🚀 Guía Completa: Solución a Problemas de Memoria en INMOVA

📊 Análisis del Problema

Tu proyecto INMOVA es **excepcionalmente grande**:
- 269+ archivos principales
- 88 módulos profesionales
- 7 verticales de negocio
- Múltiples integraciones (Stripe, Zucchetti, Bankinter, etc.)
- Dependencias pesadas (recharts, prisma, múltiples UI libraries)

Tamaño estimado del proyecto: ~5,200+ líneas de código solo en servicios

🎯 Soluciones por Prioridad

✅ NIVEL 1: Soluciones Inmediatas (Sin costo adicional)

1.1 Configuración de Memoria Node.js

Ya hemos creado el archivo `.npmrc` con:

```
node-options=--max_old_space_size=4096
```

Esto aumenta la memoria disponible para Node.js a 4GB durante el build.

1.2 Variables de Entorno para Build

Agrega a tu `.env` o configuración de Vercel:

```
# Build optimizations
NODE_OPTIONS="--max-old-space-size=4096"
NEXT_TELEMETRY_DISABLED=1

# Para análisis de bundle (solo desarrollo)
# ANALYZE=true
```

1.3 Optimización de Imports

El proyecto YA tiene lazy loading en muchos componentes pesados:
- ✅ Recharts: `@/components/ui/lazy-charts-extended`
- ✅ Tabs: `@/components/ui/lazy-tabs`
- ✅ Dialogs: `@/components/ui/lazy-dialog`

Recomendación: Continúa este patrón para otros componentes pesados.

1.4 Análisis de Bundle Size

Instala la herramienta de análisis:

```
cd /home/ubuntu/homming_vidaro/nextjs_space
yarn add -D @next/bundle-analyzer
```

Luego en `next.config.js` (requiere edición manual):

```
const withBundleAnalyzer = require('@next/bundle-analyzer')({
  enabled: process.env.ANALYZE === 'true',
})

module.exports = withBundleAnalyzer(nextConfig)
```

Ejecuta:

```
ANALYZE=true yarn build
```

⚡ NIVEL 2: Optimizaciones Next.js Avanzadas

2.1 Actualizar next.config.js

Agrega estas optimizaciones (edición manual requerida):

```
const nextConfig = {
  // ... configuración existente

  // Optimizaciones de memoria
  swcMinify: true,
  compiler: {
    removeConsole: process.env.NODE_ENV === 'production',
  },

  // Experimental: optimizar imports automáticamente
  experimental: {
    optimizePackageImports: ['lucide-react', 'recharts', 'date-fns', '@radix-ui'],
    optimizeCss: true, // Requiere Critters
  },

  // Webpack optimizations
  webpack: (config, { isServer, dev }) => {
    if (!isServer) {
      config.resolve.fallback = {
        fs: false,
        net: false,
        tls: false,
      };
    }

    // Solo en producción
    if (!dev) {
      config.optimization = {
        ...config.optimization,
        moduleIds: 'deterministic',
        splitChunks: {
          chunks: 'all',
          cacheGroups: {
            default: false,
            vendors: false,
            // Chunk grande para vendor
            vendor: {
              name: 'vendor',
              chunks: 'all',
              test: /node_modules/,
              priority: 20,
              maxSize: 244000, // ~244kb
            },
            // Separar recharts (muy pesado)
            recharts: {
              name: 'recharts',
              test: /[\\/]node_modules[\\/](recharts|d3-*)[\\/]/,
              priority: 30,
            },
            // Separar UI components
            ui: {
              name: 'ui',
              test: /[\\/]node_modules[\\/](@radix-ui)[\\/]/,
              priority: 25,
            },
            common: {
              name: 'common',
              minChunks: 2,
              chunks: 'async',
              priority: 10,
              reuseExistingChunk: true,
            },
```

```
          },
        },
      };
    }

    return config;
  },
};
```

2.2 Reducir Dependencias

Paquetes que podrías optimizar:

1. **Recharts** (muy pesado: ~500KB) - Ya lazy-loaded ✅
2. **date-fns**: Usar imports específicos
   ```javascript
   // Mal ❌
   import { format, addDays } from 'date-fns';

// Bien ✅
import format from 'date-fns/format';
import addDays from 'date-fns/addDays';
```

1. **Lucide Icons**: Ya estás usando imports específicos ✅

2.3 Implement Incremental Static Regeneration (ISR)

Para páginas que no cambian frecuentemente:

```
// En páginas estáticas como /landing
export const revalidate = 3600; // Revalidar cada hora
```

💰 NIVEL 3: ¿Es Vercel la Solución?

Comparativa de Hosting

Plataforma	Memoria Build	Memoria Runtime	Precio/mes	Recomendación
Vercel Hobby	1GB	1GB	$0	❌ Insuficiente para tu proyecto
Vercel Pro	3GB	3GB	$20	⚠️ Podría funcionar con optimizaciones
Vercel Enterprise	Custom	Custom	$$$$	✅ Ideal pero costoso
Railway	8GB	8GB	$5-20	✅ Muy buena opción
Render	7GB	7GB	$7-25	✅ Excelente relación precio/valor
AWS Amplify	Custom	Custom	Variable	✅ Escalable pero complejo
VPS (DigitalOcean)	Ilimitado	Custom	$12-48	✅ Máximo control

🎖️ Recomendación: Railway o Render

¿Por qué Railway/Render > Vercel para tu caso?

1. **Más memoria por menos dinero**
   - Railway: 8GB RAM por $10-20/mes
   - Vercel Pro: 3GB RAM por $20/mes

2. **Sin límites artificiales**
   - No hay límite de “Function Execution” time
   - Builds más largos permitidos

3. **Base de datos incluida**
   - Railway y Render ofrecen PostgreSQL managed
   - Vercel requiere servicio externo

4. **Mejor para aplicaciones complejas**
   - Tu proyecto tiene 88 módulos
   - Múltiples integraciones de terceros
   - Base de datos grande (Prisma)

🚢 NIVEL 4: Migración a Railway (Recomendado)

Pasos para migrar de Vercel a Railway:

1. **Crear cuenta en Railway**
   - Ir a https://railway.app
   - Conectar con GitHub

2. **Crear nuevo proyecto**
   `bash`
   `# Railway detectará automáticamente Next.js`

3. **Configurar variables de entorno**
   - Copiar todas las variables de Vercel
   - Especialmente: DATABASE_URL, NEXTAUTH_SECRET, etc.

4. **Configurar build settings**
   `Build Command: yarn build`
   `Start Command: yarn start`

5. **Añadir PostgreSQL (si usas Railway DB)**
   `bash`
   `# Railway provee PostgreSQL con un clic`
   `# Auto-configura DATABASE_URL`

6. **Deploy**
   `bash`
   `git push origin main`
   `# Railway hace auto-deploy`

Configuración óptima Railway:

```
# railway.json (crear en root del proyecto)
{
  "$schema": "https://railway.app/railway.schema.json",
  "build": {
    "builder": "NIXPACKS"
  },
  "deploy": {
    "numReplicas": 1,
    "restartPolicyType": "ON_FAILURE",
    "restartPolicyMaxRetries": 10
  }
}
```

```
# nixpacks.toml (crear en root del proyecto)
[phases.setup]
nixPkgs = ['nodejs-18_x']

[phases.install]
cmds = [
  'yarn install --frozen-lockfile',
  'yarn prisma generate'
]

[phases.build]
cmds = ['NODE_OPTIONS="--max-old-space-size=4096" yarn build']

[start]
cmd = 'yarn start'
```

🔍 NIVEL 5: Optimizaciones Específicas del Proyecto

5.1 Lazy Load Rutas Pesadas

```
// En app/layout.tsx o donde corresponda
import dynamic from 'next/dynamic';

const HeavyDashboard = dynamic(() => import('./dashboard/page'), {
  loading: () => <LoadingState message="Cargando dashboard..." />,
  ssr: false, // Si no necesitas SSR para esta ruta
});
```

5.2 Optimizar Prisma

```
// En lib/db.ts, asegurar singleton
import { PrismaClient } from '@prisma/client';

const globalForPrisma = global as unknown as { prisma: PrismaClient };

export const prisma =
  globalForPrisma.prisma ||
  new PrismaClient({
    log: process.env.NODE_ENV === 'development' ? ['error', 'warn'] : ['error'],
  });

if (process.env.NODE_ENV !== 'production') globalForPrisma.prisma = prisma;
```

5.3 Implementar Service Workers para Cacheo

Ya tienes PWA configurado ✅, pero puedes mejorar:

```
// public/service-worker.js
self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open('inmova-v1').then((cache) => {
      return cache.addAll([
        '/',
        '/offline',
        // Recursos estáticos críticos
        '/favicon.ico',
        '/inmova-logo-icon.jpg',
      ]);
    })
  );
});
```

📈 NIVEL 6: Monitoreo y Debugging

6.1 Herramientas de Monitoreo

```
# Instalar Vercel CLI para análisis local
npm i -g vercel

# Ver logs en tiempo real
vercel logs <deployment-url>

# Ver uso de memoria
vercel inspect <deployment-url>
```

6.2 Análisis de Performance

```
// Agregar en pages/_app.tsx
import { useReportWebVitals } from 'next/web-vitals';

export function reportWebVitals(metric) {
  console.log(metric);

  // Enviar a analytics
  if (metric.label === 'web-vital') {
    // Analytics tracking
  }
}
```

🎯 Resumen Ejecutivo

Para Quedarte en Vercel:

1. ✅ Upgrade a Vercel Pro ($20/mes)
2. ✅ Aplicar todas las optimizaciones del NIVEL 1 y 2
3. ✅ Continuar con lazy loading agresivo
4. ⚠️ Esperar que 3GB sean suficientes

Para Mejor Rendimiento (Recomendado):

1. ✅ Migrar a **Railway** ($10-20/mes)
2. ✅ 8GB RAM garantizados
3. ✅ PostgreSQL incluido

4. ✅ Mejor soporte para aplicaciones enterprise
5. ✅ Sin límites de function execution

Alternativa Intermedia:

1. ✅ Usar **Render** ($7-25/mes)
2. ✅ 7GB RAM
3. ✅ Free tier disponible para testing
4. ✅ Deploy similar a Vercel

🚀 Plan de Acción Recomendado

Fase 1: Optimización Inmediata (Esta semana)

- [x] Crear .npmrc con max_old_space_size
- [ ] Actualizar next.config.js con optimizaciones
- [ ] Instalar @next/bundle-analyzer
- [ ] Analizar bundle size
- [ ] Identificar dependencias pesadas

Fase 2: Optimización de Código (Próxima semana)

- [ ] Más lazy loading en rutas pesadas
- [ ] Optimizar imports de date-fns
- [ ] Implementar ISR donde sea posible
- [ ] Code splitting adicional

Fase 3: Migración (Si fase 1-2 no resuelve)

- [ ] Crear cuenta Railway/Render
- [ ] Configurar proyecto de prueba
- [ ] Migrar base de datos
- [ ] Deploy de prueba
- [ ] Migración completa

💡 Conclusión

Tu proyecto es enterprise-grade y merece infraestructura enterprise. Vercel Hobby es insuficiente.

Opciones viables:
1. 🥇 **Railway** - Mejor relación precio/rendimiento para tu caso
2. 🥈 **Vercel Pro** - Si quieres quedarte en Vercel y estás dispuesto a optimizar mucho
3. 🥉 **Render** - Alternativa económica y robusta

No recomendado:
- ❌ Vercel Hobby - Insuficiente para tu proyecto
- ❌ Netlify - Similares limitaciones que Vercel

📞 Próximos Pasos

¿Quieres que te ayude con:
1. Implementar las optimizaciones de next.config.js?

2. Configurar Railway para migración?
3. Analizar el bundle size en detalle?
4. Otras optimizaciones específicas?

Documentación creada: Diciembre 2025
Proyecto: INMOVA - Sistema Multi-Vertical PropTech
Estado: Proyecto Enterprise con 88 módulos profesionales