🚀 Guía de Despliegue - EscalaFin MVP

Esta guía proporciona instrucciones detalladas para desplegar EscalaFin MVP en diferentes entornos.

📋 Requisitos Previos

Servidor de Producción

- Node.js 18+
- PostgreSQL 14+
- PM2 (recomendado para producción)
- Nginx (como proxy reverso)
- Certificado SSL

Servicios Externos

- Cuenta de AWS (para S3)
- Cuenta de Openpay
- EvolutionAPI configurado
- Dominio configurado

🔧 Configuración de Variables de Entorno

Variables Críticas de Producción

```
# Database
DATABASE_URL="postgresql://usuario:contraseña@host:puerto/database"

# NextAuth
NEXTAUTH_URL="https://tu-dominio.com"
NEXTAUTH_SECRET="tu-secret-ultra-seguro-minimo-32-caracteres"

# Openpay Production
OPENPAY_MERCHANT_ID="tu-merchant-id-produccion"
OPENPAY_PRIVATE_KEY="tu-private-key-produccion"
OPENPAY_PUBLIC_KEY="tu-public-key-produccion"
OPENPAY_SANDBOX="false"

# WhatsApp - EvolutionAPI
EVOLUTIONAPI_BASE_URL="https://tu-evolution-api-produccion.com"
EVOLUTIONAPI_API_KEY="tu-api-key-produccion"
EVOLUTIONAPI_INSTANCE_NAME="EscalaFin-Prod"

# AWS S3 Production
AWS_BUCKET_NAME="escalafin-production-files"
AWS_FOLDER_PREFIX="uploads/"
AWS_ACCESS_KEY_ID="tu-access-key-id"
AWS_SECRET_ACCESS_KEY="tu-secret-access-key"
AWS_REGION="us-east-1"

# Security
NODE_ENV="production"
```

🚀 Despliegue con Vercel (Recomendado)

1. Preparación del Repositorio

```
# Asegurar que el código esté en GitHub
git add .
git commit -m "Preparar para despliegue en Vercel"
git push origin main
```

2. Configurar Vercel

1. Ve a vercel.com (https://vercel.com) y conecta tu GitHub
2. Importa el repositorio `escalafin-mvp`
3. Configura el directorio raíz: `app/`
4. Configura las variables de entorno en el dashboard de Vercel

3. Variables de Entorno en Vercel

```
# Usar la CLI de Vercel para configurar variables
npx vercel env add DATABASE_URL production
npx vercel env add NEXTAUTH_SECRET production
# ... continuar con todas las variables
```

4. Configurar Base de Datos

```
# Ejecutar migraciones en producción
npx prisma db push --accept-data-loss
npx prisma db seed
```

5. Deploy

```
npx vercel --prod
```

🐳 Despliegue con Docker

1. Dockerfile

```
FROM node:18-alpine AS deps
RUN apk add --no-cache libc6-compat
WORKDIR /app
COPY package.json yarn.lock ./
RUN yarn --frozen-lockfile

FROM node:18-alpine AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .
RUN npx prisma generate
RUN yarn build

FROM node:18-alpine AS runner
WORKDIR /app

ENV NODE_ENV production
RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone ./
COPY --from=builder --chown=nextjs:nodejs /app/.next/static ./.next/static

USER nextjs
EXPOSE 3000
ENV PORT 3000

CMD ["node", "server.js"]
```

2. Docker Compose

```yaml
version: '3.8'
services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - DATABASE_URL=${DATABASE_URL}
      - NEXTAUTH_URL=${NEXTAUTH_URL}
      - NEXTAUTH_SECRET=${NEXTAUTH_SECRET}
    depends_on:
      - db

  db:
    image: postgres:14
    environment:
      POSTGRES_DB: escalafin_db
      POSTGRES_USER: escalafin_user
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"

  nginx:
    image: nginx:alpine
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
      - ./ssl:/etc/ssl
    depends_on:
      - app

volumes:
  postgres_data:
```

3. Desplegar con Docker

```bash
# Build y ejecutar
docker-compose up -d

# Ver logs
docker-compose logs -f app

# Ejecutar migraciones
docker-compose exec app npx prisma db push
docker-compose exec app npx prisma db seed
```

☁️ Despliegue en VPS (Ubuntu/Debian)

1. Preparar el Servidor

```
# Actualizar sistema
sudo apt update && sudo apt upgrade -y

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

# Instalar Yarn
npm install -g yarn

# Instalar PostgreSQL
sudo apt install postgresql postgresql-contrib

# Instalar PM2
npm install -g pm2

# Instalar Nginx
sudo apt install nginx
```

2. Configurar PostgreSQL

```
sudo -u postgres psql

CREATE DATABASE escalafin_db;
CREATE USER escalafin_user WITH PASSWORD 'tu_contraseña_segura';
GRANT ALL PRIVILEGES ON DATABASE escalafin_db TO escalafin_user;
\q
```

3. Clonar y Configurar Aplicación

```
cd /var/www
sudo git clone https://github.com/tu-usuario/escalafin-mvp.git
sudo chown -R $USER:$USER escalafin-mvp
cd escalafin-mvp/app

# Instalar dependencias
yarn install

# Configurar variables de entorno
cp .env.example .env
nano .env  # Configurar todas las variables

# Generar Prisma client
yarn prisma generate

# Ejecutar migraciones
yarn prisma db push

# Seed de datos
yarn prisma db seed

# Build para producción
yarn build
```

4. Configurar PM2

```
# ecosystem.config.js
module.exports = {
  apps: [{
    name: 'escalafin-mvp',
    script: './node_modules/.bin/next',
    args: 'start',
    cwd: '/var/www/escalafin-mvp/app',
    instances: 'max',
    exec_mode: 'cluster',
    env: {
      NODE_ENV: 'production',
      PORT: 3000
    }
  }]
}

# Iniciar con PM2
pm2 start ecosystem.config.js
pm2 save
pm2 startup
```

5. Configurar Nginx

```
# /etc/nginx/sites-available/escalafin
server {
    listen 80;
    listen [::]:80;
    server_name tu-dominio.com www.tu-dominio.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name tu-dominio.com www.tu-dominio.com;

    ssl_certificate /path/to/certificate.crt;
    ssl_certificate_key /path/to/private.key;

    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_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;
        proxy_cache_bypass $http_upgrade;
        proxy_redirect off;
    }
}
```

```
# Habilitar sitio
sudo ln -s /etc/nginx/sites-available/escalafin /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

🔐 SSL con Let's Encrypt

```
# Instalar Certbot
sudo apt install certbot python3-certbot-nginx

# Obtener certificado
sudo certbot --nginx -d tu-dominio.com -d www.tu-dominio.com

# Configurar renovación automática
sudo crontab -e
# Agregar: 0 2 * * * /usr/bin/certbot renew --quiet && systemctl reload nginx
```

📊 Configurar Monitoreo

1. PM2 Monitoring

```
# Instalar módulo de monitoreo
pm2 install pm2-logrotate

# Ver métricas
pm2 monit

# Logs
pm2 logs escalafin-mvp
```

2. Configurar Logs

```
# Crear directorio de logs
sudo mkdir /var/log/escalafin
sudo chown $USER:$USER /var/log/escalafin

# Configurar logrotate
sudo nano /etc/logrotate.d/escalafin
```

```
/var/log/escalafin/*.log {
    daily
    missingok
    rotate 14
    compress
    notifempty
    create 644 $USER $USER
    postrotate
        pm2 reload escalafin-mvp
    endscript
}
```

🔄 Script de Despliegue Automático

```
#!/bin/bash
# deploy.sh

set -e

echo "🚀 Iniciando despliegue de EscalaFin MVP..."

# Ir al directorio de la aplicación
cd /var/www/escalafin-mvp

# Hacer backup de la base de datos
echo "📦 Creando backup de base de datos..."
pg_dump escalafin_db > backup_$(date +%Y%m%d_%H%M%S).sql

# Actualizar código
echo "📥 Actualizando código..."
git pull origin main
cd app

# Instalar/actualizar dependencias
echo "📦 Instalando dependencias..."
yarn install

# Ejecutar migraciones
echo "🗃️ Ejecutando migraciones..."
yarn prisma db push

# Build
echo "🏗️ Building aplicación..."
yarn build

# Restart con PM2
echo "🔄 Reiniciando aplicación..."
pm2 reload escalafin-mvp

echo "✅ Despliegue completado exitosamente!"
```

```
chmod +x deploy.sh
./deploy.sh
```

⚡ Optimizaciones de Producción

1. Variables de Entorno para Performance

```
# Next.js optimizations
NEXT_TELEMETRY_DISABLED=1
ANALYZE=true
BUNDLE_ANALYZER=true
```

2. Configuración de Nginx para Caching

```
# Agregar a server block
location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2)$ {
    expires 1y;
    add_header Cache-Control "public, immutable";
}

location ~* \.(html)$ {
    expires 5m;
    add_header Cache-Control "public";
}
```

3. Configurar Compression

```
# Agregar a nginx.conf
gzip on;
gzip_vary on;
gzip_min_length 1024;
gzip_comp_level 6;
gzip_types text/plain text/css text/xml text/javascript application/javascript applic-
ation/json application/xml+rss;
```

🔍 Troubleshooting

Problemas Comunes

1. **Error de conexión a base de datos**
   `bash`
   `# Verificar conexión`
   `psql -h localhost -U escalafin_user -d escalafin_db`

2. **Error de memoria en build**
   `bash`
   `# Aumentar memoria para Node.js`
   `export NODE_OPTIONS="--max-old-space-size=4096"`
   `yarn build`

3. **Problemas con SSL**
   `bash`
   `# Verificar certificado`
   `sudo certbot certificates`

Logs Útiles

```
# Logs de aplicación
pm2 logs escalafin-mvp

# Logs de Nginx
sudo tail -f /var/log/nginx/error.log
sudo tail -f /var/log/nginx/access.log

# Logs de sistema
journalctl -u nginx
journalctl -u postgresql
```

📈 Métricas Post-Despliegue

Health Checks

1. **Endpoint de Health**
   - `GET /api/health` - Verificar estado de la aplicación

2. **Monitoreo de Base de Datos**
   `sql`
   `-- Verificar conexiones activas`
   `SELECT count(*) FROM pg_stat_activity;`

3. **Verificar Servicios Externos**
   - Openpay: Verificar webhook configurado
   - WhatsApp: Verificar instancia activa
   - S3: Verificar permisos de bucket

Performance Monitoring

```
# Instalar herramientas de monitoreo
npm install -g clinic
npm install -g autocannon

# Benchmark
autocannon -c 10 -d 30 https://tu-dominio.com
```

🎉 **¡Felicidades!** EscalaFin MVP está ahora desplegado en producción.

Para cualquier problema durante el despliegue, consulta la documentación de troubleshooting (README.md#-troubleshooting) o crea un issue en GitHub (https://github.com/tu-usuario/escalafin-mvp/issues).