Guía de Deployment para INMOVA

📋 Índice

🚀 Configuración Inicial

Prerrequisitos

- Node.js 18 o superior
- Yarn instalado globalmente
- Cuenta de Vercel activa
- Cuenta de GitHub (para CI/CD)
- Git configurado

Instalación de Vercel CLI

```
npm install -g vercel@latest
```

🔑 Obtener Token de Vercel

Paso 1: Acceder a Vercel

1. Ve a vercel.com (https://vercel.com)
2. Inicia sesión con tu cuenta
3. Ve a **Settings** → **Tokens**

Paso 2: Crear un nuevo Token

1. Haz clic en **Create Token**
2. Dale un nombre descriptivo (ej: `inmova-deployment-token`)
3. Selecciona el alcance:
   - **Full Account** (recomendado para CI/CD)
   - O selecciona proyectos específicos
4. Define la expiración (recomendado: 1 año)
5. Copia el token generado (solo se muestra una vez)

Paso 3: Guardar el Token

Localmente (para deployment manual):

```
# En el archivo .env del proyecto
VERCEL_TOKEN=your_vercel_token_here
```

En GitHub (para CI/CD):

1. Ve a tu repositorio en GitHub
2. **Settings → Secrets and variables → Actions**
3. Haz clic en **New repository secret**
4. Nombre: `VERCEL_TOKEN`
5. Valor: Pega tu token de Vercel
6. Haz clic en **Add secret**

📦 Deployment Manual

Método 1: Usando Vercel CLI

Primera vez (vincular proyecto):

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

# Login en Vercel
vercel login

# Vincular proyecto
vercel link
# Selecciona: inmova.app o crea un nuevo proyecto

# Deploy a producción
vercel --prod
```

Deployments subsecuentes:

```
# Preview deployment
vercel

# Production deployment
vercel --prod

# Con token (sin login interactivo)
vercel --prod --token=$VERCEL_TOKEN
```

Método 2: Usando el script de deploy

Crea un script de deployment:

```
# Archivo: scripts/deploy.sh
#!/bin/bash

set -e

echo "🚀 Iniciando deployment a Vercel..."

# Navegar al directorio del proyecto
cd "$(dirname "$0")/../nextjs_space"

# Instalar dependencias
echo "📦 Instalando dependencias..."
yarn install --frozen-lockfile

# Generar Prisma Client
echo "🔧 Generando Prisma Client..."
yarn prisma generate

# Build del proyecto
echo "🏗️  Building proyecto..."
yarn build

# Deploy a Vercel
if [ "$1" == "prod" ] || [ "$1" == "production" ]; then
  echo "🌍 Deploying a PRODUCCIÓN..."
  vercel --prod --token=$VERCEL_TOKEN
else
  echo "👀 Deploying a PREVIEW..."
  vercel --token=$VERCEL_TOKEN
fi

echo "✅ Deployment completado!"
```

Hacer el script ejecutable y usarlo:

```
chmod +x scripts/deploy.sh

# Preview deployment
./scripts/deploy.sh

# Production deployment
./scripts/deploy.sh prod
```

⚙️ Configuración CI/CD

GitHub Actions (Recomendado)

Ya está configurado en `.github/workflows/deploy-vercel.yml`

Configurar Secrets en GitHub:

1. Ve a tu repositorio
2. **Settings → Secrets and variables → Actions**
3. Agrega los siguientes secrets:

```
VERCEL_TOKEN=<tu_token_de_vercel>
VERCEL_ORG_ID=<tu_org_id>
VERCEL_PROJECT_ID=<tu_project_id>
DATABASE_URL=<tu_database_url>
NEXTAUTH_SECRET=<tu_nextauth_secret>
NEXTAUTH_URL=https://inmova.app
```

Obtener IDs de Vercel:

```
# Desde el directorio del proyecto
cd nextjs_space

# Obtener Project ID
vercel project ls

# O desde .vercel/project.json después de hacer vercel link
cat .vercel/project.json
```

Flujo Automático:

1. **Push a** `main` → Deploy automático a producción
2. **Pull Request** → Deploy automático de preview
3. **Manual trigger** → Usar “Run workflow” en GitHub Actions

GitLab CI/CD

Si usas GitLab, crea `.gitlab-ci.yml`:

```yaml
stages:
  - build
  - deploy

variables:
  NODE_VERSION: "18"

build:
  stage: build
  image: node:${NODE_VERSION}
  script:
    - cd nextjs_space
    - yarn install --frozen-lockfile
    - yarn prisma generate
    - yarn build
  artifacts:
    paths:
      - nextjs_space/.next/
      - nextjs_space/node_modules/
    expire_in: 1 hour
  only:
    - main
    - develop

deploy_production:
  stage: deploy
  image: node:${NODE_VERSION}
  dependencies:
    - build
  script:
    - npm install -g vercel
    - cd nextjs_space
    - vercel --prod --token=$VERCEL_TOKEN
  only:
    - main
  environment:
    name: production
    url: https://inmova.app

deploy_preview:
  stage: deploy
  image: node:${NODE_VERSION}
  dependencies:
    - build
  script:
    - npm install -g vercel
    - cd nextjs_space
    - vercel --token=$VERCEL_TOKEN
  only:
    - develop
  environment:
    name: preview
```

🔐 Variables de Entorno

En Vercel Dashboard:

1. Ve a tu proyecto en Vercel

2. **Settings → Environment Variables**
3. Agrega cada variable:

Variables Requeridas:

```
# Database
DATABASE_URL=postgresql://...

# Auth
NEXTAUTH_SECRET=...
NEXTAUTH_URL=https://inmova.app

# AWS
AWS_PROFILE=hosted_storage
AWS_REGION=us-west-2
AWS_BUCKET_NAME=...
AWS_FOLDER_PREFIX=...

# Stripe
STRIPE_SECRET_KEY=sk_...
STRIPE_PUBLISHABLE_KEY=pk_...
STRIPE_WEBHOOK_SECRET=whsec_...
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_...

# Security
CRON_SECRET=...
ENCRYPTION_KEY=...
```

Para cada variable:

1. Nombre de la variable
2. Valor
3. Selecciona entornos:
   - ✅ Production
   - ✅ Preview
   - ✅ Development (opcional)
4. Click **Save**

Usando Vercel CLI:

```
# Agregar una variable
vercel env add DATABASE_URL production

# Listar variables
vercel env ls

# Eliminar una variable
vercel env rm DATABASE_URL production

# Pull variables localmente
vercel env pull .env.local
```

🛠️ Comandos Útiles

Vercel CLI:

```
# Ver información del proyecto
vercel inspect

# Ver logs en tiempo real
vercel logs --follow

# Ver deployments
vercel ls

# Rollback a un deployment anterior
vercel rollback [deployment-url]

# Remover un deployment
vercel remove [deployment-id]

# Alias de dominio
vercel alias set [deployment-url] inmova.app

# Ver dominios configurados
vercel domains ls
```

Prisma:

```
# Generar cliente
yarn prisma generate

# Aplicar migraciones
yarn prisma migrate deploy

# Ver estado de migraciones
yarn prisma migrate status

# Seed de base de datos
yarn prisma db seed
```

Next.js:

```
# Desarrollo local
yarn dev

# Build local
yarn build

# Producción local
yarn start

# Linting
yarn lint
```

🔧 Troubleshooting

Error: “Command failed with exit code 1”

Solución:

```
# Limpiar cache y reinstalar
rm -rf node_modules .next
yarn install
yarn build
```

Error: “DATABASE_URL is not defined”

Solución:

1. Verifica que la variable esté en Vercel:
   `bash`
   `vercel env ls`
2. Agrégala si no existe:
   `bash`
   `vercel env add DATABASE_URL production`
3. O actualiza `.env`:
   `bash`
   `echo "DATABASE_URL=postgresql://..." >> .env`

Error: “Prisma Client not initialized”

Solución:

```
# Regenerar Prisma Client
yarn prisma generate

# En Vercel, agregar build command personalizado:
# Settings → General → Build & Development Settings
# Build Command: yarn prisma generate && yarn build
```

Deployment lento o timeout

Solución:

1. Aumentar timeout en `vercel.json`:
   `json`
   `{`
   `"functions": {`
   `"app/api/**/*.ts": {`
   `"maxDuration": 60`
   `}`
   `}`
   `}`
2. Optimizar build:
   `bash`

```
  # Usar cache de Yarn
  vercel --prod --build-env YARN_CACHE_FOLDER=/tmp/yarn-cache
```

Error: “Invalid token”

Solución:

1. Genera un nuevo token en Vercel
2. Actualiza el secret en GitHub
3. Actualiza tu `.env` local

Error de permisos en GitHub Actions

Solución:

1. Ve a **Settings** → **Actions** → **General**
2. En “Workflow permissions”, selecciona **Read and write permissions**
3. Marca **Allow GitHub Actions to create and approve pull requests**

📚 Recursos Adicionales

- Documentación de Vercel (https://vercel.com/docs)
- Vercel CLI Docs (https://vercel.com/docs/cli)
- GitHub Actions Docs (https://docs.github.com/en/actions)
- Next.js Deployment (https://nextjs.org/docs/deployment)
- Prisma Deployment (https://www.prisma.io/docs/guides/deployment)

🎯 Checklist de Deployment

Antes de deployar:

- [ ] Todas las pruebas pasan
- [ ] No hay errores de linting
- [ ] Base de datos migrada
- [ ] Variables de entorno configuradas
- [ ] Build local exitoso
- [ ] Commit y push a GitHub

Durante el deployment:

- [ ] Verificar logs en Vercel Dashboard
- [ ] Revisar que no haya errores en consola
- [ ] Probar funcionalidades críticas

Después del deployment:

- [ ] Verificar que el sitio carga correctamente
- [ ] Probar autenticación
- [ ] Verificar conexión a base de datos
- [ ] Probar integraciones (Stripe, AWS, etc.)

- [ ] Verificar logs de errores
- [ ] Monitorear métricas de performance

💡 Tips y Best Practices

1. **Nunca commitear el token de Vercel** al repositorio
2. **Usar** `.env.local` para desarrollo local
3. **Hacer backup** de la base de datos antes de deployments mayores
4. **Usar preview deployments** para probar cambios
5. **Configurar alertas** en Vercel para monitorear errores
6. **Documentar cambios** en el CHANGELOG
7. **Usar semantic versioning** para releases
8. **Configurar health checks** para endpoints críticos
9. **Implementar rollback strategy** para emergencias
10. **Monitorear costos** de Vercel regularmente

📞 Soporte

Si tienes problemas con el deployment:

1. Revisa los logs en Vercel Dashboard
2. Consulta esta guía de troubleshooting
3. Revisa la documentación oficial de Vercel
4. Contacta al equipo de desarrollo

Última actualización: Diciembre 2024
Versión: 1.0.0
Mantenido por: Equipo INMOVA