Configuración de Despliegue y Producción

Variables de Entorno para Producción

Archivo .env.production

```
# Modo de ejecución
NODE_ENV=production

# Configuración del servidor
PORT=3000
HOST=0.0.0.0

# Configuración ATV (Producción Real)
ATV_MODE=REAL
ATV_KEY_PATH=/path/to/production/key.p12
ATV_CERT_PATH=/path/to/production/cert.crt
ATV_KEY_PASSWORD=tu_password_seguro

# URLs de Hacienda (Producción)
HACIENDA_API_URL=https://api.comprobanteselectronicos.go.cr
HACIENDA_TOKEN_URL=https://idp.comprobanteselectronicos.go.cr

# Configuración de logging
LOG_LEVEL=info
LOG_FILE_PATH=/var/log/hacienda-api/app.log

# Configuración de archivos
INVOICES_BASE_DIR=/data/invoices
MAX_FILE_SIZE=10485760
MAX_FILES_PER_DIR=10000

# Configuración de seguridad
RATE_LIMIT_WINDOW=900000
RATE_LIMIT_MAX=100
CORS_ORIGIN=https://tu-frontend.com

# Configuración de base de datos (opcional)
# DATABASE_URL=postgresql://user:pass@localhost:5432/hacienda_db
# REDIS_URL=redis://localhost:6379

# Configuración de monitoreo
HEALTH_CHECK_INTERVAL=30000
METRICS_ENABLED=true
```

Docker

Dockerfile

```dockerfile
FROM node:18-alpine AS base

# Instalar dependencias del sistema
RUN apk add --no-cache \
    ca-certificates \
    tzdata

# Configurar zona horaria
ENV TZ=America/Costa_Rica
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

# Crear usuario no-root
RUN addgroup -g 1001 -S nodejs
RUN adduser -S hacienda -u 1001

# Stage de dependencias
FROM base AS deps
WORKDIR /app

# Copiar archivos de configuración
COPY package*.json ./
COPY .npmrc* ./

# Instalar dependencias de producción
RUN npm ci --only=production --frozen-lockfile && npm cache clean --force

# Stage de construcción
FROM base AS builder
WORKDIR /app

# Copiar dependencias
COPY --from=deps /app/node_modules ./node_modules
COPY . .

# Ejecutar tests
RUN npm test

# Stage de producción
FROM base AS runner
WORKDIR /app

ENV NODE_ENV=production

# Crear directorios necesarios
RUN mkdir -p /app/invoices /app/logs /app/temp
RUN chown -R hacienda:nodejs /app

# Copiar archivos necesarios
COPY --from=deps --chown=hacienda:nodejs /app/node_modules ./node_modules
COPY --from=builder --chown=hacienda:nodejs /app/src ./src
COPY --from=builder --chown=hacienda:nodejs /app/package*.json ./
COPY --from=builder --chown=hacienda:nodejs /app/docs ./docs
COPY --from=builder --chown=hacienda:nodejs /app/examples ./examples

# Cambiar al usuario no-root
USER hacienda

# Exponer puerto
EXPOSE 3000

# Health check
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
    CMD node -e "require('http').get('http://localhost:3000/health', (res) => {
process.exit(res.statusCode === 200 ? 0 : 1) })"

# Comando de inicio
CMD ["node", "src/server.js"]
```

docker-compose.yml para Producción

```
version: '3.8'

services:
  hacienda-api:
    image: hacienda-api:latest
    build: .
    container_name: hacienda-api-prod
    restart: unless-stopped
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
    env_file:
      - .env.production
    volumes:
      # Datos persistentes
      - ./data/invoices:/app/invoices:rw
      - ./data/logs:/app/logs:rw
      # Certificados (solo lectura)
      - ./certs:/app/certs:ro
    networks:
      - hacienda-network
    depends_on:
      - redis
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.hacienda-api.rule=Host(`api.tu-dominio.com`)"
      - "traefik.http.routers.hacienda-api.tls=true"
      - "traefik.http.routers.hacienda-api.tls.certresolver=letsencrypt"

  redis:
    image: redis:7-alpine
    container_name: hacienda-redis
    restart: unless-stopped
    ports:
      - "127.0.0.1:6379:6379"
    volumes:
      - redis-data:/data
    command: redis-server --appendonly yes --requirepass ${REDIS_PASSWORD}
    networks:
      - hacienda-network

  nginx:
    image: nginx:alpine
    container_name: hacienda-nginx
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - ./nginx/ssl:/etc/nginx/ssl:ro
    networks:
      - hacienda-network
    depends_on:
      - hacienda-api

volumes:
  redis-data:

networks:
  hacienda-network:
    driver: bridge
```

Nginx (Proxy Reverso)

nginx.conf

```
events {
    worker_connections 1024;
}

http {
    upstream hacienda_api {
        server hacienda-api:3000;
    }

    # Rate limiting
    limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s;
    limit_req_zone $binary_remote_addr zone=upload:10m rate=2r/s;

    server {
```

```
    listen 80;
    server_name api.tu-dominio.com;

    # Redirect to HTTPS
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name api.tu-dominio.com;

    # SSL Configuration
    ssl_certificate /etc/nginx/ssl/cert.pem;
    ssl_certificate_key /etc/nginx/ssl/key.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512;
    ssl_prefer_server_ciphers off;

    # Security headers
    add_header X-Frame-Options DENY always;
    add_header X-Content-Type-Options nosniff always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;

    # Body size limit
    client_max_body_size 10M;

    # API endpoints
    location /api/ {
        limit_req zone=api burst=20 nodelay;

        proxy_pass http://hacienda_api;
        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;

        # Timeouts
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # Upload endpoints with different rate limit
    location /api/facturas/emitir {
        limit_req zone=upload burst=5 nodelay;

        proxy_pass http://hacienda_api;
```

```
            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;
        }

        # Health check (no rate limit)
        location /health {
            proxy_pass http://hacienda_api;
            access_log off;
        }

        # Documentation
        location / {
            proxy_pass http://hacienda_api;
            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;
        }
    }
}
```

Scripts de Despliegue

deploy.sh

```
#!/bin/bash

# Script de despliegue para producción

set -e

# Configuración
APP_NAME="hacienda-api"
DOCKER_IMAGE="$APP_NAME:latest"
CONTAINER_NAME="$APP_NAME-prod"
BACKUP_DIR="/backup/hacienda-api"
DATE=$(date +%Y%m%d_%H%M%S)

echo "🚀 Iniciando despliegue de $APP_NAME..."

# Crear backup de datos actuales
echo "📦 Creando backup..."
mkdir -p "$BACKUP_DIR"
if [ -d "./data" ]; then
    tar -czf "$BACKUP_DIR/data_backup_$DATE.tar.gz" ./data
    echo "✅ Backup creado: data_backup_$DATE.tar.gz"
fi
```

```bash
# Verificar archivos necesarios
echo "🔍 Verificando configuración..."
if [ ! -f ".env.production" ]; then
    echo "❌ Error: .env.production no encontrado"
    exit 1
fi

if [ ! -f "docker-compose.yml" ]; then
    echo "❌ Error: docker-compose.yml no encontrado"
    exit 1
fi

# Construir imagen Docker
echo "🏗 Construyendo imagen Docker..."
docker build -t "$DOCKER_IMAGE" .

# Ejecutar tests en contenedor
echo "🧪 Ejecutando tests..."
docker run --rm "$DOCKER_IMAGE" npm test

# Detener contenedor actual (si existe)
echo "🛑 Deteniendo contenedor actual..."
docker-compose -f docker-compose.yml down || true

# Iniciar nuevos servicios
echo "🚀 Iniciando servicios..."
docker-compose -f docker-compose.yml up -d

# Verificar que los servicios estén funcionando
echo "🔍 Verificando servicios..."
sleep 10

# Health check
HEALTH_URL="http://localhost:3000/health"
if curl -f -s "$HEALTH_URL" > /dev/null; then
    echo "✅ Servicios iniciados correctamente"
else
    echo "❌ Error: Servicios no responden"
    echo "📋 Logs del contenedor:"
    docker logs "$CONTAINER_NAME"
    exit 1
fi

# Limpiar imágenes antiguas
echo "🧹 Limpiando imágenes antiguas..."
docker image prune -f

echo "✅ Despliegue completado exitosamente!"
echo "🌐 API disponible en: http://localhost:3000"
echo "📚 Documentación en: http://localhost:3000/docs"
```

monitoring.sh

```bash
#!/bin/bash

# Script de monitoreo para producción

CONTAINER_NAME="hacienda-api-prod"
LOG_FILE="/var/log/hacienda-api-monitor.log"
WEBHOOK_URL="" # URL de Slack/Discord para notificaciones

log_message() {
    echo "[$(date)] $1" | tee -a "$LOG_FILE"
}

send_alert() {
    local message="$1"
    log_message "ALERT: $message"

    if [ -n "$WEBHOOK_URL" ]; then
        curl -X POST -H 'Content-type: application/json' \
            --data "{\"text\":\"🚨 Hacienda API Alert: $message\"}" \
            "$WEBHOOK_URL"
    fi
}

check_container_health() {
    if ! docker ps | grep -q "$CONTAINER_NAME"; then
        send_alert "Contenedor $CONTAINER_NAME no está ejecutándose"
        return 1
    fi

    # Health check HTTP
    if ! curl -f -s "http://localhost:3000/health" > /dev/null; then
        send_alert "Health check falló para $CONTAINER_NAME"
        return 1
    fi

    return 0
}

check_disk_space() {
    local usage=$(df /data | awk 'NR==2 {print $5}' | sed 's/%//')
    if [ "$usage" -gt 85 ]; then
        send_alert "Uso de disco alto: ${usage}%"
    fi
}

check_memory_usage() {
    local usage=$(docker stats "$CONTAINER_NAME" --no-stream --format "{{.MemPerc}}" | sed
's/%//')
    if [ "$usage" -gt 80 ]; then
```

```
        send_alert "Uso de memoria alto en $CONTAINER_NAME: ${usage}%"
    fi
}

# Ejecutar verificaciones
log_message "Iniciando monitoreo..."

if check_container_health; then
    log_message "Container health: OK"
    check_disk_space
    check_memory_usage
else
    log_message "Container health: FAILED"

    # Intentar reiniciar
    log_message "Intentando reiniciar contenedor..."
    docker-compose restart "$CONTAINER_NAME"

    sleep 30

    if check_container_health; then
        send_alert "Contenedor reiniciado exitosamente"
    else
        send_alert "Fallo al reiniciar contenedor - Intervención manual requerida"
    fi
fi

log_message "Monitoreo completado"
```

Configuración de Systemd

hacienda-api.service

```
[Unit]
Description=Hacienda API Service
Requires=docker.service
After=docker.service

[Service]
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=/opt/hacienda-api
ExecStart=/usr/local/bin/docker-compose up -d
ExecStop=/usr/local/bin/docker-compose down
TimeoutStartSec=0

[Install]
WantedBy=multi-user.target
```

Certificados SSL

Estructura de certificados

```
certs/
├── production/
│   ├── hacienda.p12          # Certificado para firmar documentos
│   ├── hacienda.crt          # Certificado público
│   └── private.key           # Llave privada
├── staging/
│   ├── hacienda-test.p12
│   ├── hacienda-test.crt
│   └── private-test.key
└── ssl/
    ├── fullchain.pem         # Certificado SSL para HTTPS
    └── privkey.pem           # Llave privada SSL
```

Backup y Recuperación

backup-script.sh

```bash
#!/bin/bash

# Script de backup automatizado

BACKUP_DIR="/backup/hacienda-api"
DATA_DIR="/opt/hacienda-api/data"
RETENTION_DAYS=30
DATE=$(date +%Y%m%d_%H%M%S)

# Crear directorio de backup
mkdir -p "$BACKUP_DIR"

# Backup de datos
echo "Creando backup de datos..."
tar -czf "$BACKUP_DIR/invoices_$DATE.tar.gz" "$DATA_DIR/invoices"
tar -czf "$BACKUP_DIR/logs_$DATE.tar.gz" "$DATA_DIR/logs"

# Backup de configuración
echo "Creando backup de configuración..."
tar -czf "$BACKUP_DIR/config_$DATE.tar.gz" \
    /opt/hacienda-api/.env.production \
    /opt/hacienda-api/docker-compose.yml \
    /opt/hacienda-api/nginx/

# Limpiar backups antiguos
echo "Limpiando backups antiguos..."
find "$BACKUP_DIR" -name "*.tar.gz" -mtime +$RETENTION_DAYS -delete
```

```
echo "Backup completado: $DATE"
```

Configuración de Cron

```
# Ejecutar cada 6 horas
0 */6 * * * /opt/hacienda-api/scripts/backup-script.sh

# Monitoreo cada 5 minutos
*/5 * * * * /opt/hacienda-api/scripts/monitoring.sh

# Limpieza de logs cada día a las 2 AM
0 2 * * * find /opt/hacienda-api/data/logs -name "*.log" -mtime +7 -delete

# Reinicio semanal (domingo 3 AM)
0 3 * * 0 /opt/hacienda-api/scripts/weekly-restart.sh
```

Consideraciones de Seguridad

1. Firewall (UFW)

```
# Permitir solo puertos necesarios
ufw allow 22/tcp   # SSH
ufw allow 80/tcp   # HTTP (redirect a HTTPS)
ufw allow 443/tcp  # HTTPS
ufw deny 3000/tcp  # Bloquear acceso directo a la API
ufw enable
```

2. Fail2Ban

```
# /etc/fail2ban/jail.local
[DEFAULT]
bantime = 3600
findtime = 600
maxretry = 5

[nginx-req-limit]
enabled = true
filter = nginx-req-limit
action = iptables-multiport[name=ReqLimit, port="http,https", protocol=tcp]
logpath = /var/log/nginx/error.log
maxretry = 10
findtime = 600
bantime = 7200
```

3. Rotación de Logs

```
# /etc/logrotate.d/hacienda-api
/opt/hacienda-api/data/logs/*.log {
    daily
    missingok
    rotate 30
    compress
    notifempty
    create 644 hacienda hacienda
    postrotate
        docker kill -s USR1 hacienda-api-prod
    endscript
}
```