# Manual de Instalación - MediTrack

## Sistema de Gestión Médica

**Versión:** 1.0  
**Fecha:** Primer semestre de 2025  
**Proyecto:** MediTrack - Sistema de Gestión Médica  
**Tecnologías:** Laravel 12, React 19, TypeScript, MySQL 8.0

## 1. Introducción

### 1.1 Propósito del Documento

Este manual proporciona instrucciones detalladas para la instalación, configuración y despliegue del sistema MediTrack en diferentes entornos, desde desarrollo local hasta producción.

### 1.2 Arquitectura del Sistema

MediTrack es una aplicación web moderna construida con:

* **Backend:** Laravel 12 (PHP 8.2+)
* **Frontend:** React 19 con TypeScript
* **Bridge:** Inertia.js para SPA sin API
* **Base de Datos:** MySQL 8.0
* **Build Tool:** Vite 6.0
* **Styling:** Tailwind CSS 4.0
* **Containerización:** Docker y Docker Compose
* **Testing:** Pest (PHPUnit)

### 1.3 Entornos de Instalación

Este manual cubre tres tipos de instalación:

1. **Desarrollo Local:** Para desarrollo y testing
2. **Staging:** Para pruebas pre-producción
3. **Producción:** Para ambiente live

## 2. Requisitos del Sistema

### 2.1 Requisitos Mínimos

#### Servidor/Local

* **Sistema Operativo:** Linux (Ubuntu 22.04+ recomendado), Windows 10+, macOS 12+
* **RAM:** 4GB mínimo, 8GB recomendado
* **Espacio en Disco:** 10GB mínimo
* **CPU:** 2 cores mínimo

#### Software Base

* **PHP:** 8.2 o superior
* **Node.js:** 22.x LTS
* **MySQL:** 8.0 o superior
* **Composer:** 2.x
* **Git:** 2.x

### 2.2 Extensiones PHP Requeridas

# Extensiones obligatorias  
php8.2-cli  
php8.2-mysql  
php8.2-mbstring  
php8.2-xml  
php8.2-zip  
php8.2-bcmath  
php8.2-curl  
php8.2-gd  
php8.2-intl  
php8.2-soap  
php8.2-readline  
  
# Extensiones opcionales para mejor rendimiento  
php8.2-redis  
php8.2-memcached  
php8.2-opcache

### 2.3 Configuración de Sistema

#### MySQL

* **Versión:** 8.0+
* **Charset:** UTF8MB4
* **Storage Engine:** InnoDB
* **Memory:** 1GB mínimo

#### PHP

* **Memory Limit:** 256MB mínimo
* **Max Execution Time:** 300 segundos
* **Upload Max Size:** 50MB
* **Post Max Size:** 50MB

## 3. Instalación con Docker (Recomendado)

### 3.1 Prerrequisitos Docker

# Instalar Docker Engine  
curl -fsSL https://get.docker.com -o get-docker.sh  
sudo sh get-docker.sh  
  
# Instalar Docker Compose  
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose  
sudo chmod +x /usr/local/bin/docker-compose  
  
# Verificar instalación  
docker --version  
docker-compose --version

### 3.2 Clonar Repositorio

# Clonar proyecto  
git clone https://github.com/tu-organizacion/meditrack.git  
cd meditrack  
  
# Verificar rama  
git branch -a  
git checkout main # o la rama apropiada

### 3.3 Configuración Inicial

#### Configurar Variables de Entorno

# Copiar archivo de configuración  
cp .env.example .env  
  
# Editar configuración para Docker  
nano .env

**Configuración .env para Docker:**

# Aplicación  
APP\_NAME=MediTrack  
APP\_ENV=local  
APP\_KEY=  
APP\_DEBUG=true  
APP\_URL=http://localhost  
  
# Base de datos (Docker)  
DB\_CONNECTION=mysql  
DB\_HOST=mysql  
DB\_PORT=3306  
DB\_DATABASE=meditrack  
DB\_USERNAME=meditrack\_user  
DB\_PASSWORD=meditrack\_password  
DB\_ROOT\_PASSWORD=root\_password  
  
# Configuración de sesión  
SESSION\_DRIVER=database  
SESSION\_LIFETIME=120  
  
# Cache  
CACHE\_DRIVER=redis  
REDIS\_HOST=redis  
REDIS\_PASSWORD=null  
REDIS\_PORT=6379  
  
# Queue  
QUEUE\_CONNECTION=redis  
  
# Mail (configurar según necesidades)  
MAIL\_MAILER=smtp  
MAIL\_HOST=smtp.gmail.com  
MAIL\_PORT=587  
MAIL\_USERNAME=tu-email@gmail.com  
MAIL\_PASSWORD=tu-password  
MAIL\_ENCRYPTION=tls  
MAIL\_FROM\_ADDRESS=noreply@meditrack.com  
MAIL\_FROM\_NAME="MediTrack"

### 3.4 Construcción e Inicio

# Construir contenedores  
docker-compose up --build -d  
  
# Verificar que los servicios estén corriendo  
docker-compose ps  
  
# Ver logs si hay problemas  
docker-compose logs -f

### 3.5 Configuración de la Aplicación

# Acceder al contenedor de la aplicación  
docker-compose exec app bash  
  
# Generar key de aplicación  
php artisan key:generate  
  
# Ejecutar migraciones  
php artisan migrate  
  
# Seedear datos iniciales  
php artisan db:seed  
  
# Limpiar y optimizar cache  
php artisan config:cache  
php artisan route:cache  
php artisan view:cache  
  
# Salir del contenedor  
exit

### 3.6 Instalación de Frontend

# Instalar dependencias de Node.js  
docker-compose exec app npm install  
  
# Construir assets para desarrollo  
docker-compose exec app npm run dev  
  
# O para producción  
docker-compose exec app npm run build

## 4. Instalación Local (Sin Docker)

### 4.1 Instalación de Dependencias

#### Ubuntu/Debian

# Actualizar sistema  
sudo apt update && sudo apt upgrade -y  
  
# Instalar PHP y extensiones  
sudo apt install -y php8.2 php8.2-cli php8.2-mysql php8.2-mbstring php8.2-xml php8.2-zip php8.2-bcmath php8.2-curl php8.2-gd php8.2-intl php8.2-soap php8.2-readline  
  
# Instalar Composer  
curl -sS https://getcomposer.org/installer | php  
sudo mv composer.phar /usr/local/bin/composer  
sudo chmod +x /usr/local/bin/composer  
  
# Instalar Node.js  
curl -fsSL https://deb.nodesource.com/setup\_22.x | sudo -E bash -  
sudo apt-get install -y nodejs  
  
# Instalar MySQL  
sudo apt install -y mysql-server mysql-client  
sudo mysql\_secure\_installation

#### macOS

# Instalar Homebrew (si no está instalado)  
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"  
  
# Instalar PHP  
brew install php@8.2  
brew link php@8.2  
  
# Instalar Composer  
brew install composer  
  
# Instalar Node.js  
brew install node@22  
brew link node@22  
  
# Instalar MySQL  
brew install mysql  
brew services start mysql

#### Windows

# Instalar Chocolatey (ejecutar como Administrador)  
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))  
  
# Instalar dependencias  
choco install php composer nodejs mysql  
  
# Configurar PHP  
# Editar php.ini para habilitar extensiones necesarias

### 4.2 Configuración de Base de Datos

-- Conectar a MySQL como root  
mysql -u root -p  
  
-- Crear base de datos  
CREATE DATABASE meditrack CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;  
  
-- Crear usuario  
CREATE USER 'meditrack\_user'@'localhost' IDENTIFIED BY 'meditrack\_password';  
  
-- Otorgar permisos  
GRANT ALL PRIVILEGES ON meditrack.\* TO 'meditrack\_user'@'localhost';  
FLUSH PRIVILEGES;  
  
-- Verificar  
SHOW DATABASES;  
SELECT User, Host FROM mysql.user WHERE User = 'meditrack\_user';  
  
-- Salir  
EXIT;

### 4.3 Configuración del Proyecto

# Clonar repositorio  
git clone https://github.com/tu-organizacion/meditrack.git  
cd meditrack  
  
# Instalar dependencias PHP  
composer install  
  
# Instalar dependencias Node.js  
npm install  
  
# Configurar variables de entorno  
cp .env.example .env

**Configuración .env para instalación local:**

APP\_NAME=MediTrack  
APP\_ENV=local  
APP\_KEY=  
APP\_DEBUG=true  
APP\_URL=http://localhost:8000  
  
DB\_CONNECTION=mysql  
DB\_HOST=127.0.0.1  
DB\_PORT=3306  
DB\_DATABASE=meditrack  
DB\_USERNAME=meditrack\_user  
DB\_PASSWORD=meditrack\_password  
  
SESSION\_DRIVER=database  
SESSION\_LIFETIME=120  
  
CACHE\_DRIVER=file  
QUEUE\_CONNECTION=sync  
  
MAIL\_MAILER=log

### 4.4 Inicialización de la Aplicación

# Generar key de aplicación  
php artisan key:generate  
  
# Ejecutar migraciones  
php artisan migrate  
  
# Seedear datos  
php artisan db:seed  
  
# Construir assets  
npm run build  
  
# Iniciar servidor de desarrollo  
php artisan serve  
  
# En otra terminal, iniciar Vite para desarrollo  
npm run dev

## 5. Configuración de Producción

### 5.1 Optimizaciones de PHP

#### php.ini Producción

; Configuraciones de rendimiento  
opcache.enable=1  
opcache.memory\_consumption=256  
opcache.max\_accelerated\_files=20000  
opcache.validate\_timestamps=0  
opcache.save\_comments=1  
opcache.fast\_shutdown=1  
  
; Configuraciones de seguridad  
expose\_php=Off  
display\_errors=Off  
log\_errors=On  
error\_log=/var/log/php\_errors.log  
  
; Configuraciones de memoria  
memory\_limit=512M  
max\_execution\_time=60  
max\_input\_time=60  
upload\_max\_filesize=50M  
post\_max\_size=50M

### 5.2 Configuración de Nginx

server {  
 listen 80;  
 listen [::]:80;  
 server\_name meditrack.com www.meditrack.com;  
   
 # Redirigir a HTTPS  
 return 301 https://$server\_name$request\_uri;  
}  
  
server {  
 listen 443 ssl http2;  
 listen [::]:443 ssl http2;  
 server\_name meditrack.com www.meditrack.com;  
   
 root /var/www/html/public;  
 index index.php;  
   
 # SSL Configuration  
 ssl\_certificate /path/to/certificate.pem;  
 ssl\_certificate\_key /path/to/private.key;  
 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 "SAMEORIGIN" always;  
 add\_header X-XSS-Protection "1; mode=block" always;  
 add\_header X-Content-Type-Options "nosniff" always;  
 add\_header Referrer-Policy "no-referrer-when-downgrade" always;  
 add\_header Content-Security-Policy "default-src 'self' http: https: data: blob: 'unsafe-inline'" always;  
 add\_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload" always;  
   
 # Gzip  
 gzip on;  
 gzip\_vary on;  
 gzip\_min\_length 1024;  
 gzip\_proxied expired no-cache no-store private must-revalidate auth;  
 gzip\_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript;  
   
 location / {  
 try\_files $uri $uri/ /index.php?$query\_string;  
 }  
   
 location = /favicon.ico { access\_log off; log\_not\_found off; }  
 location = /robots.txt { access\_log off; log\_not\_found off; }  
   
 error\_page 404 /index.php;  
   
 location ~ \.php$ {  
 fastcgi\_pass unix:/var/run/php/php8.2-fpm.sock;  
 fastcgi\_param SCRIPT\_FILENAME $realpath\_root$fastcgi\_script\_name;  
 include fastcgi\_params;  
 fastcgi\_hide\_header X-Powered-By;  
 }  
   
 location ~ /\.(?!well-known).\* {  
 deny all;  
 }  
   
 # Static files caching  
 location ~\* \.(jpg|jpeg|png|gif|ico|css|js)$ {  
 expires 1y;  
 add\_header Cache-Control "public, immutable";  
 }  
}

### 5.3 Variables de Entorno de Producción

APP\_NAME=MediTrack  
APP\_ENV=production  
APP\_KEY=base64:TU\_CLAVE\_GENERADA\_AQUI  
APP\_DEBUG=false  
APP\_URL=https://meditrack.com  
  
DB\_CONNECTION=mysql  
DB\_HOST=127.0.0.1  
DB\_PORT=3306  
DB\_DATABASE=meditrack\_prod  
DB\_USERNAME=meditrack\_prod\_user  
DB\_PASSWORD=CONTRASEÑA\_SEGURA\_AQUI  
  
BROADCAST\_DRIVER=log  
CACHE\_DRIVER=redis  
FILESYSTEM\_DISK=local  
QUEUE\_CONNECTION=redis  
SESSION\_DRIVER=redis  
SESSION\_LIFETIME=120  
  
REDIS\_HOST=127.0.0.1  
REDIS\_PASSWORD=null  
REDIS\_PORT=6379  
  
MAIL\_MAILER=smtp  
MAIL\_HOST=smtp.mailgun.org  
MAIL\_PORT=587  
MAIL\_USERNAME=tu\_usuario\_mailgun  
MAIL\_PASSWORD=tu\_password\_mailgun  
MAIL\_ENCRYPTION=tls  
MAIL\_FROM\_ADDRESS="noreply@meditrack.com"  
MAIL\_FROM\_NAME="MediTrack"  
  
# Logging  
LOG\_CHANNEL=stack  
LOG\_LEVEL=error  
  
# Configuración de aplicación médica  
MEDITRACK\_AUDIT\_ENABLED=true  
MEDITRACK\_ENCRYPTION\_KEY=TU\_CLAVE\_DE\_ENCRIPTACION  
MEDITRACK\_BACKUP\_ENABLED=true  
MEDITRACK\_BACKUP\_FREQUENCY=daily

### 5.4 Comandos de Optimización

# Optimizaciones de Laravel  
php artisan config:cache  
php artisan route:cache  
php artisan view:cache  
php artisan event:cache  
  
# Optimización de Composer  
composer install --optimize-autoloader --no-dev  
  
# Construir assets para producción  
npm run build  
  
# Limpiar caches de desarrollo  
php artisan cache:clear  
php artisan config:clear  
php artisan route:clear  
php artisan view:clear  
  
# Configurar permisos  
sudo chown -R www-data:www-data /var/www/html  
sudo chmod -R 755 /var/www/html  
sudo chmod -R 775 /var/www/html/storage  
sudo chmod -R 775 /var/www/html/bootstrap/cache

## 6. Automatización de Despliegue

### 6.1 Script de Despliegue Básico

#!/bin/bash  
# deploy.sh - Script de despliegue para MediTrack  
  
set -e  
  
echo "Iniciando despliegue de MediTrack..."  
  
# Variables  
APP\_DIR="/var/www/html"  
BACKUP\_DIR="/var/backups/meditrack"  
TIMESTAMP=$(date +%Y%m%d\_%H%M%S)  
  
# Crear backup  
echo "Creando backup..."  
mkdir -p $BACKUP\_DIR  
mysqldump --single-transaction meditrack\_prod > $BACKUP\_DIR/meditrack\_backup\_$TIMESTAMP.sql  
tar -czf $BACKUP\_DIR/files\_backup\_$TIMESTAMP.tar.gz -C $APP\_DIR storage  
  
# Activar modo de mantenimiento  
php artisan down --message="Actualizando sistema. Regresamos en breve."  
  
# Actualizar código  
echo "Actualizando código..."  
git fetch origin  
git reset --hard origin/main  
  
# Instalar dependencias  
echo "Instalando dependencias..."  
composer install --no-dev --optimize-autoloader  
npm ci  
npm run build  
  
# Ejecutar migraciones  
echo "Ejecutando migraciones..."  
php artisan migrate --force  
  
# Optimizar aplicación  
echo "Optimizando aplicación..."  
php artisan config:cache  
php artisan route:cache  
php artisan view:cache  
php artisan event:cache  
  
# Reactivar aplicación  
php artisan up  
  
echo "Despliegue completado exitosamente!"

### 6.2 GitHub Actions CI/CD

# .github/workflows/deploy.yml  
name: Deploy to Production  
  
on:  
 push:  
 branches: [ main ]  
  
jobs:  
 test:  
 runs-on: ubuntu-latest  
   
 services:  
 mysql:  
 image: mysql:8.0  
 env:  
 MYSQL\_ROOT\_PASSWORD: password  
 MYSQL\_DATABASE: meditrack\_test  
 ports:  
 - 3306:3306  
 options: --health-cmd="mysqladmin ping" --health-interval=10s --health-timeout=5s --health-retries=3  
   
 steps:  
 - uses: actions/checkout@v4  
   
 - name: Setup PHP  
 uses: shivammathur/setup-php@v2  
 with:  
 php-version: 8.2  
 extensions: mysql, mbstring, xml, zip, bcmath, curl, gd  
   
 - name: Setup Node.js  
 uses: actions/setup-node@v4  
 with:  
 node-version: '22'  
 cache: 'npm'  
   
 - name: Install PHP dependencies  
 run: composer install --no-progress --prefer-dist --optimize-autoloader  
   
 - name: Install Node.js dependencies  
 run: npm ci  
   
 - name: Build assets  
 run: npm run build  
   
 - name: Prepare Laravel  
 run: |  
 cp .env.example .env  
 php artisan key:generate  
 php artisan migrate --force  
 php artisan db:seed  
   
 - name: Run tests  
 run: vendor/bin/pest  
   
 deploy:  
 needs: test  
 runs-on: ubuntu-latest  
 if: github.ref == 'refs/heads/main'  
   
 steps:  
 - name: Deploy to server  
 uses: appleboy/ssh-action@v0.1.5  
 with:  
 host: ${{ secrets.HOST }}  
 username: ${{ secrets.USERNAME }}  
 key: ${{ secrets.SSH\_KEY }}  
 script: |  
 cd /var/www/html  
 sudo -u www-data git pull origin main  
 sudo -u www-data composer install --no-dev --optimize-autoloader  
 sudo -u www-data npm ci  
 sudo -u www-data npm run build  
 sudo -u www-data php artisan migrate --force  
 sudo -u www-data php artisan config:cache  
 sudo -u www-data php artisan route:cache  
 sudo -u www-data php artisan view:cache  
 sudo service php8.2-fpm reload  
 sudo service nginx reload

## 7. Verificación de Instalación

### 7.1 Tests de Funcionalidad

# Ejecutar suite completa de tests  
vendor/bin/pest  
  
# Tests específicos  
vendor/bin/pest --filter=AuthenticationTest  
vendor/bin/pest --filter=UserManagementTest  
vendor/bin/pest --filter=MedicationTest  
  
# Tests con coverage  
vendor/bin/pest --coverage  
  
# Tests de integración  
vendor/bin/pest tests/Feature/

### 7.2 Verificación Manual

#### Checklist de Verificación

# 1. Verificar base de datos  
mysql -u meditrack\_user -p meditrack  
SHOW TABLES;  
SELECT COUNT(\*) FROM users;  
SELECT COUNT(\*) FROM roles;  
EXIT;  
  
# 2. Verificar permisos de archivos  
ls -la storage/  
ls -la bootstrap/cache/  
  
# 3. Verificar logs  
tail -f storage/logs/laravel.log  
  
# 4. Verificar configuración  
php artisan config:show database  
php artisan config:show mail  
  
# 5. Verificar rutas  
php artisan route:list  
  
# 6. Verificar jobs y queues  
php artisan queue:work --once  
  
# 7. Test de email  
php artisan tinker  
Mail::raw('Test email', function($message) {  
 $message->to('test@example.com')->subject('Test');  
});

#### URLs de Verificación

# Páginas principales a verificar  
http://localhost/ # Página principal  
http://localhost/login # Login  
http://localhost/dashboard # Dashboard  
http://localhost/admin-dashboard # Panel admin  
http://localhost/medicamentos # Gestión medicamentos  
http://localhost/pacientes # Gestión pacientes

### 7.3 Métricas de Rendimiento

# Test de carga básico con Apache Bench  
ab -n 100 -c 10 http://localhost/  
  
# Monitoreo de recursos  
htop  
iostat -x 1  
  
# Análisis de queries MySQL  
mysql -u root -p  
SET GLOBAL slow\_query\_log = 'ON';  
SET GLOBAL long\_query\_time = 1;  
SHOW VARIABLES LIKE 'slow\_query\_log%';

## 8. Troubleshooting

### 8.1 Problemas Comunes

#### Error: “Class not found”

# Solución  
composer dump-autoload  
php artisan clear-compiled  
php artisan config:clear  
php artisan cache:clear

#### Error: “Permission denied”

# Solución para Linux/macOS  
sudo chown -R $USER:www-data storage  
sudo chown -R $USER:www-data bootstrap/cache  
chmod -R 775 storage  
chmod -R 775 bootstrap/cache  
  
# Solución para Docker  
docker-compose exec app chown -R www-data:www-data storage bootstrap/cache

#### Error: “Connection refused” (Base de datos)

# Verificar servicio MySQL  
sudo systemctl status mysql  
sudo systemctl start mysql  
  
# Para Docker  
docker-compose ps  
docker-compose logs mysql  
  
# Verificar conectividad  
mysql -h 127.0.0.1 -u meditrack\_user -p  
telnet localhost 3306

#### Error: “Mix manifest not found”

# Solución  
npm install  
npm run build  
php artisan config:clear

#### Error: “APP\_KEY not set”

# Solución  
php artisan key:generate  
php artisan config:cache

### 8.2 Logs y Diagnóstico

#### Ubicación de Logs

# Laravel logs  
tail -f storage/logs/laravel.log  
  
# Nginx logs  
sudo tail -f /var/log/nginx/access.log  
sudo tail -f /var/log/nginx/error.log  
  
# PHP-FPM logs  
sudo tail -f /var/log/php8.2-fpm.log  
  
# MySQL logs  
sudo tail -f /var/log/mysql/error.log  
  
# Docker logs  
docker-compose logs -f app  
docker-compose logs -f mysql

#### Comandos de Diagnóstico

# Información del sistema  
php -v  
composer --version  
node --version  
npm --version  
mysql --version  
  
# Estado de servicios  
sudo systemctl status nginx  
sudo systemctl status php8.2-fpm  
sudo systemctl status mysql  
  
# Verificar puertos  
netstat -tulpn | grep :80  
netstat -tulpn | grep :3306  
netstat -tulpn | grep :6379  
  
# Uso de recursos  
df -h  
free -m  
top

### 8.3 Recuperación de Fallos

#### Restaurar desde Backup

# Restaurar base de datos  
mysql -u root -p meditrack < /var/backups/meditrack/meditrack\_backup\_TIMESTAMP.sql  
  
# Restaurar archivos  
cd /var/www/html  
tar -xzf /var/backups/meditrack/files\_backup\_TIMESTAMP.tar.gz  
  
# Regenerar cache  
php artisan config:cache  
php artisan route:cache  
php artisan view:cache

#### Rollback de Código

# Ver commits recientes  
git log --oneline -10  
  
# Rollback a commit específico  
git reset --hard COMMIT\_HASH  
  
# Reinstalar dependencias  
composer install --no-dev --optimize-autoloader  
npm ci && npm run build  
  
# Reoptimizar  
php artisan config:cache  
php artisan route:cache  
php artisan view:cache

## 9. Mantenimiento

### 9.1 Tareas de Mantenimiento Regular

#### Diario

# Script de mantenimiento diario  
#!/bin/bash  
# daily\_maintenance.sh  
  
# Limpiar logs antiguos  
find storage/logs/ -name "\*.log" -mtime +30 -delete  
  
# Limpiar cache obsoleto  
php artisan cache:prune-stale-tags  
  
# Verificar espacio en disco  
df -h | grep -E "(/$|/var)"  
  
# Backup automático  
mysqldump --single-transaction meditrack\_prod > /var/backups/meditrack/daily\_$(date +%Y%m%d).sql

#### Semanal

# Script de mantenimiento semanal  
#!/bin/bash  
# weekly\_maintenance.sh  
  
# Optimizar tablas MySQL  
mysql -u root -p meditrack\_prod -e "OPTIMIZE TABLE users, roles, pacientes, medicamentos, tratamientos;"  
  
# Limpiar archivos temporales  
find /tmp -name "php\*" -mtime +7 -delete  
  
# Verificar integridad de archivos  
find /var/www/html -type f -name "\*.php" -exec php -l {} \;  
  
# Análisis de logs  
grep -i error storage/logs/laravel.log | tail -100

#### Mensual

# Script de mantenimiento mensual  
#!/bin/bash  
# monthly\_maintenance.sh  
  
# Actualizar composer  
composer self-update  
  
# Verificar vulnerabilidades  
composer audit  
  
# Actualizar npm packages (con precaución)  
npm outdated  
  
# Backup completo  
tar -czf /var/backups/meditrack/full\_backup\_$(date +%Y%m).tar.gz /var/www/html

### 9.2 Monitoreo Automatizado

#### Script de Monitoreo

#!/bin/bash  
# monitor.sh - Script de monitoreo de MediTrack  
  
# Variables  
ALERT\_EMAIL="admin@meditrack.com"  
LOG\_FILE="/var/log/meditrack\_monitor.log"  
TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')  
  
# Función para enviar alertas  
send\_alert() {  
 echo "$TIMESTAMP - ALERT: $1" >> $LOG\_FILE  
 echo "$1" | mail -s "MediTrack Alert" $ALERT\_EMAIL  
}  
  
# Verificar servicios web  
if ! curl -f -s http://localhost > /dev/null; then  
 send\_alert "Servicio web no responde"  
fi  
  
# Verificar base de datos  
if ! mysql -u meditrack\_user -p$DB\_PASSWORD -e "SELECT 1" > /dev/null 2>&1; then  
 send\_alert "Base de datos no accesible"  
fi  
  
# Verificar espacio en disco  
DISK\_USAGE=$(df -h / | awk 'NR==2 {print $5}' | sed 's/%//')  
if [ $DISK\_USAGE -gt 85 ]; then  
 send\_alert "Espacio en disco crítico: ${DISK\_USAGE}%"  
fi  
  
# Verificar memoria  
MEM\_USAGE=$(free | grep Mem | awk '{printf "%.0f", $3/$2 \* 100.0}')  
if [ $MEM\_USAGE -gt 90 ]; then  
 send\_alert "Uso de memoria crítico: ${MEM\_USAGE}%"  
fi  
  
# Verificar logs de errores  
ERROR\_COUNT=$(grep -c "ERROR" storage/logs/laravel.log | tail -100)  
if [ $ERROR\_COUNT -gt 10 ]; then  
 send\_alert "Múltiples errores detectados: $ERROR\_COUNT en últimas 100 líneas"  
fi  
  
echo "$TIMESTAMP - Monitoreo completado" >> $LOG\_FILE

### 9.3 Actualizaciones del Sistema

#### Procedimiento de Actualización

#!/bin/bash  
# update.sh - Script de actualización de MediTrack  
  
set -e  
  
echo "Iniciando actualización del sistema MediTrack..."  
  
# 1. Backup completo  
echo "Creando backup..."  
./scripts/backup.sh  
  
# 2. Modo mantenimiento  
php artisan down --message="Actualización en progreso"  
  
# 3. Actualizar código  
echo "Actualizando código..."  
git fetch origin  
git checkout main  
git pull origin main  
  
# 4. Verificar cambios en .env  
echo "Verificando configuración..."  
if [ -f ".env.example" ]; then  
 echo "Revisar .env.example para nuevas configuraciones"  
fi  
  
# 5. Actualizar dependencias  
echo "Actualizando dependencias..."  
composer install --no-dev --optimize-autoloader  
npm ci && npm run build  
  
# 6. Ejecutar migraciones  
echo "Ejecutando migraciones..."  
php artisan migrate --force  
  
# 7. Reoptimizar  
echo "Optimizando aplicación..."  
php artisan config:cache  
php artisan route:cache  
php artisan view:cache  
php artisan event:cache  
  
# 8. Verificar funcionalidad  
echo "Ejecutando tests críticos..."  
vendor/bin/pest --filter=Critical  
  
# 9. Reactivar aplicación  
php artisan up  
  
echo "Actualización completada exitosamente!"

## 10. Seguridad

### 10.1 Configuración de Seguridad

#### SSL/TLS

# Instalar Certbot para Let's Encrypt  
sudo apt install certbot python3-certbot-nginx  
  
# Obtener certificado  
sudo certbot --nginx -d meditrack.com -d www.meditrack.com  
  
# Verificar renovación automática  
sudo certbot renew --dry-run  
  
# Configurar renovación automática  
echo "0 12 \* \* \* /usr/bin/certbot renew --quiet" | sudo crontab -

#### Firewall

# Configurar UFW  
sudo ufw enable  
sudo ufw default deny incoming  
sudo ufw default allow outgoing  
sudo ufw allow ssh  
sudo ufw allow 'Nginx Full'  
sudo ufw allow mysql  
sudo ufw status

#### Fail2ban

# Instalar Fail2ban  
sudo apt install fail2ban  
  
# Configurar para Nginx  
sudo tee /etc/fail2ban/jail.local <<EOF  
[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  
findtime = 600  
bantime = 7200  
maxretry = 10  
  
[nginx-noscript]  
enabled = true  
port = http,https  
filter = nginx-noscript  
logpath = /var/log/nginx/access.log  
maxretry = 6  
EOF  
  
sudo systemctl enable fail2ban  
sudo systemctl start fail2ban

### 10.2 Backup y Recuperación

#### Script de Backup Automatizado

#!/bin/bash  
# backup.sh - Script de backup para MediTrack  
  
# Variables  
BACKUP\_DIR="/var/backups/meditrack"  
RETENTION\_DAYS=30  
TIMESTAMP=$(date +%Y%m%d\_%H%M%S)  
DB\_NAME="meditrack\_prod"  
DB\_USER="meditrack\_prod\_user"  
DB\_PASS="$DB\_PASSWORD"  
  
# Crear directorio de backup  
mkdir -p $BACKUP\_DIR  
  
# Backup de base de datos  
echo "Respaldando base de datos..."  
mysqldump --single-transaction --routines --triggers \  
 -u $DB\_USER -p$DB\_PASS $DB\_NAME \  
 | gzip > $BACKUP\_DIR/db\_backup\_$TIMESTAMP.sql.gz  
  
# Backup de archivos críticos  
echo "Respaldando archivos..."  
tar -czf $BACKUP\_DIR/files\_backup\_$TIMESTAMP.tar.gz \  
 --exclude=node\_modules \  
 --exclude=.git \  
 --exclude=storage/logs \  
 --exclude=storage/framework/cache \  
 --exclude=storage/framework/sessions \  
 --exclude=storage/framework/views \  
 /var/www/html  
  
# Backup de configuración del servidor  
echo "Respaldando configuración..."  
tar -czf $BACKUP\_DIR/config\_backup\_$TIMESTAMP.tar.gz \  
 /etc/nginx/sites-available/meditrack \  
 /etc/php/8.2/fpm/pool.d/www.conf \  
 /etc/mysql/mysql.conf.d/mysqld.cnf  
  
# Limpiar backups antiguos  
echo "Limpiando backups antiguos..."  
find $BACKUP\_DIR -name "\*.gz" -mtime +$RETENTION\_DAYS -delete  
  
# Verificar backup  
echo "Verificando integridad del backup..."  
if [ -f "$BACKUP\_DIR/db\_backup\_$TIMESTAMP.sql.gz" ] &&   
 [ -f "$BACKUP\_DIR/files\_backup\_$TIMESTAMP.tar.gz" ]; then  
 echo "Backup completado exitosamente: $TIMESTAMP"  
   
 # Opcional: subir a almacenamiento remoto  
 # aws s3 cp $BACKUP\_DIR/ s3://meditrack-backups/ --recursive  
else  
 echo "Error en el backup"  
 exit 1  
fi

## 11. Anexos

### 11.1 Anexo A: Comandos de Referencia Rápida

#### Laravel Artisan

# Aplicación  
php artisan serve # Servidor desarrollo  
php artisan down # Modo mantenimiento  
php artisan up # Salir de mantenimiento  
php artisan key:generate # Generar APP\_KEY  
  
# Base de datos  
php artisan migrate # Ejecutar migraciones  
php artisan migrate:fresh --seed # Recrear DB con datos  
php artisan db:seed # Ejecutar seeders  
  
# Cache  
php artisan config:cache # Cache configuración  
php artisan route:cache # Cache rutas  
php artisan view:cache # Cache vistas  
php artisan cache:clear # Limpiar cache  
  
# Optimización  
php artisan optimize # Optimizar para producción  
php artisan optimize:clear # Limpiar optimizaciones

#### NPM Scripts

npm run dev # Desarrollo con HMR  
npm run build # Build producción  
npm run format # Formatear código  
npm run lint # Linter  
npm run types # Verificar TypeScript

#### Docker Compose

docker-compose up -d # Iniciar servicios  
docker-compose down # Detener servicios  
docker-compose logs -f app # Ver logs aplicación  
docker-compose exec app bash # Acceder contenedor  
docker-compose restart app # Reiniciar servicio

### 11.2 Anexo B: Estructura de Directorios

meditrack/  
├── app/ # Aplicación Laravel  
│ ├── Console/ # Comandos Artisan  
│ ├── Http/ # Controladores y Middleware  
│ ├── Models/ # Modelos Eloquent  
│ ├── Policies/ # Políticas de autorización  
│ └── Services/ # Servicios de negocio  
├── bootstrap/ # Bootstrap Laravel  
├── config/ # Archivos configuración  
├── database/ # Migraciones y seeders  
├── docker/ # Archivos Docker  
├── public/ # Assets públicos  
├── resources/ # Frontend y vistas  
│ ├── css/ # Estilos  
│ ├── js/ # Código React/TypeScript  
│ └── views/ # Plantillas Blade  
├── routes/ # Definición de rutas  
├── storage/ # Almacenamiento local  
├── tests/ # Tests automatizados  
├── .env.example # Configuración ejemplo  
├── composer.json # Dependencias PHP  
├── package.json # Dependencias Node.js  
├── docker-compose.yml # Configuración Docker  
└── deploy.sh # Script despliegue

### 11.3 Anexo C: Puertos y Servicios

| Servicio | Puerto | Descripción |
| --- | --- | --- |
| Nginx | 80/443 | Servidor web |
| Laravel | 8000 | Desarrollo local |
| MySQL | 3306 | Base de datos |
| Redis | 6379 | Cache y sesiones |
| Vite HMR | 5173 | Hot Module Replacement |
| Node.js | 3000 | Desarrollo frontend |

### 11.4 Anexo D: Variables de Entorno Críticas

| Variable | Descripción | Ejemplo |
| --- | --- | --- |
| APP\_KEY | Clave aplicación | base64:… |
| APP\_ENV | Entorno | production/local |
| DB\_HOST | Host base datos | 127.0.0.1 |
| DB\_PASSWORD | Contraseña DB | contraseña\_segura |
| MAIL\_HOST | Servidor email | smtp.mailgun.org |
| REDIS\_HOST | Host Redis | 127.0.0.1 |

**Documento preparado por:** Equipo de Desarrollo MediTrack  
**Fecha de actualización:** Enero 2025  
**Versión:** 1.0  
**Estado:** Aprobado para Producción