**TRABAJO FIN DE GRADO**


UCAM
UNIVERSIDAD CATÓLICA
DE MURCIA

# ESCUELA POLITÉCNICA SUPERIOR

*Grado en Ingeniería Informática*

---

## MANUAL DE INSTALACIÓN

*Autor:*

*D. Marc Hernández Montesinos*

*Directora:*

Dra. Dña. *Angélica Guzmán Ponce*

*Murcia, Junio de 2026*

**TRABAJO FIN DE GRADO**


UCAM
UNIVERSIDAD CATÓLICA
DE MURCIA

# ESCUELA POLITÉCNICA SUPERIOR

*Grado en Ingeniería Informática*

## MANUAL DE INSTALACIÓN

*Autor:*

*D. Marc Hernández Montesinos*

*Directora:*

Dra. Dña. *Angélica Guzmán Ponce*

*Murcia, Junio de 2026*

*https://tfg.marchernandez.es/videos/video-demostrativo.mp4*

# ÍNDICE

# MANUAL DE INSTALACIÓN

Despliegue del sistema PKI + SSO sobre AlmaLinux 9 + Plesk

Documento separado del Trabajo de Fin de Grado *"Diseño e implementación de un sistema integrado de PKI y SSO para organizaciones pequeñas"* de Marc Hernández Montesinos (UCAM, Grado en Ingeniería Informática).

| Campo | Valor |
|---|---|
| Documento | Manual de Instalación |
| Versión | 1.0 |
| Fecha | 1 junio 2026 |
| URL pública | https://tfg.marchernandez.es/manuales/Manual_Instalacion.pdf |
| TFG asociado | https://tfg.marchernandez.es |

Este manual conserva la numeración interna original (sección A.x) por trazabilidad con las versiones previas del documento. Las referencias cruzadas que apuntan al cuerpo del TFG (capítulos 1-10, anexos A-D) se mantienen tal cual y son válidas frente al PDF principal del TFG.

Este anexo describe el procedimiento completo de instalación del sistema PKI + SSO sobre un VPS AlmaLinux 9 gestionado con Plesk Obsidian. Los comandos están verificados sobre el despliegue real `marchernandez.es` y reflejan las decisiones de diseño documentadas en los capítulos 5 (tecnologías) y 7.12 (despliegue) del cuerpo del TFG. La instalación completa requiere unas **2-3 horas** de trabajo manual la primera vez (sin contar la propagación DNS de los subdominios y la emisión de los certificados Let's Encrypt).

## A.1 Requisitos previos

### *A.1.1 Hardware mínimo recomendado*

| Recurso | Mínimo | Recomendado producción |
|---|---|---|
| vCPU | 2 | 4 |
| RAM | 2 GB | 4 GB |
| Disco | 20 GB | 40 GB (SSD/NVMe) |
| Red | IPv4 pública + reverse DNS configurado | IPv4 + IPv6 + reverse DNS |

El despliegue de referencia descrito en este TFG se ejecuta sobre un VPS Ionos con 2 vCPU, 2 GB de RAM y 40 GB de NVMe (véase Sección 7.10.1 del cuerpo).

### *A.1.2 Software base*

| Componente | Versión mínima | Notas |
|---|---|---|
| **AlmaLinux** | 9.x | RHEL-compatible; CentOS Stream 9 sirve si Plesk lo soporta. |
| **Plesk Obsidian** | 18.0.65+ | El despliegue prueba 18.0.69. Subdominios gestionados desde el panel. |
| **PHP** | 8.4.x | Estricto: `declare(strict_types=1)` en todo el código. |
| **MariaDB** | 10.5+ | Conexión por socket UNIX local; no se expone el puerto 3306. |
| **Apache HTTPD** | 2.4.x | Servido por Plesk; se acepta también el frontend Nginx de Plesk en modo proxy. |
| **OpenSSL** | 3.x | Disponible por defecto en AlmaLinux 9 (3.0.x o 3.2.x). |

| Componente | Versión mínima | Notas |
|---|---|---|
| **Composer** | 2.x | Para gestionar dependencias dev (PHPUnit). |
| **systemd** | 252+ | Para timers automáticos (cron alternativo). |

### *A.1.3 Extensiones PHP*

Todas vienen empaquetadas en Plesk; comprobar instalación:

```
php -m | grep -E "openssl|pdo_mysql|sodium|mbstring|json|curl"
```

Deben aparecer: `openssl`, `pdo`, `pdo_mysql`, `sodium`, `mbstring`, `json`, `curl`. Si falta alguna en un subdominio concreto, activarla desde **Plesk → Subdominio → Configuración de PHP**.

### *A.1.4 Dominios y subdominios*

Cinco subdominios necesarios, todos delegados al mismo VPS con registros A/AAAA y reverse DNS:

| Subdominio | Protocolo público | Document Root | Función |
|---|---|---|---|
| `sso.marchernandez.es` | **HTTPS** | `/var/www/vhosts/marchernandez.es/sso.marchernandez.es/public` | Identity Provider OAuth/OIDC |
| `pki.marchernandez.es` | **HTTPS** | `/var/www/vhosts/marchernandez.es/pki.marchernandez.es/public` | Portal de gestión PKI |
| `ocsp.marchernandez.es` | **HTTP** | `/var/www/vhosts/marchernandez.es/ocsp.marchernandez.es` | Responder OCSP (RFC 6960) |

| Subdominio | Protocolo público | Document Root | Función |
|---|---|---|---|
| `crl.marchernandez.es` | **HTTP** | `/var/www/vhosts/marchernandez.es/crl.marchernandez.es` | Distribución de CRLs |
| `ca.marchernandez.es` | **HTTP** | `/var/www/vhosts/marchernandez.es/ca.marchernandez.es` | Repositorio público de CAs (AIA) |

Los subdominios `ocsp`, `crl` y `ca` se sirven por HTTP, no por HTTPS. Esto es intencional y conforme a estándar: el cliente que valida una cadena de certificados todavía no confía en TLS, por lo que envolver OCSP/CRL en TLS sería un *deadlock* (véase Sección 7.7 del cuerpo y Sección D.1.2 del Anexo D). La integridad la aportan las firmas criptográficas de la propia CA.

Opcionalmente, en despliegues que usen autenticación con certificado del SSO con Nginx como frontal, se necesita un sexto subdominio:

| Subdominio | Protocolo | Función |
|---|---|---|
| `cert-auth.sso.marchernandez.es` | **HTTPS** | Mismo código que SSO pero con `ssl_verify_client optional` (mTLS para FNMT/DNIe) |

Este subdominio se sirve a través de symlinks apuntando al SSO principal (véase Sección A.3.4).

## A.2 Preparación del servidor

### *A.2.1 Actualización del sistema*

```
sudo dnf update -y
sudo dnf install -y epel-release
sudo systemctl reboot   # Si la actualización incluyó kernel
```

### *A.2.2 Paquetes adicionales*

```
sudo dnf install -y \
    openssl openssl-devel \
    git rsync wget curl \
    unzip tar gzip \
    policycoreutils-python-utils selinux-policy-targeted \
    firewalld \
    fail2ban
```

`policycoreutils-python-utils` aporta `semanage` y `restorecon`, necesarios si SELinux está en *enforcing*.

### *A.2.3 Verificación de la instalación de Plesk*

```
plesk version
plesk bin server_pref -s | grep -E "version|os"
```

Si Plesk no está instalado, seguir la guía oficial de instalación (fuera del alcance de este anexo).

### *A.2.4 Composer*

```
curl -sS https://getcomposer.org/installer | php
sudo mv composer.phar /usr/local/bin/composer
composer --version
```

### *A.2.5 Estructura de carpetas*

El sistema utiliza una separación deliberada entre lo público y lo privado:

```
/var/www/vhosts/marchernandez.es/
├── private/                                ← NUNCA accesible vía web
│   ├── config/
│   │   └── database.php                    ← credenciales BD globales
│   ├── sso/                                ← sólo claves compartidas
│   └── keys/                               ← (sin uso actual; reservado)
│
├── sso.marchernandez.es/                   ← document root subdominio SSO
│   ├── public/                             ← este sí es DocumentRoot
público
│   │   ├── api/                            ← endpoints /api/*
│   │   ├── .well-known/                    ← discovery OIDC
│   │   ├── admin/                          ← panel admin
│   │   ├── cert-auth/                      ← login por certificado
│   │   ├── assets/, index.php, login.php, ...
│   │   └── .htaccess
│   ├── src/                                ← código PSR-4 (no servido)
│   ├── config/                             ← credentials.php (chmod 600)
│   ├── keys/                               ← claves privadas JWT (chmod
600)
│   ├── templates/, tests/, cron/
│   └── install.php                         ← se elimina tras instalar
│
├── pki.marchernandez.es/                   ← document root subdominio PKI
│   ├── public/                             ← DocumentRoot público (DEBE
apuntar aquí)
│   │   ├── api/auth/callback.php
│   │   ├── admin/, assets/, config/openssl/
│   │   ├── certificates.php, request.php, revoke.php, ...
│   │   ├── composer.json
│   │   └── .htaccess
│   ├── lib/                                ← clases MaHerMo\PKI (no
servido)
│   ├── config/                             ← app.php, database.php
│   ├── ca/                                 ← almacén de claves CA (chmod
700)
│   │   ├── root/, intermediate/, intermediate_2/, ...
│   ├── logs/, temp/
│   ├── cron/generate_crl.php
│   ├── .env
│   ├── bootstrap.php, load_env.php
│   └── migrations/
│
├── ocsp.marchernandez.es/                  ← DocumentRoot = aquí mismo
│   ├── index.php, README.md, .htaccess
│   └── assets/imgs/
│
├── crl.marchernandez.es/
│   ├── index.php, download.php, .htaccess
│   ├── assets/imgs/
│   └── root.crl, intermediate.crl, ...     ← generadas por cron
│
└── ca.marchernandez.es/
    ├── index.php, download.php, .htaccess
    └── assets/imgs/
```

**Regla mnemotécnica:** todo lo que termina en `*.marchernandez.es/public/` o `*.marchernandez.es/` es el DocumentRoot público; todo lo demás debe quedar fuera del DocumentRoot o protegido por `.htaccess`.

### *A.2.6 Usuario y propiedad*

```
# El usuario Plesk que aloja el dominio (típicamente coincide con el nombre del dominio)
PLESK_USER=marchernandez
PLESK_GROUP=psacln

sudo chown -R $PLESK_USER:$PLESK_GROUP /var/www/vhosts/marchernandez.es/
```

Plesk crea automáticamente el grupo `psacln` para todos los dominios alojados.

### *A.2.7 Permisos base*

```
cd /var/www/vhosts/marchernandez.es/

# Privado: sólo el propietario
sudo chmod 700 private/
sudo find private/ -type d -exec chmod 700 {} \;
sudo find private/ -type f -exec chmod 600 {} \;

# Public web roots
for sub in sso pki ocsp crl ca; do
    sudo chmod 755 ${sub}.marchernandez.es/
done

# Carpetas que deben ser escribibles por Apache
sudo chmod 750 pki.marchernandez.es/logs/
sudo chmod 750 pki.marchernandez.es/temp/
sudo chmod 700 pki.marchernandez.es/ca/
sudo chmod 700 sso.marchernandez.es/keys/

# Archivos especialmente sensibles
sudo chmod 600 sso.marchernandez.es/config/credentials.php   2>/dev/null ||
true
sudo chmod 600 pki.marchernandez.es/.env                     2>/dev/null ||
true
sudo chmod 600 sso.marchernandez.es/keys/*.pem               2>/dev/null ||
true
sudo chmod 600 pki.marchernandez.es/ca/*/*.key               2>/dev/null ||
true
```

## A.3 Configuración de subdominios

### *A.3.1 Crear subdominios en Plesk*

Para cada uno de los cinco subdominios obligatorios:

1. **Plesk → Sitios web y dominios → marchernandez.es → Añadir subdominio**.
2. Nombre del subdominio: `sso`, `pki`, `ocsp`, `crl`, `ca` (uno cada vez).
3. **Document Root** según la tabla Sección A.1.4 (importante: `sso` y `pki` deben apuntar a `/public`, no al directorio raíz del subdominio).
4. **Versión PHP:** 8.4.x (la más alta disponible en Plesk).
5. Aplicar.

### *A.3.2 Configuración PHP por subdominio*

Para `sso` y `pki`:

```
memory_limit = 256M
upload_max_filesize = 10M
post_max_size = 10M
max_execution_time = 60
display_errors = Off
log_errors = On
error_log = /var/www/vhosts/marchernandez.es/logs/<subdominio>_php_error.log

session.cookie_httponly = 1
session.cookie_secure = 1
session.cookie_samesite = Lax
session.use_strict_mode = 1
session.use_only_cookies = 1
```

Para `ocsp`, `crl`, `ca`:

```
memory_limit = 128M
display_errors = Off
log_errors = On
```

Activar extensiones (en Plesk → Configuración de PHP → Extensiones):

- `openssl`
- `pdo_mysql`
- `sodium`
- `mbstring`
- `json`

- `curl`

### *A.3.3 HTTPS para sso y pki (Let's Encrypt)*

1. **Plesk → Subdominio → SSL/TLS Certificates → Instalar (gratuito vía Let's Encrypt)**.
2. Marcar el subdominio y también `www.<subdominio>` si procede.
3. Activar Redirección permanente segura de HTTP a HTTPS.

El `.htaccess` del SSO y del PKI ya fuerza HTTPS también a nivel aplicativo:

```
RewriteCond %{HTTPS} off
RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

### *A.3.4 HTTP para ocsp, crl y ca (con justificación)*

Los tres subdominios `ocsp`, `crl` y `ca` se sirven sólo por HTTP. Razones (RFC 5280 Sección 4.2.1.13, RFC 6960):

1. **Evitar deadlock de confianza:** un cliente que comprueba si el certificado del servidor TLS está revocado todavía no ha terminado de validar la cadena; envolver esa comprobación en otra capa TLS produce dependencia circular.
2. **Integridad por firma:** las CRL y respuestas OCSP están firmadas por la propia CA, lo que aporta integridad y autenticación sin necesidad de TLS.
3. **Compatibilidad:** muchos clientes legacy (y `openssl ocsp` mismo) asumen URLs HTTP en los campos AIA y CDP.

En Plesk:

- NO marcar Let's Encrypt para estos subdominios.
- NO habilitar redirección HTTP→HTTPS.

### *A.3.5 (Opcional) Subdominio `cert-auth.sso.marchernandez.es` para mTLS*

Si se quiere soportar login por certificado FNMT/DNIe:

1. Crear el subdominio `cert-auth.sso.marchernandez.es` en Plesk.

2. Ejecutar el script `sso/config/setup_cert_auth_domain.sh` desde el servidor (como root). Este script crea symlinks apuntando al SSO principal, de modo que ambos subdominios sirven el mismo código:

```
cd /var/www/vhosts/marchernandez.es/sso.marchernandez.es/config
sudo bash setup_cert_auth_domain.sh
```

3. Añadir el directorio del SSO a `open_basedir` del subdominio `cert-auth` (Plesk → PHP Settings).

4. Pegar las directivas de `nginx_cert_auth.conf` en **Plesk → cert-auth → Apache y Nginx → Directivas Nginx**.

5. Activar Let's Encrypt para `cert-auth.sso.marchernandez.es`.

El subdominio `sso.marchernandez.es` no debe pedir certificado (sin `ssl_verify_client`); el subdominio `cert-auth` sí (`ssl_verify_client optional`). Esto evita que cualquier visitante del login normal vea un *popup* del navegador pidiendo certificado.

## A.4 Base de datos

### *A.4.1 Crear ambas BD en Plesk*

1. **Plesk → Bases de datos → Añadir base de datos** dos veces:

| Nombre BD | Usuario | Privilegios | Servidor |
|---|---|---|---|
| `marchernandezmo_sso_nuevo_2` | `admin_sso_nuevo_2` | All | localhost |
| `marchernandezmo_pki_nuevo` | `admin_pki_nuevo` | All | localhost |

2. Generar contraseñas robustas (Plesk ofrece generador automático). Guardarlas en un gestor de secretos (KeePassXC, Bitwarden, etc.).

**Por qué dos BDs separadas.** El sistema mantiene el SSO y la PKI en bases de datos lógicamente independientes para aislar `blast radius`, permitir permisos diferenciados y separar ciclos de vida. Justificado en Sección 7.3.3 del cuerpo y Sección E.1.1 del Anexo E.

### *A.4.2 Endurecer privilegios (opcional pero recomendado)*

Plesk crea los usuarios con `ALL PRIVILEGES`, lo cual es excesivo para *runtime*. Para minimizar privilegios:

```
-- Conectar como root
mysql -u root -p

-- Crear usuarios runtime con permisos mínimos (en lugar de los que crea
Plesk con ALL)
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'admin_sso_nuevo_2'@'localhost';
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'admin_pki_nuevo'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE
    ON marchernandezmo_sso_nuevo_2.*
    TO 'admin_sso_nuevo_2'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE
    ON marchernandezmo_pki_nuevo.*
    TO 'admin_pki_nuevo'@'localhost';

-- Revocación inmediata en login por certificado Omnipresence (SSO → PKI).
-- Sin este permiso `CertificateAuth::checkPKIRevocation()` no puede leer la
tabla;
-- la confianza en revocación pasa sólo por CRL/OCSP del lado cliente hasta
que exista esta concesión o una vista equivalente.
GRANT SELECT
    ON marchernandezmo_pki_nuevo.certificates
    TO 'admin_sso_nuevo_2'@'localhost';

FLUSH PRIVILEGES;
```

Sustituir los nombres `marchernandezmo_pki_nuevo`, `admin_sso_nuevo_2` por los que figuren realmente en `sso/config/config.php` y en Plesk si difieren entre entornos.

Las migraciones DDL se aplican entonces con un usuario administrativo dedicado (p. ej. el `root` de Plesk), no con los usuarios de runtime.

### *A.4.3 Conectividad por socket UNIX*

Plesk usa `localhost` y MariaDB resuelve `localhost` a socket UNIX (`/var/lib/mysql/mysql.sock`). Esto significa que MariaDB no escucha en TCP 3306 desde fuera, lo cual es deseable. Verificar:

```
sudo ss -tlnp | grep 3306    # No debería devolver nada
sudo ls -l /var/lib/mysql/mysql.sock
```

Si por algún motivo MariaDB escucha en 3306, añadir a `/etc/my.cnf.d/server.cnf`:

```
[mysqld]
bind-address = 127.0.0.1
skip-networking = 0
```

Y reiniciar: `sudo systemctl restart mariadb`.

### *A.4.4 Importar esquema y datos iniciales*

Hay dos vías para inicializar las BDs:

#### *Vía A (recomendada): script `install.php` del SSO*

El script `sso/install.php` aplica el esquema completo del SSO (10+ tablas + migraciones 002-010 + datos iniciales), genera las claves JWT RS256 en `keys/`, crea el usuario `admin` con contraseña aleatoria y registra la aplicación PKI como cliente OAuth2.

```
cd /var/www/vhosts/marchernandez.es/sso.marchernandez.es

# Copiar credenciales de ejemplo y editar
cp config/credentials.example.php config/credentials.php
chmod 600 config/credentials.php
nano config/credentials.php
# Definir DB_PASS = '<password de marchernandezmo_sso_nuevo_2>'
# Definir ENCRYPTION_KEY = '<hex 64 chars>'  → generar con: php -r "echo
bin2hex(random_bytes(32));"

# Ejecutar instalación
php install.php

# Anotar las credenciales que muestra:
#   Admin Username: admin
#   Admin Password: <generado aleatoriamente>
#   PKI Client ID:     mhsso_pki_client
#   PKI Client Secret: <generado aleatoriamente>

# Eliminar el script tras la instalación
rm install.php
```

#### *Vía B (manual): aplicar migraciones SQL*

Si se prefiere no ejecutar el instalador automático (p. ej. para auditar cada paso), aplicar las migraciones del Anexo E Sección E.5 en orden:

```
mysql -u root -p marchernandezmo_sso_nuevo_2 <
sso/migrations/002_keypairs_filesystem.sql
mysql -u root -p marchernandezmo_sso_nuevo_2 <
sso/migrations/003_trusted_issuers_eku.sql
mysql -u root -p marchernandezmo_sso_nuevo_2 <
sso/migrations/004_hardening.sql
mysql -u root -p marchernandezmo_sso_nuevo_2 <
sso/migrations/005_oauth2_integration.sql
mysql -u root -p marchernandezmo_sso_nuevo_2 <
sso/migrations/006_multi_certificate_types.sql
mysql -u root -p marchernandezmo_sso_nuevo_2 <
sso/migrations/007_recovery_codes.sql
mysql -u root -p marchernandezmo_sso_nuevo_2 <
sso/migrations/010_application_user_access.sql
mysql -u root -p marchernandezmo_sso_nuevo_2 <
sso/migrations/011_pkce_support.sql
mysql -u root -p marchernandezmo_sso_nuevo_2 <
sso/migrations/012_post_logout_redirect_uris.sql

mysql -u root -p marchernandezmo_pki_nuevo < pki_migration.sql
mysql -u root -p marchernandezmo_pki_nuevo <
pki/migrations/add_oauth2_support.sql
mysql -u root -p marchernandezmo_pki_nuevo <
pki/migrations/008_fix_sso_user_mapping.sql
mysql -u root -p marchernandezmo_pki_nuevo <
pki/migrations/009_crl_auto_generation.sql
```

Y luego crear manualmente el usuario admin del PKI:

```
INSERT INTO pki_users (username, email, full_name, role, is_active,
created_via, password_hash, created_at)
VALUES (
    'admin',
    'admin@marchernandez.es',
    'Administrador del Sistema',
    'superadmin',
    1,
    'local',
    '<hash Argon2id>',   -- generar con: php -r "echo
password_hash('TU_CONTRASEÑA', PASSWORD_ARGON2ID);"
    NOW()
);
```

### *A.4.5 Verificar importación*

```
# Tablas SSO
mysql -u admin_sso_nuevo_2 -p marchernandezmo_sso_nuevo_2 -e "SHOW TABLES;"
# Debe devolver: sso_users, sso_applications, sso_application_users,
sso_sessions,
#                sso_authorization_codes, sso_keypairs, sso_trusted_issuers,
#                sso_user_certificates, sso_recovery_codes,
sso_password_resets,
#                sso_audit_log, sso_login_logs, sso_rate_limits

# Tablas PKI
mysql -u admin_pki_nuevo -p marchernandezmo_pki_nuevo -e "SHOW TABLES;"
# Debe devolver: pki_users, certificate_authorities, certificate_templates,
#                certificate_requests, certificates, crl_records,
crl_downloads,
#                ocsp_queries, audit_log, pki_config, user_sessions
```

## A.5 Despliegue del código

### *A.5.1 Subir el código*

Tres opciones según preferencia:

#### *Opción 1: Git (recomendada para producción)*

```
ssh marchernandez@marchernandez.es
cd /var/www/vhosts/marchernandez.es/

# Clonar a un directorio temporal
git clone git@github.com:tu-usuario/tfg-pki.git tmp

# Mover SSO al subdominio
sudo rsync -av tmp/sso/ sso.marchernandez.es/
# Mover PKI al subdominio (con el truco de DocumentRoot=/public)
sudo rsync -av tmp/pki/ pki.marchernandez.es/
# Mover OCSP/CRL/CA a sus respectivos subdominios
sudo rsync -av tmp/ocsp/ ocsp.marchernandez.es/
sudo rsync -av tmp/crl/  crl.marchernandez.es/
sudo rsync -av tmp/ca/   ca.marchernandez.es/

# Limpiar
sudo rm -rf tmp

# Reajustar propietario
sudo chown -R marchernandez:psacln /var/www/vhosts/marchernandez.es/
```

### *Opción 2: rsync directo desde el equipo de desarrollo*

```
# Desde tu equipo local
rsync -avz --delete sso/
marchernandez@marchernandez.es:/var/www/vhosts/marchernandez.es/sso.marcherna
ndez.es/
rsync -avz --delete pki/
marchernandez@marchernandez.es:/var/www/vhosts/marchernandez.es/pki.marcherna
ndez.es/
rsync -avz --delete ocsp/
marchernandez@marchernandez.es:/var/www/vhosts/marchernandez.es/ocsp.marchern
andez.es/
rsync -avz --delete crl/
marchernandez@marchernandez.es:/var/www/vhosts/marchernandez.es/crl.marcherna
ndez.es/
rsync -avz --delete ca/
marchernandez@marchernandez.es:/var/www/vhosts/marchernandez.es/ca.marchernan
dez.es/
```

### *Opción 3: Subida manual vía SFTP/Plesk File Manager*

Sólo recomendada para despliegues puntuales o si no hay SSH disponible. Usar FileZilla o el File Manager de Plesk con las mismas rutas.

## *A.5.2 Configurar variables de entorno del PKI (`.env`)*

El PKI carga variables de entorno desde un fichero `.env` mediante `load_env.php`:

```
cd /var/www/vhosts/marchernandez.es/pki.marchernandez.es/
sudo -u marchernandez nano .env
```

Contenido:

```
# Base de datos PKI
PKI_DB_HOST=localhost
PKI_DB_PORT=3306
PKI_DB_NAME=marchernandezmo_pki_nuevo
PKI_DB_USER=admin_pki_nuevo
PKI_DB_PASS=<contraseña BD pki>

# Base de datos SSO (cross-connection para verificar revocaciones desde PKI)
SSO_DB_HOST=localhost
SSO_DB_PORT=3306
SSO_DB_NAME=marchernandezmo_sso_nuevo_2
SSO_DB_USER=admin_sso_nuevo_2
SSO_DB_PASS=<contraseña BD sso>

# Cifrado interno
PKI_ENCRYPTION_KEY=<hex 64 chars>   # php -r "echo
bin2hex(random_bytes(32));"

# Modo debug (off en producción)
PKI_DEBUG=false
```

Permisos del fichero:

```
sudo chmod 600 .env
sudo chown marchernandez:psacln .env
```

### *A.5.3 Configurar credentials.php del SSO*

Si se usó la Vía A del paso A.4.4, este fichero ya está creado. Si se usó la Vía B:

```
cd /var/www/vhosts/marchernandez.es/sso.marchernandez.es/config/
sudo -u marchernandez cp credentials.example.php credentials.php
sudo -u marchernandez nano credentials.php
```

Contenido:

```
<?php
define('DB_PASS', '<contraseña de marchernandezmo_sso_nuevo_2>');
define('ENCRYPTION_KEY', '<hex 64 chars>');   // php -r "echo
bin2hex(random_bytes(32));"
```

Permisos:

```
sudo chmod 600 credentials.php
sudo chown marchernandez:psacln credentials.php
```

### *A.5.4 Vincular SSO ↔ PKI mediante `client_secret`*

Editar el fichero `pki/config/app.php` para inyectar el `client_secret` que devolvió `install.php`:

```
// pki/config/app.php
define('SSO_ENABLED', true);
define('SSO_BASE_URL', 'https://sso.marchernandez.es');
define('SSO_CLIENT_ID', 'mhsso_pki_client');
define('SSO_CLIENT_SECRET', '<client_secret generado por install.php>');
```

Replicar el mismo valor en la sección `'sso'` del array de configuración.

### *A.5.5 Composer (dependencias dev)*

El sistema no tiene dependencias de runtime (todo PSR-4 hecho a mano, evitando una dependencia externa de un autoloader). Sólo se usa Composer para PHPUnit:

```
cd /var/www/vhosts/marchernandez.es/pki.marchernandez.es/public/
composer install --no-dev --optimize-autoloader
```

En producción se ejecuta con `--no-dev` para no instalar PHPUnit. El `optimize-autoloader` no aplica en este sistema porque no hay autoloader Composer; el flag se mantiene por buenas prácticas si en futuro se añade alguna dependencia.

### *A.5.6 Claves JWT (RS256) del SSO*

Si se ejecutó `install.php`, ya hay un keypair en `sso/keys/<kid>.pem`. Para regenerar o rotar:

```
cd /var/www/vhosts/marchernandez.es/sso.marchernandez.es/

# Generar nueva clave RSA 2048 bits
openssl genrsa -out keys/$(openssl rand -hex 16).pem 2048

# Insertar en la BD (el install.php lo automatiza; manualmente):
# Ver Sección A.4.4 Vía A para el procedimiento completo.

# Ajustar permisos (importante)
chmod 700 keys/
chmod 600 keys/*.pem
chown -R marchernandez:psacln keys/
```

### *A.5.7 Estructura de directorios del PKI*

Crear las carpetas runtime del PKI:

```
cd /var/www/vhosts/marchernandez.es/pki.marchernandez.es/

sudo -u marchernandez mkdir -p ca/{root,intermediate}
sudo -u marchernandez mkdir -p logs
sudo -u marchernandez mkdir -p temp

# Permisos
sudo chmod 700 ca/
sudo chmod 750 logs/
sudo chmod 750 temp/
```

## A.6 Inicialización PKI

**Hito crítico de seguridad:** la generación de la Root CA es la operación más sensible del sistema. La clave privada de la raíz no debe acabar en el servidor de producción en un despliegue maduro (modo *Root offline*). Como se discute en Sección 7.4.1 del cuerpo y en Sección 9.3, la implementación actual del TFG mantiene la Root CA en modo de uso restringido (no estrictamente offline): la clave existe en el servidor, pero sólo se desbloquea con passphrase para emitir

intermedias. Las intermedias son las que firman certificados de usuario en operación normal.

### *A.6.1 Crear la Root CA*

Desde el portal web (recomendado para usuarios sin experiencia OpenSSL):

1. Iniciar sesión en `https://pki.marchernandez.es` con el usuario admin.
2. **Admin → CAs → Crear Root CA**.
3. Completar el formulario (CN, validez en años, tamaño de clave 4096 RSA, algoritmo SHA-384).
4. Establecer una passphrase fuerte (mínimo 16 caracteres, no almacenada en BD).
5. Guardar la passphrase en un gestor de secretos *offline* (KeePassXC + backup en USB cifrado, p. ej.).

Internamente, la PKI invoca `PKIEngine::createRootCA()`, que ejecuta `openssl genrsa` + `openssl req -x509` con la configuración de `pki/public/config/openssl/root_openssl.cnf` (Anexo F Sección F.2).

Como referencia para auditores, el script equivalente bash es `pki/public/bin/ca-init-root.sh` (no se ejecuta en producción; sirve como documentación operacional del proceso interno).

### *A.6.2 Crear CAs intermedias*

Por cada *issuance domain* (Omnipresence TrustCA Intermediate, RACV, UCAM DESI, etc.):

1. **Admin → CAs → Crear CA Intermedia**.
2. Seleccionar la Root CA como emisora.
3. CN, validez, tamaño 4096 RSA.
4. Indicar `pathlen:0` (forzado por el código desde la migración de Anexo F).
5. Confirmar con la passphrase de la Root.

Internamente: `PKIEngine::createIntermediateCA()` genera la CSR, la firma con la Root (descifrada en memoria sólo durante la operación) y construye el `index.txt` + `serial` + `crlnumber` de la intermedia.

### *A.6.3 Publicar CAs en `ca.marchernandez.es`*

La página `ca.marchernandez.es/index.php` lee dinámicamente la tabla `certificate_authorities` y sirve los PEM/DER almacenados en la columna `certificate_pem`/`certificate_der`. No se necesita copiar manualmente nada.

Verificar accediendo a:

- `http://ca.marchernandez.es/` → debe listar todas las CAs activas.
- `http://ca.marchernandez.es/root.crt` → certificado raíz en PEM.
- `http://ca.marchernandez.es/chain.pem` → cadena completa.

### *A.6.4 Generar CRL inicial*

Para cada CA, generar manualmente la primera CRL (luego el cron de Sección A.7 lo automatiza):

```
# Vía panel admin (recomendado)
# Admin → CRL → seleccionar CA → "Generar CRL ahora"

# Vía CLI (alternativa)
php
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/cron/generate_crl.php
```

Esto crea entradas en `crl_records` y publica el fichero `.crl` en `/var/www/vhosts/marchernandez.es/crl.marchernandez.es/`.

### *A.6.5 Verificar con OpenSSL*

```
# 1. Comprobar el certificado raíz
openssl x509 -in /tmp/root.crt -text -noout | head -40
# (debe mostrar Issuer == Subject, Validity 50 años, KU=keyCertSign,cRLSign,
basicConstraints CA:TRUE pathlen:0)

# 2. Comprobar cadena
curl -s http://ca.marchernandez.es/chain.pem > /tmp/chain.pem
openssl verify -CAfile /tmp/chain.pem -untrusted /tmp/chain.pem
/tmp/algun_cert_emitido.crt
# Debe imprimir: certificate.crt: OK

# 3. Comprobar CRL
curl -s http://crl.marchernandez.es/intermediate.crl > /tmp/intermediate.crl
openssl crl -in /tmp/intermediate.crl -inform DER -text -noout | head -30
# (debe mostrar Last Update / Next Update separados 4h y CRL Number > 0)

# 4. Comprobar OCSP responder (responde 'unknown' si no hay aún certificados
emitidos)
openssl ocsp \
    -issuer /tmp/intermediate.crt \
    -cert /tmp/algun_cert_emitido.crt \
    -url http://ocsp.marchernandez.es \
    -resp_text -noverify
```

### *A.6.6 Configurar trusted issuers externos (FNMT, DNIe)*

Para soportar login por certificado FNMT o DNIe en el SSO:

```
cd /var/www/vhosts/marchernandez.es/
sudo -u marchernandez bash scripts/setup-trusted-cas.sh
```

El script descarga las CAs raíz e intermedias de FNMT y DNIe desde sus URLs oficiales, las verifica y las añade al *bundle* `trusted_ca_bundle.pem` que usa Apache/Nginx para `ssl_verify_client`. Las URLs son:

- FNMT: `https://www.sede.fnmt.gob.es/certs/ACRAIZFNMTRCM.crt`, `ACUSUARIOSFNMT.crt`, etc.
- DNIe: `http://www.dnie.es/AC_RAIZ_DNIE.crt`, `AC_DNIE_002.crt`, …, `AC_DNIE_006.crt`.

Tras instalarlos, las entradas correspondientes en `sso_trusted_issuers` (creadas por `install.php`) reconocerán los certificados emitidos por FNMT/DNIe.

## A.7 Tareas programadas (systemd timers / cron)

Se documentan dos variantes equivalentes: con `cron` (estándar) o con `systemd timers` (más moderno, soportado por Plesk). En el despliegue de referencia se usa `cron` por simplicidad.

### *A.7.1 Tareas a programar*

| Tarea | Frecuencia | Script |
|---|---|---|
| **CRL rotate** | cada 4 h | `pki.marchernandez.es/cron/generate_crl.php` |
| **SSO cleanup** | diario 03:00 | `sso.marchernandez.es/cron/cleanup.php` |
| **OCSP health check** | cada 5 min | `pki.marchernandez.es/public/bin/ocsp-health-check.sh` (opcional) |
| **Backup BD** | diario 04:00 | `mysqldump` (gestionado por Plesk → Backup Manager) |
| **Renovación SSL** | mensual (auto) | Plesk → Let's Encrypt (automático) |

### *A.7.2 Variante cron (Plesk)*

**Plesk → Herramientas y configuración → Tareas programadas → Añadir tarea**.

```
# CRL: cada 4 horas
0 */4 * * * /usr/bin/php
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/cron/generate_crl.php
>> /var/www/vhosts/marchernandez.es/pki.marchernandez.es/logs/crl_cron.log
2>&1

# SSO cleanup: diario a las 03:00
0 3 * * * cd /var/www/vhosts/marchernandez.es/sso.marchernandez.es &&
/usr/bin/php cron/cleanup.php >> logs/cron.log 2>&1

# OCSP health check: cada 5 minutos (opcional)
*/5 * * * *
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/public/bin/ocsp-health-
check.sh >> /var/log/pki/ocsp-health.log 2>&1
```

### *A.7.3 Variante systemd timers*

Crear `/etc/systemd/system/pki-crl-rotate.service`:

```
[Unit]
Description=PKI CRL rotation
After=network.target

[Service]
Type=oneshot
User=marchernandez
ExecStart=/usr/bin/php
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/cron/generate_crl.php
StandardOutput=append:/var/www/vhosts/marchernandez.es/pki.marchernandez.es/l
ogs/crl_cron.log
StandardError=append:/var/www/vhosts/marchernandez.es/pki.marchernandez.es/lo
gs/crl_cron.log
```

Crear `/etc/systemd/system/pki-crl-rotate.timer`:

```
[Unit]
Description=Rotate PKI CRLs every 4 hours
Requires=pki-crl-rotate.service

[Timer]
OnCalendar=*-*-* 00/4:00:00
Persistent=true

[Install]
WantedBy=timers.target
```

Habilitar:

```
sudo systemctl daemon-reload
sudo systemctl enable --now pki-crl-rotate.timer
sudo systemctl list-timers --all | grep pki
```

Repetir el patrón para `sso-cleanup.timer` (`OnCalendar=*-*-* 03:00:00`) y `ocsp-health.timer` (`OnCalendar=*:0/5`).

### *A.7.4 Backups (Plesk Backup Manager)*

**Plesk → Herramientas → Backup Manager**:

- Periodicidad: diaria a las 04:00.
- Contenido: archivos del sitio web · bases de datos · configuración Plesk.
- Retención: 7 días local + 30 días en almacenamiento externo (S3/B2/FTP cifrado).
- Cifrar backups: activado, con passphrase guardada offline.

## A.8 Configuración de seguridad

### *A.8.1 Cabeceras HTTP*

Ya configuradas en los `.htaccess` del SSO y del PKI (los fragmentos relevantes de Apache/`.htaccess` y la CSP con nonce se documentan en el Manual de Código Relevante (Sección A.7)). Verificar:

```
curl -I https://sso.marchernandez.es/
# Debe incluir:
#   Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
#   X-Content-Type-Options: nosniff
#   X-Frame-Options: DENY
#   Referrer-Policy: strict-origin-when-cross-origin
```

### *A.8.2 Permisos finales (resumen)*

```
# Documento maestro de permisos
sudo find /var/www/vhosts/marchernandez.es/private/ -type d -exec chmod 700
{} \;
sudo find /var/www/vhosts/marchernandez.es/private/ -type f -exec chmod 600
{} \;

sudo chmod 700 /var/www/vhosts/marchernandez.es/sso.marchernandez.es/keys/
sudo chmod 600
/var/www/vhosts/marchernandez.es/sso.marchernandez.es/keys/*.pem
sudo chmod 600
/var/www/vhosts/marchernandez.es/sso.marchernandez.es/config/credentials.php

sudo chmod 700 /var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/
sudo chmod 600 /var/www/vhosts/marchernandez.es/pki.marchernandez.es/.env
sudo find /var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/ -name
'*.key' -exec chmod 600 {} \;

# Propietario uniforme
sudo chown -R marchernandez:psacln /var/www/vhosts/marchernandez.es/
```

### *A.8.3 Firewall (firewalld)*

```
sudo systemctl enable --now firewalld

# Servicios permitidos
sudo firewall-cmd --permanent --add-service=http     # 80 - OCSP/CRL/CA
sudo firewall-cmd --permanent --add-service=https    # 443 - SSO/PKI
sudo firewall-cmd --permanent --add-service=ssh      # 22 - admin
sudo firewall-cmd --permanent --add-port=8443/tcp    # Plesk panel

# Verificación
sudo firewall-cmd --reload
sudo firewall-cmd --list-all
```

**Importante:** el puerto 3306 (MariaDB) no debe estar abierto al exterior. Como el sistema usa socket UNIX local, no necesita exposición de red.

### *A.8.4 fail2ban*

Plesk incluye `fail2ban` preconfigurado para Plesk, SSH y Postfix. Para añadir protección de la API SSO:

`/etc/fail2ban/jail.d/sso-api.conf`:

```
[sso-api-bruteforce]
enabled = true
port = http,https
filter = sso-api-bruteforce
logpath = /var/www/vhosts/marchernandez.es/sso.marchernandez.es/logs/cron.log

/var/www/vhosts/marchernandez.es/logs/sso.marchernandez.es_php_error.log
maxretry = 10
findtime = 600
bantime = 3600
```

El filtro usa eventos del propio `sso_audit_log` exportados a syslog (configuración avanzada documentada en runbooks de Anexo C).

### *A.8.5 ModSecurity (opcional)*

Plesk ofrece ModSecurity con reglas OWASP CRS como extensión. Activar desde:

**Plesk → Herramientas → Configuración → Web Application Firewall (ModSecurity)**.

Ajustes recomendados:

- Modo: `Production`
- Conjunto de reglas: `OWASP ModSecurity CRS 3.x`
- Excluir patrones de la API SSO que devuelven JSON con caracteres especiales (regla CRS 942100 puede generar falsos positivos en JWT).

### *A.8.6 SELinux*

```
# Verificar modo actual
getenforce
# Idealmente: Enforcing (Plesk lo gestiona por defecto)

# Asegurar que Apache puede escribir en logs y temp
sudo semanage fcontext -a -t httpd_sys_rw_content_t
"/var/www/vhosts/marchernandez.es/pki.marchernandez.es/logs(/.*)?"
sudo semanage fcontext -a -t httpd_sys_rw_content_t
"/var/www/vhosts/marchernandez.es/pki.marchernandez.es/temp(/.*)?"
sudo semanage fcontext -a -t httpd_sys_rw_content_t
"/var/www/vhosts/marchernandez.es/crl.marchernandez.es(/.*)?"
sudo restorecon -Rv
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/logs/
sudo restorecon -Rv
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/temp/
sudo restorecon -Rv /var/www/vhosts/marchernandez.es/crl.marchernandez.es/

# Las claves CA deben permanecer en httpd_sys_content_t (sólo lectura por
Apache)
sudo semanage fcontext -a -t httpd_sys_content_t
"/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca(/.*)?"
sudo restorecon -Rv /var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/
```

## A.9 Pruebas post-instalación

Ejecutar este *smoke test* tras completar A.1-A.8 antes de considerar el sistema operativo.

### *A.9.1 Discovery OIDC*

```
curl -s https://sso.marchernandez.es/.well-known/openid-configuration | jq .
```

**Esperado:** JSON con `issuer`, `authorization_endpoint`, `token_endpoint`, `userinfo_endpoint`, `jwks_uri`. Códig HTTP 200.

### *A.9.2 JWKS*

```
curl -s https://sso.marchernandez.es/api/jwks.php | jq '.keys[0] | {kty, use,
alg, kid}'
```

**Esperado:** un elemento con `kty: "RSA"`, `use: "sig"`, `alg: "RS256"`, `kid` no vacío.

### *A.9.3 Login del SSO (interactivo)*

1. Abrir `https://sso.marchernandez.es/login.php`.
2. Usuario `admin`, contraseña obtenida en A.4.4.

3. Tras login, debe redirigir a `/dashboard.php` y mostrar el panel admin.
4. Cambiar inmediatamente la contraseña desde `/profile.php`.
5. Activar TOTP y escanear el código QR con Google Authenticator/Authy.
6. Guardar los 10 códigos de recuperación en un gestor de secretos.

### *A.9.4 Login del PKI vía SSO*

1. Abrir `https://pki.marchernandez.es/`.
2. Click en Iniciar sesión con MaHerMo SSO.
3. Debe redirigir a `https://sso.marchernandez.es/api/authorize.php?...&code_challenge=...` (verificar que se envía PKCE en la query, no sólo el `client_id`).
4. Tras autenticarse, debe volver a PKI y mostrar el dashboard.
5. Verificar en la BD del SSO:

```
SELECT event_type, severity, created_at FROM sso_audit_log
ORDER BY id DESC LIMIT 5;
-- Debe contener: 'authorization_code_issued' y
'token_exchange_success'
```

### *A.9.5 Emisión de un certificado de prueba*

1. `https://pki.marchernandez.es/request.php`.
2. Rellenar el formulario (CN, organización, plantilla = `client-auth`, key_size = 2048).
3. Enviar. Apuntar el ID de la solicitud.
4. Iniciar sesión como admin → **Admin** → **Solicitudes** → **ID** → **Aprobar** → **Emitir**.
5. Volver al usuario, descargar el certificado en PEM.
6. Verificar:

```
openssl x509 -in mi_cert.pem -text -noout | head -50
# Comprobar:
#   - Issuer = Omnipresence TrustCA Intermediate
#   - Authority Information Access apunta a
http://ca.marchernandez.es/...
#   - CRL Distribution Points apunta a http://crl.marchernandez.es/...
#   - Key Usage / Extended Key Usage según plantilla
```

### A.9.6 OCSP

```
# Necesitas el cert emitido y el cert de la intermedia
curl -s http://ca.marchernandez.es/intermediate.crt > /tmp/intermediate.crt

openssl ocsp \
    -issuer /tmp/intermediate.crt \
    -cert mi_cert.pem \
    -url http://ocsp.marchernandez.es \
    -resp_text -noverify
```

**Esperado:** `Response verify OK` y `Cert Status: good`.

### A.9.7 CRL

```
curl -s http://crl.marchernandez.es/intermediate.crl > /tmp/intermediate.crl
openssl crl -in /tmp/intermediate.crl -inform DER -text -noout | grep -E
"Last Update|Next Update|CRL Number"
```

**Esperado:** `Next Update` futuro (< 4 h), `CRL Number` > 0.

### A.9.8 Revocación end-to-end

1. Revocar el certificado de prueba desde `https://pki.marchernandez.es/revoke.php`.
2. Forzar generación de CRL: `php cron/generate_crl.php` o esperar al siguiente tick.
3. Repetir A.9.6 → ahora debe devolver `Cert Status: revoked`.
4. Repetir A.9.7 → la CRL debe incluir el `serial_number` revocado.

## A.10 Troubleshooting

### *A.10.1 "Internal Server Error" 500 al acceder al SSO o al PKI*

```
# Logs PHP
tail -50
/var/www/vhosts/marchernandez.es/logs/sso.marchernandez.es_php_error.log
tail -50
/var/www/vhosts/marchernandez.es/logs/pki.marchernandez.es_php_error.log

# Logs Apache
sudo tail -50 /var/log/httpd/error_log
sudo tail -50 /var/www/vhosts/system/marchernandez.es/logs/error_log
```

Causas más frecuentes:

| Síntoma en log | Causa probable | Solución |
|---|---|---|
| `Class 'PDO' not found` | Extensión `pdo_mysql` no activa en este subdominio | Activar en Plesk → Configuración PHP |
| `SQLSTATE[HY000] [1045] Access denied` | Contraseña BD incorrecta o usuario sin permisos | Revisar `.env` (PKI) o `credentials.php` (SSO) |
| `Permission denied` sobre `keys/*.pem` | Permisos incorrectos o propietario distinto a Apache | `chown marchernandez:psacln && chmod 600` |
| `open_basedir restriction in effect` | Falta el directorio del SSO en `open_basedir` de cert-auth | Añadirlo en Plesk → cert-auth → PHP Settings |
| `DOCUMENT_ROOT does not contain /public` | DocumentRoot del subdominio no apunta a `/public` | Cambiar en Plesk → Subdominio → DocumentRoot |

### *A.10.2 Socket de PHP-FPM no disponible*

```
# Comprobar estado
sudo systemctl status plesk-php84-fpm
sudo systemctl restart plesk-php84-fpm

# Verificar que existe el socket del dominio
sudo ls -l /var/www/vhosts/system/marchernandez.es/php-fpm.sock
```

### *A.10.3 OpenSSL devuelve "unable to load private key"*

Síntoma típico al crear o usar una CA.

```
# Verificar que la passphrase es correcta (intentar leer la clave)
openssl rsa -in
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root/root_ca.key -
noout -check
# Pedirá passphrase. Si es correcta, no devuelve nada.
```

Si la passphrase guardada en BD no coincide con la real (típico tras restaurar un backup sin restaurar los secretos del entorno), la solución es re-cifrar con la passphrase actual desde el panel admin (**Admin → CAs → Re-cifrar clave**).

### *A.10.4 Rutas de Plesk distintas en tu despliegue*

Plesk puede usar rutas alternativas según versión y modo:

| Versión Plesk | Rutas típicas |
|---|---|
| 18.x (Linux) | `/var/www/vhosts/<dominio>/` |
| Plesk con httpdocs | `/var/www/vhosts/<dominio>/httpdocs/` |
| Plesk con CageFS | `/home/<usuario>/public_html/` |

El código del PKI usa detección dinámica en `bootstrap.php`:

```
$isProduction = (PHP_OS_FAMILY !== 'Windows' &&
is_dir('/var/www/vhosts/marchernandez.es'));
```

Si tu Plesk usa rutas distintas, ajustar las constantes en `pki/bootstrap.php`.

### *A.10.5 Error "JWT signature verification failed" al hacer login en una app cliente*

Causas más frecuentes:

1. El cliente está usando una versión cacheada antigua del JWKS y no acepta el `kid` actual. → Verificar que el cliente respeta `Cache-Control: max-age=3600` y rota a tiempo.
2. La clave privada del fichero `.pem` y la pública en `sso_keypairs` no se corresponden (típico tras restaurar un backup parcial). → Regenerar el keypair (ver Sección A.5.6) y reiniciar `php-fpm`.

### *A.10.6 Subdomain `cert-auth` pide certificado en todas las páginas (incluyendo login normal)*

Síntoma: el navegador muestra el *popup* "Selecciona un certificado" antes de cargar el login.

Causa: el subdominio principal `sso.marchernandez.es` tiene activado `ssl_verify_client` en su VirtualHost o en las directivas Nginx. Esto no es lo deseado.

Solución: revisar **Plesk → sso.marchernandez.es → Apache y Nginx → Directivas Nginx** y eliminar cualquier línea `ssl_verify_client` o `ssl_client_certificate`. Esa configuración debe estar sólo en `cert-auth.sso.marchernandez.es`.

### *A.10.7 La CRL del cron no se regenera*

```
# Comprobar el log del cron
tail -100
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/logs/crl_cron.log

# Verificar que el cron está habilitado en Plesk
plesk db "SELECT * FROM ScheduledTasks WHERE script LIKE '%generate_crl%'"

# Forzar ejecución manual
sudo -u marchernandez php
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/cron/generate_crl.php
```

Si el log muestra `SKIP: Otra instancia ya está ejecutándose`, existe un fichero `.lock` huérfano:

```
sudo rm
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/logs/crl_generate.lock
```

### *A.10.8 Mensajes Telegram no se envían*

Comprobar:

1. `TELEGRAM_BOT_TOKEN` y `TELEGRAM_CHAT_ID` están definidos en el `.env` del PKI.
2. El bot tiene permisos para enviar mensajes al chat (probar manualmente con `curl https://api.telegram.org/bot<TOKEN>/sendMessage?chat_id=<ID>&text=test`).
3. El servidor tiene salida HTTPS al puerto 443 (firewall no bloquea egress).

### *A.10.9 Checklist final*

Antes de dar el sistema por instalado, verificar todo lo siguiente:

- ☐ Los cinco subdominios responden con código HTTP correcto (200 los públicos; 302/200 el SSO/PKI).
- ☐ Los certificados Let's Encrypt están instalados y no expirados.
- ☐ Las cabeceras HSTS, X-Frame-Options y CSP aparecen en las respuestas.
- ☐ Los cron jobs están registrados y se ejecutan (revisar `logs/crl_cron.log` y `logs/cron.log` tras 4 h y 24 h respectivamente).
- ☐ La contraseña inicial del admin se ha cambiado y TOTP está activo.
- ☐ El `install.php` se ha eliminado del servidor.
- ☐ Los backups de Plesk están activos y se han probado restaurando uno en un VPS espejo, no en producción.
- ☐ Existe documentación interna con todas las contraseñas en un gestor de secretos *offline*: contraseñas de BD, passphrases de las CAs, client_secret del PKI, codes de recuperación TOTP del admin.

## A.11 Referencias

- Sección 5 *Tecnologías y herramientas* del cuerpo del TFG (justificación de versiones).
- Sección 7.10 *Despliegue, infraestructura y conexiones* (decisiones de arquitectura del VPS).
- Sección 7.4 *Bloque PKI* (jerarquía Root + intermedias).
- Sección 9.3 *Limitaciones del proyecto* (Root CA en modo de uso restringido, no estrictamente offline).
- Anexo D (API REST) - endpoints expuestos tras la instalación.
- Anexo E (esquema BD) - estructura de tablas creadas por `install.php`.
- Anexo F (plantillas OpenSSL) - ficheros `.cnf` referenciados en Sección A.6.
- Manual de Código Relevante (Sección A.7) - fragmentos relevantes de configuración Apache / `.htaccess` y cabeceras HTTP de seguridad (CSP con *nonce*, HSTS, XFO, etc.).