Guia de Migração - Lazy Loading (FASE 2.5)

📋 Objetivo

Migrar imports estáticos de bibliotecas pesadas para lazy loading, reduzindo o bundle inicial de **11.5MB** para **3-4MB**.

🎯 Bibliotecas Alvo

Críticas (> 1MB)

- ✅ **jsPDF / html2pdf** (1.04MB) - Geração de PDF
- ✅ **Mapbox GL** (1.65MB) - Mapas interativos
- ✅ **Recharts** (268KB) - Gráficos
- ✅ **Chart.js** (166KB) - Gráficos alternativos
- ✅ **MQTT** (357KB) - Conectividade IoT
- ✅ **TensorFlow** (401KB) - AI/ML
- ✅ **TipTap Editor** (164KB) - Editor de texto rico
- ✅ **Three.js** (74KB) - 3D/XR
- ✅ **Framer Motion** (110KB) - Animações

📦 Componentes Lazy Disponíveis

1. LazyChart (Recharts - 268KB)

❌ **ANTES:**

```
import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer }
from "recharts";

function MyComponent() {
  return (
    <ResponsiveContainer width="100%" height={300}>
      <LineChart data={data}>
        <CartesianGrid strokeDasharray="3 3" />
        <XAxis dataKey="name" />
        <YAxis />
        <Tooltip />
        <Line type="monotone" dataKey="value" stroke="#8884d8" />
      </LineChart>
    </ResponsiveContainer>
  );
}
```

✅ **DEPOIS:**

```tsx
import { LazyChart } from "@/components/lazy";
import { Suspense } from "react";

function MyComponent() {
  return (
    <LazyChart height={300}>
      {/* Seu código de chart aqui - será lazy loaded */}
      <ResponsiveContainer width="100%" height={300}>
        <LineChart data={data}>
          <CartesianGrid strokeDasharray="3 3" />
          <XAxis dataKey="name" />
          <YAxis />
          <Tooltip />
          <Line type="monotone" dataKey="value" stroke="#8884d8" />
        </LineChart>
      </ResponsiveContainer>
    </LazyChart>
  );
}
```

✨ **OU use o dynamic import:**

```tsx
import { loadRecharts } from "@/lib/lazy-loaders";
import { useEffect, useState } from "react";

function MyComponent() {
  const [Recharts, setRecharts] = useState<any>(null);

  useEffect(() => {
    loadRecharts().then(setRecharts);
  }, []);

  if (!Recharts) return <div>Carregando gráfico...</div>;

  const { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip,
ResponsiveContainer } = Recharts;

  return (
    <ResponsiveContainer width="100%" height={300}>
      <LineChart data={data}>
        {/* ... */}
      </LineChart>
    </ResponsiveContainer>
  );
}
```

2. LazyPDFGenerator (jsPDF - 1.04MB)

❌ **ANTES:**

```
import jsPDF from "jspdf";
import { Button } from "@/components/ui/button";

function MyComponent() {
  const generatePDF = () => {
    const doc = new jsPDF();
    doc.text("Hello world!", 10, 10);
    doc.save("documento.pdf");
  };

  return <Button onClick={generatePDF}>Exportar PDF</Button>;
}
```

✅ **DEPOIS - Opção 1 (Componente):**

```
import { LazyPDFGenerator } from "@/components/lazy";

function MyComponent() {
  const generatePDF = async () => {
    const jsPDF = await import("jspdf").then(m => m.default);
    const doc = new jsPDF();
    doc.text("Hello world!", 10, 10);
    doc.save("documento.pdf");
  };

  return (
    <LazyPDFGenerator
      fileName="documento.pdf"
      buttonText="Exportar PDF"
      onGenerate={generatePDF}
    />
  );
}
```

✅ **DEPOIS - Opção 2 (Dynamic Import):**

```tsx
import { loadJsPDF } from "@/lib/lazy-loaders";
import { Button } from "@/components/ui/button";
import { useState } from "react";

function MyComponent() {
  const [loading, setLoading] = useState(false);

  const generatePDF = async () => {
    setLoading(true);
    try {
      const jsPDF = await loadJsPDF();
      const doc = new jsPDF();
      doc.text("Hello world!", 10, 10);
      doc.save("documento.pdf");
    } finally {
      setLoading(false);
    }
  };

  return (
    <Button onClick={generatePDF} disabled={loading}>
      {loading ? "Gerando..." : "Exportar PDF"}
    </Button>
  );
}
```

3. LazyMap (Mapbox GL - 1.65MB)

❌ **ANTES:**

```tsx
import mapboxgl from "mapbox-gl";
import "mapbox-gl/dist/mapbox-gl.css";
import { useEffect, useRef } from "react";

function MyComponent() {
  const mapContainer = useRef(null);

  useEffect(() => {
    if (!mapContainer.current) return;

    const map = new mapboxgl.Map({
      container: mapContainer.current,
      style: 'mapbox://styles/mapbox/streets-v12',
      center: [-46.6333, -23.5505],
      zoom: 12
    });

    return () => map.remove();
  }, []);

  return <div ref={mapContainer} style={{ height: "400px" }} />;
}
```

✅ **DEPOIS:**

```
import { LazyMap } from "@/components/lazy";

function MyComponent() {
  return (
    <LazyMap
      center={[-46.6333, -23.5505]}
      zoom={12}
      height="400px"
      onMapLoad={(map) => {
        // Configurações adicionais do mapa
        console.log("Mapa carregado!", map);
      }}
    />
  );
}
```

4. MQTT (357KB)

❌ **ANTES:**

```
import mqtt from "mqtt";

function MyComponent() {
  const connectMQTT = () => {
    const client = mqtt.connect('mqtt://broker.example.com');
    // ...
  };
  // ...
}
```

✅ **DEPOIS:**

```
import { loadMQTT } from "@/lib/lazy-loaders";

function MyComponent() {
  const connectMQTT = async () => {
    const mqtt = await loadMQTT();
    const client = mqtt.connect('mqtt://broker.example.com');
    // ...
  };
  // ...
}
```

5. TensorFlow (401KB)

❌ **ANTES:**

```
import * as tf from "@tensorflow/tfjs";
import * as cocoSsd from "@tensorflow-models/coco-ssd";

async function detectObjects() {
  await tf.ready();
  const model = await cocoSsd.load();
  // ...
}
```

✅ **DEPOIS:**

```
import { loadTensorFlow, loadCocoSSD } from "@/lib/lazy-loaders";

async function detectObjects() {
  const tf = await loadTensorFlow();
  await tf.ready();
  const cocoSsd = await loadCocoSSD();
  const model = await cocoSsd.load();
  // ...
}
```

6. TipTap Editor (164KB)

❌ **ANTES:**

```
import { useEditor } from "@tiptap/react";
import StarterKit from "@tiptap/starter-kit";

function MyEditor() {
  const editor = useEditor({
    extensions: [StarterKit],
    content: '<p>Hello World!</p>',
  });
  // ...
}
```

✅ **DEPOIS:**

```
import { loadTipTap } from "@/lib/lazy-loaders";
import { useEffect, useState } from "react";

function MyEditor() {
  const [editor, setEditor] = useState(null);

  useEffect(() => {
    loadTipTap().then(({ useEditor, StarterKit }) => {
      const editorInstance = useEditor({
        extensions: [StarterKit.default],
        content: '<p>Hello World!</p>',
      });
      setEditor(editorInstance);
    });
  }, []);

  if (!editor) return <div>Carregando editor...</div>;
  // ...
}
```

7. Framer Motion (110KB)

❌ **ANTES:**

```tsx
import { motion } from "framer-motion";

function MyComponent() {
  return (
    <motion.div
      initial={{ opacity: 0 }}
      animate={{ opacity: 1 }}
    >
      Conteúdo animado
    </motion.div>
  );
}
```

✅ **DEPOIS:**

```tsx
import { loadFramerMotion } from "@/lib/lazy-loaders";
import { useEffect, useState } from "react";

function MyComponent() {
  const [motion, setMotion] = useState<any>(null);

  useEffect(() => {
    loadFramerMotion().then(setMotion);
  }, []);

  if (!motion) return <div>Conteúdo animado</div>; // Fallback sem animação

  return (
    <motion.div
      initial={{ opacity: 0 }}
      animate={{ opacity: 1 }}
    >
      Conteúdo animado
    </motion.div>
  );
}
```

🔄 Preload Inteligente

O sistema automaticamente faz preload de módulos baseado na rota:

```
// Preload automático por rota
/admin/*            → jsPDF
/dashboard/*        → Recharts
/fleet/*            → Mapbox
/ai/*               → TensorFlow
```

Preload Manual

```
import { useManualPreload } from "@/hooks/use-lazy-preload";

function MyComponent() {
  const { preloadCharts, preloadPDF, preloadMap } = useManualPreload();

  const handleNavigateToAdmin = () => {
    preloadPDF(); // Começa a carregar PDF antes de navegar
    navigate("/admin");
  };

  return <Button onClick={handleNavigateToAdmin}>Ir para Admin</Button>;
}
```

📊 Impacto Esperado

Bundle Antes (11.5 MB)

```
pages-main:      3.06 MB
vendors:         2.69 MB
modules-misc:    2.34 MB
map:             1.65 MB
pages-admin:     1.10 MB
pdf-gen:         1.04 MB
```

Bundle Depois (~3-4 MB)

```
core-react:      ~300 KB
core-router:     ~200 KB
pages-core:      ~500 KB
ui-components:   ~800 KB
app-logic:       ~1.2 MB
TOTAL INICIAL:   ~3-4 MB ✅

(Outros chunks carregados sob demanda)
```

✅ Checklist de Migração

Para cada arquivo que usa bibliotecas pesadas:

- [ ] Identificar imports estáticos de bibliotecas > 100KB
- [ ] Substituir por dynamic import usando `loadXXX()` de `@/lib/lazy-loaders`
- [ ] Adicionar loading state apropriado
- [ ] Testar funcionalidade em dev
- [ ] Verificar bundle size após build
- [ ] Validar em produção

🚀 Próximos Passos

1. **Migrar páginas críticas primeiro:**
   - `src/pages/admin/*` (usa jsPDF)

- `src/pages/ai/*` (usa Recharts e TensorFlow)
- `src/pages/emerging/*` (usa Recharts)
- Command Centers (usam jsPDF e Recharts)

2. **Migrar módulos:**
   - Fleet management (Mapbox)
   - Compliance (jsPDF)
   - Analytics (Recharts)

3. **Validar bundle size:**
   `bash`
   `npm run build`
   `# Verificar dist/assets/*`

📝 Notas Importantes

- ⚠️ **Não** lazy load providers de contexto (AuthProvider, etc.)
- ⚠️ **Não** lazy load hooks críticos (React Query, etc.)
- ✅ **Sempre** lazy load bibliotecas de visualização/processamento
- ✅ **Sempre** adicionar loading states apropriados
- ✅ **Sempre** testar funcionalidade após migração

🆘 Troubleshooting

Erro: “Cannot read properties of undefined”

```
// ❌ ERRADO - Tenta usar antes de carregar
const Chart = await loadRecharts();
<Chart.LineChart /> // undefined!

// ✅ CORRETO - Desestrutura após carregar
const recharts = await loadRecharts();
const { LineChart } = recharts;
<LineChart />
```

Loading state não aparece

```
// ✅ Use Suspense ou estado local
<Suspense fallback={<Skeleton />}>
  <LazyComponent />
</Suspense>
```

Bundle ainda grande

```
# Verifique se os imports foram realmente removidos
grep -r "import.*from.*jspdf" src/
grep -r "import.*from.*recharts" src/
```