs/public/data/
```

2. Criar Hook Personalizado (hooks/useAcidentesData.ts)

```tsx
import { useState, useEffect } from 'react';

export interface KPI {
  id: string;
  titulo: string;
  valor: number;
  formato: string;
  tendencia: {
    valor: number | null;
    texto: string | null;
    tipo: 'aumento' | 'reducao' | 'estavel';
  };
  descricao: string;
}

export interface KPIsData {
  periodo: {
    mes_atual: string;
    mes_anterior: string;
  };
  kpis: KPI[];
  metadata: {
    ultima_atualizacao: string;
    periodo_dados: string;
  };
}

export function useKPIs() {
  const [data, setData] = useState<KPIsData | null>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<Error | null>(null);

  useEffect(() => {
    fetch('/data/kpis.json')
      .then(res => res.json())
      .then(setData)
      .catch(setError)
      .finally(() => setLoading(false));
  }, []);

  return { data, loading, error };
}

// Similar para outros dados
export function useEvolucaoMensal() { /* ... */ }
export function useCausas() { /* ... */ }
export function useDistribuicoes() { /* ... */ }
export function useRankings() { /* ... */ }
export function useAreasCriticas() { /* ... */ }
```

3. Componente de KPI (components/KPICard.tsx)

```tsx
import React from 'react';
import { KPI } from '@/hooks/useAcidentesData';

interface KPICardProps {
  kpi: KPI;
}

export function KPICard({ kpi }: KPICardProps) {
  const formatValue = (valor: number, formato: string) => {
    switch (formato) {
      case 'numero':
        return valor.toLocaleString('pt-BR');
      case 'percentual':
        return `${valor}%`;
      case 'decimal':
        return valor.toFixed(2);
      default:
        return valor;
    }
  };

  const getTrendIcon = (tipo: string) => {
    switch (tipo) {
      case 'aumento':
        return '↑';
      case 'reducao':
        return '↓';
      default:
        return '→';
    }
  };

  const getTrendColor = (tipo: string) => {
    // Para KPIs de acidentes/mortos, aumento é ruim (vermelho)
    // Para outros contextos, ajustar conforme necessário
    switch (tipo) {
      case 'aumento':
        return 'text-red-500';
      case 'reducao':
        return 'text-green-500';
      default:
        return 'text-gray-500';
    }
  };

  return (
    <div className="bg-white rounded-lg shadow-md p-6">
      <h3 className="text-sm font-medium text-gray-500 mb-2">
        {kpi.titulo}
      </h3>
      <div className="text-3xl font-bold text-gray-900 mb-2">
        {formatValue(kpi.valor, kpi.formato)}
      </div>
      {kpi.tendencia.texto && (
        <div className={`text-sm ${getTrendColor(kpi.tendencia.tipo)}`}>
          <span className="mr-1">{getTrendIcon(kpi.tendencia.tipo)}</span>
          <span>{kpi.tendencia.texto}</span>
          <span className="text-gray-500 ml-1">{kpi.descricao}</span>
        </div>
      )}
    </div>
```

```
  );
}
```

4. Página de Dashboard (app/dashboard/page.tsx)

```
'use client';

import { useKPIs, useEvolucaoMensal } from '@/hooks/useAcidentesData';
import { KPICard } from '@/components/KPICard';

export default function DashboardPage() {
  const { data: kpisData, loading: kpisLoading } = useKPIs();
  const { data: evolucaoData, loading: evolucaoLoading } = useEvolucaoMensal();

  if (kpisLoading) {
    return <div>Carregando KPIs...</div>;
  }

  return (
    <div className="container mx-auto px-4 py-8">
      <h1 className="text-3xl font-bold mb-8">
        Dashboard de Acidentes - Minas Gerais
      </h1>

      {/* KPIs Grid */}
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-5 gap-6 mb-8">
        {kpisData?.kpis.map((kpi) => (
          <KPICard key={kpi.id} kpi={kpi} />
        ))}
      </div>

      {/* Evolução Mensal Chart */}
      {evolucaoData && (
        <div className="bg-white rounded-lg shadow-md p-6">
          <h2 className="text-xl font-semibold mb-4">Evolução Mensal</h2>
          {/* Adicionar gráfico aqui usando Recharts, Chart.js, etc */}
        </div>
      )}
    </div>
  );
}
```

5. Gráfico de Evolução Mensal (usando Recharts)

```tsx
import {
  LineChart,
  Line,
  XAxis,
  YAxis,
  CartesianGrid,
  Tooltip,
  Legend,
  ResponsiveContainer
} from 'recharts';

interface EvolucaoChartProps {
  data: Array<{
    mes: string;
    total_acidentes: number;
    total_mortos: number;
    total_feridos: number;
  }>;
}

export function EvolucaoChart({ data }: EvolucaoChartProps) {
  return (
    <ResponsiveContainer width="100%" height={400}>
      <LineChart data={data}>
        <CartesianGrid strokeDasharray="3 3" />
        <XAxis
          dataKey="mes"
          tick={{ fontSize: 12 }}
        />
        <YAxis />
        <Tooltip />
        <Legend />
        <Line
          type="monotone"
          dataKey="total_acidentes"
          stroke="#8884d8"
          name="Acidentes"
          strokeWidth={2}
        />
        <Line
          type="monotone"
          dataKey="total_mortos"
          stroke="#ff7c7c"
          name="Mortos"
          strokeWidth={2}
        />
      </LineChart>
    </ResponsiveContainer>
  );
}
```

6. Componente de Causas (Gráfico de Pizza/Donut)

```tsx
import { PieChart, Pie, Cell, ResponsiveContainer, Legend, Tooltip } from 'recharts';

interface CausasChartProps {
  data: Array<{
    causa: string;
    total_acidentes: number;
    percentual: number;
  }>;
}

const COLORS = [
  '#0088FE', '#00C49F', '#FFBB28', '#FF8042', '#8884D8',
  '#82CA9D', '#FFC658', '#8DD1E1', '#D084D0', '#A4DE6C'
];

export function CausasChart({ data }: CausasChartProps) {
  return (
    <ResponsiveContainer width="100%" height={400}>
      <PieChart>
        <Pie
          data={data}
          cx="50%"
          cy="50%"
          labelLine={false}
          label={({ causa, percentual }) => `${causa}: ${percentual}%`}
          outerRadius={120}
          fill="#8884d8"
          dataKey="total_acidentes"
        >
          {data.map((entry, index) => (
            <Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
          ))}
        </Pie>
        <Tooltip />
        <Legend />
      </PieChart>
    </ResponsiveContainer>
  );
}
```

7. Tabela de Rankings

```
interface RankingTableProps {
  data: Array<{
    posicao: number;
    municipio?: string;
    br?: string;
    total_acidentes: number;
    total_mortos: number;
    total_feridos: number;
  }>;
  tipo: 'municipios' | 'brs';
}

export function RankingTable({ data, tipo }: RankingTableProps) {
  return (
    <div className="overflow-x-auto">
      <table className="min-w-full divide-y divide-gray-200">
        <thead className="bg-gray-50">
          <tr>
            <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 up-
percase">
              Posição
            </th>
            <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 up-
percase">
              {tipo === 'municipios' ? 'Município' : 'BR'}
            </th>
            <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 up-
percase">
              Acidentes
            </th>
            <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 up-
percase">
              Mortos
            </th>
            <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 up-
percase">
              Feridos
            </th>
          </tr>
        </thead>
        <tbody className="bg-white divide-y divide-gray-200">
          {data.map((item) => (
            <tr key={item.posicao}>
              <td className="px-6 py-4 whitespace-nowrap text-sm font-medium text-
gray-900">
                {item.posicao}
              </td>
              <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-900">
                {tipo === 'municipios' ? item.municipio : `BR-${item.br}`}
              </td>
              <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-500">
                {item.total_acidentes.toLocaleString('pt-BR')}
              </td>
              <td className="px-6 py-4 whitespace-nowrap text-sm text-red-500">
                {item.total_mortos.toLocaleString('pt-BR')}
              </td>
              <td className="px-6 py-4 whitespace-nowrap text-sm text-orange-500">
                {item.total_feridos.toLocaleString('pt-BR')}
              </td>
            </tr>
          ))}
        </tbody>
```

```
      </table>
    </div>
  );
}
```

📊 Bibliotecas Recomendadas

Para Gráficos

```
npm install recharts
# ou
npm install chart.js react-chartjs-2
# ou
npm install @nivo/core @nivo/line @nivo/pie
```

Para Tabelas

```
npm install @tanstack/react-table
```

Para Formatação de Dados

```
npm install date-fns
```

🔄 Atualização Automática de Dados

Para atualizar os dados periodicamente:

```
// hooks/useAutoRefresh.ts
import { useEffect } from 'react';

export function useAutoRefresh(callback: () => void, interval: number = 60000) {
  useEffect(() => {
    const timer = setInterval(callback, interval);
    return () => clearInterval(timer);
  }, [callback, interval]);
}

// Uso
function Dashboard() {
  const refetch = () => {
    // Recarregar dados
  };

  useAutoRefresh(refetch, 5 * 60 * 1000); // A cada 5 minutos
}
```

🎨 Dicas de UI/UX

1. **Cores para tendências**:
   - 🔴 Vermelho para aumento de acidentes/mortos (ruim)
   - 🟢 Verde para redução de acidentes/mortos (bom)
   - ⚪ Cinza para estável

2. **Indicadores visuais**:
   - Use ícones para facilitar a compreensão (↑, ↓, →)
   - Adicione tooltips explicativos
   - Use badges para destacar áreas críticas

3. **Responsividade**:
   - Grid adaptável para diferentes tamanhos de tela
   - Tabelas com scroll horizontal em mobile
   - Gráficos responsivos

4. **Performance**:
   - Implementar loading states
   - Cache de dados com SWR ou React Query
   - Lazy loading para componentes pesados