TRABAJO FIN DE GRADO

UCAM
UNIVERSIDAD CATÓLICA
DE MURCIA

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

IMPLEMENTACIÓN DE UNA INFRAESTRUCTURA PKI CON OCSP/CRL Y SSO PARA LA GESTIÓN DE IDENTIDADES DIGITALES

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

IMPLEMENTACIÓN DE UNA INFRAESTRUCTURA PKI CON OCSP/CRL Y SSO PARA LA GESTIÓN DE IDENTIDADES DIGITALES

Autor:

D. Marc Hernández Montesinos

Directora:

Dra. Dña. *Angélica Guzmán Ponce*

Murcia, Junio de 2026

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

AGRADECIMIENTOS

A mis padres —María y Xisco—, mis primos, mis tíos y a mi hermano Pau, por el apoyo constante durante estos meses de trabajo intensivo.

A mis abuelos —Mari, Ricardo, Elvira y Paco—, porque detrás de cada cosa buena que hago hay un poco de ellos. Por ellos sé que merece la pena quedarse con lo pequeño y verdadero, que el cariño se demuestra más estando que diciendo, y que dar bien es dar sin contar. Cada logro que lleva mi nombre guarda un poco de su ternura, de su ejemplo y de todo lo que he aprendido simplemente mirándolos; sin saberlo, siguen escribiendo, en silencio, buena parte de quien soy.

A mis amistades, por los momentos compartidos y por los que están por venir; por sostenerme con su humor y por recordarme que la vida, afortunadamente, está hecha de los ratos que no se planifican.

A los adolescentes de la promoción 2011 que estoy acompañando como su catequista de confirmación estos últimos cinco años, por su confianza, sus preguntas y por hacerme ver, con mayor o menor ironía, que lo importante es ser coherente con lo que uno enseña.

A los peques de Scampia, porque me regalaron un hogar donde menos lo esperaba, y me enseñaron que el cariño también se aprende en otro idioma. Una parte de mi corazón sigue allí, aguardando un nuevo encuentro.

A María del Señor, por las miradas de complicidad que dicen lo que no hace falta decir, por los abrazos que enseñan a desconectar incluso cuando uno no sabe cómo, y por sostener, con cariño y sin palabras, las búsquedas que muchas veces se esconden entre líneas de código.

Y a todas las personas que, de una u otra forma, han contribuido a que este proyecto llegara hasta aquí.

LISTADO DE ABREVIATURAS

Abreviatura	Significado
ACME	Automatic Certificate Management Environment
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
AIA	Authority Information Access
APA	American Psychological Association
ASN.1	Abstract Syntax Notation One
CA	Certificate Authority
CDP	CRL Distribution Points
CGNAT	Carrier-Grade Network Address Translation
COCOMO II	Constructive Cost Model II
CRL	Certificate Revocation List
CSP	Content Security Policy
CSR	Certificate Signing Request
CSRF	Cross-Site Request Forgery
DER	Distinguished Encoding Rules
DN	Distinguished Name
DNIe	Documento Nacional de Identidad electrónico

LISTADO DE ABREVIATURAS

Abreviatura	Significado
DoS	Denial of Service
DPC	Declaración de Prácticas de Certificación
DPoP	Demonstrating Proof-of-Possession
ECDSA	Elliptic Curve Digital Signature Algorithm
eIDAS	Reglamento (UE) 910/2014 sobre identificación electrónica y servicios de confianza
EKU	Extended Key Usage
ENISA	European Union Agency for Cybersecurity
ENS	Esquema Nacional de Seguridad
ETSI	European Telecommunications Standards Institute
FNMT	Fábrica Nacional de Moneda y Timbre
GCM	Galois/Counter Mode
HMAC	Hash-based Message Authentication Code
HSM	Hardware Security Module
HSTS	HTTP Strict Transport Security
IdP	Identity Provider
IETF	Internet Engineering Task Force

LISTADO DE ABREVIATURAS

Abreviatura	Significado
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
JOSE	JSON Object Signing and Encryption
JWK	JSON Web Key
JWKS	JSON Web Key Set
JWT	JSON Web Token
kid	Key Identifier (JOSE/JWT)
KU	Key Usage
LOPDGDD	Ley Orgánica 3/2018 de Protección de Datos y garantía de los derechos digitales
MFA	Multi-Factor Authentication
mTLS	mutual Transport Layer Security
NIST	National Institute of Standards and Technology
OAuth 2.0	Open Authorization 2.0 (RFC 6749)
OCSP	Online Certificate Status Protocol
OID	Object Identifier
OIDC	OpenID Connect
OWASP	Open Worldwide Application Security Project
PEM	Privacy-Enhanced Mail

LISTADO DE ABREVIATURAS

Abreviatura	Significado
PKCE	Proof Key for Code Exchange
PKCS#11	Public-Key Cryptography Standards #11
PKCS#12	Public-Key Cryptography Standards #12
PKI	Public Key Infrastructure
PKIX	Public Key Infrastructure (X.509)
RA	Registration Authority
RBAC	Role-Based Access Control
RFC	Request for Comments
RGPD	Reglamento General de Protección de Datos (UE 2016/679)
RP	Relying Party
RPO	Recovery Point Objective
RSA	Rivest-Shamir-Adleman
RTO	Recovery Time Objective
SAML 2.0	Security Assertion Markup Language 2.0
SAN	Subject Alternative Name
SHA-256	Secure Hash Algorithm de 256 bits
S/MIME	Secure/Multipurpose Internet Mail Extensions
SQLi	SQL Injection

LISTADO DE ABREVIATURAS

Abreviatura	Significado
SSO	Single Sign-On
STRIDE	Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege
SUS	System Usability Scale
TFG	Trabajo de Fin de Grado
TLS	Transport Layer Security
TOTP	Time-based One-Time Password
TSP	Trust Service Provider
UCAM	Universidad Católica San Antonio de Murcia
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VPN	Virtual Private Network
VPS	Virtual Private Server
W3C	World Wide Web Consortium
WCAG	Web Content Accessibility Guidelines
X.509	Estándar UIT-T de certificados de clave pública
XSS	Cross-Site Scripting

ÍNDICE

RESUMEN Y ABSTRACT

Resumen

La gestión de la identidad digital es uno de los problemas técnicos con mayor impacto sobre la seguridad de los servicios telemáticos: las credenciales comprometidas aparecen como factor decisivo en aproximadamente dos de cada tres incidentes recientes (Verizon, 2024). En el ecosistema de organizaciones pequeñas, ONG y comunidades educativas que tratan datos personales persiste un hueco entre la disponibilidad de componentes técnicos abiertos y la inexistencia de un diseño integrado y reproducible que los articule sobre infraestructura propia.

A partir de ese diagnóstico, se diseñó e implementó un sistema reproducible de gestión de identidad digital sobre infraestructura libre. La arquitectura se separa en cinco subdominios independientes y despliega, sobre PHP 8.4, MariaDB 10.5 y OpenSSL 3.x en un único VPS, una autoridad certificadora de dos niveles con seis perfiles X.509 v3 conformes a la RFC 5280, servicios OCSP (RFC 6960) y CRL con rotación automatizada, y un proveedor SSO basado en OAuth 2.0 Authorization Code con PKCE (RFC 7636, método S256) y OpenID Connect. Se ofrecieron dos métodos de autenticación operativos: contraseña con segundo factor TOTP y certificado FNMT sobre TLS mutuo.

La validación se estructuró en cuatro niveles de pruebas técnicas: 29 vectores oficiales del IETF para PKCE, quince escenarios STRIDE, doce escenarios de integración y veintidós requisitos funcionales reproducibles. El estudio SUS se diseñó como protocolo formal pero no se ejecutó dentro del plazo. Las limitaciones identificadas y doce vías de trabajo futuro se documentan explícitamente.

Palabras clave: infraestructura de clave pública; Single Sign-On; OAuth 2.0; OpenID Connect; autenticación multifactor; certificados X.509.

Abstract

The management of digital identity is one of the technical problems with the greatest impact on the security of online services: compromised credentials appear as a decisive factor in roughly two out of every three recent incidents (Verizon, 2024). In the ecosystem of small organisations, NGOs and educational communities that handle personal data, there is a persistent gap between the availability of open technical components and the absence of an integrated, reproducible design articulating them on top of on-premises infrastructure.

Starting from that diagnosis, a reproducible system for digital identity management was designed and implemented on top of free and open-source infrastructure. The architecture is split across five independent subdomains and deploys, on a single virtual private server with PHP 8.4, MariaDB 10.5 and OpenSSL 3.x, a two-tier certificate authority with six X.509 v3 profiles compliant with RFC 5280, OCSP (RFC 6960) and CRL validation services with automated rotation, and an SSO provider built on the OAuth 2.0 Authorization Code flow with PKCE (RFC 7636, S256 method) and OpenID Connect. Two operational authentication methods were provided: password with TOTP second factor, and Spanish FNMT digital certificates over mutual TLS.

Validation was structured along four levels of technical testing: 29 official IETF vectors for PKCE, fifteen STRIDE scenarios, twelve end-to-end integration scenarios and twenty-two functional requirements with reproducible procedure. A SUS-based usability study was designed as a formal protocol but was not executed within the project's timeframe. The identified limitations and twelve future-work items are explicitly documented.

Keywords: public key infrastructure; single sign-on; OAuth 2.0; OpenID Connect; multi-factor authentication; X.509 certificates.

CAPÍTULO 1. INTRODUCCIÓN

1.1 Motivación y propósito

La gestión de la identidad digital se ha consolidado como uno de los problemas centrales de la seguridad de los servicios telemáticos. El *Data Breach Investigations Report 2024* de Verizon señala que aproximadamente dos de cada tres incidentes analizados implican, en algún punto de la cadena de ataque, el uso indebido de credenciales legítimas (Verizon, 2024); la Agencia de la Unión Europea para la Ciberseguridad recoge esa tendencia en sus informes anuales (ENISA, 2023). Como consecuencia, el perímetro tradicional de la seguridad se ha desplazado hacia la identidad y los modelos *Zero Trust*, en los que ningún acceso se considera implícitamente confiable, han pasado a integrar las recomendaciones oficiales (Rose et al., 2020).

Frente a este escenario conviven dos realidades. La respuesta del mercado al problema de la identidad ha tendido a la centralización en un número reducido de proveedores transnacionales que actúan como *Identity Provider* para servicios de terceros mediante federación OAuth 2.0 / OpenID Connect (Apple Inc., 2024; Google LLC, 2024; Microsoft Corporation, 2024); aunque esta delegación reduce la fricción para el usuario, también concentra la identidad efectiva de buena parte del tráfico web en pocas entidades. Las soluciones empresariales que permitirían a una organización montar su propia infraestructura (Okta Workforce Identity, Microsoft Entra ID, DigiCert ONE PKI o EJBCA Enterprise) se sitúan en rangos de coste que las hacen económicamente inviables para asociaciones, ONG, comunidades educativas, federaciones deportivas, cooperativas u otras organizaciones pequeñas que también tratan datos personales sensibles. Existe, por tanto, una grieta operativa: el conocimiento técnico necesario para construir una PKI y un sistema de inicio de sesión único es público, está documentado en estándares abiertos (Cooper et al., 2008; Hardt, 2012; Sakimura et al., 2014) y se apoya en *software* libre maduro, pero la integración de los componentes en una solución reproducible que cubra de extremo a extremo el ciclo de vida del certificado y la autenticación federada continúa siendo un ejercicio complejo cuyo resultado, cuando existe en entornos no corporativos, suele consistir en piezas aisladas.

El presente trabajo aborda ese hueco. El objetivo es demostrar que, sobre un servidor virtual privado modesto y empleando únicamente *software* libre, puede construirse una solución integrada que cubra el ciclo completo del certificado digital y, simultáneamente, ofrezca un punto único de autenticación con varios métodos de acceso. La motivación se articula en tres planos: técnico (sistematizar la integración de PKI X.509 v3, OCSP, CRL, OAuth 2.0, OpenID Connect y MFA, cuya documentación está dispersa entre estándares y manuales de productos comerciales); social (ofrecer una infraestructura reproducible para organizaciones sujetas al RGPD y al ENS sin presupuesto para soluciones propietarias); y didáctico (dejar documentada la solución como referencia para futuros estudiantes o desarrolladores).

1.2 Definición del proyecto

El proyecto consiste en el diseño y la implementación de una infraestructura de clave pública con servicios de validación en línea (OCSP, RFC 6960) y diferida (CRL, RFC 5280), integrada con un sistema de inicio de sesión único multifactor que permite a usuarios y aplicaciones autenticarse contra un único proveedor de identidad y, al tiempo, solicitar y gestionar certificados digitales X.509. La solución se articula sobre cinco subdominios independientes (`pki.marchernandez.es`, `sso.marchernandez.es`, `ocsp.marchernandez.es`, `crl.marchernandez.es` y `ca.marchernandez.es`), cada uno con una responsabilidad bien definida: portal de gestión PKI, proveedor de identidad, *responder* OCSP, distribución de CRL y repositorio público de descarga de cadenas de confianza. Esta separación permite aplicar políticas distintas de cabeceras HTTP, aislamiento de procesos y caché en cada servicio, en consonancia con las recomendaciones del CA/Browser Forum (2024) y con los principios de las RFC 5280 y 6960.

El conjunto se desarrolló en PHP 8.4 con tipado estricto y autocarga PSR-4, sobre MariaDB 10.5 con acceso exclusivo mediante PDO con consultas preparadas. La criptografía se apoya íntegramente en OpenSSL 3.x, tanto a través de la extensión nativa de PHP como mediante invocaciones a la línea de comandos. Los algoritmos seleccionados son SHA-256 para resúmenes, AES-256 para cifrado simétrico (modo GCM por su carácter AEAD; modo CBC con

verificación de integridad separada en los casos en que esa propiedad no resulta necesaria), Argon2id para el almacenamiento de contraseñas (recomendado por NIST SP 800-63B; Grassi et al., 2017) y RSA-2048 con firma RS256 para los JWT emitidos por el SSO. La longitud RSA-2048 se sitúa en el límite inferior aceptable de NIST SP 800-57 (Barker, 2020); la arquitectura admite la migración a RSA-3072, RSA-4096 o ECDSA P-256/P-384 sin cambios estructurales. El despliegue se realizó sobre un VPS con AlmaLinux 9 y Plesk Obsidian, configurado con Apache 2.4 y PHP-FPM, con la base de datos restringida a conexiones internas o por VPN. El sistema soporta dos métodos de autenticación: usuario y contraseña con segundo factor TOTP (RFC 6238) y certificado digital de la FNMT sobre TLS mutuo. Adicionalmente, la PKI emite seis perfiles distintos (*Server Authentication*, *Client Authentication*, *Code Signing*, *Document Signing*, *S/MIME* y VPN) con extensiones AIA, CDP y OCSP correctamente cumplimentadas. Quedan fuera del alcance la integración con HSM físicos, la federación con Microsoft Entra ID, el soporte del DNI electrónico, las tarjetas inteligentes con PKCS#11 y la implementación de *Passkeys* / FIDO2; estas líneas se discuten como vías futuras en el capítulo 9.

1.3 Conocimientos previos necesarios

La comprensión completa de este documento presupone familiaridad con la diferencia entre cifrado simétrico y asimétrico, con las funciones *hash*, con la firma digital y con la noción de cadena de confianza (Stallings, 2017; Katz y Lindell, 2020); con la estructura del estándar X.509 v3, los componentes PKIX y los mecanismos de revocación (Cooper et al., 2008; Adams y Lloyd, 2003); con HTTP/1.1 y HTTP/2, TLS 1.2 y 1.3 y los flujos de OAuth 2.0 y OpenID Connect (Hardt, 2012; Sakimura et al., 2014); y con PHP moderno (tipado estricto, espacios de nombres, autocarga PSR-4), SQL contra MariaDB y la línea de comandos de OpenSSL en sistemas tipo Unix. Cuando alguno de estos conceptos resulta crítico para una decisión de diseño concreta, se introduce y se explica brevemente en el cuerpo; una exposición exhaustiva queda fuera del alcance.

1.4 Objetivos

1.4.1 Objetivo general

Diseñar, implementar y validar técnicamente, sobre infraestructura íntegramente libre, un sistema integrado de gestión de identidad digital que cubra de extremo a extremo el ciclo de vida del certificado X.509 v3 (emisión, validación OCSP y CRL, revocación) y el inicio de sesión único multifactor basado en OAuth 2.0 y OpenID Connect, reproducible por una organización pequeña sin recursos para licenciar soluciones comerciales.

1.4.2 Objetivos específicos

Los objetivos específicos se enuncian conforme al criterio SMART y son comprobables contra los procedimientos del capítulo 8 y la matriz de cumplimiento del capítulo 9.

- **OE-1 (PKI: jerarquía y perfiles).** Desplegar una jerarquía con al menos una CA raíz y dos CA intermedias subordinadas conforme a RFC 5280 y CA/Browser Forum, con seis perfiles emisibles. *Indicador*: 6/6 perfiles emisibles y verificables con `openssl x509 -text`, registrados en `certificate_templates`.

- **OE-2 (PKI: protección de claves de CA).** Almacenar las claves privadas de las CA fuera de la base de datos y aplicar cifrado AES-256 sobre la *passphrase*. La raíz opera en régimen de uso restringido sin requerir un HSM físico. *Indicador*: el método `signCertificate` rechaza por código cualquier petición cuya CA emisora sea raíz; auditoría sobre el almacenamiento de `passphrase.key` (permisos `0600`, cifrado AES-256-CBC).

- **OE-3 (OCSP).** Implementar un *responder* OCSP conforme a RFC 6960 con soporte de POST y GET, respuestas firmadas con certificado dedicado (EKU `id-kp-OCSPSigning`), caché y registro auditable. *Indicador*: respuestas válidas con `openssl ocsp -respin` sobre los tres estados; tabla `ocsp_queries` con `response_time_ms` y `response_status`.

- **OE-4 (CRL).** Generar y firmar CRL nuevas cada cuatro horas y publicar inmediatamente en `crl.marchernandez.es` en DER y PEM, conservando las versiones anteriores con resumen SHA-256. *Indicador*: *timer* `systemd` activo; tabla `crl_records` con solapamiento positivo entre CRL consecutivas medible con `LAG()`.

- **OE-5 (SSO: OAuth 2.0 + OIDC).** Implementar el flujo *Authorization Code* con PKCE conforme a RFC 6749 y RFC 7636 (método obligatorio S256, *plain* rechazado), emitir JWT RS256 rotables por `kid` y exponer un *endpoint* JWKS conforme a RFC 7517. *Indicador*: 29/29 vectores oficiales de PKCE verificables con `php sso/tests/test_pkce.php`.

- **OE-6 (SSO: métodos de autenticación).** Ofrecer al menos dos métodos operativos: contraseña con segundo factor TOTP (RFC 6238) y certificado FNMT sobre mTLS con verificación cruzada de revocación contra la base de datos local.

- **OE-7 (Seguridad transversal).** Aplicar las nueve capas de seguridad recogidas en el capítulo 7 con registro de las acciones críticas en una tabla de auditoría no repudiable. *Indicador*: trazabilidad RF/STRIDE en el Anexo B; auditoría firmada accesible desde el panel.

- **OE-8 (Documentación reproducible).** Publicar el diseño y el código de forma que un tercero con conocimientos equivalentes pueda replicar la infraestructura sin licencias propietarias. *Indicador*: nueve manuales separados (instalación, administrador, usuario, API REST, modelo de datos, plantillas OpenSSL, código relevante, validación y análisis de amenazas) accesibles públicamente.

1.5 Estructura del documento

La memoria se organiza en diez capítulos (introducción, problema y contexto, estado del arte, metodologías, tecnologías y herramientas, estimación y planificación, análisis y diseño, despliegue y pruebas, conclusiones, bibliografía) y cuatro anexos (código relevante, evidencia de pruebas, evidencias visuales y declaración de uso de IA). Los manuales operativos completos (instalación, administrador, usuario, API REST, modelo de datos, plantillas OpenSSL, código

relevante, validación y análisis de amenazas) se publican como documentos separados; la sección 1.6 desarrolla la convención de cita asociada.

1.6 Documentos complementarios

El cuerpo del TFG se acompaña de nueve documentos complementarios que externalizan, por motivos de extensión, el material operativo, de validación y de referencia técnica. Cada uno se publica como fichero independiente en `https://tfg.marchernandez.es/manuales/` y se entrega también junto al depósito del TFG. Para facilitar las referencias dentro del cuerpo, cada manual recibe una sigla corta que sustituye a la URL completa en las citas; el lector que necesite consultar el documento dispone, en la Tabla 1.1, de la correspondencia entre sigla, fichero y dirección pública.

Sigla	Manual	Contenido	Archivo	URL pública
MI	Manual de Instalación	Despliegue del sistema sobre AlmaLinux 9 y Plesk: VPS, base de datos, OpenSSL, *cron*, certificados Let’s Encrypt	`Manual_Instalacion.pdf`	`https://tfg.marchernandez.es/manuales/Manual_Instalacion.pdf`
MA	Manual del Administrador	Operación diaria, *runbooks*, gestión de incidentes, rotación de claves, copias de	`Manual_Administrador.pdf`	`https://tfg.marchernandez.es/manuales/Manual_Administrador.pdf`

Sigla	Manual	Contenido	Archivo	URL pública
		seguridad y multiorganización		
MU	Manual de Usuario	Registro, inicio de sesión, gestión del perfil, ciclo de solicitud, descarga y revocación de certificados, preguntas frecuentes	`Manual_Usuario.pdf`	`https://tfg.marchernandez.es/manuales/Manual_Usuario.pdf`
MAR	Manual de la API REST	*Endpoints* SSO y PKI, autenticación, autorización y ejemplos `curl` por familia de operaciones	`Manual_API_REST.pdf`	`https://tfg.marchernandez.es/manuales/Manual_API_REST.pdf`
MMD	Modelo de Datos	DDL comentado, índices, *foreign keys*, catálogo de migraciones y diagramas entidad-relación	`Manual_Modelo_Datos.pdf`	`https://tfg.marchernandez.es/manuales/Manual_Modelo_Datos.pdf`

Sigla	Manual	Contenido	Archivo	URL pública
MPO	Manual de Plantillas OpenSSL	Los seis perfiles X.509 v3 con `KeyUsage`, `ExtendedKeyUsage`, AIA, CDP y *Certificate Policies*	`Manual_Plantillas_OpenSSL.pdf`	`https://tfg.marchernandez.es/manuales/Manual_Plantillas_OpenSSL.pdf`
MCR	Manual de Código Relevante	Catálogo completo de fragmentos de código por componente, con trazas a los ficheros del repositorio	`Manual_Codigo_Relevante.pdf`	`https://tfg.marchernandez.es/manuales/Manual_Codigo_Relevante.pdf`
MVP	Manual de Validación y Pruebas	*Checklist* completa de los veintidós requisitos funcionales (RF-01 a RF-22), las quince mitigaciones STRIDE (S-01 a S-15) y las evidencias	`Manual_Validacion.pdf`	`https://tfg.marchernandez.es/manuales/Manual_Validacion.pdf`

Sigla	Manual	Contenido	Archivo	URL pública
		visuales asociadas		
MAm	Manual de Análisis de Amenazas	Matriz STRIDE completa con las treinta y dos amenazas analizadas, sus vectores, mitigaciones y justificación del riesgo residual	`Manual_Amenazas.pdf`	`https://tfg.marchernandez.es/manuales/Manual_Amenazas.pdf`

Tabla 1.1. Documentos complementarios del TFG con sus siglas, archivos y URLs públicas.

1.6.1 Convenciones de cita

A lo largo del cuerpo del TFG los documentos complementarios se citan en formato corto, mediante la sigla seguida del apartado interno cuando procede. Tres ejemplos ilustran el uso:

Cita corta en el cuerpo	Documento al que apunta
"el procedimiento se documenta en el MA, sección C.12"	Manual del Administrador, sección C.12 (Multiorganización)
"las treinta y dos amenazas con detalle figuran en el MAm, capítulo G.3"	Manual de Análisis de Amenazas, capítulo G.3 (Spoofing)

Cita corta en el cuerpo	Documento al que apunta
“el formato de las peticiones se especifica en el MAR, sección D.4”	Manual de la API REST, sección D.4 (Endpoints PKI)

La URL completa solo se reproduce en la Tabla 1.1; ni en el cuerpo ni en los anexos se repite, salvo en la primera mención introductoria de cada manual cuando se considere oportuno por contexto.

1.6.2 Trazabilidad de la numeración interna

Cada manual conserva la letra de anexo bajo la que figuró en versiones anteriores del proyecto (A para Instalación, B para Validación, C para Administrador, D para API REST, E para Modelo de Datos, F para Plantillas OpenSSL, G para Análisis de Amenazas, H para Usuario) y la numeración de sus apartados internos (A.x, B.x, C.x, …). Esta convención facilita la trazabilidad con versiones previas del documento y con las referencias cruzadas que apuntan a los manuales desde el cuerpo del TFG.

1.6.3 Versionado

Todos los manuales se publican en la versión 1.0 alineada con la entrega del TFG. Las actualizaciones posteriores (correcciones operativas, ampliaciones de procedimientos, nuevos *runbooks*) se gestionan de forma independiente del cuerpo del TFG y mantienen la URL pública estable; el campo *Versión* y *Fecha* de la cabecera de cada manual identifica la revisión vigente.

CAPÍTULO 2. PROBLEMA Y CONTEXTO

2.1 Problema actual de la identidad digital

La identidad digital del usuario medio se ha multiplicado en las últimas dos décadas. NordPass estima que un usuario activo mantiene en torno a un centenar de cuentas en línea, lo que excede la capacidad razonable de recordar contraseñas distintas y robustas (NordPass, 2023). La consecuencia operativa, documentada por la literatura, es la reutilización masiva: una misma contraseña, o variaciones triviales sobre ella, sirve para acceder a múltiples servicios sin relación entre sí (Pearman et al., 2017). Esta reutilización se ha convertido en el vector principal de los grandes incidentes recientes: el *Data Breach Investigations Report 2024* identifica el robo o uso indebido de credenciales como factor presente en aproximadamente dos de cada tres brechas (Verizon, 2024). Una vez que un proveedor sufre una filtración, las credenciales expuestas se reutilizan mediante ataques automatizados de *credential stuffing* contra el conjunto de servicios donde el usuario podría tener cuenta.

La industria ha respondido durante la última década ofreciendo al usuario la posibilidad de reutilizar una identidad ya existente. Botones como “iniciar sesión con Google”, “con Microsoft” o “con Apple” se han generalizado en una parte muy significativa de la web; sin embargo, un número reducido de proveedores (principalmente Google, Microsoft y Apple, junto con Meta en determinados contextos) concentra hoy una parte significativa de la gestión efectiva de la identidad digital a escala global (Apple Inc., 2024; Google LLC, 2024; Microsoft Corporation, 2024). La centralización resultante introduce dos riesgos sistémicos que el modelo descentralizado anterior no presentaba: el de disponibilidad -cuando uno de estos proveedores deja de estar accesible, los servicios que delegaron en él dejan de poder autenticar a sus usuarios, como evidenció el incidente global del 19 de julio de 2024 originado por una actualización defectuosa del agente *Falcon* de CrowdStrike (CrowdStrike, 2024)- y el de soberanía -el usuario que apoya toda su vida digital sobre una única cuenta queda sometido, *de facto*, a la política comercial, técnica o jurídica del proveedor; una suspensión unilateral o un cambio en los términos del servicio puede traducirse en la pérdida simultánea del acceso a múltiples servicios sin recurso

técnico inmediato-. En el plano normativo, esta dependencia choca con la dirección que marcan el RGPD (Parlamento Europeo y Consejo, 2016) y el Reglamento (UE) 2024/1183 (“eIDAS 2”), que articula la cartera europea de identidad digital (Parlamento Europeo y Consejo, 2024): ambos instrumentos parten del principio de que el ciudadano debe gestionar su identidad con el menor número de intermediarios posible y con garantías de protección de datos.

2.2 Limitaciones de las soluciones existentes

Las alternativas que el mercado ofrece para que una organización gestione de manera autónoma la identidad digital de sus miembros se agrupan en tres familias. La primera son las plataformas comerciales de gestión de identidad (Okta Workforce, Auth0, Microsoft Entra ID, Ping Identity), que ofrecen el modelo más maduro y completo (autenticación multifactor, federación, gestión del ciclo de vida y conformidad regulatoria) pero con limitaciones económicas y de encaje: los precios escalan por usuario activo y funcionalidad, y la oferta está alineada con el caso corporativo, integrándose con Active Directory, SAML 2.0 y herramientas propietarias. Para una organización de doscientos socios con presupuesto operativo modesto, una partida fija anual a licencias resulta inviable.

La segunda familia son los proveedores de identidad de código abierto desplegables en infraestructura propia: Keycloak (Red Hat, 2025), Authentik (Goauthentik, 2025), Authelia, Gluu o el clásico Shibboleth. Resuelven con eficacia la parte de SSO y soportan los protocolos modernos (OAuth 2.0 con PKCE, OpenID Connect, SAML 2.0) y los métodos habituales, incluido TOTP. Su limitación es doble: ninguna trae integrada una PKI completa, por lo que la organización debe combinarlas con un sistema independiente para emitir certificados (S/MIME, autenticación de servidores internos, firma de documentos, VPN), multiplicando la superficie operativa; y su adopción no es trivial (Keycloak requiere Java y bases relacionales para una puesta en producción robusta, con un consumo de memoria excesivo para un VPS de gama baja).

La tercera familia son las infraestructuras de clave pública abiertas: EJBCA Community Edition (PrimeKey, 2025), Step-CA (Smallstep, 2025), OpenXPKI o conjuntos de *scripts* sobre OpenSSL. Cubren con solvencia la emisión, la revocación y la validación, pero con una limitación complementaria: ninguna

incluye un proveedor de identidad, y la integración de ambos componentes recae sobre quien despliega el sistema. Adicionalmente, las soluciones libres rara vez incorporan consideraciones específicas para el contexto al que se dirige este trabajo (aceptación nativa de certificados FNMT, interfaz en valenciano (Normas del Puig de la RACV) o italiano, conformidad explícita con el ENS o cumplimiento de los plazos de retención de la legislación española de protección de datos), por lo que requieren configuraciones a medida. El balance es que ninguna de las tres familias resuelve por sí sola el problema completo: el mercado ofrece piezas, pero no una solución integrada, ligera, libre y reproducible.

2.3 Justificación del proyecto

Del análisis anterior se deriva un hueco identificable: el del *software* libre integrado de identidad y certificados, dimensionado para organizaciones pequeñas o medianas, desplegable en un VPS modesto y documentado con el nivel de detalle necesario para que su operación no exija un equipo dedicado de ingenieros de seguridad. El presente trabajo se sitúa en ese hueco. No se persigue construir una solución de talla empresarial ni competir con los actores del mercado, sino demostrar que sobre la combinación de PHP 8.4, MariaDB 10.5, OpenSSL 3.x y AlmaLinux 9 puede construirse una infraestructura que cubra el ciclo completo de la identidad digital y de los certificados X.509 asociados, con un nivel de seguridad y trazabilidad razonable para los casos de uso previstos.

Los beneficiarios potenciales son cuatro: asociaciones culturales y comunidades educativas que necesitan ofrecer acceso autenticado a sus miembros sin licenciar plataformas comerciales; ONG y federaciones que tratan datos personales sujetos al RGPD y requieren autenticación multifactor; pequeños despachos profesionales o cooperativas que utilizan certificados FNMT y desean centralizar su validación; y entornos académicos en los que el material debe ser reproducible para fines docentes. Para todos ellos, desplegar en un único VPS un portal que ofrezca autenticación multifactor, soporte de certificados FNMT, emisión propia de certificados X.509 y servicios de validación conformes con los estándares vigentes representa un avance respecto al *statu quo*.

El aporte del proyecto no se limita al código fuente: incluye las decisiones de diseño justificadas, los esquemas de base de datos comentados, las plantillas OpenSSL para los seis perfiles, el patrón de integración OAuth 2.0 con PKCE listo para incorporar a aplicaciones PHP, los temporizadores `systemd` que automatizan la rotación de CRL, las cabeceras de seguridad aplicadas a Apache y la documentación que permite replicar el sistema en pocos días por un tercero con conocimientos equivalentes. La reproducibilidad es uno de los objetivos específicos declarados en el capítulo anterior (OE-8) y opera como hilo conductor.

El alcance de la hipótesis de partida queda acotado por dos elementos: la existencia del hueco entre las soluciones empresariales de pago y las soluciones libres dispersas se sostiene a partir de la literatura referenciada, pero no se ha sometido a una validación empírica con un panel amplio de usuarios potenciales; y el protocolo de evaluación de usabilidad descrito en el capítulo 8 se ha diseñado, pero no se ha ejecutado dentro del plazo. Los resultados disponibles permiten afirmar que el sistema funciona, satisface el modelo de seguridad descrito en el capítulo 7 y es operable, pero no que sea la solución preferible para cualquier organización en cualquier contexto. Esta limitación se recoge de forma explícita en el apartado 9.3.

CAPÍTULO 3. ESTADO DEL ARTE

Este capítulo sitúa el proyecto en el panorama tecnológico y normativo actual. No constituye un manual exhaustivo de criptografía ni de protocolos de identidad. Se consolida primero la terminología (3.1), se examinan casos reales y los problemas que motivan el proyecto (3.2-3.3), se compara el ecosistema de soluciones (3.4), se desarrolla un estudio de viabilidad (3.5) y se enumeran los estándares y normativa aplicables (3.6).

3.1 Conceptos del dominio

El proyecto se apoya en los tres bloques fundamentales de la criptografía moderna. El cifrado simétrico protege datos en reposo (por ejemplo, los secretos TOTP almacenados en la base de datos del SSO); el modo GCM ofrece cifrado autenticado en una sola operación, mientras que CBC requiere verificación de integridad por separado mediante un HMAC y se reserva para escenarios donde la autenticidad ya queda garantizada por otra capa (Stallings, 2017; Katz y Lindell, 2020). El cifrado asimétrico es la base de toda la PKI: la clave privada permanece en poder del titular y se utiliza para firmar, mientras que la clave pública (incluida en el certificado digital) permite a cualquiera verificar esa firma; los algoritmos seleccionados son RSA con tamaños mínimos de 2048 bits y, alternativamente, ECDSA sobre las curvas P-256 y P-384, recomendadas por su mejor relación seguridad/rendimiento (Barker, 2020; Adams y Lloyd, 2003). Las funciones resumen producen una huella de tamaño fijo a partir de una entrada arbitraria: SHA-256 es la elección por defecto, alineada con NIST SP 800-57 (Barker, 2020) y con las cadenas emitidas por las autoridades comerciales mayoritarias.

El estándar X.509 v3 (RFC 5280, Cooper et al., 2008) es la pieza central de la PKI moderna. Un certificado X.509 vincula una clave pública con una identidad (representada mediante un *Distinguished Name* y, opcionalmente, un conjunto de *Subject Alternative Names*) y añade extensiones que condicionan su uso: *Key Usage*, *Extended Key Usage*, *Basic Constraints*, *Subject Key Identifier*, *Authority Key Identifier* y las URL de AIA (*Authority Information Access*) y CDP (*CRL Distribution Points*). La arquitectura PKIX define el ecosistema operativo: una autoridad raíz que se mantiene en régimen de uso restringido (sin firma directa

de certificados de usuario final), una o varias autoridades intermedias que firman los certificados de usuario final, los servicios de validación OCSP y CRL, y un repositorio público desde el que descargar las cadenas. La integridad de toda la jerarquía depende, en último término, de la protección efectiva de la clave privada de la raíz.

El concepto de *Single Sign-On* se ha materializado en tres familias de protocolos: SAML 2.0, dominante durante la primera década del siglo y aún de referencia en federación académica (Fett, Küsters y Schmitz, 2017); CAS, con presencia en redes universitarias; y OAuth 2.0 (Hardt, 2012) junto con OpenID Connect (Sakimura et al., 2014), estándar *de facto* para la web y el ecosistema móvil. El presente proyecto se construye exclusivamente sobre OAuth 2.0 con PKCE y OpenID Connect, decisión justificada en el capítulo 7. En paralelo, el modelo *Zero Trust* del NIST (SP 800-207, Rose et al., 2020) ha desplazado el centro de gravedad desde el perímetro de red hacia la identidad y el contexto del acceso, lo que orienta decisiones del proyecto como el *binding* de sesión a IP y huella del agente de usuario o la auditoría exhaustiva.

Sobre la autenticación multifactor, Bonneau et al. (2012) concluyeron que ninguno de los esquemas alternativos a la contraseña ofrecía simultáneamente las propiedades de usabilidad, despliegue y seguridad de la propia contraseña; más de una década después, las contraseñas siguen siendo el primer factor mayoritario, acompañadas en buena parte de los servicios sensibles por un segundo factor (Grassi et al., 2017). Los métodos de segundo factor relevantes son TOTP (RFC 6238, M'Raihi et al., 2011), implementado en este trabajo; WebAuthn / FIDO2 (W3C, 2021; FIDO Alliance, 2022), considerado el método más resistente al *phishing* y reservado a vías futuras; y las tarjetas inteligentes con interfaz PKCS#11, de las cuales el DNI electrónico es el caso emblemático.

3.2 Casos reales de uso

Cinco casos reales aportan referencia conceptual y lecciones operativas al diseño. La FNMT-RCM (2025) opera desde mediados de los noventa una autoridad certificadora reconocida que emite certificados para personas físicas, jurídicas y administraciones; junto con el DNIe constituye el mecanismo de referencia para la administración electrónica española, y su aceptación nativa la

convierte en candidato natural a método de autenticación, aceptado por el SSO desarrollado como segundo método principal. Cl@ve (Gobierno de España, 2024) unifica varios mecanismos heterogéneos bajo una única puerta de entrada, modelo no replicado, pero sí ilustrativo. El DNI electrónico (Ministerio del Interior, 2024), introducido en España en 2006, es uno de los primeros documentos nacionales con capacidades criptográficas a escala europea; sus versiones 3.0 y 4.0 incorporan interfaz NFC y figuran entre las vías futuras del capítulo 9. Google, Microsoft y Apple operan los tres servicios de federación con mayor base de usuarios (Apple Inc., 2024; Google LLC, 2024; Microsoft Corporation, 2024) sobre OAuth 2.0 con PKCE y OpenID Connect, lo que confirma la coherencia de la elección protocolaria del proyecto; los tres respaldan desde 2022 el estándar *Passkeys*, capa de usabilidad sobre WebAuthn que reemplaza la contraseña por una credencial criptográfica sincronizada (FIDO Alliance, 2022) y que figura como vía futura. Let's Encrypt, lanzada por la *Internet Security Research Group* en 2016, transformó el ecosistema TLS al ofrecer certificados gratuitos automatizados por ACME; su especialización en certificados de servidor con validación de dominio justifica su coexistencia con otras autoridades para casos distintos: el proyecto la utiliza para el cifrado en tránsito de los cinco subdominios y opera su propia PKI para los demás perfiles. Entre los proveedores de identidad libres, Keycloak (Red Hat, 2025) es el más completo (SAML 2.0, OAuth 2.0, OIDC, federación con LDAP y AD) pero exige Java, un servidor de aplicaciones derivado de WildFly y una base de datos relacional, lo que lo hace poco adecuado para despliegues modestos; Authentik (Goauthentik, 2025) ofrece una experiencia comparable basada en Python con despliegue por contenedores; Authelia se centra en autenticación y autorización detrás de un *reverse proxy* con consumo mínimo a costa de una funcionalidad más acotada. Ninguno de los tres incluye PKI integrada.

3.3 Problemas que motivan el proyecto

El *phishing* sigue siendo el vector dominante en los incidentes con afectación masiva (Verizon, 2024; APWG, 2024); su evolución reciente combina suplantación de marca con técnicas de ingeniería social cada vez más sofisticadas. Los métodos basados en contraseñas son particularmente vulnerables, los basados en TOTP lo son en menor medida y solo WebAuthn /

FIDO2 ofrece resistencia estructural al ataque al verificar criptográficamente el origen del sitio (W3C, 2021; FIDO Alliance, 2022); esta gradación informa la decisión de exigir TOTP como segundo factor obligatorio y de incluir *Passkeys* entre las vías futuras prioritarias. Las grandes brechas de credenciales del último decenio (Yahoo, LinkedIn, Marriott, Equifax y las filtraciones agregadas conocidas como *Collection #1* o *RockYou2024*) han alimentado un ecosistema de listas de credenciales reutilizadas a bajo coste para los atacantes (Verizon, 2024), lo que refuerza la necesidad de almacenamiento robusto con Argon2id (Grassi et al., 2017), comprobación contra listas conocidas durante el alta, *rate limiting* y bloqueo automático tras varios intentos fallidos. Por último, cuando un servicio depende de un proveedor de identidad externo queda sometido a sus reglas operativas y a su política comercial: una suspensión unilateral motivada por la activación automática de un sistema antifraude o por una disputa contractual se traduce en pérdida de acceso a los servicios federados; un sistema de identidad duradero debe poder operar sin depender contractualmente de un tercero ajeno (los riesgos sistémicos de la concentración de proveedores se discuten en el apartado 2.1).

3.4 Comparativa de soluciones existentes

La Tabla 3.1 resume las características de los principales productos del dominio en seis dimensiones: licencia, integración nativa de PKI, integración nativa de SSO/IdP, soporte de autenticación multifactor, viabilidad de despliegue ligero (un único VPS con menos de 2 GiB de RAM disponibles para la aplicación) y adaptación al contexto del proyecto (soporte por defecto de FNMT, interfaz multilingüe en español, valenciano (Normas del Puig de la RACV), italiano e inglés y compatibilidad con el ENS).

Producto	Licencia	PKI	SSO/IdP	MFA	Despliegue ligero	Adaptación al contexto
Okta Workforce	Comercial (SaaS)	Limitada	Sí	Sí	No aplicable	No

Producto	Licencia	PKI	SSO/IdP	MFA	Despliegue ligero	Adaptación al contexto
Microsoft Entra ID	Comercial (SaaS)	Servicio aparte	Sí	Sí	No aplicable	No
Auth0	Comercial (SaaS)	No	Sí	Sí	No aplicable	No
DigiCert ONE PKI	Comercial	Avanzado	No	Limitado	No	No
Sectigo Certificate Manager	Comercial	Avanzado	No	Limitado	No	No
EJBCA Community	Open Source (LGPL)	Avanzado	No	No	Limitado	No
Step-CA	Open Source (Apache 2.0)	Sí	No	No	Sí	No
OpenXPKI	Open Source (Apache 2.0)	Sí	No	No	Limitado	No
Keycloak	Open Source (Apache 2.0)	No	Sí	Sí	Limitado	Limitado

Producto	Licencia	PKI	SSO/IdP	MFA	Despliegue ligero	Adaptación al contexto
Authentik	Open Source (MIT)	No	Sí	Sí	Sí	Limitado
Authelia	Open Source (Apache 2.0)	No	Limitado	Sí	Sí	Limitado

Tabla 3.1. Comparativa de productos en las seis dimensiones del análisis. ***Sí*** *= soporte completo;* ***Avanzado*** *= producto especializado;* ***Limitado*** *= soporte parcial o configuración significativa;* ***No*** *= no soportado. Datos consultados en mayo de 2026.*

El sistema desarrollado no se incluye en la tabla porque no es un producto del mercado. Sus características verificables son: jerarquía PKI de dos niveles con seis perfiles X.509 v3 conformes a RFC 5280 (sección 7.4.1), proveedor SSO con OAuth 2.0 *Authorization Code* + PKCE S256 y OpenID Connect *Discovery* (sección 7.5.1), dos métodos de autenticación operativos (contraseña con TOTP y certificado FNMT sobre mTLS), despliegue sobre un único VPS de 2 GiB de RAM con AlmaLinux 9 y Plesk Obsidian, interfaz disponible en cuatro idiomas y conformidad documentada con el ENS mediante el Anexo B.

De la lectura de la tabla se desprende que ninguno de los productos comparados satisface simultáneamente las seis dimensiones: las soluciones SaaS son funcionalmente las más completas pero su modelo de despliegue y precio las descarta; las especializadas en PKI carecen de IdP integrado; las de IdP libre carecen de PKI integrada; y ninguna trae configuración por defecto para el contexto descrito.

3.5 Estudio de viabilidad

El cierre del análisis del estado del arte exige un estudio de viabilidad conforme a la metodología Métrica v3 (Ministerio de Hacienda y Administraciones Públicas, 2001). El alcance abarca cinco subdominios funcionales (portal PKI, proveedor de identidad, *responder* OCSP, distribución de CRL y repositorio de descargas) integrados sobre una base de datos común, dos métodos de autenticación implementados y seis perfiles de certificado emitidos; quedan fuera las funcionalidades del apartado 1.2 trasladadas a vías futuras del capítulo 9. La situación de partida consiste en un VPS contratado con Ionos sobre AlmaLinux 9 y Plesk Obsidian preinstalados, el dominio `marchernandez.es` con cinco subdominios derivados, certificados Let's Encrypt y MariaDB operativa con anterioridad al desarrollo. Se evaluaron cuatro alternativas al inicio del proyecto: construir todo el sistema sobre Keycloak más EJBCA (descartada por el peso operativo combinado y la complejidad de la integración); sustituir PHP por Node.js y React (descartada por incompatibilidad con Plesk, mayor curva de despliegue y menor experiencia previa); externalizar el SSO mediante Authentik y limitar el trabajo a la PKI (descartada por reducir el alcance académico y el control sobre el sistema integrado); y construir el sistema en PHP 8.4 desde cero, alternativa elegida por ofrecer el mejor equilibrio entre control completo, viabilidad técnica, compatibilidad con la infraestructura disponible y profundidad académica.

3.6 Estándares y normativa aplicable

El sistema se ajusta a un conjunto de estándares técnicos y normas legales agrupados por dominio. En PKI: RFC 5280 sobre X.509 y PKIX (Cooper et al., 2008), RFC 6960 sobre OCSP (Santesson et al., 2013), RFC 5019 sobre el perfil ligero de OCSP y los *baseline requirements* del CA/Browser Forum (2024). En SSO e identidad: RFC 6749 (OAuth 2.0, Hardt, 2012), RFC 7636 (PKCE), RFC 7519 (JWT), OpenID Connect Core 1.0 (Sakimura et al., 2014) y RFC 8252 (OAuth en aplicaciones nativas). En autenticación multifactor: RFC 4226 (HOTP), RFC 6238 (TOTP, M'Raihi et al., 2011), WebAuthn Level 2 del W3C (2021) y CTAP 2.1 de la FIDO Alliance (2022). En el dominio normativo europeo y nacional: el Reglamento (UE) 2016/679 (RGPD, Parlamento Europeo y Consejo, 2016) y su transposición en la LOPDGDD; el Reglamento eIDAS (UE)

910/2014 y su revisión eIDAS 2 (Reglamento (UE) 2024/1183, Parlamento Europeo y Consejo, 2024) que introduce la cartera europea de identidad digital; la serie ETSI EN 319 411 sobre proveedores de servicios de confianza; y el Esquema Nacional de Seguridad regulado por el Real Decreto 311/2022. En seguridad del *software* y operación: NIST SP 800-63B (Grassi et al., 2017), NIST SP 800-207 sobre Zero Trust (Rose et al., 2020), OWASP ASVS 4.0 y OWASP Top 10 2021 (la edición 2025 se encuentra en borrador en el momento de redacción). La conformidad efectiva del sistema con cada estándar se discute, según el caso, en los capítulos 7 (diseño y seguridad) y 8 (pruebas y validación).

CAPÍTULO 4. METODOLOGÍAS DE DESARROLLO

Este capítulo describe las metodologías y prácticas de trabajo aplicadas durante el desarrollo. No persigue una exposición teórica de las distintas escuelas de ingeniería del *software*, sino documentar qué se utilizó, por qué se eligió y cómo se aplicó al caso. Las decisiones se toman con un criterio pragmático: dado el carácter individual del proyecto y la limitación de recursos humanos, se prioriza la combinación de un marco metodológico de referencia (Métrica v3) con prácticas ágiles ligeras sostenibles para una persona trabajando en solitario.

4.1 Selección de la metodología

La selección partió de una comparación entre las tres grandes familias: predictivas (cascada, ciclo en V, RUP), ágiles puras (Scrum, Kanban, XP) e híbridas (Métrica v3 con sprints ágiles, SAFe ligero). La Tabla 4.1 resume las dimensiones consideradas frente al contexto del trabajo (proyecto individual, alcance acotado y predefinido, requisitos relativamente estables, ventana temporal cerrada).

Dimensión	Predictivas	Ágiles puras	Híbridas (Métrica v3 + sprints)
Adecuación a equipo individual	Baja	Media	Alta
Trazabilidad documental para defensa académica	Alta	Baja	Alta
Tolerancia a cambios en requisitos	Baja	Alta	Media-alta
Esfuerzo de gestión	Alto	Medio	Bajo

Dimensión	Predictivas	Ágiles puras	Híbridas (Métrica v3 + sprints)
Encaje en marco administrativo español	Bajo	Bajo	Alto
Idoneidad para proyecto con alcance cerrado	Media	Alta	Alta

Tabla 4.1. Comparativa de familias metodológicas para el contexto del trabajo.

Scrum puro se descartó porque las ceremonias formales (*sprint planning*, *daily*, *retrospective*) presuponen un equipo. Kanban se descartó porque la ausencia de iteraciones cerradas dificulta la rendición de cuentas necesaria para una defensa académica. RUP y el ciclo en V se descartaron por su énfasis en una documentación predictiva extensa que no se ajusta al tamaño del proyecto. La opción adoptada combina Métrica v3 como marco general de organización (Ministerio de Hacienda y Administraciones Públicas, 2001) con sprints quincenales gestionados en Jira y prácticas de documentación gestionada como código.

4.2 Métrica v3 como marco de referencia

Métrica versión 3, referencia oficial en la administración española, ha actuado como marco general de organización del trabajo y no como guía operativa estricta. Se han utilizado de manera selectiva los procesos *Planificación de Sistemas de Información* (PSI), del que se extrae el análisis de viabilidad del capítulo 3; *Análisis del Sistema de Información* (ASI) y *Diseño del Sistema de Información* (DSI), volcados en el capítulo 7; *Construcción del Sistema de Información* (CSI), correspondiente a la fase de desarrollo iterativa; e *Implantación y Aceptación del Sistema* (IAS), reflejado en el capítulo 8. Los procesos de mantenimiento y gestión de configuración se adaptan a las prácticas de control de versiones del apartado 4.7. Los productos de gestión transversales (gestión de proyectos, aseguramiento de la calidad y seguridad) no se aplican

de forma exhaustiva, sino mediante prácticas ágiles ligeras adecuadas al tamaño del equipo.

4.3 Notación gráfica del diseño

Para fijar el diseño y razonarlo antes de la construcción se emplea notación UML estándar restringida a cuatro tipos de diagrama: casos de uso agrupados por actor (usuario final, gestor de aprobación, operador de CA, administrador); clases centradas en los servicios principales (`SSOProtocol`, `JWT`, `CertificateAuth`, `IssuanceService`, `OpenSSLAdapter`); secuencia para los tres flujos críticos (inicio de sesión OAuth 2.0 con PKCE, ciclo de emisión de un certificado y consulta OCSP); y un diagrama de despliegue que representa la distribución de los cinco subdominios sobre el VPS, los límites de seguridad y los canales habilitados. Los diagramas se elaboran en Mermaid, versionable como una fuente más del repositorio, y se incluyen en el capítulo 7.

4.4 Estrategia de pruebas

La estrategia se estructura en cuatro niveles, descritos en detalle en el capítulo 8. Las pruebas unitarias se planificaron inicialmente con PHPUnit, pero el coste de adaptar el código heredado (sin contenedor de inyección de dependencias) a una arquitectura testable superaba al beneficio dentro del plazo; en su lugar se adoptaron *scripts* `php-cli` autoejecutables bajo `sso/tests/` y `pki/tests/` que invocan directamente los métodos críticos y comparan la salida contra vectores conocidos (decisión documentada en la sección 8.1.2 y reconocida como limitación L-G7). Las pruebas de integración se realizan mediante *scripts* `curl` que ejercitan los flujos OAuth 2.0 y la API REST. Las pruebas de aceptación son manuales contrastadas con un *checklist* derivado de los requisitos funcionales. Las pruebas de seguridad se apoyan en herramientas externas (`testssl.sh`, Mozilla Observatory) y en el *checklist* OWASP ASVS 4.0.

4.5 Gestión ágil con sprints en Jira

A pesar de tratarse de un proyecto individual, el desarrollo se gestiona con sprints quincenales apoyados en Jira (plan gratuito personal): la disciplina externa que impone un sprint sustituye el control habitual de un equipo cuando se trabaja en solitario, y la trazabilidad de las decisiones queda registrada en el

tablero, lo que facilita la redacción posterior de la memoria. La planificación se estructuró en tres bloques: un Sprint 0 de análisis y configuración del entorno; una fase principal de catorce sprints quincenales (S1 a S14) organizada en dos hilos parcialmente paralelos -hilo SSO (autenticación con contraseña, TOTP, certificado FNMT, panel de administración, fortalecimiento del sistema e integración OAuth 2.0) e hilo PKI (jerarquía de CA, flujo de trabajo de solicitudes, emisión, OCSP y CRL)-; y una fase de cierre de tres sprints (S15-S17) orientados a la integración SSO-PKI, las pruebas y el fortalecimiento del sistema final. Cada incidencia se etiqueta con la convención uniforme `{COMPONENTE}-{NUM}` (componentes: `SSO`, `PKI`, `OCSP-CRL`, `INFRA`, `DOC`), se clasifica por tipo (`Story`, `Task`, `Bug`) y por prioridad. Para cada sprint se mantiene un objetivo único redactado en una frase y una definición de hecho común: código versionado, pruebas básicas pasando, documentación interna mínima actualizada y revisión personal del flujo afectado en local antes del despliegue.

4.6 Documentación viva

La documentación se mantiene como código, en Markdown, dentro del propio repositorio. Esta práctica ofrece tres ventajas: las decisiones técnicas quedan registradas en el momento en que se toman, evitando la pérdida de contexto; los archivos viven en el mismo árbol del código y se versionan con Git, por lo que cualquier divergencia entre lo descrito y lo implementado es detectable; y los diagramas Mermaid embebidos son renderizables en el editor del IDE o en el navegador del repositorio sin herramientas externas. La memoria académica se redacta como capa final, consolidando, refinando y dándole forma académica al material acumulado durante el desarrollo, no sustituyéndolo.

4.7 Gestión de riesgos

El proyecto se gestiona con prácticas ágiles, pero se mantiene un registro formal de riesgos desde el Sprint 0, revisado al inicio de cada sprint: en un sistema cuya promesa de valor es la seguridad, ignorar la gestión explícita de riesgos resultaría contradictorio. La metodología es una versión simplificada de la propuesta por ISO 31000 (ISO, 2018), aplicada manualmente a una matriz 3x3 de probabilidad por impacto. Para cada riesgo se registra su categoría,

probabilidad estimada (baja, media, alta), impacto, mitigación adoptada, riesgo residual y estado actual.

Riesgo	Probabilidad	Impacto	Mitigación	Riesgo residual
Compromiso de la clave de la Root CA	Baja	Alto	Uso restringido, *passphrase* AES-256, permisos `0600`	Bajo
Pérdida del VPS por incidencia del proveedor	Baja	Alto	*Backups* cifrados, alertas Telegram, ruta de migración documentada	Bajo
Inyección SQL u otras vulnerabilidades clásicas	Media	Alto	PDO con consultas preparadas, revisión OWASP ASVS	Bajo
Caducidad inadvertida de una CA intermedia	Media	Alto	Tarea programada que verifica expiración con 60 días de antelación	Bajo
Discrepancias entre	Alta	Medio	Detección automática	Bajo

Riesgo	Probabilidad	Impacto	Mitigación	Riesgo residual
desarrollo (Windows) y producción (Linux)			de entorno en `bootstrap.php`	
Error humano en la aprobación de un certificado	Media	Medio	Trazabilidad completa en `audit_log`, revocación inmediata	Medio
Saturación o ataque DoS sobre OCSP	Media	Medio	Caché agresiva, *rate limiting*, *fail-soft policy*	Medio
Fuga de la base de datos por configuración VPN incorrecta	Baja	Alto	Acceso restringido a `127.0.0.1` o VPN	Bajo
Dependencia de servicios FNMT	Media	Bajo	Aceptación local, validación OCSP/CRL con caché propia	Bajo

Tabla 4.2. Registro principal de riesgos del proyecto y su mitigación.

La clasificación “Bajo” del riesgo residual no implica eliminación, sino reducción a un nivel aceptable conforme al contexto. La revisión periódica permitió detectar

dos riesgos no contemplados inicialmente: la divergencia de OpenSSL entre AlmaLinux 9 y las versiones de Windows (que motivó el mecanismo de detección de entorno) y la fragilidad del flujo de trabajo de aprobación con un único administrador (mitigada parcialmente con auditoría exhaustiva y registrada como vía futura en el capítulo 9).

4.8 Control de versiones y despliegue

El control de versiones se gestiona con Git con repositorio remoto privado en GitHub. La estrategia de ramas es una versión simplificada de *GitHub Flow* (Chacon y Straub, 2014): una rama `main` siempre desplegable y ramas `feature/*`, `fix/*` o `docs/*` por cada incidencia del tablero, fusionadas mediante *pull request* autorrevisado. La convención de mensajes de *commit* sigue *Conventional Commits* con prefijos `feat`, `fix`, `docs`, `refactor`, `test`, `chore` y `security`, y el alcance entre paréntesis con los identificadores del tablero (por ejemplo, `feat(sso): añadir soporte de PKCE (SSO-12)`). El `.gitignore` excluye material sensible (claves privadas, ficheros de configuración con secretos, salidas de OpenSSL con números de serie y fragmentos de auditoría que pudieran contener datos personales) y se complementa con un *pre-commit hook* obligatorio (`scripts/install-git-hooks.sh`) que rechaza por defecto cualquier *commit* con cadenas que coincidan con patrones de claves o *tokens*.

El despliegue opera con un modelo manual y deliberado, sin *pipeline* de integración continua automatizada: el alcance no justifica el coste de mantener un *pipeline*, el VPS está gestionado con Plesk (lo que introduce particularidades difíciles de automatizar) y el carácter sensible del material desplegado hace preferible un procedimiento manual con verificación posterior. El despliegue consiste en una sincronización selectiva mediante `rsync` desde la copia local hacia las rutas del VPS, seguida de verificación manual de las páginas críticas y comprobación de los *logs* de Apache y la base de datos. La ausencia de CI/CD se reconoce como limitación operativa y se incorpora a las vías futuras del capítulo 9.

CAPÍTULO 5. TECNOLOGÍAS Y HERRAMIENTAS

Este capítulo describe la pila tecnológica empleada y justifica cada elección frente a las alternativas evaluadas. La función no es enumerar componentes (para ello bastaría con un anexo) sino dejar registrado el razonamiento técnico que sostiene cada decisión. Las decisiones criptográficas concretas (modos de cifrado, longitudes de clave, algoritmos de *hashing*) se anticipan aquí en lo necesario para entender la elección de la librería de soporte; su justificación detallada se desarrolla en el capítulo 7.

5.1 Sistema operativo y entorno de ejecución

El sistema se ejecuta sobre AlmaLinux 9 (AlmaLinux OS Foundation, 2025), distribución Linux compatible binariamente con Red Hat Enterprise Linux que conserva la *Application Binary Interface* estable durante el ciclo de soporte de la versión mayor. Su elección obedece a tres motivos: la disponibilidad nativa de OpenSSL 3.0, lo que evita compilar manualmente una biblioteca criptográfica más reciente y simplifica la cadena de actualizaciones; la afinidad con el mundo empresarial (`dnf`, *targets* de SELinux, *firewalld*, *systemd*); y la maduración de AlmaLinux como heredero comunitario de CentOS tras el giro de Red Hat hacia *Stream* (Red Hat, 2020), que aporta horizonte de soporte prolongado sin coste de licencia. Las alternativas seriamente consideradas fueron Ubuntu Server LTS (descartada por el ciclo de soporte mixto y por divergencias de la familia RHEL en rutas de configuración, nomenclatura de servicios y empaquetado de OpenSSL) y Rocky Linux (prácticamente equivalente; la elección se apoyó en la mayor presencia de AlmaLinux en los catálogos de proveedores VPS al inicio del proyecto).

Sobre el sistema operativo se ejecuta Plesk Obsidian (Plesk International, 2025), panel preinstalado en la suscripción del VPS contratado con Ionos. Plesk no fue una elección estratégica sino una condición de partida; aporta tres beneficios operativos: gestión automatizada de los certificados Let’s Encrypt para los cinco subdominios, configuración aislada de cada *virtual host* con su propio usuario del sistema de ficheros e integración de un módulo *ModSecurity* con reglas básicas de OWASP CRS. El servidor web es Apache HTTPD 2.4 con PHP-FPM 8.4.17 (PHP Group, 2026): se ha mantenido Apache por defecto frente a Nginx por la

integración de las directivas `.htaccess` (útil para diferenciar políticas entre subdominios) y por la mayor abundancia de documentación específica para mTLS; la elección de PHP-FPM frente a `mod_php` responde al modelo estándar moderno (aislamiento del intérprete, ajustes por *pool* y mejor gestión de memoria bajo carga).

5.2 Lenguaje de programación y librerías

El sistema está escrito íntegramente en PHP 8.4. La elección no obedece a inercia, sino a la comparación con tres alternativas: Node.js con TypeScript (descartado por el peso de su ecosistema de dependencias y el riesgo de cadena de suministro asociado a árboles extensos -ilustrado por el incidente *event-stream* de 2018, npm Inc., 2018-, además de su mayor consumo de memoria sobre un VPS modesto), Python con Django o Flask (requiere desplegar un servidor WSGI/ASGI adicional y la integración con OpenSSL es menos directa) y Go con `net/http` (excelente técnicamente pero sin despliegue nativo en Plesk y con curva de aprendizaje que habría desplazado tiempo desde el dominio funcional al lenguaje). PHP se integra de forma nativa con Plesk, dispone de extensiones maduras para todas las primitivas necesarias (`ext-openssl`, `ext-pdo_mysql`, `ext-sodium`, `ext-mbstring`) y permite un despliegue trivial mediante sincronización de ficheros. La versión utilizada se aprovecha para escribir todo el código con `declare(strict_types=1)`, *namespaces* PSR-4 (PHP-FIG, 2014), *enums* tipados y clases `readonly` donde la inmutabilidad es deseable.

Una decisión deliberada ha sido renunciar a los grandes *frameworks* de PHP (Laravel, Symfony, CodeIgniter): minimizar la superficie de ataque, mantener control completo sobre el flujo petición/respuesta (útil en *endpoints* OAuth 2.0 donde la conformidad estricta importa más que la productividad) y reducir la sobrecarga sobre un VPS de gama baja. La gestión de dependencias mediante Composer se mantiene en mínimos: el código se apoya casi en exclusiva en las extensiones nativas, y las pocas librerías externas se utilizan solo en utilidades auxiliares (generación de códigos QR para alta de TOTP, *parsing* de respuestas OCSP en DER cuando la API nativa no resulta suficiente). Mención especial merece la decisión de implementar JWT internamente sobre `ext-openssl` en lugar de incorporar `firebase/php-jwt`: aporta control completo sobre la lógica de firma

y validación con RS256, posibilita la rotación de claves por `kid` ajustada al modelo del proyecto y reduce la superficie de dependencias en el subsistema más crítico del SSO; la implementación interna no reimplementa primitivas criptográficas, sino que compone las operaciones estándar de OpenSSL (firma RSA, codificación Base64URL, SHA-256) en la estructura definida por la RFC 7519.

5.3 Base de datos

La capa de persistencia se apoya en MariaDB 10.5 (MariaDB Foundation, 2026). La elección frente a MySQL 8 y PostgreSQL se valoró en función de la disponibilidad nativa en Plesk (donde MariaDB es la opción predeterminada), la licencia GPL frente a la doble licencia de MySQL bajo Oracle y la compatibilidad binaria con el catálogo MySQL existente. PostgreSQL habría sido una alternativa válida (tipado más estricto, transacciones complejas más robustas) pero introducir un motor adicional al provisto por Plesk supone un coste operativo no justificado por el alcance. El acceso se realiza exclusivamente mediante la extensión PDO con consultas preparadas: en todo el código del proyecto no aparece una sola consulta SQL construida por concatenación. La conectividad se restringe a conexiones internas o por VPN: el puerto 3306 no está expuesto al exterior y la base de datos escucha sobre *socket* Unix local; el acceso administrativo desde el equipo del desarrollador se canaliza por SSH con redirección de puerto.

5.4 Criptografía

La biblioteca criptográfica del proyecto es OpenSSL 3.x (OpenSSL Project Authors, 2026), accedida desde PHP por dos vías. La extensión nativa `ext-openssl` cubre la generación de pares de claves, firma y verificación de JWT, AES-256-GCM y operaciones con certificados básicos. La invocación al binario `openssl` mediante `proc_open` es necesaria para operaciones avanzadas que la API nativa de PHP no expone con la flexibilidad requerida: firma de CSR con plantillas `.cnf` específicas por perfil, generación y firma de CRL con extensiones X.509 v2 y construcción de respuestas OCSP firmadas con un certificado dedicado. Esta dualidad es deliberada: se busca el camino más simple que garantice la corrección, evitando reimplementar primitivas criptográficas en PHP.

Los algoritmos seleccionados (SHA-256, AES-256 en modos GCM y CBC, Argon2id, RSA-2048 con margen para 3072/4096 y ECDSA P-256/P-384, RS256 para JWT) se introdujeron en el capítulo 1 y se justifican en detalle en el capítulo 7. La extensión `ext-sodium` se mantiene disponible para `random_bytes` y `sodium_memcmp`, y la Web Crypto API del navegador se utiliza en cliente para validaciones previas al envío de un certificado.

5.5 Capa de presentación

La interfaz se construye con HTML5, CSS3 y JavaScript ES2022, sin depender de *frameworks* SPA (React, Vue, Angular o Svelte). La motivación es triple: superficie de ataque mínima (cada dependencia de npm en el *bundle* es una potencial vía de cadena de suministro, ilustrada por *event-stream*, *ua-parser-js*, *colors* y repetidos *typosquatting*), rendimiento sobre clientes modestos (la interfaz pesa menos de 200 KB tras minificación, sin transpilación) y control completo sobre el código que se ejecuta en el navegador (crítico cuando este maneja *tokens* de sesión, secretos TOTP en el alta y eventualmente certificados digitales). La interfaz es *responsive* con enfoque *mobile-first* y soporta cuatro idiomas (español, valenciano (Normas del Puig de la RACV), italiano e inglés), conmutables sin recarga mediante diccionarios JSON cargados bajo demanda y reescritura de los nodos `data-i18n` en el DOM.

5.6 Automatización de tareas

Las tareas periódicas se gestionan con temporizadores `systemd` (freedesktop.org, 2024) frente a `cron` por las ventajas operativas de los *timers*: registro centralizado y filtrable con `journalctl`, dependencias entre unidades, reintentos configurables y semántica clara entre fallo de unidad y fin con éxito. Las tareas operativas son cinco: rotación de CRL (`crl-rotate.timer`, cada cuatro horas, justificación en el capítulo 7); limpieza diaria de la base de datos (`cleanup.timer`, 03:00, purga sesiones expiradas, *authorization codes* consumidos y auditoría no crítica por encima del umbral de retención); comprobación de salud del *responder* OCSP (`ocsp-health.timer`, cada cinco minutos, alerta a Telegram si el servicio deja de responder); verificación de expiración de intermedias (`ca-expiration-check.timer`, diaria, aviso con 60 días de antelación); y generación de *backups* cifrados (`backup.timer`, diaria tras el *cleanup*). El detalle de cada unidad y su *service file*

se incluye en el **MI** (Manual de Instalación). El *cron* nativo de Plesk se utiliza solo para tareas que requieren el usuario web del *vhost* correspondiente (limpieza de temporales generados durante la emisión, regeneración de *thumbnails* del panel).

5.7 Seguridad transversal del entorno

A las medidas propias del *software* se superponen varias herramientas que actúan a nivel del entorno; esta combinación es tan importante como el código y forma parte de las nueve capas del modelo de seguridad del capítulo 7. Let's Encrypt (Internet Security Research Group, 2025), gestionado por Plesk, emite y renueva los certificados TLS de los cinco subdominios cada noventa días con sesenta de margen, con un *timer* de comprobación independiente que notifica si una renovación falla. fail2ban (Fail2Ban Project, 2024) vigila los *logs* de Apache (banea IP con secuencias anómalas de 401/403 típicas de fuerza bruta, o de 404 con patrones de *fuzzing*) y de SSH (intentos fallidos superados ciertos umbrales); la política se escala por reincidencia desde un cuarto de hora hasta el bloqueo permanente. ModSecurity (OWASP Foundation, 2024) opera como WAF delante de Apache con OWASP CRS a nivel paranoia bajo, complementando las validaciones del código PHP al detener ataques conocidos antes de que lleguen a la aplicación y registrarlos para identificar patrones agregados. Un bot de Telegram centraliza alertas operativas en tiempo real (intentos masivos de *login*, errores 5xx persistentes, certificados próximos a expirar y eventos críticos de auditoría) por canal privado con *token* rotable. Los respaldos se implementan en dos niveles: *snapshot* completo del VPS provisto por Ionos con retención semanal, y *backup* diario cifrado de la base de datos y los directorios críticos (`/var/pki/`, `/var/www/vhosts/marchernandez.es/private/`) con AES-256-GCM, almacenado en un destino externo cuya clave de descifrado se custodia separadamente.

5.8 Herramientas de desarrollo y asistencia

El editor principal es Visual Studio Code con Cursor (Anysphere Inc., 2025) como capa de asistencia integrada. Las extensiones empleadas son las habituales en desarrollo PHP moderno: PHP Intelephense (autocompletado y análisis estático ligero), PHP CS Fixer (formato PSR-12), GitLens (auditoría de cambios),

Mermaid Preview (validación de diagramas) y MySQL Tools (inspecciones puntuales). Las pruebas se ejecutan mediante *scripts* `php-cli` autoejecutables (la decisión de no utilizar PHPUnit se documenta en la sección 4.4 y se reconoce como limitación L-G7), las llamadas a la API REST se depuran con Postman y `curl`, el diagnóstico de certificados se apoya en el cliente OpenSSL CLI y el control de versiones se gestiona desde Git CLI sin clientes gráficos. El uso de asistencia por inteligencia artificial durante el desarrollo se declara de forma explícita en el Anexo D conforme a las recomendaciones de la séptima edición de las normas APA (American Psychological Association, 2023).

CAPÍTULO 6. ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN

Este capítulo presenta la estimación de tamaño y esfuerzo del proyecto, su planificación temporal y la valoración económica. Las magnitudes se expresan en rangos cuando la incertidumbre lo aconseja; se distingue la estimación paramétrica COCOMO II de las horas reconstruidas retrospectivamente a partir de los sprints, los entregables y la dedicación semanal. Las cifras de costes del apartado 6.6 son orientativas, basadas en los planes públicos consultados en mayo de 2026.

6.1 Tamaño real del proyecto

La medición del tamaño del *software* es notoriamente imperfecta: las líneas de código son fáciles de contar, pero no diferencian entre código denso y verboso, y los puntos función exigen un esfuerzo de análisis adicional que no se compensa con la precisión añadida en un proyecto académico individual. La aproximación adoptada combina un recuento directo de líneas con `cloc` (AlDanial, 2024) y una caracterización funcional cualitativa. El recuento se realizó al cierre de la fase principal de desarrollo excluyendo `vendor/`, `temp/` y `old/`. Los resultados se recogen en la Tabla 6.1.

Lenguaje	Líneas	Archivos
PHP 8.4 (productivo)	13 847	75
JavaScript	1 612	21
CSS	612	9
HTML	281	14
SQL	2 427	18
Configuración OpenSSL (`.cnf`)	597	7
Bash / *Shell*	217	12
Markdown (documentación viva)	6 043	18

Tabla 6.1. Recuento de líneas del proyecto (cloc, 18 de mayo de 2026).

Para los modelos paramétricos del apartado siguiente se considera código fuente productivo la suma de PHP, JavaScript, CSS, HTML y SQL (18 779 líneas); incluyendo plantillas OpenSSL y *scripts* de operación, la cifra asciende a 19 593 líneas. La franja 18-20 KSLOC utilizada en el apartado 6.2 refleja esta horquilla. Funcionalmente, el proyecto cubre los cinco subdominios descritos en capítulos anteriores, dos métodos de autenticación, seis perfiles de certificado y un panel de administración con cuatro secciones.

6.2 Estimación COCOMO II y comparación con horas reales

El modelo COCOMO II propuesto por Boehm et al. (2000) ofrece una estimación paramétrica del esfuerzo a partir del tamaño del *software* y un conjunto de factores de escala y multiplicadores. La fórmula del modelo *Early Design* es

$$\text{Esfuerzo (PM)} = A \cdot (\text{KSLOC})^{E} \cdot \prod_{i=1}^{7} \text{EM}_i$$

con A = 2,94, E = 0,91 + 0,01 · ΣSF_j, cinco factores de escala (PREC, FLEX, RESL, TEAM, PMAT) y siete multiplicadores (RCPX, RUSE, PDIF, PERS, PREX, FCIL, SCED). Los valores asignados al proyecto se sitúan en la escala estándar del modelo (Boehm et al., 2000, Tablas 2.7 y 2.8): PREC nominal (familiaridad media con PKI y SSO de la asignatura previa), FLEX alto (requisitos académicos abiertos), RESL bajo (sin proceso formal de revisión de arquitectura), TEAM extra alto (equipo unipersonal), PMAT bajo (proceso ágil ligero sin métricas formales), RCPX alto (criptografía no trivial con alcance acotado), RUSE, PDIF, PERS, FCIL y SCED nominales y PREX bajo (primera integración conjunta de todos los componentes). Con ΣSF = 17,64, el factor exponencial resulta E ≈ 1,087 y el producto de multiplicadores ∏EM ≈ 1,49; aplicando la fórmula sobre 18-20 KSLOC se obtiene un esfuerzo nominal de 96-108 personas·mes con un plazo nominal asociado de 12-14 meses. COCOMO II se utiliza aquí como referencia comparativa, no como mecanismo de validación del esfuerzo real: el modelo describe el coste de un sistema análogo en condiciones industriales.

Las horas dedicadas al proyecto se reconstruyen retrospectivamente a partir de los sprints, los entregables y la dedicación semanal aproximada. La Tabla 6.2 sintetiza los siete bloques de trabajo: análisis y estado del arte (60-80 h),

implementación SSO en seis sprints (140-180 h), implementación PKI en cinco sprints (120-160 h), servicios OCSP, CRL y repositorio CA (50-70 h), integración SSO-PKI y *hardening* (60-80 h), pruebas, depuración y documentación viva (70-100 h) y redacción de la memoria (80-120 h), con un total estimado de 580-790 horas (aproximadamente 4-5 personas·mes a tiempo completo). La discrepancia con los 96-108 PM del modelo es consistente con la literatura del propio COCOMO II, que advierte de la sobreestimación sistemática del modelo cuando se aplica a proyectos de un único desarrollador en contextos académicos o de alta reutilización (Boehm et al., 2000, capítulo 6). Cuatro factores explican la diferencia: ausencia de coordinación entre desarrolladores, inexistencia de un proceso formal de aseguramiento de la calidad y de cumplimiento regulatorio, reutilización de patrones documentados (RFC, plantillas OpenSSL, OAuth 2.0) y arquitectura modular que permite avanzar en hilos paralelos. Las dos cifras no compiten: la nominal ofrece una cota superior y un punto de comparación con la industria; la reconstruida documenta el esfuerzo realmente invertido.

6.3 Planificación temporal

La metodología empleada (sprints quincenales sobre Métrica v3, capítulo 4) no es estrictamente predictiva, por lo que la Figura 6.1 representa el calendario real seguido más que un Gantt rígido. Su inclusión facilita comprender los solapamientos entre hilos paralelos y contrastar la planificación inicial (sprints quincenales en cascada) con la efectiva (con dos sprints extendidos y la fase final de redacción solapada con el *hardening*).

Figura 6.1. Planificación temporal real del proyecto.

Los hilos SSO y PKI se desarrollan con solapamiento deliberado desde diciembre, una vez consolidada la base del SSO. La fase de pruebas integradas

se prolongó por incidencias en la emisión de CRL en formato DER y en la integración del *responder* OCSP con las CA intermedias (capítulo 8). La redacción se inició en paralelo a la fase de cierre, lo que permitió aprovechar la documentación viva acumulada durante el desarrollo. El diagrama no recoge interrupciones por estudio de otras asignaturas, fines de semana o periodos festivos.

6.4 Recursos humanos y materiales

El proyecto se ha desarrollado íntegramente por una sola persona, en condición de estudiante del Grado en Ingeniería Informática, bajo la supervisión académica de la directora. Esta condición unipersonal, ya señalada como riesgo en el capítulo 4, condiciona la planificación: ausencia de coste de coordinación, pero también ausencia de revisión por pares en la práctica diaria. Los recursos materiales empleados son el equipo personal de desarrollo (preexistente), un VPS Linux de gama baja contratado con Ionos en suscripción mensual, el dominio `marchernandez.es` con cinco subdominios (suscripción anual), certificados TLS Let's Encrypt sin coste, un *stack* de herramientas libres (VS Code, Cursor, Git, Postman, OpenSSL CLI, MariaDB Workbench), AlmaLinux 9 con Plesk Obsidian incluidos en la suscripción del VPS y acceso institucional a bases académicas, RFC y bibliografía técnica. No se ha utilizado ningún *software* propietario de pago, lo que es coherente con la posición que el proyecto sostiene desde el capítulo 2.

6.5 Valoración económica

La valoración se descompone en coste de desarrollo y coste de explotación anual. Aplicando una tarifa horaria orientativa de 25€ (perfil *junior* con conocimientos especializados en seguridad y PHP en el mercado español), las 580-790 horas se traducen en un coste de desarrollo aproximado de 14.500-19.750€, no facturado por tratarse de un trabajo académico. El coste de explotación anual integra tres conceptos: la suscripción del VPS Linux de gama baja (4 vCPU, 8 GiB de RAM y Plesk incluido) en torno a 120-180€, la renovación del dominio principal (10-15€) y unas 60 horas anuales de mantenimiento que, a la tarifa indicada, suponen alrededor de 1 500€. El coste total de explotación queda, redondeado, en 1.700-2.000€ anuales. En un proyecto académico el

concepto clásico de ROI no se aplica directamente; sí cabe identificar un retorno formativo (conocimiento reutilizable en el ejercicio profesional posterior), social (infraestructura accesible a organizaciones que de otro modo no la tendrían) y técnico (código y documentación reutilizables).

6.6 *Build vs Buy*: comparativa con soluciones comerciales

Como caso de referencia se utiliza el descrito en el capítulo 2 (organización de unos 200 usuarios con necesidad de SSO multifactor y emisión de certificados). La Tabla 6.3 resume los costes anuales estimados a partir de los planes públicos consultados en mayo de 2026.

Solución	Coste anual estimado (200 usuarios)	Cobertura
Okta Workforce Identity	4.800-36.000€	SSO/MFA completo; PKI parcial
Auth0 Professional	1.500-10.000€	SSO/MFA; sin PKI
Microsoft Entra ID P1	14.400€ (aproximadamente)	SSO/MFA
DigiCert ONE PKI	5.000-30.000€	PKI completa; sin SSO
Sectigo Certificate Manager	5.000-30.000€	PKI completa; sin SSO
EJBCA Enterprise	3.000-20.000€	PKI completa con soporte
Keycloak + EJBCA Community	0€ en licencias + horas	SSO + PKI sin soporte comercial
Sistema de este trabajo	1.700-2.000€/año + 14 500-19.750€ de desarrollo	SSO + PKI + OCSP + CRL adaptados al contexto

Tabla 6.3. Estimación orientativa de costes para 200 usuarios. Fuentes consultadas en mayo de 2026: Okta (`okta.com/pricing/`), Auth0

(auth0.com/pricing), Microsoft (microsoft.com/en/security/business/identity-access/microsoft-entra-pricing), DigiCert (digicert.com/digicert-one/pki-manager), Sectigo (sectigo.com/pricing), Keyfactor (keyfactor.com/products/ejbca-enterprise/) y Keycloak (keycloak.org/).

Para una organización del tamaño considerado, el coste anual de operar el sistema desarrollado es sensiblemente inferior al de las soluciones comerciales con cobertura comparable; la diferencia se mantiene incluso prorrateando el coste de desarrollo: en cinco años el coste acumulado del sistema propio (22.000-30.000€) sigue por debajo de la cota inferior de varias alternativas comerciales (25.000-180.000€). La comparación no es totalmente justa en el otro sentido, ya que las soluciones comerciales aportan funcionalidades ausentes (soporte profesional, certificaciones ISO 27001 o SOC 2, programa de *bug bounty*, evolución continua). Para el público objetivo del capítulo 2 (asociaciones, ONG, comunidades educativas y organizaciones pequeñas sin capacidad presupuestaria), la comparación deja de ser entre dos opciones y pasa a ser entre una solución autohospedada y la inexistencia de infraestructura propia de identidad. La decisión de construir una solución propia no se justifica por ser siempre más barata, sino por ajustarse mejor al contexto del proyecto (control, aprendizaje, reproducibilidad y adaptación a organizaciones pequeñas); para una gran corporación con presupuesto y requisitos regulatorios estrictos la elección razonable seguirá siendo una solución comercial.

CAPÍTULO 7. ANÁLISIS Y DISEÑO

Este capítulo constituye el núcleo técnico de la memoria. Frente a los capítulos anteriores, donde se discutía el contexto, las decisiones de método y la elección de la pila tecnológica, aquí se aborda qué se ha construido exactamente, cómo se ha estructurado y por qué cada componente está donde está. La extensión del capítulo y su nivel de detalle son superiores al resto, en correspondencia con el peso que tiene en la defensa del proyecto: si los anteriores justifican la pertinencia del trabajo, este justifica su solidez.

El capítulo sigue una progresión de lo general a lo particular. El apartado 7.1 expone una visión global del sistema. El apartado 7.2 traduce las necesidades identificadas en los capítulos anteriores a un conjunto cerrado de requisitos funcionales y no funcionales, formalizados mediante un diagrama UML de casos de uso. El apartado 7.3 describe la arquitectura general en cinco subdominios, con sus componentes, dependencias, modelo de datos integrado y flujo principal entre módulos. Los apartados 7.4 a 7.10 desarrollan cada bloque funcional, el modelo de seguridad transversal, el análisis de amenazas y la configuración real de despliegue.

A lo largo del capítulo, los diagramas se han limitado a aquellos que aportan información que el texto por sí solo no puede transmitir con claridad: se ha evitado la sobreproducción de diagramas decorativos, en línea con el criterio enunciado en el capítulo 4. El nivel de profundidad técnica es medio: se incluyen fragmentos breves de código o pseudocódigo cuando aclaran un punto concreto del diseño, pero los listados extensos se trasladan a los anexos.

7.1 Visión global del sistema

7.1.1 Diagrama “cajas y flechas”

La forma más rápida de entender el sistema completo es la representación gráfica de la Figura 7.1. Los rectángulos representan los cinco subsistemas del proyecto; las flechas representan las dependencias activas entre ellos. Cualquier lector puede, en unos pocos segundos, hacerse una idea correcta de cómo encajan las piezas sin necesidad de leer una sola línea técnica.

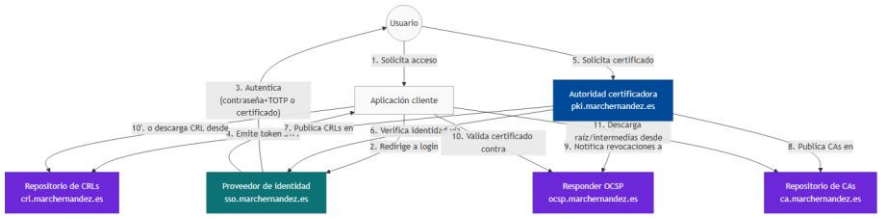

Figura 7.1. Visión global del sistema en cinco bloques.

La lectura de la figura aclara tres ideas. El sistema no es una aplicación monolítica, sino una federación de cinco servicios independientes, cada uno con su propia URL pública, su función y su *virtual host* aislado en producción; este reparto facilita el endurecimiento individual y permite escalar verticalmente el componente más cargado sin afectar al resto. Hay dos núcleos activos (SSO y PKI) y tres servicios de publicación (OCSP, CRL y CA-repo) cuya función es servir información firmada por las CA, sin lógica de negocio compleja. Por último, las flechas dibujan un grafo dirigido sin ciclos, lo que simplifica el orden de despliegue y la depuración.

7.2 Análisis de requisitos

El análisis de requisitos del proyecto se ha realizado a partir de las necesidades identificadas en los capítulos 2 (problema y contexto) y 3 (estado del arte), traducidas a un conjunto cerrado de requisitos funcionales y no funcionales. La nomenclatura adoptada es la habitual en la disciplina: RF-xx para los requisitos funcionales y RNF-xx para los no funcionales.

7.2.1 Requisitos funcionales

Los requisitos funcionales se agrupan por subsistema para facilitar la trazabilidad con los apartados siguientes de este mismo capítulo.

Subsistema de identidad (SSO). El sistema debe permitir el registro de usuarios autenticados (RF-01), el inicio de sesión mediante contraseña con segundo factor TOTP (RF-02) y mediante certificado digital de la FNMT con validación de cadena y de la *Extended Key Usage* (RF-03). Debe ofrecer un flujo OAuth 2.0 *Authorization Code* con PKCE S256 (RF-04) y un endpoint OpenID Connect *userinfo* (RF-05) para que las aplicaciones cliente puedan consultar los datos del usuario autenticado. Debe permitir el cierre de sesión de forma

centralizada (RF-06), la gestión de aplicaciones cliente desde un panel de administración (RF-07) y la consulta de la auditoría de eventos por parte de los administradores (RF-08). Las sesiones deben tener tiempo de vida configurable y deben renovarse mediante *refresh token* (RF-09).

Subsistema de PKI. El sistema debe sostener una jerarquía de CAs con una raíz operada *offline* y N CAs intermedias (RF-10), permitir la solicitud, revisión y emisión de certificados en seis perfiles distintos (RF-11), permitir la descarga del certificado emitido por parte del solicitante (RF-12) y permitir la revocación de cualquier certificado por el administrador o, en condiciones definidas, por el propio titular (RF-13). Debe registrar todas las operaciones críticas en una tabla de auditoría (RF-14) y debe ofrecer un panel de administración para revisar solicitudes pendientes, gestionar CAs, consultar certificados emitidos y operar las CRLs (RF-15).

Subsistema OCSP. El sistema debe responder a peticiones OCSP (RFC 6960) sobre cada certificado emitido (RF-16), devolviendo `good`, `revoked` o `unknown` firmado por la CA emisora o un *delegated responder* (RF-17), y debe registrar cada consulta en una tabla de métricas (RF-18).

Subsistema CRL. El sistema debe generar y publicar automáticamente CRLs por cada CA activa (RF-19), con una validez de cuatro horas y una rotación que garantice que nunca hay CRLs caducadas en circulación (RF-20). Las CRLs deben distribuirse en formatos PEM y DER desde el subdominio público (RF-21).

Subsistema CA-repo. El sistema debe ofrecer un repositorio público de descarga de la CA raíz y las CAs intermedias activas, en formatos PEM y DER (RF-22).

7.2.2 Requisitos no funcionales

Los requisitos no funcionales reflejan las propiedades transversales que el sistema debe satisfacer al margen de la funcionalidad concreta de cada subsistema.

Categoría	Requisito
Seguridad	RNF-01: cifrado en tránsito mediante TLS 1.2 o superior para todos los endpoints públicos.
	RNF-02: contraseñas almacenadas con Argon2id; nunca en claro ni con algoritmos obsoletos.
	RNF-03: protección contra CSRF en todos los formularios autenticados.
	RNF-04: validación estricta de `redirect_uri` (lista blanca por aplicación) para prevenir *Open Redirect*.
	RNF-05: *rate limiting* en endpoints sensibles (login, token, OCSP).
	RNF-06: cabeceras HTTP de seguridad (CSP, HSTS, X-Frame-Options, Referrer-Policy).
	RNF-07: cumplimiento de los principios del RGPD/LOPDGDD aplicables al alcance del proyecto.
Rendimiento	RNF-08a: respuesta del *endpoint* `/api/userinfo` y validación local de JWT por debajo de 500 ms al percentil 95 (validación en el RP sin segundo viaje al SSO).
	RNF-08b: tiempo total de inicio de sesión integrado SSO + sincronización en el RP por debajo de

Categoría	Requisito
	3 s al percentil 95 (incluye redirecciones HTTPS, validación del *userinfo* y *bootstrap* de sesión local).
	RNF-09: respuesta del *responder* OCSP por debajo de 100 ms al percentil 95.
Disponibilidad	RNF-10: objetivo de recuperación inferior a 30 minutos ante incidencias menores, mediante restauración de *backup* y reinicio de servicios; no se compromete RTO formal para incidentes mayores.
	RNF-11: ningún tiempo de CRL caducada superior a 30 minutos en condiciones nominales.
Operabilidad	RNF-12: instalación automatizable mediante el **MI** (Manual de Instalación).
	RNF-13: alertas automáticas vía Telegram para incidentes operativos críticos.
	RNF-14: rotación de logs y purga periódica de datos no críticos.
Mantenibilidad	RNF-15: código adherido a PSR-12 y PSR-4, con `declare(strict_types=1)` en todos los archivos.

Categoría	Requisito
	RNF-16: ausencia de consultas SQL construidas por concatenación de cadenas.
	RNF-17: documentación viva sincronizada con el código durante el desarrollo.
Portabilidad	RNF-18: ejecución sobre cualquier distribución Linux con PHP 8.4, MariaDB 10.5+ y OpenSSL 3.x; sin dependencias propietarias.

Tabla 7.1. Requisitos no funcionales del sistema.

7.2.3 Diagrama de casos de uso

Los actores que interactúan con el sistema son cinco. Los casos de uso se separan en dos diagramas para mantener su legibilidad: la Figura 7.2.a recoge la interacción con los subsistemas SSO y PKI mediante interfaces autenticadas, y la Figura 7.2.b recoge la interacción con los servicios públicos sin autenticación.

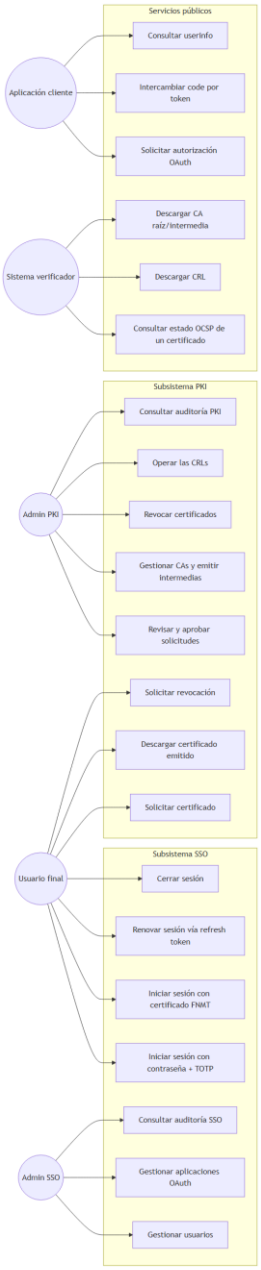

Figura 7.2.a. Diagrama UML de casos de uso para los subsistemas SSO y PKI. Los nodos en forma de cápsula representan actores, los nodos elípticos representan casos de uso y las flechas representan asociaciones entre ambos.

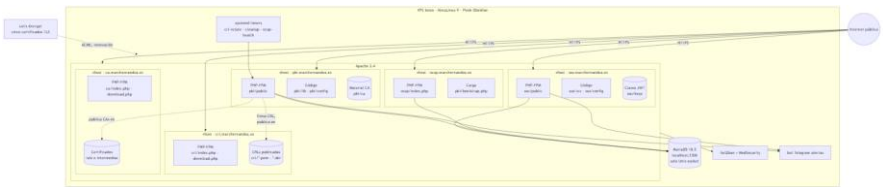

Figura 7.2.b. Diagrama UML de casos de uso para los servicios públicos. Los actores externos son no humanos: la aplicación cliente OAuth representa

cualquier relying party registrada, y el verificador externo representa cualquier consumidor de los artefactos PKI publicados.

Tres observaciones sobre los diagramas. Los administradores SSO y PKI se modelan como actores distintos porque, aunque en el despliegue actual un mismo usuario humano puede ocupar ambos roles, los permisos efectivos están separados en la base de datos y nada impide que en el futuro los asuman personas distintas. El verificador externo y la aplicación cliente OAuth corresponden a entidades de software que invocan los servicios públicos sin intervención manual. Las asociaciones se representan únicamente entre actor y caso de uso; las dependencias entre casos de uso se desarrollan en los apartados 7.4 a 7.7, donde se justifican los flujos extremo a extremo correspondientes.

7.3 Arquitectura general

7.3.1 Diagrama de despliegue de los cinco subdominios

La arquitectura física del sistema se representa en la Figura 7.3 como un diagrama UML de despliegue. El nodo exterior es el servidor virtual privado; dentro reside el entorno de ejecución Apache, que aloja los cinco *virtual hosts* del despliegue como nodos anidados. Cada *vhost* contiene los procesos PHP-FPM (componentes activos) y los artefactos de datos servidos. Las flechas continuas son invocaciones síncronas; las discontinuas, publicación o renovación asíncrona.

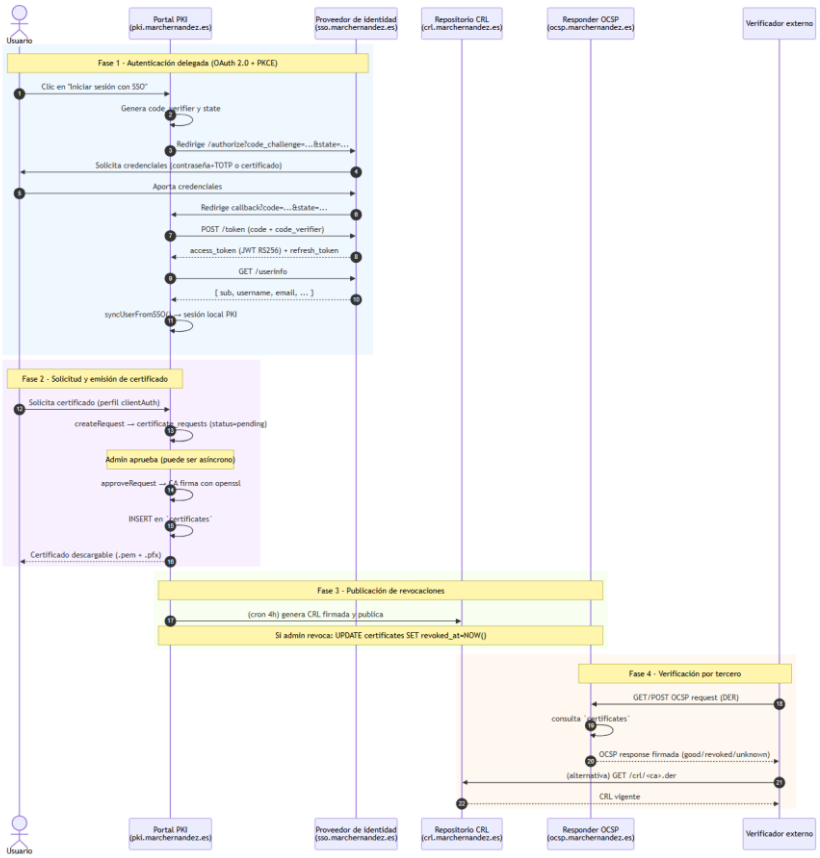

Figura 7.3. Diagrama UML de despliegue del sistema sobre el VPS de producción. El servidor físico-virtual es el nodo exterior; Apache es un entorno de ejecución que contiene los cinco vhosts; los iconos de almacenamiento representan artefactos servidos.

La figura permite identificar cinco rasgos arquitectónicos. La base de datos es una única instancia de MariaDB compartida por los tres subsistemas que la necesitan (SSO, PKI y OCSP), accesible solo desde *localhost* y desde la VPN del administrador; el puerto 3306 no está abierto al exterior. Los servicios `crl/` y `ca/` sirven artefactos estáticos firmados por las CA, lo que reduce su superficie de ataque y permite tratarlos como un repositorio público de solo lectura. Los *timers* de `systemd` orquestan las tareas periódicas (rotación de CRL cada cuatro horas, limpieza nocturna y *health check* del OCSP cada cinco minutos) en lugar del *cron* tradicional. El material de las CA reside en `pki/ca/`, fuera del *DocumentRoot* público, y solo el proceso PHP-FPM del *vhost* PKI tiene permisos de lectura sobre las claves privadas cifradas. Todos los *vhosts* delegan en Let's Encrypt la gestión de su certificado TLS, renovado cada noventa días por Plesk. Las alertas operativas se canalizan a un *bot* de Telegram que transporta

exclusivamente eventos operativos (caducidades de CA, fallos de *cron*, picos de errores HTTP 5xx, intentos de acceso anómalos) y nunca secretos, contraseñas, claves criptográficas ni datos personales sensibles; cuando la alerta requiere identificar al sujeto se utilizan identificadores internos o seudónimos (UUID de `sso_users` o de `pki_users`), nunca nombres, correos electrónicos ni DNI, lo que alinea el canal con los principios de minimización y seudonimización del RGPD y la LOPDGDD.

7.3.2 Componentes y dependencias

La Tabla 7.2 resume la composición de cada subsistema en términos de directorios y *namespaces* PHP, junto con sus dependencias activas. La intención de la tabla no es exhaustiva -el detalle se desarrolla en los apartados 7.4 a 7.7- sino servir como mapa para localizar rápidamente cualquier funcionalidad en el código.

Subsistema	Raíz en el repositorio	*Namespace* PHP	Componentes principales	Dependencias activas
SSO	`sso/`	`MaHerMoSSO\`	`SSO\SSOProtocol`, `Session\JWT`, `Session\SessionManager`, `Auth\{Password,TOTP,Certificate,SmartCard}Auth`, `Security\{CSRF,RateLimiter,SecurityHeaders,AuditLog,InputValidator}`, `Admin\{User,Admin}Manager`	MariaDB, OpenSSL, Apache (mTLS opcional)

Subsistema	Raíz en el repositorio	*Namespace* PHP	Componentes principales	Dependencias activas
PKI	`pki/`	`MaHerMo\PKI\`	`PKIEngine`, `CertificateManager`, `CRLManager`, `OCSPHandler`, `Auth`, `UserManager`, `AuditLogger`, `Database`, `View`	MariaDB, OpenSSL CLI + ext, SSO (vía OAuth 2.0 + PKCE)
OCSP	`ocsp/`	(delegado a `MaHerMo\PKI\OCSPHandler`)	`ocsp/index.php` carga `pki/bootstrap.php`	MariaDB, OpenSSL
CRL	`crl/`	(sirve estáticos)	`crl/index.php`, `crl/download.php`	Sistema de ficheros únicamente
CA-repo	`ca/`	(sirve estáticos)	`ca/index.php`, `ca/download.php`	Sistema de ficheros únicamente

Tabla 7.2. Componentes y dependencias por subsistema.

Dos observaciones se desprenden de la tabla. La primera, que la dependencia entre PKI y SSO es unidireccional: el portal PKI utiliza al SSO como proveedor de identidad para el login de los usuarios finales (paralelamente al login local que se mantiene para administradores), pero el SSO no consume nada del PKI. Esta dirección de dependencia se ha elegido para que el SSO funcione de manera autónoma y que un fallo en el PKI no comprometa la disponibilidad de la autenticación. La segunda, que los subsistemas OCSP, CRL y CA-repo son, en términos de código, prácticamente vacíos: la complejidad real reside en el subsistema PKI, que es quien genera las CRLs, atiende las consultas OCSP a

través de `OCSPHandler` y mantiene actualizado el contenido del repositorio público de CAs. Esto coincide con el principio de seguridad de separar la generación de la publicación: el material firmado se produce dentro del *vhost* protegido y se sirve desde *vhosts* casi inertes con permisos de solo lectura.

7.3.3 Modelo de datos completo

La capa de persistencia se reparte en dos bases de datos lógicas -una para el SSO y otra para el PKI- alojadas en la misma instancia de MariaDB, pero administradas por usuarios distintos con permisos disjuntos. La separación obedece a dos motivos: contención de daños en caso de compromiso de uno de los subsistemas, y claridad operativa sobre qué proceso accede a qué información.

La Tabla 7.3 resume las entidades principales. Las columnas se han omitido salvo cuando son centrales para entender el papel de la tabla; el DDL completo se incluye en el **MMD** (Manual del Modelo de Datos).

BD	Tabla	Papel	Columnas clave
SSO	`sso_users`	Usuarios del proveedor de identidad	`id` (UUID), `username`, `email`, `password_hash` (Argon2id), `totp_secret`, `role`
SSO	`sso_sessions`	Sesiones activas	`id`, `user_id`, `access_token`, `refresh_token`, `ua_fingerprint`, `expires_at`
SSO	`sso_applications`	Aplicaciones cliente OAuth registradas	`id`, `client_id`, `client_secret_hash`, `redirect_uri`, `allowed_domains`, `access_mode`

BD	Tabla	Papel	Columnas clave
SSO	`sso_application_users`	Acceso explícito de usuarios a aplicaciones restringidas	`application_id`, `user_id`
SSO	`sso_authorization_codes`	Códigos de autorización emitidos	`code` (hash SHA-256), `user_id`, `application_id`, `cert_serial`, **`code_challenge`**, **`code_challenge_method`**, `used`, `expires_at`
SSO	`sso_keypairs`	Pares de claves para firma JWT	`kid`, `public_key_pem`, `key_file_path`, `algorithm`, `is_active`
SSO	`sso_trusted_issuers`	Emisores de certificado cliente aceptados (FNMT, etc.)	`subject_dn`, `policy_oids`, `eku_oids`
SSO	`sso_user_certificates`	Certificados vinculados al usuario	`user_id`, `subject_dn`, `serial`, `issuer_dn`, `cert_type`
SSO	`sso_audit_log`	Registro inmutable de eventos	`event`, `user_id`, `ip`, `payload_json`, `created_at`

BD	Tabla	Papel	Columnas clave
SSO	`sso_rate_limits`	Contadores por IP y endpoint	`ip`, `endpoint`, `hits`, `window_start`
PKI	`pki_users`	Usuarios del portal PKI	`id`, `username`, `email`, **`sso_user_id`** (FK lógica con `sso_users.id`), `password_hash`, `role`, `created_via`
PKI	`user_sessions`	Sesiones del portal PKI	`user_id`, `session_token`, `auth_method`, `expires_at`
PKI	`certificate_authorities`	Jerarquía de CAs (raíz + intermedias)	`id`, `parent_ca_id`, `type` (root/intermediate), `subject_dn`, `key_file_path`, `passphrase_file_path`
PKI	`certificate_templates`	Perfiles emitibles (6 perfiles activos)	`id`, `slug`, `display_name`, `openssl_cnf_path`, `eku_oids`
PKI	`certificate_requests`	Solicitudes de emisión	`id`, `user_id`, `ca_id`, `template_id`, `csr_pem`, `status`
PKI	`certificates`	Certificados emitidos	`id`, `ca_id`, `serial`, `subject_dn`, `not_after`,

BD	Tabla	Papel	Columnas clave
			`revoked_at`, `revocation_reason`
PKI	`crl_records`	Histórico de CRLs publicadas	`id`, `ca_id`, `crl_number`, `entries`, `sha256`, `published_at`
PKI	`ocsp_queries`	Métricas de consultas OCSP	`ca_id`, `serial`, `status` (good/revoked/unknown), `responded_at`
PKI	`audit_log`	Auditoría del portal PKI	`user_id`, `action`, `category`, `result`, `created_at`
PKI	`pki_config`	Parámetros operativos	`config_key`, `config_value`

Tabla 7.3. Modelo de datos del sistema, agrupado por subsistema.

Tres aspectos del modelo de datos merecen un comentario explícito. En primer lugar, el vínculo entre las dos bases de datos se establece mediante la columna `pki_users.sso_user_id`, que almacena el UUID del usuario obtenido del *userinfo* del SSO. La relación es lógica y no una *foreign key* declarada en MariaDB, ya que las dos bases de datos son administrativamente independientes y el SSO puede revocar un usuario sin invalidar la integridad referencial del PKI: en ese caso, el portal PKI deja de aceptar el inicio de sesión y conserva la traza histórica del usuario. En segundo lugar, la tabla `sso_authorization_codes` incluye las columnas `code_challenge` y `code_challenge_method` que materializan el soporte de PKCE (RFC 7636) descrito en el apartado 7.5.1. En tercer lugar, las dos tablas de auditoría (`sso_audit_log` y `audit_log`) registran cada evento con marca temporal y carga útil estructurada, pero no implementan encadenamiento criptográfico de filas en el despliegue actual: la inalterabilidad efectiva se

sostiene mediante permisos a nivel de base de datos y registro de operaciones DDL, y la evolución hacia un esquema con encadenamiento criptográfico y anclado externo queda contemplada en el apartado 9.4.

7.3.4 Flujo de datos integrado entre PKI, SSO y OCSP/CRL

La interacción cruzada entre los cinco subsistemas se ilustra en la Figura 7.4 con el flujo de extremo a extremo más representativo del sistema: un usuario que inicia sesión en el portal PKI a través del SSO y, posteriormente, emite un certificado cuya validez será comprobada por una tercera parte mediante OCSP. El detalle de cada submódulo se desarrolla en los apartados siguientes; aquí se ofrece la panorámica.

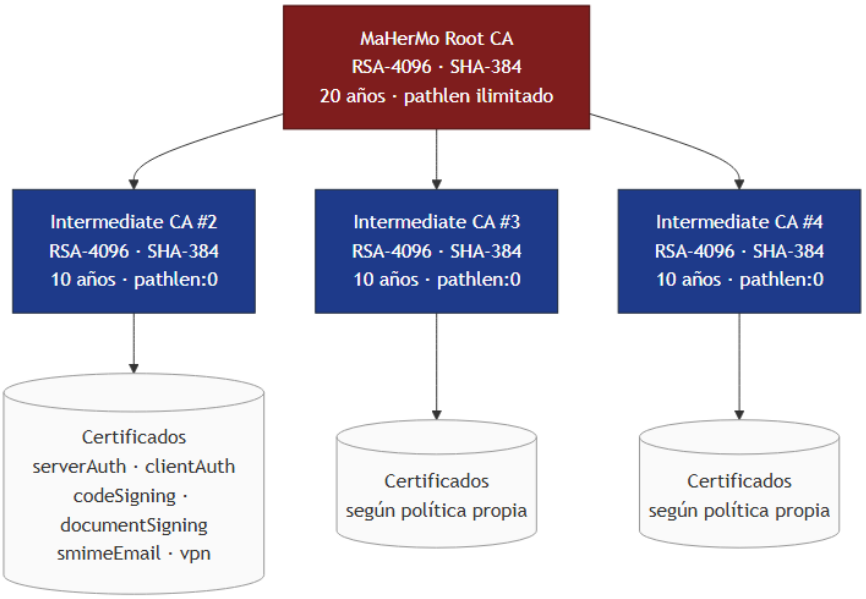

Figura 7.4. Flujo integrado de autenticación, emisión y verificación.

El diagrama deja claras dos propiedades estructurales del sistema. La fase de autenticación (1) es independiente de la fase de emisión (2): el sistema admite el funcionamiento sin SSO mediante *login* local, pero la integración con SSO permite que un usuario ya autenticado en cualquier aplicación de la organización emita certificados sin reautenticarse. La fase de verificación (4) es independiente de las tres anteriores: un tercero puede comprobar el estado de un certificado emitido sin haber participado en su solicitud, propiedad esencial de una infraestructura PKI. La publicación (fase 3) actúa como puente entre ambas mitades del sistema.

7.4 Bloque PKI (`pki.marchernandez.es`)

El subsistema PKI es, en términos de líneas de código y complejidad funcional, el componente más extenso del proyecto. Su función es triple: mantener la jerarquía de autoridades certificadoras de la organización, emitir certificados X.509 v3 a usuarios autenticados y publicar el estado de cada certificado para que cualquier tercero pueda verificarlo. Los apartados siguientes desarrollan estas funciones desde la estructura estática (jerarquía y perfiles) hasta los flujos dinámicos (emisión, almacenamiento de claves, revocación) y la configuración multiorganización.

7.4.1 Jerarquía Root offline + N intermedias

Modelo lógico

El modelo de confianza adoptado es el clásico de dos niveles, recomendado por RFC 5280 sección 3.2 y por la práctica habitual de operadores PKI maduros: una **CA raíz** que solo firma a otras CAs, y una o varias **CAs intermedias** que firman a los usuarios finales. En el despliegue actual, la jerarquía instanciada se compone de **una raíz activa y tres intermedias**, todas firmadas directamente por la raíz, según refleja la Figura 7.5.

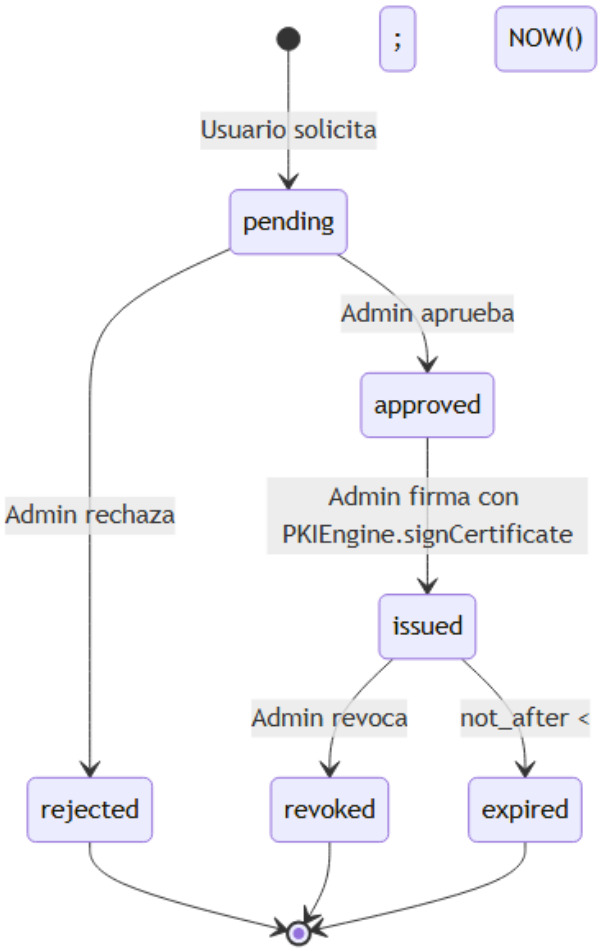

Figura 7.5. Jerarquía actual de autoridades certificadoras (1 raíz + 3 intermedias).

La elección de tres intermedias en lugar de una sola no es un capricho organizativo sino una decisión técnica: cada CA intermedia mantiene su propio conjunto de plantillas de emisión (sección 7.4.6), su propio histórico de CRLs y su propio sello en los certificados emitidos. Esto permite, sobre la misma infraestructura, atender simultáneamente a más de una organización cliente sin mezclar políticas, o segregar perfiles muy distintos en CAs independientes (por ejemplo, una intermedia exclusiva para firma de código y otra para autenticación de usuarios).

Restricciones técnicas grabadas en los certificados

La separación entre raíz e intermedias está reforzada **a** nivel de las propias extensiones X.509 v3 que el sistema graba en cada certificado, no solo a nivel de código de aplicación. La raíz lleva la extensión `basicConstraints = critical,`

`CA:true` sin restricción de longitud de cadena, mientras que cada intermedia lleva `basicConstraints = critical, CA:true, pathlen:0`. Esto evita criptográficamente que una intermedia -incluso una intermedia comprometida- pueda emitir nuevas CAs subordinadas: cualquier validador conforme a RFC 5280 sección 6.1.4 (k) rechazará una sub-CA firmada por una intermedia con `pathlen:0`, independientemente de lo que su atacante haya hecho. La protección es estructural, grabada en el propio certificado, y no depende del código de la aplicación. El fragmento (7.1) muestra el bloque de extensiones que `PKIEngine` inserta cuando se solicita la firma de una nueva intermedia, con la profundidad de cadena calculada automáticamente por el método privado `getCAChainDepth`.

```
[v3_intermediate_ca]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints       = critical, CA:true, pathlen:0
keyUsage               = critical, digitalSignature, cRLSign, keyCertSign
certificatePolicies    = @pol_section
authorityInfoAccess    =
OCSP;URI:https://ocsp.marchernandez.es,caIssuers;URI:https://ca.marchernandez
.es/intermediate_2.crt
crlDistributionPoints  = URI:http://crl.marchernandez.es/intermediate_2.crl

[pol_section]
policyIdentifier       = 2.5.29.32.0
CPS.1                  = https://pki.marchernandez.es/dpc/2
```

Fragmento 7.1. Extensiones X.509 v3 grabadas en una CA intermedia (generadas por `PKIEngine::createIntermediateSigningConfig`).

Tres aspectos del fragmento merecen comentario. La extensión `keyUsage` se marca como *critical*, lo que obliga a cualquier validador conforme a RFC 5280 a rechazar el certificado si no entiende esa extensión. Los certificados (de CA y de usuario final) llevan punteros explícitos a sus puntos de distribución de CRL (`crlDistributionPoints`) y a su *responder* OCSP (`authorityInfoAccess`), de modo que cualquier cliente que reciba uno descubre dónde verificar su estado sin configuración externa. Por último, el OID `2.5.29.32.0` (`anyPolicy`) se combina con un enlace HTTP a la Declaración de Prácticas de Certificación específica de cada CA (`pki.marchernandez.es/dpc/{id}`), donde se documentan el alcance y las limitaciones de los certificados emitidos por esa rama.

Régimen operativo de la CA raíz

La literatura de PKI suele hablar de la raíz como una entidad "*offline*", entendiendo por tal una CA alojada en un equipo físicamente desconectado de la red, que solo se enciende para firmar nuevas intermedias. En el despliegue real de este proyecto, la raíz no está físicamente desconectada -reside en el mismo VPS que el resto de la infraestructura- sino que opera bajo un régimen de uso restringido: su material criptográfico se encuentra cifrado en disco con AES-256-CBC, su passphrase no coincide con la de ninguna intermedia, y el código de `PKIEngine::signCertificate` rechaza explícitamente cualquier intento de firmar certificados de usuario final con la raíz (línea de defensa adicional al `pathlen:0` que ya bloquea la cadena criptográficamente). El fragmento 7.2 reproduce esta verificación.

```php
// PKIEngine::signCertificate, líneas 463-468
if ($ca['type'] === 'root') {
    return [
        'success' => false,
        'error' => 'La CA Raíz no puede emitir certificados de usuario.
Utilice una CA Intermedia.'
    ];
}
```

Fragmento 7.2. Restricción de seguridad: la raíz nunca firma certificados de usuario final.

Esta diferencia entre "raíz *offline* en sentido estricto" y "raíz bajo régimen restringido" se ha hecho explícita en la DPC publicada y forma parte del análisis de amenazas del apartado 7.9. En el capítulo 9 se contempla, como evolución natural del proyecto, la migración de la raíz a un host físicamente aislado o, de manera equivalente, a un módulo HSM dedicado; no se hace en el despliegue actual por motivos de coste.

7.4.2 Seis perfiles de certificado emitibles

Cada solicitud de emisión se asocia a una plantilla de certificado (`certificate_templates` en BD), que determina los usos clave permitidos, las extensiones X.509 grabadas y la validez máxima recomendada. El sistema ofrece seis perfiles operativos cubriendo los casos de uso más habituales en una PKI corporativa, recogidos en la Tabla 7.4.

slug	Nombre comercial	keyUsage	extendedKeyUsage	nsCertType	Validez
server-auth	Autenticación de Servidor (SSL/TLS)	digitalSignature, keyEncipherment	serverAuth	server	398 d
client-auth	Autenticación de Cliente (mTLS)	digitalSignature, keyAgreement	clientAuth	client, email	825 d
code-signing	Firma de Código	digitalSignature	codeSigning	objsign	1 095 d
document-signing	Firma de Documentos	digitalSignature, nonRepudiation	emailProtection	client	730 d
smime-email	Cifrado y firma S/MIME	digitalSignature, keyEncipherment	emailProtection	client, email	825 d
vpn	Acceso VPN (cliente)	digitalSignature, keyAgreement	clientAuth	client	825 d

Tabla 7.4. Plantillas de certificado emitibles. `nsCertType` se conserva por compatibilidad con clientes Netscape-style; los validadores modernos solo consultan `keyUsage` y `extendedKeyUsage`.

La validez de 398 días para `server-auth` no es arbitraria: corresponde al límite máximo aceptado por el CA/Browser Forum desde septiembre de 2020 para certificados TLS de servidor en navegadores comerciales. Ningún certificado

emitido bajo este perfil podrá superar ese plazo, aunque el solicitante pida más; el código `signCertificate` aplica el valor más bajo entre lo solicitado, lo aprobado por el administrador y el `validity_days` de la plantilla.

El conjunto de seis perfiles cubre las tres familias canónicas de uso de certificados X.509: (a) autenticación de entidades (servidor TLS, cliente TLS, VPN), (b) firma de software o documentos (firma de código y firma de documentos) y (c) cifrado y firma de correo (S/MIME). Casos de uso no contemplados que podrían añadirse simplemente insertando una nueva fila en `certificate_templates` -por ejemplo, certificados para *timestamping* (RFC 3161) o para identificación de organizaciones bajo perfiles eIDAS PSD2- quedan como evolución natural del sistema. La cláusula de extensibilidad es relevante para el TFG: añadir un séptimo perfil no requiere modificar código, solo la base de datos.

Aplicación efectiva del perfil en la firma

El método `PKIEngine::createCertSigningConfig` traduce la plantilla seleccionada a un fichero `.cnf` temporal -uno por cada emisión- que se borra inmediatamente después de la firma. La sección `[usr_cert]` resultante extiende el patrón del fragmento 7.1 con tres diferencias relevantes respecto a una intermedia: `basicConstraints = CA:FALSE` (el certificado emitido no puede a su vez firmar), `extendedKeyUsage` específico del perfil (por ejemplo, `serverAuth` para una plantilla `server-auth`) y, opcionalmente, un bloque `subjectAltName` con los DNS o IPs declarados en la solicitud. El resto de extensiones -AIA con URI OCSP y caIssuers, `crlDistributionPoints` y `certificatePolicies` con CPS- se inyectan exactamente con la misma estructura que en el fragmento 7.1, sustituyendo el OID `2.5.29.32.0` (`anyPolicy`) por el OID `1.3.6.1.4.1.55032.1.1` que identifica a la organización del despliegue. El detalle completo del `.cnf` por perfil se documenta en el **MPO** (Manual de Plantillas OpenSSL).

El OID `1.3.6.1.4.1.55032.1.1` es un identificador bajo el arco IANA Private Enterprise Number asignado a la organización del despliegue. Su uso permite distinguir, en un análisis forense o en un cliente que mantenga una *Trust Policy Database*, los certificados emitidos por esta PKI de los emitidos por otras autoridades, aunque compartan el resto de la cadena.

7.4.3 Flujo de trabajo: solicitud, emisión y descarga

El ciclo de vida completo de un certificado se modela como una máquina de estados de seis estados (Figura 7.6), gobernada por dos actores: el usuario (que inicia y descarga) y el administrador PKI (que aprueba, rechaza o revoca).

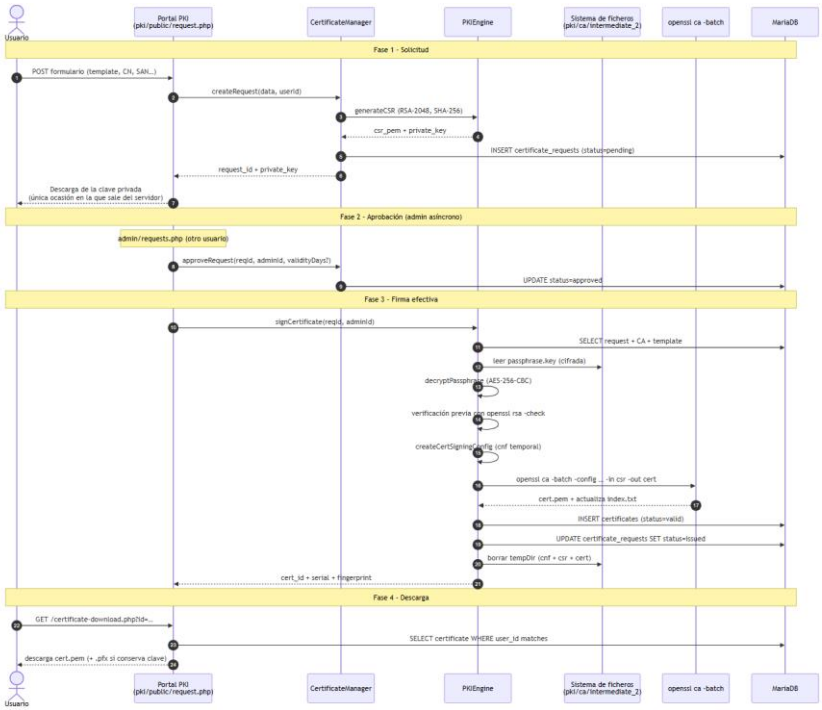

Figura 7.6. Máquina de estados de un certificado y su solicitud asociada.

La separación entre `pending` y `approved` introduce una revisión humana explícita antes de comprometer la firma de la CA: aunque el sistema podría firmar automáticamente toda solicitud bien formada, se ha optado por aprobación manual para añadir un segundo par de ojos sobre cada emisión. Aunque PHP dispone de la extensión nativa `ext-openssl`, la emisión y la gestión de CRL y respuestas OCSP se apoyan en la CLI oficial de OpenSSL invocada vía `exec()`, ya que determinadas operaciones avanzadas (manipulación de `index.txt`, coherencia entre certificados emitidos y CRL, firma de respuestas OCSP con extensiones `OCSPSigning`) no están expuestas íntegramente por la API nativa; la extensión se reserva para tareas más acotadas (parseo, *fingerprints*, generación inicial de CSR, verificación rápida de cadenas).

El flujo se desarrolla en cuatro fases. En la solicitud, el usuario completa el formulario en `pki/public/request.php`, `CertificateManager::createRequest` invoca

`PKIEngine::generateCSR` con RSA-2048 y SHA-256, se inserta el registro en `certificate_requests` con estado `pending` y la clave privada se devuelve al usuario para descarga inmediata, sin persistirse en el servidor. En la aprobación, un administrador (asíncronamente) invoca `approveRequest` desde `admin/requests.php`, que actualiza el estado a `approved`. En la firma efectiva, `signCertificate` lee la *passphrase* cifrada, la descifra con AES-256-CBC, ejecuta una verificación previa con `openssl rsa -check`, compone una configuración temporal con `createCertSigningConfig`, invoca `openssl ca -batch -config ... -in csr -out cert`, inserta la fila en `certificates`, actualiza la solicitud a `issued` y elimina el directorio temporal. En la descarga, el usuario solicita el certificado por `/certificate-download.php` y el portal sirve el `.pem` (y el `.pfx` si conserva la clave). La clave privada del solicitante solo sale del servidor en el momento de la solicitud y nunca se almacena en la base de datos; los ficheros intermedios (`*.cnf`, `*.csr`, `*.pem`) se generan en un directorio temporal con nombre aleatorio y se eliminan tras la firma; y antes de invocar a OpenSSL se ejecuta `openssl rsa -check` para detectar problemas de configuración antes de tocar el `index.txt` de la CA. El comando OpenSSL efectivo es del tipo `openssl ca -batch -utf8 -config <cnf_temporal> -extensions usr_cert -days N -md sha256 -in <csr> -out <cert.pem> -passin pass:<phrase> -cert <ca.pem> -keyfile <ca.key>`, donde `N` es el mínimo entre los días solicitados, el límite aprobado por el administrador y `template.validity_days`. La opción `-batch` deshabilita la interacción con `stdin` (necesaria para ejecutar la emisión desde PHP-FPM sin bloqueo) y `-utf8` garantiza la correcta serialización de DN con caracteres acentuados. La composición completa se incluye en el Anexo A.

7.4.4 Almacenamiento y desencriptado de claves de CA

Las claves privadas de las CA constituyen el activo criptográfico más valioso del sistema y se almacenan bajo `pki/ca/`, fuera del *DocumentRoot* público, con permisos `0600`. La protección aplica defensa en profundidad y mínimo privilegio mediante dos capas encadenadas: la clave privada se almacena cifrada con la *passphrase* mediante el cifrador por defecto de OpenSSL (AES-256-CBC), y la *passphrase* a su vez se almacena cifrada con AES-256-CBC mediante una clave maestra (`PKI_ENCRYPTION_KEY`) que reside fuera del repositorio, en el `.env` del servidor, y que se carga en memoria al arrancar PHP-FPM. La estructura

completa del directorio (ficheros `*_ca.pem`, `*.der`, `chain.pem`, `index.txt`, `serial`, `crlnumber`, `crl.pem` y `ocsp_signer.key`), los procedimientos de generación (`PKIEngine::generateRootCA`, `generateIntermediate`), descifrado (`decryptPassphrase`) y migración del formato (`reEncryptPassphrase`) se documentan en el **MA** (Manual del Administrador). El modelo protege frente a compromisos parciales (filtración de copias de seguridad, lectura no autorizada de `pki/ca/`, exposición del `.env` por separado), pero no equivale a las garantías de un HSM certificado FIPS 140-2 nivel 3 o superior; la migración a un HSM se contempla como vía evolutiva en el capítulo 9.

7.4.5 Revocación, CRL y publicación

La revocación es la operación inversa de la emisión: convierte un certificado válido en uno cuyo estado debe rechazar todo verificador. Es irreversible y debe poder ejecutarse en plazo corto, pues su demora extiende la ventana durante la cual una clave comprometida puede usarse con apariencia de validez. El sistema implementa los diez motivos definidos en RFC 5280 sección 5.3.1 (`keyCompromise`, `caCompromise`, `affiliationChanged`, `superseded`, `cessationOfOperation`, `certificateHold`, `removeFromCRL`, `privilegeWithdrawn`, `aACompromise` y `unspecified`); el motivo más sensible (`keyCompromise`) dispara una alerta operativa al canal de Telegram. El flujo es el siguiente: el administrador invoca `POST revoke(cert_id, reason)` desde el panel `pki/public/admin/certificates.php`, que delega en `PKIEngine::revokeCertificate(cert_id, reason, adminId)`; el motor actualiza el estado en BD e invoca `CRLManager::generate(ca_id, source='revocation')`, que selecciona los certificados revocados, reconstruye `index.txt`, ejecuta `openssl ca -gencrl -config ...`, registra la nueva CRL en `crl_records` y la publica copiando los ficheros `.pem` y `.der` al *vhost* `crl.marchernandez.es`. La regeneración de la CRL es automática e inmediata: si un certificado se revoca a las 14:03, a las 14:03 la CRL publicada en `crl.marchernandez.es` contiene ya la nueva entrada. Adicionalmente, un *timer* de `systemd` ejecuta `cron/generate_crl.php` cada cuatro horas para regenerar todas las CRL activas y renovar la cabecera `nextUpdate`.

El diseño combina dos magnitudes complementarias. La frecuencia de regeneración (la rotación visible al administrador) es de cuatro horas, marcada por el *timer* `pki-crl-rotate.timer` y por el parámetro aplicativo `crl.validity_hours`

`= 4`. La validez del DER firmado (el `nextUpdate` que un validador externo verá al inspeccionar el fichero `.crl`) es de veinticuatro horas, porque el comando OpenSSL `ca -gencrl -crldays N` solo acepta granularidad de días enteros. Esta asimetría es funcional: la rotación frecuente garantiza que el repositorio sirve siempre una CRL fresca, mientras que el `nextUpdate` de veinticuatro horas absorbe hasta seis ejecuciones perdidas del *timer* antes de que el sistema sirva una CRL caducada. Limitar la validez del DER a las cuatro horas reales requeriría calcular manualmente `-startdate` y `-enddate` por cada CA; la sustitución se documenta como vía futura en el apartado 9.4.

Las CRL son adecuadas para verificadores que pueden tolerar la descarga de un fichero cada pocas horas. Para casos en los que se necesita una respuesta punto a punto y en tiempo real, el sistema ofrece adicionalmente un *responder* OCSP conforme a RFC 6960 en el subdominio `ocsp.marchernandez.es`. Las respuestas OCSP no se firman con la clave de la CA, sino con un certificado *delegated responder* dedicado, generado automáticamente por la propia CA con la extensión `extendedKeyUsage = OCSPSigning` (OID `1.3.6.1.5.5.7.3.9`); Windows rechaza por diseño las respuestas OCSP firmadas directamente por la CA, exigiendo este nivel adicional de separación. La validez configurada del *responder* es de noventa días, lo que obliga a una rotación periódica que limita el impacto de un eventual compromiso de la clave del firmador. El flujo de una consulta OCSP comprende cinco pasos: el validador externo envía una petición DER a `ocsp.marchernandez.es` (POST conforme a RFC 6960 sección A.1 o GET con el DER en *base64* en el *path*); `OCSPHandler::handleRequest` extrae el *serial*, consulta la BD por la fila correspondiente y, si existe, invoca `getOCSPSignerForCA` (que regenera el *signer* si caducó) y `openssl ocsp -index ... -rsigner ... -reqin ... -respout ...` para obtener una respuesta firmada `good`, `revoked` o `unknown`; la respuesta se inserta en `ocsp_queries` con su tiempo de respuesta y se devuelve con `Content-Type: application/ocsp-response`. La inserción en `ocsp_queries` permite verificar empíricamente el cumplimiento del RNF-09 y detectar patrones anómalos (consultas masivas a un mismo *serial*).

El subsistema de revocación presenta tres limitaciones documentadas: el *responder* OCSP no implementa *OCSP stapling* (RFC 6066); las respuestas se generan en línea para cada petición sin prefirmado ni caché de servicio, lo que

añade la latencia de la firma; y, cuando el certificado del *delegated responder* no existe o no puede generarse, el código firma como *fallback* con la clave de la CA, comportamiento aceptado por OpenSSL, pero rechazado por la pila de validación de Windows. El subsistema ofrece dos vías independientes de verificación (CRL distribuible y OCSP en línea) con propagación inmediata y firma criptográfica preservada en ambos canales.

7.4.6 Configuración multiorganización

El último aspecto del subsistema PKI es la flexibilidad de configuración para adaptar el despliegue a varias organizaciones sin duplicar la infraestructura. La clave de diseño es la separación entre tres conceptos (CA emisora, plantilla de certificado y solicitud) y, en particular, la columna `ca_id` opcional de `certificate_templates`: cuando se rellena, restringe la plantilla a una CA concreta; si es `NULL`, la plantilla es emitible desde cualquier CA. Esta separación admite cuatro escenarios sobre la misma instalación (despliegue unifamiliar, segregación por uso, multiorganización y *sandbox* de pruebas) sin cambios de código. En el despliegue actual se utiliza el primer escenario; el detalle operativo de cada escenario, incluidas las consultas SQL, las configuraciones de plantilla y el *runbook* de migración entre escenarios, se externaliza al **MA**, sección C.12 Multiorganización.

7.5 Bloque SSO (`sso.marchernandez.es` - MaHerMo SSO)

El subsistema SSO es el otro pilar del proyecto. Su función es autenticar a los usuarios (por cualquiera de los métodos descritos en la sección 7.6) y federar esa autenticación hacia las aplicaciones de la organización mediante OAuth 2.0 y OpenID Connect. Frente al subsistema PKI (fabricante de evidencias criptográficas a largo plazo), el SSO es un intermediario de sesión a corto plazo: los *tokens* que emite duran minutos, no años, y su misión es facilitar que el usuario no tenga que volver a teclear su contraseña al saltar de una aplicación a otra.

7.5.1 OAuth 2.0 Authorization Code + PKCE (RFC 7636 S256) + OIDC con discovery

El estándar OAuth 2.0 (RFC 6749) define varios *flows*. El proyecto implementa únicamente el *Authorization Code Flow*, el único recomendado por la *OAuth 2.0 Security Best Current Practice* (RFC 9700) para casos de uso web. Los *flows Implicit* y *Resource Owner Password Credentials*, aún presentes en despliegues heredados, se descartan: el primero expone el token en el fragmento de la URL, vulnerable a fugas por *referrers* y por registros de servidor; el segundo obliga al cliente a manipular las credenciales del usuario en claro, lo que anula el propósito del SSO. Sobre el flujo *Authorization Code* canónico, el proyecto añade dos refuerzos consolidados en la práctica reciente: PKCE S256 (RFC 7636) para evitar la interceptación del código de autorización, y OpenID Connect Core 1.0 con *Discovery* 1.0 para que los clientes descubran automáticamente los *endpoints* del proveedor.

Flujo completo paso a paso

La Figura 7.7 representa el flujo *Authorization Code* con PKCE tal y como lo implementa el sistema, con anotaciones sobre qué endpoint del SSO interviene en cada paso. La narrativa se mantiene independiente del método de autenticación: el usuario podría haber elegido cualquiera de los de la sección 7.6 sin cambiar el resto del diagrama.

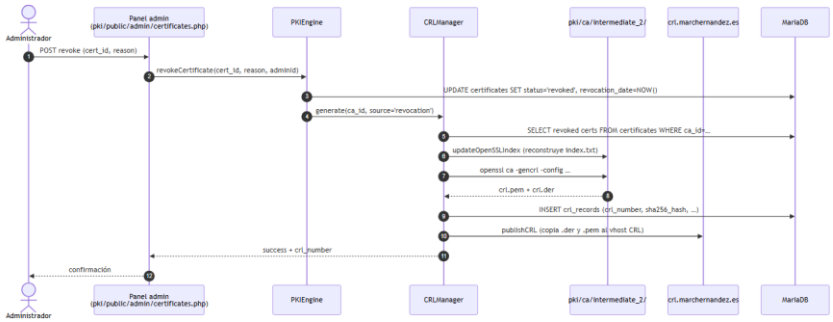

Figura 7.7. Flujo OAuth 2.0 Authorization Code + PKCE + OIDC userinfo, con los endpoints concretos del SSO.

Tres observaciones sobre el diagrama. El parámetro `state` viaja en paralelo al `code_challenge` y cumple un papel complementario: PKCE protege frente al robo del `code` en el canal de retorno, mientras que `state` protege frente a CSRF del

propio flujo. El `code` se almacena con *hash* SHA-256 en `sso_authorization_codes`, de modo que un atacante con acceso a la base de datos no puede reproducir un `code` observado. El `client_secret` se compara contra un *hash* Argon2id mediante `password_verify`, lo que impone tiempo y memoria de cómputo a cualquier intento de fuerza bruta sobre el `client_secret_hash` filtrado.

Validación efectiva de PKCE

La parte sensible del flujo es el momento en que `/api/token.php` recibe el `code_verifier` y debe decidir si coincide con el `code_challenge` registrado al emitirse el código. `SSOProtocol::verifyCodeVerifier` aplica el algoritmo en tres pasos: comprueba que el `code_verifier` tenga entre 43 y 128 caracteres y use el alfabeto BASE64URL (RFC 7636 sección 4.1); calcula `BASE64URL(SHA256(verifier))`; y compara mediante `hash_equals` para evitar canales laterales temporales. El sistema rechaza explícitamente el método `plain` definido en la RFC: el código `validatePkceParams` solo acepta `S256`, restricción alineada con la RFC 9700 sección 2.1.1. Aunque PKCE nació como protección para clientes públicos, el sistema lo soporta para cualquier cliente registrado, sea público o confidencial, porque PKCE protege contra el robo del `code` en su tránsito por el navegador, vía de ataque independiente del secreto del cliente. La evidencia experimental se documenta en el capítulo 8 con 29 casos de prueba, incluido el vector oficial de RFC 7636 sección B.1.

El descubrimiento dinámico de los *endpoints* se realiza mediante el documento `/.well-known/openid-configuration`. Su estructura publica los seis bloques canónicos (identificación del *issuer*, URL de los cuatro *endpoints* `authorization`, `token`, `userinfo` y `jwks_uri`, parámetros declarados y métodos PKCE soportados). El *endpoint* `jwks_uri` complementa el *discovery* publicando las claves públicas activas con su `kid`, lo que permite a un RP validar firmas localmente sin invocar de nuevo al SSO. La combinación de *Authorization Code* con PKCE S256, OIDC con *discovery* y JWKS sitúa al sistema al nivel de prácticas recomendadas por la *OAuth 2.0 Security BCP* y la OpenID Foundation.

7.5.2 Modelo de aplicaciones cliente

Cada aplicación que delega autenticación en el SSO se modela como una fila en la tabla `sso_applications`, con cinco campos clave que gobiernan el flujo completo. La Tabla 7.5 los recoge junto a su semántica.

Campo	Tipo	Papel
`client_id`	string	Identificador público que el RP usa en `/authorize` y `/token`.
`client_secret_hash`	string (Argon2id)	Secreto compartido cuya verificación gobierna el acceso al *endpoint* de token.
`redirect_uri`	string	URI canónica de retorno tras autorización (exact-match en `/token`).
`allowed_domains`	string	Dominios bajo los que se aceptan `redirect_uri` y `post_logout_redirect_uri` adicionales.
`post_logout_redirect_uris`	text	URIs específicas autorizadas para RP-Initiated Logout (sección 7.4.5 SSO Logout).
`access_mode`	enum	`public` (cualquier usuario del SSO) o `restricted` (solo usuarios listados en `sso_application_users`).

Tabla 7.5. Campos críticos de cada aplicación cliente registrada en el SSO.

El campo `access_mode` introduce un nivel adicional de control: por defecto, todo usuario autenticado puede acceder a toda aplicación registrada, lo que es deseable para una asociación pequeña en la que el SSO actúa como portal único. Cuando una aplicación maneja datos sensibles, su `access_mode` se cambia a `restricted` y se rellena `sso_application_users` con la lista explícita de usuarios autorizados; cualquier intento de acceso por parte de otro usuario se rechaza en `/authorize` con un `access_denied` antes de generar el código de autorización.

Una clase de vulnerabilidad recurrente en proveedores OAuth mal configurados es el *Open Redirect*, en el que un atacante consigue que el proveedor redirija al usuario a un dominio bajo su control tras autenticarse, capturando así el `code` o el `access_token`. `SSOProtocol::isRedirectUriAllowed` aplica cinco verificaciones acumulativas (basta con que falle una para rechazar la URI) antes de admitir cualquier `redirect_uri`: esquema `https` (con la única excepción de `http://localhost` y `127.0.0.1` durante el desarrollo); ausencia de fragmento `#...`; ausencia de credenciales en la URI; ausencia de IP directa salvo `127.0.0.1`; y coincidencia exacta o por subdominio con uno de los registrados en `allowed_domains`.

7.5.3 Sesiones (JWT RS256 con `kid` y binding User-Agent + IP)

Los *access tokens* emitidos por el SSO son JSON Web Tokens conforme a RFC 7519, firmados con RSA-2048 y SHA-256 (algoritmo RS256). El uso de firma asimétrica permite que cualquier RP verifique los *tokens* localmente sin acceso a la clave privada del SSO. La carga útil contiene los *claims* estándar (`iss`, `sub`, `iat`, `nbf`, `exp`, `jti`) más los específicos del negocio (`email`, `username`, `display_name`, `role`, `auth_method` y `cert_serial` cuando se autenticó con certificado). La cabecera incluye el campo `kid`, que identifica el par de claves con el que se firmó y habilita la rotación sin invalidar *tokens* vigentes. El sistema mantiene los pares en `sso_keypairs` (clave pública) y como ficheros `0600` en `sso/keys/` (clave privada); en cada instante hay como máximo una clave activa, pero pueden coexistir varias históricas mientras existan *tokens* en circulación firmados con ellas. La rotación se ejecuta en tres pasos (generar par nuevo y marcarlo como activo, mantener publicada la clave anterior durante el tiempo de vida máximo de un *access* o un *refresh*, retirar la clave anterior tras ese plazo); el despliegue actual no programa

la rotación automática y su automatización mediante un *timer* mensual figura entre las vías futuras.

El sistema añade a la sesión persistida en `sso_sessions` una huella derivada del navegador (`ua_fingerprint`) y la dirección IP de origen (`ip_address`). Cuando un *access token* se valida en `validateSession`, ambos campos se comparan con los del cliente actual; si no coinciden, la sesión se destruye y se registra el evento `session_binding_violation`. El alcance de esta protección es parcial: el *binding* eleva el coste de un robo de *token* al obligar al atacante a replicar el navegador y la IP exactos, y reduce la ventana de explotación de *tokens* filtrados por error (registros, capturas, *referrers*); pero no es absoluto, ya que en redes con CGNAT la IP puede ser compartida por miles de usuarios, el *roaming* legítimo entre redes provoca cambios de IP que el sistema trata como violación, los proxies corporativos y las VPN alteran la IP percibida y el *User-Agent* puede ser parcialmente falsificable. Por ello, el *binding* se entiende como una señal contextual de riesgo cuya divergencia dispara una reautenticación preventiva, no como una garantía criptográfica de posesión del *token*. La auténtica garantía requeriría mecanismos del tipo *Token Binding* (RFC 8471) o DPoP (RFC 9449), recogidos en el capítulo 9 como evolución futura.

Una sesión presenta dos estados: `active` y `destroyed`. Tras un inicio satisfactorio queda en `active` y permanece así mientras se invocan `validateSession` (con IP y *User-Agent* coincidentes) o `refreshSession` (que rota el *access token* sin alterar el estado lógico). La transición a `destroyed` se produce por cuatro causas no exclusivas: violación del *binding*, cierre de sesión explícito, vencimiento de `expires_at` o invocación administrativa de `destroyAllUserSessions`. Los *refresh tokens* tienen una vida útil de 30 días frente a los 60 minutos del *access* y se rotan en cada uso: el SSO emite un nuevo *access* y un nuevo *refresh* e invalida el anterior, lo que limita el daño de un *refresh* filtrado.

7.5.4 Panel de administración

El panel de administración se sirve como un conjunto de páginas PHP autenticadas bajo `sso/public/admin/`, accesibles únicamente por usuarios cuyo `role` es `admin` en `sso_users`. No se emplea ningún *framework* SPA: el panel es operativo sin JavaScript habilitado, lo que reduce su superficie de ataque y

simplifica las pruebas; JavaScript se utiliza solo como capa de mejora (validación de formularios, confirmaciones modales, refresco asíncrono de tablas). Las cinco vistas principales son el *Dashboard* (estado global del SSO y sesiones activas), Usuarios (alta, edición, desactivación y gestión de rol sobre `sso_users`), Aplicaciones (alta de RP, generación de `client_id` y `client_secret`, edición de `redirect_uri` y `post_logout_redirect_uris` sobre `sso_applications`), Acceso por aplicación (gestión de la lista blanca para apps con `access_mode='restricted'` sobre `sso_application_users`) y Auditoría (búsqueda y filtrado del *log* de eventos en `sso_audit_log`). Cada evento de auditoría lleva una carga útil JSON (`payload_json`) con los parámetros relevantes (cliente, IP, motivo, *fingerprint*); el esquema no es uniforme entre eventos por diseño, ya que cada tipo tiene sus campos propios y forzar un esquema único habría llevado a columnas vacías en la mayoría de casos. La búsqueda sobre `payload_json` se realiza con los operadores `JSON_EXTRACT` de MariaDB cuando es necesario filtrar por un campo concreto.

7.6 Métodos de autenticación implementados

El SSO soporta dos métodos de autenticación operativos en producción, complementarios y diseñados para escenarios distintos. La elección entre uno y otro la realiza el usuario en la pantalla de *login* y no es una decisión binaria a nivel de cuenta: un mismo usuario puede tener vinculados ambos métodos.

7.6.1 Usuario, contraseña y TOTP

El método clásico combina dos factores: algo que el usuario sabe (la contraseña) y algo que tiene (un dispositivo con una aplicación TOTP). Las contraseñas se almacenan exclusivamente como *hash* Argon2id (algoritmo recomendado actualmente por OWASP), calculado con `password_hash($plain, PASSWORD_ARGON2ID)`; PHP elige parámetros sensatos por defecto (`memory_cost = 65 536 KiB`, `time_cost = 4`, `parallelism = 1`), incrementables en el `.env` si el *hardware* lo admite. La verificación se realiza con `password_verify`, resistente a ataques de tiempo, y el rehash automático se activa cuando `password_needs_rehash` detecta parámetros inferiores a los actuales. Una vez verificada la contraseña, el sistema exige (si el usuario activó el segundo factor) un código de seis dígitos TOTP conforme a RFC 6238. El secreto compartido se

almacena en `sso_users.totp_secret` (base32) con permisos de columna restringidos y se entrega al usuario una sola vez (al activar el TOTP) mediante un código QR en formato `otpauth://totp/...` reconocido por Google Authenticator, Authy, Microsoft Authenticator y otras aplicaciones estándar. La validación admite una ventana de ±1 paso, lo que tolera desincronizaciones de reloj de hasta 30 segundos; cada código se rechaza tras su primer uso correcto dentro de la misma sesión de verificación. Para el escenario de pérdida del dispositivo TOTP, el sistema dispone de códigos de recuperación de un solo uso, generados en grupos de diez al activar el segundo factor, almacenados como *hash* SHA-256 e invalidados tras su uso.

7.6.2 Certificado digital de la FNMT

El segundo método es la autenticación basada en certificados X.509 v3 emitidos por una autoridad de confianza, presentados durante el *handshake* TLS mutuo (mTLS) con el SSO. La autenticación se delega en Apache mediante `mod_ssl`: el *vhost* `sso.marchernandez.es` se configura con `SSLVerifyClient optional`, `SSLCACertificatePath` apuntando a la cadena de CA aceptadas (FNMT, DNIe y la propia raíz del proyecto) y `SSLVerifyDepth` ajustado a la profundidad máxima esperada. Cuando el usuario presenta un certificado en el *handshake*, Apache verifica la firma de la CA emisora y la cadena completa hasta una raíz conocida; si la verificación tiene éxito, expone al *backend* PHP las variables `SSL_CLIENT_*`. La validación se reparte entre dos capas: Apache asume las verificaciones criptográficas (firma de la cadena, validez temporal y conformidad ASN.1) y la aplicación, mediante `CertificateAuth`, decide los aspectos semánticos. Sobre esa división, `CertificateAuth::authenticate` ejecuta cinco controles en cascada: filtrado por *Extended Key Usage* (lista de OID prohibidos como autenticación de servidor TLS o firma de código, y permitidos como `clientAuth`, `emailProtection` o *Smart Card Logon*); verificación de la *Certificate Policy* (para FNMT y DNIe se exige el OID de política, por ejemplo `2.16.724.1.2.2.4.1` para persona física FNMT y `2.16.724.1.3.5.5.1` para autenticación DNIe); confianza del emisor (lista en `sso_trusted_issuers` y, redundantemente, en `config/trusted_cas.php`); verificación de revocación contra el PKI propio (consulta directa a `certificates` por *serial*, complementaria a la verificación de Apache contra la CRL); y vinculación con un usuario del sistema mediante el *thumbprint* SHA-256 buscado

en `sso_user_certificates`. Si el certificado es válido, pero no está vinculado, el flujo redirige a `/link-cert.php`; si está revocado o presenta un EKU inválido, redirige a `/login.php` con el código de error correspondiente. El sistema distingue, además, mediante heurísticas combinadas (DN del emisor, cabecera `X-SmartCard-Auth`, OID `1.3.6.1.4.1.311.20.2.2` en el EKU), los certificados emitidos por *hardware* (DNIe, *smartcards*) de los emitidos por *software* (FNMT instalada en el navegador), lo que permite políticas diferenciadas. La autenticación por certificado traslada parte de la confianza al navegador y al almacén de certificados del sistema operativo del usuario, lo que reduce la exposición al *phishing* clásico (el navegador no entrega el certificado a un dominio fraudulento, aunque imite gráficamente al legítimo) y aporta una identidad atribuible verificada por una autoridad de certificación reconocida.

7.6.3 Comparativa de los dos métodos

La Tabla 7.6 resume las propiedades de los dos métodos en cinco dimensiones relevantes para el público objetivo. La resistencia al *phishing* del certificado merece comentario: un atacante que monte un dominio fraudulento como `sso.marchernamdez.es` puede engañar al usuario para que introduzca su contraseña y su TOTP, pero no puede engañar al navegador para que entregue el certificado, ya que el navegador solo presenta certificados al dominio legítimo cuya CA reside en su almacén de confianza. Esta propiedad estructural, intrínseca a mTLS, hace del certificado el método preferente para escenarios de alto riesgo.

Propiedad	Contraseña con TOTP	Certificado FNMT o DNIe
Coste de adopción	Bajo (instalar aplicación TOTP)	Medio-alto (obtener certificado o DNIe)
Recuperación tras pérdida	Códigos de recuperación	Reemisión del certificado
Garantías de identidad	Identidad declarada en alta	Identidad verificada por AC reconocida

Propiedad	Contraseña con TOTP	Certificado FNMT o DNIe
Resistencia al *phishing*	Vulnerable si el dominio se mimetiza	Alta (el certificado solo se entrega al dominio legítimo)
Idoneidad	Acceso cotidiano a aplicaciones internas	Operaciones de alta confianza (firma, gestión PKI)

Tabla 7.6. Comparativa de los dos métodos de autenticación operativos del SSO.

7.7 Bloques OCSP, CRL y repositorio CA

Los tres subsistemas restantes (OCSP, CRL y repositorio CA) tienen un volumen de código propio reducido y la complejidad criptográfica reside en los componentes ya descritos en el apartado 7.4.5. Su separación en *vhosts* dedicados cumple dos funciones operativas: aislamiento (una caída de cualquiera de los tres no afecta al portal PKI ni al SSO) y políticas de caché diferenciadas que evitan mezclar el contenido OCSP con el del portal o el repositorio.

El *responder* OCSP (`ocsp.marchernandez.es`) atiende consultas tanto por POST (DER en el cuerpo, RFC 6960 sección A.1) como por GET (DER en *base64* en el *path*, para validadores antiguos). El tiempo de respuesta y el estado se registran en `ocsp_queries`, lo que permite construir métricas operativas medibles. La Tabla 7.7 resume las expectativas conforme al RNF-09; la validación experimental se realiza en el capítulo 8.

Métrica	Objetivo (RNF-09)	Origen del dato
Latencia p50	< 50 ms	`ocsp_queries.response_time_ms`
Latencia p95	< 100 ms	`ocsp_queries.response_time_ms`
Disponibilidad	> 99 % mensual	*Health check* externo cada 5 min

Métrica	Objetivo (RNF-09)	Origen del dato
Tasa `unknown`	< 1 %	Recuento `response_status='unknown'`

Tabla 7.7. Métricas operativas del responder OCSP.

El subsistema CRL (`crl.marchernandez.es`) publica como ficheros estáticos `.pem` y `.der` la lista de revocación de cada CA activa. El diseño combina dos parámetros independientes: la frecuencia de regeneración (cuatro horas) y la validez del DER firmado (veinticuatro horas), justificadas en el apartado 7.4.5. Las cuatro estrategias evaluadas se resumen en la Tabla 7.8.

Estrategia	Regeneración	`nextUpdate`	Adecuación
Diaria	24 h	24 h	Insuficiente ante compromiso de clave
Adoptada	4 h	24 h	Frescura del repositorio + margen ante fallos del *timer*
Subhoraria	1 h	1 h (requiere `-startdate`)	Sobredimensionada y rompe caché HTTP
Solo OCSP	-	-	Frágil ante caídas del *responder* y clientes solo-CRL

Tabla 7.8. Estrategias de publicación de CRL evaluadas y decisión adoptada.

La distribución se realiza con cabeceras HTTP de caché más cortas que el `nextUpdate` del DER (`Cache-Control: public, max-age=3600`), de modo que un validador conforme con HTTP revalida cada hora, aunque el `nextUpdate` interno aún no haya caducado.

El repositorio CA (`ca.marchernandez.es`) es el componente más simple del sistema, por diseño. Su única función es publicar los certificados de las autoridades en formatos PEM y DER (`/root.crt`, `/intermediate_<id>.crt`, `/chain.pem`) para que cualquier validador construya la cadena de confianza. No incorpora lógica de negocio, autenticación ni base de datos: únicamente ficheros estáticos con permisos `0644` servidos por Apache. La simplicidad del componente es una propiedad de seguridad, no un déficit de funcionalidad: cuanto menos código se ejecuta al servir una petición, menos vectores de ataque hay; el subsistema podría sobrevivir intacto a una caída completa del subsistema PKI, ya que los certificados raíz e intermedios son inmutables durante años.

7.8 Modelo de seguridad transversal

Los subsistemas descritos hasta este punto no operan en un vacío: comparten una arquitectura de seguridad transversal de nueve capas que se aplica de manera homogénea a todos los *vhosts*. La estrategia es la clásica de *defense in depth*: ninguna capa es individualmente suficiente, pero la composición de todas ellas exige al atacante comprometer múltiples mecanismos antes de causar daño. La Figura 7.8 representa el *stack* completo, y la Tabla 7.9 lo conecta con tecnologías concretas y con las amenazas STRIDE que cada capa mitiga.

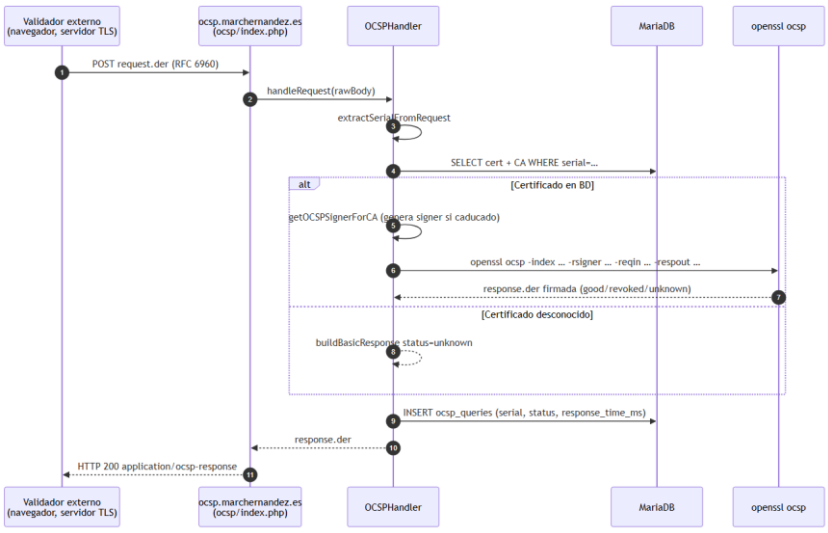

Figura 7.8. Arquitectura de seguridad en nueve capas (stack horizontal, de la más superficial a la más profunda).

#	Capa	Implementación principal	Estándares de referencia	Amenazas STRIDE mitigadas
1	TLS y aislamiento de red	Let's Encrypt + HSTS + MariaDB sobre Unix socket + fail2ban + ModSecurity	RFC 8446 (TLS 1.3), RFC 6797 (HSTS)	S, I
2	Autenticación MFA / mTLS	`PasswordAuth`+ `TOTPAuth` o `CertificateAuth`	RFC 6238 (TOTP), RFC 5280 (X.509)	S
3	Autorización RBAC	`role` en `sso_users` + `sso_application_users` (access_mode restricted)	NIST RBAC	E
4	Cifrado en reposo	Argon2id (contraseñas), AES-256-CBC (passphrase CA), `mariadb` con `data-at-rest` opcional	OWASP Password Storage, NIST SP 800-38A	I
5	Validación de entrada	`InputValidator` + PDO prepared	OWASP ASVS V5	T, I

#	Capa	Implementación principal	Estándares de referencia	Amenazas STRIDE mitigadas
		statements + `filter_var`		
6	Protección CSRF	Tokens por sesión + cookies `SameSite=Lax`	OWASP CSRF Prevention	T, E
7	Rate limiting	`RateLimiter` por IP+endpoint con almacenamiento en `sso_rate_limits`	OWASP Brute Force Prevention	D, E
8	Cabeceras HTTP de seguridad	CSP, HSTS, X-Frame-Options, X-Content-Type-Options, Referrer-Policy, Permissions-Policy	OWASP Secure Headers	T, I
9	Auditoría y monitorización	`sso_audit_log` + `audit_log` (PKI) + alertas Telegram +	NIST SP 800-92	R, I

#	Capa	Implementación principal	Estándares de referencia	Amenazas STRIDE mitigadas
		métricas OCSP		

Tabla 7.9. Mapeo de las nueve capas a tecnologías concretas, estándares de referencia y dimensiones STRIDE mitigadas (S=Spoofing, T=Tampering, R=Repudiation, I=Information disclosure, D=Denial of service, E=Elevation of privilege).

Tres observaciones sobre la arquitectura. Cada amenaza STRIDE recibe al menos dos capas independientes de mitigación, de modo que ningún riesgo descansa sobre un único control. Las capas 1 y 2 (red y autenticación) son responsabilidad principal del administrador del sistema, mientras que las capas 3 a 9 son responsabilidad del código de la aplicación. Las capas 8 y 9 actúan como controles detectivos: aunque las capas anteriores fallen, las cabeceras HTTP limitan el daño y la auditoría permite la reconstrucción posterior. El conjunto se mide contra un modelo de amenazas explícito en el apartado 7.9.

7.9 Modelo de amenazas STRIDE

El análisis se organiza siguiendo la taxonomía STRIDE de Microsoft (Howard y LeBlanc, 2003). Por motivos de extensión, las matrices completas (treinta y dos amenazas con sus mitigaciones y conclusiones por categoría) se publican en el **MAm** (Manual de Análisis de Amenazas); la memoria conserva el resumen de la Tabla 7.10 y la matriz global de la Figura 7.9. La tabla sintetiza la cobertura por categoría con el número total de amenazas analizadas, las mitigadas y las que quedan parcialmente mitigadas o como riesgo residual asumido.

Categoría STRIDE	Analizadas	Mitigadas	Parciales	Residuales
Spoofing (suplantación de identidad)	6	3	2	1

Categoría STRIDE	Analizadas	Mitigadas	Parciales	Residuales
Tampering (manipulación de datos)	6	5	1	0
Repudiation (repudio de acciones)	4	3	1	0
Information disclosure (fuga de información)	6	4	2	0
Denial of service (denegación de servicio)	5	3	1	1
Elevation of privilege (escalada de privilegios)	5	5	0	0
Total	32	23	7	2

Tabla 7.10. Cobertura agregada de las amenazas analizadas por categoría STRIDE.

Las amenazas con riesgo residual se concentran en dos vectores que el sistema no controla por completo: el *phishing* avanzado sobre el método contraseña con TOTP, frente al cual ningún control puramente técnico ofrece protección absoluta y para el que el método de certificado FNMT o DNIe se documenta como alternativa preferente para operaciones de alta confianza; y el DoS volumétrico de red, cuya mitigación depende del *upstream* del proveedor de infraestructura como servicio. Las siete amenazas parcialmente mitigadas se trazan, en el **MAm**, con la limitación correspondiente del apartado 8.4 o con la vía futura del apartado 9.4.

7.9.1 Matriz global impacto × probabilidad

La síntesis del análisis se representa en la Figura 7.9, donde cada amenaza identificada se posiciona en función de su impacto (severidad del daño si se materializa) y su probabilidad estimada (cualitativa, dadas las mitigaciones implementadas). La clasificación corresponde al riesgo residual tras aplicar las mitigaciones descritas, no al riesgo bruto.

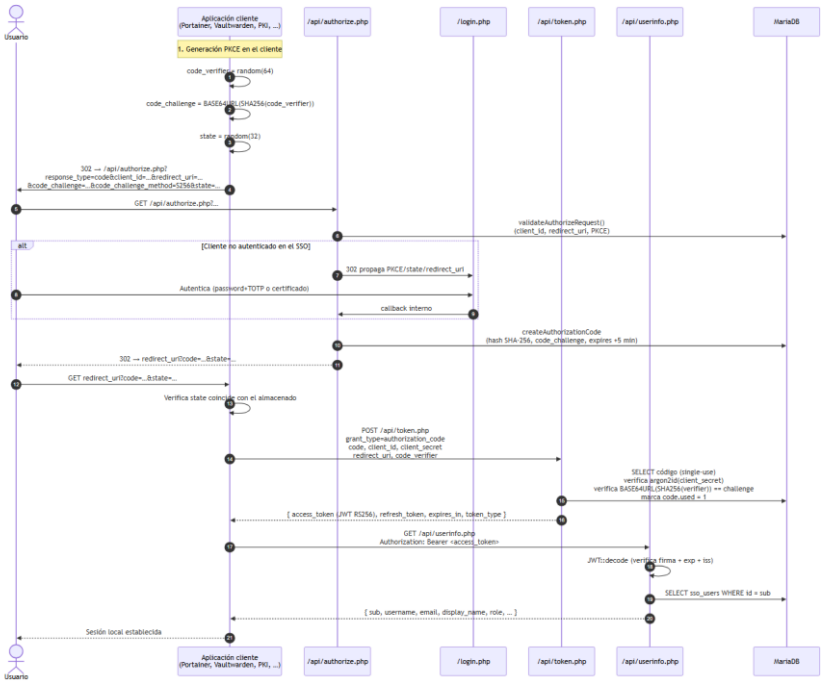

*Figura 7.9. Matriz de riesgo residual tras aplicar las mitigaciones descritas en el **MAm**.*

Tres observaciones se desprenden de la matriz. Ninguna amenaza queda en el cuadrante de probabilidad y impacto altos tras aplicar las mitigaciones. Los tres riesgos con mayor impacto residual (ID-02 claves de CA, DO-05 DoS volumétrico y SP-02 *phishing*) corresponden a amenazas mitigadas de forma parcial, registradas como tales en el **MAm** y reclasificadas como vías futuras en el apartado 9.4. La mayor parte de los riesgos se concentra en el cuadrante de probabilidad media e impacto medio, característico de amenazas controladas por las mitigaciones descritas, pero no eliminadas; el control depende de la persistencia operativa de esas mitigaciones a lo largo del tiempo.

7.10 Operación y despliegue

El capítulo cierra describiendo la operación viva del sistema: los procesos automatizados que mantienen el servicio entre intervenciones humanas, la estrategia de copias de seguridad y los protocolos de recuperación ante fallos.

7.10.1 Tareas periódicas (systemd timers)

El sistema utiliza timers de systemd -no `cron` clásico- por tres razones: granularidad de control por servicio, *journaling* nativo en `journalctl`, y posibilidad

de declarar dependencias entre unidades. La Tabla 7.11 enumera los cinco timers activos en el despliegue actual.

Timer	Frecuencia	Función	Falla si
`pki-crl-rotate.timer`	cada 4 h	Regenera CRL de cada CA activa (sección 7.7)	Marca métrica en BD, envía alerta a Telegram
`pki-cleanup.timer`	diaria 03:00	Purga `tempDir` huérfanos, `ocsp_queries` >90 d, sesiones expiradas	Idempotente, reintento al día siguiente
`pki-ocsp-health.timer`	cada 5 min	Consulta de prueba contra el *responder* OCSP	Tras 3 fallos consecutivos, alerta a Telegram
`pki-ca-expiration.timer`	semanal	Detecta CAs y certificados que expiran en <30 d	Alerta a Telegram con la lista de afectados
`pki-backup.timer`	diaria 02:00	Vuelca BDs + `pki/ca/` cifrado + sube a *off-site*	Reintento + alerta si falla 2 días seguidos

Tabla 7.11. Tareas automatizadas del sistema y comportamiento ante fallo.

7.10.2 Copias de seguridad y recuperación

Las copias de seguridad se realizan a tres niveles, con frecuencias y políticas de retención diferenciadas:

Nivel	Contenido	Frecuencia	Retención	Cifrado
BD	`mysqldump` de SSO y PKI	diaria	30 días	GPG con clave del administrador
Material PKI	`pki/ca/` completo (claves, passphrases, índices)	diaria	30 días	GPG con clave del administrador
Configuración	`.env`, `vhosts`, scripts	semanal	12 semanas	GPG con clave del administrador

Tabla 7.12. Estrategia de copias de seguridad.

Las copias se almacenan en un servidor *off-site* distinto del VPS de producción, accesible solo por SSH con clave pública. El objetivo de recuperación (RNF-10) -inferior a 30 minutos ante incidencias menores- se sostiene sobre tres procedimientos documentados en el **MA**: (1) restauración de BD desde el dump más reciente, (2) restauración de un fichero de CA si se ha corrompido individualmente, y (3) reinicio del stack completo de Apache+PHP-FPM+MariaDB con verificación posterior.

Para incidentes mayores -comprometimiento de la clave maestra del `.env`, ataque dirigido con persistencia, fallo total del VPS- el procedimiento implica una reinstalación completa sobre infraestructura limpia y la rotación de todas las claves CA, lo que sale del RTO de 30 minutos y se describe explícitamente como tal en el capítulo 9.

7.10.3 Monitorización operativa

La monitorización combina tres canales:

1. **Métricas internas en BD**: `ocsp_queries`, `sso_audit_log`, `audit_log`, `crl_records`, `sso_rate_limits`. Consultadas por el panel admin y por los *timers* de *health check*.

2. **Logs del sistema**: `journalctl` para los *timers*, `php-fpm error_log` para fallos de aplicación, Apache `access_log`/`error_log` para tráfico HTTP.

3. **Alertas Telegram**: los eventos definidos en la sección 7.3.1 (caducidades, fallos de *cron*, picos de errores HTTP, intentos de acceso anómalos) se canalizan al *bot* del administrador, con la política de minimización de datos personales descrita.

Las tres vías son complementarias, no redundantes: las métricas BD sirven para análisis a posteriori; los logs, para diagnóstico fino tras un incidente; las alertas Telegram, para notificación reactiva en tiempo casi real. Ningún incidente operativo crítico debería pasar desapercibido más de cinco minutos, gracias al solapamiento de los tres canales y al *health check* periódico del OCSP.

La validación experimental de las decisiones de diseño aquí justificadas, junto con el procedimiento de despliegue y el plan de pruebas, se desarrolla en el capítulo siguiente.

CAPÍTULO 8. DESPLIEGUE, PRUEBAS Y VALIDACIÓN

Este capítulo presenta el plan de verificación del sistema implementado en el capítulo 7. El apartado 8.1 describe el plan de pruebas técnicas en cuatro niveles (unitarias, integración, aceptación y seguridad). El apartado 8.2 documenta el protocolo de evaluación de usabilidad diseñado, pero no ejecutado dentro del plazo del proyecto. El apartado 8.3 recoge las mediciones operativas del sistema desplegado, alineadas con los requisitos no funcionales RNF-08 y RNF-09 (apartado 7.2.2). El apartado 8.4 declara las limitaciones del programa de pruebas. El apartado 8.5 sintetiza la capacidad estimada del despliegue, el mantenimiento operativo y el plan de formación. La metodología y los criterios de aceptación se documentan aquí; las mediciones cuantitativas sobre el sistema desplegado se publican en el repositorio del proyecto y se sintetizan en el apartado 8.3.8.

8.1 Plan de pruebas

8.1.1 Marco general

El plan adopta una pirámide de pruebas adaptada al alcance del trabajo, con cuatro niveles complementarios. La proporción entre niveles se desvía de la canónica industrial (donde las pruebas unitarias dominan en volumen) porque un sistema PKI concentra su lógica crítica en el comportamiento de borde criptográfico, que se valida mejor con casos dirigidos a los vectores oficiales del IETF y con integración extremo a extremo que con conjuntos amplios de pruebas unitarias de utilidad. La Tabla 8.1 sintetiza el reparto resultante.

Nivel	Qué prueba	Herramienta principal	Volumen actual	Volumen objetivo
Unitarias	Lógica criptográfica y de utilidad sin dependencia de BD/HTTP	PHP standalone (vectores RFC, *reflection*)	29 (PKCE)	~60

Nivel	Qué prueba	Herramienta principal	Volumen actual	Volumen objetivo
Integración	Flujos completos: OAuth, emisión, revocación, CRL, OCSP, mTLS	Scripts PHP + `curl` + colecciones Postman	Manual	~12 escenarios automatizados
Aceptación	Cumplimiento de los requisitos funcionales RF-01 a RF-22	Checklist guiado + capturas	Manual	22 verificaciones documentadas
Seguridad	Mitigaciones STRIDE: Open Redirect, SQLi, XSS, CSRF, *rate limit*, cabeceras	Tests dirigidos + herramientas pasivas (`securityheaders.com`, `testssl.sh`)	Parcial	~15 escenarios

Tabla 8.1. Niveles de prueba del proyecto y volumen objetivo de cobertura.

Cada nivel se desarrolla en los apartados siguientes con sus particularidades, criterios de éxito y herramienta concreta.

8.1.2 Pruebas unitarias

Las pruebas unitarias se aplican a las funciones críticas aisladas cuya lógica no depende de la base de datos ni de una petición HTTP. En un sistema PKI estas funciones son menos numerosas que en una aplicación CRUD típica (gran parte

del código corresponde a orquestación), pero las que existen condicionan la seguridad del conjunto: un fallo en cualquiera de ellas compromete el sistema. El listado actual y previsto figura en la Tabla 8.2.

Conjunto	Métodos cubiertos	Casos cubiertos
PKCE (`sso/tests/test_pkce.php`, 29/29 PASS)	`validatePkceParams`, `verifyCodeVerifier`	Vectores RFC 7636 sección B.1, generación cliente-servidor, alfabeto BASE64URL, longitud 43-128, rechazo de `plain`, compatibilidad con clientes anteriores
Cifrado de *passphrase*	`PKIEngine::encryptPassphrase`, `decryptPassphrase`	Formato nuevo (IV hex 32), formato anterior (IV binario 16), retrocompatibilidad, error con clave incorrecta
Firma y verificación JWT	`JWT::encode`, `JWT::decode`	RS256 *round-trip*, rotación por `kid`, rechazo si la firma se altera, rechazo si `exp` ha vencido
Validación de `redirect_uri`	`SSOProtocol::isRedirectUriAllowed`	Los cinco controles del apartado 7.5.2 con casos válidos e inválidos
Validación EKU/Policy de certificados	`CertificateAuth::validateCertificateType`	FNMT persona física aceptado, *server-auth* rechazado, certificado sin EKU aceptado por

Conjunto	Métodos cubiertos	Casos cubiertos
		compatibilidad, OID bloqueado rechazado
TOTP	`TOTPAuth::verify`	Código vigente, ventana ±1, código no reutilizable
Cálculo de *thumbprint*	`CertificateAuth::calculateThumbprint`	Coincide con `openssl x509 -fingerprint -sha256`

Tabla 8.2. Conjuntos de pruebas unitarias y alcance funcional cubierto por cada uno.

La opción canónica en la industria sería PHPUnit (Bergmann et al., 2024), pero el coste de la configuración inicial (autoloader, *fixtures* de base de datos) resulta desproporcionado para el volumen de utilidad criptográfica aislable de este proyecto. Se adopta en su lugar un enfoque de ficheros de test ejecutables como CLI con dependencia exclusiva de `php-cli` y `ext-openssl`, que permite ejecutar las pruebas en cualquier entorno con un único comando y sostiene la reproducibilidad de los resultados. Esta decisión queda registrada como limitación L-06 en el apartado 8.4 y como vía futura en el apartado 9.4.

8.1.3 Pruebas de integración

Las pruebas de integración validan flujos completos que atraviesan múltiples componentes. Se diseñan a partir de los diagramas de flujo del capítulo 7 (Figura 7.4 -flujo integrado de autenticación, emisión y verificación- y Figura 7.7 -flujo OAuth 2.0 Authorization Code con PKCE-) y cubren las interacciones críticas entre subsistemas. La Tabla 8.3 enumera los doce escenarios objetivo.

#	Escenario	Subsistemas atravesados	Criterio de éxito
I-01	Login con contraseña + TOTP	SSO → BD	Sesión activa, JWT válido

#	Escenario	Subsistemas atravesados	Criterio de éxito
			emitido, IP y UA registrados
I-02	Login con certificado FNMT	Apache + SSO + PKI	mTLS éxito, EKU validado, cuenta vinculada, sesión activa
I-03	Flujo OAuth + PKCE completo	RP → SSO → RP → SSO	RP recibe access_token JWT, code marcado *used* tras intercambio
I-04	Refresh token rotation	SSO → BD	Nuevo *access+refresh*, antiguo *refresh* invalidado
I-05	Logout RP-Initiated	RP → SSO	Sesión destruida, redirect a `post_logout_redirect_uri` validado
I-06	Solicitud de certificado clientAuth	PKI → BD	Solicitud en estado `pending`
I-07	Aprobación + emisión de certificado	PKI → BD → FS → OpenSSL	Estado `issued`, fila en `certificates`, `index.txt` actualizado

#	Escenario	Subsistemas atravesados	Criterio de éxito
I-08	Revocación + regeneración CRL inmediata	PKI → CRL público	CRL actualizada, serial presente, `nextUpdate` válido
I-09	Consulta OCSP de certificado válido	OCSP → PKI BD	Respuesta `good` firmada por OCSP signer
I-10	Consulta OCSP de certificado revocado	OCSP → PKI BD	Respuesta `revoked` con motivo correcto
I-11	Bloqueo por *rate limit* en `/login`	SSO → `sso_rate_limits`	Bloqueo tras N fallos, mensaje 429
I-12	Violación de *binding* IP/UA	SSO → BD	Sesión destruida, evento auditado, reautenticación requerida

Tabla 8.3. Escenarios de integración objetivo, con criterios de éxito explícitos.

Cada escenario se automatiza con un *script* PHP que ejecuta la secuencia y verifica el estado final. Los flujos que requieren mTLS (I-02) se prueban con `curl --cert` y `--key` contra el *vhost* de pruebas. El conjunto se ejecuta como prueba de humo tras cada despliegue significativo.

8.1.4 Pruebas de aceptación

Las pruebas de aceptación verifican que los veintidós requisitos funcionales del apartado 7.2.1 están satisfechos por el sistema desplegado. Se documentan como una *checklist* en la que cada requisito se valida con cuatro elementos: los pasos de reproducción, el resultado esperado según el requisito, el resultado obtenido con su evidencia (captura o salida de consola) y un veredicto de tres

valores (cumple, cumple parcialmente, no cumple). La *checklist* completa figura en el Anexo B con una entrada por requisito. La Tabla 8.4 muestra el formato aplicado a tres requisitos representativos.

RF	Validación	Resultado
RF-04	Solicitud a `/api/authorize.php` con PKCE S256 + intercambio en `/api/token.php` con `code_verifier` válido	Sí. Access token emitido; ejecución de `sso/tests/test_pkce.php` 29/29 PASS
RF-10	Inspección de `pki/ca/` y BD `certificate_authorities`: presencia de raíz y N intermedias activas con `is_active=1`	Sí. 1 raíz + 3 intermedias activas (capturas en anexo C)
RF-19	Verificar publicación automática de CRL por CA tras revocación y tras *cron* de 4 h	Sí. Archivos `intermediate_<id>.crl` actualizados; entrada en `crl_records`

Tabla 8.4. Extracto del formato de la checklist de aceptación (3 RFs de los 22 totales).

8.1.5 Pruebas de seguridad

Las pruebas de seguridad se diseñan para verificar que los comportamientos no deseados quedan efectivamente bloqueados por las mitigaciones declaradas. Se organizan según la taxonomía STRIDE del apartado 7.9 (Howard y LeBlanc, 2003), con un escenario por mitigación clave. La Tabla 8.5 enumera los quince escenarios objetivo.

#	Categoría STRIDE	Vector probado	Mitigación que valida	Herramienta
S-01	Spoofing	Login con contraseña común tras N intentos	*Rate limit* + Argon2id	`curl` + script
S-02	Spoofing	Reuso de JWT desde otra IP	*Binding* IP+UA	`curl` desde dos IPs
S-03	Tampering	SQLi pasivo en formularios	PDO prepared	`sqlmap` modo pasivo
S-04	Tampering	XSS reflejado en `state` y `error`	Escape HTML + CSP	Payloads OWASP
S-05	Tampering	Manipulación de JWT (cambiar `alg` a `none`)	RS256 estricto, `kid` válido	Script de manipulación
S-06	Repudiation	Verificar que `audit_log` recoge cada acción crítica	Auditoría en cada *write*	Inspección SQL
S-07	Information Disclosure	Open Redirect en `redirect_uri` (5 vectores: subdominio fuera de whitelist,	`isRedirectUriAllowed` 5 controles	`curl`

#	Categoría STRIDE	Vector probado	Mitigación que valida	Herramienta
		`https://evil.com#x@host`, IP directa, fragmento, credenciales)		
S-08	Information Disclosure	Open Redirect en `post_logout_redirect_uri`	Validación del cierre de sesión, apartado 7.5.1	`curl`
S-09	Information Disclosure	Acceso directo a `pki/ca/intermediate_*/intermediate_ca.key` vía HTTP	Permisos FS + DocumentRoot	`curl` directo
S-10	Information Disclosure	Cabeceras de seguridad presentes y correctas	Capa 8 de la sección 7.8	`securityheaders.com`, `testssl.sh`
S-11	DoS	*Flood* de `/api/token.php` desde una IP	*Rate limit* L7	`ab` o `wrk`
S-12	DoS	*Flood* de `/api/ocsp` con peticiones malformadas	*Rate limit* + validación temprana	Script DER inválido

#	Categoría STRIDE	Vector probado	Mitigación que valida	Herramienta
S-13	Elevation of Privilege	Acceso a `/admin/*` con sesión de rol `user`	Middleware de autorización	`curl` con cookie no-admin
S-14	Elevation of Privilege	Solicitud de firma a CA raíz	Restricción explícita en `signCertificate`	Llamada directa al método
S-15	Elevation of Privilege	Token de aplicación A usado contra aplicación B	Validación `aud` en el RP	`curl` con token cruzado

Tabla 8.5. Escenarios de pruebas de seguridad, alineados con STRIDE.

Las pruebas se documentan en el Anexo B con el mismo formato que las de aceptación (pasos, resultado esperado, resultado obtenido y veredicto), añadiendo el identificador STRIDE correspondiente y la evidencia gráfica.

8.2 Protocolo de evaluación de usabilidad (diseñado, no ejecutado)

Las pruebas técnicas verifican el cumplimiento del comportamiento especificado; la evaluación de usabilidad verifica que el sistema es operable por los perfiles a los que se destina. Ambos planos son complementarios. El protocolo descrito en este apartado se diseñó como artefacto formal de evaluación cualitativa, pero no se ha ejecutado dentro del plazo del trabajo, en línea con la limitación L-01 del apartado 8.4. La memoria opta por declarar este hecho de forma explícita en lugar de incorporar puntuaciones del cuestionario *System Usability Scale* o tasas de éxito no obtenidas experimentalmente, en coherencia con los principios de la *Publication Manual* de la APA (2020) sobre la separación entre métodos planificados y resultados observados. El detalle del protocolo (consentimiento informado, guion de moderación, instrumentos de medida y plantilla de informe

de resultados) se publica en el **MVP** (Manual de Validación y Pruebas) y se sintetiza a continuación.

El protocolo persigue tres objetivos: medir la usabilidad de los portales SSO y PKI mediante el cuestionario *System Usability Scale* (Brooke, 1996; Sauro, 2011); identificar puntos de fricción en los flujos críticos de registro, inicio de sesión, solicitud y descarga de certificado, y revocación; y comprobar la comprensibilidad de los mensajes de error y de los textos legales para usuarios no técnicos. El diseño previsto es intrasujeto, con un mínimo de ocho participantes y un máximo de doce, lo que sigue la recomendación de Nielsen (1993, 2000) sobre saturación práctica de problemas de usabilidad. La captación combina, a partes iguales, dos perfiles equilibrados: usuarios sin experiencia previa en certificados digitales y usuarios con experiencia previa en certificados FNMT o DNI electrónico. Cada sesión, de cuarenta y cinco minutos de duración estimada, ejecuta seis tareas que cubren los flujos críticos del sistema (Tabla 8.6). Por tarea se recogen cuatro métricas (tasa de éxito binaria, tiempo de finalización, número de errores y satisfacción inmediata en escala de uno a cinco); por sesión, la puntuación SUS calculada con la fórmula original de Brooke. El umbral de aceptación se fija en 68 puntos por portal, valor de referencia consolidado por Sauro (2011), con justificación específica del menor umbral exigido al portal PKI por la complejidad intrínseca de su dominio.

ID	Tarea	Criterio de éxito	Métrica principal
T1	Crear cuenta en el SSO y activar TOTP	Cuenta activa con segundo factor	Tiempo, errores
T2	Iniciar sesión con contraseña y TOTP	Acceso al portal	Tiempo, errores, satisfacción
T3	Iniciar sesión con certificado FNMT	Acceso al portal	Tiempo, errores

ID	Tarea	Criterio de éxito	Métrica principal
T4	Solicitar un certificado *clientAuth* con CN propio	Solicitud en estado `pending`	Tiempo, claridad del formulario
T5	Descargar el certificado tras aprobación simulada	Fichero `.pfx` descargado	Tiempo, comprensión de los formatos
T6	Revocar el certificado descargado	Estado `revoked` confirmado	Tiempo, claridad de los motivos RFC 5280

Tabla 8.6. Tareas del protocolo de evaluación de usabilidad.

Cada sesión sigue un guion de cuatro fases: presentación y firma del consentimiento informado (5 min), tareas guiadas con protocolo *think-aloud* (Lewis y Rieman, 1993), cuestionario SUS autoadministrado y entrevista abierta sobre puntos críticos. Los principios éticos aplicados son cuatro: consentimiento informado previo con derecho de retirada, anonimización de los datos publicados con cifrado de los datos brutos y borrado al cierre del proyecto, minimización (no se almacenan grabaciones audiovisuales, solo notas y métricas numéricas) y separación del entorno respecto del despliegue de producción. Cuando el protocolo se ejecute, los resultados se publicarán con la estructura habitual de la disciplina (Sauro y Lewis, 2016) en el **MVP**. Mientras tanto, la validación del sistema descansa sobre los cuarenta y nueve indicadores cuantificables del Anexo B (veintidós requisitos funcionales, quince mitigaciones STRIDE y siete conjuntos unitarios) y los cuatro indicadores operativos del apartado 8.3.8.

8.3 Resultados de pruebas técnicas

Esta sección recoge las mediciones operativas del sistema desplegado, alineadas con los requisitos no funcionales RNF-08 y RNF-09 del apartado 7.2.2 y con las decisiones de los apartados 7.5 (OAuth, PKCE y OIDC), 7.7 (OCSP) y

7.8 (CRL). Para cada prueba se indica su objetivo y la tabla de criterios con sus umbrales de aceptación. Los procedimientos completos (comandos `curl`, consultas SQL parametrizadas y *scripts* de medición) se publican en el **MVP**, accesible desde el repositorio del proyecto, junto con la trazabilidad detallada de cada criterio.

8.3.1 Metodología de medición y justificación de los umbrales

Las mediciones se realizan en dos entornos complementarios. El entorno de preproducción reproduce las características del de producción, pero aislado, lo que permite ejecutar las pruebas activas (emisión y revocación de certificados, carga controlada del *responder* OCSP, ejecuciones manuales del *cron* de CRL) sin afectar al servicio real. El entorno de producción se observa pasivamente a través de las tablas operativas (`ocsp_queries`, `crl_records`, `sso_audit_log` y `audit_log`) acumuladas durante la ventana de observación, sin inyección de carga sintética. Las latencias se reportan como p50, p95 y máximo sobre la ventana indicada, no como promedios aritméticos, dado que la distribución en este tipo de servicios presenta cola larga (Dean y Barroso, 2013); cada prueba indica explícitamente el número de muestras N que la sustenta.

Los umbrales de aceptación se justifican como sigue. El umbral del percentil 95 de latencia OCSP en 500 milisegundos sigue las recomendaciones operativas de Chandramouli (2022, NIST SP 800-204C) sobre validación de identidad en arquitecturas distribuidas, ajustadas a un *responder* que firma cada respuesta en línea sin caché ni prefirmado. El umbral de 5 segundos para el flujo OAuth federado completo se deriva del estudio de Nielsen (1993) sobre tiempos de respuesta tolerables en interacción humano-computador, que sitúa los diez segundos como límite del enganche atencional del usuario; el margen restante absorbe la reacción humana al introducir el segundo factor TOTP. El umbral de 5 segundos para la propagación de revocación a OCSP se justifica por la regeneración inmediata de la respuesta firmada al ejecutar `revokeCertificate` (apartado 7.4.5). El umbral de 30 segundos para la propagación a CRL es operativo y refleja el tiempo entre revocación y publicación física del fichero, no el `nextUpdate` declarado en la propia CRL, que admite hasta varias horas conforme al apartado 7.8 y a RFC 5280 sección 5.

8.3.2 Prueba 1: Discovery OIDC y JWKS

Verifica que los *endpoints* normativos de OIDC (`/.well-known/openid-configuration` conforme a RFC 8414 y *OIDC Discovery* 1.0, y `/api/jwks.php` conforme a RFC 7517) están publicados, son cacheables y contienen los campos obligatorios. Los procedimientos `curl` y `jq` se publican en el **MVP**.

Criterio	Umbral de aceptación
HTTP discovery	`200`
Tiempo de descarga del discovery	< 300 ms
Campos obligatorios del discovery	8 / 8
HTTP JWKS	`200`
Número de claves en JWKS	≥ 1
`Cache-Control` JWKS	`public, max-age ≥ 3600`

Tabla 8.7.a. Verificación del descubrimiento OIDC y del JWKS.

8.3.3 Prueba 2: flujo SSO a PKI con Authorization Code y PKCE

Mide el tiempo extremo a extremo del inicio de sesión federado entre el portal PKI (cliente OAuth) y el SSO, desde el inicio del *login* en el RP hasta la sesión activa en el RP, y verifica que el intercambio respeta RFC 7636 sección 4.6. La medición se sostiene sobre los eventos correlativos de `sso_audit_log` (`login_success`, `authorization_code_issued`, `token_exchange_success`) y sobre el porcentaje de *authorization codes* con `code_challenge` no nulo y método `S256`. Las consultas SQL se publican en el **MVP**.

Criterio	Umbral de aceptación
p50 de *login* SSO a PKI (extremo a extremo)	< 3 s
p95 de *login* SSO a PKI	< 5 s

Criterio	Umbral de aceptación
p95 del *back-channel* `/token` (RP a SSO)	< 500 ms
Porcentaje de *authorization codes* con PKCE (últimos 7 días)	≥ 80 %
Porcentaje de PKCE con método `S256`	100 %
Tests PKCE estándar (`sso/tests/test_pkce.php`)	29 / 29

Tabla 8.7.b. Resultados del flujo OAuth con PKCE y OIDC. El umbral de 5 segundos absorbe la reacción humana al segundo factor TOTP cuando el usuario no tenía sesión previa en el SSO.

8.3.4 Prueba 3: emisión de certificado

Mide la latencia desde la aprobación de una solicitud hasta la disponibilidad del certificado para descarga, y verifica la integridad criptográfica del resultado mediante `openssl verify` y la inspección de las extensiones X.509 críticas. Las consultas SQL y los comandos OpenSSL se publican en el **MVP**.

Criterio	Umbral de aceptación
p50 aprobación a emisión	< 3 s
p95 aprobación a emisión	< 5 s
Algoritmo de firma observado	`sha256WithRSAEncryption`
Tamaño de clave observado	≥ 2048
`openssl verify` cadena completa	`OK`
Presencia de `crlDistributionPoints` (CDP)	Sí

Criterio	Umbral de aceptación
Presencia de `authorityInfoAccess` (AIA OCSP)	Sí

Tabla 8.7.c. Emisión de certificado y verificación criptográfica externa.

8.3.5 Prueba 4: revocación y propagación a OCSP y CRL

Mide el tiempo de propagación entre la revocación de un certificado en el panel administrativo y la primera respuesta OCSP `revoked` (T1) y la presencia del *serial* en la CRL pública (T2). El protocolo emplea bucles de muestreo controlado contra el *responder* (`openssl ocsp`) y la CRL (`openssl crl -text`), reproducibles con los comandos publicados en el **MVP**.

Criterio	Umbral de aceptación
T1 menos T0 (revocación a OCSP `revoked`)	< 5 s
T2 menos T0 (revocación a CRL publica el *serial*)	< 30 s
`revocationReason` reportado por OCSP coincide con BD	Idéntico
`crlNumber` aumenta tras la revocación	Estrictamente creciente
Firma de la nueva CRL con `openssl crl -verify`	`verify OK`

Tabla 8.7.d. Tiempo de propagación de la revocación. El umbral de 30 segundos para CRL es operativo y refleja el tiempo entre revocación y publicación del fichero, no el `nextUpdate` (que admite hasta varias horas según el apartado 7.7).

8.3.6 Prueba 5: rendimiento del responder OCSP

Caracteriza la latencia del *responder* en condiciones reales sobre el tráfico acumulado en la tabla `ocsp_queries` y mediante carga activa controlada (N = 200 consultas con `openssl ocsp`). El SQL completo de cálculo de p50, p95 y máximo se publica en el **MVP**.

Criterio	Umbral de aceptación
p50 OCSP (24 h)	< 200 ms
p95 OCSP (24 h)	< 500 ms
Máximo OCSP (24 h)	< 2 000 ms
Tasa de respuestas `error`	< 1 %
Disponibilidad mensual del *responder*	> 99 %
Coherencia OCSP frente a BD (muestra de 30 *seriales*)	30 / 30

Tabla 8.7.e. Rendimiento y disponibilidad del responder OCSP. El umbral del p95 en 500 milisegundos corresponde a un responder que firma cada respuesta en línea sin prefirmado; la cifra "ideal prefirmado" del orden de 100 milisegundos figura en el apartado 9.4 como vía futura.

8.3.7 Prueba 6: rotación y generación de CRL

Verifica que el *timer* `pki-crl-rotate.timer` se ejecuta dentro del intervalo previsto, que la regeneración es rápida y que la cadena `this_update` y `next_update` queda cubierta por la siguiente rotación. La consulta SQL sobre `crl_records` con cálculo de la latencia entre rotaciones consecutivas y la verificación de la firma con `openssl crl -verify` se publican en el **MVP**.

Criterio	Umbral de aceptación
Duración de la regeneración	< 10 s
Intervalo medio entre rotaciones (`source = 'cron'`)	240 ± 5 min
Ejecuciones extra por revocación (`source = 'revocation'`)	≥ 1 por revocación

Criterio	Umbral de aceptación
`next_update` menos `this_update` declarado	≥ 4 h
`crlNumber` estrictamente creciente sobre 7 días	Sí
`openssl crl -verify` sobre CRL pública	`verify OK`
Tamaño medio de las CRL publicadas	< 50 KB

Tabla 8.7.f. Generación, intervalo y firma de las CRL.

8.3.8 Síntesis de resultados

La Tabla 8.8 consolida los cuatro indicadores principales alineados con los requisitos no funcionales del apartado 7.2.2 y con los umbrales declarados en cada subprueba.

#	Indicador principal	Umbral de aceptación
1	Percentil 95 de latencia OCSP (apartado 8.3.6)	< 500 ms
2	Percentil 95 del *login* federado SSO a PKI (apartado 8.3.3)	< 5 s
3	Duración de la generación de CRL (apartado 8.3.7)	< 10 s
4	Propagación de la revocación a OCSP (apartado 8.3.5)	< 5 s

Tabla 8.8. Síntesis cuantitativa del apartado 8.3.

Junto a estos cuatro indicadores, el sistema acumula evidencia secundaria sobre integridad criptográfica (la cadena valida con `openssl verify`, las CRL validan con `openssl crl -verify` y las respuestas OCSP llevan firma de un *responder* dedicado), trazabilidad (`audit_log` y `sso_audit_log` registran cada operación crítica) e interoperabilidad estándar (la CLI de OpenSSL de un cliente arbitrario reproduce las operaciones sin ajustes propietarios). Los cuatro umbrales reflejan objetivos operativos para un despliegue monoinstancia sobre VPS, no estándares de la industria a escala global; los caminos para reducirlos (OCSP prefirmado, caché L7 y *responder* dedicado a múltiples CA) están identificados en el apartado 9.4 como vías futuras.

8.4 Limitaciones del programa de pruebas

Las limitaciones del programa de pruebas se declaran de forma explícita para que el lector calibre el alcance real de la validación. Seis son estructurales (Tabla 8.9) y se complementan con dos limitaciones contextuales del entorno de prueba.

#	Limitación	Impacto en la validación	Mitigación
L-01	Protocolo de evaluación de usabilidad no ejecutado dentro del plazo (apartado 8.2)	La validación descansa en las pruebas técnicas y en el Anexo B, sin datos cualitativos	Diseño íntegramente documentado para ejecución posterior; no se publican métricas SUS no medidas
L-02	Perfil de captación previsto homogéneo (formación universitaria y uso	Menor representatividad para perfiles fuera de ese rango (usuarios mayores sin uso	Criterios de exclusión documentados; accesibilidad WCAG completa como vía futura

#	Limitación	Impacto en la validación	Mitigación
	habitual de internet)	digital, accesibilidad visual)	
L-03	Sesgo de proximidad previsto: captación en círculos cercanos al desarrollo	Riesgo de amabilidad excesiva y menor disposición a criticar	Protocolo *think-aloud* para captar confusión no verbalizada como crítica
L-04	Ausencia de pruebas longitudinales en el protocolo (interacción única por participante)	No se mide curva de aprendizaje ni retención a medio plazo	Vía futura
L-05	Pruebas de rendimiento sintéticas, no replicación de patrones de uso reales	Los valores reflejan condiciones controladas, no la operación cotidiana	Métricas reales acumuladas por `ocsp_queries` y logs de producción
L-06	Adopción de *scripts* PHP CLI en lugar de PHPUnit completo	Cobertura limitada de pruebas unitarias automáticas a aproximadamente 60 casos	Migración a PHPUnit registrada como vía futura (apartado 9.4)

Tabla 8.9. Limitaciones estructurales del programa de pruebas y su mitigación.

Las dos limitaciones contextuales adicionales son las siguientes. La primera: las pruebas se realizan sobre el despliegue actual monoinstancia, sin ejecución en una configuración con balanceador, dado que el alcance del proyecto no contempla escalado horizontal en esta versión. La segunda: los certificados FNMT y DNI electrónico utilizados son los propios de los participantes o emitidos por la PKI del proyecto a efectos de prueba, lo que limita la diversidad de emisores reales probados; la heterogeneidad práctica de los emisores queda parcialmente cubierta por las reglas declarativas de la tabla `sso_trusted_issuers`.

8.5 Escalabilidad, mantenimiento y formación

8.5.1 Capacidad y caminos de evolución

El despliegue actual es monoinstancia con base de datos local, en coherencia con el público objetivo del capítulo 2 (organizaciones pequeñas con cientos de usuarios). El diseño no impide el crecimiento, pero no incorpora escalado horizontal automático. La Tabla 8.10 sintetiza la capacidad estimada de cada eje y el camino de evolución asociado, identificado, pero no implementado en este trabajo. El sistema está diseñado para una vida útil mínima de tres años sin reescritura significativa: PHP 8.4 alcanza fin de soporte en noviembre de 2028, MariaDB 10.5 lo hizo en junio de 2025 y se migrará a 10.11 LTS en el primer semestre tras la entrega, OpenSSL 3.x mantiene soporte más allá de 2026, Apache 2.4 no tiene fin de vida anunciado, la CA raíz es válida durante veinte años con rotación programada antes del año dieciocho y las CA intermedias son válidas durante diez años con rotación antes del año ocho.

Escenario	Capacidad estimada	Cuello de botella	Camino de evolución
Usuarios activos simultáneos	500-1 000	PHP-FPM en VPS de 4 vCPU	Aumento de recursos del VPS o *pool* externo
Certificados emitidos/día	100-200	Aprobación manual humana	Aprobación automática por

Escenario	Capacidad estimada	Cuello de botella	Camino de evolución
			reglas (soportada en el modelo)
Consultas OCSP/segundo	50-100	Generación en línea sin caché	Prefirmado o caché de capa 7
Sesiones activas	10 000	Tabla `sso_sessions` en BD local	Migración de sesiones a Redis (cambio acotado)
Aplicaciones cliente	Sin límite práctico	-	-

Tabla 8.10. Capacidad estimada del despliegue monoinstancia y caminos de evolución.

8.5.2 Mantenimiento operativo

La operación cotidiana requiere dos perfiles. El administrador funcional (no técnico en seguridad) gestiona usuarios, aplicaciones cliente y solicitudes de certificado, con una dedicación previsible de aproximadamente treinta minutos semanales en condiciones nominales. El administrador técnico (con conocimientos de Linux y PKI) mantiene el VPS, supervisa los *timers* `systemd`, gestiona las renovaciones de Let's Encrypt y ejecuta procedimientos de recuperación, con una dedicación de una a dos horas mensuales. Las tareas automatizadas mediante los *timers* descritos en el apartado 7.10.1 (rotación de CRL, regeneración del *responder* OCSP cuando caduca, limpieza de tablas temporales, *backups* y alertas) no requieren intervención humana. Los eventos no rutinarios (emisión de una nueva CA intermedia, rotación de la clave maestra del entorno, respuesta a alertas críticas como la proximidad del fin de validez de una CA o el fallo persistente de un *timer*) se documentan mediante *runbooks* en el Manual del Administrador.

8.5.3 Plan de formación

La formación se estructura en tres niveles. Para el usuario final el objetivo es que una persona sin experiencia previa pueda registrarse, iniciar sesión y completar el ciclo de solicitud, descarga y revocación de un certificado en menos de treinta minutos; el material asociado incluye el **MU** (Manual de Usuario) de aproximadamente treinta páginas con capturas y pasos numerados, una sección de preguntas frecuentes integrada en el portal y, opcionalmente, tres microvídeos de dos minutos sobre los flujos más solicitados. Para el administrador funcional el objetivo es la gestión autónoma de usuarios, aplicaciones cliente y solicitudes de certificado sin asistencia técnica; el material principal es el **MA** (sección funcional, aproximadamente ochenta páginas) complementado con *runbooks* específicos (creación de un usuario, registro de una aplicación, revocación de emergencia y consulta de auditoría) y una sesión de transferencia de dos horas con casos reales sobre el entorno de preproducción. Para el administrador técnico el objetivo es el mantenimiento del sistema y la aplicación de procedimientos de recuperación; el material incluye el **MI** (despliegue desde cero), la sección técnica del **MA** (*backup* y restauración, rotación de claves, intervención manual sobre `pki/ca/`, lectura de `journalctl`), el **MAR** (Manual de la API REST) y una sesión de transferencia de cuatro horas con ejercicios de simulación de incidencias.

CAPÍTULO 9. CONCLUSIONES

El capítulo contrasta los resultados obtenidos con los objetivos formulados en el apartado 1.4, sintetiza las conclusiones técnicas y metodológicas extraídas del trabajo, enumera las limitaciones estructurales del sistema y describe las vías futuras de evolución previstas.

9.1 Cumplimiento de los objetivos

El sistema desplegado satisface el objetivo general formulado en el apartado 1.4.1 y la práctica totalidad de los ocho objetivos específicos enunciados en el apartado 1.4.2. La autoridad certificadora propia opera con una raíz y tres intermedias activas (sección 7.4.1) y emite los seis perfiles X.509 v3 documentados en la sección 7.4.2 (Server Authentication, Client Authentication, Code Signing, Document Signing, S/MIME y VPN), todos verificables con `openssl x509 -text` y con extensiones AIA, CDP y *Certificate Policies* correctamente cumplimentadas. Los servicios de validación están operativos: el *responder* OCSP, ajustado a la RFC 6960 (Santesson et al., 2013), atiende peticiones por POST y por GET con respuestas firmadas por un *responder* delegado y registra cada consulta en `ocsp_queries`; el sistema de CRL regenera los listados cada cuatro horas mediante un *timer* `systemd` y los publica de forma inmediata, conservando histórico con resumen SHA-256 en `crl_records`. El subsistema SSO implementa el flujo *Authorization Code* con PKCE S256 (RFC 7636) y rechaza explícitamente el método `plain`, conforme a la RFC 9700, con verificación experimental de los veintinueve vectores oficiales (apartado 8.1.2 y Anexo B). La integración federada se completa con la publicación del documento de *discovery* y del JWKS conforme a OpenID Connect Core 1.0 (Sakimura et al., 2014) y RFC 7517. Los dos métodos de autenticación previstos (contraseña con segundo factor TOTP y certificado FNMT sobre mTLS) están operativos.

Cuatro objetivos se cumplieron de manera parcial, con matización explícita: la operación de la raíz “fuera de línea” en su forma fuerte se sustituye por un régimen de uso restringido (clave cifrada en disco con *passphrase* doblemente cifrada, código que rechaza la firma de certificados de usuario final con la raíz); el vinculación de sesión por IP y huella del agente de usuario se entiende como señal contextual de riesgo, no como garantía criptográfica de posesión del *token*

(las garantías reales requieren mTLS *Token Binding*, RFC 8705, o DPoP, RFC 9449); el registro de auditoría se firma a nivel de fila y se protege con permisos de base de datos pero sin encadenamiento criptográfico de filas; y la rotación periódica programada de claves de firma JWT dispone de la infraestructura (modelo `kid` operativo, *runbook* documentado) pero no de la automatización. Un objetivo no se ha cumplido: la interfaz multilingüe operativa en español, valenciano (Normas del Puig de la RACV), italiano e inglés. La arquitectura separa contenido y lenguaje, pero la traducción a los tres idiomas adicionales no se ha completado dentro del plazo y se reclasifica como acción futura planificada.

La verificación experimental descansa sobre cuatro tipos de evidencia reproducibles por un tercero: los veintinueve vectores oficiales de PKCE (RFC 7636 sección B), los veintidós requisitos funcionales contrastados en el Anexo B (RF-01 a RF-22), los quince escenarios STRIDE (S-01 a S-15) y las mediciones operativas del apartado 8.3 sobre las tablas `ocsp_queries`, `sso_audit_log`, `audit_log` y `crl_records`.

9.2 Conclusiones técnicas y metodológicas

Tres conclusiones técnicas se extraen del trabajo. La primera se refiere al coste real de operar una PKI: emitir un certificado X.509 v3 conforme a RFC 5280 (Cooper et al., 2008) con OpenSSL CLI requiere unas pocas líneas de configuración, mientras que el coste real se concentra en la gestión del estado (coherencia entre `index.txt`, tablas de base de datos y ficheros del sistema, rotación de CRL y propagación inmediata de revocaciones a OCSP y CRL). La decisión de apoyarse en la CLI oficial de OpenSSL en lugar de depender únicamente de la extensión nativa `ext-openssl` alinea el código con la práctica de operadores PKI maduros (Adams y Lloyd, 2003) y permite que cualquier auditor familiarizado con OpenSSL inspeccione la infraestructura sin leer el código PHP. La restricción `pathlen:0` grabada en las CA intermedias (RFC 5280 sección 6.1.4 (k)) ejemplifica la defensa estructural sobre la procedimental: la limitación está embebida en el certificado y la aplica cualquier validador X.509 conforme.

La segunda conclusión técnica se refiere a la madurez actual de OAuth 2.0. La práctica profesional reciente, codificada en la RFC 9700 y en las consideraciones de seguridad y privacidad de OpenID Connect (Sakimura et al., 2014), considera

insuficiente el flujo *Authorization Code* sin PKCE incluso para clientes confidenciales. La caracterización del *binding* de sesión como mecanismo contextual y no como garantía criptográfica coincide con el análisis de Hardt (2012) y con el modelo de amenazas implícito en RFC 8705.

La tercera conclusión técnica se refiere a la concepción multiplicativa de la defensa en profundidad. Las nueve capas descritas en la sección 7.8 actúan como composición y no como suma de filtros independientes: el riesgo residual de una clase de ataque es la intersección de las capas que el ataque debe atravesar. El análisis STRIDE por componente del apartado 7.9 produce un mapa que identifica de forma sistemática las zonas con mitigación parcial donde el riesgo residual debe declararse de forma explícita.

La conclusión metodológica se refiere al carácter híbrido del proceso. La combinación de Métrica v3 como marco general (Ministerio de Hacienda y Administraciones Públicas, 2001) con sprints quincenales en Jira y prácticas de *documentation as code* resulta adecuada para un proyecto individual; la adopción literal de Scrum, con sus ceremonias presupuestas para un equipo, habría introducido sobrecoste sin contrapartida. El uso de asistentes de IA como acelerador y revisor, no como sustituto del juicio técnico, se declara con detalle en el Anexo D y se considera la postura más defensible para un trabajo académico contemporáneo (American Psychological Association, 2023).

9.3 Limitaciones generales del proyecto

Las limitaciones del programa de pruebas se declararon en el apartado 8.4 y no se reiteran. Esta sección recoge seis limitaciones estructurales del sistema y la mitigación asociada a cada una. La primera limitación es el despliegue monoinstancia: el sistema opera sobre un único VPS sin alta disponibilidad activa-activa ni balanceador, modelo aceptable para el público objetivo (organizaciones pequeñas con cientos de usuarios) pero corto para escenarios de mayor exigencia; la mitigación pasa por migrar las sesiones a un *backend* compartible (Redis), levantar una segunda instancia tras un balanceador y replicar la base de datos en modo activo-pasivo. La segunda es la ausencia de un módulo de seguridad *hardware*: las claves privadas de las CA intermedias se almacenan cifradas en el sistema de ficheros con `0600` y *passphrase* cifrada con

la clave maestra del entorno, arquitectura que no equivale a un HSM certificado FIPS 140-2 nivel 3; la mitigación pasa por la integración con un HSM físico (Thales Luna, AWS CloudHSM) o un HSM *software* con *boundaries* reforzados. La tercera es la inalterabilidad limitada del registro de auditoría: la firma a nivel de fila no se complementa con encadenamiento criptográfico entre filas, por lo que la mitigación pasa por implementar *hash-chain* y *anchoring* periódico del resumen en un servicio externo (correo electrónico con sello temporal, *blockchain* pública u OpenTimestamps). La cuarta es la cobertura acotada de métodos de autenticación: solo dos están operativos, dejando fuera del alcance temporal el DNI electrónico, *smartcards* genéricas vía PKCS#11 y *Passkeys* / FIDO2 / WebAuthn (FIDO Alliance, 2022; W3C, 2021). La quinta es la validación únicamente en español, lo que restringe el público objetivo a la comunidad hispanohablante. La sexta es la ausencia de certificación formal: el sistema no se ha sometido a evaluación contra esquemas reconocidos (ENS categoría Alta, ETSI EN 319 411, ISO 27001), por lo que no puede operar como *Trust Service Provider* cualificado en el sentido del Reglamento eIDAS (Parlamento Europeo y Consejo, 2024).

9.4 Vías futuras

Doce líneas de evolución se identifican como continuación natural del proyecto, agrupadas en cuatro bloques temáticos.

El bloque A (ampliación de métodos de autenticación) incluye cuatro vías: aceptar como método de *login* los certificados *clientAuth* emitidos por las propias intermedias del sistema (prioridad alta, una o dos semanas); soporte completo del DNI electrónico, incluyendo DNIe 4.0 con interfaz NFC (dos o tres semanas con *hardware* real); soporte de *smartcards* genéricas mediante PKCS#11 sobre varias familias (Gemalto, Athena, IDEMIA); e incorporación de *Passkeys* / FIDO2 / WebAuthn (prioridad alta, tres o cuatro semanas).

El bloque B (infraestructura y producto) incluye la federación con Microsoft Entra ID mediante SAML 2.0 u OpenID Connect, la aplicación móvil progresiva del autenticador con Capacitor, ZXing y Android Keystore, y la internacionalización efectiva a valenciano (Normas del Puig de la RACV), italiano e inglés (única vía que cierra un objetivo específico no cumplido).

El bloque C (protocolos avanzados y criptografía) incluye el *Token Binding* criptográfico mediante mTLS (RFC 8705) o DPoP (RFC 9449), que sustituiría el *binding* contextual por uno verificable criptográficamente; el encadenamiento criptográfico de la auditoría con *anchoring* periódico; y la rotación programada de las claves de firma JWT mediante un *timer* `systemd` mensual.

El bloque D (evolución institucional) agrupa cinco direcciones avanzadas que abren el sistema a casos de uso institucionales: integración con HSM físico para cumplir la limitación L-G2, criptografía poscuántica con certificados híbridos (ML-KEM y ML-DSA del NIST), implementación de un servidor ACME propio (RFC 8555), federación SAML 2.0 con federaciones académicas (RedIRIS, eduGAIN) y publicación en *logs* de *Certificate Transparency* (RFC 6962).

Las vías de prioridad alta (internacionalización, certificados propios como método de *login*, DNI electrónico, *Passkeys* y DPoP) cubren los huecos más visibles del trabajo entregado y son ejecutables en un horizonte de tres a cuatro meses; el resto se distribuye en horizontes de medio y largo plazo según prioridad y dependencias.

9.5 Cierre

El sistema desplegado integra, sobre un único servidor virtual privado y empleando exclusivamente componentes libres, una autoridad certificadora con jerarquía de dos niveles y seis perfiles X.509 v3, los servicios de validación OCSP y CRL conformes con RFC 6960 y RFC 5280, un proveedor de identidad SSO basado en OAuth 2.0 con PKCE y OpenID Connect *Discovery*, dos métodos de autenticación operativos (contraseña con TOTP y certificado FNMT sobre TLS mutuo) y nueve capas de seguridad transversales. La verificación experimental se sostiene sobre los veintinueve vectores oficiales de PKCE, la matriz de los veintidós requisitos funcionales, los quince escenarios STRIDE y los cuatro indicadores operativos del apartado 8.3.8.

De los ocho objetivos específicos formulados en el apartado 1.4.2, tres se alcanzaron de forma íntegra (jerarquía y perfiles, OCSP, CRL), cuatro de forma parcial con la matización del apartado 9.1 (raíz en uso restringido, *binding* contextual de sesión, auditoría con firma por fila, rotación JWT manual) y uno

quedó pendiente (interfaz multilingüe efectiva). Las seis limitaciones del apartado 9.3 y las doce vías futuras del apartado 9.4 conforman un plan de continuación con horizontes que oscilan entre tres meses y varios años.

CAPÍTULO 10. BIBLIOGRAFÍA

10.1 RFCs e IETF

Adams, C., Cain, P., Pinkas, D., & Zuccherato, R. (2001). *Internet X.509 public key infrastructure time-stamp protocol (TSP)* (RFC 3161). Internet Engineering Task Force. https://doi.org/10.17487/RFC3161

M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., & Ranen, O. (2005). *HOTP: An HMAC-based one-time password algorithm* (RFC 4226). Internet Engineering Task Force. https://doi.org/10.17487/RFC4226

Deacon, A., & Hurst, R. (2007). *The lightweight online certificate status protocol (OCSP) profile for high-volume environments* (RFC 5019). Internet Engineering Task Force. https://doi.org/10.17487/RFC5019

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., & Polk, W. (2008). *Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile* (RFC 5280). Internet Engineering Task Force. https://doi.org/10.17487/RFC5280

Eastlake, D. (2011). *Transport layer security (TLS) extensions: Extension definitions* (RFC 6066). Internet Engineering Task Force. https://doi.org/10.17487/RFC6066

M'Raihi, D., Machani, S., Pei, M., & Rydell, J. (2011). *TOTP: Time-based one-time password algorithm* (RFC 6238). Internet Engineering Task Force. https://doi.org/10.17487/RFC6238

Hardt, D. (2012). *The OAuth 2.0 authorization framework* (RFC 6749). Internet Engineering Task Force. https://doi.org/10.17487/RFC6749

Hodges, J., Jackson, C., & Barth, A. (2012). *HTTP strict transport security (HSTS)* (RFC 6797). Internet Engineering Task Force. https://doi.org/10.17487/RFC6797

Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., & Adams, C. (2013). *X.509 internet public key infrastructure online certificate status protocol - OCSP* (RFC 6960). Internet Engineering Task Force. https://doi.org/10.17487/RFC6960

Laurie, B., Langley, A., & Kasper, E. (2013). *Certificate transparency* (RFC 6962). Internet Engineering Task Force. https://doi.org/10.17487/RFC6962

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON web token (JWT)* (RFC 7519). Internet Engineering Task Force. https://doi.org/10.17487/RFC7519

Sakimura, N., Bradley, J., & Agarwal, N. (2015). *Proof key for code exchange by OAuth public clients* (RFC 7636). Internet Engineering Task Force. https://doi.org/10.17487/RFC7636

Denniss, W., & Bradley, J. (2017). *OAuth 2.0 for native apps* (RFC 8252). Internet Engineering Task Force. https://doi.org/10.17487/RFC8252

Rescorla, E. (2018). *The transport layer security (TLS) protocol version 1.3* (RFC 8446). Internet Engineering Task Force. https://doi.org/10.17487/RFC8446

Popov, A., Nyström, M., Balfanz, D., & Langley, A. (2018). *The token binding protocol version 1.0* (RFC 8471). Internet Engineering Task Force. https://doi.org/10.17487/RFC8471

Barnes, R., Hoffman-Andrews, J., McCarney, D., & Kasten, J. (2019). *Automatic certificate management environment (ACME)* (RFC 8555). Internet Engineering Task Force. https://doi.org/10.17487/RFC8555

Campbell, B., Bradley, J., Sakimura, N., & Lodderstedt, T. (2020). *OAuth 2.0 mutual-TLS client authentication and certificate-bound access tokens* (RFC 8705). Internet Engineering Task Force. https://doi.org/10.17487/RFC8705

Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., & Waite, D. (2023). *OAuth 2.0 demonstrating proof of possession (DPoP)* (RFC 9449). Internet Engineering Task Force. https://doi.org/10.17487/RFC9449

Lodderstedt, T., Bradley, J., Labunets, A., & Fett, D. (2025). *Best current practice for OAuth 2.0 security* (RFC 9700). Internet Engineering Task Force. https://doi.org/10.17487/RFC9700

10.2 Estándares NIST y OWASP

Dworkin, M. (2001). *NIST special publication 800-38A: Recommendation for block cipher modes of operation*. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-38A

Grassi, P. A., Garcia, M. E., & Fenton, J. L. (2017). *NIST special publication 800-63B: Digital identity guidelines - Authentication and lifecycle management*. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-63b

Kent, K., & Souppaya, M. (2006). *NIST special publication 800-92: Guide to computer security log management*. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-92

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *NIST special publication 800-207: Zero trust architecture*. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207

Chandramouli, R. (2022). *NIST special publication 800-204C: Implementation of DevSecOps for a microservices-based application with service mesh*. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-204C

OWASP Foundation. (2021). *OWASP Top 10 - 2021*. https://owasp.org/Top10/

OWASP Foundation. (2019). *Application Security Verification Standard (ASVS) v4.0.3*. https://owasp.org/www-project-application-security-verification-standard/

OWASP Foundation. (2024). *Password storage cheat sheet*. https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

OWASP Foundation. (2024). *Cross-site request forgery (CSRF) prevention cheat sheet*. https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

OWASP Foundation. (2024). *Authentication cheat sheet - Brute force prevention*. https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

OWASP Foundation. (2024). *Secure Headers Project*. https://owasp.org/www-project-secure-headers/

OWASP Foundation. (2024). *Cross-site scripting (XSS) prevention cheat sheet*. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

10.3 Normativa europea y española

Comisión Europea. (2014). Reglamento (UE) n.º 910/2014 del Parlamento Europeo y del Consejo, de 23 de julio de 2014, relativo a la identificación electrónica y los servicios de confianza para las transacciones electrónicas en el mercado interior y por el que se deroga la Directiva 1999/93/CE [eIDAS]. *Diario Oficial de la Unión Europea*, L 257, 73-114. https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX:32014R0910

Parlamento Europeo y Consejo de la Unión Europea. (2016). Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos) [RGPD]. *Diario Oficial de la Unión Europea*, L 119, 1-88. https://eur-lex.europa.eu/eli/reg/2016/679/oj

Jefatura del Estado. (2018). Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales [LOPDGDD]. *Boletín Oficial del Estado*, 294, 119788-119857. https://www.boe.es/eli/es/lo/2018/12/05/3

Ministerio de Asuntos Económicos y Transformación Digital. (2022). Real Decreto 311/2022, de 3 de mayo, por el que se regula el Esquema Nacional de Seguridad. *Boletín Oficial del Estado*, 106, 60472-60525. https://www.boe.es/eli/es/rd/2022/05/03/311

Parlamento Europeo y Consejo de la Unión Europea. (2024). Reglamento (UE) 2024/1183 del Parlamento Europeo y del Consejo, de 11 de abril de 2024, por el que se modifica el Reglamento (UE) n.º 910/2014 en lo que respecta al establecimiento del marco europeo de identidad digital [eIDAS 2]. *Diario Oficial de la Unión Europea*, L 1183. https://eur-lex.europa.eu/eli/reg/2024/1183/oj

European Telecommunications Standards Institute. (2023). *ETSI EN 319 411-1 V1.4.1: Electronic signatures and trust infrastructures (ESI); Policy and security requirements for trust service providers issuing certificates*. ETSI. https://www.etsi.org/deliver/etsi_en/319400_319499/31941101/

CA/Browser Forum. (2024). *Baseline requirements for the issuance and management of publicly-trusted certificates, version 2.0.5*. https://cabforum.org/working-groups/server/baseline-requirements/documents/

International Organization for Standardization. (2022). *ISO/IEC 27001:2022 - Information security, cybersecurity and privacy protection - Information security management systems - Requirements*. ISO. https://www.iso.org/standard/27001

10.4 PKI, OAuth e OIDC: libros y artículos académicos

Adams, C., & Lloyd, S. (2003). *Understanding PKI: Concepts, standards, and deployment considerations* (2nd ed.). Addison-Wesley.

Anderson, R. J. (2020). *Security engineering: A guide to building dependable distributed systems* (3rd ed.). Wiley. https://www.cl.cam.ac.uk/~rja14/book.html

Bonneau, J., Herley, C., Van Oorschot, P. C., & Stajano, F. (2012). The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, 553-567. https://doi.org/10.1109/SP.2012.44

Ferguson, N., Schneier, B., & Kohno, T. (2010). *Cryptography engineering: Design principles and practical applications*. Wiley.

Fett, D., Küsters, R., & Schmitz, G. (2017). The web SSO standard OpenID Connect: In-depth formal security analysis and security guidelines. *Proceedings*

of the 2017 IEEE 30th Computer Security Foundations Symposium, 189-202. https://doi.org/10.1109/CSF.2017.20

Howard, M., & LeBlanc, D. (2003). *Writing secure code* (2nd ed.). Microsoft Press.

Katz, J., & Lindell, Y. (2020). *Introduction to modern cryptography* (3rd ed.). CRC Press.

Schneier, B. (2015). *Applied cryptography: Protocols, algorithms, and source code in C* (20th anniversary ed.). Wiley.

Shostack, A. (2014). *Threat modeling: Designing for security*. Wiley.

Stallings, W. (2017). *Cryptography and network security: Principles and practice* (7th ed.). Pearson.

Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., & Mortimore, C. (2014). *OpenID Connect Core 1.0 incorporating errata set 2*. OpenID Foundation. https://openid.net/specs/openid-connect-core-1_0.html

Sakimura, N., Bradley, J., Jones, M., & Jay, E. (2014). *OpenID Connect Discovery 1.0 incorporating errata set 2*. OpenID Foundation. https://openid.net/specs/openid-connect-discovery-1_0.html

Dean, J., & Barroso, L. A. (2013). The tail at scale. *Communications of the ACM*, 56(2), 74-80. https://doi.org/10.1145/2408776.2408794

Sandhu, R., Ferraiolo, D., & Kuhn, R. (2000). The NIST model for role-based access control: Towards a unified standard. *Proceedings of the 5th ACM Workshop on Role-Based Access Control*, 47-63. https://doi.org/10.1145/344287.344301

10.5 Usabilidad y System Usability Scale

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3), 114-123. https://uxpajournal.org/determining-what-individual-sus-scores-mean-adding-an-adjective-rating-scale/

Brooke, J. (1996). SUS: A “quick and dirty” usability scale. En P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), *Usability evaluation in industry* (pp. 189-194). Taylor & Francis.

Lewis, C., & Rieman, J. (1993). *Task-centered user interface design: A practical introduction*. University of Colorado, Boulder. https://hcibib.org/tcuid/

Lewis, J. R. (2018). The System Usability Scale: Past, present, and future. *International Journal of Human-Computer Interaction*, 34(7), 577-590. https://doi.org/10.1080/10447318.2018.1455307

Nielsen, J. (1993). *Usability engineering*. Academic Press.

Nielsen, J. (1994). *Heuristic evaluation*. En J. Nielsen & R. L. Mack (Eds.), *Usability inspection methods* (pp. 25-62). Wiley.

Nielsen, J. (2000, March 19). *Why you only need to test with 5 users*. Nielsen Norman Group. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

Sauro, J. (2011, February 2). *Measuring usability with the System Usability Scale (SUS)*. MeasuringU. https://measuringu.com/sus/

Sauro, J., & Lewis, J. R. (2016). *Quantifying the user experience: Practical statistics for user research* (2nd ed.). Morgan Kaufmann.

Tullis, T., & Albert, B. (2013). *Measuring the user experience: Collecting, analyzing, and presenting usability metrics* (2nd ed.). Morgan Kaufmann.

10.6 Herramientas y tecnologías (documentación oficial)

AlmaLinux OS Foundation. (s. f.). *AlmaLinux 9 documentation*. https://wiki.almalinux.org/

Bergmann, S. (2024). *PHPUnit manual: The PHP testing framework*. Sebastian Bergmann. https://docs.phpunit.de/

Apache Software Foundation. (s. f.). *Apache HTTP Server documentation, version 2.4*. https://httpd.apache.org/docs/2.4/

Atlassian. (s. f.). *Jira Software documentation*. https://support.atlassian.com/jira-software-cloud/

Composer Authors. (s. f.). *Composer documentation*. https://getcomposer.org/doc/

MariaDB Foundation. (s. f.). *MariaDB Server documentation*. https://mariadb.com/kb/en/

GitHub Inc. (s. f.). *GitHub Docs - Pull requests and branches*. https://docs.github.com/en

Internet Security Research Group. (s. f.). *Let's Encrypt documentation*. https://letsencrypt.org/docs/

Ionic. (s. f.). *Capacitor documentation*. https://capacitorjs.com/docs

Mozilla Developer Network. (s. f.). *Web crypto API*. https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API

OpenSSL Project Authors. (s. f.). *OpenSSL 3.x manual pages*. https://www.openssl.org/docs/man3.0/

PHP Group. (s. f.). *PHP 8 manual - OpenSSL, PDO, password hashing, sodium*. https://www.php.net/manual/en/

PHP-FIG. (2019). *PSR-4: Autoloader*. https://www.php-fig.org/psr/psr-4/

PHP-FIG. (2019). *PSR-12: Extended coding style*. https://www.php-fig.org/psr/psr-12/

Plesk International GmbH. (s. f.). *Plesk Obsidian administrator guide*. https://docs.plesk.com/

systemd Project. (s. f.). *systemd.timer - Timer unit configuration*. https://www.freedesktop.org/software/systemd/man/systemd.timer.html

ZXing Authors. (2024). *ZXing ("Zebra Crossing") barcode image processing library*. https://github.com/zxing/zxing

10.7 Informes y otras fuentes

American Psychological Association. (2020). *Publication manual of the American Psychological Association* (7th ed.). https://doi.org/10.1037/0000165-000

Anti-Phishing Working Group. (2024). *Phishing activity trends report, Q4 2024*. APWG. https://apwg.org/trendsreports/

Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R., Reifer, D. J., & Steece, B. (2000). *Software cost estimation with COCOMO II*. Prentice Hall.

European Union Agency for Cybersecurity (ENISA). (2023). *ENISA threat landscape 2023*. Publications Office of the European Union. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023

European Union Agency for Cybersecurity (ENISA). (2024). *ENISA threat landscape 2024*. Publications Office of the European Union. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024

FIDO Alliance & World Wide Web Consortium. (2021). *Web authentication: An API for accessing public key credentials - Level 2*. W3C Recommendation. https://www.w3.org/TR/webauthn-2/

Howard, M., LeBlanc, D., & Viega, J. (2009). *24 deadly sins of software security: Programming flaws and how to fix them*. McGraw-Hill.

Ministerio de Política Territorial y Función Pública. (2001). *Métrica versión 3*. Consejo Superior de Administración Electrónica, Gobierno de España. https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html

Verizon. (2024). *2024 data breach investigations report*. Verizon Business. https://www.verizon.com/business/resources/reports/dbir/

World Wide Web Consortium. (2023). *Web content accessibility guidelines (WCAG) 2.2 - W3C Recommendation*. https://www.w3.org/TR/WCAG22/

ANEXO A. CÓDIGO RELEVANTE SELECCIONADO

Este anexo deja constancia, de forma compacta, del código que sostiene las decisiones técnicas del proyecto. El cuerpo de cada fragmento (con el detalle completo de las funciones, los comentarios línea a línea y las trazas a los archivos del repositorio) se publica en el **MCR** (Manual de Código Relevante), también disponible en el repositorio público del proyecto. El presente anexo aporta la justificación de las decisiones, la cobertura sobre los estándares aplicables y la trazabilidad cruzada con los capítulos del cuerpo; el lector que necesite reproducir o auditar las funciones referidas puede consultar el **MCR** del repositorio (`https://github.com/marchernandezmo/TFG-Repo`).

A.1 Cobertura del MCR

El **MCR** se organiza en siete secciones que corresponden a las siete áreas funcionales del proyecto. La primera, *TOTP RFC 6238*, recoge el cifrado del secreto compartido con AES-256-GCM y la verificación del código mediante la ventana de tolerancia de ±30 segundos en `sso/src/Auth/TOTPAuth.php`. La segunda, *firma JWT RS256*, recoge la firma del *access token* con clave en disco e identificador `kid` para rotación, en `sso/src/Session/JWT.php`. La tercera, *PKCE RFC 7636*, recoge la validación del *code challenge* en `/api/authorize.php` y la verificación del *code verifier* en `/api/token.php`, ambas en `sso/src/SSO/SSOProtocol.php`, con rechazo explícito del método `plain`. La cuarta, *motor PKI*, recoge la generación de la *root* CA, la restricción `pathlen:0` estructural en intermedias, los flujos de revocación con regeneración inmediata de CRL y el *responder* OCSP con auditoría por consulta, en `pki/lib/PKIEngine.php` y `pki/lib/OCSPHandler.php`. La quinta, *cifrado jerárquico de passphrases*, recoge la protección AES-256-CBC del fichero `passphrase.key` y el manejo del IV en hexadecimal, en `pki/lib/PKIEngine.php`. La sexta, *verificación cruzada certificado-BD*, recoge la consulta de revocación inmediata desde `sso/src/Auth/CertificateAuth.php` contra la tabla `certificates` del PKI. La séptima, *componentes auxiliares*, agrupa el *bootstrap*, los *middleware* de CSRF, *rate limiting* y auditoría, las plantillas OpenSSL, los *scripts* de `cron` y la configuración de Apache.

A.2 Métricas del código

La Tabla A.1 recoge el tamaño del código operativo del proyecto, agrupado por componente.

Componente	Archivos PHP	LOC aprox.
SSO (`src/`, núcleo)	17	3 500
SSO (`public/api/`, *endpoints*)	6	700
SSO (`public/admin/`, panel)	5	1 500
SSO (vistas y plantillas)	10-15	2 500
PKI (`lib/`, motor y servicios)	9	5 000
PKI (`public/admin/`, panel)	9	2 500
PKI (`public/`, portal)	11	2 000
PKI (`cron/` y `bin/`)	4	400
Responder OCSP	1	150
Repositorios CRL y CA	4	400
SQL (esquema y 12 migraciones)	13	1 500
OpenSSL `.cnf` (raíz y 6 perfiles)	7	300
Apache, Nginx y `.htaccess`	12	250
Tests (`sso/tests/`)	7	1 200
Total código operativo	110	22 000

Tabla A.1. Tamaño del código por componente. La suma total excluye `vendor/`, copias de seguridad y snapshots.

Tres cifras destacan respecto a un proyecto comparable de su tamaño y se justifican por las decisiones tomadas en el capítulo 6: el sistema declara cero dependencias en *runtime* (autoloader, OAuth, JWT, CSRF, *rate limiting* y OCSP

se implementan en el propio repositorio), un único idioma de interfaz (la internacionalización figura como vía futura en el apartado 9.4) y un único módulo monolítico para el motor PKI (`PKIEngine.php` con 1 600 LOC, cuya división en componentes con interfaces estables figura también como vía futura).

A.3 Trazabilidad con el cuerpo

La Tabla A.2 cruza cada bloque del **MCR** con el subapartado del cuerpo donde se justifica la decisión correspondiente.

Bloque del **MCR**	Sección del cuerpo
TOTP RFC 6238	7.6 (autenticación), 7.9 (Spoofing)
Firma JWT RS256	7.5 (SSO), 7.9 (Tampering)
PKCE RFC 7636	7.5.1, 7.5.2
Motor PKI (raíz, intermedias, OCSP, CRL)	7.4.1, 7.4.3, 7.4.5, 7.7, 7.8
Cifrado jerárquico de passphrases	7.4.4, 7.9
Verificación cruzada certificado-BD	7.6.3
Componentes auxiliares	7.8 (capas), 7.10 (operación)

*Tabla A.2. Trazabilidad de los bloques del **MCR** con los apartados del cuerpo.*

ANEXO B. EVIDENCIA DE PRUEBAS Y VALIDACIÓN

Este anexo presenta la evidencia consolidada de que el sistema cumple sus requisitos y mitiga las amenazas identificadas. Se construye como una *checklist* trazable: cada entrada corresponde a un requisito funcional (RF), un escenario STRIDE (S) o un escenario de integración (I) del capítulo 8. La memoria conserva el material de síntesis (introducción, convenciones, tablas resumen y matriz de trazabilidad agregada); el detalle entrada a entrada (RF-01 a RF-22 y S-01 a S-15) se publica en el **MVP** (Manual de Validación y Pruebas), accesible en el repositorio público del proyecto. El alcance de la verificación es triple: aceptación funcional de los veintidós requisitos del apartado 7.2.1, validación de las quince mitigaciones STRIDE del apartado 8.1.5 e integración extremo a extremo de los doce escenarios de la Tabla 8.3 ejecutados como *smoke test* del sistema desplegado.

B.1 Convenciones de la checklist

Cada entrada del **MVP** sigue un formato común de seis campos: *cómo se valida*, *resultado esperado*, *resultado obtenido*, *veredicto* (`Cumple`, `Cumple parcialmente` o `No cumple`), *evidencia* (figura del Anexo C, salida de comando o consulta SQL) y *notas* opcionales. El veredicto `Cumple parcialmente` se aplica cuando el criterio principal se ha satisfecho, pero existe una limitación documentada en el apartado 8.4 o 9.3. La verificación con resultado formal disponible al cierre del proyecto corresponde al conjunto unitario de PKCE (29/29 PASS, vectores oficiales de RFC 7636 sección B.1, reproducibles con `php sso/tests/test_pkce.php`); el resto de bloques sigue el procedimiento documentado en el manual externo.

B.2 Resumen de bloques

La Tabla B.1 resume los doce escenarios de integración descritos en el apartado 8.1.3.

ID	Escenario	Evidencia
I-01	Login contraseña + TOTP	RF-02; Figuras C-02, C-03

ID	Escenario	Evidencia
I-02	Login con certificado FNMT	RF-03; Figura C-04
I-03	Flujo OAuth + PKCE	apartado 8.3.3; Figuras C-05, C-13
I-04	Rotación de *refresh token*	RF-09; salida `/token` y SQL `sso_sessions`
I-05	Cierre de sesión RP-Initiated	RF-06; `curl -sSI` 302 y SQL `sso_audit_log`
I-06	Solicitud de certificado *clientAuth*	apartado 8.3.4; Figura C-06
I-07	Aprobación y emisión	apartado 8.3.4; Figuras C-07, C-10
I-08	Revocación con regeneración inmediata de CRL	apartado 8.3.5; Figuras C-08, C-09
I-09	OCSP de certificado válido	RF-17; Figura C-11
I-10	OCSP de certificado revocado	RF-17; Figura C-11
I-11	*Rate limit* en `/login`	S-01; `curl` 429 y SQL `sso_audit_log`
I-12	Violación de *binding* IP/UA	S-02; SQL `sso_audit_log`

Tabla B.1. Escenarios de integración del smoke test*.*

La Tabla B.2 resume el plan de pruebas unitarias declarado en el apartado 8.1.2. El conjunto PKCE constituye la única verificación con resultado formal a fecha

de entrega; los seis conjuntos restantes se documentan en el **MVP** con el procedimiento de ejecución correspondiente.

Conjunto	Casos	Estado
PKCE (`sso/tests/test_pkce.php`)	29	29/29 PASS
Cifrado de *passphrase*	8	Implementación parcial
Firma y verificación JWT	6	Implementación parcial
Validación de `redirect_uri`	10	Implementación parcial
Validación de EKU y *policy*	6	Implementación parcial
TOTP	4	Implementación parcial
Cálculo de *thumbprint*	2	Implementación parcial

Tabla B.2. Conjuntos unitarios.

B.3 Matriz de trazabilidad agregada

La Tabla B.3 cruza cada bloque de requisitos y de escenarios STRIDE con el apartado del cuerpo donde se diseña la solución y con el **MVP** que aporta la evidencia entrada a entrada.

Bloque	Diseño en el cuerpo	Evidencia
RF-01 a RF-09 (SSO: registro, autenticación, sesión, OIDC)	apartados 7.5 y 7.6	**MVP** y Figuras C-01 a C-05, C-13
RF-10 a RF-15 (PKI: jerarquía, perfiles, emisión, revocación, panel)	apartados 7.4 y 7.10	**MVP** y Figuras C-06, C-07, C-10, C-16
RF-16 a RF-18 (OCSP)	apartado 7.7	apartado 8.3.6 y Figura C-11

Bloque	Diseño en el cuerpo	Evidencia
RF-19 a RF-22 (CRL y CA-repo)	apartado 7.7	apartado 8.3.7 y Figuras C-09, C-12
S-01 a S-15 (mitigaciones STRIDE)	apartados 7.8 y 7.9	**MVP** y Figuras C-14 a C-18

Tabla B.3. Matriz de trazabilidad agregada por bloque.

B.4 Síntesis cuantitativa

Bloque	Total	Estado
Requisitos funcionales (RF-01 a RF-22)	22	Pasada formal pendiente
Mitigaciones STRIDE (S-01 a S-15)	15	Pasada formal pendiente
Integración (I-01 a I-12)	12	Pasada formal pendiente
Unidad (PKCE)	29 vectores	Verificado, RFC 7636 sección B.1
Unidad (resto de conjuntos)	6 conjuntos	Implementación parcial

Tabla B.4. Síntesis cuantitativa del programa de pruebas.

El criterio de aceptación global exige al menos el 90 % de cumplimiento en requisitos funcionales, el 80 % en mitigaciones STRIDE y el 90 % en integración. Por debajo de cualquiera de esos umbrales, las entradas con veredicto `No cumple` se documentan en el apartado 9.3 y, si procede, se planifican como vía futura en el apartado 9.4. El sello de la última ejecución completa (`git rev-parse --short HEAD`) y la fecha asociada se publican en el **MVP**, lo que permite reproducir cualquier verificación posterior contra el mismo estado del código.

ANEXO C. EVIDENCIAS VISUALES Y SALIDAS DE VALIDACIÓN

Este anexo enumera, mediante una tabla índice, las evidencias visuales (capturas de pantalla y salidas de consola) que respaldan los flujos críticos y las pruebas más relevantes documentadas en el cuerpo. Las imágenes íntegras, los recortes en alta resolución y las salidas de consola completas se publican en el **MVP** (Manual de Validación y Pruebas), también accesible en el repositorio público del proyecto. El presente anexo aporta exclusivamente la trazabilidad cruzada con el Anexo B.

C.1 Índice de evidencias

La Tabla C.1 enumera las dieciocho figuras incluidas en el **MVP**, con su descripción breve, la entrada del Anexo B que respaldan y el subapartado del cuerpo del que parten.

Figura	Descripción	Respalda (Anexo B)	Apartado del cuerpo
C-01	Registro de usuario y TOTP activado	RF-01	7.5 / 7.6.1
C-02	Aplicación TOTP en dispositivo móvil	RF-02	7.6.1
C-03	Panel SSO tras inicio de sesión	RF-02	7.5
C-04	Diálogo de selección de certificado	RF-03	7.6.2
C-05	RP autenticado mediante OAuth con PKCE	RF-04	7.5.1, 8.3.3

Figura	Descripción	Respalda (Anexo B)	Apartado del cuerpo
C-06	Formulario de solicitud de certificado	I-06	7.4.3, 8.3.4
C-07	Aprobación y descarga de certificado	I-07	7.4.3, 8.3.4
C-08	Bucle OCSP en transición *good* a *revoked*	RF-13	7.4.5, 8.3.5
C-09	CRL antes y después de la revocación	RF-19, RF-20	7.8, 8.3.7
C-10	Verificación de cadena con `openssl verify`	RF-10, RF-11	7.4.1, 8.3.4
C-11	Tres respuestas OCSP: *good*, *revoked*, *unknown*	RF-17	7.7
C-12	CRL firmada y validada con `openssl crl -verify`	RF-21	7.8, 8.3.7
C-13	Conjunto unitario PKCE 29/29 PASS	RF-04	8.1.2, 8.3.3

Figura	Descripción	Respalda (Anexo B)	Apartado del cuerpo
C-14	Cabeceras de seguridad y `testssl.sh`	S-10	7.8, capa 8
C-15	Rechazo de `redirect_uri` malicioso	S-07	7.5.1
C-16	Acceso a `/admin/*` con rol `user` denegado	S-13	7.5.4
C-17	Acceso a claves privadas vía HTTP denegado	S-09	7.4.4
C-18	`sqlmap` declara los formularios no inyectables	S-03	7.8, capa 5

*Tabla C.1. Índice de evidencias visuales del **MVP**.*

Las restantes entradas del Anexo B se respaldan mediante consultas SQL, salidas de comandos `curl` o `openssl` y descripciones reproducibles documentadas en el propio Anexo B y, con mayor detalle, en el **MVP**. Las convenciones técnicas aplicadas a las capturas (resolución mínima de 1280 × 800 píxeles, anonimización de identificadores reales y formato PNG con nombrado `fig_C-NN_<slug>.png`) se especifican en la sección introductoria del manual externo.

ANEXO D. DECLARACIÓN DE USO DE INTELIGENCIA ARTIFICIAL

Este anexo deja constancia, de forma breve y trazable, del uso de herramientas de inteligencia artificial generativa durante el desarrollo del proyecto y la redacción de la memoria, en cumplimiento de las recomendaciones de la séptima edición de las normas APA sobre transparencia metodológica y de las directrices recientes de centros académicos sobre transparencia en el uso de IA.

Durante el trabajo se han empleado dos herramientas. La primera es Cursor, un editor de código integrado con asistencia de modelos de lenguaje (Anthropic Claude y OpenAI GPT), utilizado como capa adicional de revisión sobre fragmentos previamente escritos. La segunda es la versión web de ChatGPT y Claude, en sesiones aisladas y puntuales, para consultas terminológicas, validación de redacción y comprobación cruzada de referencias bibliográficas. No se han utilizado generadores de imagen, traductores automáticos ni servicios de generación de código de extremo a extremo.

El uso se ha mantenido acotado a tres ámbitos. El primero, la revisión de código previamente escrito, con foco en errores comunes, adherencia a buenas prácticas de seguridad (comparación segura, validación de entradas, manejo de excepciones) y consistencia interna entre módulos; los componentes críticos (motor PKI, *responder* OCSP, autenticación por certificado, gestión de JWT, gestor de sesiones y protección CSRF) se han razonado y escrito manualmente y solo se han pasado por la herramienta de revisión a posteriori. El segundo, apoyo a la redacción mediante pulido lingüístico, sugerencias de reformulación y detección ortográfica; el contenido técnico (decisiones criptográficas, justificaciones de seguridad y análisis del modelo de amenazas) se ha redactado manualmente. El tercero, verificación cruzada de citas (RFC, NIST, OWASP, ENISA y normativa UE) frente al contenido referenciado.

No se han delegado en IA las decisiones de arquitectura (separación de bases de datos, modelo de capas, integración SSO con PKI), las decisiones criptográficas (RSA-2048, AES-256-GCM, Argon2id, SHA-256, perfiles X.509 v3 y política de revocación), el análisis de amenazas STRIDE y el diseño de las mitigaciones, la implementación de los componentes críticos del motor PKI y del flujo OAuth 2.0 con PKCE, ni las decisiones de despliegue (régimen restringido

de la raíz, separación de subdominios, política de copias de seguridad). Cada sugerencia incorporada se verificó frente a la documentación oficial del estándar correspondiente, la ejecución y prueba del propio código o la consulta directa con la directora del trabajo. Las decisiones de seguridad documentadas en los capítulos 7, 8 y 9 corresponden a razonamientos propios sostenidos en los estándares citados en el capítulo 10.

El conjunto del trabajo (código, diseño, análisis de seguridad, validación experimental y redacción de la memoria) es de elaboración propia. Las herramientas de inteligencia artificial han actuado únicamente como apoyo a la revisión, sin sustituir el razonamiento técnico ni la justificación académica.