Guía de Deployment - Plataforma de Transparencia Municipal

📋 Requisitos Previos

Sistema

- Node.js 18.x o superior
- PostgreSQL 14.x o superior
- Yarn o npm
- Docker y Docker Compose (opcional)

Variables de Entorno Requeridas

```
# Base de Datos
DATABASE_URL="postgresql://user:password@host:5432/dbname"

# Autenticación
NEXTAUTH_URL="https://tu-dominio.com"
NEXTAUTH_SECRET="tu-secret-seguro-aquí"

# API Backend
NEXT_PUBLIC_API_URL="https://api.tu-dominio.com"

# IA y LLM
ABACUSAI_API_KEY="tu-api-key-aquí"

# AWS S3 (para almacenamiento de archivos)
AWS_REGION="us-west-2"
AWS_BUCKET_NAME="tu-bucket"
AWS_FOLDER_PREFIX="uploads/"
```

🚀 Deployment con Docker

1. Construcción de la Imagen

```
# Desde el directorio raíz del proyecto
docker build -t municipal-platform:latest -f frontend/Dockerfile .
```

2. Ejecutar con Docker Compose

```
# Iniciar todos los servicios
docker-compose up -d

# Ver logs
docker-compose logs -f

# Detener servicios
docker-compose down
```

3. Docker Compose Configuración

```yaml
version: '3.8'

services:
  postgres:
    image: postgres:14-alpine
    environment:
      POSTGRES_DB: municipal_db
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"

  backend:
    build:
      context: ./apps/api
      dockerfile: Dockerfile
    environment:
      DATABASE_URL: postgresql://admin:${DB_PASSWORD}@postgres:5432/municipal_db
      PORT: 3000
    ports:
      - "3000:3000"
    depends_on:
      - postgres

  frontend:
    build:
      context: ./frontend/nextjs_space
      dockerfile: Dockerfile
    environment:
      NEXT_PUBLIC_API_URL: http://backend:3000
      DATABASE_URL: postgresql://admin:${DB_PASSWORD}@postgres:5432/municipal_db
      NEXTAUTH_URL: ${NEXTAUTH_URL}
      NEXTAUTH_SECRET: ${NEXTAUTH_SECRET}
    ports:
      - "80:3000"
    depends_on:
      - backend

volumes:
  postgres_data:
```

🔧 Deployment Manual

Frontend (Next.js)

```bash
# 1. Instalar dependencias
cd frontend/nextjs_space
yarn install

# 2. Construir la aplicación
yarn build

# 3. Iniciar en producción
yarn start
```

Backend (NestJS)

```
# 1. Instalar dependencias
cd apps/api
npm install

# 2. Ejecutar migraciones
npm run migrate

# 3. Iniciar en producción
npm run start:prod
```

☁️ Deployment en Plataformas Cloud

Vercel (Recomendado para Frontend)

1. Conecta tu repositorio de GitHub
2. Configura las variables de entorno
3. Vercel detectará automáticamente Next.js
4. Deploy automático con cada push

```
# O usando Vercel CLI
vercel --prod
```

Heroku (Backend)

```
# 1. Login
heroku login

# 2. Crear aplicación
heroku create municipal-platform-api

# 3. Configurar variables
heroku config:set DATABASE_URL=postgresql://...

# 4. Deploy
git push heroku main
```

AWS (Producción Completa)

Frontend (S3 + CloudFront)

```
# 1. Build estático
yarn build && yarn export

# 2. Subir a S3
aws s3 sync out/ s3://tu-bucket/ --delete

# 3. Invalidar CloudFront
aws cloudfront create-invalidation --distribution-id XXX --paths "/*"
```

Backend (EC2 o ECS)

- Usar AMI optimizada para Node.js
- Configurar Auto Scaling Group

- Load Balancer (ALB)
- RDS para PostgreSQL

🔐 Configuración de Seguridad

SSL/TLS

```
# Usando Certbot para Let's Encrypt
sudo certbot --nginx -d tu-dominio.com
```

Firewall

```
# UFW (Ubuntu)
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw allow 22/tcp
sudo ufw enable
```

Rate Limiting

```
// Configurar en next.config.js
module.exports = {
  async headers() {
    return [
      {
        source: '/api/:path*',
        headers: [
          {
            key: 'X-RateLimit-Limit',
            value: '100'
          }
        ]
      }
    ]
  }
}
```

📊 Monitoreo y Logs

PM2 (Process Manager)

```
# Instalar PM2
npm install -g pm2

# Iniciar aplicación
pm2 start npm --name "municipal-frontend" -- start

# Ver logs
pm2 logs

# Monitoreo
pm2 monit

# Auto-restart en reboot
pm2 startup
pm2 save
```

Logs Centralizados

- Usar servicios como:
- Datadog
- New Relic
- Sentry (para errores)
- CloudWatch (AWS)

🔄 CI/CD Pipeline

GitHub Actions

```yaml
name: Deploy Production

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Setup Node
        uses: actions/setup-node@v2
        with:
          node-version: '18'

      - name: Install dependencies
        run: yarn install

      - name: Run tests
        run: yarn test

      - name: Build
        run: yarn build

      - name: Deploy to Vercel
        uses: amondnet/vercel-action@v20
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}
          vercel-org-id: ${{ secrets.ORG_ID }}
          vercel-project-id: ${{ secrets.PROJECT_ID }}
```

🧪 Testing Pre-Deployment

```
# Tests unitarios
yarn test

# Tests E2E
yarn test:e2e

# Lighthouse (Performance)
lighthouse https://tu-dominio.com --view

# Security audit
npm audit
```

📝 Checklist de Deployment

- [ ] Variables de entorno configuradas
- [ ] Base de datos con backup habilitado

- [ ] SSL/TLS certificado instalado
- [ ] Firewall configurado
- [ ] Logs centralizados
- [ ] Monitoreo activo
- [ ] Backups automáticos
- [ ] Rate limiting configurado
- [ ] CDN configurado (si aplica)
- [ ] DNS apuntando correctamente
- [ ] Tests pasando
- [ ] Documentación actualizada

🆘 Troubleshooting

Problema: Error de conexión a base de datos

```
# Verificar conexión
psql -h host -U user -d database

# Verificar variables de entorno
echo $DATABASE_URL
```

Problema: Build falla

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

Problema: Performance lento

- Verificar queries N+1 en backend
- Implementar caching con Redis
- Optimizar imágenes con Next.js Image
- Habilitar Incremental Static Regeneration

📞 Soporte

Para problemas o preguntas:
- Email: soporte@municipal-platform.cl
- Documentación: https://docs.municipal-platform.cl
- Issues: GitHub Issues