Reporte de Optimización INMOVA - Deployment a inmova.app

Fecha: 6 de Diciembre de 2025

Resumen Ejecutivo

Se ha realizado una revisión exhaustiva del código, base de datos y sistema de la aplicación INMOVA. Este documento detalla los problemas encontrados y las optimizaciones necesarias para un deployment óptimo.

---

1. PROBLEMAS CRÍTICOS DETECTADOS

1.1 Configuración de Next.js

Problema:

- El `next.config.js` es extremadamente básico y carece de optimizaciones críticas para producción
- Faltan configuraciones de seguridad HTTP headers
- No hay optimización de imágenes configurada
- Bundle size no está optimizado

Solución:

Se debe implementar un `next.config.js` optimizado con:
- Compresión de assets
- Headers de seguridad (CSP, HSTS, X-Frame-Options)
- Optimización de imágenes
- Tree-shaking mejorado
- Code splitting adecuado

1.2 Console Statements en Producción

Problema:

- Se detectaron **339 instancias** de console.log/error/warn en el código
- Estos statements deben ser reemplazados por el logger estructurado
- Exponen información sensible en producción

Impacto:

- Riesgo de seguridad: exposición de datos sensibles
- Performance: overhead innecesario en producción
- Logs no estructurados dificultan debugging

Solución:

Reemplazar todos los console statements por el logger de Winston:

```
// Antes
console.log('User data:', userData);

// Después
logger.info('User data processed', { userId: user.id });
```

1.3 Variables de Entorno y Secrets

Problemas Detectados:

1. **Secrets de prueba en .env:**
   - `STRIPE_SECRET_KEY=sk_test_placeholder`
   - `STRIPE_PUBLISHABLE_KEY=pk_test_placeholder`
   - `STRIPE_WEBHOOK_SECRET=whsec_placeholder`

2. **URLs hardcoded:**
   - `NEXTAUTH_URL` debe ser dinámico para inmova.app

3. **Claves de integración vacías:**
   - DocuSign, Redsys, y otras integraciones con placeholders

Solución Inmediata:

```
# Actualizar .env con valores reales de producción
NEXTAUTH_URL=https://www.inmova.app
STRIPE_SECRET_KEY=<clave_real_produccion>
STRIPE_PUBLISHABLE_KEY=<clave_real_produccion>
STRIPE_WEBHOOK_SECRET=<clave_real_produccion>
```

1.4 Base de Datos - Índices Faltantes

Problema:

Se detectaron queries lentas debido a índices faltantes en tablas críticas:

1. **Tabla User:**
   - Falta índice en `companyId` para queries filtradas por empresa
   - Falta índice en `role` para queries de autorización

2. **Tabla Payment:**
   - Falta índice en `estado` para filtrado rápido
   - Falta índice compuesto en (`contractId` , `fechaVencimiento`) para queries de vencimientos

3. **Tabla Notification:**
   - Falta índice en `leido` para queries de notificaciones pendientes
   - Falta índice compuesto en (`companyId` , `leido` , `createdAt`)

4. **Tabla MaintenanceRequest:**
   - Falta índice en `estado` y `prioridad`

Impacto:

- Queries lentas en producción con muchos registros

- Timeout de database connections
- Mala experiencia de usuario

Solución:

Agregar índices optimizados al schema de Prisma:

```
model User {
  // campos existentes...
  @@index([companyId])
  @@index([role])
  @@index([companyId, role])
}

model Payment {
  // campos existentes...
  @@index([estado])
  @@index([contractId, fechaVencimiento])
  @@index([companyId, estado, fechaVencimiento])
}

model Notification {
  // campos existentes...
  @@index([leido])
  @@index([companyId, leido, createdAt])
}

model MaintenanceRequest {
  // campos existentes...
  @@index([estado])
  @@index([prioridad])
  @@index([unitId, estado])
}
```

1.5 Problemas de TypeScript

Problemas Detectados:

- Errores de tipos con dependencias de testing (@types)
- Algunos usos de `any` que deberían ser tipados
- Imports no utilizados en varios archivos

Solución:

1. Actualizar tsconfig.json para excluir mejor los archivos de test
2. Audit de tipos `any` y reemplazar por tipos específicos
3. Limpiar imports no utilizados con ESLint

1.6 Bundle Size y Performance

Problemas:

1. **Librerías pesadas no lazy-loaded:**
   - recharts (~400KB)
   - react-big-calendar
   - Componentes de charts cargados eagerly

2. **Dependencias duplicadas:**
   - Múltiples versiones de date-fns
   - react-query y tanstack/react-query

3. **Assets no optimizados:**
   - Imágenes sin next/image
   - PDFs y documentos estáticos sin compresión

Solución:

```
// Lazy load charts
const LazyCharts = dynamic(() => import('./ChartComponents'), {
  loading: () => <LoadingSpinner />,
  ssr: false
});

// Optimizar images
<Image
  src="/image.jpg"
  width={800}
  height={600}
  loading="lazy"
  alt="Description"
/>
```

---

2. OPTIMIZACIONES RECOMENDADAS

2.1 Implementar CDN para Assets Estáticos

- Configurar CloudFront o similar para servir assets estáticos
- Reducir carga en servidor principal
- Mejorar tiempos de carga globalmente

2.2 Implementar Rate Limiting en APIs

```
// Ya existe infraestructura, pero necesita configuración
import { rateLimiters } from '@/lib/rate-limit-enhanced';

// Configurar limits específicos por ruta
const apiLimits = {
  '/api/auth/*': { requests: 5, window: '15m' },
  '/api/*': { requests: 100, window: '1m' }
};
```

2.3 Monitoreo y Logging

1. **Implementar Sentry para error tracking:**

```
javascript
   // Ya está instalado, necesita configuración
   Sentry.init({
     dsn: process.env.SENTRY_DSN,
     environment: process.env.NODE_ENV,
```

```
    tracesSampleRate: 0.1,
  });
```

2. **Configurar Winston para logs estructurados:**
   - Logs a archivo en producción
   - Rotación diaria de logs
   - Niveles apropiados (error, warn, info)

2.4 Caché Strategy

```typescript
// Implementar caché de Redis para queries frecuentes
const getCachedDashboardStats = async (companyId: string) => {
  const cacheKey = `dashboard:stats:${companyId}`;
  const cached = await redis.get(cacheKey);

  if (cached) return JSON.parse(cached);

  const stats = await prisma./* query */;
  await redis.setex(cacheKey, 300, JSON.stringify(stats)); // 5 min cache
  return stats;
};
```

---

3. CHECKLIST DE DEPLOYMENT

Pre-Deployment

- [ ] Actualizar todas las variables de entorno con valores de producción
- [ ] Ejecutar `prisma generate` con nuevo schema optimizado
- [ ] Ejecutar `prisma migrate deploy` para aplicar nuevos índices
- [ ] Limpiar console.log statements
- [ ] Verificar que typescript compile sin errores: `yarn tsc --noEmit`
- [ ] Ejecutar lint: `yarn lint --fix`
- [ ] Optimizar imágenes con next/image
- [ ] Configurar CDN para assets estáticos
- [ ] Configurar monitoreo (Sentry)
- [ ] Configurar rate limiting
- [ ] Backup de base de datos

Durante Deployment

- [ ] Build optimizado: `yarn build`
- [ ] Verificar tamaño de bundle: analizar `.next/build-manifest.json`
- [ ] Test de smoke en staging
- [ ] Migrar database si es necesario
- [ ] Deploy a inmova.app
- [ ] Verificar SSL/TLS configurado
- [ ] Configurar redirects HTTP -> HTTPS

Post-Deployment

- [ ] Verificar todos los endpoints críticos funcionan

- [ ] Test de autenticación (login/logout)
- [ ] Test de pagos con Stripe
- [ ] Verificar emails se envían correctamente
- [ ] Monitorear logs por 24 horas
- [ ] Configurar alertas de error
- [ ] Documentar versión deployed

---

4. CONFIGURACIÓN OPTIMIZADA

next.config.js Optimizado

Ver archivo `next.config.optimized.js` en el proyecto.

Variables de Entorno de Producción

Ver archivo `.env.production.template` en el proyecto.

---

5. MÉTRICAS DE ÉXITO

Performance Targets

- **First Contentful Paint (FCP):** < 1.5s
- **Largest Contentful Paint (LCP):** < 2.5s
- **Time to Interactive (TTI):** < 3.5s
- **Total Bundle Size:** < 300KB (JS inicial)
- **Database Query Time:** < 100ms (p95)
- **API Response Time:** < 200ms (p95)

Reliability Targets

- **Uptime:** > 99.9%
- **Error Rate:** < 0.1%
- **Success Rate (APIs):** > 99%

---

6. CONTACTO Y SOPORTE

Para dudas sobre este reporte:
- Equipo Técnico INMOVA
- Email: tech@inmova.app

Última actualización: 6 de Diciembre de 2025
Versión del reporte: 1.0