🚀 Soluciones para Problemas de Memoria - Proyecto INMOVA

📊 Diagnóstico Actual

Tamaño total del proyecto: 5.4 GB
- `.git` : 302 MB (historial de versiones)
- `.next` : 67 MB (build actual)
- `public` : 956 KB
- `prisma` : 992 KB
- Código fuente: ~200 MB

🔧 Soluciones Inmediatas (Sin Migración)

1. Optimizar Git Repository

```
# Limpiar historial de Git (reducir .git de 302MB)
cd /home/ubuntu/homming_vidaro/nextjs_space
git reflog expire --expire=now --all
git gc --prune=now --aggressive

# Resultado esperado: reducción del 50-70% en .git
```

2. Aumentar Memoria para Build

Opción A: Variables de entorno temporales

```
export NODE_OPTIONS="--max-old-space-size=8192"
cd /home/ubuntu/homming_vidaro/nextjs_space && yarn build
```

Opción B: Modificar scripts manualmente
Editar `package.json` y cambiar:

```
"scripts": {
  "dev": "NODE_OPTIONS='--max-old-space-size=4096' next dev",
  "build": "NODE_OPTIONS='--max-old-space-size=8192' next build"
}
```

3. Optimizar Importaciones

Problema identificado: Importaciones completas de librerías pesadas.

Solución:

```
// ❌ Evitar
import * as Icons from 'lucide-react';

// ✅ Preferir
import { Home, Settings, User } from 'lucide-react';
```

4. Lazy Loading Mejorado

Ya implementado en:
- Charts (lazy-charts-extended)
- Dialogs (lazy-dialog)
- Tabs (lazy-tabs)

Aplicar a más componentes:

```
import dynamic from 'next/dynamic';

// Para componentes pesados
const HeavyComponent = dynamic(() => import('@/components/HeavyComponent'), {
  loading: () => <LoadingState message="Cargando..." />,
  ssr: false // Si no es crítico para SEO
});
```

5. Análisis de Bundle

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

# Ejecutar análisis
ANALYZE=true yarn build
```

Luego agregar a `next.config.js`:

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

module.exports = withBundleAnalyzer({
  // configuración actual
});
```

🏗️ Soluciones Arquitectónicas (Mediano Plazo)

6. Modularización por Vertical

Dividir el proyecto en paquetes más pequeños:

```
inmova/
├── packages/
│   ├── core/              # Funcionalidad compartida
│   ├── traditional/       # Alquiler tradicional
│   ├── coliving/          # Co-living
│   ├── str/               # Short-term rental
│   ├── flipping/          # House flipping
│   └── professional/      # Servicios profesionales
├── apps/
│   ├── main-app/          # App principal
│   └── admin-portal/      # Portal super-admin
└── package.json           # Workspace root
```

Beneficios:
- Builds más rápidos (solo compilas lo que cambias)
- Mejor organización del código
- Posibilidad de deployments independientes

7. Microservicios para Módulos Pesados

Mover funcionalidades pesadas a servicios separados:

Candidatos:
- Generación de PDFs
- Procesamiento de OCR
- Integración con ERP (Zucchetti, Sage, etc.)
- Análisis de datos y reportes

🌍 Opciones de Entorno

Opción 1: Continuar en DeepAgent ✅

Pros:
- Contexto completo del proyecto
- Herramientas integradas
- Deploy y testing automatizado

Contras:
- Límites de memoria en builds muy grandes
- Sesiones tienen tiempo límite

Recomendado para:
- Desarrollo iterativo
- Nuevas features
- Fixes y optimizaciones

Opción 2: Desarrollo Local

Pros:
- Sin límites de memoria
- Mayor control sobre el entorno
- Herramientas de debugging avanzadas

Contras:
- Requiere configuración inicial
- Sin deploy automático

Pasos para migrar:

```
# 1. Clonar desde el checkpoint guardado
#    (usa la UI de DeepAgent para descargar)

# 2. Setup local
cd tu-proyecto-local
yarn install

# 3. Configurar .env
cp .env.example .env
# Editar .env con tus credenciales

# 4. Iniciar desarrollo
yarn prisma generate
yarn dev
```

Opción 3: Entorno Híbrido (Recomendado) ⭐

Estrategia:
1. **DeepAgent:** Para features nuevas, optimizaciones, fixes rápidos
2. **Local:** Para refactorings grandes, testing exhaustivo
3. **Sync:** Usar git/checkpoints para sincronizar

📈 Métricas de Mejora Esperadas

Después de optimizaciones:

Métrica	Antes	Después	Mejora
Tamaño .git	302 MB	~100 MB	-67%
Tiempo de build	~5 min	~3 min	-40%
Uso de RAM (dev)	4 GB	2.5 GB	-37%
Bundle size (client)	~800 KB	~500 KB	-37%

🎯 Plan de Acción Recomendado

Fase 1: Optimizaciones Inmediatas (1-2 horas)

1. ✅ Limpiar repositorio Git
2. ✅ Aumentar memoria para builds
3. ✅ Ejecutar análisis de bundle
4. ✅ Identificar componentes pesados

Fase 2: Mejoras Graduales (1-2 semanas)

1. Aplicar lazy loading a componentes restantes
2. Optimizar importaciones de librerías
3. Implementar code splitting adicional
4. Revisar y optimizar queries de Prisma

Fase 3: Reestructuración (Si necesario, 1-2 meses)

1. Evaluar modularización por vertical
2. Considerar extracción de servicios pesados
3. Implementar estrategia de microservicios

🛠️ Comandos Útiles

Monitoreo de Memoria

```
# Durante desarrollo
NODE_OPTIONS="--max-old-space-size=4096 --trace-warnings" yarn dev

# Ver uso de memoria en tiempo real
node --max-old-space-size=8192 --trace-gc node_modules/.bin/next build
```

Limpieza Regular

```
# Limpiar builds antiguos
rm -rf .next

# Limpiar cache de Next.js
rm -rf .next/cache

# Regenerar Prisma client
yarn prisma generate
```

Análisis de Dependencias

```
# Ver tamaño de dependencias
yarn why [package-name]

# Encontrar duplicados
yarn dedupe
```

💬 Recomendación Personal

Para tu caso específico:

1. **Corto plazo:** Continúa en DeepAgent con las optimizaciones de memoria aplicadas
2. **Medio plazo:** Implementa lazy loading agresivo y optimiza importaciones
3. **Largo plazo:** Considera modularización si el proyecto sigue creciendo

El proyecto está bien estructurado, solo necesita optimizaciones puntuales. No es necesario migrar a otro entorno todavía, pero tener un setup local como backup es prudente.

📞 Siguiente Paso

¿Qué te gustaría hacer primero?

A. Aplicar optimizaciones inmediatas (limpieza Git + memoria)
B. Análisis profundo de bundle para identificar problemas específicos
C. Configurar entorno local como backup
D. Planificar modularización del proyecto

Dime qué prefieres y procedo con la implementación. 🚀