📋 CONFIGURACIÓN DE PRODUCCIÓN - INMOVA

> **Documento actualizado:** Diciembre 2025
> **Dominio principal:** inmova.app
> **Versión:** 1.0

✅ CHECKLIST DE CONFIGURACIÓN OBLIGATORIA

1. ✅ NODE_ENV=production

Estado: ✅ CONFIGURADO

```
NODE_ENV=production
```

- Archivo: `.env.production`
- **Acción requerida:** Verificar en variables de entorno del servidor

---

2. ✅ Database Connection Pool

Estado: ✅ CONFIGURADO

Configuración actual:

```
DATABASE_URL='postgresql://
role_587683780:5kWw7vKJBDp9ZA2Jfkt5BdWrAjR0XDe5@db-587683780.db003.hosteddb.reai.io:
5432/587683780?connect_timeout=15&pool_timeout=15&connection_limit=20&schema=public'
```

Parámetros optimizados:
- `connection_limit=20` - Máximo 20 conexiones simultáneas
- `pool_timeout=15` - Timeout de 15 segundos para obtener conexión del pool
- `connect_timeout=15` - Timeout de conexión inicial

Recomendaciones:
1. Monitorear uso del pool con Prisma Studio
2. Ajustar `connection_limit` según tráfico real (recomendado: 10-50)
3. Configurar alertas si el pool se satura (>80% uso)

Comando para verificar conexiones activas:

```
SELECT count(*) FROM pg_stat_activity WHERE datname = '587683780';
```

---

3. ⚠️ Logs a Servicio Externo

Estado: ⚠️ PENDIENTE CONFIGURAR

Opción A: Sentry (Recomendado)

Pasos de configuración:

1. **Crear cuenta en Sentry:**
   - Web: https://sentry.io
   - Crear organización “INMOVA”
   - Crear proyecto “inmova-production”

2. **Obtener DSN:**
   `https://[KEY]@[ORG].ingest.sentry.io/[PROJECT_ID]`

3. **Configurar variables de entorno:**
   ```
   env
   NEXT_PUBLIC_SENTRY_DSN=https://[KEY]@[ORG].ingest.sentry.io/[PROJECT_ID]
   SENTRY_ORG=inmova
   SENTRY_PROJECT=inmova-production
   SENTRY_AUTH_TOKEN=[TOKEN]
   ```

4. **Ya está integrado en el código:**
   - Archivo: `app/providers.tsx`
   - Sentry está instalado: `@sentry/nextjs@^10.29.0`

5. **Configurar source maps:**
   ```
   bash
   cd /home/ubuntu/homming_vidaro/nextjs_space
   yarn add -D @sentry/webpack-plugin
   ```

6. **Verificar integración:**
   ```
   javascript
   // Test error
   throw new Error('Test Sentry Integration');
   ```

Opción B: LogRocket

1. Crear cuenta: https://logrocket.com
2. Obtener App ID
3. Configurar:
   ```
   env
   NEXT_PUBLIC_LOGROCKET_APP_ID=[YOUR_APP_ID]
   ```

Beneficios:
- Session replay (video de sesiones de usuario)
- Performance monitoring
- Console logs capturados
- Network request tracking

---

4. ⚠️ Monitoring Activo

Estado: ⚠️ PENDIENTE CONFIGURAR

A. Uptime Monitoring

Opciones recomendadas:

1. UptimeRobot (Gratuito)

- Web: https://uptimerobot.com
- **Configuración:**
- Monitor Type: HTTP(s)
- URL: https://inmova.app/api/health
- Interval: 5 minutos
- Alert contacts: equipo@inmova.com

2. Pingdom

- Web: https://pingdom.com
- Monitoreo desde múltiples ubicaciones
- Alertas vía email/SMS

3. Better Uptime

- Web: https://betteruptime.com
- Status page público
- On-call scheduling

Crear endpoint de health check:

```
// app/api/health/route.ts
import { NextResponse } from 'next/server';
import { prisma } from '@/lib/db';

export const dynamic = 'force-dynamic';

export async function GET() {
  try {
    // Verificar base de datos
    await prisma.$queryRaw`SELECT 1`;

    return NextResponse.json({
      status: 'healthy',
      timestamp: new Date().toISOString(),
      database: 'connected',
      version: process.env.npm_package_version
    }, { status: 200 });
  } catch (error) {
    return NextResponse.json({
      status: 'unhealthy',
      error: error.message
    }, { status: 503 });
  }
}
```

B. Application Performance Monitoring (APM)

Opciones:

1. New Relic

- Web: https://newrelic.com
- Full-stack observability

- Real user monitoring (RUM)

Configuración:

```
cd /home/ubuntu/homming_vidaro/nextjs_space
yarn add newrelic
```

```
NEW_RELIC_LICENSE_KEY=[KEY]
NEW_RELIC_APP_NAME=INMOVA Production
```

2. Datadog

- Web: https://datadoghq.com
- Logs + APM + Infrastructure

3. Vercel Analytics (Si se usa Vercel)

- Incluido con el hosting
- Web Vitals tracking
- Real-time analytics

---

5. ⚠️ Backups Automáticos

Estado: ⚠️ PENDIENTE CONFIGURAR

Estrategia de Backup Recomendada

A. Backup de Base de Datos PostgreSQL

Configuración Diaria:

```
#!/bin/bash
# /home/ubuntu/scripts/backup-daily.sh

DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_DIR="/home/ubuntu/backups/daily"
DB_URL="postgresql://
role_587683780:5kWw7vKJBDp9ZA2Jfkt5BdWrAjR0XDe5@db-587683780.db003.hosteddb.reai.io:
5432/587683780"
S3_BUCKET="s3://inmova-backups/database/daily/"

# Crear directorio si no existe
mkdir -p $BACKUP_DIR

# Crear backup
pg_dump $DB_URL | gzip > $BACKUP_DIR/inmova_$DATE.sql.gz

# Subir a S3
aws s3 cp $BACKUP_DIR/inmova_$DATE.sql.gz $S3_BUCKET

# Eliminar backups locales mayores a 7 días
find $BACKUP_DIR -name "*.sql.gz" -mtime +7 -delete

echo "Backup completado: inmova_$DATE.sql.gz"
```

Configuración Semanal:

```
#!/bin/bash
# /home/ubuntu/scripts/backup-weekly.sh

DATE=$(date +%Y%m%d)
BACKUP_DIR="/home/ubuntu/backups/weekly"
DB_URL="postgresql://
role_587683780:5kWw7vKJBDp9ZA2Jfkt5BdWrAjR0XDe5@db-587683780.db003.hosteddb.reai.io:
5432/587683780"
S3_BUCKET="s3://inmova-backups/database/weekly/"

mkdir -p $BACKUP_DIR
pg_dump $DB_URL | gzip > $BACKUP_DIR/inmova_weekly_$DATE.sql.gz
aws s3 cp $BACKUP_DIR/inmova_weekly_$DATE.sql.gz $S3_BUCKET

# Mantener 12 semanas (3 meses)
find $BACKUP_DIR -name "*.sql.gz" -mtime +84 -delete

echo "Backup semanal completado: inmova_weekly_$DATE.sql.gz"
```

Configurar Cron Jobs:

```
# Editar crontab
crontab -e

# Añadir líneas:
# Backup diario a las 3:00 AM
0 3 * * * /home/ubuntu/scripts/backup-daily.sh >> /home/ubuntu/logs/backup-daily.log
2>&1

# Backup semanal todos los domingos a las 2:00 AM
0 2 * * 0 /home/ubuntu/scripts/backup-weekly.sh >> /home/ubuntu/logs/backup-
weekly.log 2>&1
```

B. Backup de Archivos (S3)

Los archivos ya están en S3, pero configurar:
- Versionado de bucket
- Lifecycle policies
- Replicación cross-region

C. Configurar AWS S3 Bucket para Backups:

```
# Crear bucket de backups
aws s3 mb s3://inmova-backups --region eu-west-1

# Habilitar versionado
aws s3api put-bucket-versioning \
  --bucket inmova-backups \
  --versioning-configuration Status=Enabled

# Configurar lifecycle (eliminar después de 90 días)
aws s3api put-bucket-lifecycle-configuration \
  --bucket inmova-backups \
  --lifecycle-configuration file://lifecycle.json
```

lifecycle.json:

```
{
  "Rules": [{
    "Id": "Delete old backups",
    "Status": "Enabled",
    "Expiration": { "Days": 90 },
    "NoncurrentVersionExpiration": { "Days": 30 }
  }]
}
```

---

6. ⚠️ CDN para Assets Estáticos

Estado: ⚠️ PENDIENTE CONFIGURAR

Opción A: Cloudflare (Recomendado)

Pasos:

1. **Crear cuenta en Cloudflare:** https://cloudflare.com
2. **Añadir dominio `inmova.app`:**
   - Add site
   - Seguir pasos de configuración DNS
3. **Actualizar nameservers en registrador de dominio:**
   `ns1.cloudflare.com`
   `ns2.cloudflare.com`
4. **Configurar reglas de caché:**
   - Cache Level: Standard
   - Browser Cache TTL: 4 hours
   - Page Rules:
     - `inmova.app/assets/*` → Cache Everything, Edge TTL: 1 month
     - `inmova.app/_next/static/*` → Cache Everything, Edge TTL: 1 year
5. **Optimizaciones:**
   - ✅ Auto Minify (HTML, CSS, JS)
   - ✅ Brotli compression
   - ✅ HTTP/3 (QUIC)
   - ✅ Early Hints
6. **Configurar en Next.js:**
   `env`
   `NEXT_PUBLIC_CDN_URL=https://cdn.inmova.app`

Opción B: AWS CloudFront

Si ya se usa AWS:

```
# Crear distribución CloudFront
aws cloudfront create-distribution \
  --origin-domain-name inmova.app \
  --default-root-object index.html
```

Beneficios:
- Integración con S3
- Costos más predecibles
- Control granular

Opción C: Vercel CDN

Si se despliega en Vercel:
- CDN global automático
- Edge Functions
- Imagen optimization integrado
- Sin configuración adicional necesaria

---

7. ✅ DNS Correctamente Configurado

Estado: ✅ CONFIGURADO (inmova.app)

Registros DNS recomendados:

```
; Registro A principal
inmova.app.          A       [IP_SERVIDOR]

; Registro CNAME para www
www.inmova.app.      CNAME   inmova.app.

; Subdominios
api.inmova.app.      CNAME   inmova.app.
cdn.inmova.app.      CNAME   [CDN_DOMAIN]

; Email (si se usa dominio propio)
@                    MX 10   mx1.sendgrid.net.
@                    MX 20   mx2.sendgrid.net.

; SPF para email
@                    TXT     "v=spf1 include:sendgrid.net ~all"

; DKIM (proporciona SendGrid)
_domainkey           TXT     [SENDGRID_DKIM]

; DMARC
_dmarc               TXT     "v=DMARC1; p=quarantine; rua=mailto:dmarc@inmova.app"
```

Verificar configuración:

```
# Verificar registros DNS
dig inmova.app
dig www.inmova.app
nslookup inmova.app

# Verificar propagación DNS
https://dnschecker.org
```

---

8. ✅ Dominio Custom Funcionando

Estado: ✅ CONFIGURADO

Dominio principal: `inmova.app`

Verificaciones:

```
# 1. Verificar HTTPS
curl -I https://inmova.app

# 2. Verificar redirección www
curl -I https://www.inmova.app

# 3. Verificar certificado SSL
openssl s_client -connect inmova.app:443 -servername inmova.app
```

Configurar SSL/TLS:

- Si se usa Cloudflare: SSL/TLS → Full (strict)
- Si se usa Let's Encrypt:
  `bash`
  `certbot --nginx -d inmova.app -d www.inmova.app`

Renovación automática:

```
# Cron job para renovar certificado
0 0 1 * * certbot renew --quiet
```

---

9. ⚠️ Webhooks de Stripe Apuntan a Producción

Estado: ⚠️ PENDIENTE CONFIGURAR

Configuración de Webhook de Stripe

Pasos:

1. **Acceder a Dashboard de Stripe:**
   - https://dashboard.stripe.com
   - Cambiar a modo LIVE (toggle arriba-derecha)

2. **Configurar Webhook:**
   - Ir a: Developers → Webhooks
   - Click "Add endpoint"
   - URL: `https://inmova.app/api/stripe/webhook`
   - Description: "INMOVA Production Webhook"

3. **Seleccionar eventos:**
   `✓ payment_intent.succeeded`
   `✓ payment_intent.payment_failed`
   `✓ payment_intent.canceled`
   `✓ customer.subscription.created`
   `✓ customer.subscription.updated`
   `✓ customer.subscription.deleted`

✓ invoice.payment_succeeded
✓ invoice.payment_failed
✓ charge.succeeded
✓ charge.failed
✓ charge.refunded

4. **Obtener Signing Secret:**
   - Copiar `whsec_...`
   - Añadir a `.env.production`:
   `env`
   `STRIPE_WEBHOOK_SECRET=whsec_[SIGNING_SECRET]`

5. **Actualizar claves de Stripe a LIVE:**
   `env`
   `STRIPE_SECRET_KEY=sk_live_[LIVE_KEY]`
   `STRIPE_PUBLISHABLE_KEY=pk_live_[LIVE_KEY]`
   `NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_live_[LIVE_KEY]`

6. **Probar webhook:**
   ```bash
   # Instalar Stripe CLI
   stripe listen –forward-to https://inmova.app/api/stripe/webhook

# Enviar evento de prueba
stripe trigger payment_intent.succeeded
```

1. **Verificar logs:**
   - Dashboard → Webhooks → [Tu endpoint] → Logs
   - Verificar respuestas 200 OK

⚠️ **IMPORTANTE:**
- Eliminar webhooks de testing/desarrollo
- Configurar solo el webhook de producción
- Verificar que la URL sea HTTPS
- Mantener el signing secret seguro

---

10. ⚠️ Email Transaccional Configurado

Estado: ⚠️ PENDIENTE CONFIGURAR

Opción A: SendGrid (Recomendado)

Pasos de configuración:

1. **Crear cuenta SendGrid:**
   - https://sendgrid.com
   - Plan recomendado: Essentials ($19.95/mes, 50k emails)

2. **Crear API Key:**
   - Settings → API Keys → Create API Key
   - Permissions: Full Access
   - Copiar clave (solo se muestra una vez)

3. **Verificar dominio:**
   - Settings → Sender Authentication
   - Authenticate Your Domain
   - Dominio: `inmova.app`
   - Añadir registros DNS proporcionados:

```
dns
    em8643.inmova.app    CNAME   u8643.wl.sendgrid.net
    s1._domainkey        CNAME   s1.domainkey.u8643.wl.sendgrid.net
    s2._domainkey        CNAME   s2.domainkey.u8643.wl.sendgrid.net
```

4. **Configurar variables de entorno:**

```
env
  SENDGRID_API_KEY=SG.xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  SENDGRID_FROM_EMAIL=noreply@inmova.app
  SENDGRID_FROM_NAME=INMOVA
```

5. **Crear templates en SendGrid:**
   - Email Templates → Create Template
   - Templates necesarios:
     - Welcome email
     - Password reset
     - Payment confirmation
     - Contract reminder
     - Maintenance notification

6. **Configurar supresiones:**
   - Suppression Management
   - Configurar bounce handling
   - Configurar spam reports

7. **Probar envío:**
   ```typescript
   // Test email
   import { sendEmail } from '@/lib/email-config';

await sendEmail({
to: 'test@example.com',
subject: 'Test Email INMOVA',
html: '

Este es un email de prueba

'

});
```

Opción B: Amazon SES

Si ya se usa AWS:

```
AWS_SES_REGION=eu-west-1
AWS_SES_FROM_EMAIL=noreply@inmova.app
```

Ventajas:
- Muy económico ($0.10 por 1,000 emails)
- Integración con AWS ecosystem
- Alto deliverability

Pasos:
1. Verificar dominio en SES
2. Salir de sandbox mode
3. Configurar DKIM
4. Configurar bounce handling con SNS

Opción C: Postmark

Especializado en transaccional:
- Alta deliverability
- Reporting detallado
- $10/mes por 10k emails

---

11. ✅ Plan de Rollback Definido

Estado: ✅ DOCUMENTADO

🔄 PLAN DE ROLLBACK Y RECUPERACIÓN

Estrategia de Despliegue

1. Deployment con Zero Downtime:

```
#!/bin/bash
# /home/ubuntu/scripts/deploy.sh

set -e

echo "🚀 Iniciando despliegue INMOVA..."

# 1. Backup pre-despliegue
echo "📦 Creando backup..."
/home/ubuntu/scripts/backup-daily.sh

# 2. Pull latest code
echo "📥 Descargando última versión..."
cd /home/ubuntu/homming_vidaro/nextjs_space
git fetch origin
git checkout main
git pull origin main

# 3. Install dependencies
echo "📚 Instalando dependencias..."
yarn install --frozen-lockfile

# 4. Database migrations
echo "🗄️  Aplicando migraciones..."
yarn prisma migrate deploy
yarn prisma generate

# 5. Build
echo "🔨 Building aplicación..."
yarn build

# 6. Verificar build
if [ ! -d ".next" ]; then
  echo "❌ Build falló!"
  exit 1
fi

# 7. Restart service
echo "🔄 Reiniciando servicio..."
pm2 restart inmova-app

# 8. Health check
echo "🏥 Verificando health..."
sleep 10
curl -f https://inmova.app/api/health || {
  echo "❌ Health check falló! Ejecutando rollback..."
  /home/ubuntu/scripts/rollback.sh
  exit 1
}

echo "✅ Despliegue completado exitosamente!"
```

2. Script de Rollback:

```
#!/bin/bash
# /home/ubuntu/scripts/rollback.sh

set -e

echo "⚠️  INICIANDO ROLLBACK..."

cd /home/ubuntu/homming_vidaro/nextjs_space

# 1. Volver al commit anterior
echo "📍 Volviendo a versión anterior..."
git reset --hard HEAD~1

# 2. Restaurar dependencies
echo "📚 Restaurando dependencias..."
yarn install --frozen-lockfile

# 3. Rebuild
echo "🔨 Rebuilding..."
yarn build

# 4. Rollback database si es necesario
echo "🗄️  Verificando migraciones..."
# Si hay migraciones problemáticas, restaurar desde backup
# pg_restore -d DATABASE_URL < /path/to/backup.sql

# 5. Restart
echo "🔄 Reiniciando..."
pm2 restart inmova-app

# 6. Verify
sleep 10
curl -f https://inmova.app/api/health && {
  echo "✅ Rollback completado exitosamente"
} || {
  echo "❌ Rollback falló! Intervención manual requerida"
  # Enviar alerta
  /home/ubuntu/scripts/send-alert.sh "CRITICAL: Rollback failed"
}
```

3. Restauración desde Backup:

```
#!/bin/bash
# /home/ubuntu/scripts/restore-from-backup.sh

BACKUP_FILE=$1

if [ -z "$BACKUP_FILE" ]; then
  echo "Uso: ./restore-from-backup.sh <backup-file>"
  echo "Ejemplo: ./restore-from-backup.sh inmova_20250108_030000.sql.gz"
  exit 1
fi

echo "⚠️  RESTAURANDO DESDE BACKUP: $BACKUP_FILE"
echo "Esta operación sobrescribirá la base de datos actual."
read -p "¿Continuar? (yes/no): " confirm

if [ "$confirm" != "yes" ]; then
  echo "Operación cancelada."
  exit 0
fi

# 1. Descargar backup desde S3 si es necesario
if [[ $BACKUP_FILE == s3://* ]]; then
  echo "📥 Descargando desde S3..."
  aws s3 cp $BACKUP_FILE /tmp/restore.sql.gz
  BACKUP_FILE="/tmp/restore.sql.gz"
fi

# 2. Descomprimir
echo "📦 Descomprimiendo..."
gunzip -c $BACKUP_FILE > /tmp/restore.sql

# 3. Crear backup de seguridad de la DB actual
echo "💾 Creando backup de seguridad..."
pg_dump $DATABASE_URL | gzip > /tmp/pre_restore_backup_$(date +%Y%m%d_%H%M%S).sql.gz

# 4. Restaurar
echo "🔄 Restaurando base de datos..."
psql $DATABASE_URL < /tmp/restore.sql

# 5. Verify
echo "✅ Verificando..."
psql $DATABASE_URL -c "SELECT COUNT(*) FROM \"User\";"

echo "✅ Restauración completada!"

# 6. Cleanup
rm /tmp/restore.sql
```

4. Checklist de Rollback:

```
## ⚠️ CHECKLIST DE ROLLBACK

### Pre-Rollback
- [ ] Identificar versión a la que volver
- [ ] Verificar que existe backup de la DB
- [ ] Notificar al equipo
- [ ] Poner sitio en modo mantenimiento (opcional)

### Durante Rollback
- [ ] Ejecutar script de rollback
- [ ] Verificar que la app arranca
- [ ] Verificar conexión a DB
- [ ] Verificar endpoints críticos
- [ ] Verificar integraciones externas

### Post-Rollback
- [ ] Verificar funcionalidad completa
- [ ] Revisar logs de errores
- [ ] Verificar métricas de rendimiento
- [ ] Notificar resolución al equipo
- [ ] Documentar causa raíz
- [ ] Planear fix y re-deploy
```

5. Configurar Alertas:

```
#!/bin/bash
# /home/ubuntu/scripts/send-alert.sh

MESSAGE=$1

# Enviar a Slack
curl -X POST https://hooks.slack.com/services/YOUR/WEBHOOK/URL \
  -H 'Content-Type: application/json' \
  -d "{\"text\": \"🚨 INMOVA ALERT: $MESSAGE\"}"

# Enviar email
echo "$MESSAGE" | mail -s "[INMOVA] Critical Alert" ops@inmova.app

# Enviar SMS (Twilio)
curl -X POST https://api.twilio.com/2010-04-01/Accounts/YOUR_ACCOUNT/Messages.json \
  --data-urlencode "Body=$MESSAGE" \
  --data-urlencode "From=+1234567890" \
  --data-urlencode "To=+34XXXXXXXXX" \
  -u YOUR_ACCOUNT_SID:YOUR_AUTH_TOKEN
```

📊 MÉTRICAS Y MONITOREO

KPIs a Monitorear

1. **Uptime:**
   - Target: 99.9% (< 43 minutos downtime/mes)
   - Herramienta: UptimeRobot

2. **Response Time:**
   - Target: p95 < 500ms

- Target: p99 < 1s
- Herramienta: New Relic / Sentry

3. **Error Rate:**
   - Target: < 0.1%
   - Herramienta: Sentry

4. **Database Performance:**
   - Query time p95 < 100ms
   - Connection pool usage < 80%
   - Herramienta: Prisma Studio + Custom monitoring

5. **API Success Rate:**
   - Target: > 99.5%
   - Herramienta: Sentry + Custom logs

6. **Stripe Webhook Success:**
   - Target: 100%
   - Herramienta: Stripe Dashboard

Dashboard Recomendado

Grafana Dashboard:

```
Panels:
  - Uptime (Last 30 days)
  - Response Time (p50, p95, p99)
  - Error Rate by endpoint
  - Active Users
  - Database connections
  - API requests/min
  - Memory usage
  - CPU usage
  - Disk usage
```

🔐 SEGURIDAD ADICIONAL

Recomendaciones de Seguridad

1. **WAF (Web Application Firewall):**
   - Cloudflare WAF (incluido en plan Pro)
   - Protección contra:
     - SQL Injection
     - XSS
     - CSRF
     - DDoS

2. **Rate Limiting:**
   - Ya implementado en el código
   - Configurar Redis para rate limiting distribuido:
   `env`
   `REDIS_URL=redis://your-redis-instance:6379`

3. **HTTPS Only:**
   - Forzar HTTPS en Cloudflare
   - HSTS header configurado
   - TLS 1.3

4. **Security Headers:**

```
typescript
  // next.config.js
  headers: async () => [
    {
      source: '/:path*',
      headers: [
        { key: 'X-DNS-Prefetch-Control', value: 'on' },
        { key: 'Strict-Transport-Security', value: 'max-age=63072000' },
        { key: 'X-Frame-Options', value: 'SAMEORIGIN' },
        { key: 'X-Content-Type-Options', value: 'nosniff' },
        { key: 'Referrer-Policy', value: 'origin-when-cross-origin' },
        { key: 'Permissions-Policy', value: 'camera=(), microphone=(), geolocation=()' }
      ]
    }
  ]
```

5. **Secrets Management:**
   - Usar AWS Secrets Manager o HashiCorp Vault
   - Rotar claves cada 90 días
   - Nunca commitear secretos a Git

6. **Auditoría:**
   - Logs de acceso habilitados
   - Revisión mensual de permisos
   - Alertas para accesos sospechosos

📝 CHECKLIST FINAL PRE-LANZAMIENTO

```
### Infraestructura
- [x] NODE_ENV=production configurado
- [x] Database pool optimizado
- [ ] Sentry configurado y verificado
- [ ] UptimeRobot monitoreando
- [ ] Backups automáticos funcionando
- [ ] CDN configurado (Cloudflare)
- [x] DNS configurado correctamente
- [x] Dominio custom funcionando (inmova.app)
- [ ] SSL/TLS configurado y verificado

### Servicios Externos
- [ ] Stripe webhooks apuntando a producción
- [ ] Stripe claves LIVE configuradas
- [ ] SendGrid configurado y dominio verificado
- [ ] Templates de email creados
- [ ] Email deliverability verificado

### Seguridad
- [ ] WAF activo
- [ ] Rate limiting configurado
- [ ] Security headers verificados
- [ ] Secrets rotados y seguros
- [ ] HTTPS forzado
- [ ] HSTS activado

### Monitoreo
- [ ] Sentry recibiendo errores
- [ ] Uptime monitoring activo
- [ ] APM configurado
- [ ] Logs centralizados
- [ ] Alertas configuradas (email/Slack/SMS)
- [ ] Dashboard de métricas creado

### Disaster Recovery
- [ ] Backups diarios funcionando
- [ ] Backups semanales funcionando
- [ ] Scripts de rollback probados
- [ ] Procedimiento de restauración documentado
- [ ] Equipo entrenado en procedimientos

### Performance
- [ ] CDN cacheando assets
- [ ] Database queries optimizadas
- [ ] Connection pool dimensionado
- [ ] Images optimizadas
- [ ] Lazy loading implementado
- [ ] Code splitting verificado

### Compliance
- [ ] GDPR compliance verificado
- [ ] Privacy policy actualizada
- [ ] Terms of service actualizados
- [ ] Cookie consent implementado
- [ ] Data retention policies configuradas
```

📞 CONTACTOS DE EMERGENCIA

```
Equipo Técnico:
  - DevOps Lead: [NOMBRE] - [TELÉFONO] - [EMAIL]
  - Backend Lead: [NOMBRE] - [TELÉFONO] - [EMAIL]
  - Frontend Lead: [NOMBRE] - [TELÉFONO] - [EMAIL]

Proveedores Externos:
  - AWS Support: [PLAN] - https://console.aws.amazon.com/support
  - Stripe Support: https://support.stripe.com
  - SendGrid Support: https://support.sendgrid.com
  - Cloudflare Support: [PLAN] - https://support.cloudflare.com

Escalación:
  - P0 (Critical): Llamar + Email + Slack
  - P1 (High): Email + Slack
  - P2 (Medium): Slack
  - P3 (Low): Ticket sistema
```

📚 RECURSOS ADICIONALES

- Documentación Técnica INMOVA (/TECHNICAL_DOCUMENTATION.md)
- Guía de Arquitectura (/ARCHITECTURE.md)
- API Documentation (/api-docs)
- Runbook Operacional (/RUNBOOK.md)
- Incident Response Plan (/INCIDENT_RESPONSE.md)

Documento creado: Diciembre 2025
Última actualización: [FECHA]
Responsable: Equipo DevOps INMOVA
Versión: 1.0