🚀 GUÍA PASO A PASO - CONFIGURACIÓN DE PRODUCCIÓN INMOVA

> 🎯 **Objetivo:** Completar todas las configuraciones de producción pendientes
> ⏱️ **Tiempo estimado:** 2-3 horas
> 👤 **Responsable:** DevOps / Equipo Técnico

📋 ESTADO ACTUAL

✅ COMPLETADO

- [x] **NODE_ENV=production** - Configurado en `.env`
- [x] **Dominio custom** - `inmova.app` funcionando
- [x] **Health check endpoint** - `/api/health` creado
- [x] **Scripts de backup** - Creados y ejecutables
- [x] **Scripts de deploy y rollback** - Listos para usar
- [x] **Configuración de email** - Módulo creado (pendiente credenciales)

⚠️ PENDIENTE

- [ ] Sentry (Error Tracking)
- [ ] UptimeRobot (Monitoring)
- [ ] Backups automáticos (Cron jobs)
- [ ] CDN (Cloudflare)
- [ ] Webhooks de Stripe (producción)
- [ ] SendGrid (Email transaccional)

1️⃣ CONFIGURAR SENTRY (Error Tracking)

⌛ Tiempo: 15 minutos

Pasos:

1.1 Crear cuenta y proyecto

1. Ir a https://sentry.io
2. Registrarse o iniciar sesión
3. Hacer clic en “Create Project”
4. Seleccionar:
   - Platform: **Next.js**
   - Project name: **inmova-production**
   - Alert frequency: **On every new issue**
5. Hacer clic en “Create Project”

1.2 Obtener DSN

1. En el proyecto recién creado, ir a **Settings** > **Projects** > **inmova-production**

2. En el menú izquierdo, ir a **Client Keys (DSN)**
3. Copiar el **DSN** (formato: `https://[KEY]@[ORG].ingest.sentry.io/[PROJECT_ID]`)

1.3 Obtener Auth Token

1. Ir a **Settings** > **Account** > **API** > **Auth Tokens**
2. Hacer clic en "Create New Token"
3. Configurar:
   - Name: `inmova-deployment`
   - Scopes: Seleccionar:
     - [x] `project:read`
     - [x] `project:releases`
     - [x] `project:write`
     - [x] `org:read`
4. Copiar el token (solo se muestra una vez)

1.4 Configurar variables de entorno

Ejecutar desde el servidor:

```
cd /home/ubuntu/homming_vidaro/nextjs_space

# Añadir a .env (reemplazar con tus valores reales)
echo 'NEXT_PUBLIC_SENTRY_DSN=https://[TU_KEY]@[TU_ORG].ingest.sentry.io/[PROJECT_ID]'
>> .env
echo 'SENTRY_ORG=[TU_ORG]' >> .env
echo 'SENTRY_PROJECT=inmova-production' >> .env
echo 'SENTRY_AUTH_TOKEN=[TU_TOKEN]' >> .env
```

1.5 Verificar integración

```
# Rebuild la app
cd /home/ubuntu/homming_vidaro/nextjs_space
yarn build

# Reiniciar
pm2 restart inmova-app

# Probar que Sentry está capturando errores
curl https://inmova.app/api/test-error
```

Deberías ver el error en el dashboard de Sentry en 1-2 minutos.

1.6 Configurar alertas

1. En Sentry, ir a **Alerts** > **Create Alert**
2. Seleccionar: **Issues**
3. Configurar:
   - When: `A new issue is created`
   - Then: `Send a notification to [tu email]`
4. Guardar

✅ **Sentry configurado**

2️⃣ CONFIGURAR UPTIMEROBOT (Monitoring)

⏳ Tiempo: 10 minutos

Pasos:

2.1 Crear cuenta

1. Ir a https://uptimerobot.com
2. Registrarse (plan gratuito es suficiente)
3. Verificar email

2.2 Crear monitor

1. Hacer clic en “+ Add New Monitor”
2. Configurar:
   - **Monitor Type:** HTTP(s)
   - **Friendly Name:** INMOVA Production
   - **URL:** `https://inmova.app/api/health`
   - **Monitoring Interval:** 5 minutes
   - **Monitor Timeout:** 30 seconds
3. En **Advanced Settings:**
   - **Keyword:** Buscar la palabra `"healthy"` en la respuesta
   - **Keyword Type:** Exists

2.3 Configurar alertas

1. Ir a **My Settings** > **Alert Contacts**
2. Agregar contactos:
   - **Email:** Tu email de operaciones
   - **SMS:** (opcional) Tu teléfono
   - **Slack:** (opcional) Webhook de Slack
3. Volver al monitor y en **Alert Contacts** seleccionar los contactos

2.4 Configurar status page (opcional)

1. Ir a **Public Status Pages**
2. Hacer clic en “Add Status Page”
3. Configurar:
   - **Name:** INMOVA Status
   - **URL:** `inmova-status` (será https://stats.uptimerobot.com/inmova-status)
   - Seleccionar el monitor creado
4. Crear

✅ **UptimeRobot configurado**

3️⃣ CONFIGURAR BACKUPS AUTOMÁTICOS

⏳ Tiempo: 10 minutos

Pasos:

3.1 Crear directorios

```
sudo mkdir -p /home/ubuntu/backups/daily
sudo mkdir -p /home/ubuntu/backups/weekly
sudo mkdir -p /home/ubuntu/logs
sudo chown -R ubuntu:ubuntu /home/ubuntu/backups
sudo chown -R ubuntu:ubuntu /home/ubuntu/logs
```

3.2 Instalar PostgreSQL client (si no está)

```
# Verificar si ya está instalado
which pg_dump

# Si no está, instalar
sudo apt-get update
sudo apt-get install -y postgresql-client

# Verificar versión
pg_dump --version
```

3.3 Probar script de backup manualmente

```
# Probar backup diario
/home/ubuntu/scripts/backup-daily.sh

# Verificar que se creó el backup
ls -lh /home/ubuntu/backups/daily/

# Ver log
tail -f /home/ubuntu/logs/backup-daily-*.log
```

3.4 Configurar Cron jobs

```
# Editar crontab
crontab -e

# Añadir estas líneas al final:
# 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

# Limpiar logs antiguos (cada mes)
0 4 1 * * find /home/ubuntu/logs -name "*.log" -mtime +30 -delete
```

3.5 Verificar cron jobs

```
# Listar cron jobs activos
crontab -l

# Verificar que el servicio cron está activo
sudo systemctl status cron
```

3.6 (Opcional) Configurar AWS S3 para backups remotos

```
# Instalar AWS CLI si no está
sudo apt-get install -y awscli

# Configurar AWS CLI
aws configure
# Ingresar:
# - AWS Access Key ID
# - AWS Secret Access Key
# - Region: eu-west-1
# - Output format: json

# 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

# Probar subida
/home/ubuntu/scripts/backup-daily.sh
aws s3 ls s3://inmova-backups/database/daily/
```

✅ **Backups automáticos configurados**

4️⃣ CONFIGURAR CLOUDFLARE CDN

⏳ Tiempo: 20 minutos

Pasos:

4.1 Crear cuenta en Cloudflare

1. Ir a https://cloudflare.com
2. Registrarse o iniciar sesión
3. Hacer clic en “Add a Site”

4.2 Añadir dominio

1. Ingresar: `inmova.app`
2. Hacer clic en “Add site”
3. Seleccionar plan **Free** (o Pro si necesitas WAF avanzado)
4. Hacer clic en “Continue”

4.3 Verificar registros DNS

Cloudflare escaneará tus registros DNS actuales. Verificar que estén:

```
Tipo    Nombre    Contenido                          Proxy
A       @         [IP_SERVIDOR]                       Proxied (🧡)
CNAME   www       inmova.app                         Proxied (🧡)
```

4.4 Cambiar nameservers

Cloudflare te dará dos nameservers, algo como:

```
ns1.cloudflare.com
ns2.cloudflare.com
```

Ir al registrador de tu dominio (donde compraste `inmova.app`) y cambiar los nameservers:

1. Iniciar sesión en tu registrador (GoDaddy, Namecheap, etc.)
2. Buscar configuración DNS de `inmova.app`
3. Cambiar nameservers a los proporcionados por Cloudflare
4. Guardar cambios

⏳ **Esperar 2-24 horas para propagación DNS**

4.5 Configurar SSL/TLS

1. En Cloudflare, ir a **SSL/TLS**
2. Seleccionar: **Full (strict)**
3. En **Edge Certificates:**
   - [x] Always Use HTTPS: **On**
   - [x] HTTP Strict Transport Security (HSTS): **Enable**
   - [x] Minimum TLS Version: **TLS 1.2**
   - [x] Opportunistic Encryption: **On**
   - [x] TLS 1.3: **On**

4.6 Configurar reglas de caché

1. Ir a **Rules** > **Page Rules** > **Create Page Rule**

Regla 1: Assets estáticos

```
URL: inmova.app/_next/static/*
Settings:
  - Cache Level: Cache Everything
  - Edge Cache TTL: 1 year
  - Browser Cache TTL: 1 year
```

Regla 2: Imágenes

```
URL: inmova.app/assets/*
Settings:
  - Cache Level: Cache Everything
  - Edge Cache TTL: 1 month
  - Browser Cache TTL: 1 week
```

Regla 3: API (sin caché)

```
URL: inmova.app/api/*
Settings:
  - Cache Level: Bypass
```

4.7 Optimizaciones

1. Ir a **Speed** > **Optimization**
2. Activar:
   - [x] **Auto Minify:** HTML, CSS, JavaScript
   - [x] **Brotli:** On
   - [x] **Early Hints:** On
   - [x] **Rocket Loader:** Off (puede causar problemas con React)
3. Ir a **Network**
   - [x] **HTTP/3 (with QUIC):** On
   - [x] **0-RTT Connection Resumption:** On
   - [x] **WebSockets:** On

4.8 Configurar WAF (Web Application Firewall)

1. Ir a **Security** > **WAF**
2. Reglas recomendadas:
   - [x] **Cloudflare Managed Ruleset:** On
   - [x] **Cloudflare OWASP Core Ruleset:** On

4.9 Verificar configuración

```
# Verificar que DNS resuelve a Cloudflare
dig inmova.app

# Verificar headers de Cloudflare
curl -I https://inmova.app
# Deberías ver: server: cloudflare

# Verificar caché de assets
curl -I https://inmova.app/_next/static/css/app.css
# Deberías ver: cf-cache-status: HIT
```

✅ Cloudflare CDN configurado

5️⃣ CONFIGURAR STRIPE PRODUCCIÓN

⏳ Tiempo: 15 minutos

Pasos:

5.1 Activar modo LIVE en Stripe

1. Ir a https://dashboard.stripe.com
2. En la parte superior derecha, cambiar de “Test” a “Live” (toggle)
3. Si es tu primera vez, completar:
   - Business information

- Bank account details
- Identity verification

5.2 Obtener claves LIVE

1. Ir a **Developers** > **API keys** (en modo Live)
2. Copiar:
   - **Publishable key:** `pk_live_...`
   - **Secret key:** `sk_live_...` (hacer clic en “Reveal”)

5.3 Configurar Webhook

1. Ir a **Developers** > **Webhooks**
2. Hacer clic en “Add endpoint”
3. Configurar:
   - **Endpoint URL:** `https://inmova.app/api/stripe/webhook`
   - **Description:** INMOVA Production Webhook
   - **Events to send:** Seleccionar:

```
✓ 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. Hacer clic en “Add endpoint”
5. Copiar el **Signing secret:** `whsec_...`

5.4 Actualizar variables de entorno

```
cd /home/ubuntu/homming_vidaro/nextjs_space

# IMPORTANTE: Hacer backup del .env actual
cp .env .env.backup

# Editar .env manualmente
nano .env

# Reemplazar las claves de Stripe:
STRIPE_SECRET_KEY=sk_live_[TU_CLAVE_LIVE]
STRIPE_PUBLISHABLE_KEY=pk_live_[TU_CLAVE_LIVE]
STRIPE_WEBHOOK_SECRET=whsec_[TU_SIGNING_SECRET]
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_live_[TU_CLAVE_LIVE]

# Guardar (Ctrl+O) y salir (Ctrl+X)
```

5.5 Probar webhook

```
# Instalar Stripe CLI
curl -s https://packages.stripe.com/api/security/keypair/stripe-cli-gpg/public | gpg -
-dearmor | sudo tee /usr/share/keyrings/stripe.gpg
echo "deb [signed-by=/usr/share/keyrings/stripe.gpg] https://packages.stripe.com/
stripe-cli-debian-local stable main" | sudo tee -a /etc/apt/sources.list.d/stripe.list
sudo apt update
sudo apt install stripe

# Login
stripe login

# Forward eventos al webhook local (para testing)
stripe listen --forward-to https://inmova.app/api/stripe/webhook

# En otra terminal, enviar evento de prueba
stripe trigger payment_intent.succeeded
```

5.6 Verificar en Dashboard

1. Ir a **Developers** > **Webhooks** > [Tu endpoint]
2. Ver la pestaña **Logs**
3. Deberías ver respuestas **200 OK**

5.7 Rebuild y reiniciar

```
cd /home/ubuntu/homming_vidaro/nextjs_space
yarn build
pm2 restart inmova-app
```

⚠️ **IMPORTANTE: Eliminar webhooks de testing**

1. Ir a webhooks en modo Test
2. Eliminar todos los endpoints
3. Solo mantener el webhook de producción

✅ **Stripe producción configurado**

6️⃣ CONFIGURAR SENDGRID (Email)

⏳ Tiempo: 20 minutos

Pasos:

6.1 Crear cuenta SendGrid

1. Ir a https://sendgrid.com
2. Hacer clic en “Sign Up”
3. Completar registro
4. Verificar email
5. Completar información de la cuenta

6.2 Crear API Key

1. Ir a **Settings** > **API Keys**

2. Hacer clic en “Create API Key”
3. Configurar:
   - **API Key Name:** `inmova-production`
   - **API Key Permissions: Full Access**
4. Hacer clic en “Create & View”
5. **COPIAR LA CLAVE** (solo se muestra una vez)
6. Guardarla en un lugar seguro

6.3 Verificar dominio

1. Ir a **Settings** > **Sender Authentication**
2. Hacer clic en “Get Started” en **Domain Authentication**
3. Configurar:
   - **Select DNS Host:** [Tu proveedor DNS o “Other Host”]
   - **Domain:** `inmova.app`
   - **Advanced Settings:**
     - [x] Use automated security
     - [x] DKIM/SPF
4. Hacer clic en “Next”

SendGrid te dará registros DNS para añadir:

6.4 Añadir registros DNS

Si usas Cloudflare:

1. Ir a Cloudflare Dashboard
2. Seleccionar `inmova.app`
3. Ir a **DNS** > **Records**
4. Añadir los registros proporcionados por SendGrid:

```
Tipo    Nombre                    Contenido                           Proxy
CNAME   em8643                    u8643.wl.sendgrid.net               DNS only
CNAME   s1._domainkey             s1.domainkey.u8643.wl.sendgrid.net  DNS only
CNAME   s2._domainkey             s2.domainkey.u8643.wl.sendgrid.net  DNS only
TXT     @                         v=spf1 include:sendgrid.net ~all    DNS only
```

⚠️ **IMPORTANTE:** Estos registros deben estar en **DNS only** (nube gris), NO proxied.

1. Guardar cambios
2. Volver a SendGrid y hacer clic en “Verify”

⏳ **La verificación puede tardar 24-48 horas**

6.5 Crear remitente verificado (mientras tanto)

Mientras el dominio se verifica, puedes usar un email verificado:

1. Ir a **Settings** > **Sender Authentication**
2. En **Single Sender Verification**, hacer clic en “Get Started”
3. Configurar:
   - **From Name:** INMOVA
   - **From Email Address:** tu-email@gmail.com (temporalmente)

- **Reply To:** tu-email@gmail.com
- Completar resto de campos

4. Hacer clic en “Create”
5. Verificar el email que recibirás

6.6 Configurar variables de entorno

```
cd /home/ubuntu/homming_vidaro/nextjs_space
nano .env

# Añadir:
SENDGRID_API_KEY=SG.xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
SENDGRID_FROM_EMAIL=noreply@inmova.app
SENDGRID_FROM_NAME=INMOVA

# O si aún no está verificado el dominio, usar temporalmente:
SENDGRID_FROM_EMAIL=tu-email-verificado@gmail.com

# Guardar y salir
```

6.7 Probar envío de email

Crear archivo de prueba:

```
cd /home/ubuntu/homming_vidaro/nextjs_space

# Crear script de test
cat > test-email.ts << 'EOF'
import { sendEmail, emailTemplates } from './lib/email-config';

async function testEmail() {
  const result = await sendEmail({
    to: 'tu-email@example.com',
    subject: 'Test Email - INMOVA',
    ...emailTemplates.welcome('Usuario Test', 'https://inmova.app/login')
  });

  console.log('Result:', result);
}

testEmail();
EOF

# Ejecutar test
npx tsx test-email.ts
```

Deberías recibir el email en 1-2 minutos.

6.8 Configurar supresiones

1. En SendGrid, ir a **Suppressions**
2. Configurar:
   - **Bounce:** Activar automatic bounce handling
   - **Spam Reports:** Activar automatic spam report handling
   - **Blocks:** Activar automatic block handling

6.9 Crear templates (opcional pero recomendado)

1. Ir a **Email API** > **Dynamic Templates**

2. Crear templates para:
   - Bienvenida
   - Recuperación de contraseña
   - Confirmación de pago
   - Recordatorios

6.10 Rebuild y reiniciar

```
cd /home/ubuntu/homming_vidaro/nextjs_space
yarn build
pm2 restart inmova-app
```

✅ **SendGrid configurado**

7️⃣ VERIFICACIÓN FINAL

⏳ Tiempo: 15 minutos

Checklist de verificación:

```
# 1. Verificar health check
curl https://inmova.app/api/health
# Debería retornar: {"status":"healthy", ...}

# 2. Verificar Sentry
# Ir a dashboard de Sentry y verificar que está recibiendo datos

# 3. Verificar UptimeRobot
# Ir a dashboard y verificar que el monitor está "Up"

# 4. Verificar backups
ls -lh /home/ubuntu/backups/daily/
cat /home/ubuntu/logs/backup-daily.log

# 5. Verificar Cloudflare
curl -I https://inmova.app | grep cloudflare

# 6. Verificar Stripe
# Ir a dashboard de Stripe y verificar webhook logs

# 7. Verificar email
# Enviar email de prueba desde la aplicación

# 8. Verificar variables de entorno
cd /home/ubuntu/homming_vidaro/nextjs_space
cat .env | grep NODE_ENV
# Debería mostrar: NODE_ENV=production
```

Pruebas funcionales:

1. **Login/Logout:** Verificar que funciona
2. **Crear propiedad:** Verificar que se guarda en DB
3. **Subir imagen:** Verificar que se sube a S3
4. **Crear contrato:** Verificar que se genera PDF

5. **Simular pago:** Verificar webhook de Stripe
6. **Recuperar contraseña:** Verificar que llega email

Verificar logs:

```
# Ver logs de la aplicación
pm2 logs inmova-app

# Ver logs de nginx (si aplica)
sudo tail -f /var/log/nginx/access.log
sudo tail -f /var/log/nginx/error.log

# Ver logs de sistema
sudo journalctl -u [tu-servicio] -f
```

8️⃣ DOCUMENTACIÓN Y HANDOFF

Documentos a completar:

1. **Runbook operacional:**
   - Procedimientos de emergencia
   - Contactos de escalación
   - Troubleshooting común

2. **Credenciales:**
   - Usar gestor de contraseñas (1Password, LastPass, etc.)
   - Documentar todas las cuentas:
     - Sentry
     - UptimeRobot
     - Cloudflare
     - Stripe
     - SendGrid
     - AWS

3. **Calendario de mantenimiento:**
   - Rotación de claves (cada 90 días)
   - Revisión de backups (mensual)
   - Actualización de dependencias (mensual)
   - Revisión de seguridad (trimestral)

🎉 ¡COMPLETADO!

Si llegaste hasta aquí y todo está funcionando, ¡felicitaciones!

Tu aplicación INMOVA ahora tiene:

- ✅ Monitoring 24/7
- ✅ Error tracking
- ✅ Backups automáticos

- ✅ CDN global
- ✅ Pagos en producción
- ✅ Email transaccional
- ✅ Plan de rollback
- ✅ Scripts de deploy

Próximos pasos recomendados:

1. **Configurar alertas adicionales:**
   - PagerDuty para incidentes críticos
   - Slack notifications
   - SMS para downtime

2. **Optimizaciones de rendimiento:**
   - Configurar Redis para caching
   - Implementar rate limiting más agresivo
   - Optimizar queries de base de datos

3. **Seguridad adicional:**
   - Implementar 2FA para usuarios
   - Configurar IP allowlisting
   - Auditoría de seguridad externa

4. **Escalabilidad:**
   - Configurar auto-scaling
   - Load balancer
   - Database read replicas

📞 **Soporte:**

Si tienes problemas, revisa:
1. Logs de la aplicación: `pm2 logs inmova-app`
2. Documento de configuración: `/home/ubuntu/homming_vidaro/CONFIGURACION_PRODUCCION.md`
3. Dashboard de monitoring

📚 **Recursos:**
- Documentación Next.js: https://nextjs.org/docs
- Documentación Prisma: https://prisma.io/docs
- Documentación Stripe: https://stripe.com/docs
- Documentación SendGrid: https://docs.sendgrid.com

Documento creado: Diciembre 2025
Versión: 1.0
Mantenido por: Equipo DevOps INMOVA