

Guía Completa de Deployment a Producción

Tabla de Contenidos

1. [Deploy con Vercel + Neon \(Recomendado\)](#)
 2. [Deploy con Railway \(Todo-en-Uno\)](#)
 3. [Deploy con AWS \(Empresarial\)](#)
 4. [Deploy con DigitalOcean](#)
 5. [Configuración Post-Deploy](#)
 6. [Troubleshooting](#)
-

1. Deploy con Vercel + Neon (Recomendado)

Tiempo: 30-45 minutos

Costo: \$0 (free tier) o \$19-25/mes (plan pro)

Dificultad: ★ Fácil

Paso 1.1: Crear Base de Datos en Neon

1. Ir a Neon

`https://neon.tech`

2. Crear cuenta

- Click "Sign Up"
- Login con GitHub (recomendado)

3. Crear proyecto

- Click "New Project"
- Nombre: `hotel-pms-produccion`
- Region: Seleccionar más cercana (ej: US East para América)
- PostgreSQL Version: 15 (latest)
- Click "Create Project"

4. Copiar Connection String

- Una vez creado, verás la conexión
- Formato: `postgresql://user:pass@ep-cool-name.region.aws.neon.tech/dbname`
- Click en el icono de copiar
- **GUARDAR ESTE STRING** - lo necesitarás en el paso 1.3

Ejemplo:

`postgresql://neondb_owner:AbCd1234@ep-fancy-cloud-12345678.us-east-2.aws.neon.tech/neondb?sslmode=require`

Paso 1.2: Preparar Código para Deploy

1. Abrir terminal

```
bash
cd /home/ubuntu/hotel_pms_paseo_las_mercedes
```

2. Crear repositorio Git (si no existe)

```
```bash
Inicializar git
git init
```

# Agregar todos los archivos

```
git add .
```

# Hacer commit

```
git commit -m "Preparar para deploy de producción v9.0"
```

```
```
```

1. Crear repositorio en GitHub

- Ir a <https://github.com>
- Click "New repository"
- Nombre: `hotel-pms-paseo-las-mercedes`
- Privado (recomendado)
- No agregar README (ya existe)
- Click "Create repository"

2. Push a GitHub

```
```bash
Agregar remote (reemplazar con tu usuario)
git remote add origin https://github.com/TU-USUARIO/hotel-pms-paseo-las-mercedes.git
```

# Push

```
git branch -M main
```

```
git push -u origin main
```

```
```
```

Paso 1.3: Deploy en Vercel

1. Ir a Vercel

```
https://vercel.com
```

2. Crear cuenta / Login

- Click "Sign Up" o "Login"
- Login con GitHub (usar misma cuenta que en GitHub)
- Autorizar Vercel a acceder a GitHub

3. Importar Proyecto

- Click "Add New..." → "Project"
- Vercel mostrará tus repos de GitHub
- Buscar `hotel-pms-paseo-las-mercedes`
- Click "Import"

4. Configurar Proyecto

- **Framework Preset:** Next.js (detectado automáticamente)
- **Root Directory:** Click “Edit” → cambiar a `app` → Click “Continue”
- **Build Settings:** Dejar por defecto
- NO hacer deploy todavía

5. Agregar Variables de Entorno

Click en “Environment Variables”

Agregar una por una:

DATABASE_URL (requerido)

[Tu connection string de Neon del paso 1.1]

Ejemplo: `postgresql://neondb_owner:AbCd1234@ep-fancy-cloud-12345678.us-east-2.aws.neon.tech/neondb?sslmode=require`

NEXTAUTH_URL (requerido)

`https://tu-proyecto.vercel.app`

Nota: Después del deploy, actualizar con la URL real

NEXTAUTH_SECRET (requerido)

`bash`

Generar en terminal:

`openssl rand -base64 32`

Copiar el resultado. Ejemplo: `Kx3k9vJ2mPqR8wT4nB6cE5dF7gH8iL0m=`

STRIPE_SECRET_KEY (requerido para pagos)

`sk_live_[tu-clave-live-de-stripe]`

Obtener de: <https://dashboard.stripe.com/apikeys> (cambiar a Live mode)

STRIPE_PUBLISHABLE_KEY (requerido para pagos)

`pk_live_[tu-clave-live-de-stripe]`

STRIPE_WEBHOOK_SECRET (requerido para pagos)

`whsec_[tu-webhook-secret]`

Lo configuraremos después del deploy en el paso 1.5

1. Deploy

- Marcar “Production” para todas las variables
- Click “Deploy”
- ⌚ Esperar 2-5 minutos...
- ✅ Ver “Congratulations! 🎉”

2. Copiar URL de Producción

- Click “Visit” o copiar la URL
- Ejemplo: `https://hotel-pms-paseo-las-mercedes.vercel.app`

Paso 1.4: Aplicar Schema de Base de Datos

1. Actualizar NEXTAUTH_URL (si es necesario)

Si la URL real de Vercel es diferente a la que pusiste:

- En Vercel: Settings → Environment Variables
- Editar NEXTAUTH_URL
- Poner URL real: `https://hotel-pms-paseo-las-mercedes.vercel.app`
- Redeploy: Deployments → últimas → “...” → Redeploy

1. Aplicar Schema

Desde tu máquina local:

```
``bash
cd /home/ubuntu/hotel_pms_paseo_las_mercedes/app

# Usar DATABASE_URL de Neon
DATABASE_URL="[tu-connection-string-de-neon]" npx prisma db push
``
```

Verás:

- ✓ Generated Prisma Client
- ✓ Pushed schema to database

1. Verificar Base de Datos

```
bash
DATABASE_URL="[tu-connection-string-de-neon]" npx prisma studio
```

- Abrirá en `http://localhost:5555`
- Deberías ver todas las tablas creadas (vacías)
- Cerrar con Ctrl+C

Paso 1.5: Configurar Stripe Webhook

1. Ir a Stripe Dashboard

`https://dashboard.stripe.com`

2. Cambiar a Live Mode

- Toggle en la esquina superior derecha: “Test mode” → “Live mode”

3. Ir a Webhooks

- Menú lateral: Developers → Webhooks
- Click “Add endpoint”

4. Configurar Endpoint

- **Endpoint URL:**
`https://[tu-url-vercel].vercel.app/api/stripe/webhook`
- **Description:** “Hotel PMS Production Webhook”
- **Events to send:** Click “Select events”

Marcar estos eventos:

- ☒ checkout.session.completed
- ☒ payment_intent.succeeded
- ☒ payment_intent.payment_failed
- ☒ invoice.paid
- ☒ invoice.payment_failed
- Click “Add events”




- Click "Add endpoint"
1. **Copiar Webhook Secret**
 - En la página del webhook, verás "Signing secret"
 - Click "Reveal"
 - Copiar el valor (empieza con `whsec_...`)
 2. **Actualizar STRIPE_WEBHOOK_SECRET en Vercel**
 - Ir a tu proyecto en Vercel
 - Settings → Environment Variables
 - Buscar STRIPE_WEBHOOK_SECRET
 - Click "..." → Edit
 - Pegar el webhook secret
 - Save
 - Redeploy el proyecto
-

Paso 1.6: Crear Usuario Admin Inicial

1. **Abrir la aplicación**
`https://[tu-url-vercel].vercel.app`
 2. **Ir a Sign Up**
 - Crear cuenta con:
 - Email: admin@paseolasmercedeshotel.com (o el que prefieras)
 - Password: (una contraseña segura)
 - Full Name: Administrador
 - Hotel Name: Paseo Las Mercedes
 3. **Verificar Login**
 - Hacer logout
 - Login con las credenciales
 - Deberías entrar al dashboard
-

Paso 1.7: Verificar Funcionalidad

Checklist de Verificación:

1.  **Login funciona**
 - Puedes hacer login/logout
2.  **Dashboard carga**
 - No hay errores en consola (F12)
 - Métricas muestran 0 (normal, base de datos nueva)
3.  **Crear habitación**
 - Ir a Rooms → Create Room
 - Crear una habitación de prueba
 - Verificar que aparece en la lista

4. **✓ Crear huésped**
 - Ir a Guests → Add Guest
 - Crear un huésped de prueba
5. **✓ Crear reservación**
 - Ir a Reservations → New Reservation
 - Crear una reservación de prueba
 - Verificar que aparece
6. **✓ Probar pago (test)**
 - Hacer check-out de una reservación
 - Usar tarjeta de prueba de Stripe:
 - Card: 4242 4242 4242 4242
 - Exp: 12/34
 - CVC: 123
 - Verificar que el pago se procesa

¡Deploy Completo con Vercel + Neon!

Tu sistema está en producción en:

```
https://[tu-proyecto].vercel.app
```

2. Deploy con Railway (Todo-en-Uno)

Tiempo: 20-30 minutos

Costo: ~\$10-15/mes

Dificultad: ★ Más Fácil

Railway incluye automáticamente:

- **✓** Hosting de la app
- **✓** Base de datos PostgreSQL
- **✓** Variable DATABASE_URL configurada automáticamente
- **✓** Certificado SSL

Paso 2.1: Preparar Código

```
cd /home/ubuntu/hotel_pms_paseo_las_mercedes

# Asegurar que el código está en GitHub (igual que Vercel, paso 1.2)
git init
git add .
git commit -m "Deploy a Railway v9.0"
git remote add origin https://github.com/TU-USUARIO/hotel-pms-paseo-las-mercedes.git
git push -u origin main
```

Paso 2.2: Deploy en Railway

1. Ir a Railway

`https://railway.app`

2. Crear cuenta

- Click "Login with GitHub"
- Autorizar Railway

3. Crear Proyecto

- Click "New Project"
- Seleccionar "Deploy from GitHub repo"
- Buscar tu repositorio: `hotel-pms-paseo-las-mercedes`
- Click en el repo

4. Railway Detecta Automáticamente

- Framework: Next.js 
- Root directory: Necesitamos configurarlo

5. Configurar Root Directory

- Click en el servicio desplegado
- Settings → Service Settings
- **Root Directory:** `app`
- **Start Command:** (dejar vacío, usa el default de Next.js)
- **Build Command:** (dejar vacío)
- Save

6. Agregar Base de Datos PostgreSQL

- Click "+ New" en el proyecto
- Seleccionar "Database" → "Add PostgreSQL"
- Railway crea automáticamente la DB y configura `DATABASE_URL`

Paso 2.3: Configurar Variables de Entorno

1. En Railway, click en tu servicio

2. Variables tab

3. Agregar variables:

```
# NEXTAUTH_URL - Railway te da un URL único
NEXTAUTH_URL=https://[tu-proyecto].up.railway.app

# NEXTAUTH_SECRET - generar nuevo
NEXTAUTH_SECRET=[generar-con-openssl-rand-base64-32]

# Stripe Live Keys
STRIPE_SECRET_KEY=sk_live_[tu-clave]
STRIPE_PUBLISHABLE_KEY=pk_live_[tu-clave]
STRIPE_WEBHOOK_SECRET=whsec_[configurar-después]

# Node Environment
NODE_ENV=production
```

1. Save

2. Railway hace redeploy automáticamente

Paso 2.4: Aplicar Schema

Railway aplicará el schema automáticamente en el primer deploy si tienes el script correcto en `package.json`.

Para verificar/aplicar manualmente:

```
# Copiar DATABASE_URL de Railway
# Railway Variables → DATABASE_URL → Copy

cd /home/ubuntu/hotel_pms_paseo_las_mercedes/app

DATABASE_URL="[railway-database-url]" npx prisma db push
```

Paso 2.5: Configurar Stripe Webhook

Igual que en Vercel (paso 1.5), pero usar la URL de Railway:

```
https://[tu-proyecto].up.railway.app/api/stripe/webhook
```

Paso 2.6: Verificar

Abrir `https://[tu-proyecto].up.railway.app` y verificar igual que en paso 1.7

3. Deploy con AWS (Empresarial)

Tiempo: 2-4 horas

Costo: ~\$80-150/mes

Dificultad: ★★★ Avanzado

Arquitectura AWS:

- **RDS PostgreSQL:** Base de datos
- **EC2:** Servidor de aplicación
- **Application Load Balancer:** Distribución de tráfico
- **S3:** Assets estáticos
- **CloudFront:** CDN (opcional)
- **Route 53:** DNS (opcional)

Paso 3.1: Crear RDS PostgreSQL

1. Ir a AWS Console → RDS
2. Create database
 - Engine: PostgreSQL

- Version: 15.x
- Template: Production (o Dev/Test para ahorrar)
- **DB instance identifier:** `hotel-pms-production`
- **Master username:** `postgres`
- **Master password:** [contraseña segura - guardarla]
- **DB instance class:**
 - Hotel pequeño: `db.t3.micro` (\$15/mes)
 - Hotel mediano: `db.t3.small` (\$30/mes)
 - Hotel grande: `db.t3.medium` (\$60/mes)
 - **Storage:** 20GB SSD (autoescalable)
 - **VPC:** Default (o crear una dedicada)
 - **Public access:** Yes (para desarrollo, luego cambiar a No)
 - **VPC security group:** Crear nuevo
 - **Database name:** `hotelpms`

3. Create database

4. Esperar 5-10 minutos

5. Copiar endpoint cuando esté disponible

- Ejemplo: `hotel-pms-production.c1a2b3c4d5e6.us-east-1.rds.amazonaws.com`

Connection String:

```
postgresql://postgres:[password]@[endpoint]:5432/hotelpms
```

Paso 3.2: Crear EC2 Instance

1. EC2 Console → Launch Instance

2. Configuración:

- **Name:** `hotel-pms-app-server`
- **AMI:** Ubuntu Server 22.04 LTS
- **Instance type:**
 - Pequeño: `t3.small` (\$15/mes)
 - Mediano: `t3.medium` (\$30/mes)
 - **Key pair:** Crear nuevo o usar existente (para SSH)
 - **Network:** Same VPC as RDS
 - **Security group:**
 - Allow SSH (22) from your IP
 - Allow HTTP (80) from anywhere
 - Allow HTTPS (443) from anywhere
 - Allow Custom TCP (3000) from anywhere (temporalmente)

3. Launch instance

4. Esperar 2-3 minutos

Paso 3.3: Configurar EC2 Instance

1. Conectar vía SSH

```
bash
ssh -i your-key.pem ubuntu@[ec2-public-ip]
```

2. Instalar Node.js 18

```
``bash
# Update system
sudo apt update && sudo apt upgrade -y

# Install Node.js 18
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# Verify
node -version # Should show v18.x.x
npm -version
``
```

1. Instalar PM2 (Process Manager)

```
bash
sudo npm install -g pm2
sudo npm install -g yarn
```

2. Instalar Nginx

```
bash
sudo apt install -y nginx
```

3. Instalar PostgreSQL client

```
bash
sudo apt install -y postgresql-client
```

Paso 3.4: Deploy de la Aplicación

1. Clonar repositorio

```
bash
cd /home/ubuntu
git clone https://github.com/TU-USUARIO/hotel-pms-paseo-las-mercedes.git
cd hotel-pms-paseo-las-mercedes/app
```

2. Crear .env

```
bash
nano .env
```

Contenido:

```
env
DATABASE_URL="postgresql://postgres:[password]@[rds-endpoint]:5432/hotelpms"
NEXTAUTH_URL="https://[tu-dominio-o-ip-ec2]"
NEXTAUTH_SECRET="[generar-nuevo]"
STRIPE_SECRET_KEY="sk_live_..."
STRIPE_PUBLISHABLE_KEY="pk_live_..."
```

```
STRIPE_WEBHOOK_SECRET="whsec_..."
```

```
NODE_ENV="production"
```

Guardar: Ctrl+X, Y, Enter

1. Instalar dependencias

```
bash
```

```
yarn install
```

2. Aplicar schema de DB

```
bash
```

```
npx prisma db push
```

```
npx prisma generate
```

3. Build de producción

```
bash
```

```
yarn build
```

4. Iniciar con PM2

```
bash
```

```
pm2 start yarn --name "hotel-pms" -- start
```

```
pm2 save
```

```
pm2 startup
```

5. Verificar que corre

```
bash
```

```
pm2 status
```

```
pm2 logs hotel-pms
```

Paso 3.5: Configurar Nginx como Reverse Proxy

1. Crear configuración

```
bash
```

```
sudo nano /etc/nginx/sites-available/hotel-pms
```

2. Contenido:

```
```nginx
server {
 listen 80;
 server_name [tu-dominio-o-ip-ec2];

 location / {
 proxy_pass http://localhost:3000;
 proxy_http_version 1.1;
 proxy_set_header Upgrade $http_upgrade;
 proxy_set_header Connection 'upgrade';
 proxy_set_header Host $host;
 proxy_cache_bypass $http_upgrade;
 proxy_set_header X-Real-IP $remote_addr;
 proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
 proxy_set_header X-Forwarded-Proto $scheme;
 }
}
```

```
}
...
```

### 3. Activar sitio

```
bash
sudo ln -s /etc/nginx/sites-available/hotel-pms /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

---

## Paso 3.6: Configurar SSL con Let's Encrypt

### 1. Instalar Certbot

```
bash
sudo apt install -y certbot python3-certbot-nginx
```

### 2. Obtener certificado

```
bash
sudo certbot --nginx -d [tu-dominio.com]
```

### 3. Seguir instrucciones

- Ingresar email
- Aceptar términos
- Certbot configurará automáticamente HTTPS

### 4. Auto-renovación

```
bash
sudo certbot renew --dry-run
```

---

## Paso 3.7: Configurar Security Group de RDS

Una vez que la app funciona, asegurar RDS:

### 1. RDS Console → Security Groups

### 2. Editar Inbound Rules del Security Group de RDS

### 3. Cambiar:

- Type: PostgreSQL
- Source: Security Group de EC2 (en lugar de "Anywhere")

### 4. Save

Ahora RDS solo acepta conexiones desde EC2.

---

## Paso 3.8: Configurar Backups

### 1. RDS Backups (automático)

- RDS hace backups diarios automáticos
- Retention: 7 días (default) o más

### 2. Backup de código

```
bash
```

```
En EC2, crear script de backup
nano ~/backup-app.sh
```

Contenido:

```
bash
#!/bin/bash
DATE=$(date +%Y%m%d_%H%M%S)
cd /home/ubuntu/hotel-pms-paseo-las-mercedes
tar -czf /home/ubuntu/backups/app_${DATE}.tar.gz .
Mantener solo últimos 7 backups
ls -t /home/ubuntu/backups/app_*.tar.gz | tail -n +8 | xargs rm -f
```

```
```bash
chmod +x ~/backup-app.sh
mkdir ~/backups

# Agregar a cron (diario a las 3 AM)
crontab -e
```
```

Agregar línea:

```
0 3 * * * /home/ubuntu/backup-app.sh
```

---

## 4. Deploy con DigitalOcean

---

**Tiempo:** 1-2 horas

**Costo:** ~\$30-50/mes

**Dificultad:** ★★ Medio

### Opción A: DigitalOcean App Platform (Más Fácil)

Similar a Railway:

#### 1. Ir a DigitalOcean

```
https://cloud.digitalocean.com
```

#### 2. Create → Apps

#### 3. Connect GitHub

#### 4. Seleccionar repo

#### 5. Configurar:

- Type: Web Service
- Build Command: `cd app && yarn install && yarn build`
- Run Command: `cd app && yarn start`
- HTTP Port: 3000

#### 6. Add Database

- Add Component → Database → PostgreSQL

#### 7. Add Environment Variables

(igual que Railway/Vercel)

## 8. Deploy

---

### Opción B: DigitalOcean Droplet (Más Control)

Similar a AWS EC2:

1. **Create Droplet**

- Ubuntu 22.04
- Basic Plan: \$12-24/mes
- Add SSH key

2. **Create Managed Database**

- PostgreSQL
- Basic: \$15/mes

3. **Configurar Droplet**

(mismo proceso que AWS EC2, pasos 3.3-3.6)

---

## 5. Configuración Post-Deploy

---

### 5.1: Crear Datos Iniciales

#### Opción 1: Seed Scripts

```
cd app

Seed de habitaciones (si aplica)
npx tsx scripts/seed-rooms.ts

Seed de staff (si aplica)
npx tsx scripts/seed-staff.ts

Seed de inventario (si aplica)
npx tsx scripts/seed-inventory.ts
```

#### Opción 2: Manual desde UI

Entrar a la aplicación y crear:

- Usuario admin
  - Tipos de habitaciones
  - Habitaciones
  - Configuración del hotel
- 

### 5.2: Configurar Dominio Personalizado

#### En Vercel:

1. **Comprar dominio** (ej: paseolasmercedeshotel.com en Namecheap, GoDaddy, etc.)
2. **En Vercel:**
  - Settings → Domains

- Add Domain
- Ingresar tu dominio

### 3. **Configurar DNS** (en tu registrador):

- Tipo: A
- Name: @
- Value: [IP proporcionada por Vercel]

O para subdomain (pms.tuhotel.com):

- Tipo: CNAME
- Name: pms
- Value: cname.vercel-dns.com

#### 1. **Esperar propagación** (5 min - 48 horas)

### En Railway:

1. **Railway Dashboard** → **Settings** → **Custom Domain**
2. **Agregar dominio**
3. **Configurar DNS** similar a Vercel

### En AWS:

1. **Usar Route 53** (opcional, \$0.50/mes + \$0.40 por millón de queries)
2. **O configurar DNS en registrador** apuntando a:
  - IP pública de EC2, o
  - DNS del Load Balancer

## 5.3: Configurar Monitoreo

### Opción 1: Vercel Analytics (incluido)

Vercel incluye analytics básico gratuito.

### Opción 2: Sentry (Tracking de Errores)

#### 1. **Crear cuenta en Sentry**

`https://sentry.io`

#### 2. **Crear proyecto Next.js**

#### 3. **Instalar Sentry**

```
bash
cd app
yarn add @sentry/nextjs
npx @sentry/wizard -i nextjs
```

#### 4. **Configurar variables:**

```
env
NEXT_PUBLIC_SENTRY_DSN=[tu-dsn]
SENTRY_AUTH_TOKEN=[tu-token]
```

#### 5. **Redeploy**

### Opción 3: Monitoring AWS (CloudWatch)

Para AWS EC2:

#### 1. Instalar CloudWatch Agent

```
bash
wget https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-
cloudwatch-agent.deb
sudo dpkg -i -E ./amazon-cloudwatch-agent.deb
```

#### 2. Configurar métricas

#### 3. Ver en CloudWatch Dashboard

## 5.4: Configurar Backups Automáticos

### En Neon:

- Backups automáticos incluidos
- Retention: 7 días (free), 30 días (pro)

### En Railway:

- Backups automáticos incluidos
- Puede hacer backup manual: Database → Backups → Create

### En AWS RDS:

- Configurado en creación
- Retention: 7-35 días
- Backups diarios automáticos

### Backup Manual de Base de Datos:

```
Crear backup
pg_dump "[DATABASE_URL]" > backup_$(date +%Y%m%d).sql

Restaurar backup
psql "[DATABASE_URL]" < backup_20250101.sql
```

Automatizar con cron (Linux/Mac):

```
Editar cron
crontab -e

Agregar (backup diario a las 2 AM):
0 2 * * * pg_dump "[DATABASE_URL]" > ~/backups/hotel_pms_$(date +%Y\%m\%d).sql

Rotar (mantener solo 30 días)
0 3 * * * find ~/backups -name "hotel_pms_*.sql" -mtime +30 -delete
```



## 5.5: SSL/HTTPS

### Vercel/Railway:

-  HTTPS automático incluido

### AWS/DigitalOcean:

- Usar Let's Encrypt (paso 3.6)
  - O usar AWS Certificate Manager para ALB
- 

## 6. Troubleshooting

### Error: “Build failed”

**Causa:** Error de TypeScript o dependencias

**Solución:**

```
Verificar build localmente
cd app
yarn install
yarn build

Ver errores específicos
```

---

### Error: “Can’t connect to database”

**Causa:** DATABASE\_URL incorrecto o base de datos no accesible

**Solución:**

1. Verificar DATABASE\_URL en variables de entorno
2. Verificar formato: `postgresql://user:pass@host:5432/database`
3. Verificar security groups (AWS) o firewall
4. Probar conexión:

```
bash
```

```
psql "[DATABASE_URL]"
```

---

### Error: “NextAuth configuration error”

**Causa:** NEXTAUTH\_URL o NEXTAUTH\_SECRET faltante/incorrecto

**Solución:**

1. Verificar que NEXTAUTH\_URL coincida con la URL real
  2. Asegurar que NEXTAUTH\_SECRET esté configurado (32+ caracteres)
  3. Redeploy después de cambiar variables
- 

### Error: “Stripe webhook signature verification failed”

**Causa:** STRIPE\_WEBHOOK\_SECRET incorrecto

**Solución:**

1. Ir a Stripe Dashboard → Webhooks
  2. Verificar que el endpoint URL sea correcto
  3. Copiar el signing secret correcto
  4. Actualizar STRIPE\_WEBHOOK\_SECRET en variables de entorno
  5. Redeploy
- 

## Error 500 en producción

**Causa:** Error de runtime no capturado

**Solución:**

1. Ver logs:
    - **Vercel:** Deployments → última → Logs
    - **Railway:** Service → Deployments → View Logs
    - **AWS EC2:** `pm2 logs hotel-pms`
  1. Buscar el error específico
  2. Verificar todas las variables de entorno
- 

## Prisma Client Error

**Causa:** Schema no aplicado en base de datos

**Solución:**

```
Aplicar schema
DATABASE_URL="[url-produccion]" npx prisma db push

Regenerar cliente
npx prisma generate
```

---

## Performance Lento

**Causas posibles:**

1. Base de datos muy pequeña (escalar)
2. Queries no optimizados
3. Falta de índices en DB
4. Servidor muy pequeño (escalar)

**Soluciones:**

1. Agregar índices en Prisma:

```
prisma
model Reservation {
 // ...
 @@index([checkInDate, checkOutDate])
}
```

```
@@index([status])
```

```
}
```

1. Escalar base de datos o servidor
2. Implementar caching (Redis)



## Comparación Final de Opciones

| Característica | Vercel + Neon | Railway     | AWS          | DigitalOcean |
|----------------|---------------|-------------|--------------|--------------|
| Setup Time     | 30-45 min     | 20-30 min   | 2-4 hrs      | 1-2 hrs      |
| Dificultad     | ★ Fácil       | ★ Fácil     | ★★★★ Difícil | ★★ Medio     |
| Costo/mes      | \$0-40        | \$10-15     | \$80-150     | \$30-50      |
| Escalabilidad  | Alta          | Media       | Muy Alta     | Alta         |
| HTTPS          | ✅ Auto        | ✅ Auto      | Manual       | Manual       |
| Backups        | ✅ Auto        | ✅ Auto      | ✅ Config     | ✅ Config     |
| Mejor para     | Mayoría       | Simplicidad | Empresas     | Balance      |



## Checklist Final Post-Deploy

- ☐ Aplicación accesible vía HTTPS
- ☐ Login funciona correctamente
- ☐ Puede crear habitaciones
- ☐ Puede crear reservaciones
- ☐ Puede hacer check-in
- ☐ Puede hacer check-out
- ☐ Pagos Stripe funcionan (usar tarjeta de prueba)
- ☐ Webhook de Stripe configurado y funciona
- ☐ Todas las páginas cargan sin errores
- ☐ Logs no muestran errores críticos
- ☐ Backups configurados
- ☐ Dominio personalizado (opcional)
- ☐ Monitoreo configurado (opcional)
- ☐ SSL/HTTPS activo
- ☐ Variables de entorno verificadas

## ¡Sistema en Producción!

---

Felicitaciones, tu Hotel PMS está en producción y listo para recibir reservaciones reales.

---

**Última actualización:** 1 de Octubre, 2025

**Sistema:** Hotel PMS Paseo Las Mercedes v9.0

**Estado:**  PRODUCTION READY