JBH LEGAL

# Documentación Técnica de la Plataforma

Arquitectura, infraestructura, seguridad, protección de datos y anonimización, motor de IA jurídica, superficie de API y operación del portal para abogados y clientes.

EDICIÓN PÚBLICA

**Organización**
JBH FINANCIAL GROUP, S.L. · CIF B06770143

**Fecha de emisión**
20 · 07 · 2026

**Alcance**
Monorepo completo: 8 aplicaciones · 18 paquetes · 140 tablas

**Plataforma**
jbhasesorialegal.com (CNAE 6910)

**Versión**
1.0 · edición pública (datos reservados marcados)

**Nota de seguridad**
Edición pública: credenciales e identificadores de infraestructura reservados

# Índice

# 1

# Visión general y arquitectura

> *JBH Legal es una plataforma SaaS multi-tenant de intermediación jurídica especializada en defensa penal, construida como un monorepo TypeScript (pnpm + Turborepo) con ocho aplicaciones Next.js/Node y diecisiete paquetes compartidos, desplegada sobre un VPS Plesk con MariaDB, Cloudflare R2 y la API de Anthropic como motor del «Letrado IA». Este capítulo describe la identidad del producto, el stack, el inventario completo de aplicaciones y paquetes, la arquitectura global de despliegue y los principios rectores que condicionan cualquier cambio en el sistema.*

## 1.1 Identidad del producto y modelo de negocio

JBH Legal (`jbhasesorialegal.com`) es una **plataforma de intermediación jurídica** (CNAE 6910), **no un despacho de abogados**. La plataforma coordina la prestación de servicios con abogados penalistas colegiados externos; esta distinción no es cosmética: condiciona todo el copy de marketing (el linter `marketing-banned.ts` prohíbe expresiones del tipo «nuestros abogados» o cualquier garantía de resultado), las páginas legales y la arquitectura de permisos. La sociedad titular es **JBH FINANCIAL GROUP, S.L.** (CIF B06770143), con Delegado de Protección de Datos externo designado (**VeraSafe**, `dpo@jbhasesorialegal.com`) y modelo 036 dado de alta en servicios jurídicos.

El producto se comercializa bajo dos modelos que conviven sobre el mismo código y la misma base de datos, diferenciados únicamente por el tenant:

| Modelo | Cliente | Tenant | Facturación |
|---|---|---|---|
| **Modelo B** | Despachos de abogados (SaaS) | Fila propia en `organizations`, con `organization_subscriptions` (Stripe), `memberships` y rol `LAWYER` | Suscripción por plan + consumo de IA por tramos de 100 € |
| **Modelo A** | Clientes directos (particulares) | Tenant paraguas de la plataforma (`JBH_ORG_ID`) | 500 €/caso + consumo de IA |

En ambos modelos rige la oferta comercial de **«primer caso gratis»** (sin tarjeta y sin límite de tiempo). El cobro del consumo de IA es *metered*: cada llamada al modelo queda registrada en `ai_request_log` con su coste crudo, y el importe facturable se calcula con `billableAiCents` aplicando un margen configurable por la variable de entorno `AI_BILLING_MARGIN`; de cara al cliente se presenta siempre como precio «por uso», nunca desglosado en coste más margen. Un fallo de cargo bloquea el acceso a las funciones de IA (gate en `packages/payments/src/ai-budget-gate.ts`).

El núcleo funcional es el **Letrado IA**: un arsenal de aproximadamente 45 generadores de documentos y análisis de defensa penal (escritos de defensa, recursos, calculadoras de pena y prescripción, detección de contradicciones, «Claves del caso», etc. — el directorio `packages/ai/src/prompts/` contiene 93 módulos de prompt), complementado por un producto de herencias/ISD, firma electrónica eIDAS in-house, CRM comercial, asistentes de voz (Telnyx AI Assistant), WhatsApp y email.

## 1.2 El monorepo: stack y tooling

El repositorio vive en la ruta local `/Users/lorenzoballantimoran/Documents/JBH Legal` (la carpeta **termina en un espacio**; toda referencia por shell debe ir entrecomillada) y en GitHub como `WayPool/jbh-asesoria-legal` (privado). La rama viva de producción es `phase-17-letrado-ia`.

El `package.json` raíz (`jbh-asesoria-legal@0.1.0`) fija el stack de tooling:

| Componente | Versión / configuración | Notas |
|---|---|---|
| Gestor de paquetes | `pnpm@10.26.0` (campo `packageManager`) | `engines`: Node ≥ 22, pnpm ≥ 10; workspaces en `pnpm-workspace.yaml` (`apps/*` + `packages/*`) |
| Orquestador de builds | `turbo ^2.3.3` | Tareas `build`, `dev`, `lint`, `typecheck`, `test`, `test:e2e`, `clean` |
| Lenguaje | TypeScript `^5.7.2` estricto | `tsconfig.base.json` con `noUncheckedIndexedAccess` |
| Framework web | Next.js 15 (App Router, output `standalone`) + React 19 | Tailwind v4, `next-intl`, Auth.js v5 |
| Persistencia | Drizzle ORM + `mysql2` sobre MariaDB | ~119 ficheros de schema en `packages/db/src/schema/`, 125 migraciones SQL journaled |
| Almacenamiento binario | Cloudflare R2 (API S3) | `packages/storage`; ver capítulo 3 |
| Pagos | Stripe | `packages/payments` (checkout, suscripciones, webhooks, presupuesto IA) |
| IA | `@anthropic-ai/sdk` (Claude) | `packages/ai`; selección de modelo por tier con `pickModel(scope, tier)` |
| Observabilidad | Sentry + Pino (+ Better Stack/Axiom) | `packages/logger`; `@sentry/nextjs` en cada app Next |

El `turbo.json` declara `globalDependencies` sobre `**/.env.*local` y `tsconfig.base.json` (cualquier cambio en ellos invalida la caché global). La tarea `build` depende de `^build` (los paquetes se construyen antes que las apps que los consumen), emite `.next/**` (excluyendo `.next/cache/**`) y `dist/**`, y

lista como entorno relevante `NODE_ENV`, `DATABASE_URL` y `AUTH_SECRET`. La tarea `test:e2e` depende de `build` y emite `playwright-report/**` y `test-results/**`.

ℹ **transpilePackages.** Las apps Next declaran los paquetes del workspace en `transpilePackages`, de modo que el código de `packages/*` se compila *dentro* del bundle de cada app (`.next/server`). Consecuencia operativa clave: un cambio en `packages/ai` (por ejemplo, un modelo) se despliega reconstruyendo y swapeando el `.next` de la app afectada, sin tocar `node_modules` en el servidor.

⚠ **Higiene del árbol.** El repo vive en iCloud Drive y arrastra artefactos de sincronización: existen duplicados con sufijo « `2.ts` » / « `3.ts` » (p. ej. `analyze-case 2.ts`) y carpetas `node_modules 2` o `_TRASH`. Son copias muertas de iCloud, **no** forman parte del build y deben ignorarse en cualquier análisis o refactor. El fichero canónico es siempre el que no lleva sufijo.

## 1.3 Inventario de aplicaciones (`apps/`)

El monorepo contiene ocho aplicaciones. Seis son frontales Next.js 15 standalone; una es una API Hono; otra es un worker de cola sin HTTP público. Cada app se ejecuta en producción como proceso pm2 propio escuchando en un puerto de loopback, detrás de Apache (Plesk) y Cloudflare (capítulo 2).

| App | Paquete | Framework | Puerto prod | Dominio | Propósito |
|---|---|---|---|---|---|
| `apps/web` | `@jbhlegal/web` | Next 15 standalone | 4103 | `jbhasesorialegal.com` (apex) | Web pública de captación: marketing, landings por área, enciclopedia penal, intake `/evaluar`, booking `/agendar`, triage IA público, webhooks Telnyx AI (`api/public/telnyx-ai/*`). Sin autenticación. |
| `apps/portal` | `@jbhlegal/portal` | Next 15 standalone | 4100 | `portal.jbhasesorialegal.com` | Aplicación del abogado y back-office: casos y causas, Letrado IA (escritos, análisis, Claves del caso), CRM (`/portal/crm`: pipeline kanban sobre `intakes`, presupuestos con aceptación pública por token), jurisprudencia, colaboradores, administración (analytics, DSR, usuarios, superadmin). Auth.js + 2FA TOTP. |
| `apps/cliente` | `@jbhlegal/cliente` | Next 15 standalone | 4101 | `cliente.jbhasesorialegal.com` | Área del cliente final (rol `CLIENT` + staff): estado del caso, documentos, mensajería, cuestionarios, reservas, videollamada (Jitsi), derechos RGPD (DSR). |
| `apps/firmas` | `@jbhlegal/firmas` | Next 15 standalone | 4102 | `firmas.jbhasesorialegal.com` | Vault y firma electrónica eIDAS por enlace tokenizado (JWT): el firmante accede sin cuenta, firma con nombre visible y se genera el PDF con evidencias (`packages/signing`). |
| `apps/herencias` | `@jbhlegal/herencias` | Next 15 standalone | 4104 | `herencias.jbhasesorialegal.com` | Producto sucesorio: consulta pública, calculadora ISD autonómica, inventario del caudal, informes sucesorios IA (`packages/ai/src/isd/`), modelos 650/655, área cliente y back-office. |
| `apps/peritos` | `@jbhlegal/peritos` | Next 15 standalone | — | `peritos.jbhasesorialegal.com` (previsto) | Marketplace/panel de peritos judiciales (encargos, pagos vía `perito-payment-checkout.ts`). La app existe y compila en el monorepo, pero **no tiene proceso en el ecosystem pm2 de producción** a fecha de este documento. |
| `apps/api` | `@jbhlegal/api` | Hono + `@hono/node-server` (tsup → `dist/server.js`) | 4000 (interno) | `api.jbhasesorialegal.com` | API auxiliar: webhook de WhatsApp Cloud API, lead-followup, health. Incluye un segundo entrypoint `email-worker.ts`: worker IMAP (`imapflow` + `mailparser`) que atiende los buzones soporte@/info@/despacho@ con el runner de `packages/assistant`. |
| `apps/ai-worker` | `@jbhlegal/ai-worker` | Node 22 (tsup → `dist/server.js`; health :9700) | — | — (sin exposición pública) | Consumidor de la cola `document_analysis_jobs`: extrae texto legible de PDF/Office (`pdfjs-dist`, `unpdf`, `mammoth`, `xlsx`), transcribe escaneados por visión, ejecuta análisis de documento/caso/causa/cuestionario/herencia con Claude y persiste resultados. Scripts auxiliares: `backfill-precedent-rag.ts`, `dedup-cleanup.ts`. |

ℹ **Puertos de desarrollo ≠ producción.** Los `package.json` de las apps definen puertos de `next dev` (web 3000, portal 3002, cliente 3098, firmas 3001…). En producción el puerto lo impone el bloque `env.PORT` del ecosystem pm2 (`infra/plesk/ecosystem.config.cjs`), que es la fuente de verdad de la tabla anterior. El puerto 3000 del servidor está ocupado por otra aplicación del VPS (OpoSignal), motivo por el que la web pública usa el 4103.

### 1.3.1 Superficie de API del portal

La app más grande es `apps/portal`. Sus grupos de rutas de API (`apps/portal/src/app/api/`) dan una radiografía funcional del producto:

| Grupo de rutas | Dominio funcional |
|---|---|
| `cases/`, `causas/` | Núcleo del Letrado IA: documentos (upload multipart, descarga, análisis), escritos por `kind`, cuestionarios, aclaraciones, hitos, tareas, reuniones/grabaciones, eventos judiciales, colaboradores por caso, pagos. |
| `intakes/`, `crm/`, `clientes/` | Captación y CRM: pipeline de leads (los intakes son el lead), ficha 360º, tareas comerciales, presupuestos con aceptación pública por token, mensajes IA (email/WhatsApp). |
| `jurisprudence/` | Biblioteca penal compartida: alta de resoluciones, ficheros PDF, búsqueda. |
| `admin/` | Back-office: analytics, leads, marketing, DSR, usuarios, verificaciones de colegiado, superadmin (monitor de actividad). |
| `registro-despacho/`, `registro-profesional/`, `aceptar-invitacion/`, `onboarding/` | Alta multi-tenant del Modelo B: registro del despacho, invitación y onboarding de clientes. |
| `stripe/`, `referral/`, `marketplace/`, `peritos/` | Pagos y ecosistema: webhooks Stripe, programa de colaboradores/referidos (`/r/{alias}`), marketplace de peritos. |
| `auth/`, `me/`, `users/`, `branding/`, `agenda/`, `bookings/`, `suggestions/`, `support/`, `health/`, `demo-login/` | Sesión, perfil, branding por despacho, agenda/reservas, soporte y salud. |

### 1.3.2 Modelo de roles

Los roles del sistema están tipados en `packages/auth/src/types.ts`: `ROLES = ['ADMIN', 'COORDINADOR', 'COLABORADOR', 'CLIENT', 'PERITO', 'ANONYMIZED', 'LAWYER']`. Las constantes derivadas definen qué rol entra en qué app: `PORTAL_ACCESS_ROLES` añade `LAWYER` (el abogado suscriptor del Modelo B) a los tres roles internos; `CLIENT_AREA_ROLES` admite en el área cliente a staff, peritos y clientes, pero cada ruta se limita además a los datos propios (`clientUserId === session.user.id`); `ANONYMIZED` es el estado terminal de una cuenta tras un DSR de supresión. La membresía de organización (tabla `memberships`) cruza usuario × tenant y es la que evalúan los gates de acceso (§1.7).

## 1.4 Inventario de paquetes (`packages/`)

Diecisiete paquetes de workspace encapsulan la lógica compartida. Todos se consumen como `workspace:*` y se transpilan dentro de cada app (§1.2).

| Paquete | Contenido | Consumidores principales |
|---|---|---|
| `ai` | Todo el motor Claude: cliente (`anthropic-client.ts` con `withModelFallback`), selección de modelo por tier (`models.ts`, `pickModel`), 93 prompts (`src/prompts/`), ~45 generadores `generate-*.ts` y analizadores `analyze-{case,causa,document,from-text}.ts`, triage, grounding de citas (`citation-grounding.ts`), coste (`cost.ts`, tarifa por familia never-throw), redacción de PII, módulo ISD (`isd/`), CENDOJ (`cendoj/`). | portal, web, cliente, herencias, peritos, api, ai-worker |
| `db` | **Fuente del modelo de datos**: schema Drizzle fichero-por-tabla (~119 módulos en `src/schema/`), cliente MariaDB (`client.ts`), scoping multi-tenant (`tenant.ts`), queries compartidas, 125 migraciones SQL. | Todas las apps y la mayoría de paquetes |
| `auth` | Auth.js v5: config, adapter Drizzle, magic-link, OTP, TOTP 2FA (`totp.ts`, `totp-crypto.ts`), roles (`roles.ts`, `types.ts`), guards de sesión, branding por despacho (`firm-brand.ts`), política MFA. | portal, cliente, herencias, peritos, web |
| `payments` | Stripe: `stripe-client.ts`, checkout de caso y consulta, suscripciones de organización (`org-subscriptions.ts`, `plans.ts`), webhook, reembolsos, facturación fiscal (`invoice-fiscal.ts`), presupuesto/gating de IA (`ai-budget.ts`, `ai-budget-gate.ts`), pagos a peritos. | portal, cliente, web, firmas, herencias, peritos |
| `storage` | Abstracción de almacenamiento: adapter R2 multipart (`r2-adapter.ts`), adapter filesystem, límites (500 MB, partes de 8 MB), allowlist de 15 extensiones/14 MIME, detección de MIME y ejecutables (`mime.ts`). Ver capítulo 3. | portal, cliente, web, firmas, herencias, peritos, ai-worker |
| `signing` | Firma eIDAS real: hash SHA-256 (`hash.ts`), JWT de firma (`jwt.ts`), plantillas PDF React (`template.tsx`, `contract-mercantil.tsx`, `declaracion-independencia.tsx`), hoja de evidencias (`evidence.tsx`), merge de PDF. | firmas, portal, cliente, herencias |
| `signatures` | **LEGACY** — esqueleto no importado por ninguna app; la firma real es `signing`. | — (ninguno) |
| `assistant` | Runner de asistentes conversacionales de email/WhatsApp (`runner.ts`, `actions.ts`, `leads.ts`). | api (webhook WhatsApp + email-worker) |
| `booking` | Reservas de consulta: pricing por tipo, generación de slots, tokens HMAC de feed ICS (`feed-token.ts`, `hmac.ts`). | web, portal, cliente |
| `google-calendar` | OAuth de Google, cifrado de tokens (`token-crypto.ts`, clave `GOOGLE_TOKEN_ENC_KEY`), free/busy, alta y listado de eventos. | web, portal, cliente |
| `colegiado` | Verificación de la colegiación del abogado: parsing de certificados (PEM/CMS), cadena de confianza y OIDs de colegios (`chain.ts`, `trust.ts`, `oids.ts`), revocación, policy engine. | portal (flujo de verificación asistida de colegiado) |
| `email` | Envío transaccional: SMTP (Nodemailer) con fallback Resend, reintentos, plantillas. | Todas las apps + ai-worker |
| `video` | Videollamadas Jitsi (generación de salas/JWT). | cliente, portal |

| Paquete | Contenido | Consumidores principales |
|---|---|---|
| `ui` | Design system compartido: componentes, tokens (`tokens.css`, `tokens.ts`), uploader multipart, notificaciones, exports PDF de markdown y preparación de juicio (`markdown-pdf.tsx`, `trial-prep-pdf.tsx`), facturas. | portal, cliente, firmas, herencias, peritos, web |
| `logger` | Pino preconfigurado, request-id, transporte. | Todas las apps |
| `compliance` | Esqueleto (Fase 1) reservado para hardening de cumplimiento; exporta solo `COMPLIANCE_PACKAGE_VERSION`. | — (placeholder) |
| `config` | Presets compartidos de ESLint y TypeScript (sin `src/`; directorios `eslint/` y `typescript/`). | Todas las apps y paquetes (devDependency) |

#### 1.4.1 Selección de modelo de IA por tier

Dentro de `packages/ai`, la política de modelos es «cada opción su campeón» y está centralizada en `models.ts`:

| Contexto | Modelo | Mecanismo |
|---|---|---|
| Flujos directos de pago / flagship | `claude-opus-4-8` (`FLAGSHIP_MODEL`) | `pickModel(scope, tier)` con `tier = 'paid'` |
| Cuentas gratuitas | Sonnet / Haiku según el scope | `pickModel(scope, tier)` con `tier = 'free'` |
| Razonamiento jurídico crítico (p. ej. «Claves del caso») | Configurable; env `BEST_LEGAL_REASONING_MODEL` | `bestLegalReasoningModel()` |
| Override operativo sin desplegar | Cualquiera | Variables `AI_*_MODEL` leídas en runtime |
| Caída de un modelo | Modelo similar de la misma familia | `fallbackChain` / `withModelFallback`, aplicado como wrapper de `createAnthropicClient` (incluido streaming) |

El cálculo de coste (`cost.ts`) tarifa por *familia* de modelo con política never-throw: un modelo desconocido no rompe la petición, se tarifa por su familia. Toda llamada pasa por los helpers `generateMarkdownDraft` / `generateJson` (salida estructurada validada con Zod); está prohibido instanciar el SDK de Anthropic a mano fuera de `anthropic-client.ts`.

## 1.5 Arquitectura global de despliegue

Toda la plataforma corre en un único VPS Plesk ([IP reservada]) detrás de Cloudflare. Apache termina TLS en el origen y actúa de proxy inverso hacia los procesos Node gestionados por pm2; los servicios de datos (MariaDB local, R2, Anthropic, Stripe, Telnyx, SMTP) se consumen desde los procesos Node. El detalle operativo se desarrolla en los capítulos 2 (servidor) y 3 (Cloudflare/R2).

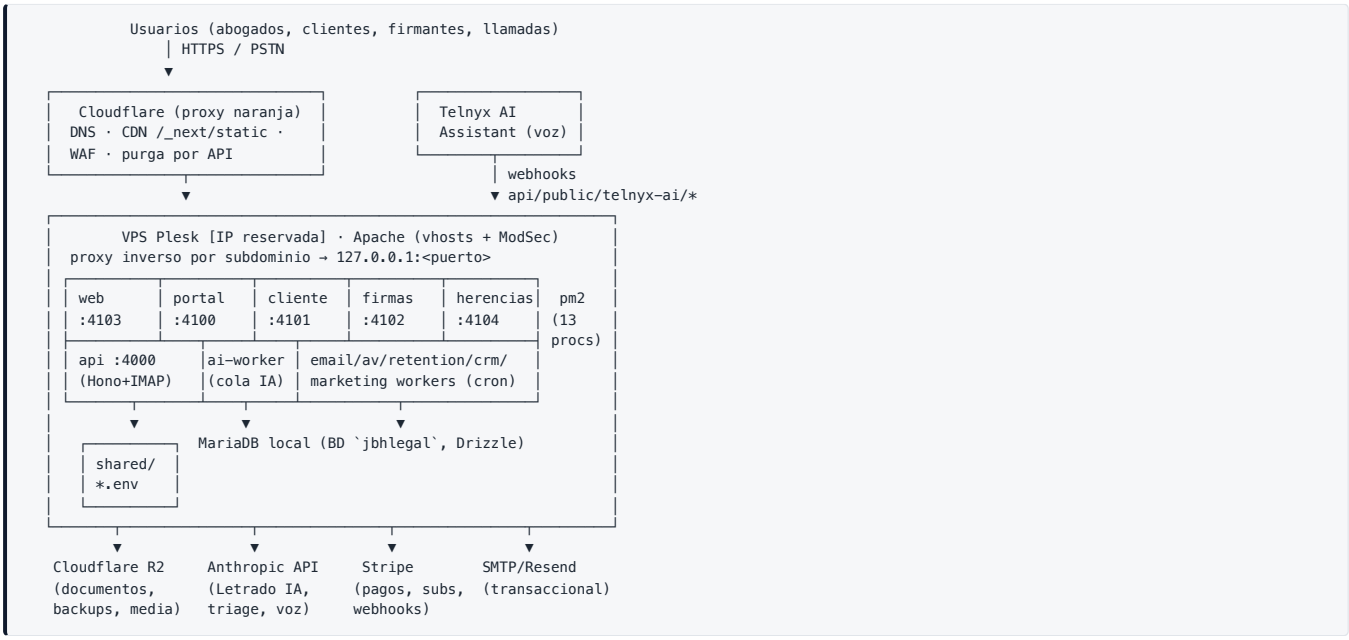


Fig. 1.1 — Arquitectura global: usuarios → Cloudflare → Apache/Plesk → procesos pm2 → servicios de datos.

El flujo de una petición típica al portal: el navegador resuelve `portal.jbhasesorialegal.com` contra Cloudflare (proxy activado); Cloudflare sirve de caché para `/_next/static` y reenvía el resto al origen; Apache aplica las reglas del vhost (incluida la neutralización del antibot ModSecurity, §3.3) y proxea a `127.0.0.1:4100`, donde el server standalone de Next (proceso pm2 `jbhlegal-portal`) ejecuta el App Router; las rutas de API acceden a MariaDB vía Drizzle, generan URLs prefirmadas de R2 para los binarios y encolan trabajos de IA que el `ai-worker` consumirá de forma asíncrona.

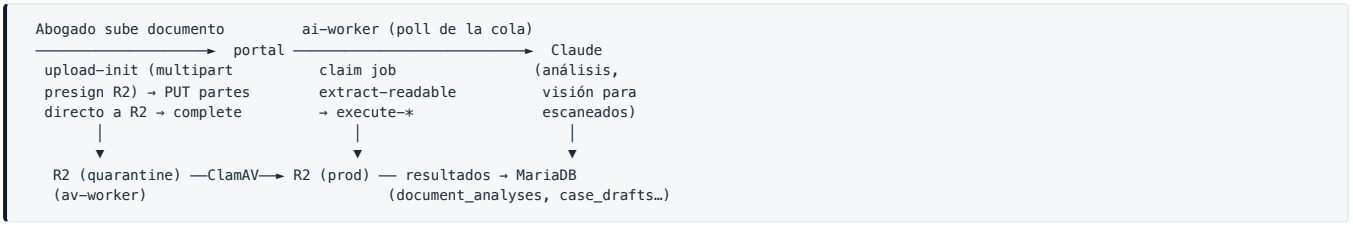


Fig. 1.2 — Flujo asíncrono documento → antivirus → análisis IA.

El `ai-worker` estructura su trabajo en módulos por tipo de job (`apps/ai-worker/src/`): `poller.ts` y `claim.ts` (reclamación atómica de jobs de `document_analysis_jobs`, con barrido de zombis `sweepStuckJobs`), `extract-readable.ts` (texto legible de PDF/Office, con transcripción por visión de escaneados), y los ejecutores `execute-per-doc.ts`, `execute-case.ts`, `execute-causa.ts`, `execute-questionnaire.ts`, `execute-answer-eval.ts`, `execute-meeting.ts` (grabaciones), `execute-trial-prep.ts`, `execute-herencia-{questionnaire,report}.ts` y la familia `*-precedent*` (ingesta e indexación RAG de jurisprudencia). Transversales: `budget.ts` (gate de presupuesto IA), `cost-tracking.ts`, `dedup-check.ts` y `health-server.ts` (endpoint de salud en el puerto 9700).

## 1.6 Principios rectores

Cinco reglas innegociables gobiernan el diseño y cualquier modificación del sistema. Están codificadas en el repo (linters, wrappers, gitignore) y no son negociables en revisiones de código.

### 1.6.1 Datos penales = categoría especial (art. 10 RGPD)

La plataforma trata datos relativos a condenas e infracciones penales, la categoría más sensible del RGPD. Consecuencias arquitectónicas: los expedientes, informes de caso, datos de cliente y secretos **jamás entran en git/GitHub** (el `.gitignore` cubre `EXPEDIENTES*/`, `DOCUMENTACIÓN/{SOCIEDAD,CONTRATOS}/` y `.env*`; hubo una fuga real purgada con `git filter-repo` y repetirla se considera incidente crítico); los documentos viven cifrados en R2 (UE, AES-256 en reposo) y no en la base de datos; existe un circuito DSR completo (`apps/portal/src/lib/dsr-{exporter,anonymizer,worker,cleanup-worker}.ts`) con retención de 10 años para documentos de caso y purga de intakes a 12 meses; y el proveedor de IA opera con Zero Data Retention (ZDR) acordado con Anthropic.

### 1.6.2 No garantizar resultados

Como plataforma de intermediación, ningún texto — marketing, UI, salidas de IA — puede prometer resultados judiciales. El linter `marketing-banned.ts` bloquea frases prohibidas en el pipeline de contenidos, y el registro de copy aprobado vive en `docs/marketing/frases-aprobadas.md`. Los generadores de IA incorporan la `LAWFUL_ASSISTANCE_NOTE`: asistencia lícita, sin fabricar prueba ni coartadas.

### 1.6.3 Citas legales verificadas o marcadas `[VERIFICAR]`

Un LLM puede inventar artículos, penas y jurisprudencia. El módulo `packages/ai/src/citation-grounding.ts` extrae y clasifica las citas legales (CP, LECrim, STS/ECLI…) de cada salida; toda cita que no pueda contrastarse contra la biblioteca de jurisprudencia o las fuentes oficiales se marca explícitamente `[VERIFICAR]` en el texto entregado al abogado. Ninguna herramienta del Letrado IA presenta una cita como verificada sin ese contraste.

### 1.6.4 Texto del usuario = datos, no instrucciones

Todo contenido procedente del cliente (relatos, respuestas de cuestionario, documentos transcritos, transcripciones de llamadas) es un vector de inyección de prompts. El paquete `ai` lo neutraliza envolviéndolo con `wrapUntrusted()` y anteponiendo la `UNTRUSTED_SYSTEM_NOTE` (ambos en `packages/ai/src/prompts/untrusted.ts`, aplicados desde `prompts/base-context.ts` y en los generadores y analizadores: `analyze-document.ts`, `analyze-from-text.ts`, `generate-refine-draft.ts`, etc.). Ningún generador nuevo puede interpolar texto de usuario sin este envoltorio.

### 1.6.5 Secretos out-of-band

Ningún valor secreto (API keys, contraseñas, `DATABASE_URL` con credenciales, claves privadas) vive en el repositorio, en los prompts ni en los logs. Los nombres de variable sí se documentan (p. ej. `ANTHROPIC_API_KEY`, `R2_SECRET_ACCESS_KEY`); sus valores residen únicamente en el Keychain de la máquina local del operador y en los ficheros `shared/*.env` del servidor (un fichero por app, cargados con `--env-file` nativo de Node 22 — ver §2.4), con backup nocturno cifrado asimétricamente (§3.7).

### 1.6.6 El modelo de datos a vista de pájaro

El esquema (`packages/db/src/schema/`, un fichero por tabla, IDs `varchar(36)` UUID) se organiza en grupos funcionales; conocerlos orienta cualquier búsqueda:

| Grupo | Tablas representativas |
|---|---|
| Auth | `users` (con `role`, `colegiacionNum`, secreto TOTP), `accounts`, `sessions` (con `mfaVerifiedAt`: 2FA por sesión, sliding 14 días) |
| Multi-tenancy | `organizations`, `organization_subscriptions`, `memberships`, `client_invitations` |
| Captación | `intakes` (+ notas/eventos/documentos), `leads`, `referral_*` (programa de colaboradores) |
| Casos | `cases`, `causas`, `case_documents` / `messages` / `milestones` / `tasks` / `meetings` / `court_events` / `collaborators` / `payments` / `questionnaires` / `drafts` / `clarifications` |

| Grupo | Tablas representativas |
|---|---|
| IA | `document_analyses`, `document_analysis_jobs` (la cola del ai-worker), `case_analyses`, `case_drafts` (escritos versionados por `kind`), `ai_request_log` (coste crudo por llamada) |
| Firmas | `signature_requests` (compartida penal+herencias vía `herenciaId` sin FK), `signatures` |
| Booking | `bookings`, `lawyer_calendars`, `lawyer_availability` |
| Herencias | `herencia_consultas` / `inventario` / `herederos` / `reports`, `ccaa_herencia_reglas` |
| Asistentes y RGPD | `whatsapp_links` (vinculación por OTP), `notifications`, `dsr_requests`, `content_pieces` (marketing), `stripe_webhook_events` |

## 1.7 Convenciones transversales

Además de los principios anteriores, tres convenciones estructurales aparecen en todo el código y conviene fijar desde el primer capítulo:

| Convención | Detalle | Riesgo si se ignora |
|---|---|---|
| Multi-tenancy por `org_id` | ~59 tablas llevan columna de organización; toda ruta bajo `cases/[id]` o `intakes/[id]` pasa por su gate de acceso (`requireCaseAccess` / `requireIntakeAccess`), que valida pertenencia y rol (el Modelo B admite `LAWYER`). | Añadir un rol a una ruta sin pasar por el gate = IDOR entre despachos. |
| IDs y colación | Claves primarias `varchar(36)` UUID. La tabla histórica `cases.id` es `utf8_general_ci` (utf8mb3): toda FK nueva contra ella debe igualar la colación o la migración falla con errno 150. | Migraciones rotas en producción. |
| Escrituras seguras | Numeración por `MAX+1` (nunca `COUNT+1`), validaciones Zod `.nonnegative()` en vez de `.positive()` donde el 0 es legal, y `db.transaction` en toda escritura multi-paso. | Colisiones de numeración y estados a medias. |

## 1.8 Ciclo de desarrollo

Comandos canónicos del día a día (siempre vía pnpm con filtro de workspace, nunca `npx` — que instalaría versiones fuera del lockfile):

```
pnpm install                                    # respeta pnpm-lock.yaml
pnpm --filter @jbhlegal/portal exec tsc --noEmit    # typecheck de una app
pnpm --filter @jbhlegal/portal exec vitest run      # tests unitarios
TURBO_CONCURRENCY=1 pnpm --filter @jbhlegal/portal build   # build de producción
pnpm --filter @jbhlegal/db generate                 # nueva migración Drizzle
```

Dos advertencias de baseline: existen tests **pre-rotos conocidos** por entorno/mocks (especialmente en `packages/ai`); antes de atribuirse una regresión, comparar contra la baseline con `git stash`. Y el árbol de producción va por delante de `main` (rama `phase-17-letrado-ia` con trabajo sin commitear): no se hace push a `main` ni force-push sin entender la divergencia — la rama remota fue reescrita con `filter-repo` tras la purga de datos (§1.6.1).

> ✓ **Verificación doble.** Regla operativa de todo el proyecto: cada afirmación publicada (cita legal, cifra, contenido desplegado) se verifica dos veces — contra la fuente (BOE/CENDOJ, código real) antes de escribirla y contra el resultado en vivo después de desplegarla. Esta documentación sigue el mismo criterio: todos los nombres de fichero, procesos, puertos y variables citados proceden del árbol real del repo y de la configuración de producción.

# 2

# Infraestructura de servidores

> *Toda la plataforma JBH Legal corre en un único VPS gestionado con Plesk, bajo el usuario de la suscripción del dominio* `jbhasesorialegal.com`*. Este capítulo documenta el acceso al servidor, la topología completa de los 13 procesos pm2, el mapa de dominios y puertos, la estructura del vhost en disco, el crontab del usuario, el procedimiento real de despliegue (swap del* `.next` *de Next standalone) y la caché de build local en iCloud con sus gotchas.*

## 2.1 El VPS y el acceso

Parámetros del servidor de producción:

| Parámetro | Valor |
|---|---|
| IP pública | `[IP reservada]` |
| Panel | Plesk (Apache como servidor web; el dominio no expone directivas nginx) |
| SSH | Puerto `[puerto reservado]`, usuario del vhost `[usuario del sistema reservado]` |
| Autenticación SSH | Clave ed25519 dedicada (`~/.ssh/[clave SSH reservada]` en la máquina del operador; valor out-of-band). La autenticación por contraseña está restringida por IP; la autenticación por clave no lo está. |
| Node.js | Node 22 provisto por Plesk en `/opt/plesk/node/22/bin` (binario `/opt/plesk/node/22/bin/node`) |
| pm2 | `~/.npm-global/bin/pm2` (instalación de usuario, no global del sistema) |
| Base de datos | MariaDB local (`localhost:3306`), BD única `jbhlegal`; credenciales en `DATABASE_URL` dentro de `shared/portal.env` (out-of-band) |
| `$HOME` del vhost | `/var/www/vhosts/jbhasesorialegal.com/` |

```
ssh -i ~/.ssh/[clave SSH reservada] -p [puerto reservado] [usuario del sistema reservado]@[IP reservada]
# En el servidor, para cualquier comando node/pm2:
export PATH=/opt/plesk/node/22/bin:$PATH
~/.npm-global/bin/pm2 ls
```

> ⚠ **El scheduler de Plesk está «enjaulado».** Existe un canal alternativo de ejecución vía API REST de Plesk (`scheduler`), útil cuando la IP del operador no puede abrir SSH. Ese canal **sí escribe ficheros** en el vhost, pero `pm2 restart`, `curl 127.0.0.1:puerto` y `ss` lanzados desde él **no ven ni tocan los procesos reales**: los reinicios son no-ops silenciosos y las verificaciones locales dan falsos negativos. Regla operativa: el transporte de artefactos puede ir por cualquier canal, pero el `pm2 restart` y la verificación final se hacen por SSH real o contra la URL pública externa. Además, `scheduler --run` no devuelve el stdout del comando (solo «Successfully completed» / «Completed with error»), por lo que el resultado debe codificarse en el exit code o en un log escrito a disco.

## 2.2 Topología pm2: los 13 procesos

La fuente de verdad de la topología es `infra/plesk/ecosystem.config.cjs` (en el servidor, `~/ecosystem.config.cjs`). Todos los procesos corren en `exec_mode: 'fork'` con `instances: 1`, cargan su entorno con el flag nativo de Node 22 `--env-file=/var/www/vhosts/jbhasesorialegal.com/shared/<app>.env` (más fiable que el `env_file` de pm2) y escriben logs con timestamp a `~/logs/<nombre>-{out,error}.log`.

| Proceso | cwd (bajo `$HOME`) | Script | Puerto | Modo | Env file | Mem. límite |
|---|---|---|---|---|---|---|
| `jbhlegal-api` | `api/` | `server.js` | 4000 (interno) | Always-on | `api.env` | 512M |
| `jbhlegal-portal` | `portal/` | `apps/portal/server.js` | 4100 | Always-on | `portal.env` | 512M |
| `jbhlegal-cliente` | `cliente/` | `apps/cliente/server.js` | 4101 | Always-on | `cliente.env` | 512M |
| `jbhlegal-firmas` | `firmas/` | `apps/firmas/server.js` | 4102 | Always-on | `firmas.env` | 512M |
| `jbhlegal-herencias` | `herencias/` | `apps/herencias/server.js` | 4104 | Always-on | `herencias.env` | 512M |
| `jbhlegal-web` | `web/` | `apps/web/server.js` | 4103 | Always-on | `web.env` | 512M |
| `jbhlegal-email-worker` | `api/` | `email-worker.js` | — | Always-on (IMAP) | `email-worker.env` | 512M |
| `jbhlegal-av-worker` | `workers/` | `av-worker-entry.cjs` | — | Always-on (escáner) | `portal.env` | 256M |

| Proceso | cwd (bajo `$HOME`) | Script | Puerto | Modo | Env file | Mem. límite |
|---|---|---|---|---|---|---|
| `jbhlegal-retention-worker` | `workers/` | `retention-worker-entry.cjs` | — | **Cron-run** `0 2 * * *` (02:00 UTC) | `portal.env` | — |
| `jbhlegal-crm-automations` | `workers/` | `crm-automations-entry.cjs` | — | **Cron-run** `15 * * * *` (cada hora a :15) | `portal.env` | — |
| `jbhlegal-ai-worker` | `ai-worker/` | `dist/server.js` | 9700 (health) | Always-on (cola) | `ai-worker.env` | 512M |
| `jbhlegal-marketing-worker` | `workers/` | `marketing-worker-entry.cjs` | — | **Cron-run** `0 8 * * *` (08:00 UTC) | `portal.env` | 512M |
| `jbhlegal-marketing-watchdog` | `workers/` | `marketing-watchdog-entry.cjs` | — | **Cron-run** `0 13 * * *` (13:00 UTC) | `portal.env` | 256M |

Funciones de cada worker no-HTTP:

| Worker | Qué hace |
|---|---|
| `jbhlegal-email-worker` | Worker IMAP (entrypoint `email-worker.ts` de `apps/api`): monitoriza los buzones soporte@/info@/despacho@ y responde con el runner de `packages/assistant`. |
| `jbhlegal-av-worker` | Escáner antivirus de larga duración: consume el bucket R2 de cuarentena y pasa cada documento por ClamAV antes de promoverlo al bucket de producción (§3.6). |
| `jbhlegal-retention-worker` | Cron diario de retención RGPD: aplica la política de 10 años a documentos de caso y la purga a 12 meses de intakes. |
| `jbhlegal-crm-automations` | Automatizaciones comerciales del CRM (E4): SLA de respuesta 24 h, seguimiento de presupuesto a las 72 h, reactivación de lead frío a los 14 días. Corre una vez por hora y sale. |
| `jbhlegal-ai-worker` | Cola de análisis IA (capítulo 1, §1.3): poll de `document_analysis_jobs`, extracción de texto, análisis con Claude, persistencia de resultados. |
| `jbhlegal-marketing-worker` | Ejecución diaria del marketing IA (blog + social); idempotente por día UTC (si ya corrió hoy, sale sin hacer nada). |
| `jbhlegal-marketing-watchdog` | Vigía: cinco horas después del slot de marketing, alerta por email si la corrida diaria no ocurrió (worker muerto o env ausente). |

ℹ **«stopped» entre ejecuciones es NORMAL.** Los cuatro procesos cron-run (`retention-worker`, `crm-automations`, `marketing-worker`, `marketing-watchdog`) están configurados con `cron_restart + autorestart: false`: pm2 los arranca a la hora programada, hacen su trabajo, terminan y quedan en estado `stopped` hasta la siguiente ejecución. Ver estos procesos en `stopped` en `pm2 ls` **no es una incidencia** y no deben «arreglarse» arrancándolos a mano fuera de horario.

### 2.2.1 Concurrencia del ai-worker

El proceso `jbhlegal-ai-worker` tiene una aritmética de concurrencia propia que hay que respetar: el número real de llamadas de visión simultáneas contra la API de Anthropic es el producto `WORKER_CONCURRENCY × PDF_TRANSCRIBE_CONCURRENCY`. La configuración aplicada en producción tras el incidente de saturación (una config 4×4 = 16 llamadas concurrentes provocaba timeouts sistemáticos en PDFs escaneados grandes) es:

| Variable (en `shared/ai-worker.env`) | Valor | Significado |
|---|---|---|
| `WORKER_CONCURRENCY` | 3 | Jobs de la cola procesados en paralelo (lotes de `Promise.all`) |
| `PAGES_PER_CHUNK` | 3 | Páginas por chunk en la transcripción por visión |
| `PDF_TRANSCRIBE_CONCURRENCY` | 2 | Chunks transcritos en paralelo por documento |

Diagnóstico de un documento «no se pudo analizar»: consultar el job en `document_analysis_jobs` y el log de salida real del proceso (`pm2 jlist` → campo `pm_out_log_path`); un error R2 «specified key does not exist» indica que el objeto no llegó a promoverse desde cuarentena.

⚠ **Los workers de `~/workers/` son bundles autocontenidos.** Se generan con `tsup` desde `apps/portal/tsup.workers.config.ts` (entradas en `apps/portal/scripts/*-entry.ts`, salida `apps/portal/dist/workers/*.js`, formato CJS, target node22, `noExternal: [/.*/]` — se bundlea absolutamente todo, incluidas dependencias del workspace y `@sentry/node`) precisamente para que en producción no necesiten `node_modules` al lado. Desplegar un worker = copiar su `.cjs` a `~/workers/` y reiniciar su proceso.

### 2.2.2 Operación pm2 cotidiana

Referencia rápida de operación (todos los comandos con `PATH=/opt/plesk/node/22/bin:$PATH` y el binario `~/.npm-global/bin/pm2`):

| Comando | Uso | Precaución |
|---|---|---|
| `pm2 ls` | Estado general | Los 4 cron-run en `stopped` fuera de horario = normal |
| `pm2 jlist` | JSON con `pm_exec_path`, `pm_cwd`, `pm_out_log_path` | Fuente de verdad para localizar el `.next` real y los logs reales de un proceso (no asumir rutas) |

| Comando | Uso | Precaución |
|---|---|---|
| `pm2 restart <proc>` | Tras un swap de build | Ejecutar por SSH real (por scheduler es un no-op, §2.1); prohibido en `jbhlegal-ai-worker` con análisis en vuelo |
| `pm2 reload <proc> --update-env` | Recarga con re-lectura del `--env-file` | Para cambios solo de entorno en `shared/*.env` |
| `pm2 save` | Persistir topología | Obligatorio tras cualquier restart/cambio (§2.5) |
| `pm2 logs <proc>` / `tail ~/logs/*.log` | Diagnóstico | Los logs llevan timestamp (`time: true` en el ecosystem) |

## 2.3 Dominios, subdominios y puertos

Apache (Plesk) enruta cada (sub)dominio a su puerto de loopback. Todos los subdominios pasan por Cloudflare con proxy activado (capítulo 3).

| Dominio | App / proceso | Origen | Notas |
|---|---|---|---|
| `jbhasesorialegal.com` (apex) | `apps/web` · `jbhlegal-web` | `127.0.0.1:4103` | Web pública. `/_next/static` servido por Apache desde disco con `Alias` + `ProxyPass !` (fix del doble URL-decode de `%5B`, ver §3.2). |
| `portal.jbhasesorialegal.com` | `apps/portal` · `jbhlegal-portal` | `127.0.0.1:4100` | Portal del abogado + CRM + admin. `robots.txt Disallow: /` + `X-Robots-Tag: noindex`. |
| `cliente.jbhasesorialegal.com` | `apps/cliente` · `jbhlegal-cliente` | `127.0.0.1:4101` | Área cliente. Mismo tratamiento noindex. |
| `firmas.jbhasesorialegal.com` | `apps/firmas` · `jbhlegal-firmas` | `127.0.0.1:4102` | Firma por token; sin auth de sesión. |
| `herencias.jbhasesorialegal.com` | `apps/herencias` · `jbhlegal-herencias` | `127.0.0.1:4104` | Producto sucesorio. |
| `api.jbhasesorialegal.com` | `apps/api` · `jbhlegal-api` | `127.0.0.1:4000` | Webhooks WhatsApp, lead-followup, health. Con HSTS. |
| `reels.jbhasesorialegal.com` | JBH Reels Studio | `127.0.0.1:4200` | **Repo independiente** (`~/Documents/jbh-reels-studio`, Next 15 + Prisma/SQLite); comparte vhost y pm2 pero no pertenece a este monorepo. |
| `peritos.jbhasesorialegal.com` | `apps/peritos` | — | Previsto; la app compila en el monorepo pero no tiene proceso pm2 en producción. |

### 2.3.1 Proxy inverso Apache

La plantilla de referencia del proxy está versionada en `infra/plesk/apache-additional.conf` y se aplica por vhost en Plesk («Additional directives for HTTPS»). Patrón por dominio:

```
ProxyPreserveHost On
ProxyTimeout 60
ProxyRequests Off

ProxyPass /.well-known/ !                    # ACME/verificaciones se sirven de disco
ProxyPass / http://127.0.0.1:<puerto>/
ProxyPassReverse / http://127.0.0.1:<puerto>/
RequestHeader set X-Forwarded-Proto "https"
RequestHeader set X-Forwarded-Port "443"
```

Excepciones al proxy añadidas con el tiempo: `/_next/static` del apex se sirve desde disco con `Alias` (fix del doble URL-decode, §3.2), y `robots.txt` de portal/cliente se sirve estático con `ProxyPass /robots.txt !` (§3.3). Las cabeceras `X-Forwarded-*` son necesarias para que Auth.js y las URLs absolutas de Next generen `https://` correctamente detrás del doble proxy Cloudflare→Apache.

## 2.4 Estructura del vhost en disco

Cada app se despliega como bundle Next *standalone* independiente bajo el `$HOME` del vhost. La estructura relevante:

```
/var/www/vhosts/jbhasesorialegal.com/
├── ecosystem.config.cjs          # topología pm2 (copia de infra/plesk/)
├── httpdocs/                     # docroot Plesk del apex (todo proxea a Node)
├── web/       └── apps/web/{server.js, .next/, public/}  + node_modules + packages
├── portal/    └── apps/portal/{server.js, .next/, public/}   ← swap de deploy AQUÍ
├── cliente/   └── apps/cliente/{server.js, .next/, public/}
├── firmas/    └── apps/firmas/{server.js, .next/, public/}
├── herencias/ └── apps/herencias/{server.js, .next/, public/}
├── api/       ├── server.js  └── email-worker.js      (bundles tsup de apps/api)
├── ai-worker/ └── dist/server.js                      (bundle tsup de apps/ai-worker)
├── workers/   ├── av-worker-entry.cjs                 (bundles §2.2)
│              ├── retention-worker-entry.cjs
│              ├── crm-automations-entry.cjs
│              ├── marketing-worker-entry.cjs
│              ├── marketing-watchdog-entry.cjs
│              ├── jbh-nightly-backup.sh               (backup nocturno, §3.7)
│              └── jbh-r2.mjs                          (cliente R2 SigV4 solo-Node)
├── shared/    ├── api.env · portal.env · cliente.env · firmas.env
│              ├── herencias.env · web.env · email-worker.env · ai-worker.env …
│              └── jbh-dr-cert.pem                     (cert. público de cifrado DR)
└── logs/      └── <proceso>-{out,error}.log           (pm2, con timestamps)
```

Fig. 2.1 — Layout del vhost. Cada app es un bundle standalone autónomo; los secretos viven exclusivamente en `shared/*.env`.

Puntos clave del layout:

- `shared/*.env`**: un fichero de entorno por app** (10 en total), con permisos restringidos al usuario del vhost. Son la única residencia en el servidor de secretos como `DATABASE_URL`, `ANTHROPIC_API_KEY`, `R2_*`, claves Stripe o SMTP. Se cargan por proceso con `--env-file` (§2.2), nunca se copian al repo, y se respaldan cada noche cifrados asimétricamente (§3.7).
- **Variables compartidas entre envs**: algunas claves deben ser idénticas en varios ficheros — `GOOGLE_TOKEN_ENC_KEY` debe coincidir en `web.env` y `portal.env` (el cifrado de tokens OAuth es simétrico entre apps), y `ANTHROPIC_API_KEY` está presente en `web.env` (triage público), `portal.env` (escritos), `herencias.env` (extracción) y `ai-worker.env` (análisis); su ausencia en cualquiera de ellos se manifiesta como 502 o análisis colgados, no como un error claro.
- **No hay** `.git` **en el servidor**: producción son bundles construidos en local y transportados; el árbol de producción corresponde a un working tree local *sin commitear* de la rama `phase-17-letrado-ia` (desplegar desde un HEAD limpio revertiría features vivas).
- `httpdocs/` **no sirve contenido**: todo el tráfico del apex proxea al proceso Node; intentar leer ficheros arbitrarios por HTTP devuelve el 404 de la app.

## 2.5 Crontab del usuario del vhost

Además de los cron-run gestionados por pm2 (§2.2) y de la tarea de backup del scheduler de Plesk (id 1038, 02:30 UTC — §3.7), el crontab del usuario del vhost contiene:

| Programación | Tarea | Función |
|---|---|---|
| `0 9 * * *` (09:00) | voice-reminders | Recordatorios de citas/consultas por el canal de voz. |
| `0 10 * * *` y `30 10 * * *` | lead-followup | Seguimiento automático de leads (endpoint de `apps/api`), dos pasadas matinales. |
| `@reboot` | `pm2 resurrect` | Restaura los 13 procesos tras un reinicio del servidor (con el `pm2 save` previo como fuente; ver §2.6, paso 6). |

> ⚠ `pm2 save` **es parte del deploy.** Si un deploy reinicia procesos y no ejecuta `pm2 save`, el `pm2 resurrect` del próximo reboot restaurará una topología obsoleta. La ausencia de `jbhlegal-web` en un `save` antiguo dejó el sitio público en 503 tras un reinicio (incidencia 2026-06-16), motivo por el que la web se añadió explícitamente al ecosystem.

## 2.6 Procedimiento real de despliegue (swap del `.next`)

El pipeline de GitHub Actions (`deploy-production.yml` sobre push a `main`) está abandonado: producción va por delante de `main` y corre desde un árbol sin commitear. El procedimiento vigente y verificado es el **swap del** `.next` por SSH real, con backup y verificación en vivo. Se describe para `apps/portal`; es idéntico para web/cliente cambiando puerto y directorio.

### 2.6.1 Pasos

```
# 1 · Build local (Node 22; concurrencia 1 por OOM; caché .next limpia antes — §2.7)
rm -rf apps/portal/.next/*
TURBO_CONCURRENCY=1 pnpm --filter @jbhlegal/portal build

# 2 · Staging: server standalone SIN source-maps ni caché + static + public
#     (el build son ~133 MB; sin *.map baja a ~48 MB y el tar.gz a ~10 MB)
rsync -a --exclude='*.map' --exclude='cache' \
  apps/portal/.next/standalone/apps/portal/.next/  stage/.next/
rsync -a apps/portal/.next/static/  stage/.next/static/   # Next standalone NO copia static
rsync -a apps/portal/public/        stage/public/

# 3 · Empaquetado sin metadatos xattr de macOS
COPYFILE_DISABLE=1 tar czf portal-swap.tgz -C stage .

# 4 · Transporte
scp -i ~/.ssh/[clave SSH reservada] -P [puerto reservado] portal-swap.tgz \
  [usuario del sistema reservado]@[IP reservada]:/tmp/

# 5 · Swap en el servidor, con backup fechado
PDIR=/var/www/vhosts/jbhasesorialegal.com/portal/apps/portal
mv $PDIR/.next $PDIR/.next.bak-$(date +%Y%m%d-%H%M%S)
mkdir -p $PDIR/.next && tar xzf /tmp/portal-swap.tgz -C $PDIR --strip-components=0
# (extraer .next/ y public/ del tar sobre $PDIR)

# 6 · Reinicio REAL + persistencia de topología
export PATH=/opt/plesk/node/22/bin:$PATH
~/.npm-global/bin/pm2 restart jbhlegal-portal && ~/.npm-global/bin/pm2 save

# 7 · Verificación por BUILD_ID (la única fiable) ANTES de tocar la CDN
BID=$(cat $PDIR/.next/BUILD_ID)
curl -s -o /dev/null -w '%{http_code}' \
  http://127.0.0.1:4100/_next/static/$BID/_buildManifest.js    # debe dar 200

# 8 · Purga de la CDN (capítulo 3) y verificación en vivo por la URL pública
```

Retención de backups: se conservan los **dos últimos** `.next.bak-*` y se podan los anteriores. Rollback = `mv` del backup de vuelta + `pm2 restart`.

### 2.6.2 Reglas y gotchas del deploy

⚠ **El swap va en `dirname(pm_exec_path)`, NO en `pm_cwd`.** `pm2 jlist` reporta para `jbhlegal-portal` un `pm_cwd` en la raíz (`…/portal`) pero el proceso ejecuta `pm_exec_path = …/portal/apps/portal/server.js`: el `.next` que el server usa es `…/portal/apps/portal/.next`. Swapear en la raíz deja el proceso con el build viejo mientras el 307 de la home «parece OK»; el manifest del BUILD_ID nuevo daría 404. Verificar siempre el paso 7 antes de purgar la CDN: un 401 de una ruta API *no* prueba que el build exista (el middleware devuelve 401 a cualquier `/api/**` sin sesión).

⚠ `.next/static` **viaja DENTRO del** `.next`**.** El output standalone de Next no copia `static`; si no se añade a mano (paso 2), los chunks nuevos devuelven 404 en origen y los viejos aparentan funcionar solo porque Cloudflare aún los sirve de caché (`cf-cache-status: HIT` con `age` de días) — un fallo particularmente traicionero.

⚠ **NUNCA reiniciar `jbhlegal-ai-worker` con un análisis en vuelo.** Un SIGINT/restart/deploy aborta el análisis en curso y lo reencola desde cero (en documentos grandes eso son ~15 minutos perdidos, a veces horas si se repite). Antes de reiniciarlo: comprobar en BD que la cola está ociosa y el latido de `sweepStuckJobs`. El indicador `lastPollAt: null` o «stale» es a nivel de LOTE (la salud solo se actualiza al terminar el `Promise.all` de 3 jobs) y un worker a 0 % CPU durante minutos suele estar esperando red del modelo, no colgado. Reclamar jobs zombis es una operación solo-BD que no requiere reiniciar.

✓ **Verificación en dos niveles.** El deploy no se da por bueno hasta que (a) `_buildManifest.js` del BUILD_ID nuevo responde 200 en el origen (`127.0.0.1:<puerto>`) y (b) tras la purga, la URL pública sirve el build nuevo (las rutas dinámicas son `cf-cache-status: DYNAMIC`, por lo que la verificación funcional puede hacerse directamente en público con cache-buster).

### 2.6.3 Despliegue de workers y del ai-worker

Los procesos no-Next tienen su propio artefacto:

- **Workers de** `~/workers/`: `pnpm --filter @jbhlegal/portal build:workers` (tsup, §2.2) → copiar el `.cjs` resultante de `apps/portal/dist/workers/` a `~/workers/` → reiniciar el proceso pm2 correspondiente (para los cron-run basta con que el fichero nuevo esté en su sitio antes de la siguiente ejecución programada).
- `jbhlegal-ai-worker`: `pnpm --filter @jbhlegal/ai-worker build` (tsup → `dist/`) → reemplazar el contenido de `~/ai-worker/dist/` → reinicio *condicionado*: verificar primero que la cola está ociosa (§2.6.2). El bundle es autocontenido; no hay `node_modules` que sincronizar.
- `jbhlegal-api` / `jbhlegal-email-worker`: bundles tsup de `apps/api` (`server.js` y `email-worker.js`) desplegados a `~/api/`.

### 2.6.4 Migraciones de base de datos

**El deploy normal de las apps no ejecuta migraciones.** Los cambios de esquema se aplican como paso separado y explícito contra la MariaDB del servidor: un *ensure-script* idempotente (SQL `CREATE TABLE IF NOT EXISTS` / `ALTER` comprobando `information_schema`) ejecutado con el cliente `mysql` del servidor, leyendo las credenciales de `DATABASE_URL` (parseada con un one-liner de Node desde `shared/portal.env`) hacia un fichero `.cnf` temporal. Reglas: backup on-demand previo (§3.7), colación explícita en toda FK contra tablas históricas (`cases.id` es `utf8_general_ci`; una FK con colación distinta falla con errno

150), y verificación posterior con `SHOW CREATE TABLE`. El journal Drizzle del repo (125 migraciones en `packages/db`) sigue siendo la fuente de verdad del esquema; el ensure-script es el vehículo de aplicación en producción.

### 2.6.5 Checklist de deploy

| # | Comprobación | Cómo |
|---|---|---|
| 1 | Dominio/app destino correctos | Regla de seguridad de despliegue: confirmar vhost y directorio antes de subir nada; nunca reemplazar un proyecto por otro. |
| 2 | Symlinks de build sanos | `ls -la apps/<app> \| grep .next` → debe ser `lrwx` (§2.7). |
| 3 | Caché `.next` limpia | `rm -rf apps/<app>/.next/*` antes del build. |
| 4 | Cola IA ociosa (si se toca el ai-worker) | BD + latido `sweepStuckJobs`; jamás reiniciar con análisis en vuelo. |
| 5 | Backup del `.next` anterior | `.next.bak-<TS>`; retener 2. |
| 6 | Build vivo | `_buildManifest.js` del BUILD_ID nuevo = 200 en origen. |
| 7 | `pm2 save` | Tras cada restart (§2.5). |
| 8 | Verificación pública post-purga | URL pública con cache-buster; funcionalidad cambiada probada en vivo. |

## 2.7 Caché de build local (iCloud) y sus gotchas

El repo local vive en iCloud Drive, que corrompe y sincroniza masivamente los artefactos de build. Para evitarlo, `infra/scripts/icloud-unsync-build.sh` redirige todo lo regenerable — `node_modules` (raíz y de cada workspace), `.next`, `.turbo`, `dist` — a **symlinks** que apuntan a `~/Library/Caches/jbh-legal-build/<hash-del-path>/`, fuera de iCloud. iCloud solo ve symlinks minúsculos.

| Síntoma | Causa | Remedio |
|---|---|---|
| `next: command not found`, `node_modules missing` | Se borró `~/Library/Caches/jbh-legal-build/` a pelo → symlinks colgando | `find . -maxdepth 3 -type l \( -name node_modules -o -name .next -o -name .turbo \) -delete` y re-ejecutar `icloud-unsync-build.sh` (recrea symlinks + `pnpm install --prefer-offline`, ~7 s) |
| `TypeError: Cannot read properties of undefined (reading 'length')` en `WasmHash._updateWithBuffer` | Caché incremental de webpack corrupta (intermitente: el primer build tras limpiar funciona, el incremental peta) | **Antes de cada build del portal:** `rm -rf apps/portal/.next/*` (sigue el symlink) y build limpio. No es un problema de Sentry aunque aparezca tras sus warnings. |
| Build de horas en vez de ~90 s | El symlink `.next` se convirtió en directorio REAL (iCloud dejó duplicados «.next 2/3/4») y iCloud sincroniza cada artefacto | `ls -la apps/<app> \| grep .next` debe mostrar `lrwx`; si es `drwx`, borrar y recrear el symlink hacia la caché. Revisarlo antes de cada build de `apps/web`. |
| OOM durante el build | Concurrencia de turbo con dependencias pesadas (googleapis) | `TURBO_CONCURRENCY=1` obligatorio |
| Se instala Next 16 en vez del 15.5 del workspace | Uso de `npx next build` | Construir siempre vía `pnpm --filter @jbhlegal/<app> build` |

> ℹ **Node 22 también en local.** El proyecto fija `.nvmrc` a 22 (igual que el servidor). En la máquina del operador existe un `prefix` en `~/.npmrc` que choca con nvm; usar `nvm use --delete-prefix 22`. Los patrones de `.gitignore` para los directorios redirigidos van *sin barra final*, porque son symlinks y no directorios.

## 2.8 Registro de incidencias con lección operativa

Las reglas de este capítulo no son teóricas: cada una cristaliza una incidencia real. Se registran aquí como memoria institucional, en orden cronológico:

| Fecha | Incidencia | Regla resultante |
|---|---|---|
| 2026-06-16 | Web pública en 503 tras un reinicio del servidor: `jbhlegal-web` no estaba en el ecosystem y `pm2 resurrect` no la levantó. | El ecosystem del repo es la fuente de verdad y debe incluir TODOS los procesos; `pm2 save` tras cada cambio (§2.5). |
| 2026-06-19 | Chunks de rutas dinámicas (`%5Bslug%5D`) en 404 por doble URL-decode en la cadena Cloudflare→Apache→Node. | `/_next/static` del apex servido por Apache desde disco (`Alias` + `ProxyPass !`, §3.2). |
| 2026-07-06 | Borrado a pelo de la caché de build → symlinks colgando y crashes WasmHash intermitentes. | Recuperación por script + limpieza de `.next` antes de cada build (§2.7). |
| 2026-07-10 | Deploys por scheduler «exitosos» que no reiniciaban nada (jaula del scheduler) y swap de `.next` en el directorio equivocado (`pm_cwd` vs `pm_exec_path`): rutas nuevas en 404 con el proceso sirviendo el manifiesto viejo desde memoria. | Restart y verificación siempre por SSH real; swap en `dirname(pm_exec_path)`; verificación por BUILD_ID (§2.6). |
| 2026-07-11 | Timeouts en PDFs escaneados grandes por 16 llamadas de visión concurrentes (4×4). | Concurrencia 3/3/2 del ai-worker (§2.2.1). |
| 2026-07-12 | Reinicio del ai-worker con un análisis de 69 páginas en vuelo: el análisis se abortó y reencoló, perdiendo horas. | Nunca reiniciar el ai-worker sin verificar cola ociosa; «parece colgado» casi siempre es espera de red del modelo (§2.6.2). |

| Fecha | Incidencia | Regla resultante |
|---|---|---|
| 2026-07-15 | Build de `apps/web` de 3+ horas: el symlink `.next` se había convertido en directorio real y iCloud sincronizaba cada artefacto. | Comprobar `lrwx` del symlink antes de cada build (checklist §2.6.5). |
| 2026-07-16 | Falsos «enlaces rotos» 4XX en herramientas SEO hacia portal/cliente por el antibot ModSecurity del proveedor. | Neutralización por-vhost + noindex de subdominios de app (§3.3). |

# 3
# Cloudflare, CDN y almacenamiento R2

> *Cloudflare cumple dos papeles en JBH Legal: es el borde de red de todos los dominios (DNS con proxy, CDN de los estáticos de Next, purga por API) y es el proveedor del almacenamiento de objetos R2 donde viven todos los binarios de la plataforma — documentos de casos, grabaciones, exports RGPD y los backups nocturnos de base de datos y secretos. Este capítulo documenta la zona, el comportamiento de caché y sus trampas de verificación, el antibot ModSecurity del VPS, la arquitectura de buckets R2, la generación de URLs prefirmadas en* `packages/storage` *y el sistema de backup a R2.*

## 3.1 La zona Cloudflare

El dominio `jbhasesorialegal.com` y todos sus subdominios están gestionados en una zona Cloudflare con el **zone id** `[id de cuenta reservado]` (un identificador público de recurso, no un secreto). Todos los registros de app llevan el **proxy activado** («nube naranja»): el tráfico entra por la red de Cloudflare, que termina TLS de cara al usuario y reenvía al origen (Apache en [IP reservada], capítulo 2).

Las operaciones de mantenimiento sobre la zona se hacen por la API v4 de Cloudflare con un token de API dedicado, guardado en el Keychain local del operador bajo el nombre `CLOUDFLARE_API_TOKEN_JBH` (valor out-of-band; nunca en el repo ni en el servidor). La operación más frecuente es la purga total de caché tras un deploy:

```
# Purga de caché tras un deploy (token desde Keychain local)
TOKEN=$(security find-generic-password -s CLOUDFLARE_API_TOKEN_JBH -w)
curl -X POST \
  "https://api.cloudflare.com/client/v4/zones/[id de cuenta reservado]/purge_cache" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  --data '{"purge_everything":true}'
```

ℹ **Ajustes de zona relevantes.** La normalización de URLs hacia el origen («Normalize URLs to origin») está **desactivada**: se comprobó durante el diagnóstico del bug de doble URL-decode de `%5B` (§3.2) que no era la causa, y se dejó OFF. Las rutas dinámicas de las apps se sirven con `cf-cache-status: DYNAMIC` (sin caché), lo que permite verificar deploys directamente contra la URL pública.

## 3.2 Caché de `/_next/static` y verificación de builds

Next.js emite sus assets bajo `/_next/static/` con nombres content-hashed, y Cloudflare los cachea agresivamente (respuestas `immutable`, `cf-cache-status: HIT` con `age` de días). Esto es deseable en operación normal y peligroso durante los deploys, porque produce tres ilusiones ópticas documentadas en incidentes reales:

1. **Chunks viejos que «funcionan»**: tras un swap incompleto (sin `static` dentro del `.next`, §2.6.2), los chunks sin cambios se siguen sirviendo desde la caché de Cloudflare aunque ya no existan en el origen. La app parece sana hasta que un usuario sin caché pide un chunk nuevo → 404.
2. **Readbacks cacheados**: los ficheros de log/verificación que los scripts de deploy escriben en rutas públicas quedan cacheados como `immutable`; releerlos muestra la corrida *anterior* aunque el deploy nuevo haya ocurrido.
3. **Falsos 200 de rutas API**: un 401 de `/api/**` no prueba nada — el middleware de auth devuelve 401 a cualquier ruta API sin sesión, exista o no en el build desplegado.

Lectura operativa de la cabecera `cf-cache-status` durante un deploy:

| `cf-cache-status` | Significado | Implicación en verificación |
|---|---|---|
| `HIT` (+ `age` alto) | Servido desde la caché de Cloudflare | No prueba nada sobre el origen: puede ser un asset ya inexistente en disco. |
| `MISS` / `EXPIRED` | Fue al origen en esta petición | Respuesta representativa del estado real. |
| `BYPASS` | Caché saltada por configuración/cabeceras | Útil para forzar readbacks fiables de ficheros nuevos. |
| `DYNAMIC` | Ruta no cacheable (HTML/APIs de las apps) | Las verificaciones funcionales en público son siempre en vivo. |

De ahí las dos reglas de verificación del proyecto:

✓ **Patrón BUILD_ID.** La única verificación fiable de que un build está sirviéndose es pedir el manifest del BUILD_ID nuevo: `GET https://<host>/_next/static/<BUILD_ID>/_buildManifest.js` → **200 = ese build está vivo; 404 = el swap no se aplicó**. El BUILD_ID se lee de `.next/BUILD_ID` del artefacto recién construido. Se comprueba primero contra el origen (`127.0.0.1:<puerto>`, por SSH) y después contra la URL pública tras purgar.

⚠ **Cloudflare ignora `?cb=` para estos readbacks: usar nombres de archivo ÚNICOS.** Añadir un query-string cache-buster a un fichero cacheado como immutable no garantiza un MISS. Todo fichero que un script escriba para ser releído por HTTP (logs de deploy, marcadores de backup) debe llevar un **nombre único por corrida** (p. ej. `_dplog_<app>_<timestamp>.txt`) en lugar de reutilizar la misma ruta. El mismo principio aplica al transporte por la GitHub Contents API (también cacheada por ruta en el borde): los artefactos y scripts de deploy se descargan por **blob-SHA** (`/repos/<owner>/<repo>/git/blobs/<sha>` con `Accept: application/vnd.github.raw`), que es content-addressed e inmune a caché.

Relacionado con el servicio de estáticos: en el apex (`apps/web`) los chunks de rutas dinámicas del App Router (`%5Bslug%5D` en el nombre de fichero) sufrían un **doble URL-decode en la cadena Cloudflare → Apache → Node** que convertía `%5B` en `[` antes de llegar al server de Next → 404 sistemático de esos chunks (rompía la hidratación del blog y los hubs de área; el SSR sí funcionaba). La solución aplicada en las directivas Apache del dominio (HTTP y HTTPS) fue sacar `/_next/static` del proxy y servirlo desde disco:

```
<Location "/_next/static">
  ProxyPass !
</Location>
Alias /_next/static "/var/www/vhosts/jbhasesorialegal.com/web/apps/web/.next/static"
<Directory "/var/www/vhosts/jbhasesorialegal.com/web/apps/web/.next/static">
  Require all granted
</Directory>
```

⚠ **Consecuencia para los deploys de web.** Al servirse `/_next/static` del apex desde disco por Apache, cada deploy de `apps/web` debe dejar el `static` nuevo exactamente en esa ruta de disco; un swap que solo actualice el server de Node dejará los estáticos viejos servidos por Apache.

## 3.3 ModSecurity: el antibot server-wide del proveedor

El VPS trae un conjunto de reglas ModSecurity del proveedor en `/etc/httpd/conf/plesk.conf.d/modsecurity.conf` (marcadas «PH RULES — NO MODIFICAR, se sobreescriben»). La cadena 90000/90001/**90002** deniega — visible como el 404 de la app Next, porque el deny ocurre en la fase 1 de Apache, antes del proxy — cualquier petición cuyo User-Agent contenga `bot|spider|check|crawl|grequests`, con una allowlist limitada (`Google|Uptime|Pingdom|Bing|Plesk|…`). La regla 90013 bloquea además el UA `python`, lo que afecta a scripts `urllib/requests` contra cualquier dominio del VPS.

| Aspecto | Detalle |
|---|---|
| Alcance | Server-wide, pero efectivo solo en los vhosts con el WAF activado. `web` y `firmas` lo tienen desactivado (no les afecta); `portal` y `cliente` lo tenían activado. |
| Síntoma típico | Herramientas SEO (Semrush/Ahrefs) reportan «enlaces rotos» 4XX hacia `portal.` / `cliente.`; los `access_log` de Apache muestran 404 a bots mientras el origen Node (`127.0.0.1:4100/4101`) responde 200. |
| Neutralización aplicada (2026-07-16) | En los `vhost.conf` / `vhost_ssl.conf` editables de `/var/www/vhosts/system/{portal,cliente}.jbhasesorialegal.com/conf/` (Plesk los preserva): `SecRuleRemoveById 90002` + `robots.txt` estático con `Disallow: /` servido desde el docroot con `ProxyPass /robots.txt !` + `Header always set X-Robots-Tag "noindex, nofollow"` + HSTS. `api.jbhasesorialegal.com` también con HSTS. Backups `.bak.<epoch>` junto a cada conf. |
| Criterio | Los subdominios de aplicación **no deben indexarse** (por eso el noindex explícito acompaña a la retirada de la regla); el sitio público del apex queda intacto y indexable. |

Diagnóstico rápido cuando se sospeche del antibot: comparar la misma URL con dos User-Agents desde fuera del VPS — `curl -A "Mozilla/5.0 …" https://portal.jbhasesorialegal.com/` (200/307) frente a `curl -A "MiBot/1.0 crawl" …` (404 con la página not-found de Next). Si divergen, el deny es de ModSecurity en Apache, no de la aplicación; confirmarlo en los `access_log` del vhost y en el audit log de ModSecurity antes de tocar nada. Recordar también la regla 90013 al escribir automatizaciones internas: cualquier script Python contra dominios del VPS debe fijar un User-Agent de navegador o fallará con 404 engañosos.

ℹ **Cambios como root sin SSH root.** El patrón usado para tocar configuración de Apache: escribir el script en el vhost por SSH de la suscripción → crear una tarea con la API Plesk (`POST /api/v2/cli/scheduler/call` con `-user root`) → ejecutarla → borrarla → leer la salida en `~/logs/`. Siempre con backup previo y `/usr/sbin/httpd -t` (configtest; `apachectl` no está en el PATH del cron) antes de `systemctl reload httpd`, con rollback automático si el configtest falla.

## 3.4 R2: arquitectura de buckets y contenido

Cloudflare R2 (API compatible S3) es el almacén de todos los binarios. La plataforma usa tres roles de bucket, resueltos por `bucketName(role)` en el adapter (§3.5):

| Rol | Bucket | Contenido |
|---|---|---|
| `prod` | `[bucket principal reservado]` (env `R2_BUCKET_PROD`) | Documentos de casos e intakes ya escaneados, grabaciones de reuniones, exports DSR, media de marketing, y los prefijos de backup `db-backups/` y `secrets-backup/` (§3.7). Localización UE. |
| `quarantine` | Bucket de cuarentena (env `R2_BUCKET_QUARANTINE`) | Destino inicial de toda subida de usuario, a la espera del escaneo ClamAV (§3.6). |
| `jurisprudence` | Bucket del módulo de jurisprudencia (env `R2_BUCKET_JURISPRUDENCE`) | PDFs de la biblioteca penal compartida (módulo Letrado IA). Es *opcional* en la config del adapter: solo lo pasan los callers del módulo (portal, ai-worker); si un caller usa el rol sin configurarlo, `bucketName` lanza un error explícito. |

Convención de claves de objeto para documentos (función `objectKey` de `packages/storage/src/r2-adapter.ts`):

```
<keyPrefix>/<caseId>/<docId>.<ext>      # keyPrefix ∈ {'cases', 'intakes'} (default 'cases')

Prefijos del bucket [bucket principal reservado]:
  cases/<caseId>/…            documentos y grabaciones de casos
  intakes/<intakeId>/…        documentos aportados en el intake
  db-backups/jbhlegal-<STAMP>.sql.gz          backup nocturno de BD
  secrets-backup/jbh-env-<STAMP>.tar.gz.enc   backup cifrado de shared/*.env
```

✓ **Cifrado en reposo y en tránsito.** R2 cifra *todo* objeto en reposo con AES-256 y claves gestionadas por Cloudflare, de forma incondicional; las cabeceras SSE de S3 son no-ops en R2, y por eso `putObject` en el adapter no envía `ServerSideEncryption` (comentario explícito en `r2-adapter.ts`). La residencia de datos en la UE se fija a nivel de bucket, no por petición. El acceso es siempre por HTTPS (TLS) con credenciales SigV4, y la descarga por los usuarios se hace mediante URLs prefirmadas de vida corta (§3.5.3), nunca con el bucket público.

La configuración del cliente se construye por app desde variables de entorno (patrón en `apps/portal/src/lib/storage.ts`, que valida la presencia de todas antes de crear el adapter): `R2_ACCOUNT_ID`, `R2_ACCESS_KEY_ID`, `R2_SECRET_ACCESS_KEY`, `R2_ENDPOINT`, `R2_BUCKET_PROD`, `R2_BUCKET_QUARANTINE`, `R2_BUCKET_JURISPRUDENCE`. Los valores viven en `shared/*.env` del servidor y en el Keychain local (out-of-band). El `R2_ENDPOINT` sigue el formato estándar de R2 (`https://<accountId>.r2.cloudflarestorage.com`) y el cliente S3 se configura con `forcePathStyle: true`, de modo que el bucket va en el path y no en el host — requisito para que las URLs prefirmadas resuelvan correctamente contra R2.

ℹ **Dos clientes R2 coexisten a propósito.** Las apps y workers del monorepo usan el adapter del SDK de AWS (`packages/storage`), que se bundlea con cada build. El backup nocturno usa en cambio `workers/jbh-r2.mjs`, un firmador SigV4 de ~200 líneas sin dependencias: puede ejecutarse desde un shell script del scheduler de Plesk con el Node pelado de `/opt/plesk/node/22/bin`, sin `node_modules`, y sobrevive a cualquier estado de los bundles de aplicación — exactamente lo que se quiere de una herramienta de disaster recovery.

## 3.5 `packages/storage`: adapter, límites y URLs prefirmadas

El paquete `packages/storage` define la interfaz `MultipartAdapter` (`src/types.ts`) con dos implementaciones: `createR2Adapter` (producción, `src/r2-adapter.ts`, sobre `@aws-sdk/client-s3` + `@aws-sdk/s3-request-presigner`, con `region: 'auto'` y `forcePathStyle: true`) y `createFilesystemAdapter` (desarrollo). La superficie completa del adapter:

| Método | Función | Detalle de implementación R2 |
|---|---|---|
| `createMultipartUpload` | Inicia una subida | Clave `<prefijo>/<caseId>/<docId>.<ext>`; devuelve `{uploadId, key, partSize, partCount}`; error explícito si R2 no devuelve `UploadId` |
| `getPresignedPartUrls` | Firma cada parte | Un `UploadPartCommand` firmado por parte, TTL por defecto 3600 s |
| `completeMultipartUpload` | Cierra la subida | Ordena las partes por número y limpia comillas de los ETags (`stripQuotes`) |
| `abortMultipartUpload` | Cancela | Idempotente: absorbe `NoSuchUpload` |
| `getPresignedDownloadUrl` | Descarga temporal | Fuerza `ResponseContentType` y `Content-Disposition` (inline/attachment + RFC 5987); TTL por defecto 300 s |
| `copyObject` / `deleteObject` | Promoción y borrado | Usados por el pipeline AV (§3.6) y la retención RGPD |
| `listObjects` | Listado por prefijo | `MaxKeys` por defecto 100; base de la poda de backups |
| `getObjectStream` / `getObjectBuffer` / `getObjectHead` | Lectura server-side | `getObjectHead` lee solo un rango `bytes=0-N` (sniffing MIME sin descargar el fichero) |
| `putObject` | Escritura server-side | Sin cabeceras SSE (no-op en R2; AES-256 incondicional) |

### 3.5.1 Límites y tipos permitidos (`src/limits.ts`)

| Constante | Valor | Función |
|---|---|---|
| `MAX_FILE_BYTES` | 500 MB (`500 * 1024 * 1024`) | Tamaño máximo por documento |
| `PART_SIZE` | 8 MB | Tamaño de parte multipart; `partCount = ceil(sizeBytes / PART_SIZE)`, mínimo 1 |
| `ALLOWED_EXTENSIONS` | 15 extensiones: `pdf docx doc odt jpg jpeg png zip xlsx xls pptx ppt csv rtf txt` | Allowlist dura de subida; mapeadas 1:1 a MIME en `EXTENSION_TO_MIME` |
| `detectMimeType` / `isLikelyExecutable` | (`src/mime.ts`) | Sniffing del contenido real y rechazo de ejecutables camuflados |

### 3.5.2 Flujo de subida multipart

La subida es *directa del navegador a R2*: el servidor solo firma URLs, nunca ve los bytes. Rutas del portal: `api/cases/[id]/documents/upload-init` / `upload-complete` / `upload-abort` (el componente `uploader` de `packages/ui` orquesta el cliente).

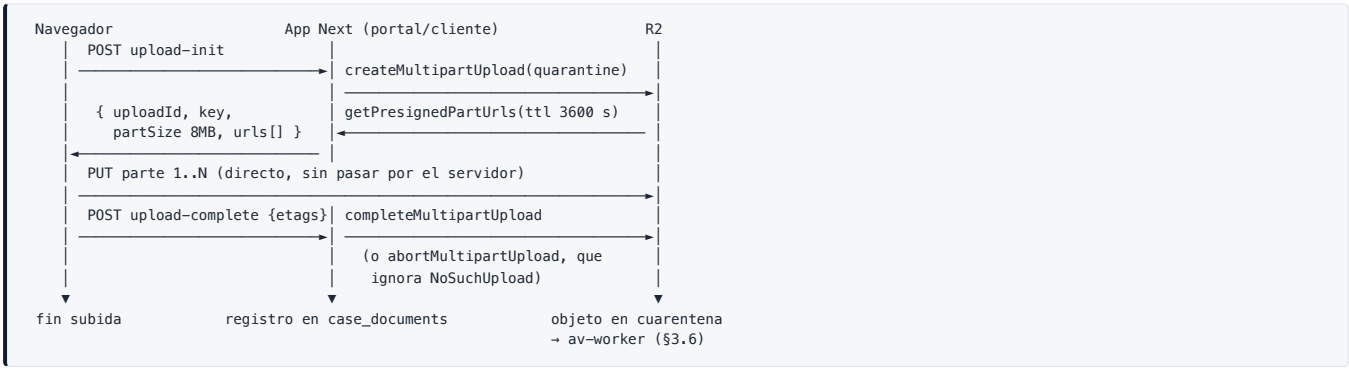


Fig. 3.1 — Subida multipart con URLs prefirmadas: el binario nunca atraviesa el servidor de aplicación.

### 3.5.3 URLs prefirmadas de descarga: TTLs reales

`getPresignedDownloadUrl` firma un `GetObjectCommand` forzando `ResponseContentType` y `ResponseContentDisposition` (`inline` o `attachment`, con el filename saneado y su variante RFC 5987 `filename*=UTF-8''…` generada por `encodeRfc5987`). El TTL por defecto es **300 s**; cada ruta fija el suyo según el riesgo del recurso:

| Ruta (apps/portal) | Operación | TTL |
|---|---|---|
| `api/cases/[id]/documents/[did]/download` | Descarga de documento de caso | 300 s |
| `api/intakes/[id]/documents/[did]/download` | Descarga de documento de intake | 120 s |
| `api/jurisprudence/[id]/file` | PDF de la biblioteca de jurisprudencia | 300 s |
| `api/cases/[id]/meetings/[mid]/audio` | Audio de grabación de reunión | 600 s |
| `api/cases/[id]/meetings` (subida de grabación) | Presign de subida | 3600 s |
| `api/cases/[id]/documents/upload-init` | Presign de partes multipart | 3600 s (default del adapter para partes) |

Toda ruta que firma una URL pasa antes por el gate de acceso del recurso (`requireCaseAccess` / `requireIntakeAccess`, §1.7): la URL prefirmada es la *segunda* barrera, no la primera. El resto de la interfaz del adapter — `copyObject` (promoción cuarentena→prod), `deleteObject`, `listObjects`, `getObjectStream` / `getObjectBuffer` / `getObjectHead` (lectura server-side para el ai-worker y el exporter DSR, que en `apps/portal/src/lib/dsr-exporter.ts` hace streaming de cada `storagePath` del bucket `prod` al ZIP de exportación), `putObject` — se usa solo desde código de servidor.

### 3.5.4 Paridad dev/prod: el adapter de filesystem

`createFilesystemAdapter` implementa la misma interfaz `MultipartAdapter` sobre disco local, lo que permite desarrollar y testear todos los flujos de documentos sin credenciales R2. Los tests de `packages/storage/src/__tests__/` y los de rutas del portal (p. ej. `intake-documents-download-route.test.ts`, que verifica que la descarga de intake pide `{bucket: 'prod', ttlSec: 120}`) mockean el adapter, no el SDK: cualquier cambio en la interfaz rompe los tests de todas las rutas consumidoras, que es el comportamiento deseado. El exporter DSR distingue por documento el origen `'local'` (vault filesystem histórico) de `'r2'`, de modo que los documentos antiguos siguen siendo exportables.

## 3.6 Pipeline antivirus: cuarentena → producción

Ninguna subida de usuario aterriza directamente en `[bucket principal reservado]`. El flujo: la subida multipart deposita el objeto en el bucket de cuarentena; el proceso pm2 `jbhlegal-av-worker` (always-on, §2.2) lo recoge, lo escanea con ClamAV (aprovisionado con `infra/plesk/provision-clamav.sh`) y, si está limpio, lo promueve con `copyObject({fromBucket:'quarantine', toBucket:'prod'})` + borrado del original; si no, lo marca infectado y el documento nunca es descargable. Los estados se reflejan en la fila de `case_documents` / `intake_documents`, y la UI de subida muestra errores accionables al usuario.

### 3.6.1 Otros flujos que escriben en R2

- **Grabaciones de reuniones**: la pestaña «Grabaciones» del caso sube el audio (grabación en móvil con locución de consentimiento) directamente al bucket `prod` con presign de 3600 s (`api/cases/[id]/meetings`); el `ai-worker` (`execute-meeting.ts`) lo transcribe y resume. La inclusión de la grabación en los análisis del caso es *opt-in* (`include_in_analysis`, por defecto `false`: privilegio abogado-cliente); la cola referencia la reunión por `meeting_id` propio, sin FK a `document_id`. Retención RGPD de 10 años con soporte DSR.
- **Jurisprudencia**: los PDFs de la biblioteca penal compartida viven en el bucket del rol `jurisprudence`; el `ai-worker` los ingiere (`extract-precedent-text.ts`, `precedent-batch.ts`, `precedent-persist.ts`) y los indexa para búsqueda semántica (`index-precedent-rag.ts`, con el backfill `backfill-precedent-rag.ts`). La descarga hacia el abogado usa presign de 300 s (`api/jurisprudence/[id]/file`).
- **Exports DSR**: el ejercicio de derechos de acceso/portabilidad (`dsr-exporter.ts`) construye un ZIP haciendo streaming de cada documento del bucket `prod` (distinguiendo el origen `'local'` filesystem legacy de `'r2'` por documento) y lo entrega por canal autenticado.

- **Media de marketing**: las tarjetas editoriales del worker de marketing (`apps/portal/src/lib/marketing/ig-image.ts`) se suben con su propio adapter mínimo si el env R2 está presente; su ausencia degrada sin romper (devuelve `null`).

## 3.7 Backup nocturno a R2 y cliente SigV4 propio

El backup nocturno **no depende de pm2**: es la tarea **1038 del scheduler de Plesk**, programada a las **02:30 UTC**, que ejecuta `/var/www/vhosts/jbhasesorialegal.com/workers/jbh-nightly-backup.sh`. El script usa `workers/jbh-r2.mjs`, un **cliente R2 con firma AWS SigV4 implementado solo con Node** (sin rclone ni aws-cli), que lee las credenciales `R2_ACCESS_KEY_ID` / `R2_SECRET_ACCESS_KEY` / `R2_ENDPOINT` / `R2_BUCKET_PROD` de los `shared/*.env` del propio servidor: los secretos nunca salen de la máquina.


Fig. 3.2 — Backup nocturno de BD y secretos a R2. El dump es pequeño porque los documentos viven en R2, no en la BD.

| Elemento | Detalle |
|---|---|
| BD | `mysqldump --single-transaction` de la BD única `jbhlegal` \| `gzip` → `db-backups/jbhlegal-<UTCstamp>.sql.gz` |
| Secretos | Los 10 `shared/*.env` en un tar cifrado **asimétricamente** con el certificado público `shared/jbh-dr-cert.pem` → `secrets-backup/jbh-env-<STAMP>.tar.gz.enc`. Solo la clave privada DR (offline, fuera del servidor: `~/Documents/jbh-recovery/jbh-dr-private.pem` en la máquina del operador + gestor de contraseñas) puede descifrarlos: un atacante con acceso al servidor no puede leer los backups de secretos. |
| Retención | 30 días; la poda solo toca los prefijos `db-backups/jbhlegal-` y `secrets-backup/jbh-env-` (nunca `jbh-recovery-local-*`). |
| Verificación | El log de la última corrida se publica en una ruta estática del apex y debe terminar en `DONE` con `db http=200` y `secrets http=200`. Consciente del gotcha de caché de §3.2, cualquier automatización que lo relea debe contar con que Cloudflare puede servir la versión anterior. |
| Runbook | `docs/runbooks/12-disaster-recovery.md` (DR completo = acceso Cloudflare + clave privada DR). El método histórico con rclone (`docs/runbooks/mariadb-backup.md`) nunca se instaló y está obsoleto. |

⚠ **Regla RGPD del backup.** El volcado de BD contiene datos penales (categoría especial, arts. 9/10 RGPD): **jamás** se sube a GitHub ni a ningún destino fuera de R2-UE o un contenedor cifrado. El mismo criterio excluye del repo los PDFs societarios y cualquier fichero de credenciales, aunque el repositorio sea privado.

✓ **Backup on-demand.** Además del nocturno, existe un procedimiento manual con el mismo mecanismo (script de dump + subida SigV4 lanzado por el canal scheduler), útil antes de migraciones de BD. El deploy normal de las apps *no* ejecuta migraciones: las tablas nuevas se aplican con un ensure-script dedicado contra MariaDB, con backup previo (§2.6.4).

### 3.7.1 Recuperación (disaster recovery)

El escenario de pérdida total del servidor se recupera con dos llaves y un runbook:

| Pieza | Dónde vive | Para qué |
|---|---|---|
| Acceso a la cuenta Cloudflare | Credenciales del operador (out-of-band) | Recuperar DNS de la zona y acceso a los buckets R2 con los backups |
| Clave privada DR `jbh-dr-private.pem` | `~/Documents/jbh-recovery/` (máquina del operador, sincronizada a iCloud) + gestor de contraseñas | Descifrar `secrets-backup/jbh-env-*.tar.gz.enc` → recuperar los 10 `shared/*.env` |
| Runbook | `docs/runbooks/12-disaster-recovery.md` | Secuencia completa: restaurar BD desde `db-backups/`, reponer `shared/*.env`, redeplegar bundles, `pm2 resurrect` |

⚠ **La clave privada DR es el single point of failure del descifrado.** El certificado público `shared/jbh-dr-cert.pem` del servidor solo sirve para *cifrar*; sin la clave privada offline no existe ningún camino para descifrar los backups de secretos. Debe mantenerse al menos una copia en un gestor de contraseñas además del directorio local.

## 3.8 Resumen de dependencias Cloudflare

Recapitulación de todo lo que la plataforma delega en Cloudflare, como referencia rápida para evaluar el impacto de una incidencia del proveedor:

| Servicio | Uso en JBH Legal | Impacto si cae |
|---|---|---|
| DNS + proxy (zona `[id reservado]`) | Resolución y borde de todos los dominios | Plataforma inaccesible (el origen no acepta tráfico directo útil de cara al público) |
| CDN | Caché de `/_next/static` | Más carga en Apache/Node; sin pérdida funcional |
| API v4 (purga) | Post-deploy (token `CLOUDFLARE_API_TOKEN_JBH`) | Los deploys de estáticos tardan en propagarse hasta expirar la caché |
| R2 ( `[bucket principal reservado]` + cuarentena + jurisprudencia) | Todos los binarios, backups y evidencias | Subida/descarga de documentos y análisis de IA sobre documentos inoperativos; la BD y las apps siguen vivas |

# 4
# Base de datos y multi-tenancy

> *La persistencia de JBH Legal vive en una única instancia MariaDB en el VPS de producción, modelada con Drizzle ORM en el paquete `@jbhlegal/db` (`packages/db`). El esquema comprende 140 tablas definidas en 119 archivos TypeScript, con un modelo multi-tenant por columna `org_id` que permitió pasar del «Modelo A» (clientes directos de JBH) al «Modelo B» (despachos como tenants) sin ruptura. Este capítulo documenta la topología, el inventario del esquema por dominios funcionales, las reglas de colación, la arquitectura de tenancy con sus clases de bug conocidas, las tablas con garantías especiales (`audit_log`, `internal_coupons`) y el estado del cifrado en reposo.*

## 4.1 Motor, topología y acceso

El motor es **MariaDB** (InnoDB), desplegado en el propio VPS Plesk de producción y escuchando exclusivamente en `localhost:3306`. No existe exposición del puerto de base de datos a Internet: todas las aplicaciones (`apps/portal`, `apps/cliente`, `apps/web`, `apps/firmas`, `apps/herencias`, `apps/peritos`, `apps/api` y el proceso `ai-worker`) corren en la misma máquina y se conectan por loopback. El acceso administrativo remoto se realiza únicamente mediante **túnel SSH**:

```
ssh -L 3307:127.0.0.1:3306 <usuario>@<host-vps>
# y en local:  DATABASE_URL='mysql://…@127.0.0.1:3307/…'
```

La cadena de conexión vive en la variable de entorno `DATABASE_URL` (formato `mysql://usuario:contraseña@host:puerto/base`). Su valor es secreto y se gestiona out-of-band (Keychain local / `shared/*.env` del servidor); nunca aparece en el repositorio, en prompts ni en logs. Los ficheros `shared/*.env` del servidor se respaldan cada noche cifrados asimétricamente en R2 como parte del plan de disaster recovery (runbook `docs/runbooks/12-disaster-recovery.md`).

El cliente de aplicación es **Drizzle ORM** (`drizzle-orm/mysql2`, versión `^0.45.2`) sobre el driver `mysql2`. El punto de entrada es `packages/db/src/client.ts`, que exporta la instancia `db` y el tipo `Database`; el esquema completo se re-exporta desde `packages/db/src/schema/index.ts` y el paquete se consume vía los exports `@jbhlegal/db` y `@jbhlegal/db/schema`.

> ℹ **Copias de seguridad.** La base se respalda cada noche (tarea del scheduler de Plesk, 02:30 UTC) con `mysqldump | gzip` hacia el bucket R2 `[bucket principal reservado]` (prefijo `db-backups/`), con retención de 30 días. Los dumps salen en claro (leen datos descifrados), por lo que el plan de cifrado en reposo (§4.8) no exime de proteger también los ficheros de backup. Runbook: `docs/runbooks/mariadb-backup.md`.

## 4.2 Charset y colación: la regla crítica

El estándar del proyecto es `CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci`, declarado *explícitamente* en cada `CREATE TABLE` de las migraciones. Esta regla no es estética: es una de las lecciones operativas más caras del proyecto.

> ⚠ **Gotcha de colación (regla crítica).** Mezclar colaciones revienta los JOIN. Si una tabla nueva se crea con la colación por defecto del servidor (`utf8mb4_unicode_ci`) y se une por una columna `varchar` contra una tabla existente en `utf8mb4_general_ci`, MariaDB o bien falla con *«Illegal mix of collations»* o bien descarta el índice y degrada el JOIN a un escaneo con conversión implícita. Además, el intento de declarar una clave foránea entre columnas de colaciones distintas falla con **errno 150**. Caso real: `cases.id` conserva la colación legada `utf8_general_ci` (utf8mb3) de las primeras migraciones, por lo que cualquier FK declarativa contra `cases` desde una tabla `utf8mb4` es imposible — véase §4.6. Toda migración DEBE llevar `COLLATE utf8mb4_general_ci` explícito; nunca confiar en el default del servidor.

## 4.3 Inventario del esquema por dominios funcionales

El esquema se define en `packages/db/src/schema/`: 119 archivos `.ts` (un archivo por tabla o por familia pequeña de tablas) que producen **140 tablas**. En lugar de enumerarlas una a una, se presentan agrupadas por dominios funcionales, con las tablas principales de cada dominio y sus columnas clave. Salvo indicación contraria, todas las tablas usan claves primarias `varchar(36)` con `DEFAULT (UUID())`, timestamps `created_at` / `updated_at` y, si son tenant-owned, la columna `org_id` (§4.4).

### 4.3.1 Identidad, acceso y organizaciones

| Tabla | Archivo | Columnas clave / propósito |
|---|---|---|
| `users` | `users.ts` | Identidad global (email, nombre, rol base). El tenant NO se guarda aquí: se resuelve por membership. |
| `user_roles`, `user_preferences` | `user-roles.ts`, `user-preferences.ts` | Roles adicionales y preferencias (digest de notificaciones) atadas al usuario, no al despacho. |
| `organizations` | `organizations.ts` | El despacho/tenant: `id`, `name`, `legal_name`, `slug` (único), `type`, `status`, `owner_user_id`, datos de colegiado verificado (`colegiado_num`, `colegiado_verified_at`, `colegiado_nif_hash`), clave Anthropic propia cifrada (`anthropic_key_enc`, `anthropic_key_last4`), datos públicos de marketplace y `branding_json`. |

| Tabla | Archivo | Columnas clave / propósito |
|---|---|---|
| `memberships` | `memberships.ts` | Pertenencia usuario↔org: `user_id`, `org_id`, `role` (p. ej. `LAWYER`), `status`, `permissions_json`; UNIQUE `uq_membership_user_org`. |
| `organization_subscriptions` | `organization-subscriptions.ts` | Suscripción del despacho: `plan`, `period`, `status`, `stripe_customer_id`, `stripe_subscription_id`, cupos (`seats_included`, `expedientes_override`, `ai_quota_json`), consentimiento de overage de IA (`ai_overage_enabled`, `ai_overage_consent_at`), planes custom (`is_custom`, `custom_price_cents`). |
| `member_invitations`, `client_invitations`, `org_custom_plan_requests`, `org_invoicing_settings` | homónimos | Invitación de miembros del despacho, invitación/onboarding de clientes del Modelo B, solicitudes de plan a medida y configuración de facturación por org. |
| `colegiado_identities`, `colegiado_challenges`, `colegiado_assisted_requests` | `colegiado-*.ts` | Verificación de la condición de abogado colegiado (retos de verificación y flujo asistido por superadmin). |
| `client_access_links` | `client-access-links.ts` | Enlaces de acceso del cliente (login OTP / acceso delegado). |

#### 4.3.2 Expedientes y causas

| Tabla | Columnas clave / propósito |
|---|---|
| `cases` | Núcleo del expediente penal: `expediente_num` (formato `JBH-YYYY-NNN`, único), `org_id`, `client_user_id`, `lead_lawyer_id`, `coordinator_id`, `status`, `area`, `penal_category`, `fase_procesal`, `tipo_procedimiento` (delitos leves ≠ abreviado: ramifica los escritos), `posicion_cliente` (defensa/acusación: invierte el enfoque del Letrado IA), `rol_cliente`, `situacion_libertad`, `detencion_json`, `hechos`, `antecedentes`, `base_context` (+ `base_context_source` / `_validated_at`), `case_model` (A/B), `firm_managed`, setup fee Stripe (`setup_fee_cents`, `setup_fee_status`, `setup_fee_stripe_payment_intent_id`), `hoja_encargo_status`, soft-delete (`deleted_at`, `deleted_by_id`), `client_visibility_json`, `causa_id`. |
| `causas` | Agrupación multi-expediente (causa con varios investigados) para análisis comparativo. |
| `case_events`, `case_milestones` | Cronología de negocio del expediente (eventos e hitos visibles en la pestaña Actividad). |
| `case_tasks`, `case_court_events` | Tareas del caso y señalamientos judiciales (vistas, plazos). |
| `case_collaborators`, `case_lawyer_inputs`, `case_lawyer_invites` | Abogados colaboradores asignados al caso, sus aportaciones y las invitaciones. |
| `case_witnesses`, `case_trial_prep` | Testigos del caso y preparación del juicio oral. |
| `case_proposals` | Marketplace de casos externos; el `org_id` es el despacho *destino*, por lo que la bandeja se scopea como el resto de tablas tenant-owned. |
| `case_fianza_consents` | Consentimientos relativos a fianzas. |

#### 4.3.3 Cuestionarios y clarificaciones

| Tabla | Propósito |
|---|---|
| `case_questionnaires` | Instancia de cuestionario del caso (estado, límite `questionnaire_max_questions` en `cases`). |
| `questionnaire_questions` / `questionnaire_answers` | Preguntas generadas por IA (calibradas: solo imprescindibles, con dedupe semántico) y respuestas del cliente. |
| `case_answer_evaluations` | Evaluación IA de las respuestas (coherencia, riesgos para el testimonio). |
| `case_clarifications` | Aclaraciones puntuales solicitadas al cliente fuera del cuestionario. |
| `causa_core_questions` | Preguntas nucleares a nivel de causa (multi-investigado). |

#### 4.3.4 Documentos y análisis IA

| Tabla | Columnas clave / propósito |
|---|---|
| `case_documents` | Documentos del expediente; el binario vive en R2 (cifrado AES-256), aquí los metadatos y la clave de objeto. |
| `document_analyses` | Análisis IA por documento: `document_id` (UNIQUE `uq_doc_analyses_doc`), `status`, `attempts`, `classification`, `extracted_text`, `summary`, `entities_json`, `key_evidences_json`, progreso por páginas (`progress_pages` / `progress_total`), coste (`model`, `tokens_input` / `tokens_output`, `cost_cents`). |
| `document_analysis_jobs` | Cola de trabajos del `ai-worker` (reclamación de jobs, reintentos, zombies). |
| `case_analyses` / `causa_analyses` | Informe global del caso / de la causa generado por IA. |
| `causa_comparativa_contradicciones` | Contradicciones detectadas entre expedientes de una misma causa. |

| Tabla | Columnas clave / propósito |
|---|---|
| `case_executive_summaries`, `case_prognosis_snapshots` | Resumen ejecutivo y fotos periódicas del pronóstico del caso. |
| `case_drafts` / `case_draft_versions` / `causa_drafts` | Escritos del Letrado IA (~45 generadores): `kind` (p. ej. `claves_del_caso`; UNIQUE `uq_case_drafts_case_kind`), `status` (`generating` / `ready` / `failed`), `content_markdown`, `version`, auditoría jurídica observable (`audit_status`, `audit_json`, `audited_at`), validación humana (`validated_by_id`, `validated_at`) y coste. |
| `ai_request_log` | Registro de TODA llamada a IA: `scope`, `model`, `tokens_input` / `tokens_output`, `cost_cents` (coste crudo, sin margen), `latency_ms`, `status_code`, `case_id`, `document_id`, `org_id` — base de la imputación de consumo por despacho. |
| `ai_overage_charges` | Cargos por consumo de IA por tramos: `period_month`, `block_index` (UNIQUE `uq_ai_overage_block`), `amount_cents`, `stripe_payment_intent_id`, `status`, `failure_reason`. |

#### 4.3.5 Mensajería y notificaciones

| Tabla | Propósito |
|---|---|
| `case_messages` / `case_message_attachments` | Mensajería del expediente entre despacho y cliente, con adjuntos. |
| `case_chat_threads` / `case_chat_messages` | Chat IA contextual del caso (hilos + mensajes). |
| `notifications` / `message_notification_queue` | Notificaciones in-app y cola de avisos por email de mensajes no leídos. |
| `whatsapp_links` / `whatsapp_events` | Vinculación de números de WhatsApp y eventos del canal. |
| `perito_messages` | Canal Q&A perito↔abogado↔cliente dentro del flujo de peritaje. |

#### 4.3.6 Pagos y facturación

| Tabla | Columnas clave / propósito |
|---|---|
| `case_payments` | Cobros por caso vía Stripe: `kind`, `amount_cents`, `status`, `stripe_payment_intent_id` / `stripe_session_id` / `stripe_charge_id` (UNIQUE por sesión), reembolsos (`refund_amount_cents`) y disputas (`dispute_id`, `dispute_status`). |
| `invoices` + `invoice_counters` | Facturación propia: numeración legal por serie y año (`series`, `year`, `legal_number`; el contador `invoice_counters.last_number` garantiza correlativos), `doc_type` (`factura` / `rectificativa` con `rectifies_id`), desglose `base_cents` / `iva_cents` / `total_cents`, `payer_type`, snapshots `issuer_json` / `recipient_json` y referencias Stripe. |
| `lawyer_invoices` | Facturas emitidas por el despacho al cliente desde el portal. |
| `stripe_webhook_events` | Idempotencia de webhooks de Stripe (tabla global, sin `org_id`). |
| `internal_coupons` | Cupones internos que descuentan el TOTAL (cuota + IA). Diseño de seguridad en §4.5.2. |
| `herencia_payments` | Cobros del producto de herencias. |

#### 4.3.7 Intakes, leads y CRM

| Tabla | Columnas clave / propósito |
|---|---|
| `intakes` | Entrada del embudo y a la vez «lead» del CRM (enfoque B): datos de contacto, `estado` / `status`, triaje IA (`ia_score`, `ia_resumen`, `ia_recomendacion`, `triage_result`), consentimientos versionados (`consent_datos_penales`, `consent_privacidad`, `privacy_policy_version`), pipeline CRM (`lead_type`, `probability`, `value_estimated_cents`, `next_action_at`, `lost_reason`), atribución (`utm_source` / `utm_medium` / `utm_campaign`, `source`) y enlace al caso (`case_id`). |
| `intake_documents`, `intake_events`, `intake_notes` | Documentos aportados en el alta, eventos del pipeline y notas internas. |
| `leads`, `lead_captures` | Leads de marketing y capturas del lead magnet. |
| `crm_tasks` | Tareas del CRM (incluye las generadas por IA ante urgencias procesales). |
| `crm_presupuestos` | Presupuestos con aceptación pública por token: `numero` (correlativo por org, UNIQUE `uq_crm_presu_org_numero`, calculado MAX+1), `lineas` (JSON), `subtotal_cents` / `iva_percent` / `total_cents`, `status` (borrador→enviado→visto→aceptado/rechazado/caducado), `public_token_hash` (hash del token de aceptación, un solo uso), evidencia de decisión (`decided_ip`, `decided_user_agent`). |

#### 4.3.8 Firma electrónica, agenda y reuniones

| Tabla | Columnas clave / propósito |
|---|---|
| `signature_requests` | Solicitudes de firma eIDAS: `document_id`, `document_hash_sha256` (huella del PDF firmado), `signer_user_id` + snapshots (`signer_name_snapshot`, `signer_email_snapshot`), `token_jti` (UNIQUE, anti-replay), `status`, `expires_at`, revocación (`revoked_at`, `revocation_reason`). |
| `signatures` | Firma consumada con su evidencia. |
| `bookings`, `booking_audit_events` | Reservas de cita con pago y su rastro de auditoría. |

| Tabla | Columnas clave / propósito |
|---|---|
| `lawyer_availability`, `lawyer_calendars`, `lawyer_booking_settings` | Disponibilidad, calendarios (Google Calendar) y configuración de reservas por letrado. |
| `case_meetings` | Reuniones/videollamadas del caso. |
| `meeting_recordings` | Grabaciones de reuniones (flujo Plaud): `r2_key`, `status` (desde `uploading`), `transcript_json`, `speakers_json`, `summary_md`, `consent_info` (locución de consentimiento) y el flag de privilegio `include_in_analysis` (opt-in, `false` por defecto: la grabación NO entra en los análisis IA salvo decisión expresa). |

#### 4.3.9 Herencias / ISD

El producto sucesorio usa un sub-esquema propio, todo tenant-owned (en producción pertenece al tenant JBH): `herencia_consultas`, `herencia_questions`, `herencia_answers`, `herencia_herederos`, `herencia_inventario`, `herencia_documents`, `herencia_document_analyses`, `herencia_reports` y `herencia_payments`; más la tabla de referencia global `ccaa_herencia_reglas` (matriz autonómica del ISD, sembrada por `db:seed-ccaa`).

#### 4.3.10 Jurisprudencia (biblioteca penal)

| Tabla | Propósito |
|---|---|
| `legal_precedents` | Resoluciones: `ecli`, `court` / `chamber` / `section`, `resolution_number` / `resolution_date`, resúmenes (`summary_short`, `summary_technical`, `facts_summary`, `legal_grounds_summary`, `doctrine`), `criminal_categories`, `legal_articles`, dedupe (`content_simhash`, `normalized_text_hash`, `duplicate_status`), `visibility` (separación dura común/privado por org: `uploaded_by_org_id`) y `validation_status`. |
| `legal_precedent_chunks` | Chunks para RAG (embeddings en Cloudflare Vectorize; aquí el texto fuente). |
| `legal_precedent_collections`, `legal_precedent_favorites`, `legal_precedent_links`, `legal_precedent_reports`, `legal_precedent_audit`, `precedent_batches` | Colecciones, favoritos, vínculos precedente↔caso, informes, auditoría de uso y lotes de ingesta. |
| `case_jurisprudence_suggestions` | Sugerencias del Motor Favorable: jurisprudencia favorable destacada por caso con su motivo. |
| `criminal_taxonomy` | Taxonomía penal de referencia (categorías de delito). |

#### 4.3.11 Referidos, peritos/colaboradores, DSR, auditoría, soporte y marketing

| Dominio | Tablas | Notas |
|---|---|---|
| Programa de Colaboradores (referidos) | `referral_profiles`, `referral_aliases`, `referral_links`, `referral_clicks`, `referral_invites`, `referral_subjects`, `referral_attributions`, `referral_events` (append-only), `referral_economics`, `referral_payouts` | Alias único global (`/r/{alias}`), clicks sin PII en claro (hashes truncados, `ts_day` para dedupe), dedupe de sujetos por `email_hash` / `nif_hash`. |
| Peritos y red de colaboradores | `professional_profiles` (global: perfil de la persona), `perito_case_grants`, `perito_doc_grants`, `perito_presupuestos`, `perito_report_drafts`; `collaborator_profiles`, `collaborator_documents`, `collaborator_invitations`, `collaborator_practice_areas`, `collaborator_subscriptions` | El acceso del perito se concede por caso/documento (grants tenant-owned del despacho que concede). |
| RGPD / DSR | `dsr_requests` | Derechos del interesado: `kind` (acceso/portabilidad/supresión), `status`, decisión motivada (`decided_by_id`, `decision_reason`) y artefacto de exportación con caducidad (`artifact_path`, `artifact_token`, `artifact_expires_at`). |
| Auditoría | `audit_log` (§4.5.1), `superadmin_audit` | `superadmin_audit` es global y cross-tenant: el tenant objetivo va en `target_org_id`. |
| Soporte y feedback | `support_tickets`, `support_ticket_messages`, `suggestions` | Tickets con hilo de mensajes (`last_sender_role`, `last_message_at`) y buzón de sugerencias. |
| Marketing y web | `content_pieces`, `marketing_settings`, `marketing_topics`, `linkbuilding_links`, `office_addresses`, `lead_captures` | `content_pieces` es tenant-owned (piezas publicadas); el resto, configuración/tracking global. |

## 4.4 Multi-tenancy por `org_id`

#### 4.4.1 El tenant plataforma: `JBH_ORG_ID`

La pieza central del modelo es la constante `JBH_ORG_ID` (`packages/db/src/constants.ts`):

```
export const JBH_ORG_ID = '00000000-0000-4000-8000-0000000a1b01' as const;
```

Es un UUID v4 sintácticamente válido (versión 4, variante 8) y **fijo**: identifica a la propia plataforma JBH como un tenant más, y debe coincidir exactamente con el `id` de la fila sembrada en `organizations` (seed de `0078_orgid_tenant_columns.sql`). No debe cambiarse nunca en producción. Este diseño permitió la transición sin ruptura del Modelo A al Modelo B en tres movimientos:

| # | Movimiento | Efecto |
|---|---|---|
| 1 | Añadir `org_id varchar(36) NOT NULL DEFAULT '<JBH_ORG_ID>'` a todas las tablas tenant-owned | Al ejecutar el `ALTER`, MariaDB rellena las filas EXISTENTES con el default → backfill automático de todo el histórico al tenant JBH, sin scripts de migración de datos. |
| 2 | Los INSERT preexistentes del Modelo A (que no pasaban `orgId`) siguen funcionando | El DEFAULT asigna cada fila nueva a JBH sin tocar código. |
| 3 | El código del Modelo B sobreescribe el default con el `orgId` del despacho | El scoping por tenant se activa de forma incremental, tabla a tabla y ruta a ruta. |

#### 4.4.2 Clasificación de tablas: tenant-owned vs globales

El módulo `packages/db/src/tenant.ts` mantiene dos listas blancas explícitas que son la fuente de verdad de la clasificación:

- `TENANT_OWNED_TABLES`: las tablas cuyas filas pertenecen a UNA organización y cuyas consultas DEBEN filtrarse por `org_id`. La lista salió de una auditoría exhaustiva del esquema (una sesentena de tablas en la migración `0078`, ampliada después con cada fase: peritos, member_invitations, case_proposals…) y hoy cubre el núcleo de casos/causas/intakes, todo el contenido del caso (documentos, mensajes, cuestionarios, escritos, análisis), firmas, notificaciones, DSR, agenda, WhatsApp, `content_pieces`, `ai_request_log` (para imputar coste por despacho) y todo el sub-esquema de herencias.
- `GLOBAL_TABLES`: identidad y Auth (`users`, `accounts`, `sessions`, `verification_tokens`), el propio modelo de tenant (`organizations`, `memberships`), perfiles de profesionales externos (`professional_profiles`: son de la persona, no del despacho), referencia e infra (`ccaa_herencia_reglas`, `stripe_webhook_events`, `office_addresses`, `marketing_settings`, `marketing_topics`, `user_preferences`) y `superadmin_audit`.

El módulo expone además `withTenant(orgId)`: helpers `select/insert/update/delete` que envuelven el cliente Drizzle e inyectan SIEMPRE la condición `org_id = :orgId` (y, en insert, fuerzan el valor de `org_id`), de modo que sea imposible leer o escribir filas de otro despacho aunque el `where` del llamante «olvide» el tenant. Su corrección está cubierta por los tests de aislamiento `packages/db/src/__tests__/tenant-isolation.test.ts`, complementados por tests de aislamiento cross-org a nivel de aplicación y un lint de gates.


Fig. 4.1 — Cadena de aislamiento multi-tenant: gate por entidad → scoping por `org_id` → defaults de columna.

#### 4.4.3 La clase de bug «INSERT sin `orgId` → fila invisible»

> ⚠ **Clase de bug sistémica (barrida el 2026-07-20).** El mismo DEFAULT que hizo indolora la transición crea una trampa simétrica: si un flujo que actúa *en nombre de un despacho* (Modelo B) hace un INSERT en una tabla tenant-scoped sin fijar `orgId`, la fila no falla — se crea silenciosamente asignada a `JBH_ORG_ID`. Como todas las lecturas del despacho filtran por su `org_id`, la fila queda **invisible para el despacho que la creó** (y potencialmente visible en paneles de la plataforma). No hay error, no hay log: solo un dato que «desaparece». En la barrida del 2026-07-20 se auditaron todos los INSERT sobre tablas tenant-scoped y se corrigieron **7 sitios** que omitían el `orgId`.

> ✓ **Regla vigente.** Todo INSERT en una tabla tenant-scoped fija `orgId` *explícitamente*, aunque el flujo actual solo opere para el tenant JBH. El DEFAULT de columna es una red de compatibilidad con código legado del Modelo A, no una API: el código nuevo nunca debe apoyarse en él.

Relacionadas con esta clase de bug están las demás reglas sistémicas del multi-tenant documentadas en el proyecto: toda ruta `cases/[id]` o `intakes/[id]` pasa por su gate (`requireCaseAccess` / `requireIntakeAccess`); añadir un rol a una ruta sin su gate equivale a un IDOR; y los UNIQUE que representen correlativos por despacho deben ser compuestos con `org_id` (p. ej. `uq_crm_presu_org_numero`, `uq_audit_log_org_seq`).

## 4.5 Tablas con garantías especiales

#### 4.5.1 `audit_log`: registro append-only con cadena de hash

Definida en `packages/db/src/schema/audit-log.ts` (migración `0084_audit_log.sql`), es el rastro forense de propósito general de la plataforma, distinto de `case_events` / `case_milestones` (eventos de negocio) y de `superadmin_audit` (acciones cross-tenant). Sus columnas: `id`, `org_id`, `seq` (posición monotónica

1,2,3… dentro de la cadena de la org), `actor_id` / `actor_email` (null = acción del sistema: cron, worker), `action` (verbo estable: `case.create`, `document.download`, `draft.generate`, `member.invite`, `dsr.export` …), `entity_type` / `entity_id`, `data` (JSON estructurado, sin PII en claro; la IP del actor solo se guarda como `ipHash` SHA-256), `prev_hash`, `hash` y `created_at` con precisión de milisegundos (`datetime(3)`).

La integridad criptográfica se implementa en dos módulos con separación estricta:

- `packages/db/src/lib/audit-hash.ts` — núcleo **puro** (sin BD): `canonicalize()` (serialización JSON determinista con claves ordenadas recursivamente en todos los niveles), `computeAuditHash()`, `verifyChain()` y la constante `GENESIS_HASH` (64 ceros). Testeado en aislamiento en `__tests__/audit-hash.test.ts`.
- `packages/db/src/lib/audit.ts` — acceso a BD: `appendAuditLog()` y `verifyAuditChain()`.

La cadena es exactamente:

```
hash(n) = SHA-256( hash(n-1) || ":" || canonical(fila n) )
hash(0) previo = GENESIS_HASH = "000…0" (64 ceros)
```

En el cálculo entran, en orden canónico estable: `seq`, `orgId`, `actorId`, `actorEmail`, `action`, `entityType`, `entityId`, `data` y `createdAt` (ISO-8601 UTC). Cada org tiene su propia cadena independiente; el UNIQUE `uq_audit_log_org_seq (org_id, seq)` garantiza el orden monotónico y bloquea duplicados de secuencia.

```
org A:  GENESIS ──► fila 1 ──► fila 2 ──► fila 3 ──► …
        (64×'0')   seq=1      seq=2      seq=3
                   prev=GEN   prev=h1    prev=h2
                   hash=h1    hash=h2    hash=h3

h2 = SHA-256( h1 || ":" || canonical{seq:2, orgId, actorId, actorEmail,
                                     action, entityType, entityId,
                                     data, createdAt} )

Alterar/borrar/insertar la fila 2  ⇒  h2' ≠ h2  ⇒  rompe h3, h4, …
verifyChain(): 1) seq contiguo desde 1   2) prevHash == hash anterior
               3) hash recomputado == hash almacenado
               → { valid, brokenAtSeq, reason, checked }
```

Fig. 4.2 — Cadena de hash de `audit_log`: registro *tamper-evident* por organización.

`appendAuditLog()` serializa por org dentro de una `db.transaction`: lee la última fila de la org (`ORDER BY seq DESC LIMIT 1`), calcula `seq = prev.seq + 1` y `prevHash = prev.hash` (o `GENESIS_HASH`), computa el hash e inserta. Si dos escrituras concurrentes calculan el mismo `seq`, el índice único hace fallar a una (`ER_DUP_ENTRY` /1062) y la función reintenta una única vez. La tabla es append-only *por convención de código*: nunca se hace UPDATE ni DELETE, y toda escritura pasa por `appendAuditLog()`. La manipulación no se impide a nivel de motor, pero `verifyAuditChain(orgId)` — pensada para una consola de auditoría o un cron de control de integridad — detecta cualquier alteración, borrado o inserción retroactiva devolviendo el primer `seq` roto y el motivo (`seq no contiguo` / `prevHash no enlaza` / `hash no coincide`). Esta trazabilidad verificable es un requisito de rendición de cuentas frente a los tenants y de tratamiento de datos penales (arts. 9/10 RGPD).

### 4.5.2 `internal_coupons`: cupones sin código en claro

Los cupones internos (schema `internal-coupons.ts`; lógica en `apps/portal/src/lib/internal-coupons.ts`) descuentan el TOTAL de la factura del despacho — cuota fija *y* consumo de IA (`scope='total'`) — a diferencia de los cupones de Stripe, que solo tocan la cuota. Su diseño de seguridad:

- **Nunca se almacena el código en claro.** En BD solo existe `code_hash = sha256(SECRET:código)`, con sal por el secreto de servidor `INTERNAL_COUPON_SECRET` (valor out-of-band). Ni con acceso total a la BD se pueden derivar códigos válidos. El código se muestra EN CLARO una única vez al generarlo (`generateInternalCoupon()`) y no puede volver a leerse.
- **Alta entropía:** 160 bits aleatorios codificados en Base32 sin caracteres ambiguos (alfabeto Crockford-ish sin O/0/I/1), formateados como `JBH-XXXX-XXXX-XXXX-XXXX` → no enumerables.
- **Validez siempre respaldada por BD:** porcentaje limitado a {10, 25, 50, 75, 100}, caducidad obligatoria (`expires_at`, 6 meses por defecto), límite de canjes (`max_redemptions` / `times_redeemed`) y estado revocable (`status: active|revoked`). UNIQUE `uq_internal_coupon_hash` sobre el hash.

El canje (`redeemInternalCoupon(code, orgId)`) es idempotente por org y distingue dos modos:

| Modo | Condición | Comportamiento |
|---|---|---|
| Un solo uso (clásico) | `max_redemptions = 1` (default) | El cupón se LIGA a la org que lo canjea (`org_id` en la propia fila) con un update CAS sobre `times_redeemed` para no exceder el cupo en carrera. Canje repetido por la misma org → idempotente; por otra org → `other_org`. |
| Compartido (campañas) | `max_redemptions > 1` | El código maestro NUNCA se liga. Cada canje incrementa el contador del maestro (CAS) y crea una **fila hija** ligada a la org, con el mismo %/scope/caducidad, `code_hash` derivado de `código#orgId` (respeta el UNIQUE y habilita el check idempotente) y `created_by` = **id del cupón maestro**. Revocar el maestro NO revoca las hijas ya emitidas; se revocan aparte localizándolas por `created_by`. |

`getOrgInternalDiscountPercent(orgId)` resuelve el descuento vigente de una org (máximo entre cupones activos y no caducados, caché de 30 s) y, ante cualquier error, devuelve 0: fallo del subsistema de cupones = cobro íntegro, el comportamiento seguro para el negocio.

## 4.6 Integridad referencial en código: sin claves foráneas

⚠ **Sin FKs en las tablas SaaS.** Las tablas de las fases SaaS **no declaran claves foráneas**. El motivo es empírico: la BD de producción arrastra tablas antiguas en colación `utf8_general_ci` (utf8mb3) — señaladamente `cases.id` — y una constraint entre columnas de colaciones distintas falla con **errno 150** al crear la tabla. En lugar de reescribir la colación de tablas con datos penales en caliente, la decisión de arquitectura es usar «FKs suaves»: la columna referencia existe y está indexada, pero la constraint no se declara y la **integridad referencial vive en el código de servicio** (gates de acceso, borrados en cascada explícitos, y validaciones al escribir). El patrón está documentado en cabecera de las migraciones (p. ej. `0123_referral_core.sql`: «FKs suaves SIN constraint (colación legacy… → errno 150); integridad en app»).

Dos convenciones acompañan a esta decisión:

- **Numeración MAX+1, no COUNT+1.** Los correlativos (`cases.expediente_num`, `crm_presupuestos.numero`, contadores de factura) se calculan con `MAX(sufijo)+1`, nunca con `COUNT(*)+1`. El helper `apps/portal/src/lib/expediente-num.ts` lo explica: la numeración puede tener huecos (soft-deletes incluidos en el índice único, borrados históricos), así que COUNT+1 puede proponer un número ya existente y reventar el alta con *Duplicate entry*; MAX del sufijo es robusto frente a huecos:

  ```
  SELECT MAX(CAST(SUBSTRING(expediente_num, LENGTH('JBH-YYYY-')+1) AS UNSIGNED))
    FROM cases WHERE expediente_num LIKE 'JBH-YYYY-%';   -- → maxn + 1
  ```

- `db.transaction` **en toda escritura multi-paso.** Sin FKs ni triggers, la atomicidad de operaciones compuestas (crear caso + cuestionario + evento; canje de cupón compartido; append del audit_log) depende de envolverlas en `db.transaction`. Es regla de revisión: una escritura multi-paso fuera de transacción es un defecto.

## 4.7 Semillas y utilidades del paquete

El paquete `@jbhlegal/db` expone scripts operativos vía pnpm (definidos en `packages/db/package.json`): `db:seed-admin` (alta del administrador), `db:seed-ccaa` (matriz autonómica del ISD), `db:seed-demo` más `db:demo-lock` / `db:demo-unlock` (cuentas demo con casos ficticios y su bloqueo), además de los del pipeline de migraciones (`db:apply`, `db:drift`) que se documentan en el capítulo 5. Los queries compartidos viven en `packages/db/src/queries/` y las utilidades de visibilidad cliente/caso en `src/case-client-visibility.ts` y `src/lib/client-visibility.ts`.

## 4.8 Cifrado en reposo: estado real

Los datos penales (análisis IA, respuestas de cuestionario, intakes, mensajes) residen hoy en MariaDB **en claro**; los documentos binarios en R2 sí están cifrados en reposo (AES-256). El plan aprobado es **TDE de MariaDB** (InnoDB tablespace encryption), documentado en el runbook `docs/runbooks/mariadb-encryption-at-rest.md` y con este estado:

ℹ **Estado: pendiente de ejecución.** El TDE es configuración del servidor MariaDB (plugin `file_key_management`, edición de `my.cnf` y reinicio del servicio incluido) y un error en `my.cnf` puede dejar la plataforma entera sin BD; por eso NO se automatiza desde el deploy de la app y debe ejecutarlo el administrador en ventana de mantenimiento. El runbook cubre: generación y custodia de la clave (keyfile cifrado con passphrase; la clave raíz se guarda en gestor de secretos con copia offline — si se pierde, los datos son irrecuperables), configuración (`innodb_encrypt_tables=ON`, `innodb_encrypt_log=ON`, `encrypt_binlog=ON`, `encrypt_tmp_files=ON`, `aria_encrypt_tables=ON`, algoritmo AES_CTR), reinicio controlado con verificación del plugin, `ALTER TABLE … ENCRYPTED=YES` para las tablas sensibles (`case_analyses`, `causa_analyses`, `document_analyses`, `questionnaire_answers`, `intakes`, `case_messages` …), verificación vía `information_schema.innodb_tablespaces_encryption`, rotación de claves sin downtime y rollback. Se eligió TDE frente al cifrado por campo porque es transparente para la app, cubre toda la BD (tablas + redo log + binlog) y no rompe búsquedas, ordenación ni índices. Nota operativa clave: los `mysqldump` siguen saliendo en claro, así que los backups se cifran aparte.

# 5
# Pipeline de migraciones y verificación de esquema

> *El esquema de JBH Legal ha evolucionado a través de 125 archivos SQL en `packages/db/migrations` (numerados `0000` → `0125`), pero el migrator estándar de Drizzle dejó de ser fiable en producción: su journal está desincronizado y un bug de su comparador reporta migraciones como aplicadas sin ejecutarlas. El proyecto sustituyó ese eslabón por un pipeline propio de dos piezas — un runner de SQL idempotente (`db:apply`) y un detector de drift que compara el esquema Drizzle contra la base real (`db:drift`) — complementado por reglas estrictas de redacción de migraciones y un método de respaldo sin SSH vía el scheduler de Plesk. Este capítulo documenta la historia, el pipeline oficial, ambos scripts en detalle y el estado verificado del esquema.*

## 5.1 Historia: del migrator de Drizzle al pipeline propio

### 5.1.1 El estado del directorio de migraciones

El directorio `packages/db/migrations/` contiene **125 archivos SQL** con numeración `NNNN_nombre.sql` desde `0000_initial.sql` hasta `0125_referral_payouts.sql` (la secuencia tiene algún hueco histórico, p. ej. no existe `0001`), más el `README.md` del pipeline y el subdirectorio `meta/` de Drizzle. Cada archivo es una unidad de cambio de esquema: tablas nuevas, columnas, índices, backfills. Ejemplos representativos del recorrido: `0004_cases_and_related.sql`, `0018_stripe_webhook_events.sql`, `0078_orgid_tenant_columns.sql` (la columna `org_id` multi-tenant), `0084_audit_log.sql`, `0118_crm_core.sql`, `0121_colegiado_asistida.sql`, `0123_referral_core.sql`.

La historia completa puede leerse como una estratigrafía del producto. A grandes rasgos:

| Rango | Época / dominio | Migraciones representativas |
|---|---|---|
| `0000` – `0018` | Fundación: intake, casos, documentos, mensajes, firmas, pagos, DSR, notificaciones, R2, Stripe | `0000_initial`, `0004_cases_and_related`, `0007_signatures`, `0010_dsr_requests`, `0017_perf_indexes`, `0018_stripe_webhook_events` |
| `0019` – `0028` | Reservas y cuestionarios | `0019_booking_system`, `0021_questionnaires`, `0024_case_answer_evaluations`, `0026_case_clarifications`, `0028_message_notification_queue` |
| `0029` – `0048` | Letrado IA fase 17: causas, escritos, chat, pronóstico, señalamientos, WhatsApp | `0029_causas`, `0032_executive_summaries`, `0034_case_drafts` (+ `0035_fix_case_drafts`, `0036_case_drafts_status`), `0037_case_chat`, `0040_case_prognosis_snapshots`, `0044_whatsapp`, `0046_case_soft_delete` |
| `0049` – `0076` | Herencias/ISD, endurecimiento (2FA), leads, correcciones de esquema | `0049_herencias`, `0059_utf8mb4_text_columns`, `0064_totp_2fa`, `0067_intake_triage`, `0073_leads`, `0076_fix_uuid_defaults` |
| `0077` – `0082` | **Multi-tenancy (SaaS abogados)** | `0077_organizations_memberships`, `0078_orgid_tenant_columns` (la columna `org_id` + backfill), `0079_organization_subscriptions`, `0080_client_invitations`, `0081_case_proposals`, `0082_member_invitations` |
| `0083` – `0098` | Jurisprudencia, auditoría, soporte, red de letrados, fixes de colación | `0083_jurisprudencia`, `0084_audit_log`, `0086_jurisprudence_dedup`, `0088_support_suggestions_tickets`, `0091_case_drafts_audit`, `0096_case_witnesses`, `0098_fix_new_tables_collation` |
| `0099` – `0117` | Facturación propia y cobro de IA metered | `0099_invoices` (+ `0101_invoices_rectificativas`), `0103_ai_overage`, `0108_case_trial_prep`, `0113_metered_ai_billing`, `0114_internal_coupons`, `0115_client_case_metered` |
| `0118` – `0125` | CRM jurídico, colegiación asistida y Programa de Colaboradores | `0118_crm_core`, `0119_crm_presupuestos`, `0120_lawyer_invoices`, `0121_colegiado_asistida`, `0123_referral_core`, `0124_referral_economics`, `0125_referral_payouts` |

> ℹ **Las migraciones «fix» son historia viva de los gotchas.** Varias migraciones existen solo para corregir clases de fallo documentadas en este manual: `0035_fix_case_drafts` (secuela directa del bug del migrator), `0059_utf8mb4_text_columns`, `0089_herencias_utf8mb4` y `0098_fix_new_tables_collation` (la colación heredada del default del servidor, §4.2) y `0076_fix_uuid_defaults` (defaults `UUID()` que no quedaron aplicados). Son el argumento empírico de las reglas de redacción de §5.3.

El journal de Drizzle, `packages/db/migrations/meta/_journal.json`, registra sin embargo solo **36 entradas** — la última es `0036_case_drafts_status`. A partir de ahí, las migraciones se escribieron y aplicaron fuera del ciclo `drizzle-kit generate` +journal, de modo que el journal quedó permanentemente desincronizado (36 entradas frente a 125 archivos). Esto por sí solo ya inutiliza a `drizzle-kit migrate` como fuente de verdad de «qué está aplicado».

### 5.1.2 El bug del migrator en producción (comparador `folderMillis`)

> ⚠ **«Applied successfully» sin crear las tablas.** Además del journal desincronizado, el migrator de drizzle-kit exhibió en producción un fallo peor que un error: un **falso éxito**. Su comparador decide qué migraciones ejecutar comparando el timestamp artificial `folderMillis` / `when` de cada migración con el último `created_at` registrado en la tabla `__drizzle_migrations`. Si el último `created_at` es mayor que el `when` de una migración pendiente, el migrator la da por aplicada *sin ejecutar su SQL* y reporta «applied successfully». Está documentado en cabecera de `packages/db/scripts/ensure-phase17-tables.mjs`: las migraciones de la fase 17 (`0032` case_executive_summaries, `0033` columna `alerted_window_days`, `0034` case_drafts) «no crearon sus objetos» pese al éxito reportado, y el GET/POST de `/escritos` devolvía 500 porque `case_drafts` no existía. El caso más grave fue posterior: `0084_audit_log` figuraba como aplicada y **la tabla `audit_log` no existió en producción hasta el 2026-07-20**, cuando el detector de drift la destapó.

La anatomía del falso éxito merece detallarse porque explica por qué era tan difícil de detectar. El flujo interno de `drizzle-kit migrate` es: leer el journal → asignar a cada migración un timestamp sintético (`when` / `folderMillis`) → consultar la última fila de `__drizzle_migrations` en la BD destino → ejecutar solo las migraciones cuyo timestamp sea posterior. Como los `when` del journal de JBH son valores artificiales (p. ej. `1780000015000` para `0036`) y la tabla `__drizzle_migrations` de producción contenía filas con `created_at` posteriores, el comparador concluía «nada pendiente», marcaba el run como exitoso y el deploy continuaba en verde. El fallo solo aflora *después*, cuando el código toca la tabla ausente: en fase 17 fue un 500 en `/escritos` (no existía `case_drafts`); con `0084` fue el propio detector de drift, meses más tarde, quien reveló que `audit_log` nunca se creó.

> ⚠ **Regla operativa. NO usar `drizzle-kit migrate` contra producción.** Los scripts `db:migrate` / `migrate:prod` siguen existiendo en `packages/db/package.json` por compatibilidad histórica, pero el pipeline vigente no los emplea. `drizzle-kit generate` sí sigue siendo útil para *redactar* el SQL de una migración a partir del esquema TypeScript; lo que no es fiable es su *aplicación*. La configuración de drizzle-kit vive en `packages/db/drizzle.config.ts`: `schema: './src/schema/index.ts'`, `out: './migrations'`, `dialect: 'mysql'`, credenciales tomadas de `DATABASE_URL`, y flags `verbose` + `strict`.

## 5.2 El pipeline oficial: SQL idempotente → `db:apply` → `db:drift`

El pipeline vigente está descrito en `packages/db/migrations/README.md` y consta de tres pasos:

| Paso | Herramienta | Comando |
|---|---|---|
| 1. Escribir | Editor (+ `drizzle-kit generate` opcional como borrador) | Crear `migrations/NNNN_nombre.sql` idempotente y conforme a las reglas de §5.3. |
| 2. Aplicar | `scripts/apply-sql.ts` | `DATABASE_URL='mysql://…' pnpm --filter @jbhlegal/db db:apply migrations/NNNN_x.sql` |
| 3. Verificar | `scripts/check-schema-drift.ts` | `DATABASE_URL='…' pnpm --filter @jbhlegal/db db:drift` → debe imprimir «SIN DRIFT». |


Fig. 5.1 — Pipeline oficial de migraciones: el esquema TypeScript es la fuente de verdad y el drift-checker la red de seguridad que el migrator no daba.

La lógica del diseño: como las migraciones son idempotentes, el runner no necesita llevar contabilidad de qué se aplicó (re-ejecutar es inocuo), y como el verificador compara el estado *real* de la BD contra el esquema completo, un run limpio demuestra positivamente que todo existe — exactamente la garantía que el journal corrupto no podía dar.

### 5.2.1 `apply-sql.ts`: el runner oficial

`packages/db/scripts/apply-sql.ts` (script pnpm `db:apply`, ejecutado con `tsx`) aplica un archivo `.sql` contra `DATABASE_URL`, **statement a statement**, mostrando el resultado de cada uno. Su funcionamiento:

- **Entrada:** `DATABASE_URL` por entorno y la ruta del archivo como primer argumento. Si falta cualquiera de los dos, imprime el uso y sale con **código 2**.
- **Troceo (`splitStatements`):** elimina las líneas de comentario completas (las que empiezan por `--`, donde viven el contexto y el rollback documentado) y divide el texto por `;` a fin de línea (regex `/;\s*(?:\n|$)/`). No soporta delimitadores custom ni procedimientos almacenados — no se usan en el proyecto.

- **Ejecución:** abre una conexión `mysql2/promise` y ejecuta cada statement en orden, imprimiendo ✅ `[i/N] <primeros 90 caracteres>` o 🔴 `[i/N] …` con el mensaje de error. **Al primer fallo se detiene** (no sigue ejecutando statements posteriores sobre un estado a medias: se revisa a mano) y fija código de salida **1**.
- **Sin registro:** deliberadamente NO escribe en `__drizzle_migrations` ni en ninguna tabla de contabilidad. La idempotencia del SQL es la que hace segura la re-ejecución, y el drift-checker es quien certifica el resultado.

```
DATABASE_URL='mysql://…' pnpm --filter @jbhlegal/db db:apply migrations/0084_audit_log.sql
# migrations/0084_audit_log.sql: 1 statement(s)
#   ✅ [1/1] CREATE TABLE IF NOT EXISTS audit_log ( id varchar(36) NOT NULL DEFAULT (UUID()), …
```

#### 5.2.2 `check-schema-drift.ts`: el detector de drift

`packages/db/scripts/check-schema-drift.ts` (script pnpm `db:drift`) compara el esquema Drizzle — la fuente de verdad en `src/schema` — contra la base de datos REAL usando `information_schema`. En detalle:

1. **Lado esperado:** importa `* as schemaMod from '../src/schema'` y recorre todos los exports aplicando `getTableConfig()` de `drizzle-orm/mysql-core`; los exports que no son tablas (relations, enums, helpers) lanzan y se ignoran. Obtiene así la lista deduplicada y ordenada de tablas esperadas con TODAS sus columnas (nombres SQL reales).
2. **Lado real:** una única consulta `SELECT TABLE_NAME, COLUMN_NAME FROM information_schema.COLUMNS WHERE TABLE_SCHEMA = DATABASE()`, materializada como mapa tabla → set de columnas.
3. **Comparación y salida:** imprime el recuento (`Esquema: N tablas esperadas · BD: M tablas reales`) y clasifica:
   - 🔴 **Tablas del esquema que NO existen en la BD** — exactamente el artefacto que produce el bug del migrator «applied sin crear».
   - 🔴 **Columnas definidas que faltan** en una tabla existente (migración de `ADD COLUMN` no aplicada).
   - ℹ️ **Tablas presentes en la BD sin definición en el esquema** — solo informativo: legado, tablas de framework, `__drizzle_migrations`…

| Código de salida | `db:apply` | `db:drift` |
|---|---|---|
| `0` | Todos los statements ejecutados. | «✅ SIN DRIFT: todas las tablas y columnas del esquema existen en la BD.» |
| `1` | Un statement falló (se detiene ahí; revisar a mano). | Hay drift 🔴 (tablas o columnas ausentes). Integrable en CI/deploy como gate. |
| `2` | Uso incorrecto (falta `DATABASE_URL` o el archivo). | Falta `DATABASE_URL`, o no se pudo derivar ninguna tabla del esquema. |

✓ **Garantía positiva.** Un run limpio de `db:drift` no dice «no consta nada pendiente» (la garantía débil del journal): demuestra que *cada una de las 140 tablas y cada una de sus columnas* existe físicamente en la base de datos consultada. Es la red de seguridad que convirtió el incidente de `audit_log` en una clase de fallo detectable en segundos.

#### 5.2.3 Ejecución contra producción: túnel SSH

MariaDB solo escucha en `localhost:3306` del VPS (§4.1), de modo que ambos scripts se lanzan desde la máquina de trabajo a través de un túnel SSH con reenvío de puerto local:

```
ssh -L 3307:127.0.0.1:3306 <usuario>@<host-vps>     # terminal 1 (mantener abierta)

# terminal 2 — aplicar y verificar contra prod por el túnel:
DATABASE_URL='mysql://…@127.0.0.1:3307/…' pnpm --filter @jbhlegal/db db:apply migrations/NNNN_x.sql
DATABASE_URL='mysql://…@127.0.0.1:3307/…' pnpm --filter @jbhlegal/db db:drift
```

Las credenciales de la URL son secretas y viven out-of-band (Keychain local / `shared/*.env` del servidor); nunca se escriben en el repo ni en scripts commiteados.

## 5.3 Reglas de redacción de migraciones

Toda migración nueva debe cumplir cuatro reglas, que a la vez hacen posible el pipeline (idempotencia) y evitan las clases de fallo conocidas (colación, FKs):

| Regla | Formulación | Motivo |
|---|---|---|
| 1. Idempotencia | `CREATE TABLE IF NOT EXISTS`, `ADD COLUMN IF NOT EXISTS`, `CREATE INDEX IF NOT EXISTS` … siempre que la sintaxis de MariaDB lo permita. | El runner no lleva registro de aplicadas: re-ejecutar un archivo debe ser inocuo. También habilita los reparadores `ensure-*` y el método sin SSH. |
| 2. Colación explícita | `ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci` en cada `CREATE TABLE`. | El default del servidor (`unicode_ci`) revienta los JOIN contra las tablas existentes y las FKs con errno 150 (§4.2). |
| 3. Sin claves foráneas | Columnas de referencia indexadas, pero SIN `FOREIGN KEY … REFERENCES`. | Colación legada en tablas antiguas (`cases.id` es `utf8_general_ci`/utf8mb3) → errno 150 en prod. Integridad referencial en el código de servicio (§4.6). |
| 4. Rollback documentado | Comentario en cabecera con el SQL inverso (p. ej. `-- Rollback: DROP TABLE IF EXISTS audit_log;`). | El runner no genera «down migrations»; el camino de vuelta se decide y documenta al escribir, cuando el contexto está fresco. |

Ejemplo real que exhibe las cuatro reglas (cabecera y cierre de `migrations/0084_audit_log.sql`):

```
-- Fase 4 (SaaS abogados) — audit_log: registro append-only a prueba de
-- manipulación (hash encadenado) por tenant.
-- Todo ADITIVO (tabla nueva, sin filas previas; JBH intacto). Como el resto de
-- tablas SaaS, NO se declaran claves foráneas (errno 150 en la BD de
-- producción): la integridad referencial vive en el código de servicio.
-- CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci (colación crítica del proyecto).
--
-- Rollback:
--   DROP TABLE IF EXISTS audit_log;

CREATE TABLE IF NOT EXISTS audit_log (
  id varchar(36) NOT NULL DEFAULT (UUID()),
  org_id varchar(36) NOT NULL DEFAULT '00000000-0000-4000-8000-0000000a1b01',
  …
  UNIQUE KEY uq_audit_log_org_seq (org_id, seq),
  …
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Checklist operativa al redactar y aplicar una migración nueva (resume las reglas anteriores en pasos verificables):

1. Nombrar `NNNN_snake_case.sql` con el siguiente número libre del directorio (hoy: a partir de `0126`).
2. Actualizar primero el esquema TypeScript en `packages/db/src/schema/` (y exportarlo desde `index.ts`): el drift-checker deriva de ahí lo que va a exigir, de modo que esquema y SQL no pueden divergir sin que el paso 6 lo delate.
3. Si la tabla es tenant-owned: columna `org_id varchar(36) NOT NULL DEFAULT '<JBH_ORG_ID>'`, índice por `org_id`, alta en `TENANT_OWNED_TABLES` (`src/tenant.ts`) y UNIQUEs de correlativos compuestos con `org_id` (§4.4).
4. Redactar el SQL idempotente, con colación explícita, sin FKs y con el rollback comentado en cabecera.
5. Probar en local: `db:apply` dos veces seguidas (la segunda debe ser un no-op limpio: prueba de idempotencia) + `db:drift`.
6. Aplicar en producción por el túnel (`db:apply`) y certificar con `db:drift` → «SIN DRIFT».
7. Desplegar el código que usa el esquema nuevo *después* (o a la vez, si el cambio es aditivo y el código viejo lo tolera). El deploy normal de las apps no ejecuta migraciones (§5.6).

ℹ **Convenciones adicionales.** Numeración `NNNN_snake_case.sql` correlativa al último archivo existente; cambios ADITIVOS siempre que sea posible (columnas nuevas con DEFAULT, tablas nuevas) para que el código viejo y el nuevo convivan durante el deploy; los backfills de datos, como el de `org_id` (`0078`), se apoyan en `NOT NULL DEFAULT` para que el propio `ALTER` rellene el histórico; y los UNIQUE de correlativos por despacho se declaran compuestos con `org_id`.

## 5.4 Los reparadores históricos `ensure-*`

Antes de que existiera el pipeline `db:apply` + `db:drift`, el bug del migrator se parcheó con scripts de reparación puntual en `packages/db/scripts/`, que hoy se conservan como herramienta histórica/de emergencia:

- `ensure-phase17-tables.mjs` — diagnóstico + reparación best-effort de los objetos de la fase 17 que el migrator dio por aplicados sin crear (`case_executive_summaries`, la columna `alerted_window_days`, `case_drafts`). No depende de drizzle: conecta directamente con `mysql2` y ejecuta `CREATE TABLE / ADD COLUMN IF NOT EXISTS`. Además imprime el estado real (existencia de tablas antes/después y la cola de `__drizzle_migrations`) para diagnóstico. Es idempotente, seguro de re-ejecutar en cada deploy y *nunca rompe el deploy*: si falta `DATABASE_URL` o algo falla, sale con código 0 tras loguearlo.
- `ensure-support-tables.mjs` — mismo patrón para las tablas de la migración `0088` (`suggestions`, `support_tickets`, `support_ticket_messages`): comprueba con `SHOW TABLES LIKE`, crea lo que falte de forma idempotente y reporta el antes/después.

El contrato de comportamiento de ambos, que los distingue del pipeline oficial, es el siguiente:

| Propiedad | `ensure-*.mjs` (reparador) | `db:apply` + `db:drift` (pipeline) |
|---|---|---|
| Alcance | Un conjunto cerrado de objetos, cableado en el script | Cualquier archivo de `migrations/` / todo el esquema |
| Ante error | Loguea y sale con código 0 («nunca rompe el deploy») | Código de salida distinto de 0: el fallo es visible y bloqueante |
| Diagnóstico | Imprime existencia antes/después + cola de `__drizzle_migrations` | `db:drift` certifica el esquema completo (140 tablas) |
| Cuándo usarlo | Embebido en un deploy desatendido; reparación de emergencia | Siempre, para toda migración nueva |

ℹ **Estado actual.** Los `ensure-*` quedan como reparadores puntuales y ejemplo del patrón «asegurar esquema sin drizzle». Para migraciones nuevas la vía es siempre `db:apply` + `db:drift`; un `ensure-script` solo se justifica cuando debe ejecutarse embebido en un deploy sin intervención (véase §5.5).

## 5.5 Método de respaldo sin SSH: el scheduler de Plesk

Cuando el acceso SSH directo no está disponible (p. ej. bloqueo por IP de origen), existe un camino de respaldo para aplicar cambios de BD en producción usando la API REST de Plesk y su programador de tareas (*scheduler*): se sube un script a la máquina (vía la API de ficheros o el propio artefacto de deploy), se crea una tarea programada que lo ejecute una vez, y se lee el resultado por HTTP. El patrón, probado en producción, es:

1. **Preparar un ensure-script idempotente** (patrón §5.4) o un `deploy-db.sh` que: parsee `DATABASE_URL` desde el `.env` del servidor con `node` (nunca credenciales inline en la tarea ni en el SQL), invoque el cliente `mysql` con un fichero `.cnf` temporal de opciones, ejecute el SQL idempotente y escriba un

log de resultado. Esqueleto del patrón (sin valores de secretos; las credenciales solo existen en el `.env` del servidor y en el `.cnf` temporal, que se borra al terminar):

```
# deploy-db.sh (esquema del patrón, se ejecuta EN el servidor)
URL=$(node -e "const u=new URL(process.env.DATABASE_URL); \
  console.log([u.hostname,u.port,u.username,u.password,u.pathname.slice(1)].join(' '))")
# → generar un my.cnf temporal ([client] host/port/user/password) con umask 077
mysql --defaults-extra-file="$TMP_CNF" "$DB" < NNNN_x.sql > "$LOG" 2>&1
cp "$LOG" <app>/.next/static/<nombre-unico>.txt   # readback por HTTP
rm -f "$TMP_CNF"
```

2. **Crear la tarea** vía API de Plesk apuntando al script y ejecutarla inmediatamente (o con hora de disparo al minuto siguiente).
3. **Leer el resultado** («readback») por HTTP: el script escribe su log como fichero estático bajo `_next/static/` de una de las apps con un **nombre único** por ejecución (p. ej. sufijo aleatorio), que se descarga con el navegador o `curl` y se borra después.

```
Máquina local                    VPS Plesk (prod)
─────────────                    ──────────────────────────────
script.sh ──API REST──►  fichero en disco
(subida por Contents API,        │
 descargar por blob-SHA)         ▼
crear tarea ──API──►     scheduler: /bin/sh script.sh
(--command CORTO,                │  parsea DATABASE_URL (node)
 ≤980 chars, sin '%')            │  mysql --defaults-extra-file=….cnf < NNNN_x.sql
                                 ▼
curl https://…/_next/static/<nombre-único>.txt   ◄── log de resultado
```

Fig. 5.2 — Camino de respaldo sin SSH: API de Plesk + scheduler + readback por fichero estático.

> ⚠ **Tres límites que muerden.** (1) El campo `--command` de la tarea del scheduler tiene un límite práctico de ~**980 caracteres**; un inline largo hace que la tarea simplemente *no se cree* — el comando debe ser corto y delegar en un script subido a disco. (2) El carácter `%` es **metacarácter de cron** (inicia una nueva línea/stdin en la especificación de cron): cualquier `%` en el comando debe evitarse o escaparse, o el comando se trunca silenciosamente. (3) El readback debe usar un **nombre de fichero único por ejecución**: Cloudflare cachea las rutas (incluida la Contents API, que conviene consultar por *blob-SHA* y no por ruta), y reutilizar el mismo nombre puede devolver el log de una ejecución anterior y dar por buena una migración que falló.

Este método es de **respaldo**: más lento y con más fricción de verificación que el túnel SSH. Tras usarlo, la validación final es la misma de siempre: un run de `db:drift` (por túnel cuando vuelva a estar disponible, o un ensure-script de solo-lectura equivalente) que confirme «SIN DRIFT».

## 5.6 Verificación del esquema: estado a 2026-07-20

Salida real (forma) de un run limpio del verificador contra producción:

```
$ DATABASE_URL='mysql://…@127.0.0.1:3307/…' pnpm --filter @jbhlegal/db db:drift
Esquema: 140 tablas esperadas · BD: 1XX tablas reales

ℹ️  Tablas en BD sin definición en el esquema (…): __drizzle_migrations, …
✅ SIN DRIFT: todas las tablas y columnas del esquema existen en la BD.
$ echo $?
0
```

Las tablas «extra» del aviso informativo son esperables y no constituyen drift: la contabilidad histórica de drizzle (`__drizzle_migrations`) y restos legados que el esquema TypeScript ya no modela. El criterio de éxito es exclusivamente la línea «SIN DRIFT» y el código de salida 0.

> ✓ **SIN DRIFT (verificado 2026-07-20).** Tras destapar y aplicar la migración ausente `0084_audit_log` (el caso real del bug del migrator: figuraba «applied» y la tabla no existía), se ejecutó `db:drift` contra producción a través del túnel SSH con resultado limpio: **las 140 tablas del esquema Drizzle y todas sus columnas existen en la base de datos de producción**. Ese mismo día se completó la barrida de INSERTs sin `orgId` (7 sitios corregidos, §4.4.3), dejando esquema y código de escritura alineados con el modelo multi-tenant.

Disciplina operativa derivada, en tres invariantes:

- Toda migración se aplica con `db:apply` y se certifica con `db:drift`; «SIN DRIFT» es la única definición de «aplicada». El deploy normal de las apps **no ejecuta migraciones**: son un paso explícito y verificado, previo o coordinado con el despliegue del código que las necesita.
- El journal de Drizzle y la tabla `__drizzle_migrations` se consideran artefactos históricos sin autoridad: nadie decide nada leyéndolos.
- `db:drift` sale con código 1 ante cualquier ausencia, por lo que puede integrarse como gate en CI o en el script de deploy para impedir desplegar código contra un esquema incompleto.

# 6

# Autenticación, roles y control de acceso

> *Este capítulo documenta el sistema completo de identidad y autorización de la plataforma JBH Legal: la autenticación sin contraseña por código de un solo uso (OTP), el modelo de sesiones de base de datos, la matriz de siete roles globales más los roles de membership de despacho, los gates de acceso a recursos (`requireCaseAccess`, `requireIntakeAccess`, `requireClientSession`, `requireOrgManager`, `requireCrmContext`), el aislamiento multi-tenant verificado por tests y auditoría, el superadmin con impersonación de solo lectura, el gating de escritura del Modelo B, la política de MFA y las defensas anti-abuso (anti-enumeración, rate limiting, Turnstile y cabeceras de seguridad).*

## 6.1 Arquitectura de identidad: visión general

La identidad es compartida por todas las aplicaciones del monorepo a través del paquete `packages/auth` (`@jbhlegal/auth`), que centraliza la configuración de Auth.js (NextAuth v5), el adaptador de base de datos, el catálogo de roles, los guards de sesión y la implementación TOTP. Cada aplicación (portal del abogado, área cliente, peritos, firmas) instancia Auth.js con esta configuración común y añade sus propios helpers de autorización en `src/lib`.

El control de acceso está construido en **capas concéntricas**, de forma que ningún control depende de un único punto:

| Capa | Dónde vive | Qué decide |
|---|---|---|
| 1. Autenticación | `packages/auth/src/config.ts` (Auth.js, provider email OTP) | Quién es la persona; sin cuenta previa no hay sesión (no hay auto-registro). |
| 2. Middleware de app | `apps/portal/src/middleware.ts`, `apps/cliente/src/middleware.ts` | Entrada a la aplicación por conjunto de roles; rate limit del OTP; bloqueo demo; MFA; cabeceras de seguridad en toda respuesta. |
| 3. Guard de ruta | `requireRole` / `requireSession` (`apps/*/src/lib/auth-helpers.ts`) | Qué roles pueden invocar cada handler concreto. |
| 4. Gate de recurso | `requireCaseAccess`, `requireIntakeAccess`, etc. | Si ESTA sesión puede tocar ESTE expediente/lead: tenant (`org_id`), titularidad, co-letrados, soft-delete. |
| 5. Guard de escritura | `assertOrgCanWrite` (`apps/portal/src/lib/org-billing.ts`) | Si la escritura está permitida: impersonación de solo lectura, verificación de colegiado, eje de pago (suscripción / primer caso gratis). |
| 6. Predicado SQL | Filtro `eq(tabla.orgId, orgId)` en la consulta | Defensa en profundidad: la fila de otro tenant no se resuelve aunque las capas anteriores fallaran. |

## 6.2 Autenticación sin contraseña: OTP por email

### 6.2.1 El mecanismo

Las tres superficies de acceso (portal, área cliente, peritos) usan el **mismo mecanismo passwordless**: un código de un solo uso de 6 dígitos que el usuario recibe por email y teclea en el formulario de verificación. No se envía ningún enlace clicable de verificación. La decisión está documentada en el propio código (`packages/auth/src/config.ts` y `otp.ts`): los escáneres de correo y Safe Browsing pre-visitan los enlaces de un solo uso y consumían el token antes de que el usuario hiciera clic (error `Verification`); un código tecleado no tiene nada que pre-visitar.

La implementación se apoya en el provider `email` de Auth.js con dos overrides:

```
// packages/auth/src/config.ts
generateVerificationToken: async () => randomInt(100000, 1000000).toString(),
sendVerificationRequest: async ({ identifier, token }) => {
  await sendClientOtp({ email: identifier, code: token });
},
```

- El token de verificación *es* el código de 6 dígitos, generado con `crypto.randomInt` (CSPRNG). Auth.js lo almacena **hasheado con** `AUTH_SECRET` tanto al guardarlo como al verificarlo, por lo que la tabla `verification_tokens` nunca contiene el código en claro.
- Caducidad: `CLIENT_OTP_TTL_MIN = 30` minutos (constante en `packages/auth/src/otp.ts`), tiempo suficiente para cambiar a la app de correo y volver.
- Un solo uso: el token se consume al verificarse; un segundo intento con el mismo código produce `error=Verification`.
- Remitente por defecto: `despacho@jbhasesorialegal.com` (env `EMAIL_FROM`).

En el Modelo B, el email del OTP se **brandea con la marca del despacho** del cliente: `sendClientOtp` resuelve `resolveClientFirmBrand(email)` (best-effort, nunca lanza) y, si el cliente pertenece a un único despacho con nombre comercial, la plantilla usa esa marca; para JBH/Modelo A se usa la marca neutra.

### 6.2.2 Sesiones y cookies

Las sesiones usan la estrategia `database` de Auth.js: la fila de la tabla `sessions` es la fuente de verdad y la cookie solo transporta el token de sesión opaco. La vida de la sesión es una **ventana de inactividad deslizante**: la columna `expires` se empuja hacia delante con la actividad, de modo que la sesión muere tras `maxAge` sin uso.

| Parámetro | Valor por defecto | Env de override | Notas |
|---|---|---|---|
| `session.maxAge` | 14 días | `SESSION_MAX_AGE_DAYS` | Reducido de 30 a 14 días por auditoría (M10): un portal con material de defensa penal no debe mantener sesiones de un mes. |
| `session.updateAge` | 24 horas | `SESSION_UPDATE_AGE_HOURS` | El `expires` se refresca como pronto una vez al día. |
| TTL del OTP | 30 min | — (constante) | `CLIENT_OTP_TTL_MIN` en `packages/auth/src/otp.ts`. |

La lectura del token de sesión desde cookies está centralizada en `packages/auth/src/session-token.ts`, que prueba los nombres de cookie **en orden estricto de preferencia**:

```
1. __Secure-authjs.session-token      (Auth.js v5, HTTPS — la cookie viva canónica)
2. authjs.session-token               (Auth.js v5, solo dev local sin HTTPS)
3. __Secure-next-auth.session-token   (legado v4, solo como último recurso)
4. next-auth.session-token            (legado v4)
```

⚠ **El orden importa (seguridad).** Tras la migración de NextAuth v4 a v5 pueden coexistir una cookie legada muerta y la cookie v5 viva. Si se leyera primero el nombre legado, el flag MFA por sesión (`sessions.mfa_verified_at`, §6.9) se anclaría a una fila de sesión distinta de la que el middleware está exigiendo, rompiendo la verificación o, peor, validándola contra la fila equivocada. Leer primero `__Secure-authjs.session-token` garantiza que enforcement y verificación operan sobre la misma fila viva. Un token presente tampoco es garantía de sesión válida: el caller debe resolverlo contra `sessions`.

#### 6.2.3 El callback `signIn`: sin auto-registro y bloqueo de anonimizados

El callback `signIn` de `authConfig` impone dos reglas antes de emitir sesión:

- **Sin auto-registro:** solo inicia sesión un email que *ya exista* en `users`. Pedir un código para un email desconocido termina en `AccessDenied`, que el área cliente presenta con mensaje neutro (§6.10.1). Las altas se hacen por invitación (cliente), registro de despacho (Modelo B) u onboarding de colaborador — nunca desde el login.
- **RGPD:** un usuario con rol `ANONYMIZED` (estado terminal tras una supresión DSR) no puede iniciar sesión jamás, y la resolución multi-rol (§6.3.1) tampoco lo cuenta nunca como rol de acceso.

El callback `session` proyecta sobre la sesión: `user.id`, el rol canónico `user.role`, el conjunto completo `user.roles` (multi-rol), `user.twoFactorEnabled` y `user.isDemo` (normalizado de forma robusta porque el adaptador puede devolver el `tinyint` como `1` / `'1'`).

## 6.3 Modelo de roles

#### 6.3.1 Rol canónico y multi-rol aditivo

El catálogo global vive en `packages/auth/src/types.ts`: `ROLES = ['ADMIN', 'COORDINADOR', 'COLABORADOR', 'CLIENT', 'PERITO', 'ANONYMIZED', 'LAWYER']`. El modelo distingue dos conceptos:

- `users.role` **— rol canónico** (un único valor). Todo el scoping de tenant y la lógica de autorización por recurso se derivan de él.
- `user_roles` **— roles extra aditivos.** Una misma persona puede ser a la vez cliente, perito y abogado con el mismo email. El conjunto efectivo de acceso es la unión `{users.role} ∪ user_roles`, resuelto por `resolveAccessRolesForUser` / `resolveUserAccessRoles` (`packages/auth/src/roles.ts`) y proyectado en `session.user.roles`.

ℹ **Regla de uso.** El conjunto `roles` se usa *solo* para las puertas de **entrada** a cada app (`hasAnyRole` en los middlewares): quien sea staff o LAWYER en cualquiera de sus roles entra. El **scoping de tenant y los gates de recurso siguen usando el rol canónico** (`session.user.role`); derivar scoping del conjunto multi-rol está expresamente prohibido en el código. `ANONYMIZED` nunca forma parte del conjunto ni es concedible como rol extra (`GRANTABLE_ROLES` lo excluye), y la concesión/retirada (`grantRole` / `revokeRole`) es idempotente con UNIQUE `(user_id, role)` y queda auditada en `superadmin_audit` (`role_grant` / `role_revoke`).

#### 6.3.2 Matriz de roles globales

| Rol | Quién es | Portal abogado | Área cliente | Alcance sobre expedientes / leads | Módulos de plataforma (`/portal/admin`, `/api/admin`) |
|---|---|---|---|---|---|
| `ADMIN` | Staff de plataforma JBH (dirección) | Sí | Sí (como cliente de sus propios casos) | Todos los expedientes e intakes del tenant JBH; gestiona equipos de caso. | Sí (completo). El superadmin (§6.7) es además un `ADMIN` allowlisted. |
| `COORDINADOR` | Staff JBH de coordinación/triaje | Sí | Sí (ídem) | Todos los expedientes e intakes del tenant JBH; gestiona equipos de caso. | Sí (el middleware admite `ADMIN` + `COORDINADOR`; rutas concretas estrechan a `ADMIN`). |
| `COLABORADOR` | Abogado colaborador del despacho JBH (Modelo A) | Sí | Sí (ídem) | SOLO los casos donde es `leadLawyerId` o co-letrado (`case_collaborators`); SOLO los intakes donde es `assignedToUserId` (los sin asignar le son invisibles). | No (403 en middleware). |

| Rol | Quién es | Portal abogado | Área cliente | Alcance sobre expedientes / leads | Módulos de plataforma (`/portal/admin`, `/api/admin`) |
|---|---|---|---|---|---|
| `LAWYER` | Abogado-suscriptor de un despacho externo (Modelo B) | Sí, scopeado a SU organización | Sí (como cliente) | Dentro de su tenant: el titular (membership OWNER/FIRM_ADMIN) supervisa todo; el abogado no-titular solo sus casos (lead o co-letrado). Nunca resuelve al tenant JBH. | No: `/portal/admin` y `/api/admin` vetados; tampoco analítica global, marketing, leads JBH, coste de IA ni DSR global. |
| `CLIENT` | Cliente (investigado/encausado o su familia) | No (403/redirect) | Sí | Solo los casos donde `cases.clientUserId === session.user.id`; solo documentos con `visibility='shared'`. | No. |
| `PERITO` | Perito del directorio | No | Sí (app peritos / área cliente para sus propios asuntos) | Encargos periciales que le conciernen; sus informes quedan bloqueados a descarga hasta el pago del peritaje (cap. 7). | No. |
| `ANONYMIZED` | Usuario suprimido vía DSR (art. 17 RGPD) | Ningún acceso: `signIn` lo rechaza, nunca entra en el conjunto multi-rol, no es concedible. Estado terminal. | | | |

Los agregados de conveniencia del paquete auth: `PORTAL_ROLES` (staff JBH), `PORTAL_ACCESS_ROLES` (staff + `LAWYER`: quién entra al portal), `CLIENT_AREA_ROLES` (todos los roles reales salvo `LAWYER`… en concreto `['ADMIN','COORDINADOR','COLABORADOR','PERITO','CLIENT']`) y `GRANTABLE_ROLES` (todos menos `ANONYMIZED`).

#### 6.3.3 Roles de membership (Modelo B) y `ActiveContext`

Dentro de un despacho externo, el rol global de *todo* miembro es `LAWYER`; la distinción titular/abogado vive en `memberships.role`: `OWNER`, `FIRM_ADMIN` (gestores del despacho, `FIRM_MANAGER_ROLES`), `LAWYER`, `STAFF`, `COLLABORATOR`, `PERITO`. El helper `getActiveContext(session)` (`apps/portal/src/lib/active-org.ts`) resuelve por petición (cacheado con `react.cache`):

```
interface ActiveContext {
  orgId: string;                 // tenant activo
  orgType: 'jbh_platform' | 'law_firm';
  membershipRole: string | null; // OWNER/FIRM_ADMIN/LAWYER/STAFF/... o null (staff JBH)
  plan: OrgPlan | null;          // de organization_subscriptions
  isDespachoExperience: boolean; // plan multi-asiento (PLAN_META.seatsIncluded !== 1)
  isFirmManager: boolean;        // membership OWNER/FIRM_ADMIN
  impersonating: boolean;        // superadmin "ver como" (§6.7)
  readOnly: boolean;             // siempre que impersonating
}
```

La resolución del tenant (`getActiveOrgIdForSession`) sigue este orden: (1) org impersonada por el superadmin si la cookie firmada valida; (2) staff JBH → `JBH_ORG_ID` sin tocar BD; (3) `LAWYER` → su primera membership activa. Contiene una regla de seguridad crítica:

> ⚠ **Un LAWYER nunca cae a JBH.** Si un `LAWYER` no resuelve membership activa (datos inconsistentes), `active-org.ts` **lanza** en vez de degradar al tenant por defecto: caer a `JBH_ORG_ID` filtraría el tenant de la plataforma (y sus datos penales) a un abogado externo. El fallback a `JBH_ORG_ID` existe solo para el staff de plataforma.

## 6.4 El middleware del portal: defensa por capas en cada petición

`apps/portal/src/middleware.ts` (runtime `nodejs`, matcher sobre `/portal/*`, `/api/intakes/*`, `/api/admin/*`, `/api/auth/*`, `/api/crm/*`, `/api/cases/*`, `/api/causas/*`, `/api/me/*`) aplica, en orden:

1. **Rate limit del OTP** sobre las rutas internas de Auth.js — el envío (`/api/auth/signin/email`) es email-bombable y el código de 6 dígitos es fuerza-brutable, así que ambos se limitan a **12 peticiones / 10 min por IP y por email** (429 al exceder), espejo de los middlewares de cliente y peritos.
2. **Rutas públicas token-gated** (onboarding de colaborador, aceptación de invitación de miembro) pasan sin sesión: su seguridad es el token de un solo uso, no la cookie.
3. **Sin sesión** → 401 JSON en `/api/*` (con `reason: 'session_expired'`) o redirect a `/login?reason=session_expired` en páginas.
4. `/portal/admin` **y** `/api/admin` → solo `hasAnyRole(session, ['ADMIN','COORDINADOR'])`. (Una versión anterior eximía `GET /api/admin/users` y filtraba el directorio de usuarios a COLABORADOR — corregido en PULIDO P1-2.)
5. `/portal/admin/superadmin` → gate adicional estricto `isSuperadmin(session)`; la página revalida (doble gate).
6. **Entrada al portal** → `hasAnyRole(session, PORTAL_ACCESS_ROLES)`; un CLIENT/PERITO recibe redirección con motivo (`forbidden_role`) en páginas y 403 JSON en APIs.
7. **Cuenta demo** (`users.is_demo`) → toda mutación (`POST/PUT/PATCH/DELETE` sobre `/api/*`) responde 403 `read_only_demo`.
8. **MFA** → alta obligatoria y desafío de segundo factor (§6.9).
9. **Toda respuesta**, incluidas las de error, sale con `applySecurityHeaders` (§6.10.3) y un `x-request-id` de correlación.

## 6.5 Gates de acceso a recursos

### 6.5.1 `requireCaseAccess` — el gate de expediente

Toda ruta bajo `/api/cases/[id]` comienza con `requireCaseAccess(caseId)` (`apps/portal/src/lib/case-access.ts`). Sin él, cualquier COLABORADOR autenticado podría leer expedientes arbitrarios por UUID — el IDOR clásico que motivó el fix PULIDO P1-1. El flujo completo:


Fig. 6.1 — Flujo completo de `requireCaseAccess`.

Detalles que merecen nota:

- El gate devuelve una **copia superficial** de la fila: varias rutas capturan el estado pre-update (p. ej. la ruta de estado construye el evento de auditoría «de → a» leyendo `cur.status` después del UPDATE) y sin la copia un driver que devolviera la referencia del store la mutaría por debajo.
- `canManageCase` es la variante para gestionar el *equipo* del caso (añadir/quitar co-letrados): staff de oversight o el lead; un co-letrado tiene acceso pleno al caso pero **no** gestiona el equipo.
- Un expediente con `deleted_at` no nulo se trata como inexistente: toda ruta case-scoped responde 404.

### 6.5.2 `requireIntakeAccess` — el gate de lead

`apps/portal/src/lib/intake-access.ts` replica el patrón para `intakes`: `requireRole` → `getActiveContext` → consulta con `eq(intakes.orgId, orgId)` → `canAccessIntake`. La regla de titularidad: ADMIN/COORDINADOR (y el titular de despacho) ven todo; COLABORADOR (o abogado no-titular) solo los intakes donde es `assignedToUserId` — **los intakes sin asignar (`assignedToUserId = null`) nunca son accesibles para un COLABORADOR**. Al no-dueño que pasó el gate de rol se le responde `NotFoundError`, no 403 (§6.5.5). También ejecuta `enterOrgAiContext(orgId)` al final.

### 6.5.3 `requireClientSession` — el gate del área cliente

`apps/cliente/src/lib/auth-helpers.ts` admite `CLIENT_AREA_ROLES` (cualquier rol real: un abogado o admin también puede ser cliente). Esto es seguro porque **ninguna ruta del área cliente confía en el rol**: cada consulta se acota además a los datos propios del usuario (`cases.clientUserId === session.user.id`), de modo que un no-CLIENT solo ve los casos en los que él mismo es el cliente. El middleware del área cliente añade el mismo bloqueo de cuentas demo y las cabeceras de seguridad (matcher: `/cliente/*`, `/api/my/*`, `/api/auth/*`).

### 6.5.4 `requireOrgManager` y `requireCrmContext`

- `requireOrgManager(session, orgId)` (`apps/portal/src/lib/marketplace.ts`): exige membership activa `OWNER`/`FIRM_ADMIN` en la org (reutiliza `canManageOrgBilling`, que consulta `memberships` con `status='active'`); si no, `ForbiddenError(['LAWYER'])`. Protege la gestión del perfil de marketplace, las respuestas a propuestas de casos externos y, vía `ORG_BILLING_MANAGER_ROLES`, toda la gestión de pago del despacho. También lo consumen `org-team.ts`, `org-public-profile.ts`, `org-branding.ts` y `org-billing-profile.ts`.
- `requireCrmContext()` (`apps/portal/src/lib/crm/access.ts`): contexto común de las rutas CRM. Hace `requireRole(['ADMIN','COORDINADOR','COLABORADOR','LAWYER'])` + `getActiveContext` y calcula `restrictToOwn = (rol === 'COLABORADOR') ||`

`(law_firm && !isFirmManager)` : el mismo criterio de visibilidad que la bandeja de intakes, aplicado como filtro adicional en las consultas del pipeline (defensa en profundidad junto a `requireIntakeAccess`).

#### 6.5.5 Regla «404 antes que 403»

> ✓ **No filtrar existencia.** Cuando un recurso no se resuelve para el tenant/dueño actual, los gates responden **404**, no 403: un 403 confirmaría a un tercero que el UUID existe (metadato sensible cuando la mera existencia de un expediente penal contra una persona es en sí dato de categoría especial). El 403 queda reservado a los fallos de *rol*, que se evalúan *antes* de tocar la base de datos, por lo que no revelan nada sobre ningún recurso concreto. El área cliente aplica la misma regla incluso a la visibilidad de documentos: un documento `internal` responde 404 al cliente, no 403.

## 6.6 Aislamiento multi-tenant verificado

El scoping por `org_id` abarca ~59 tablas tenant-owned. Su verificación no descansa en la revisión manual:

- **Test de aislamiento cross-org** (`apps/portal/src/__tests__/case-access-cross-org.test.ts`): en lugar de un mock que devolvería la fila ignorando el WHERE (y no probaría nada), el doble de BD **renderiza el predicado real de Drizzle con** `MySqlDialect().sqlToQuery()` y solo devuelve el caso sembrado si el `id` y el `org_id` figuran entre los parámetros del predicado — exactamente lo que haría MariaDB con el AND. Aserciones: (1) la consulta incluye el predicado `org_id = <org activo>`; (2) un letrado del despacho A que pide por id un caso del despacho B recibe 404 (no 403, no el caso ajeno); (3) el mismo caso en SU despacho sí se resuelve. Si alguien eliminara el `eq(cases.orgId, orgId)` del gate, el test se pone en rojo. Lo complementan `case-access.test.ts` (matriz rol × propiedad) y `packages/db` `tenant-isolation.test.ts` (capa `withTenant`).
- **Barrido de rutas (auditoría 2026-07-20):** revisión sistemática de las **335 rutas API** del producto (**268** en `apps/portal/src/app/api` + **67** en `apps/cliente/src/app/api`) comprobando que toda ruta scoped a `cases/[id]` / `intakes/[id]` pasa por su gate y que las consultas llevan el filtro de tenant o de dueño. Resultado: **LIMPIO** — sin rutas sin gate ni IDOR detectados.

> ⚠ **Clase de bug conocida (gating Modelo B).** El error recurrente en este código no es olvidar el rol, sino añadir `LAWYER` a un `requireRole` de ruta *sin* pasar por `requireCaseAccess` / `requireIntakeAccess`: el rol solo abre la puerta de la app; sin el gate de recurso el LAWYER alcanzaría recursos de otros tenants por UUID (IDOR). Toda ruta nueva scoped a un recurso debe invocar su gate.

## 6.7 Superadmin e impersonación «ver como» (solo lectura)

El superadmin (`apps/portal/src/lib/superadmin.ts`) es un rol *operativo*, no un rol de BD: un operador de plataforma capaz de ver el directorio de despachos con métricas globales y de «ver como» cualquier despacho para diagnosticar errores navegando el portal exactamente como lo ve ese tenant. Como es acceso cross-tenant a datos penales, se implementa con el máximo de candados:

| Candado | Implementación |
|---|---|
| Identidad estricta | `isSuperadmin(session)`: rol `ADMIN` **y** email ∈ allowlist `SUPERADMIN_EMAILS` (case-insensitive; valor out-of-band). Otros ADMIN no allowlisted → false. |
| Cookie firmada | `jbh_su_view` (`SU_VIEW_COOKIE`): httpOnly + secure + SameSite=Lax, valor `base64url({orgId}) . base64url(HMAC-SHA256(payload, AUTH_SECRET))`, comparación en tiempo constante (`timingSafeEqual`), caducidad 8 h. Manipularla no concede nada: `getImpersonatedOrgId` revalida `isSuperadmin` en cada request y para cualquier otro usuario ignora la cookie. |
| Solo lectura | `getActiveContext` marca el contexto `impersonating: true, readOnly: true` (y lo presenta como titular: `membershipRole='OWNER'`, `isFirmManager=true`, para que el portal entero se renderice como ese despacho sin tocar las queries). El guard central `assertNotImpersonating()` — invocado como paso 0 de `assertOrgCanWrite` — lanza `ReadOnlyImpersonationError` → 403 `read_only_impersonation` en cualquier escritura. |
| Auditoría RGPD | Cada inicio/fin («`view_as_start`»/«`view_as_stop`») y las demás acciones transversales (`console_view`, `role_grant`, `set_custom_plan`, acciones de referidos…) se insertan en `superadmin_audit` con actor, org objetivo y **hash SHA-256 de la IP** (nunca la IP en claro). La auditoría nunca lanza: un fallo del insert no tumba la acción, se loguea. |
| Doble gate de ruta | El middleware bloquea `/portal/admin/superadmin` a no-superadmins y la página revalida. |

## 6.8 Gating de escritura del Modelo B: `assertOrgCanWrite`

Toda escritura del despacho (crear expediente, subir documento, invitar cliente, lanzar IA…) pasa por `assertOrgCanWrite(orgId, intent)` (`apps/portal/src/lib/org-billing.ts`), con `intent ∈ {'create','work'}` (por defecto `'work'`; las rutas que crean un expediente pasan `'create'`). Orden de evaluación:

```
assertOrgCanWrite(orgId, intent)
  │
  ├─ 0. assertNotImpersonating()            impersonación superadmin
  │       └─ impersonando → 403 ReadOnlyImpersonationError
  │
  ├─ 1. (Colegiado — ver nota 2026-07-17)   ya NO se asserta aquí
  │
  └─ 2. guardOrgWrite(orgId, intent)        eje de PAGO
        │
        ├─ isPlatformOrg(orgId) (JBH)          → ok  (Modelo A NUNCA se bloquea)
        ├─ gating.canWrite (active/trialing)   → ok
        ├─ status canceled | past_due          → 402 reason='read_only'
        ├─ intent='work'  (plan free/none)     → ok  (el caso gratis se TRABAJA completo)
        └─ intent='create' (plan free/none):
             ├─ countActiveExpedientes ≥ 1     → 402 reason='first_case_used'
             ├─ colegiadoFreeCaseAlreadyUsed   → 402 reason='first_case_used'
             │    (capa 4: un caso gratis por NIF, en CUALQUIER despacho)
             └─ si no                          → ok  (primer caso gratis)
```

Fig. 6.2 — Árbol de decisión del guard de escritura del despacho.

- **Modo lectura:** con la suscripción vencida (`canceled`, o `past_due` — incluso dentro de gracia, donde `canWrite=false`) el despacho conserva la lectura y la exportación de sus datos, pero ni crea ni trabaja; el mensaje de UI lo produce `billingBlockMessage(reason)`.
- **Primer caso gratis completo:** con `intent='work'` el plan free siempre permite trabajar (invitar cliente, IA, escritos, tareas…), de modo que el primer caso es un trial real de principio a fin; solo crear el 2.º expediente exige suscripción.
- **Capa 4 anti-autotrato:** el caso gratis se cuenta por *persona* (hash del NIF del colegiado en `colegiado_identities.free_case_used_at`, indexado por `nif_hash`), no por org: abrir varios despachos con el mismo NIF no regala varios casos. `markColegiadoFreeCaseUsed` lo marca de forma idempotente y atómica (UPDATE con `WHERE free_case_used_at IS NULL`).
- **Verificación de colegiado ASISTIDA (decisión de conversión, 2026-07-17):** el primer caso se puede crear y trabajar *sin* verificar el certificado ACA — el abogado prueba el producto desde el primer minuto con un banner persistente recordándole completar el registro. La verificación (`assertOrgColegiadoVerified`, que lanza `ColegiadoVerificationRequiredError` → 403 si el `law_firm` no tiene `colegiado_verified_at`) sigue siendo obligatoria para **escalar**: la mantienen las rutas de `checkout` y `change-plan`, y por tanto condiciona del 2.º expediente en adelante. JBH (`jbh_platform`) está exento de todas las capas.
- **Planes a medida:** `assertExpedienteWithinCustomLimit` añade, solo para orgs `is_custom` con `expedientes_override`, un límite duro de expedientes (402 `expedientes_limit`).

## 6.9 MFA: TOTP opcional hoy, obligatorio por flag

El segundo factor es TOTP estándar (RFC 6238: SHA-1, paso de 30 s, 6 dígitos — compatible con Google Authenticator, Authy, 1Password…), implementado sin dependencias sobre `node:crypto` en `packages/auth/src/totp.ts`, con secreto base32 de 160 bits generado por `generateTotpSecret`. El secreto compartido se guarda **cifrado en reposo con AES-256-GCM** (`totp-crypto.ts`, clave derivada de `AUTH_SECRET`; formato `enc:`, con re-cifrado transparente de valores legados en la siguiente escritura — auditoría M1).

La aplicación tiene tres piezas:

1. **Enforcement por sesión:** el flag «esta sesión pasó el 2FA» vive en la fila (`sessions.mfa_verified_at`). Con `MFA_ENFORCED=true`, un usuario con 2FA activada cuya sesión actual no lo ha satisfecho es redirigido a `/login/2fa` por el middleware, y — cierre de la auditoría A2 — *todos* los guards `requireRole`/`requireSession` del portal lo comprueban también (`assertMfaSatisfied` en `apps/portal/src/lib/auth-helpers.ts`, que lanza `MfaRequiredError`), porque el matcher del middleware no cubre todas las rutas `/api/**`. Los endpoints de alta/desafío pasan `{ mfaOptional: true }` para que un usuario aún no verificado pueda alcanzarlos.
2. **Alta obligatoria por rol (anti-lockout):** `apps/portal/src/lib/mfa-policy.ts` define `DEFAULT_MFA_MANDATORY_ROLES = ['ADMIN','COORDINADOR','COLABORADOR','LAWYER']` (todos los roles del portal manejan expedientes penales), ampliable/restringible sin desplegar vía env `MFA_MANDATORY_ROLES`. La función pura `mfaEnrollmentDecision` devuelve `'allow'` / `'redirect'` (páginas → `/portal/perfil?mfa=setup`) / `'block-api'` (APIs → 403 `mfa_enrollment_required`). `/portal/perfil` y `/api/me/*` quedan siempre exentos: nadie puede quedar encerrado sin poder darse de alta.
3. **Kill-switch:** todo cuelga del flag `MFA_ENFORCED`, **apagado hoy en producción**. Mientras esté apagado, ambas piezas son inertes (cambio seguro de mergear); encenderlo también es seguro precisamente por el diseño anti-lockout.

## 6.10 Defensas anti-abuso

### 6.10.1 Anti-enumeración en el login del cliente

Los clientes del área cliente están ligados a causas penales: confirmar que un email «es cliente» ya revela un dato de categoría especial. Por eso el login (`apps/cliente/src/app/cliente/login/page.tsx`) responde al `AccessDenied` del callback `signIn` (email inexistente) con un **mensaje neutro idéntico al del éxito implícito**: «Si tu email está registrado, te hemos enviado un código de acceso…», con CTA a comprobar el email de alta, escribir a `despacho@jbhasesorialegal.com` o evaluar el caso en la web pública. No se redirige al registro (eso confirmaría que la cuenta no existe) ni se distingue en tiempo o texto entre cuenta existente e inexistente. El error `Verification` (código inválido/caducado) sí se comunica, porque en ese punto el usuario ya demostró conocer un email registrado… y ni siquiera: el mismo texto se muestra a cualquiera que teclee mal el código.

### 6.10.2 Rate limiting y Turnstile

- `consume({ key, max, windowMs })` (`lib/rate-limit.ts` en portal y cliente): limitador in-memory por buckets de ventana fija que lanza `RateLimitError` (→ 429 con `Retry-After`) al exceder. Ejemplos reales: OTP 12/10 min por IP y por email (middlewares); descargas de documentos 200/h por usuario en

portal y 60/h en cliente; subidas de onboarding 30/h por token.

- **Cloudflare Turnstile** en los formularios públicos (registro de despacho, registro profesional, contacto, intakes públicos de la web): verificación server-side (`verifyTurnstileDetailed`) contra `challenges.cloudflare.com/turnstile/v0/siteverify`, con `TURNSTILE_SECRET_KEY` obligatoria en producción (lanzarse sin ella es un error; en dev sin secret hay bypass). La función `isBenignTurnstileFailure(errorCodes)` distingue el fallo *benigno* de un usuario real (token caducado o reutilizado — `timeout-or-duplicate` —, errores de transporte/CDN `network-error` / `http-*`, `internal-error` de Cloudflare), que se deja pasar apoyándose en el rate limit por IP como guardia dura, del token genuinamente inválido/forjado (`invalid-input-response`, `bad-request`, `missing-input-*`), que se bloquea como señal de bot.

### 6.10.3 Cabeceras de seguridad

Cada app define su `src/lib/security-headers.ts` con la misma base y allowances específicas; el middleware las aplica a **todas** las respuestas, incluidas las de error:

| Cabecera | Valor (portal / cliente) |
|---|---|
| `Content-Security-Policy` | `default-src 'self'` + allowances por app: portal → `js.stripe.com` (Checkout), `accounts.google.com` / `apis.google.com` (Google Calendar de letrados), Sentry, y `data:` / `blob:` en `connect-src` para el WASM de `@react-pdf`; cliente → Jitsi/8x8 (`meet.jit.si`, `*.8x8.vc` incl. `wss:`) para videoconsultas, Sentry, y `https://*.r2.cloudflarestorage.com` en `connect-src` (sin él, el navegador bloquea el PUT prefirmado de la subida multiparte). Ambas: `frame-ancestors 'none'`, `object-src 'none'`, `form-action 'self'`, `base-uri 'self'`, `upgrade-insecure-requests`. |
| `Strict-Transport-Security` | `max-age=63072000; includeSubDomains; preload` (2 años). |
| `Referrer-Policy` | `strict-origin-when-cross-origin`. |
| `X-Content-Type-Options` | `nosniff`. |
| `X-Frame-Options` | `DENY` (redundante con `frame-ancestors`, para agentes antiguos). |
| `Permissions-Policy` | `camera=(self), microphone=(self), geolocation=(), interest-cohort=()`. |

> ✓ **Resumen de garantías del capítulo.** Sin cuenta previa no hay sesión; el código OTP viaja hasheado y caduca en 30 min; la sesión muere a los 14 días de inactividad; toda ruta de recurso pasa por un gate que filtra por tenant y titularidad con 404-antes-que-403; el aislamiento cross-org está cubierto por un test que renderiza el SQL real y por un barrido limpio de las 335 rutas API (auditoría 2026-07-20); el superadmin solo impersona en lectura, con cookie firmada HMAC y auditoría; y las escrituras del Modelo B están gobernadas por un guard único de cuatro capas.

# 7
# Seguridad documental: subida, antivirus, almacenamiento y descarga

> *Los documentos de un expediente penal (atestados, autos, pruebas, DNI, poderes) son datos de categoría especial (arts. 9/10 RGPD). Este capítulo documenta la cadena completa de custodia digital de esos ficheros: la subida multiparte con URLs prefirmadas y doble validación de tipo, la cuarentena con escaneo antivirus ClamAV y política fail-closed en producción, el almacenamiento en Cloudflare R2 con cifrado en reposo, la descarga exclusivamente por URL prefirmada de vida corta, y el ciclo de vida de retención (soft-delete, borrado físico diferido y purga de la PII derivada de los análisis IA).*

## 7.1 Principios de diseño

- **Los buckets son privados, siempre.** Ningún objeto es alcanzable por URL pública; toda lectura pasa por una URL prefirmada de vida corta acuñada en el servidor *después* de autenticación + autorización por caso (`infra/scripts/setup-r2-buckets.md` exige «Public access = Not allowed», sin URL `*.r2.dev` ni dominio público).
- **Cuarentena por defecto.** Todo fichero entra en un bucket de cuarentena y solo se promociona al bucket de producción tras el veredicto del antivirus (o la degradación controlada documentada en §7.5).
- **Fail-closed en producción.** Sin antivirus operativo (`AV_ENABLED` ≠ `true`) un documento subido en producción queda retenido en cuarentena, no se promociona ni se analiza con IA.
- **Doble validación de tipo.** Extensión + MIME declarado en la iniciación; firma binaria (magic bytes) y denylist de ejecutables sobre los bytes reales.
- **El servidor no toca el contenido en el camino caliente.** Las subidas grandes van directas navegador→R2 (PUT prefirmado por partes) y las descargas son un 302 a R2: el Node del portal ni proxya ni bufferiza gigabytes.
- **Trazabilidad completa.** Iniciación, finalización, aborto, infección y descarga de cada documento quedan en la auditoría del caso (`logCaseEvent`), incluyendo IP y user-agent en las descargas.

## 7.2 Modelo de datos y máquina de estados

La tabla central es `case_documents` (`packages/db/src/schema/case-documents.ts`); `intake_documents` y `herencia_documents` siguen el mismo patrón para leads y herencias. Columnas relevantes para la seguridad:

| Columna | Tipo | Función |
|---|---|---|
| `id` | varchar(36) UUID | Identificador; forma parte de la clave R2. |
| `case_id` / `org_id` | varchar(36) | Scoping por expediente y por tenant (`org_id` NOT NULL, default `JBH_ORG_ID`, índice `idx_case_documents_org`). En la subida se estampa el `orgId` *del caso*, no el default — en Modelo B, usar el default dejaría el documento invisible para el despacho. |
| `uploader_id` / `uploader_role` | varchar | Quién subió y con qué rol (p. ej. gating de informes de PERITO, §7.7). |
| `kind` | varchar(40) | `general` \| `hoja_encargo` \| `dni` \| `poder` \| `prueba` \| `escrito` \| `propuesta` \| `informe` \| `factura` \| `otro`. |
| `visibility` | varchar(20) | `internal` (solo portal) \| `shared` (visible al cliente). El área cliente responde 404 —no 403— ante un documento `internal`. |
| `storage_provider` / `storage_path` / `r2_etag` | varchar | `r2` (actual) o `local` (legado filesystem). `storage_path` = clave del objeto; `r2_etag` no nulo = el objeto existe físicamente. |
| `size_bytes` / `mime_type` / `original_filename` | bigint / varchar | Metadatos validados; el nombre se sanea (`sanitizeFilename`: se corta a 500, se eliminan separadores de ruta y caracteres fuera de una whitelist). |
| `upload_status` / `upload_session_id` / `upload_completed_at` | varchar(20) / varchar(600) / datetime | Máquina de estados (abajo) y UploadId multiparte de R2 (600 caracteres: los UploadId de R2 superan los 300 y con 100 fallaba el INSERT «Data too long»). |
| `scan_verdict` / `scan_message` / `scan_completed_at` | varchar(20) / text / datetime | Resultado AV: `clean` \| `infected` \| `error` \| `pending` \| `skipped` \| `unscanned` \| `rejected`. |
| `deleted_at` / `deleted_by_id` / `retention_until` | datetime / varchar(36) | Soft-delete y fecha límite de retención (índices dedicados `idx_case_documents_upload_status` y `idx_case_documents_retention_until`). |
| `content_hash` | varchar(64) | SHA-256 del binario, para deduplicar análisis IA *dentro del mismo despacho* (si un documento idéntico de la misma org ya tiene análisis `done`, se reutiliza). |

Estados de `upload_status` observados en el código:

| Estado | Significado | Dónde está el binario |
|---|---|---|
| `pending` | Fila creada, subida aún no iniciada (default de `intake_documents`). | En ningún sitio. |
| `uploading` | Multiparte iniciada; el navegador está haciendo PUT de partes. | Partes parciales en cuarentena. |
| `scanning` | Subida completada; pendiente de veredicto AV (o retenido fail-closed, §7.4). | Bucket de cuarentena. |
| `ready` | Promocionado; descargable y analizable por IA. | Bucket de producción. |
| `completed` | Legado: subida directa antigua al filesystem; tan definitiva como `ready`. | Filesystem del servidor (`storage_provider='local'`). |
| `infected` | Malware detectado (o firma de ejecutable): binario eliminado, no descargable. | Borrado. |
| `failed` | Subida abortada por el usuario o fallida. | Multiparte abortada en cuarentena. |

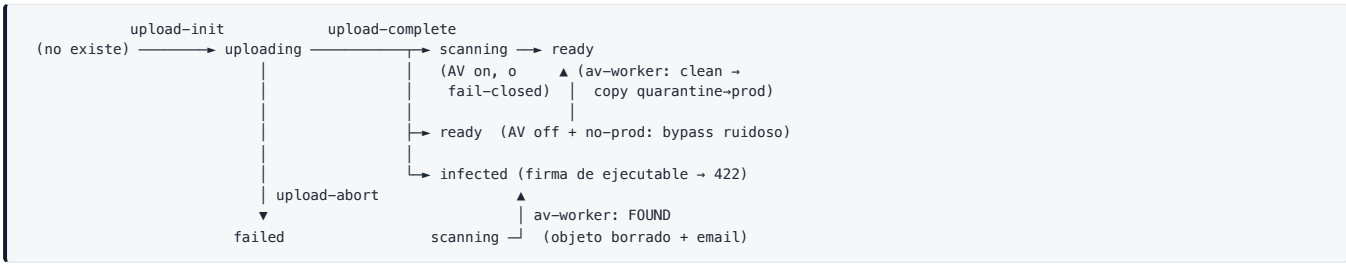


Fig. 7.1 — Máquina de estados de `upload_status`.

### 7.2.1 Adaptadores de almacenamiento y configuración

El paquete `packages/storage` expone dos contratos que las apps consumen sin conocer el backend:

- `MultipartAdapter` (implementación `r2-adapter.ts` sobre `@aws-sdk/client-s3` con SigV4 y `forcePathStyle`): `createMultipartUpload`, `getPresignedPartUrls`, `completeMultipartUpload`, `abortMultipartUpload`, `getPresignedDownloadUrl`, `copyObject`, `deleteObject`, `listObjects`, `getObjectStream`, `getObjectBuffer`, `getObjectHead` (GET con `Range` para leer solo los primeros bytes sin descargar el fichero) y `putObject`. Cada operación recibe el *rol* de bucket (`'prod' | 'quarantine' | 'jurisprudence'`), nunca el nombre físico: la traducción rol→bucket vive en un único punto (`bucketName`), y usar el rol `jurisprudence` sin haberlo configurado lanza un error explícito en vez de degradar en silencio.
- `StorageAdapter` (implementación `filesystem-adapter.ts`): el backend legado en disco bajo `VAULT_ROOT` (default `/var/www/vhosts/jbhasesorialegal.com/shared/vault`), que sobrevive solo para leer documentos antiguos con `storage_provider='local'` y para algunas subidas directas pequeñas vía `saveDocumentBufferAndInsert` (que estampa el `orgId` del caso cuando el caller lo pasa).

La factoría por app (`apps/portal/src/lib/storage.ts`, análoga en cliente) construye ambos adaptadores como singletons perezosos y **valida en frío la configuración completa**: `getR2()` exige la presencia de `R2_ACCOUNT_ID`, `R2_ACCESS_KEY_ID`, `R2_SECRET_ACCESS_KEY`, `R2_ENDPOINT`, `R2_BUCKET_PROD`, `R2_BUCKET_QUARANTINE` y (en el portal) `R2_BUCKET_JURISPRUDENCE`, y falla con «Missing env var: …» a la primera llamada si falta alguna — una app mal configurada no puede subir «a medias» a un bucket inesperado. Los valores viven out-of-band (`shared/*.env` del servidor / Keychain local).

## 7.3 Flujo de subida multiparte (portal y área cliente)

Las subidas de documentos de caso usan tres endpoints simétricos en ambas apps — portal: `apps/portal/src/app/api/cases/[id]/documents/{upload-init,upload-complete,upload-abort}/route.ts`; cliente: `apps/cliente/src/app/api/my/cases/[id]/documents/{upload-init,upload-complete,upload-abort}/route.ts` — sobre el adaptador multiparte de `packages/storage` (`r2-adapter.ts`).

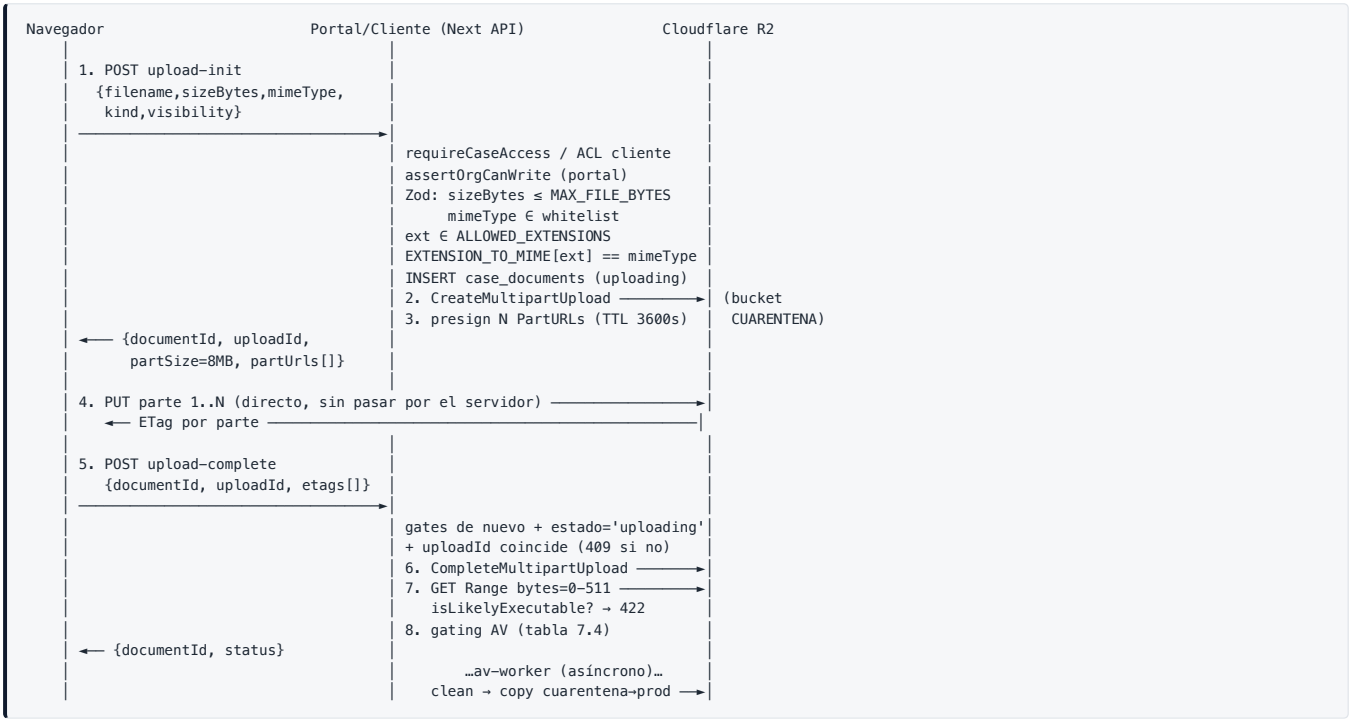


Fig. 7.2 — Subida multiparte con URLs prefirmadas.

#### 7.3.1 `upload-init`: validación antes de tocar R2

El handler valida con Zod antes de crear nada: `filename` (1–500 caracteres), `sizeBytes` entero positivo con techo `MAX_FILE_BYTES = 500 MB`, `mimeType` contra la whitelist `ALLOWED_MIME_TYPES`, `kind` y `visibility` como enums. Después aplica la doble comprobación extensión ↔ MIME: la extensión debe pertenecer a `ALLOWED_EXTENSIONS` (15 extensiones: `pdf, docx, doc, odt, jpg, jpeg, png, zip, xlsx, xls, pptx, ppt, csv, rtf, txt`) y el MIME declarado debe ser exactamente `EXTENSION_TO_MIME[ext]` (400 `extension_not_allowed` / `mime_extension_mismatch` si no). Solo entonces crea la fila (`uploading`) y la multiparte en el bucket de **cuarentena**, y devuelve las URLs prefirmadas de parte (chunk de 8 MB — `PART_SIZE` —, TTL 3600 s). En el portal, la subida es escritura: pasa `assertOrgCanWrite(caso.orgId)` (impersonación/colegiado/pago, cap. 6). En el área cliente, el ACL comprueba `cases.clientUserId === session.user.id` con 404 si no (sin filtrar existencia).

#### 7.3.2 `upload-complete`: verificación y decisión

Reaplica los gates, verifica que el documento está en `uploading` (409 `invalid_status` si no), que el `uploadId` coincide con `upload_session_id` (409 `upload_id_mismatch`) y cierra la multiparte con los ETags ordenados. A continuación:

- **Denylist de ejecutables (auditoría A1):** descarga solo los primeros 512 bytes (`getObjectHead`, GET con `Range`) y ejecuta `isLikelyExecutable` (`packages/storage/src/mime.ts`): firmas PE/DOS (`MZ`), shebang (`#!`), ELF (`\x7FELF`) y Mach-O (incluidos binarios fat). Si coincide, borra el objeto de cuarentena, marca `infected` / `rejected` y responde 422 `executable_signature` — los formatos documentales nunca casan con estas firmas.
- **Gating antivirus** (§7.4) y, en la rama en que el documento queda legible, encolado del análisis IA solo si el presupuesto de IA de la org lo permite (`aiBudgetAllows` — sin consumo silencioso).
- **Auditoría + auto-avance:** `logCaseEvent('document_upload_completed')` y `maybeAdvanceCaseStatus(caseId, 'en_analisis')` (best-effort, forward-only).

El `upload-complete` del área cliente (`apps/cliente/src/app/api/my/cases/[id]/documents/upload-complete/route.ts`) es un espejo exacto de esta lógica — misma denylist de ejecutables sobre `getObjectHead`, mismo gating AV con fail-closed en producción y mismo bypass ruidoso en no-producción — con la diferencia de que el gate previo es el ACL de cliente (`cases.clientUserId === session.user.id`, 404 si no) en lugar de `requireCaseAccess` + `assertOrgCanWrite`: el cliente no está sujeto al eje de pago del despacho, pero sí a todas las garantías de contenido.

#### 7.3.3 `upload-abort`

Si el `uploadId` coincide con el registrado, aborta la multiparte en cuarentena (tolerando `NoSuchUpload`) y marca la fila `failed` con `scan_message='Aborted by user'`, limpiando `upload_session_id`. Queda auditado (`document_upload_aborted`). Como red de seguridad de infraestructura, los buckets tienen reglas de ciclo de vida que abortan multipartes incompletas (7 días en prod, 2 en cuarentena) y limpian la cuarentena estancada a los 7 días.

#### 7.3.4 Taxonomía de errores de la cadena de subida

Todos los handlers de documentos van envueltos en `withErrorHandler` (`apps/portal/src/lib/errors.ts` y homólogo en cliente), que captura las excepciones tipadas, las reporta a Sentry con el `x-request-id` de correlación como tag y las traduce a respuestas JSON estables que la UI puede accionar. Los códigos relevantes para la cadena documental:

| Excepción / condición | HTTP | Cuerpo (`error`) | Cuándo |
|---|---|---|---|
| `UnauthorizedError` | 401 | `unauthorized` | Sin sesión. |

| Excepción / condición | HTTP | Cuerpo (`error`) | Cuándo |
|---|---|---|---|
| `ForbiddenError` / `MfaRequiredError` / `ReadOnlyImpersonationError` | 403 | `forbidden` / `mfa_required` / `read_only_impersonation` | Rol no permitido; segundo factor pendiente; superadmin en «ver como». |
| `NotFoundError` | 404 | `not_found` | Caso/documento inexistente, de otro tenant, soft-borrado o no visible (regla 404-antes-que-403). |
| `SubscriptionRequiredError` | 402 | `subscription_required` + `reason` | Eje de pago del despacho (modo lectura / primer caso consumido). |
| Validación de init | 400 | `extension_not_allowed` / `mime_extension_mismatch` | Extensión fuera de whitelist o MIME que no casa con ella. |
| Estado inconsistente | 409 | `invalid_status` / `upload_id_mismatch` / `not_ready` | Completar dos veces, UploadId ajeno, descargar antes de `ready`. |
| `FileTooLargeError` | 413 | `file_too_large` + `maxBytes` | Subida directa por encima del cap (10 MB). |
| `UnsupportedMimeTypeError` | 415 | `unsupported_mime_type` | Extensión no admitida para ese tipo de documento. |
| `MimeMismatchError` / `MultipartParseError` | 400 | `mime_mismatch` / `multipart_parse_failed` | Magic bytes que no casan; FormData malformado. |
| Firma de ejecutable | 422 | `file_rejected` (`reason: 'executable_signature'`) | Los bytes reales son un binario nativo o script. |
| `RateLimitError` | 429 | `rate_limited` + `Retry-After` | Cuotas de subida/descarga superadas. |

ℹ **Errores accionables, no páginas en blanco.** Estos códigos estables alimentan la UX de errores de subida accionables del portal (auditoría UX 2026-07-12): el usuario ve exactamente por qué se rechazó el fichero (tamaño, tipo, contenido) y qué hacer, en lugar de un fallo genérico.

## 7.4 Subidas directas y verificación por magic bytes

Las subidas pequeñas (formularios de onboarding y variantes directas) no usan multiparte: el fichero viaja en `FormData` al servidor, que *lee los bytes reales* antes de guardar:

- `detectMimeType(buffer, hintedExt)` (`packages/storage/src/mime.ts`): sniffing de firma binaria — `%PDF`, PNG, JPEG, ZIP (`PK\x03\x04`, aceptado como docx/odt/zip por ser contenedores ZIP) y OLE2 (`doc` legado). Si la firma no existe o no casa con la extensión declarada → `MimeMismatchError` (el fichero se rechaza aunque la extensión fuera válida).
- **Onboarding de colaborador** (`apps/portal/src/app/api/onboarding/[token]/upload-doc/route.ts`): sesión por token de onboarding (no cookie), rate limit 30/h por `jti`, **cap de 10 MB** (`MAX_BYTES` → `FileTooLargeError`), extensiones por tipo de documento (`colegiacion`: pdf/jpg/jpeg/png; `seguro_rc`: solo pdf), magic bytes obligatorios y reemplazo suave del documento anterior del mismo tipo.
- **Área cliente / onboarding de cliente** (`apps/cliente/src/app/api/my/cases/[id]/documents/route.ts` y `…/api/onboarding/[token]/cases/[id]/documents/route.ts`): mismo patrón `FileTooLargeError` / `UnsupportedMimeTypeError` / `MimeMismatchError` / `MultipartParseError` con `detectMimeType` sobre el buffer.

## 7.5 Antivirus: `jbhlegal-av-worker` (ClamAV)

### 7.5.1 Gating en `upload-complete` y política fail-closed

| Escenario | Qué ocurre | Estado / veredicto resultante |
|---|---|---|
| `AV_ENABLED=true` | El fichero permanece en cuarentena; el av-worker lo escaneará y promocionará. | `scanning`; luego `ready/clean` o `infected`. |
| AV apagado + **producción** | **FAIL CLOSED**: el documento queda retenido en cuarentena, NO se promociona a prod y NO se encola su análisis IA. La subida «tiene éxito» para el usuario, con aviso de pendiente de revisión antivírica; se procesará al provisionar el AV. `console.error` ruidoso. | `scanning` + `scan_verdict='pending'`, respuesta con `pendingScan: true`. |
| AV apagado + **no-producción** | Bypass de desarrollo: promoción inline cuarentena→prod sin escaneo, con warning ruidoso y rastro en la auditoría del caso (`avBypassed`). | `ready` + `scan_verdict='skipped'`. |

⚠ **Regla de producción.** Un fichero jamás llega al bucket de producción sin escanear *en producción* por la vía normal. La única excepción deliberada es la degradación del propio worker cuando `clamd` está caído (abajo), que promociona marcando `scan_verdict='unscanned'` — trazable y reversible provisionando clamd, y preferible a dejar el expediente inservible.

### 7.5.2 El worker

Proceso pm2 `jbhlegal-av-worker` (`infra/plesk/ecosystem.config.cjs`; entry `apps/portal/scripts/av-worker-entry.ts` compilado a `workers/av-worker-entry.cjs`, env compartida de `shared/portal.env`). Cada `SCAN_POLL_INTERVAL_MS` (30 s por defecto) ejecuta `runAvScanCycle` (`apps/portal/src/lib/av-worker.ts`):

1. Lista hasta 20 objetos del bucket de cuarentena (`batchSize`). Si el listado falla (caída de R2), lanza `R2ListUnavailableError` y el bucle exterior aplica **backoff exponencial** 30 s → 60 s → 2 m → 4 m con techo `SCAN_BACKOFF_CAP_MS` (5 min), en vez de martillear el bucket; errores por objeto se registran sin cortar el ciclo (y van a Sentry si hay `SENTRY_DSN`).
2. Por objeto, resuelve la fila por `storage_path` (objeto sin fila → log `orphan_object`) y lo hace fluir *en streaming* por `clamd` vía socket UNIX (`CLAMD_SOCKET`, default `/var/run/clamd.scan/clamd.sock`) con el protocolo **INSTREAM** (`apps/portal/src/lib/clamav-scanner.ts`: `zINSTREAM\0`, chunks con prefijo de longitud BE de 4 bytes, terminador cero; respuesta `stream: OK` o `stream: <firma> FOUND`; timeout 60 s).
3. **Limpio** → `copyObject` cuarentena→prod, borrado en cuarentena, fila a `ready/clean`, y encolado del análisis IA (si `aiBudgetAllows`).
4. **Infectado** → objeto **eliminado** (no se conserva), fila a `infected` con la firma en `scan_message`, evento `document_infected` en la auditoría del caso y **email al uploader** explicando el rechazo.
5. `clamd` **no disponible** (ENOENT/ECONNREFUSED/timeout…) → degradación documentada: promociona a prod con `scan_verdict='unscanned'` y mensaje «AV no disponible — promovido sin escaneo»; provisionar clamd restaura el escaneo real automáticamente. Cualquier otro error de escaneo deja `scan_verdict='error'` con el mensaje truncado a 500 caracteres, sin promocionar.

El aprovisionamiento de clamd está guionizado en `infra/plesk/provision-clamav.sh`; al completarse se establece `AV_ENABLED=true` y `CLAMD_SOCKET` en la env del portal.

## 7.6 Almacenamiento en Cloudflare R2

| Rol (código) | Bucket | Uso | Ciclo de vida |
|---|---|---|---|
| `prod` | `[bucket principal reservado]` (WEUR, Ámsterdam) | Documentos promocionados; versionado de objetos (máx. 5 versiones, expiran a 90 días). | Borrado de soft-deleted a 30 días (por tag), aborto de multipartes a 7 días. |
| `quarantine` | `[bucket de cuarentena reservado]` (WEUR) | Zona de tránsito pre-escaneo; sin versionado. | Limpieza total a 7 días, aborto de multipartes a 2 días. |
| `jurisprudence` | bucket separado (opcional en `R2Config`) | Biblioteca de jurisprudencia del Letrado IA; solo lo configuran portal y ai-worker. | — |

- **Claves de objeto:** `objectKey` en `packages/storage/src/r2-adapter.ts` produce `cases/<caseId>/<docId>.<ext>` (o prefijo `intakes/`): rutas no adivinables (dos UUID) y agrupadas por expediente. La misma clave se usa en cuarentena y en prod, lo que hace la promoción una copia 1:1.
- **Cifrado:** R2 cifra *todo* objeto en reposo con **AES-256** y claves gestionadas por Cloudflare, incondicionalmente (las cabeceras SSE de S3 son no-op en R2 — comentario en `putObject` y `packages/storage/README.md`, M3); el tránsito va siempre por TLS (endpoint `https://<accountId>.r2.cloudflarestorage.com`, SigV4). La residencia europea se fija a nivel de bucket (WEUR).
- **Credenciales segregadas:** dos tokens R2 de mínimo privilegio — el de app (lectura/escritura en ambos buckets, envs `R2_ACCESS_KEY_ID`/`R2_SECRET_ACCESS_KEY` de `portal.env`/`cliente.env`/`firmas.env`) y el del worker (lectura en cuarentena, RW+delete en prod, delete en cuarentena). Los valores viven out-of-band (Keychain local / `shared/*.env` del servidor); en el repo solo están los nombres. Config completa: `R2_ACCOUNT_ID`, `R2_BUCKET_PROD`, `R2_BUCKET_QUARANTINE`, `R2_ENDPOINT`, `STORAGE_PROVIDER=r2`.

## 7.7 Descarga: siempre 302 a URL prefirmada

Rutas: portal `apps/portal/src/app/api/cases/[id]/documents/[did]/download/route.ts`; cliente `apps/cliente/src/app/api/my/cases/[id]/documents/[did]/download/route.ts` (más la variante de onboarding por token). Flujo del portal:

1. `requireCaseAccess(id)` — todo el gate del capítulo 6.
2. Rate limit por usuario: `doc-dl:<userId>` 200/h en portal; `doc-dl-cliente:<userId>` 60/h en cliente.
3. Resolución del documento con `eq(id) AND eq(caseId) AND deleted_at IS NULL` → 404 si no (un soft-borrado no se descarga).
4. **Bloqueo anti-fraude de peritajes (Fase 17 · P2):** un informe pericial (`kind='informe'`, `uploader_role='PERITO'`) solo se descarga si el presupuesto del peritaje (relación por `case_id` + `perito_user_id`) está `paid`; si no, 402 `peritaje_unpaid`. El área cliente aplica el mismo gating.
5. Estado: solo `ready` o `completed` (legado) se descargan; otro estado → 409 `not_ready`. Un documento en cuarentena o infectado es indescargable por construcción.
6. Auditoría: `logCaseEvent('document_downloaded')` con IP (primer salto de `x-forwarded-for`) y user-agent truncado.
7. Respuesta: para `storage_provider='r2'`, **302** a `getPresignedDownloadUrl` con **TTL 300 s**, `ResponseContentDisposition: attachment` (nombre saneado + RFC 5987; `inline` solo para reproducción de audio/vídeo) y `Cache-Control: private, no-store` para que ningún proxy/CDN cachee la URL efímera. El contenido **nunca** se proxya por Node — la única excepción es el legado `local`, que se sirve en streaming desde el filesystem con las mismas cabeceras privadas.

En el área cliente se suman dos condiciones de ACL previas: el caso debe pertenecer al usuario (`cases.clientUserId === session.user.id`) y el documento debe ser `visibility='shared'`; ambas fallan con 404 para no filtrar la existencia de documentos internos.

> ✓ **Garantía.** No existe ninguna ruta que sirva un documento de caso sin (a) sesión válida, (b) gate de caso/ACL de cliente, (c) estado descargable y (d) URL prefirmada de vida corta. Los buckets no tienen acceso público, por lo que la posesión de la clave del objeto no otorga acceso.

## 7.8 Ciclo de vida: soft-delete, retención y purga

El borrado de un documento por el usuario (DELETE `/api/cases/[id]/documents/[did]`) es un **soft-delete**: estampa `deleted_at` (+ `deleted_by_id`); el objeto R2 sobrevive 30 días como periodo de gracia frente a cierres accidentales. El borrado físico lo ejecuta el **retention worker** (pm2 `jbhlegal-retention-worker`, entry `apps/portal/scripts/retention-worker-entry.ts`, `cron_restart: '0 2 * * *'` — diario a las 02:00 UTC), cuyo ciclo (`apps/portal/src/lib/document-retention-worker.ts`) es:

| # | Paso | Regla |
|---|---|---|
| 1 | Estampado de retención | `retention_until = fecha_cierre + 10 años` para documentos de casos cerrados que aún no la tengan (obligación deontológica de conservación del expediente). |
| 2 | Soft-delete por caducidad | `deleted_at = NOW()` cuando `retention_until` queda en el pasado. |
| 2b | Grabaciones de reuniones | Módulo Plaud: misma ventana de 10 años tras `fecha_cierre`, sin soft-delete intermedio — se borra el objeto R2 y después la fila (si R2 falla, la fila queda y se reintenta). |
| 3 | **Hard-delete diferido (+30 días)** | Para documentos con `deleted_at < NOW() - 30 días`, `storage_provider='r2'` y `r2_etag` no nulo: borra el objeto del bucket prod y anula `r2_etag` (la fila de metadatos persiste como rastro auditable, ya sin binario). |
| 3b | **Purga de `document_analyses` (PRIV-06)** | El análisis IA del documento (resumen + entidades extraídas: **DNI, IBAN, direcciones** + excerpts literales) es PII derivada con la misma retención que el fichero: se borra en cascada con los documentos hard-borrados del paso 3; sin esto sobreviviría indefinidamente en la BD tras desaparecer el fichero. |
| 4 | Documentos de intake | Purga a 30 días de `intakes.created_at` para leads `pending/nuevo/asignado` y a 7 días para `descartado`; nunca toca los promovidos a `case_documents` (gobernados por la política de caso). El fallo del delete R2 no bloquea el marcado (mejor un blob huérfano que una fila atascada reintentándose cada ciclo). |
| 5–6 | Leads y reservas no convertidos (RGPD T2) | Intakes >12 meses sin convertir (sin `case_id`, no `contratado`) y bookings de visitante >12 meses sin cliente/caso: borrado de filas en lotes de 100 con limpieza previa best-effort de sus objetos R2 y borrado explícito de `crm_tasks` / `crm_presupuestos` (FK suave) antes de la cascada. |

ℹ **Barrido global por diseño.** El retention worker es el único proceso que consulta `case_documents` sin filtro de `org_id`: la política de conservación es idéntica para todos los tenants y no hay sesión activa. Proyecta `org_id` en sus SELECT solo para trazabilidad en logs; las cascadas posteriores se acotan por los IDs ya seleccionados (org-acotados por su documento padre).

## 7.9 Aislamiento y auditoría de la cadena documental

- **Scoping doble:** cada documento lleva `case_id` + `org_id`; el acceso pasa por `requireCaseAccess` (que filtra el caso por tenant) y la consulta del documento vuelve a anclar `eq(caseId)`. Un UUID de documento válido de otro tenant no se resuelve.
- **Barrido anti-IDOR (auditoría 2026-07-20):** el sweep de las 335 rutas API (268 portal + 67 cliente, cap. 6 §6.6) incluyó específicamente las rutas de documentos (init/complete/abort/download/delete de portal, cliente y onboarding) con resultado **limpio**: todas pasan por su gate y ninguna resuelve recursos fuera del tenant o del dueño.
- **Rastro completo:** eventos `document_upload_initiated`, `document_upload_completed` (con flags `avBypassed` / `quarantined` / `rejected`), `document_upload_aborted`, `document_infected` y `document_downloaded` (con IP y user-agent) en la auditoría del caso.

## 7.10 Garantías del sistema vs. responsabilidades del operador

| El sistema garantiza (por código) | El operador debe garantizar (configuración/infra) |
|---|---|
| Whitelist de extensiones/MIME, techo de 500 MB (10 MB en directas), coherencia extensión↔MIME y magic bytes en subidas directas. | Mantener `AV_ENABLED=true` y `clamd` provisionado y vivo (`provision-clamav.sh`; socket en `CLAMD_SOCKET`): sin ello, en prod los documentos se acumulan en cuarentena (fail-closed) y con clamd caído el worker promociona `unscanned`. |
| Denylist de ejecutables sobre los bytes reales (PE/ELF/Mach-O/shebang) incluso antes del AV. | Mantener los buckets R2 **privados**: jamás habilitar «Public Development URL» (`*.r2.dev`) ni dominio público — haría legibles los objetos por clave sin credenciales. |
| Cuarentena por defecto; promoción a prod solo tras veredicto; objeto infectado eliminado y uploader notificado. | Custodiar los tokens R2 (app y worker) out-of-band y conservar su segregación de privilegios; rotarlos ante sospecha. |
| Descarga solo autenticada+autorizada, por 302 a URL prefirmada de 300 s, con `no-store`; nunca proxy ni URL pública. | Vigilar los procesos pm2 `jbhlegal-av-worker` y `jbhlegal-retention-worker` (logs en `/var/www/vhosts/jbhasesorialegal.com/logs/`): si el retention worker no corre, no hay hard-delete ni purga RGPD. |
| Cifrado AES-256 en reposo (incondicional en R2) y TLS en tránsito; residencia WEUR. | Mantener las reglas de ciclo de vida y el versionado de los buckets tal como define `infra/scripts/setup-r2-buckets.md`; DPA de Cloudflare firmado. |
| Retención 10 años tras cierre, soft-delete con gracia de 30 días, purga en cascada de `document_analyses` (DNI/IBAN) y de leads/reservas no convertidos a 12 meses. | Revisar los documentos retenidos en cuarentena tras un periodo con AV apagado, y monitorizar `scan_verdict IN ('error','unscanned','pending')` para reprocesarlos. |
| Auditoría por evento (subida/aborto/infección/descarga con IP+UA). | Backups nocturnos de BD y secretos (tarea Plesk) y prueba periódica de recuperación — el binario en R2 no se respalda por esta vía; el versionado de objetos (5 versiones/90 días) es la red frente a sobrescrituras. |

⚠ **Síntesis del riesgo residual.** Los dos únicos caminos por los que un binario llega a prod sin veredicto `clean` son deliberados, ruidosos y trazables: el bypass de no-producción (`skipped`) y la degradación por clamd caído (`unscanned`). Ambos son consultables por `scan_verdict` y reversibles reprocesando; ninguno ocurre silenciosamente en producción con el AV operativo.

# 8

# Protección de datos, RGPD y anonimización

> *JBH Legal trata datos relativos a condenas e infracciones penales — la categoría de dato personal más protegida del ordenamiento europeo (art. 10 RGPD) — junto a datos de salud y otros datos de categoría especial (art. 9 RGPD) que aparecen en expedientes penales. Este capítulo documenta, con referencia al código real del monorepo, el sistema completo de cumplimiento: gobernanza y encargados del tratamiento, el flujo de derechos del interesado (DSR), la anonimización irreversible del art. 17, la retención y purga automáticas, la seudonimización hacia la IA, el Zero Data Retention contractual con Anthropic, el consentimiento versionado con valor probatorio, los backups cifrados de disaster recovery y el rastro de auditoría forense.*

## 8.1 Marco normativo y gobernanza

### 8.1.1 Naturaleza de los datos tratados

La plataforma trata, como actividad principal, expedientes de defensa penal: relatos de hechos, atestados, autos judiciales, declaraciones, informes periciales, transcripciones de reuniones abogado–cliente y análisis de IA derivados de todo lo anterior. En términos del RGPD, esto es tratamiento de **datos relativos a condenas e infracciones penales (art. 10 RGPD)**, un régimen que en España el art. 6.2 de la LOPDGDD (LO 3/2018) reserva esencialmente a quienes los tratan bajo el amparo de una norma o para el ejercicio/defensa de reclamaciones — exactamente el supuesto de la asistencia letrada. Además, los documentos del expediente contienen con frecuencia **categorías especiales del art. 9 RGPD**: datos de salud (informes forenses, partes de lesiones, informes psicológicos y de toxicología), vida sexual (delitos contra la libertad sexual), origen étnico o ideología cuando son relevantes en la causa.

| Categoría de dato | Ejemplos en la plataforma | Régimen RGPD | Dónde vive |
|---|---|---|---|
| Identificativos | `users.name`, `users.email`, `intakes.nombre/telefono/email` | Art. 6 | MariaDB |
| Penales | `cases.hechos`, `cases.detencion_json`, documentos del caso, análisis IA, transcripciones | Art. 10 RGPD + art. 10 LOPDGDD | MariaDB + R2 (`[bucket principal reservado]`) |
| Categorías especiales | Datos de salud/sexuales dentro de documentos y análisis | Art. 9 RGPD | MariaDB + R2 |
| Biométricos indirectos | Voz del cliente en `meeting_recordings` (audio R2) | Art. 9 (máxima cautela) | R2 |
| Económicos | `case_payments`, `lawyer_invoices`, `crm_presupuestos` | Art. 6 + retención mercantil | MariaDB (+ Stripe) |
| De terceros | `case_witnesses` (nombre, email, teléfono, ubicación de testigos) | Art. 6/14 | MariaDB |
| Técnicos/prueba | `intakes.ip_address`, `intakes.user_agent`, firmas eIDAS | Art. 6 (interés legítimo/obligación) | MariaDB |

### 8.1.2 Base jurídica y responsable

La base jurídica principal del tratamiento es el **art. 6.1.b RGPD**: el tratamiento es necesario para la ejecución del contrato de servicios (encargo de intermediación jurídica y coordinación de la defensa) o para la aplicación de medidas precontractuales a petición del interesado (el intake «Evaluar mi caso» es la medida precontractual por excelencia). Para el tratamiento de datos penales, la habilitación material es el ejercicio y defensa de reclamaciones en el marco de la asistencia letrada. Las comunicaciones comerciales tienen base propia de **consentimiento (art. 6.1.a)**, capturado como opt-in separado (§8.7). El responsable del tratamiento es **JBH FINANCIAL GROUP, S.L. (CIF B06770143)**, y el **Delegado de Protección de Datos es externo: VeraSafe, LLC**, con canal de contacto `dpo@jbhasesorialegal.com`. Ambos constan en las páginas legales publicadas de `apps/web/src/app/(corporate)/legal/` (aviso legal, política de privacidad, cookies y condiciones), que están en producción y sin marcas de borrador.

> ℹ **Plazo de respuesta (art. 12 RGPD).** Toda solicitud de derechos debe atenderse en el plazo máximo de **1 mes** desde su recepción (prorrogable dos meses en casos complejos, informando al interesado). El README del export lo declara expresamente («Plazo legal de respuesta: 1 mes») y el runbook `docs/runbooks/11-dsr-rgpd.md` incluye el procedimiento de escalado si llega una reclamación de la AEPD por solicitud no atendida en plazo: localizar la fila en `dsr_requests`, ejecutarla manualmente y documentar la respuesta en el dossier del DPO.

### 8.1.3 Encargados del tratamiento declarados

La política de privacidad (`apps/web/src/app/(corporate)/legal/privacidad/page.tsx`, sección «4. Destinatarios y encargados del tratamiento») declara la lista completa de encargados, todos vinculados por contrato de encargo (DPA) conforme al art. 28 RGPD y, para las transferencias internacionales, mediante Cláusulas Contractuales Tipo (SCC):

| Encargado | Servicio | Datos que ve | Garantía de transferencia |
|---|---|---|---|
| Resend, Inc. | Envío de email transaccional (OTP, notificaciones) | Email, nombre, asunto del mensaje | SCC |
| Stripe Payments Europe, Ltd. | Procesamiento de pagos y suscripciones | Identidad de facturación, importes; nunca datos del expediente | Entidad UE; DPA Stripe |

| Encargado | Servicio | Datos que ve | Garantía de transferencia |
|---|---|---|---|
| Cloudflare, Inc. | CDN, WAF, DNS, Turnstile | Metadatos de tráfico | SCC + cifrado en tránsito |
| Cloudflare R2 (Cloudflare, Inc.) | Almacenamiento de documentos, audio y backups | Documentos del expediente (cifrados en reposo) | SCC + cifrado en reposo y en tránsito |
| Anthropic PBC | API de IA (Letrado IA) | Contenido del caso seudonimizable | SCC + **Zero Data Retention** (§8.6) |
| 8x8, Inc. (Jitsi Meet — meet.jit.si) | Videollamadas abogado–cliente | Flujo audio/vídeo de la llamada | SCC + E2EE |
| Google Ireland Ltd. (GA4) | Analítica web (solo con consentimiento de cookies) | Métricas de navegación | SCC + `anonymize_ip` |
| Functional Software, Inc. (Sentry) | Monitorización de errores | Trazas técnicas | SCC + scrubbing de PII en el cliente |
| Hosting (VPS Plesk) | Infraestructura de aplicación y MariaDB | Todo el dato en reposo del lado servidor | DPA de hosting; servidor en la UE |

⚠ **Regla de mínimos hacia terceros.** Ningún encargado recibe más de lo imprescindible para su función: Stripe nunca ve contenido del expediente; Resend solo ve destinatario y plantilla; Sentry recibe trazas con scrubbing de PII activado en el SDK de cliente; y hacia Anthropic el contenido penal viaja bajo ZDR contractual y, adicionalmente, seudonimizado cuando hay roster de nombres (§8.5). Añadir un nuevo tercero exige actualizar la tabla de la política de privacidad y pasar el checklist de §8.10.

## 8.2 Derechos del interesado (DSR): acceso, portabilidad y supresión

El sistema DSR (*Data Subject Rights*) es **self-service** para los roles `CLIENT` y `COLABORADOR` desde el área de cliente, con un gate administrativo sobre la supresión. La pieza central es la tabla `dsr_requests` (`packages/db/src/schema/dsr-requests.ts`) y los módulos de `apps/cliente/src/lib/`: `dsr-service.ts` (máquina de estados), `dsr-exporter.ts` (export art. 15/20), `dsr-anonymizer.ts` (supresión art. 17) y `dsr-types.ts` (constantes de política).

### 8.2.1 La tabla `dsr_requests`

| Columna | Tipo | Función |
|---|---|---|
| `id` | varchar(36) PK | UUID de la solicitud |
| `user_id` | varchar(36) FK → `users.id` | Interesado. FK con `onDelete: 'restrict'`: no se puede borrar un usuario con solicitudes vivas |
| `kind` | varchar(20) | `export` \| `delete` (constante `DSR_KINDS`) |
| `status` | varchar(30) | Estado de la máquina (§8.2.2); default `pending` |
| `reason` | text | Motivo libre aportado por el interesado en la solicitud de borrado |
| `decided_by_id` | varchar(36) FK → `users.id` | Admin que aprobó/rechazó (`onDelete: 'set null'`) |
| `decision_reason` | text | Motivo del rechazo, o mensaje de error si `failed` |
| `artifact_path` | varchar(500) | Ruta del ZIP en el vault (`dsr/<userId>/<requestId>.zip` bajo `VAULT_ROOT`) |
| `artifact_token` | varchar(64) | Token de descarga de un solo artefacto |
| `artifact_expires_at` | datetime | Caducidad del token (TTL 7 días) |
| `created_at` / `updated_at` / `completed_at` | datetime | Trazabilidad temporal completa (prueba del plazo de 1 mes) |
| `org_id` | varchar(36) | Tenant (multi-tenancy Modelo B); índice `idx_dsr_requests_org` |

Los índices `idx_dsr_user_status` (usuario + estado) y `idx_dsr_status_kind` (estado + tipo) sirven respectivamente al panel del interesado y a la cola de revisión del administrador.

### 8.2.2 Máquina de estados y gate administrativo

Los estados válidos (`DSR_STATUSES`) son `pending`, `pending_review`, `in_progress`, `completed`, `rejected` y `failed`. La diferencia clave entre los dos tipos de solicitud está en el arranque: un **export** nace en `pending` y el worker lo procesa sin intervención humana; un **delete** nace en `pending_review` y *requiere aprobación expresa de un ADMIN* antes de ejecutarse, porque la anonimización es irreversible (§8.3) y porque puede colisionar con obligaciones de conservación o con un procedimiento judicial en curso.

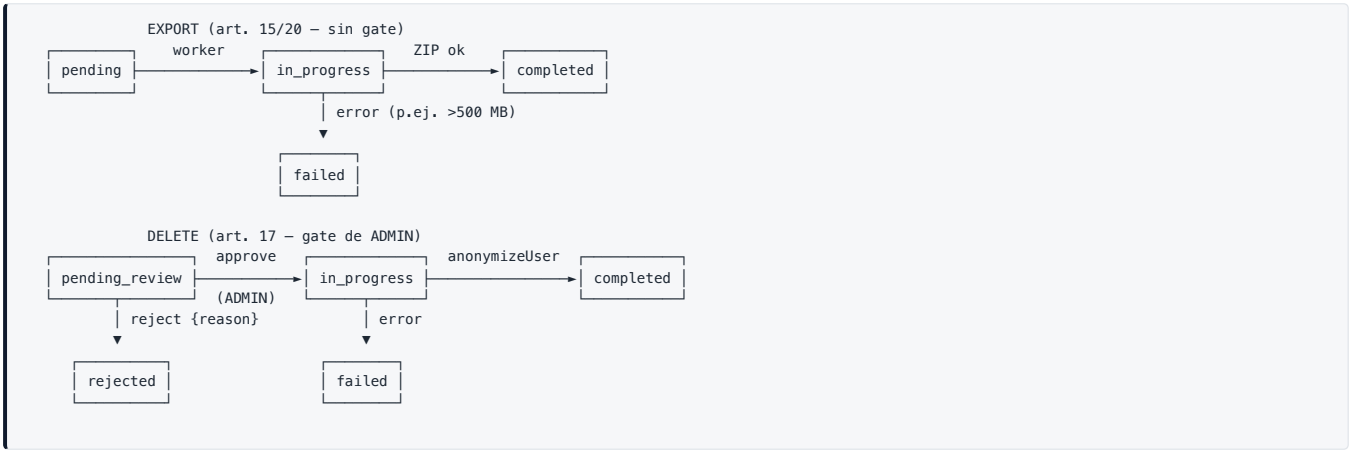


Fig. 8.1 — Máquinas de estados de `dsr_requests` por tipo de solicitud.

Las transiciones están protegidas contra carreras: `transitionToInProgress()` y `approveDelete()` (en `dsr-service.ts`) abren transacción y toman un `SELECT … FOR UPDATE` sobre la fila antes de validar el estado origen; una transición desde un estado no permitido lanza `DsrInvalidStateError`. Así dos workers concurrentes no pueden procesar la misma solicitud, y un admin no puede aprobar dos veces.

### 8.2.3 Endpoints y límites de tasa

| Endpoint | Rol | Función | Límite |
|---|---|---|---|
| `POST /api/me/dsr/export` | CLIENT, COLABORADOR | Crear solicitud de export; dispara worker asíncrono | `EXPORT_RATE_LIMIT_PER_DAY = 5` por 24 h |
| `POST /api/me/dsr/delete` | CLIENT, COLABORADOR | Crear solicitud de borrado (queda `pending_review`) | `DELETE_RATE_LIMIT_DAYS = 30`: 1 solicitud de borrado cada 30 días |
| `GET /api/me/dsr` | CLIENT, COLABORADOR | Listar solicitudes propias | — |
| `GET /api/me/dsr/[id]/download?token=…` | Interesado | Stream del ZIP si el token es válido y no ha caducado | TTL 7 días |
| `GET /api/admin/dsr` | ADMIN + COORDINADOR (lectura) | Lista global con filtros | — |
| `GET /api/admin/dsr/[id]` | ADMIN + COORDINADOR | Detalle + *preview de anonimización* (recuentos de mensajes, documentos, pagos, firmas y casos afectados vía `getAnonymizationPreview()`) | — |
| `POST /api/admin/dsr/[id]/approve` | ADMIN | Aprueba el borrado y dispara el worker | — |
| `POST /api/admin/dsr/[id]/reject` | ADMIN | Rechaza con motivo obligatorio (`{reason}`) | — |

Además del rate limit, `assertCanRequestExport()` implementa una salvaguarda de coherencia: si el usuario tiene un **borrado abierto** (`pending_review` o `in_progress`), los exports quedan bloqueados — evita exportar una foto de datos que están a punto de anonimizarse y, de paso, cierra el resquicio de «exporto todo y luego borro» como mecanismo de carrera contra el gate del admin.

### 8.2.4 El export: ZIP de acceso y portabilidad

`buildExportBundle()` (`apps/cliente/src/lib/dsr-exporter.ts`) construye un ZIP con esta estructura:

```
dsr/<userId>/<requestId>.zip
├── README.txt          — explicación en lenguaje llano: estructura, rol del usuario,
│                         base legal (art. 15/20), plazo de 1 mes y contacto del DPO
├── data.json           — TODOS los datos estructurados, agrupados por categoría
└── documents/
    ├── case-<caseId>/<originalFilename>   — documentos subidos por el usuario, por caso
    └── colaborador/<originalFilename>      — documentación de onboarding del colaborador
```

El orquestador `collectUserData()` delega en colectores por dominio, testables por separado. Las categorías de `data.json` son un inventario exhaustivo de la huella del usuario:

| Categoría en `data.json` | Colector | Contenido | Roles |
|---|---|---|---|
| `identity` | `collectIdentity` | Nombre, email, fecha de alta | Todos |

| Categoría en `data.json` | Colector | Contenido | Roles |
|---|---|---|---|
| `intakes`, `crm_tasks`, `crm_presupuestos`, `lawyer_invoices` | `collectIntakes` | Solicitudes de evaluación (resueltas por email), tareas comerciales ligadas a sus intakes (sin notas internas ajenas), presupuestos y facturas de honorarios emitidas al interesado | CLIENT, COLABORADOR |
| `cases` | `collectCases` | Expedientes como cliente o como letrado, incluyendo `hechos` y `detencion_json` (relato y datos de la detención: son datos del interesado y deben ser portables) | CLIENT, COLABORADOR |
| `messages` | `collectMessages` | Mensajes enviados por el usuario (`case_messages.sender_user_id`) | CLIENT, COLABORADOR |
| `documents` + `documents/` | `collectCaseDocuments` | Metadatos y bytes de los documentos subidos por el usuario | CLIENT, COLABORADOR |
| `signatures` | `collectSignatures` | Solicitudes de firma como firmante u owner, con `documentHashSha256` | CLIENT, COLABORADOR |
| `caseWitnesses`, `caseLawyerInputs` | `collectCaseWitnesses/-LawyerInputs` | Testigos propuestos y criterio del letrado sobre el expediente del usuario | CLIENT, COLABORADOR |
| `meetingRecordings` | `collectMeetingRecordings` | Transcripción diarizada, resumen y metadatos de consentimiento de las grabaciones de sus casos. El AUDIO no va en el ZIP (tamaño); se entrega bajo petición específica | CLIENT, COLABORADOR |
| `payments` | `collectPayments` | `case_payments` de sus casos (resueltos vía caso; la tabla no tiene `clientUserId`) | CLIENT |
| `collaboratorProfile`, `collaboratorPracticeAreas` + documentos | `collectColaborador*` | Perfil profesional, áreas de práctica y documentación de colegiación | COLABORADOR |
| `herencia`, `herenciaAnswers`, `herenciaReports` | `collectHerenciaData` | Consultas de sucesiones, respuestas del cuestionario, informes entregados y documentos. Corre *incondicionalmente* para cualquier rol (fix A11: la brecha era el export, no el borrado) | Todos |
| `events` | `collectEvents` | Eventos de `case_events` donde el usuario es actor | Todos |

> ✓ **Portabilidad real de los documentos R2 (fix A10).** El exportador ramifica por `storageProvider`: cuando el documento vive en R2, `storagePath` es una *clave de bucket*, no una ruta de disco, y el archivo se transmite con `getR2().getObjectStream()` hacia el ZIP. El código antiguo usaba `archive.file()` (que espera una ruta local) y habría *omitido silenciosamente* los documentos R2, vulnerando el art. 20. El límite duro es `MAX_EXPORT_BYTES = 500 MB`: superado el acumulado, el exportador aborta el archivo y lanza `DsrExportTooLargeError` (la solicitud queda `failed` y se gestiona manualmente).

Completado el export, la solicitud guarda `artifact_path`, un `artifact_token` aleatorio y `artifact_expires_at` con `EXPORT_TOKEN_TTL_MS = 7 días`, y el interesado recibe el enlace de descarga por email. El worker de limpieza `apps/portal/src/lib/dsr-cleanup-worker.ts` corre cada 24 h: borra del filesystem los ZIP con `artifact_expires_at < NOW()` y anula `artifact_path`/`artifact_token`, dejando la fila como pura auditoría. Opcionalmente hace ping a `BETTERSTACK_DSR_HEARTBEAT_URL` como latido de monitorización.

## 8.3 Anonimización irreversible (art. 17 — derecho de supresión)

La supresión no se implementa como borrado de la fila de `users` (imposible: decenas de FKs y obligaciones de conservación) sino como **anonimización irreversible** en `anonymizeUser()` (`apps/cliente/src/lib/dsr-anonymizer.ts`): se destruye todo dato que identifique al interesado y todo contenido personal sin base de retención, y se conservan exclusivamente las filas que una obligación legal impone conservar — ya desvinculadas de una identidad real. El resultado devuelto (`AnonymizationResult`) enumera `categoriesAnonymized`, `filesDeleted` y `retainedForLegal`, y este último se muestra al interesado como explicación transparente de qué se retiene y por qué.

### 8.3.1 Identidad: hash y sentinelas

El primer paso, dentro de una **transacción única** de base de datos, reescribe la identidad:

```
email  → anon-<sha256(userId)[0..16]>@deleted.local   // determinista, no reversible, no colisionable en la práctica
name   → '[ANONIMIZADO]'                               // constante ANON_NAME
colegio, colegiacion_num → NULL
role   → 'ANONYMIZED'                                  // marca terminal del estado
```

El email anónimo es un **hash SHA-256 del userId truncado a 16 hex** bajo el dominio inexistente `deleted.local`: mantiene la unicidad que exige el índice de `users.email` sin conservar ningún rastro del email real. El rol `ANONYMIZED` cumple doble función: impide el login (ningún flujo de auth lo admite) y hace la operación **idempotente** — reejecutar `anonymizeUser()` sobre un usuario ya anonimizado lanza `DsrAnonymizationFailedError` con `stage='already_anonymized'` y no toca nada.

### 8.3.2 Tabla campo a campo: qué pasa con cada categoría

| Tabla / dato | Acción | Detalle |
|---|---|---|
| `users` | Anonimizar in situ | Email→hash, nombre→ `[ANONIMIZADO]`, colegiación a NULL, rol→ `ANONYMIZED` |
| `sessions`, `accounts` (Auth.js) | DELETE | Cierre inmediato de toda sesión y cuenta vinculada |
| `verification_tokens` | DELETE | Por `identifier = email original` (tokens OTP pendientes) |
| `intakes` (por email original) | Anonimizar in situ | `descripcion` / `nombre` → `[ANONIMIZADO]`, `email` →hash, `telefono` →'', `next_action_note` y `lost_reason_detail` →NULL |
| `crm_tasks` de sus intakes | Anonimizar in situ | `title` → `[ANONIMIZADO]`, `description` →NULL |
| `crm_presupuestos` | Scrub parcial / retener | `reject_reason`, `decided_ip`, `decided_user_agent` →NULL. Los presupuestos *aceptados* se retienen 6 años como registro contractual (art. 30 CCo); la identidad ya quedó anonimizada vía intake |
| `lawyer_invoices` | RETENER íntegras | Registro fiscal y mercantil obligatorio (arts. 29–30 CCo y normativa de facturación) — 6 años |
| `case_messages` del usuario | Anonimizar in situ | `body` → `[ANONIMIZADO]` (se conserva la estructura del hilo para el resto de partícipes) |
| `case_documents` subidos por el usuario | Borrar fichero + scrub fila | Objeto eliminado (R2 `deleteObject` o `unlink` local, según `storageProvider`); fila con `original_filename` → `[ANONIMIZADO].pdf`, `storage_path` →'' (sentinela «fichero eliminado»; la columna es NOT NULL) y `r2_etag` →NULL para que el worker de retención no re-apunte a una clave ya borrada |
| `case_analyses`, `case_executive_summaries`, `case_clarifications` | DELETE | Análisis de IA 100 % derivados del contenido del caso; sin base de retención propia |
| `case_chat_threads` + `case_chat_messages` | DELETE | Chat del expediente completo (mensajes por `threadId`, hilos por `caseId`) |
| `document_analyses` | DELETE | Contiene las *entidades extraídas* de los documentos: DNI, IBAN, direcciones — PII derivada de máxima sensibilidad |
| `questionnaire_answers` | DELETE | Relato propio del cliente (cadena casos → cuestionarios → preguntas → respuestas) |
| `case_witnesses`, `case_lawyer_inputs` | DELETE | Incluyen PII de terceros (nombre, email, teléfono, ubicación de testigos) y el criterio del letrado; sin base de retención propia |
| `meeting_recordings` + audio R2 | DELETE fila + objeto | La VOZ del cliente + transcripción + resumen: máxima sensibilidad (art. 9/10, secreto profesional). Se borra el objeto `r2_key` y después la fila |
| `causa_analyses` | NO se toca | Análisis *conjunto* de la causa multi-investigado, compartido entre co-investigados y ya seudonimizado («Investigado N»): no contiene el nombre real del cliente, y borrarlo destruiría datos de terceros |
| `case_payments` | RETENER | Obligación de conservación mercantil — art. 30 Código de Comercio, 6 años |
| Herencias: `herencia_documents` (+R2), `herencia_questions` / `_answers`, `herencia_reports` | DELETE | Todo el contenido personal de la consulta sucesoria se purga; la fila de `herencia_consultas` se CONSERVA con scrub (`descripcion` / `parentesco` → `[ANONIMIZADO]`, `telefono` / `nota_interna` / `intake_ip` / `intake_user_agent` →NULL) porque de ella cascan los `herencia_payments` retenidos por art. 30 CCo |
| COLABORADOR: `collaborator_profiles` | Scrub + revocar | `bio`, `photo_url`, `hourly_rate_eur`, motivos de rechazo/suspensión→NULL; `status_onboarding` → `revoked` |
| COLABORADOR: `collaborator_practice_areas` | DELETE | Áreas de práctica completas |
| COLABORADOR: `collaborator_documents` + `case_documents` subyacentes | DELETE + borrar ficheros | Documentación de colegiación (títulos, carnets) eliminada de BD y almacenamiento |
| COLABORADOR: `collaborator_invitations` | Scrub + revocar | `email_snapshot` →hash anónimo, `status` → `revoked` |
| Firmas eIDAS (`signature_requests` firmadas) | RETENER | Reglamento (UE) 910/2014 — validez y auditabilidad de la firma electrónica; 6 años |
| `case_events` | RETENER | Audit log del sistema; los eventos no contienen PII en claro (§8.9) |

### 8.3.3 Qué se retiene y con qué base — explicado al interesado

El derecho de supresión del art. 17 RGPD no es absoluto: el art. 17.3.b lo excepciona cuando el tratamiento es necesario «para el cumplimiento de una obligación legal». `retainedForLegal` materializa esa excepción con literales explicativos que llegan al interesado:

| Dato retenido | Base legal | Plazo | Por qué es legítimo |
|---|---|---|---|
| Registros de pago (`case_payments`, `herencia_payments`) | Art. 30 Código de Comercio | 6 años | Obligación mercantil de conservar libros, correspondencia y justificantes del negocio. Tras la anonimización, los pagos cuelgan de un caso cuyo cliente ya no es identificable |

| Dato retenido | Base legal | Plazo | Por qué es legítimo |
|---|---|---|---|
| Facturas de honorarios emitidas | Arts. 29–30 CCo + normativa de facturación | 6 años | Obligación fiscal y mercantil; prueba frente a AEAT |
| Presupuestos aceptados | Art. 30 CCo (registro contractual) | 6 años | Documento contractual del encargo aceptado |
| Firmas electrónicas eIDAS completadas | Reglamento (UE) 910/2014 (art. 32: validación de firmas) | 6 años | La evidencia de firma (hash del documento, sellado temporal) debe poder validarse mientras el documento firmado despliegue efectos |
| `case_events` (audit trail) | Interés legítimo / integridad del sistema | Vida del sistema | Sin PII del interesado en claro; necesario para la trazabilidad forense, incluida la del propio proceso DSR |

### 8.3.4 Orden de operaciones y garantías

```
anonymizeUser(userId)
─────────────────────
 1. Cargar usuario · role==='ANONYMIZED'? → DsrAnonymizationFailedError (idempotencia)
 2. ┌──────────────── db.transaction ────────────────┐
    │ identidad → sesiones → tokens → herencias → rama CLIENT      │
    │ (intakes/CRM/mensajes/documentos/IA/testigos/grabaciones)    │
    │ o rama COLABORADOR (perfil/áreas/docs/invitaciones)          │
    │ · Los ficheros NO se borran aquí: solo se ACUMULAN en        │
    │   filesToDelete[{storagePath, storageProvider}]              │
    └──────────────────────────────────────────────┘
 3. FUERA de la transacción: por cada fichero acumulado
      provider==='r2' → getR2().deleteObject({bucket:'prod', key})
      provider local  → unlink(path)
    Fallos → logger.warn, NUNCA lanzan (la anonimización de BD ya es firme)
 4. return { categoriesAnonymized, filesDeleted, retainedForLegal }
```

Fig. 8.2 — Secuencia de `anonymizeUser()`: transacción atómica en BD; borrado de almacenamiento fuera de ella.

⚠ **Irreversible por diseño.** El runbook `docs/runbooks/11-dsr-rgpd.md` lo dice sin ambigüedad: restaurar un usuario anonimizado **NO ES POSIBLE**. No hay soft-delete, no hay papelera, no hay copia lateral. Si la ejecución fue un error operacional, el único camino es crear una cuenta nueva con otro `userId` y comunicar al usuario la pérdida del historial. Esa irreversibilidad es una *propiedad de cumplimiento*, no un defecto: una anonimización reversible sería una seudonimización, y el art. 17 exige supresión efectiva. Por eso el gate de aprobación admin (§8.2.2) y la preview de impacto existen: la decisión se toma *antes*, con los números delante.

ℹ **Por qué los ficheros se borran fuera de la transacción.** Las operaciones de filesystem y de R2 no participan en el rollback de MariaDB. Si se borraran dentro y la transacción fallara después, habría documentos destruidos con filas intactas apuntando a ellos — el peor estado posible. El orden elegido garantiza que la BD queda anonimizada de forma atómica y los objetos huérfanos restantes (si algún delete falla) no identifican a nadie: son bytes sin fila, y el fallo queda registrado para limpieza manual.

## 8.4 Retención y purga automática de datos

El principio de limitación del plazo de conservación (art. 5.1.e RGPD) está automatizado en el **worker de retención**: `apps/portal/src/lib/document-retention-worker.ts`, arrancado por `apps/portal/scripts/retention-worker-entry.ts` como proceso pm2 `jbhlegal-retention-worker` con `cron_restart` diario a las 02:00 (patrón «corre una vez y sale»; pm2 lo relanza cada día — ejecución verificada en producción el 2026-07-20). El proceso reporta errores a Sentry si `SENTRY_DSN` está definido y emite logs JSON estructurados por etapa.

### 8.4.1 Matriz de políticas de retención

| Dato | Plazo | Disparador | Mecanismo |
|---|---|---|---|
| Documentos de caso (`case_documents`) | 10 años tras el cierre del caso | `cases.fecha_cierre` | Sello `retention_until = fecha_cierre + 10 años` → soft-delete al vencer → hard-delete del objeto R2 a los +30 días |
| Análisis IA por documento (`document_analyses`) | La del documento padre | Hard-delete del documento | Purga en cascada en el mismo ciclo (PRIV-06) |
| Grabaciones de reuniones (`meeting_recordings`) | 10 años tras el cierre del caso | `cases.fecha_cierre` | Sin soft-delete intermedio: objeto R2 primero, fila después (si R2 falla, la fila queda y se reintenta al día siguiente) |
| Intakes no convertidos (`intakes`) | 12 meses (`INTAKE_RETENTION_DAYS = 365`) | `created_at` + no convertido (`case_id IS NULL` y `status != 'contratado'`) | DELETE en lotes de 100; cascada FK a `intake_events` / `intake_notes` / `intake_documents`; borrado explícito previo de `crm_tasks` y `crm_presupuestos` (FK suave sin constraint) |
| Documentos de intake (`intake_documents`) | 30 días (`PENDING_MAX_AGE_DAYS`) si el intake sigue `pending/nuevo/asignado`; **7 días** (`DESCARTADO_MAX_AGE_DAYS`) si `descartado` | `intakes.created_at` (proxy conservador: solo puede alargar la ventana) | Delete R2 (bucket `prod` si `uploadStatus='ready'`, `quarantine` en cualquier otro estado) + `deleted_at`. Nunca toca documentos promovidos (`promoted_to_case_doc_id IS NOT NULL`): esos ya se rigen por la política de caso |

| Dato | Plazo | Disparador | Mecanismo |
|---|---|---|---|
| Reservas de visitantes (`bookings`) | 12 meses (`BOOKING_RETENTION_DAYS = 365`) | `slot_end` + `client_user_id IS NULL` + `case_id IS NULL` | DELETE en lotes; `booking_audit_events` cae por FK `ON DELETE CASCADE` |
| ZIPs de export DSR | 7 días | `artifact_expires_at` | Worker `dsr-cleanup-worker.ts` cada 24 h (§8.2.4) |
| Backups nocturnos en R2 | 30 días | Timestamp del objeto | Poda automática del script de backup (§8.8) |

### 8.4.2 El ciclo diario

```
pm2 jbhlegal-retention-worker · cron 02:00 diario
──────────────────────────────────────────────────
runRetentionCycle()
 │
 ├─1. UPDATE case_documents ⋈ cases
 │     SET retention_until = fecha_cierre + 10 AÑOS   (solo casos cerrados sin sello)
 │
 ├─2. UPDATE case_documents SET deleted_at = NOW()
 │     WHERE retention_until < NOW()                  (soft-delete: 30 días de gracia)
 │
 ├─2b. meeting_recordings ⋈ cases: fecha_cierre+10a < NOW()
 │      → deleteObject(R2) → DELETE fila              (sin soft-delete; reintento diario)
 │
 ├─3. case_documents con deleted_at < NOW()-30d ∧ provider='r2' ∧ r2_etag≠NULL
 │      → deleteObject(R2) → r2_etag = NULL           (hard-delete definitivo)
 │      → DELETE document_analyses (documentId IN expirados)   [PRIV-06]
 │
 ├─4. purgeStaleIntakeDocuments()   30d pendiente / 7d descartado → R2 + deleted_at
 ├─5. purgeStaleIntakes()           >12 meses sin convertir → DELETE + cascadas
 └─6. purgeStaleBookings()          >12 meses visitante sin caso → DELETE + cascada audit
```

Fig. 8.3 — Ciclo diario del worker de retención (todas las etapas con log estructurado `retention.*`).

ℹ **Barrido global multi-tenant por diseño.** El worker procesa los documentos de *todos* los tenants en cada ciclo: la política de conservación es idéntica para todos y no existe sesión/tenant activo en un cron. El SELECT proyecta `org_id` únicamente para trazabilidad en logs; las cascadas posteriores se acotan por los IDs de documento ya seleccionados, que están org-acotados por su fila padre. Es la excepción documentada a la regla general de scoping por `org_id`.

✓ **Ventana de gracia de 30 días.** El soft-delete intermedio de los documentos de caso protege contra el cierre accidental de un expediente: si un caso se cierra por error y el sello de retención de un documento antiguo vence, hay 30 días para reabrir antes de que el objeto R2 desaparezca de forma definitiva. Las purgas de R2 son *best-effort con reintento*: un fallo de borrado se loguea y no bloquea el resto del ciclo, y un objeto huérfano en el bucket es siempre preferible a una fila atascada reintentada eternamente.

## 8.5 Seudonimización hacia la IA: `redact-pii.ts`

Antes de que el contenido de un caso viaje a la API de Anthropic, la plataforma aplica una capa de **seudonimización de nombres** implementada en `packages/ai/src/redact-pii.ts`. El módulo expone dos factorías simétricas:

| Función | Firma | Comportamiento |
|---|---|---|
| `buildNameRedactor(roster)` | `(text) => string` | Sustituye cada nombre real del roster por su etiqueta seudónima («Investigado 1», «Cliente», «Testigo 2»…) en el texto que va a salir hacia el modelo. Case-insensitive y a prueba de metacaracteres de regex (`escapeRegex`) |
| `buildNameRestorer(roster)` | `(text) => string` | Operación inversa sobre la *salida* del modelo: reemplaza las etiquetas por los nombres reales, procesando las etiquetas más largas primero para no cortar parciales («Investigado 10» antes que «Investigado 1») |

El tipo `RedactionRoster` es una lista de pares `{ name, label }` con el censo de personas del caso: cliente, víctimas, testigos y coimputados. Los nombres vacíos o de menos de 3 caracteres se ignoran (`validPairs`) para evitar reemplazos espurios sobre palabras comunes. El patrón de uso es *redact → llamada al modelo → restore*:

```
const redact  = buildNameRedactor(roster);   // «Juan Pérez» → «Investigado 1»
const restore = buildNameRestorer(roster);   // «Investigado 1» → «Juan Pérez»

const prompt   = redact(textoDelCaso);        // lo que viaja a Anthropic
const salida   = await generateMarkdownDraft(…);
const informe  = restore(salida);             // lo que ve el letrado
```

La seudonimización nació en el flujo de **CAUSA multi-investigado** — el más sensible de la plataforma, donde varios co-investigados comparten un análisis conjunto y cada uno aparece como «Investigado N» — y se ha extendido a todos los flujos que envían datos del caso a Anthropic (análisis de documentos, generación de escritos, cuestionario, intake) cuando se les pasa un roster. Es **opcional por llamada con firmas estables**: si no hay roster, el comportamiento del generador no cambia. Este diseño tiene además un efecto estructural valioso: el análisis conjunto persistido en `causa_analyses` queda seudonimizado *en reposo*, lo que permite conservarlo cuando un co-investigado ejerce su derecho de supresión (§8.3.2) sin afectar a los demás.

⚠ **Defensa en profundidad, no control primario (PRIV-07).** El control de privacidad *primario* frente a Anthropic es el Zero Data Retention contractual a nivel de cuenta (§8.6). La seudonimización es una capa *adicional*: incluso bajo ZDR, reduce la superficie de exposición durante el procesamiento y protege frente a errores humanos (un log accidental, un prompt copiado a un ticket). La regla operativa del propio código es explícita: si la cuenta dejara de tener ZDR/enterprise, habría que dejar de enviar contenido de casos o seudonimizarlo íntegramente — la seudonimización pasaría de refuerzo a requisito.

## 8.6 Privacidad frente a Anthropic: ZDR contractual y BYOK

El cliente único de Anthropic vive en `packages/ai/src/anthropic-client.ts` (`createAnthropicClient()`): toda llamada del monorepo pasa por esta factoría, lo que centraliza timeout acotado (240 s frente a los ~10 min por defecto del SDK), reintentos, fallback de modelo y — lo relevante aquí — la política de privacidad.

### 8.6.1 Qué es (y qué no es) el Zero Data Retention

El **Zero Data Retention (ZDR)** de Anthropic *no se activa con una cabecera por petición*: es una configuración **contractual a nivel de cuenta/organización**, parte del acuerdo enterprise con DPA y con Cláusulas Contractuales Tipo (SCC) para la transferencia internacional. El comentario de cabecera del propio `createAnthropicClient()` lo documenta como decisión de diseño: la función *no fija ninguna cabecera de retención* porque «no existe una cabecera zero-retention»; la garantía la aplica Anthropic sobre la cuenta cuyas credenciales usa `ANTHROPIC_API_KEY` (valor out-of-band, en `shared/*.env` del servidor). En la práctica significa que Anthropic **no conserva el contenido enviado una vez servida la respuesta**: los prompts y salidas no se almacenan ni se usan para entrenamiento. La política de privacidad publicada declara exactamente este marco al interesado («SCCs + retención cero (ZDR)»).

⚠ **No añadir «cabeceras ZDR» inventadas.** Cualquier intento de «activar» ZDR por petición con una cabecera es un error de bulto que da falsa sensación de seguridad: el SDK la ignoraría o la API la rechazaría, y el desarrollador creería haber protegido datos que viajan bajo el régimen que tenga la cuenta. La única acción válida ante un cambio del marco contractual es la documentada en el código: dejar de enviar contenido de casos o seudonimizarlo por completo.

### 8.6.2 BYOK: la clave del despacho, el marco del despacho

En el modelo multi-tenant B, un despacho puede configurar su **propia clave de API de Anthropic** (BYOK, *bring your own key*). La factoría resuelve la clave por contexto: `apiKey: contextApiKey() ?? process.env.ANTHROPIC_API_KEY` — si el flujo corre dentro de `enterOrgAiContext` de una organización con clave propia, se usa *su* clave (y consume *sus* créditos); si no, la de JBH. La consecuencia de cumplimiento es directa: con BYOK, la relación de encargo con Anthropic respecto de esos prompts se establece bajo la cuenta *del despacho*, con el marco contractual (y el ZDR, o su ausencia) que ese despacho haya pactado. Es responsabilidad del despacho que su cuenta ofrezca garantías equivalentes; la plataforma mantiene en todo caso la seudonimización como defensa en profundidad.

## 8.7 Consentimiento versionado y con valor probatorio

### 8.7.1 El consentimiento del intake

El endpoint público de captación `apps/web/src/app/api/public/intakes/route.ts` («Evaluar mi caso») implementa la captura probatoria del consentimiento (control C-6 de la auditoría de privacidad). En el esquema Zod del body, `consentPrivacidad: z.literal(true)` — no basta un booleano: el literal obliga a que el checkbox venga marcado o la petición falla con error de validación; el consentimiento no puede «colarse» como false ni omitirse. Junto a él, `consentComunicaciones` es un opt-in *separado* con `default(false)`: las comunicaciones comerciales jamás se presumen (granularidad exigida por el art. 7 RGPD).

Al persistir el intake se escribe el expediente probatorio completo en `packages/db/src/schema/intakes.ts`:

| Columna de `intakes` | Contenido | Valor probatorio |
|---|---|---|
| `consent_privacidad` | `true` (obligatorio) | Hecho del consentimiento |
| `consent_privacidad_at` | Timestamp del servidor en el momento del alta | Cuándo se prestó |
| `privacy_policy_version` | Constante `PRIVACY_POLICY_VERSION` (valor vigente: `'2026-05-26'`) | *Qué texto exacto* aceptó el interesado. La constante se incrementa a la vez que la página de la política cada vez que cambia su redacción |
| `consent_comunicaciones` | Opt-in independiente, default false | Base del art. 6.1.a para marketing, separable y revocable |
| `consent_datos_penales` | Booleano *nullable* (andamiaje C-6) | Reservado para el checkbox específico de categoría especial que el DPO debe redactar antes de poblarlo |
| `ip_address` | Primera IP de `x-forwarded-for` | Origen técnico de la declaración |
| `user_agent` | User-Agent truncado a 500 caracteres | Contexto técnico del dispositivo |

La combinación *versión de política* + *timestamp* + *IP* + *user-agent* permite reconstruir, ante la AEPD o en juicio, exactamente qué aceptó cada interesado, cuándo y desde dónde — el estándar de «consentimiento demostrable» del art. 7.1 RGPD. El mismo patrón se replica en el alta de consultas de herencias (`apps/herencias/src/app/api/public/consultas/route.ts`), y esos campos de rastro (`intake_ip`, `intake_user_agent`) se purgan en la anonimización (§8.3.2), porque su función probatoria decae con la relación.

### 8.7.2 Consentimiento en grabaciones de reuniones

El módulo de grabaciones (`packages/db/src/schema/meeting-recordings.ts`) lleva el consentimiento *dentro de la propia grabación*: al iniciar, el portal reproduce una **locución de aviso** y pide a cada asistente decir su nombre y su consentimiento en voz alta; la transcripción lo captura y el campo JSON `consent_info` lo estructura — `{ disclaimerDetected: boolean, consents: [{ speaker, name, consented }] }`. La prueba del consentimiento es así inseparable del dato consentido. Dos defaults protegen el privilegio abogado–cliente: `include_in_analysis = false` (la conversación **no entra en los análisis ni escritos de IA** salvo opt-in expreso del letrado, grabación a grabación) y `shared_with_client = false` (visible solo para el equipo del despacho hasta decisión contraria). Audio y transcripción son datos penales a todos los efectos: cifrado en reposo en R2, retención pareja a los documentos del caso (10 años post-cierre, §8.4) y borrado que alcanza siempre al objeto R2, tanto en el DELETE de la ruta como en la supresión DSR.

## 8.8 Backups cifrados y disaster recovery

### 8.8.1 El backup nocturno

La continuidad del dato está garantizada por la tarea nocturna del scheduler de Plesk (id **1038**, cron `30 2 * * *` — 02:30 UTC) que ejecuta `workers/jbh-nightly-backup.sh` en el servidor, apoyado en el helper `workers/jbh-r2.mjs` (cliente S3 SigV4 para R2 en Node puro, sin dependencias). Cada noche produce dos artefactos en el bucket R2 `[bucket principal reservado]`:

| Artefacto | Prefijo R2 | Contenido | Cifrado |
|---|---|---|---|
| `jbhlegal-<STAMP>.sql.gz` | `db-backups/` | `mysqldump` completo (single-transaction, con rutinas, triggers y eventos) comprimido con gzip (~1 MB) | En reposo por R2; el dump viaja por TLS |
| `jbh-env-<STAMP>.tar.gz.enc` | `secrets-backup/` | Los **10 ficheros** `shared/*.env` del servidor (ai-worker, api, av-worker, cliente, email-worker, firmas, herencias, peritos, portal, web) — es decir, TODAS las credenciales de la plataforma | **Cifrado asimétrico** con la clave pública `shared/jbh-dr-cert.pem` (RSA 4096, formato S/MIME DER) |

La **retención es de 30 días** con poda automática sobre los prefijos `db-backups/jbhlegal-` y `secrets-backup/jbh-env-` (los blobs de recuperación manuales `jbh-recovery-local-*` quedan fuera de la poda). El resultado de la última ejecución es consultable sin SSH en un log público ofuscado: `https://jbhasesorialegal.com/_next/static/[ruta reservada].txt`, que debe terminar en `DONE` con `db http=200` y `secrets http=200`.

### 8.8.2 Cifrado asimétrico: el servidor no puede leer sus propios backups de secretos

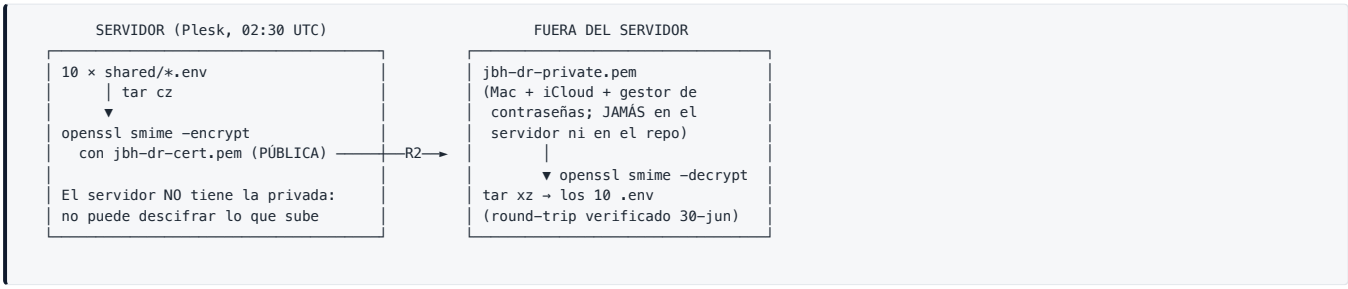


Fig. 8.4 — Esquema de cifrado asimétrico del backup de secretos: quien respalda no puede leer.

La propiedad de seguridad es elegante: el servidor cifra cada noche con la **clave pública** y no almacena ningún secreto de descifrado, de modo que un atacante que comprometa el servidor (o el bucket) obtiene blobs ilegibles. La **clave privada DR** (`jbh-dr-private.pem`) vive exclusivamente out-of-band — equipo local, iCloud y gestor de contraseñas — y es **imprescindible**: sin ella, los backups de secretos son criptográficamente inútiles. El runbook `docs/runbooks/12-disaster-recovery.md` reduce el disaster recovery completo a conservar dos cosas fuera del servidor: (1) el acceso a la cuenta de Cloudflare (para bajar de R2; nótese el círculo documentado — las claves R2 están *dentro* del blob de secretos, por lo que el primer acceso se hace por el dashboard de Cloudflare) y (2) la clave privada DR. Con ambas, la recuperación en un servidor nuevo es un procedimiento de cuatro pasos: bajar los últimos artefactos, descifrar los secretos (`openssl smime -decrypt -inform DER … | tar xz`), restaurar la BD (`gunzip -c … | mysql` con las credenciales del `DATABASE_URL` recuperado) y redesplegar las apps con los `.env` en `shared/`. El procedimiento completo, incluido el descifrado round-trip, fue **ejecutado y verificado el 30-jun-2026**. La rotación del par DR está igualmente pautada: generar par nuevo, subir el certificado a `shared/`, guardar la privada en el gestor y conservar la antigua mientras existan blobs cifrados con ella (≤30 días).

### 8.8.3 La regla «datos penales NUNCA a GitHub»

> ⚠ **Regla innegociable, aprendida de un incidente real.** Ningún dato penal, expediente, informe de caso, dato de cliente ni secreto sale hacia git/GitHub. La plataforma sufrió una fuga real de material sensible al repositorio, que obligó a una purga de historia completa con `git filter-repo` (dos pasadas), a reescribir la rama remota y a mover el material confidencial a una **bóveda local fuera del repositorio**, con un bundle de respaldo pre-purga conservado offline. Desde entonces el `.gitignore` excluye `EXPEDIENTES*/`, `DOCUMENTACIÓN/{SOCIEDAD,CONTRATOS}/` y `.env*`, y el protocolo de commit exige revisar el set staged y escanear secretos antes de cada commit — jamás `git add -A` a ciegas. Los backups siguen la misma lógica: el destino es R2 (encargado declarado, cifrado), nunca un repositorio de código.

## 8.9 Registro de auditoría

### 8.9.1 Eventos DSR en `case_events`

Cada paso del ciclo de vida de una solicitud DSR emite un evento de negocio en `case_events`, con el campo `collaborator_id` reutilizado como *subject_user_id* (el interesado). Son **10 kinds**:

| Flujo | Kinds |
|---|---|
| Export | `dsr_export_requested` · `dsr_export_completed` · `dsr_export_failed` · `dsr_export_downloaded` |
| Delete | `dsr_delete_requested` · `dsr_delete_approved` · `dsr_delete_rejected` · `dsr_delete_in_progress` · `dsr_delete_completed` · `dsr_delete_failed` |

```
-- Histórico DSR completo de un interesado (runbook 11):
SELECT kind, data, created_at, actor_id
FROM case_events
WHERE collaborator_id = '<userId>' AND kind LIKE 'dsr_%'
ORDER BY created_at;
```

Este rastro sobrevive a la anonimización (los eventos no llevan PII en claro) y es la prueba documental de que cada solicitud se atendió, quién la aprobó y en qué plazos — la evidencia que se aporta ante una inspección de la AEPD.

### 8.9.2 `audit_log`: cadena de hash a prueba de manipulación

Por encima de los eventos de negocio, la tabla `audit_log` (`packages/db/src/schema/audit-log.ts`) es el **rastro forense de propósito general**: append-only, por tenant, y con integridad criptográfica verificable. Cada fila registra actor (`actor_id` + `actor_email`, o NULL para acciones del sistema — cron y workers), verbo estable (`action`: `case.create`, `document.download`, `draft.generate`, `member.invite`, `dsr.export`…), entidad (`entity_type`/`entity_id`) y detalle estructurado en `data` — con la regla explícita de *redactar la PII antes de escribir*. La integridad se garantiza con un **hash encadenado**:

```
hash(n) = SHA-256( hash(n-1) || ":" || canonical(fila n) )

seq        — monotónico por org (1,2,3…), UNIQUE(org_id, seq)
prev_hash  — hash de la fila anterior de la MISMA org
hash       — hash de esta fila
```

Alterar, borrar o insertar una fila pasada rompe todos los hashes siguientes de la cadena de esa organización: la manipulación se detecta con una verificación lineal. La escritura pasa *siempre* por `appendAuditLog()` (`lib/audit.ts`), que serializa por org dentro de una transacción para evitar carreras sobre `seq`/`prev_hash`; nunca hay UPDATE ni DELETE sobre la tabla. Cada tenant tiene su cadena independiente, de modo que la verificación de integridad de un despacho no depende (ni expone) la actividad de otro.

> ✓ **Tres capas de trazabilidad complementarias.** `case_events`/`case_milestones` registran los eventos de negocio del expediente (visibles en la pestaña Actividad); `superadmin_audit` registra las acciones transversales de plataforma; y `audit_log` añade la capa forense con integridad criptográfica. Juntas cubren el art. 5.2 RGPD (responsabilidad proactiva): la plataforma no solo cumple, sino que puede *demostrar* que cumple.

## 8.10 Checklist de cumplimiento para toda feature nueva con datos personales

Toda feature que recoja, almacene, muestre, envíe a terceros o borre datos personales debe pasar esta lista *antes* de darse por buena (la skill de revisión `rgpd-datos-penales` del repositorio la operacionaliza). No es burocracia: cada punto corresponde a un control real ya implementado en la plataforma, y saltárselo significa abrir una brecha respecto del estándar existente.

| # | Control | Pregunta de verificación | Referencia en la plataforma |
|---|---|---|---|
| 1 | **Base jurídica** | ¿Qué apartado del art. 6.1 (y, si hay datos penales/especiales, qué habilitación de los arts. 9/10) ampara este tratamiento? ¿Está reflejado en la política de privacidad? | §8.1.2; `legal/privacidad/page.tsx` |
| 2 | **Minimización** | ¿Cada campo nuevo es imprescindible para la finalidad? ¿Se puede derivar, truncar o no guardar? | Patrón `user_agent.slice(0, 500)`; export sin notas internas ajenas |
| 3 | **DSR completo** | ¿El dato nuevo se incluye en el export (`dsr-exporter.ts`: nuevo colector o categoría) Y se borra/anonimiza en `dsr-anonymizer.ts`? Recordar el precedente A11: la brecha típica es cubrir solo una de las dos mitades | §8.2.4, §8.3 |
| 4 | **Retención** | ¿Qué plazo tiene y qué worker lo purga? Un dato sin política de retención es un incumplimiento del art. 5.1.e en diferido | §8.4; `document-retention-worker.ts` |
| 5 | **Terceros** | ¿Sale el dato hacia algún encargado? Si es uno nuevo: DPA + SCC + fila en la tabla de la política de privacidad ANTES de desplegar | §8.1.3 |
| 6 | **Cifrado** | ¿Reposo (R2/MariaDB) y tránsito (TLS) cubiertos? ¿Los secretos nuevos viven out-of-band (`shared/*.env`, Keychain), nunca en el repo? | §8.8; regla 4 del núcleo del CLAUDE.md |
| 7 | **Datos penales hacia IA** | Si el contenido viaja a Anthropic: ¿pasa por `createAnthropicClient()` (cuenta ZDR)? ¿Admite roster de seudonimización? ¿Se ha considerado BYOK? | §8.5, §8.6 |

| # | Control | Pregunta de verificación | Referencia en la plataforma |
|---|---|---|---|
| 8 | **No decisiones automatizadas (art. 22)** | ¿La feature produce efectos jurídicos sin intervención humana? La IA de la plataforma es siempre *asistiva*: el letrado revisa y decide. Ninguna feature puede romper ese principio | Gate admin del delete (§8.2.2); revisión letrada de todo output IA |
| 9 | **Auditoría** | ¿Las acciones relevantes emiten evento (`case_events`) y/o entrada forense (`appendAuditLog()`) — con la PII redactada antes de escribir `data`? | §8.9 |
| 10 | **Versión de política** | Si la feature cambia qué se trata o con quién se comparte: ¿se ha actualizado la política de privacidad Y se ha incrementado `PRIVACY_POLICY_VERSION` para que los consentimientos nuevos queden ligados al texto nuevo? | §8.7.1 |

ℹ **Síntesis del capítulo.** El cumplimiento RGPD de JBH Legal no es una página legal más un propósito: es código en producción. Los derechos del interesado son self-service con máquina de estados y gate humano; la supresión es una anonimización transaccional, idempotente e irreversible que distingue con precisión jurídica qué se destruye y qué se retiene (y por qué); la retención se ejecuta sola cada madrugada; los datos penales viajan a la IA seudonimizados y bajo ZDR contractual; el consentimiento queda probado con versión, timestamp, IP y user-agent; los backups están cifrados de forma que ni el propio servidor puede leerlos; y todo deja un rastro de auditoría con integridad criptográfica. Para el cliente — un cliente cuyo dato es la categoría más sensible que existe — esa es la diferencia entre prometer privacidad y poder demostrarla.

# 9
# Motor de IA: el Letrado IA

> *El Letrado IA es el corazón del producto: un motor de generación jurídica construido sobre la API de Anthropic que analiza expedientes penales completos, redacta escritos procesales, calcula penas y prescripciones, y asiste al abogado con ~45 herramientas especializadas. Este capítulo documenta su arquitectura (`packages/ai` + `apps/ai-worker`), la selección y resiliencia de modelos, el arsenal de generadores, la cola asíncrona de trabajos, las defensas de seguridad de prompts y el modelo de facturación por consumo.*

## 9.1 Arquitectura general

Todo el código de IA vive en dos piezas del monorepo. El paquete `packages/ai` es la biblioteca compartida: contiene el cliente de Anthropic, la selección de modelos, la tabla de tarifas, los ~60 módulos `generate-*.ts` / `analyze-*.ts`, los system prompts (`packages/ai/src/prompts/`, ~60 archivos), los schemas Zod de salida (`schemas.ts`, `schemas-herencia.ts`, `schemas-trial-prep.ts`), la seudonimización (`redact-pii.ts`), el grounding de citas (`citation-grounding.ts`) y el subsistema RAG de jurisprudencia (`rag/`). La aplicación `apps/ai-worker` es el proceso de larga duración (pm2 `jbhlegal-ai-worker`) que ejecuta los análisis pesados de forma asíncrona a través de una cola de jobs en MariaDB.

Los flujos de IA se dividen en dos familias según su latencia:

- **Síncronos** (petición HTTP → respuesta): la generación de escritos desde el portal (`apps/portal/src/app/api/cases/[id]/escritos/[kind]/route.ts`), el chat con el expediente (streaming SSE), el triaje del evaluador público, la auditoría jurídica de un borrador.
- **Asíncronos** (encolados vía `packages/ai/src/enqueue.ts` y procesados por el worker): análisis por documento, síntesis del caso, análisis de causa multi-investigado, generación y evaluación del cuestionario, informe sucesorio, transcripción de grabaciones, catalogación de precedentes, preparación de vistas.

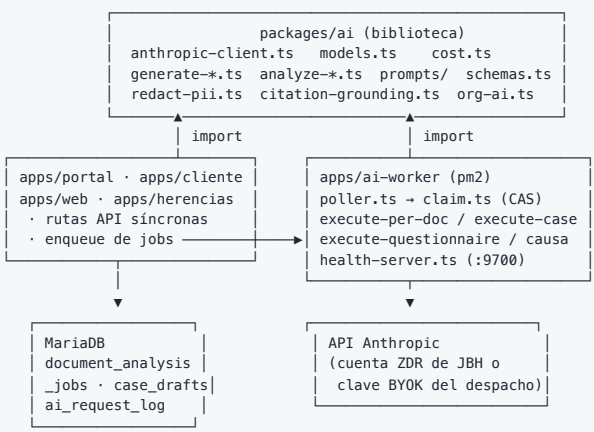


Fig. 9.1 — Arquitectura del motor de IA: biblioteca compartida, apps consumidoras y worker asíncrono.

## 9.2 Cliente Anthropic: factoría única y helpers de generación

### 9.2.1 `createAnthropicClient()`

Ninguna pieza del sistema instancia `new Anthropic(...)` a mano: la factoría única `createAnthropicClient()` (`packages/ai/src/anthropic-client.ts`) centraliza tres decisiones críticas:

1. **Timeout acotado**. El SDK usa por defecto ~10 minutos y además reintenta los timeouts, con lo que una llamada colgada podría bloquear un slot del worker durante mucho más. La factoría fija `DEFAULT_ANTHROPIC_TIMEOUT_MS = 240_000` (4 min) y `maxRetries: 2`; el worker ya reintenta los jobs fallidos, así que fallar pronto y liberar el slot es más seguro que colgarse.
2. **BYOK**. La clave se resuelve como `contextApiKey() ?? process.env.ANTHROPIC_API_KEY`: si el flujo corre bajo el contexto de una organización con clave propia (§9.7.4), se usa la del despacho; si no, la de JBH. El valor de `ANTHROPIC_API_KEY` vive out-of-band (`shared/*.env` del servidor).
3. **Fallback de modelo transparente**. La factoría envuelve `messages.create` con `withModelFallbackClient`: si el modelo pedido falla por estar deprecado/retirado/desconocido, la llamada se reintenta automáticamente con un modelo similar (§9.3.3). Las llamadas con `stream: true` no se envuelven (el error de modelo aflora durante la iteración); los generadores de streaming aplican el fallback explícitamente.

> ℹ **Zero Data Retention.** El ZDR de Anthropic *no* se activa con una cabecera por petición: es una configuración contractual a nivel de cuenta (acuerdo enterprise + DPA con SCC). Por eso el cliente no fija ninguna «cabecera de retención» (no existe). La garantía la aplica Anthropic sobre la cuenta cuyas credenciales usa `ANTHROPIC_API_KEY`. Si la cuenta dejara de tener ZDR, habría que dejar de enviar contenido de casos (datos penales, art. 10 RGPD) o seudonimizarlo íntegramente.

#### 9.2.2 `generateMarkdownDraft()` y `generateJson()`

Sobre el cliente se apoyan dos helpers de generación que los generadores reutilizan en lugar de llamar al SDK directamente:

- `generateMarkdownDraft(system, user, model, maxTokens, roster?, jurisprudencia?)` (`packages/ai/src/generate-legal-draft.ts`) es el núcleo de todos los borradores largos (escritos ESC-*, informes periciales). Fija `DRAFT_MAX_TOKENS = 16000` (8000 producía escritos truncados, hallazgo crítico de auditoría) y un timeout ampliado `DRAFT_TIMEOUT_MS = 360_000` (240 s se quedaba corto cuando Opus escribe hasta el tope de salida densa). Añade a todo system prompt la `FORENSIC_QUALITY_NOTE` (el mismo catálogo de calidad con el que después audita el validador penalista·fiscal·juez), acepta un `RedactionRoster` opcional para seudonimizar nombres antes de enviar y restaurarlos en la salida, inyecta la jurisprudencia favorable grounded si se aporta, detecta truncamiento por `stop_reason === 'max_tokens'` (añade una nota visible al pie) y pasa el Markdown resultante por `ensureCitationsMarked` (§9.6.3).
- `generateJson(system, user, schema, maxTokens)` (en `packages/ai/src/marketing.ts`) genera salida JSON validada contra un schema Zod; lo usa la cadena de marketing (planificación de temas, redacción, revisión de seguridad deontológica). El resto de flujos estructurados (análisis de caso/documento, auditoría jurídica, triaje) siguen el mismo patrón: pedir JSON, quitar fences con `stripFences`, parsear con tolerancia (`parse-analysis-json.ts`) y validar con el schema Zod correspondiente de `schemas.ts`; un JSON malformado lanza `MalformedAnalysisError` y el worker lo trata como error reintentable.

#### 9.2.3 Prompt caching

`anthropic-client.ts` expone además dos helpers de caché de prompts de Anthropic: `cacheableSystem(text, ttl='1h')` marca el system prompt como cacheable (pequeño y muy reutilizado entre re-análisis y entre casos del mismo tipo) y `cacheableUserBlocks(cachedText, restText, ttl='5m')` parte el contenido de usuario en un bloque estable cacheado (p. ej. los análisis de documentos) y un resto variable. La función `billableInputTokens(usage)` pondera los tokens para que `computeCost` siga siendo exacto: entrada normal ×1, escritura de caché ×1.25, lectura de caché ×0.10 — el panel de consumo refleja el ahorro real sin tocar la tarifa.

## 9.3 Selección de modelos: «cada opción su campeón»

#### 9.3.1 Constantes y mapa por scope

`packages/ai/src/models.ts` es la única fuente de verdad de la selección de modelo. Define cuatro constantes, todas con override por entorno sin desplegar (solo se acepta un id de familia conocida opus/sonnet/haiku/fable, para que la tarificación por familia no se rompa; un valor inválido mantiene el default):

| Constante | Default | Override env | Uso |
|---|---|---|---|
| `FLAGSHIP_MODEL` | `claude-opus-4-8` | `AI_FLAGSHIP_MODEL` | Flujos directos no tarificados por plan: triaje público, auditoría jurídica, chat con el expediente, jurisprudencia, herencias. |
| `PAID_FLAGSHIP_MODEL` | `claude-opus-4-8` | `AI_PAID_FLAGSHIP_MODEL` | Campeón de las cuentas de pago (tier `paid` + tenant JBH) en los scopes sensibles a la calidad. Históricamente Fable 5; cambiado a Opus 4.8 el 2026-07-07 por decisión de coste (la síntesis de caso en Fable 5, ~58k tokens de salida, disparó el consumo). Para reactivar Fable 5 basta el env o volver al literal. |
| `DRAFT_MODEL` / `ECONOMY_MODEL` | `claude-sonnet-4-6` | `AI_DRAFT_MODEL` / `AI_ECONOMY_MODEL` | Embudo gratuito de alto volumen (lead scoring de intakes) y scopes de calidad visible del plan free. |
| `HAIKU_MODEL` | `claude-haiku-4-5-20251001` | `AI_HAIKU_MODEL` | Análisis documental masivo del plan gratuito y valoración de favorabilidad de precedentes en free. |

`FLAGSHIP_MODEL` es un literal a propósito: TypeScript lo estrecha a `SupportedModel`, de modo que el compilador se niega a construir con un flagship que no esté tarifado en `cost.ts` — subir el modelo obliga a añadir su precio y el panel de consumo nunca lee un coste cero o erróneo.

La función `pickModel(scope, tier)` cruza el plan del despacho (`AiTier = 'free' | 'paid'`) con el tipo de operación (`AiScope`: `per-doc`, `case`, `questionnaire`, `answer-eval`, `escrito`, `trial-prep`, `peritaje`, `jurisprudencia` — los mismos valores que registra `ai_request_log.scope`):

| Scope | Tier `paid` (y tenant JBH) | Tier `free` | Racional |
|---|---|---|---|
| `per-doc` (escaneo por documento) | Sonnet | Haiku | Extracción: el coste se multiplica por nº de documentos; Sonnet reduce ~80% el coste del scope más caro sin resentir la calidad. |
| `case` (síntesis del caso) | Campeón de pago | Sonnet | Informe de análisis: sensible a la calidad. |
| `questionnaire` | Campeón de pago | Sonnet | El cuestionario moldea el relato del cliente: no se abarata. |
| `answer-eval` | Campeón de pago | Sonnet | Detecta contradicciones en las respuestas: sensible a la calidad. |
| `escrito` | Campeón de pago | Sonnet | Escritos legales: calidad máxima. |
| `trial-prep` | Campeón de pago | Sonnet | Moldea la declaración real del cliente en vista. |
| `peritaje` | Sonnet | Sonnet | Borrador que revisa un perito humano. |
| `jurisprudencia` | Sonnet | Haiku | Pre-filtro de favorabilidad con red de seguridad (el letrado acepta/rechaza). |

> ⚠ **Regla de negocio «cada opción su campeón».** Fable 5 se reserva a las cuentas de pago vía `PAID_FLAGSHIP_MODEL` / `pickModel`; nunca debe asignarse al plan gratuito ni a los flujos directos (que usan el flagship Opus). El triaje público de captación no pasa por `pickModel`: usa `DRAFT_MODEL` (Sonnet), el modelo barato del embudo gratuito de alto volumen, donde no se requiere profundidad del 100%.

#### 9.3.2 `bestLegalReasoningModel()`

El módulo estratégico crítico «Claves del caso» (y futuros módulos de estrategia) no usa `pickModel` sino `bestLegalReasoningModel()`: lee `BEST_LEGAL_REASONING_MODEL` (o el alias `LEGAL_STRATEGY_MODEL`) y acepta cualquier modelo de familia conocida, incluidas versiones nuevas aún no listadas (el coste se infiere por familia); si el env no es válido, cae al campeón de pago. Este módulo *nunca* usa un modelo económico. El endpoint de escritos lo aplica así: `const model = kind === 'claves_del_caso' ? bestLegalReasoningModel() : pickModel('escrito', tier)`.

#### 9.3.3 Resiliencia ante la evolución de modelos

Los modelos de Anthropic evolucionan: salen versiones superiores y se retiran las obsoletas. El sistema incorpora tres mecanismos (introducidos el 2026-07-11) para no quedarse anclado ni caerse:

1. **Tarifa por familia never-throw** (`cost.ts`). `PRICING_PER_MTOKEN_USD` lista las tarifas exactas (Opus 4.7/4.8 y Fable 5: 15/75 USD por Mtok entrada/salida; Sonnet 4.6: 3/15; Haiku 4.5: 1/5 — la de Fable 5 está marcada como estimada en `ESTIMATED_PRICING_MODELS`). Para modelos no listados, `resolveModelPricing()` infiere la tarifa por familia con `FAMILY_PRICING` (regex `opus|fable` → flagship, `sonnet` → intermedio, `haiku` → económico): un `claude-opus-4-9` puesto por env se tarifa como opus y `computeCost` nunca lanza con un modelo de familia conocida. `computeCostSafe` devuelve `null` en vez de lanzar para modelos totalmente desconocidos (se registran los tokens aunque no el coste).
2. **Overrides por entorno** (`AI_FLAGSHIP_MODEL`, `AI_PAID_FLAGSHIP_MODEL`, `AI_DRAFT_MODEL`, `AI_ECONOMY_MODEL`, `AI_HAIKU_MODEL`, `BEST_LEGAL_REASONING_MODEL`): cada modelo se sube a la última versión sin desplegar código.
3. **Cadenas de fallback automático**. `MODEL_FALLBACKS` define alternativas explícitas por modelo, similares en coste y calidad (Opus 4.8 → Opus 4.7 → Fable 5 → Sonnet; Sonnet → Opus 4.8 → Haiku; Haiku → Sonnet); `familyFallbacks()` cubre ids nuevos no listados; `fallbackChain(model)` compone la cadena ordenada con deduplicación y red de seguridad final (flagship → economy), y `withModelFallback(models, run)` la recorre. El fallback *solo* se dispara ante un error de «modelo no disponible» (`isModelUnavailableError`: `not_found_error`, 404, o 400 con mensaje de modelo deprecado/retirado/inválido); rate-limit (429), timeout, sobrecarga (529) y errores de red se reintentan con el mismo modelo (SDK/worker). La respuesta de Anthropic incluye `.model` — el modelo realmente usado — y los generadores que registran coste tarifan `resp.model`, no el pedido.

## 9.4 El arsenal: ~45 generadores del Letrado IA

El catálogo de herramientas del abogado se mantiene en `docs/roadmap-herramientas-abogado.md` (documento vivo, con IDs por familia). La mayoría de herramientas de generación de texto son «kinds» del registro `GENERATORS` del endpoint `apps/portal/src/app/api/cases/[id]/escritos/[kind]/route.ts` (~40 kinds), que mapea cada `kind` a su función `generate-*.ts` de `packages/ai`; los resultados se persisten versionados en `case_drafts` (+ snapshots por versión, validación humana `validatedAt` y veredicto de auditoría `auditJson` / `auditStatus`). Todos reciben los mismos `DraftGenArgs` (área, categoría penal, posición del cliente, relato, diagnóstico, estrategia, documentos, party graph, timeline, debilidades/fortalezas, situación de libertad, fase procesal, tipo de procedimiento, Q&A del cuestionario, hallazgos documentales, precedentes de biblioteca, testigos) y el modelo elegido por plan. La invariante BA-5 (`kindAllowedForPosition`) impide generar escritos de acusación en un caso de defensa y viceversa.

| Familia | ID / kind | Herramienta | Módulo en `packages/ai/src` |
|---|---|---|---|
| Escritos (ESC) | ESC-01 · `escrito_defensa` | Escrito de defensa (calificación provisional/definitiva) | `generate-escrito-defensa.ts` |
| | ESC-02 · `escrito_proposicion_prueba` | Proposición de prueba (+ `proposicion_testifical` con testigos del caso) | `generate-proposicion-prueba.ts`, `generate-proposicion-testifical.ts` |
| | ESC-03 · `escrito_recurso` | Recursos (reforma/apelación/casación) | `generate-escrito-recurso.ts` |
| | ESC-04 · `escrito_oposicion_acusacion` | Oposición/alegaciones al escrito de acusación | `generate-oposicion-acusacion.ts` |
| | ESC-05 · `guion_interrogatorio` | Guion de interrogatorio y contrainterrogatorio | `generate-guion-interrogatorio.ts` |
| | ESC-06 · `habeas_corpus` | Habeas corpus / diligencias urgentes de detención | `generate-habeas-corpus.ts` |
| | ESC-07 · `minuta_visita` | Minuta/acta de visita (sin citas legales) | `generate-minuta-visita.ts` |
| | ESC-08 + `escrito_recusacion` | Escritos coordinados multi-investigado (vía causas) y recusación | `generate-estrategia-coordinada.ts`, `generate-escrito-recusacion.ts` |
| Cautelares y libertad | `oposicion_prision_provisional` | Oposición a la prisión provisional | `generate-escritos-cautelares.ts` |
| | `solicitud_libertad_provisional` | Solicitud de libertad provisional | |
| | `recurso_auto_prision` | Recurso contra el auto de prisión | |

| Familia | ID / kind | Herramienta | Módulo en `packages/ai/src` |
|---|---|---|---|
| Jurisprudencia (JUR) | JUR-01 | Verificador determinista de citas (BOE/CENDOJ, ECLI) — sin IA, en `@jbhlegal/ui` + `citation-grounding.ts` | `citation-grounding.ts` |
| | JUR-02 · `busqueda_jurisprudencia` | Búsqueda/aplicación de jurisprudencia favorable (motor de 4 fases + RAG) | `generate-busqueda-jurisprudencia.ts`, `jurisprudencia-favorable.ts`, `rag/`, `cendoj/` |
| Prueba (PRU) | PRU-01 | Auditor del cuestionario pre-envío | `audit-questionnaire.ts` |
| | PRU-02 · `detector_contradicciones` | Detector de contradicciones relato↔respuestas↔documentos | `generate-detector-contradicciones.ts` |
| | PRU-03 · `mapa_hechos_prueba` | Mapa hechos ↔ prueba | `generate-mapa-hechos-prueba.ts` |
| | PRU-04 · `banco_descargo` | Banco de prueba de descargo | `generate-banco-descargo.ts` |
| | PRU-05 · `lagunas_probatorias` / PRU-06 · `nulidades_prueba_ilicita` | Inspector de lagunas probatorias · Nulidades y prueba ilícita (art. 11.1 LOPJ) | `generate-lagunas-probatorias.ts`, `generate-nulidades-prueba-ilicita.ts` |
| | PRU-07 | Matriz de contradicciones entre investigados (causa; seudonimizada «Investigado N») | `detect-comparativa-contradicciones.ts` |
| Estrategia (EST) | EST-01 / EST-09 | Calendario de plazos + alertas · Timeline procesal unificada (sin IA) | — (portal, `case_tasks` / `case_milestones`) |
| | EST-02 · `calculadora_pena` | Calculadora de pena: marco, individualización art. 66 CP, horquilla, margen de conformidad | `generate-calculadora-pena.ts` |
| | EST-03 · `calculadora_prescripcion` | Prescripción del delito: plazo art. 131 CP, dies a quo, interrupciones | `generate-calculadora-prescripcion.ts` |
| | EST-04 · `simulador_escenarios` / EST-07 · `asistente_conformidad` | Simulador juicio/conformidad/sobreseimiento · Asistente de conformidad (atenuantes 21.4/21.5/21.6 CP) | `generate-simulador-escenarios.ts`, `generate-asistente-conformidad.ts` |
| | EST-05 · `teoria_del_caso` / EST-06 · `red_team_acusacion` | Constructor de teoría del caso · Red-team (anticipa el ataque del fiscal) | `generate-teoria-del-caso.ts`, `generate-red-team-acusacion.ts` |
| | EST-08 · `checklist_fase_procesal` + `claves_del_caso` | Checklist por fase (instrucción/intermedia/juicio/recursos) · **Claves del caso** (14 apartados, enfoque auto-invertido por `posicionCliente`, modelo `bestLegalReasoningModel()`) | `generate-checklist-fase-procesal.ts`, `generate-claves-del-caso.ts`, `generate-claves-acciones.ts` |
| Espejo de ACUSACIÓN | `denuncia`, `querella`, `escrito_acusacion_particular`, `solicitud_medidas_proteccion` | Escritos del cliente en posición de acusación | `generate-escritos-acusacion.ts` |
| | `analisis_tipicidad`, `red_team_defensa`, `banco_cargo`, `lagunas_cargo`, `blindaje_probatorio`, `detector_contradicciones_acusacion`, `peticion_pena_acusacion`, `checklist_fase_acusacion` | Arsenal analítico espejo, gateado a posición=acusación | `generate-analisis-acusacion.ts` y afines |
| Despacho (PRO) | PRO-01 | Chat con el expediente (full-context, streaming SSE, hilos en `case_chat_threads`) | `case-dossier.ts`, `stream-case-chat.ts` |
| | PRO-02 · `comparador_versiones` / PRO-03/04/05 | Comparador de versiones del relato · Mapa de coordinación de causa · Evolución del pronóstico · Panel de aclaraciones | `generate-comparador-versiones.ts` (+ endpoints del portal) |
| Herencias / ISD | — | Cuestionario e informe sucesorio, extracción de inventario, motor ISD verificado (estatal + autonómico), atipicidad JUR-03 (`analisis_atipicidad`) | `generate-herencia-*.ts`, `extract-herencia-inventario.ts`, `isd/`, `ccaa.ts` |
| | — | Peritajes: borrador de informe pericial | `generate-informe-pericial.ts`, `peritajes/` |

Fuera del registro de escritos operan además los analizadores del worker (`analyze-document.ts`, `analyze-case.ts`, `analyze-causa.ts`, `analyze-from-text.ts`), la preparación de vistas (`generate-trial-prep.ts` + simulador de interrogatorio), la sugerencia de testigos (`generate-witness-suggestions.ts`), el acuerdo de defensa conjunta y el conflicto de interés en causas (`generate-acuerdo-defensa-conjunta.ts`, `generate-conflicto-interes.ts`) y el refinado de borradores (`generate-refine-draft.ts`).

## 9.5 `apps/ai-worker`: la cola asíncrona

### 9.5.1 Cola, claim atómico y ejecución

El worker sondea la tabla `document_analysis_jobs` cada `WORKER_POLL_INTERVAL_MS` (5 s). `claimNextJob()` (`claim.ts`) selecciona el siguiente job `pending` con `attempts < max_attempts` y lo reclama con un CAS atómico (UPDATE condicional que fija `status='processing'`, `locked_at`, `locked_by` e incrementa `attempts`), de modo que varios runners concurrentes reclaman jobs distintos sin colisión. Cada job lleva un `scope` que lo enruta a su executor: `execute-per-doc.ts` (análisis por documento, con transcripción de PDF escaneado por visión), `execute-case.ts` (síntesis map-reduce del caso + jurisprudencia favorable), `execute-causa.ts` (causa multi-investigado, seudonimizada), `execute-questionnaire.ts` / `execute-answer-eval.ts` (cuestionario), `execute-herencia-questionnaire.ts` / `execute-herencia-report.ts`, `execute-precedent.ts` (catalogación de jurisprudencia + indexado RAG), `execute-meeting.ts` (transcripción de grabaciones) y `execute-trial-prep.ts`.

### 9.5.2 Concurrencia y el gotcha 4×4=16

La concurrencia real de llamadas de visión es el *producto* de dos diales: `WORKER_CONCURRENCY` (nº de jobs en paralelo; default 4 en código, acotado 1–8) × `PDF_TRANSCRIBE_CONCURRENCY` (páginas/chunks de un PDF transcritos en paralelo dentro de un job). Con 4×4 = 16 llamadas concurrentes el rate-limit se saturaba y los PDFs escaneados grandes daban timeouts en cascada. La configuración aplicada en producción es `WORKER_CONCURRENCY=3`, `PDF_TRANSCRIBE_CONCURRENCY=2` y `PAGES_PER_CHUNK=3` (páginas por llamada de visión), es decir, 6 llamadas de visión concurrentes como máximo.

> ⚠ **Nunca reiniciar con un análisis en vuelo.** `SIGINT`, `pm2 restart` o un deploy abortan el análisis en curso y lo reencolan (se pierde todo el progreso y el gasto de tokens). El síntoma «no avanza / colgado» es casi siempre falsa alarma: el health se actualiza a nivel de *lote* (solo al terminar el `Promise.all` de los jobs del ciclo), y un documento grande (p. ej. 69 páginas → ~110K caracteres a Opus) puede tardar ~15 min a 0% de CPU (espera de red). Antes de tocar nada: mirar el OUT log real (`pm2 jlist` → `pm_out_log_path`) y la BD; reclamar jobs zombis es una operación solo-BD que no requiere reinicio.

### 9.5.3 Reintentos, backoff, sweep y presupuesto

- **Reintentos**: cada job tiene `max_attempts` (3 por defecto al encolar). Un error permanente (`isPermanentError`) o agotar los intentos marca el job `failed`; un error transitorio lo devuelve a `pending` con backoff exponencial con jitter (`computeBackoffMs`: base 60 s, duplica por intento, techo 30 min, jitter ±20% — anti thundering-herd; los errores transitorios de IA esperan al menos 3 min).
- `sweepStuckJobs()` (`sweep-stuck.ts`): en cada ciclo, devuelve a `pending` (y limpia `locked_at`/`locked_by`) todo job en `processing` cuyo lock supere 10 minutos — rescata jobs de un worker muerto. Lo complementan `reconcileStuckCaseAnalyses`/`reconcileStuckDocAnalyses` (`reconcile-stuck-cases.ts`), que reconcilian los estados de análisis huérfanos en las tablas de negocio.
- **Presupuesto global** (`budget.ts`): antes de reclamar, `budgetExceeded()` suma el `cost_cents` del mes en `ai_request_log` contra `AI_MONTHLY_BUDGET_CENTS` (default 50000 = 500 USD); si se excede, el worker pausa (espera 5 min) y envía una alerta por email (throttle 1 h) a `INTAKE_NOTIFY_TO`.
- **Health server** (`health-server.ts`): HTTP en `127.0.0.1:9700` (`HEALTH_PORT`; `0` desactiva) expone `/health` con el snapshot en memoria (`startedAt`, `lastPollAt`, `lastResult`: `job-done|job-failed|no-job|over-budget|error`); Apache/Plesk lo puede proxyar para Better Stack/Sentry.


Fig. 9.2 — Ciclo de vida de un job del ai-worker.

## 9.6 Seguridad de prompts

El motor procesa datos penales (categoría especial, art. 10 RGPD) y texto libre de clientes potencialmente hostil. La seguridad se construye en cinco capas complementarias, todas en código y verificables.

### 9.6.1 Anti prompt-injection: el texto del usuario son DATOS

`packages/ai/src/prompts/untrusted.ts` define la convención transversal: todo texto libre aportado por el usuario/cliente (relato del intake, respuestas del cuestionario, aclaraciones, transcripciones, contenido de documentos, datos de precedentes subidos) se encierra con `wrapUntrusted(text)` entre los marcadores `<<<DATOS_USUARIO_INICIO>>>` y `<<<DATOS_USUARIO_FIN>>>`, y el system prompt correspondiente incluye la `UNTRUSTED_SYSTEM_NOTE`: el contenido entre marcadores es información a analizar, nunca instrucciones; el modelo debe ignorar cualquier intento de cambiar sus instrucciones, revelar el prompt o alterar el formato de salida. La convención tiene más de 230 puntos de uso en `packages/ai` y las apps (~234 en el barrido de 2026-07-20, que auditó prompt a prompt y corrigió una única fuga: el bloque de situación procesal — `prompts/situacion-procesal.ts` interpolaba texto libre del expediente sin envolver; hoy su helper `add()` pasa todo valor libre por `wrapUntrusted`).

#### 9.6.2 Asistencia lícita: `LAWFUL_ASSISTANCE_NOTE`

`prompts/lawful-assistance.ts` define el guardarraíl ético transversal que se añade a todos los system prompts del Letrado IA: prohibición innegociable de proponer o ayudar a mentir, inventar o fabricar hechos, coartadas o pruebas; alterar o destruir pruebas; manipular o aleccionar testigos o peritos; coordinar declaraciones falsas; presentar denuncias o querellas a sabiendas falsas; ocultar bienes; eludir resoluciones judiciales u obstruir la justicia. Está redactado de forma *neutral de rol*: cubre tanto la posición de defensa («coartada falsa») como la de acusación («denuncia a sabiendas falsa»). Nació porque la cláusula vivía en un único prompt (el asistente de WhatsApp) mientras generadores que incentivan «narrativa persuasiva» o «blindar contradicciones» carecían de freno.

#### 9.6.3 Grounding de citas: verificadas o `[VERIFICAR]`

El riesgo crítico C1 (el LLM inventa artículos, penas y jurisprudencia) se cierra en dos capas:

- **Post-proceso determinista** (`citation-grounding.ts`): `ensureCitationsMarked(text)` detecta toda cita legal (artículos de CP/CE/LECrim/CC/LEC, sentencias STS/SAP/STC, identificadores ECLI) y le inserta el marcador `[VERIFICAR]` si no lo lleva ya en una ventana cercana; `findSuspiciousCitations(text)` reporta las citas sin marcar para pintarlas en la UI. `generateMarkdownDraft` lo aplica a todo borrador antes de persistir.
- **Auditoría por cita en la UI** (JUR-01): `extractLegalCitations()` en `@jbhlegal/ui` valida el número de artículo contra el rango real de 5 leyes (ids BOE verificados contra boe.es) y adjunta enlace de verificación de un clic (ancla del artículo en el BOE; buscador de CENDOJ para jurisprudencia, incluidos ECLI). El chip es honesto: «artículo existente» significa que el número existe en el código, no que la cita sea correcta — el letrado confirma.

El motor de jurisprudencia favorable (JUR-02) aplica la misma disciplina en sentido inverso: los precedentes reales de la biblioteca se inyectan como bloque grounded (cita por tribunal/fecha/ECLI reales, con `[VERIFICAR]`, prohibición explícita de inventar *otras* resoluciones) y sus datos van envueltos con `wrapUntrusted`.

#### 9.6.4 Completitud de evidencia

Regla crítica del producto: ningún informe o escrito se genera sin leer y considerar *todos* los documentos, respuestas del cliente y peritajes — un solo documento puede dar la vuelta a un procedimiento penal. La implementación es observable: `computeCaseCoverage()` (`apps/portal/src/lib/case-coverage.ts`, expuesta en `GET /api/cases/[id]/coverage`) calcula la cobertura de evidencia del expediente (documentos analizados vs. fallidos/pendientes, cuestionario respondido, peritajes), y la UI muestra un banner nombrando exactamente qué falta o falló, con la opción de relanzar el análisis. La skill de gobernanza `.claude/skills/evidence-completeness/` obliga a preservar esta invariante en cada cambio.

#### 9.6.5 Auditoría jurídica observable

Todo borrador puede pasar (y los flujos principales pasan) por `auditLegalDocument()` (`generate-legal-audit.ts`): una segunda llamada al flagship con la triple mirada letrado-penalista · fiscal · juez que devuelve un veredicto estructurado (`LegalAuditSchema`) con errores de ley/artículo/jurisprudencia, incoherencias, argumentos débiles e indicaciones de corrección punto por punto (`max_tokens: 16000` — 4000 truncaba el JSON del veredicto). El resultado se persiste en `case_drafts` (`auditJson`, `auditedAt`) junto al estado del último *intento* (`auditStatus: 'ok'|'failed'` + `auditError`): si la auditoría falla, la UI lo muestra con un banner y un botón de reintento (`POST /api/cases/[id]/escritos/[kind]/audit`) en lugar de fingir que el escrito está auditado.

#### 9.6.6 Defensa en profundidad de privacidad y deontología

- **Seudonimización** (`redact-pii.ts`): `buildNameRedactor` / `buildNameRestorer` sustituyen los nombres reales (cliente, víctimas, testigos, coimputados) por etiquetas estables («Investigado 1», «Testigo 2») antes de enviar a Anthropic y los restauran en la salida. Es defensa en profundidad adicional al control primario (ZDR contractual); nació en el flujo de causa multi-investigado (donde el modelo solo ve claves «Investigado N» y el remapeo clave↔caso ocurre en el servidor) y es opcional por llamada vía roster.
- **Deontología publicitaria** (`marketing-banned.ts`): linter determinista, insensible a mayúsculas y acentos, que bloquea promesas de resultado en el copy generado («garantizamos», «resultado garantizado», «absolución asegurada»…) sin prohibir el sustantivo legítimo «garantías procesales»; cualquier coincidencia retiene la pieza para revisión humana en lugar de autopublicarse.

## 9.7 Facturación del consumo de IA

#### 9.7.1 Registro del coste crudo: `ai_request_log`

Cada llamada al modelo registra una fila en `ai_request_log`: scope, caso/entidad, modelo realmente servido (`resp.model`), tokens de entrada facturables (ponderando caché, §9.2.3), tokens de salida y `cost_cents` calculado con `computeCost()` a tarifa de proveedor (USD). Este es el *coste crudo*: alimenta el panel de consumo del despacho, el presupuesto mensual del worker y la conciliación de `admin/ai-usage`.

#### 9.7.2 Margen y tramos

La cara facturable la calcula `billableAiCents(rawCostCents)` en `packages/payments/src/ai-budget.ts`: aplica un multiplicador de margen configurable por `AI_BILLING_MARGIN` (default `DEFAULT_AI_BILLING_MARGIN = 1.35`, es decir +35%; valores fuera de [1, 5] o no numéricos caen al default). El consumo se cobra por **tramos de 100 €**: al superar el presupuesto de IA incluido en el plan (por defecto el 25% del precio del plan), el sistema exige autorizar el exceso (error tipado `AiOverageRequiredError` → HTTP 402 con el diálogo de consentimiento y vinculación de tarjeta) y carga bloques; si el cargo del bloque falla (`AiOveragePaymentFailedError`), la IA queda bloqueada hasta actualizar el método de pago. El modelo comercial: despacho = suscripción + consumo por tramos; cliente directo = 500 €/caso + consumo; primer caso gratis.

> ⚠ **Presentación del margen.** El marketing nunca desglosa «coste + margen»: el consumo se presenta como precio «por uso». El desglose interno (coste crudo vs. facturable) existe solo en `ai_request_log` y los paneles de administración.

### 9.7.3 Presupuestos y bloqueos

| Control | Dónde | Efecto al dispararse |
|---|---|---|
| Presupuesto global mensual de plataforma | `apps/ai-worker/src/budget.ts` (`AI_MONTHLY_BUDGET_CENTS`) | El worker pausa el procesamiento + email de alerta (throttle 1 h). |
| Presupuesto de IA por despacho (% del plan) | `packages/payments` + errores `AiBudgetExceededError` / `AiOverageRequiredError` | HTTP 402 con payload de upsell / diálogo de autorización del exceso. |
| Cargo de bloque fallido | `AiOveragePaymentFailedError` | HTTP 402; IA bloqueada hasta reparar el método de pago. |
| Suscripción vencida / primer caso consumido | `SubscriptionRequiredError` (cap. 10) | HTTP 402; escritura bloqueada (modo lectura). |

### 9.7.4 BYOK: la clave propia del despacho

`packages/ai/src/org-ai.ts` implementa «Bring Your Own Key» de forma mínimamente invasiva: la clave de Anthropic del despacho se guarda cifrada con AES-256-GCM (clave de cifrado en env `ORG_AI_KEY_ENC_KEY`, 32 bytes hex; formato `base64(iv‖tag‖ct)`) en `organizations.anthropic_key_enc`, gestionada desde `PUT /api/me/despacho/ai-key`. Un `AsyncLocalStorage` transporta `{apiKey, byok}` durante todo el flujo: `enterOrgAiContext(orgId)` se invoca en los *gates* de acceso (`requireCaseAccess`, `requireIntakeAccess`…) y en los executors del worker, y desde ahí lo heredan todos los generadores sin tocarlos — `createAnthropicClient()` usa la clave del contexto y `computeCost()` devuelve 0 en contexto BYOK (los tokens los paga la cuenta del despacho, así que el consumo no se factura, aunque los tokens sí se registran). Con BYOK, las garantías de retención y el DPA aplicables son los de la cuenta del despacho, no los del contrato de JBH; la UI lo advierte expresamente antes de activar la clave.

## 9.8 Otros usos de IA en la plataforma

| Uso | Módulos | Modelo | Notas |
|---|---|---|---|
| Triaje del evaluador público | `triage.ts`, `intake.ts`, `evaluate-answers.ts` | `DRAFT_MODEL` (Sonnet) | Lead scoring del embudo gratuito de alto volumen; no requiere profundidad del 100%. |
| Cuestionario calibrado | `generate-questionnaire.ts`, `question-dedup.ts`, `audit-questionnaire.ts` | `pickModel('questionnaire', tier)` | Solo preguntas imprescindibles + protección del testimonio. Deduplicación léxica `QuestionDeduper`: umbrales por defecto Jaccard 0.8 / containment 0.85, apretados a **0.72 / 0.8** en `execute-questionnaire.ts` (el falso positivo — perder una pregunta — es barato frente a repetir preguntas al cliente). |
| Transcripción de grabaciones de reuniones | `apps/ai-worker/src/execute-meeting.ts`, `generate-meeting-transcript.ts` | Whisper CF + Haiku | Audio desde R2 → Cloudflare Workers AI `@cf/openai/whisper-large-v3-turbo` → estructuración/resumen con Haiku. Privilegio opt-in: `include_in_analysis` es `false` por defecto. |
| Marketing worker | `marketing.ts` + `marketing-banned.ts` | `generateJson` (flagship/economy según pieza) | Plan de temas → redacción → autochequeo de seguridad (`MarketingSafetySchema`) → linter determinista; cualquier alerta retiene para revisión humana. |
| RAG de jurisprudencia | `rag/` (embed, Vectorize, chunking, rerank), `index-precedent-rag.ts` | Embeddings CF `@cf/baai/bge-m3` (o Voyage si hay clave) + rerank Haiku | Env-gated (`CLOUDFLARE_ACCOUNT_ID` / `CLOUDFLARE_API_TOKEN` / `VECTORIZE_INDEX`). El aislamiento multi-tenant se reimpone en SQL (`precedentVisibilityFilter`): la metadata del vector nunca autoriza. |
| Chat con el expediente / WhatsApp | `stream-case-chat.ts`, `prompts/whatsapp-client-chat.ts` | Flagship (streaming) | Grounding estricto sobre el dossier («No consta en el expediente»), SSE token a token. |
| OCR / transcripción de PDF escaneado | `ocr.ts`, `transcribe-pdf.ts` | Visión por chunks (`PAGES_PER_CHUNK=3`) | Concurrencia limitada por `PDF_TRANSCRIBE_CONCURRENCY` (§9.5.2). |

> ✓ **Invariantes del motor.** (1) Toda llamada a Anthropic pasa por `createAnthropicClient()` — timeout acotado, BYOK y fallback garantizados. (2) Todo modelo posible está tarifado (exacto o por familia): `computeCost` no puede romper la facturación. (3) Todo texto libre del usuario viaja entre marcadores `DATOS_USUARIO`. (4) Toda cita legal sale verificada por JUR-01 o marcada `[VERIFICAR]`. (5) Ningún borrador se da por auditado sin un veredicto persistido con `auditStatus='ok'`.

# 10
# Superficie de API

> *La plataforma expone su lógica de negocio a través de rutas API de Next.js (App Router, un `route.ts` por endpoint): 268 rutas en el portal del abogado (`apps/portal/src/app/api`), 67 en la app del cliente (`apps/cliente/src/app/api`), más la API pública de captación de `apps/web` y el microservicio Hono de `apps/api`. Este capítulo documenta las convenciones transversales (manejo de errores tipados, validación Zod, rate limiting, anti-enumeración, gates de acceso multi-tenant) y el inventario por grupos funcionales con sus rutas más relevantes.*

## 10.1 Panorama y convenciones transversales

### 10.1.1 `withErrorHandler` y el mapeo de errores tipados

Prácticamente toda ruta del portal y del cliente envuelve su handler con `withErrorHandler(fn)` (`apps/portal/src/lib/errors.ts` y su espejo en cliente): propaga un `request_id` (leído de cabeceras vía `@jbhlegal/logger`, etiquetado en Sentry), captura cualquier excepción con `Sentry.captureException` y la traduce con `mapError(err)` a una respuesta HTTP estable. Las rutas *lanzan* errores tipados en lugar de construir respuestas de error a mano; el mapeo es la única fuente de verdad del contrato de errores:

| Error tipado | HTTP | `error` devuelto / payload |
|---|---|---|
| `UnauthorizedError` (sin sesión) | 401 | `unauthorized` |
| `ForbiddenError` / `MfaRequiredError` / `ReadOnlyImpersonationError` / `ColegiadoVerificationRequiredError` | 403 | `forbidden` · `mfa_required` (la UI redirige al reto 2FA) · `read_only_impersonation` (el superadmin en «ver como» nunca escribe) · `colegiado_verification_required` |
| `NotFoundError` | 404 | `not_found` |
| `ValidationError` | 400 | string humano tal cual; un `ZodError` se traduce a español legible + mapa por campo (`zodErrorToSpanish`) |
| `SubscriptionRequiredError` (`first_case_used` · `read_only` · `expedientes_limit`) | 402 | `subscription_required` — la UI muestra el upsell; nunca aplica al tenant JBH |
| `AiBudgetExceededError` / `AiOverageRequiredError` / `AiOveragePaymentFailedError` | 402 | payload con `budgetEur` / `spentEur` /plan → upsell, diálogo de autorización del exceso o reparación del método de pago (cap. 9 §9.7) |
| `RateLimitError` | 429 | `rate_limited` + `retryAfter` (segundos) |
| `RoleConflictError`, `ExpedienteNumDuplicateError`, `IntakeAlreadyConvertedError`, `PaymentInvalidStateError`, `SignatureRequestAlreadyPendingError`, `Collaborator*` (duplicate/invalid-state/not-active), `DsrInvalidStateError` | 409 | código estable homónimo (conflicto de estado) |
| `SignatureRequestInvalidStateError`, `SignatureHashMismatchError`, `CollaboratorInvitationInvalidError` (`expired\|revoked\|invalid_token`), `DsrDownloadExpiredError` | 410 | recurso caducado/retirado |
| `FileTooLargeError` · `UnsupportedMimeTypeError` · `MimeMismatchError` / `MultipartParseError` | 413 · 415 · 400 | subidas: tope de bytes (`maxBytes`), MIME no admitido, contenido≠extensión, multipart ilegible |
| `PaymentSessionCreateError` | 502 | `payment_session_create_failed` + `reason` (fallo aguas arriba en Stripe) |
| `AiServiceError` (de `@jbhlegal/ai`) | según `err.status` | mensaje en español + `code` estable accionable (p. ej. `no_credit` / `byok_no_credit` → CTA a «Mi despacho») |

### 10.1.2 Validación, rate limiting y anti-enumeración

- **Zod en el borde**: los bodies y params se validan con schemas Zod; el fallo lanza `ValidationError(zodError)` y llega al cliente como 400 con mensaje en español y errores por campo. TypeScript estricto (`noUncheckedIndexedAccess`) obliga además a validar todo acceso indexado.
- **Rate limiting**: `consume({ key, max, windowMs })` (`apps/portal/src/lib/rate-limit.ts`, con copias idénticas en cliente y peritos) implementa buckets en memoria por clave (típicamente `ruta:usuario` o `ruta:IP`) y lanza `RateLimitError(retryAfter)` al superar el cupo. Se aplica en los endpoints sensibles: registro, invitaciones, subidas (la subida masiva de jurisprudencia usa 500/h), triaje público, generación de IA.
- **Respuestas neutras anti-enumeración**: los endpoints que tocan identidades (`registro-despacho`, `registro-profesional`, `aceptar-invitacion`, login OTP) responden siempre lo mismo exista o no la cuenta («revisa tu correo»), sin revelar si un email pertenece a un usuario ni su rol. La misma disciplina rige el orden 404-antes-que-403 de los gates (§10.1.3): no filtrar la *existencia* de recursos ajenos.

### 10.1.3 Gates de acceso multi-tenant

Toda ruta anidada bajo `cases/[id]`, `intakes/[id]` o `causas/[id]` pasa por su gate: `requireCaseAccess(caseId, allowedRoles)` (`apps/portal/src/lib/case-access.ts`), `requireIntakeAccess` o el gate de causa. El gate: (1) exige sesión y rol de portal permitido (un CLIENT/PERITO recibe 403 antes de tocar la BD); (2) resuelve el tenant activo (`getActiveContext`) y filtra por `org_id` — un expediente de otro despacho devuelve 404 aunque

se conozca su id, y un expediente con soft-delete (`deleted_at`) se trata como inexistente; (3) para el rol `COLABORADOR`/`LAWYER` no titular, comprueba la co-letraduría en `case_collaborators`; (4) invoca `enterOrgAiContext(orgId)`, de modo que todo el flujo de IA de la petición hereda la clave BYOK del despacho (cap. 9 §9.7.4); y (5) devuelve `{ session, case, ctx }` con copia defensiva de la fila.

> ⚠ **Rol sin gate = IDOR.** El bug sistémico recurrente del Modelo B: añadir el rol `LAWYER` a una ruta sin que ésta pase por `requireCaseAccess`/`requireIntakeAccess` permite acceder a recursos de otros tenants por id. Toda ruta nueva bajo `[id]` debe empezar por su gate; existe un lint de gates y tests de aislamiento cross-org que lo verifican.

### 10.1.4 Convenciones de datos y transporte

- **Subida de ficheros en tres fases**. Todos los grupos documentales (portal, cliente, onboarding por token, intake público de la web) comparten el mismo patrón hacia R2: `upload-init` (valida nombre/MIME/tamaño declarados, crea el registro y devuelve URLs firmadas por parte), subida directa del navegador a R2, y `upload-complete` (verifica lo recibido, fija el estado y encola el análisis IA); `upload-abort` limpia los restos. Los errores de subida son accionables y tipados: `file_too_large` (413, con `maxBytes`), `unsupported_mime_type` (415), `mime_mismatch` (400, el contenido real no coincide con la extensión — sniffing en servidor) y `multipart_parse_failed` (400).
- **Streaming SSE**. Las respuestas largas de IA con latencia visible (chat con el expediente en `cases/[id]/chat/threads/[tid]/messages`) se sirven como Server-Sent Events token a token — además de mejorar la UX, esquiva el `ProxyTimeout` del Apache que hay delante de los procesos Next en Plesk.
- **Escrituras multi-paso en transacción**. Toda mutación que toca varias tablas (promoción de intake a caso, generación de escrito + snapshot de versión, liquidaciones) va dentro de `db.transaction`; los contadores correlativos se calculan con `MAX+1` dentro de la transacción (nunca `COUNT+1`, que colisiona tras un borrado) — dos de los gotchas sistémicos documentados del monorepo.
- **Descargas mediadas**. Ningún endpoint devuelve URLs permanentes de R2: las descargas (`.../download`, `jurisprudence/[id]/file`, `me/dsr/[id]/download`) generan URLs firmadas de corta vida tras pasar el gate, y las de DSR caducan con error tipado `dsr_download_expired` (410).
- **Idempotencia**. Los webhooks deduplican por id de evento (§10.2.7); las altas por invitación y los registros públicos son idempotentes por diseño (repetir el POST no crea duplicados ni revela estado).

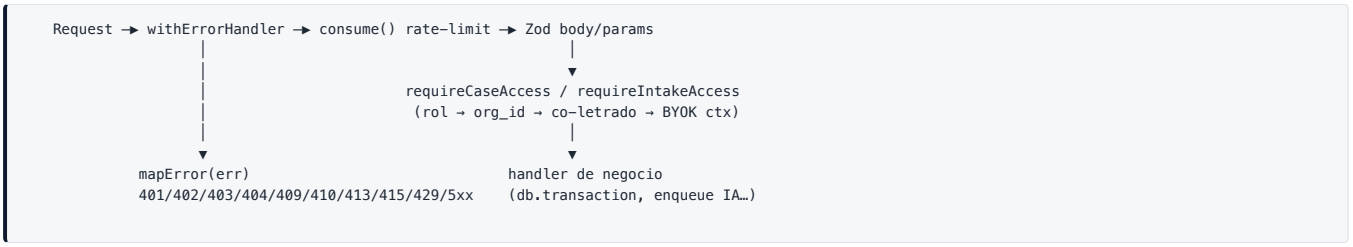


Fig. 10.1 — Canalización estándar de una petición API del portal.

## 10.2 Portal del abogado — `apps/portal/src/app/api` (268 rutas)

Prefijo público `/api/...` bajo el dominio del portal. Salvo indicación, el gate es «sesión de portal + rol permitido» y, bajo `cases/[id]`, siempre `requireCaseAccess`. La distribución por grupos del inventario completo (censado con `find apps/portal/src/app/api -name route.ts`):

| Grupo | Rutas | Gate dominante |
|---|---|---|
| `cases/**` | ~95 | `requireCaseAccess` (+ `enterOrgAiContext`) |
| `me/**` | ~45 | sesión del propio usuario / rol de gestión del despacho |
| `admin/**` (incl. `superadmin/**`) | ~37 | rol admin de plataforma / superadmin (impersonación solo-lectura) |
| `crm/**` + `intakes/**` + `bookings` | ~33 | `requireIntakeAccess` / rol de gestión |
| `jurisprudence/**` | 11 | sesión + `precedentVisibilityFilter` (aislamiento en SQL) |
| `colaboradores/**` + `referral/**` + `marketplace/**` | ~20 | rol de gestión / panel del referidor |
| `causas/**` | 7 | gate de causa (mismo patrón que casos) |
| público + webhooks + auth + onboarding + salud + registro | ~20 | token opaco · firma criptográfica · respuesta neutra |

Se listan a continuación las rutas representativas de cada grupo (métodos entre corchetes).

### 10.2.1 Expedientes: `cases/**` (~95 rutas)

| Ruta | Métodos | Propósito |
|---|---|---|
| `cases` · `cases/[id]` | GET · GET,DELETE | Listado del tenant; detalle y baja (soft-delete) del expediente. |
| `cases/[id]/status` · `workflow` | POST · GET | Transición de estado (con evento de auditoría «from → to») y máquina de estados del caso. |

| Ruta | Métodos | Propósito |
|---|---|---|
| `cases/[id]/posicion-cliente` · `penal-category` · `situacion-procesal` · `detencion` · `hechos` · `base-context` (+ `/suggest`) | GET,PUT (suggest: POST) | Datos rectores del caso: rol procesal defensa/acusación, familia penal, fase y situación de libertad, datos de la detención, relato de hechos y contexto base (con sugerencia IA). |
| `cases/[id]/client-invitation` · `client-visibility` | POST · GET,PATCH | Invitar al cliente al portal (Modelo B) y controlar qué ve. |
| `cases/[id]/co-letrados` (+ `/[userId]`) · `lead-lawyer` · `lawyer-invite` · `pending-lawyer` | GET,POST / DELETE · POST · GET,POST,DELETE · GET | Co-letraduría (`case_collaborators`), cambio de letrado titular e invitación de abogado externo. |
| `cases/[id]/documents` · `documents/upload-init\|upload-complete\|upload-abort` | GET,POST · POST | Inventario documental y subida multiparte directa a R2 (init → parts → complete, con abort). |
| `cases/[id]/documents/[did]` · `[did]/download` · `[did]/status` · `[did]/visibility` | DELETE · GET · GET · POST | Borrado, descarga firmada, estado de análisis y visibilidad para el cliente de cada documento. |
| `cases/[id]/documents/[did]/analysis` (+ `/re-run`) · `[did]/analyze-text` | GET / POST · POST | Análisis IA por documento: lectura, re-encolado y análisis de texto pegado (sin fichero). |
| `cases/[id]/analysis` (+ `/re-run`) · `coverage` · `prognosis-history` · `clarifications` (+ `/[cid]`) | GET / POST · GET · GET · GET,POST / DELETE | Informe de análisis del caso, cobertura de evidencia (completitud, cap. 9 §9.6.4), evolución del pronóstico por versión y aclaraciones pendientes con coste estimado del re-análisis. |
| `cases/[id]/escritos` · `escritos/[kind]` · `escritos/[kind]/audit` | GET · GET,POST,PUT,PATCH · POST | El arsenal del Letrado IA: estado de todos los borradores; generar (POST, con registro `GENERATORS` ~40 kinds, invariante posición↔kind), editar (PUT/PATCH con snapshot de versión) y auditar (triple mirada penalista·fiscal·juez, persistiendo `audit_status`). |
| `cases/[id]/claves/estado` · `claves/acciones` (+ `/crear`) | GET · POST | «Claves del caso»: estado del módulo y conversión de acciones recomendadas en tareas. |
| `cases/[id]/questionnaire` · `questionnaire/questions` (+ `/[qid]`) · `questionnaire/send` · `questionnaire/audit` · `questionnaire/evaluation` | GET,POST,PATCH · POST / PATCH,DELETE · POST · POST · GET | Cuestionario aclaratorio: generación IA, edición de preguntas, envío al cliente, auditoría pre-envío (PRU-01) y evaluación IA de las respuestas. |
| `cases/[id]/chat/threads` (+ `/[tid]`, `/[tid]/messages`) | GET,POST / DELETE,PATCH / GET,POST | Chat con el expediente (PRO-01): hilos y mensajes; la respuesta IA llega por streaming SSE. |
| `cases/[id]/messages` (+ `/[mid]`, `/read`) | GET,POST / PATCH,DELETE / POST | Mensajería abogado↔cliente con ventana de edición (`edit_window_expired`) y marcado de leídos. |
| `cases/[id]/tasks` (+ `/[tid]`, `/[tid]/complete`) · `milestones` (+ `/[mid]`) · `court-events` (+ `/[eventId]`) · `novedades` (+ `/seen`) | POST / PATCH,DELETE / POST · POST / PATCH,DELETE · GET,POST / PATCH,DELETE · GET / POST | Plazos y tareas (EST-01), hitos, señalamientos judiciales y feed de novedades del caso. |
| `cases/[id]/meetings` (+ `/[mid]`, `/[mid]/audio`, `/[mid]/complete`) · `meeting` · `meeting/[mid]/video-token` · `meeting/cancel` | GET,POST / GET,PATCH,DELETE / GET / POST · PUT · POST · POST | Reuniones y grabaciones (subida de audio a R2 → Whisper CF → transcripción IA) y videollamada (token de sala + cancelación). |
| `cases/[id]/payments/[paymentId]/resend-link` · `hoja-encargo` · `ai-consumption` | POST · POST · GET | Reenviar enlace de pago Stripe, generar la hoja de encargo y consultar el consumo de IA del caso. |
| `cases/[id]/signature-requests` (+ `/[reqId]`, `/[reqId]/resend`) | GET,POST / DELETE / POST | Solicitudes de firma electrónica de documentos del caso (integración con la app de firmas). |
| `cases/[id]/peritos` (+ `/[peritoUserId]/messages`, `/doc-grants`, `/presupuestos/[pid]`, `/workflow`) | GET,POST,DELETE / GET,POST / POST,DELETE / POST / GET | Peritos del caso: asignación, mensajería, concesión de acceso a documentos concretos, aceptación de presupuestos y flujo del peritaje. |
| `cases/[id]/witnesses` (+ `/[wid]`, `/suggest`) · `investigados` · `trial-prep` (+ `/simulator`) · `resumen-ejecutivo` · `valoraciones` (+ `/[vid]`) | GET,POST / PATCH,DELETE / POST · GET,POST · GET,POST,PUT,PATCH / POST · GET,POST · GET,POST / PATCH,DELETE | Testigos (con sugerencia IA), co-investigados, preparación de vista/juicio (+ simulador de interrogatorio), resumen ejecutivo de defensa (ESC-09) y valoraciones. |
| `cases/[id]/jurisprudence` · `jurisprudence/suggestions` (+ `/[sid]`) | GET · GET / PATCH | Precedentes vinculados al caso y sugerencias del motor de jurisprudencia favorable (aceptar/descartar/vincular). |

#### 10.2.2 Causas multi-investigado: `causas/**`

| Ruta | Métodos | Propósito |
|---|---|---|
| `causas/[id]/analisis` · `coordination` | GET · GET | Análisis de la causa (seudonimizado «Investigado N» hacia el modelo) y mapa de coordinación de defensas re-identificado en servidor (PRO-03). |
| `causas/[id]/comparativa` (+ `/contradicciones`) · `core-questions` (+ `/[qid]`) | GET / GET,POST · GET,POST / PATCH | Matriz comparativa de respuestas núcleo entre investigados, detección IA de contradicciones (PRU-07) y gestión de las preguntas núcleo. |
| `causas/[id]/escritos/[kind]` | GET,POST,PATCH | Escritos coordinados de la causa (ESC-08: estrategia coordinada, acuerdo de defensa conjunta, conflicto de interés) con su propio registro `GENERATORS`. |

#### 10.2.3 Captación y CRM: `intakes/**`, `crm/**`, `bookings`

| Ruta | Métodos | Propósito |
|---|---|---|
| `intakes` · `intakes/stats` · `intakes/[id]` | GET · GET · GET,DELETE | Bandeja de consultas entrantes (leads) del tenant, métricas y detalle (gate `requireIntakeAccess`). |
| `intakes/[id]/assign` · `status` · `notes` · `email` · `reia` | POST | Asignar responsable, mover de estado, anotar, contactar por email y re-lanzar la evaluación IA del intake. |
| `intakes/[id]/promote` | POST | Convertir el intake cualificado en expediente (lanza `intake_not_qualified` / `intake_already_converted` si no procede). |
| `intakes/[id]/documents` (+ `/[did]`, `/[did]/download`) | GET / DELETE / GET | Documentación aportada en fase de consulta. |
| `crm/board` (+ `/column`) · `crm/stats` · `crm/report` | GET | Pipeline kanban sobre `intakes` (enfoque intake=lead), carga por columna, métricas y embudo/informes. |
| `crm/intakes/[id]` (+ `/message`) · `crm/tasks` (+ `/[id]`) · `crm/quick-intake` · `crm/import` · `crm/unqualified` (+ `/[id]/promote`) | GET,PATCH / POST · GET,POST / PATCH · POST · POST · GET / POST | Ficha 360º del lead, borradores IA de email/WhatsApp, tareas comerciales (incl. urgencias procesales detectadas por IA), alta rápida, importación CSV y recuperación de no cualificados. |
| `crm/presupuestos` (+ `/[id]`, `/[id]/send`) · `crm/facturas` (+ `/[id]`, `/[id]/pdf`) · `crm/invoicing-settings` | GET,POST / GET,PATCH / POST · GET,POST / GET,PATCH / POST · GET,PUT | Presupuestos con aceptación pública por token (§10.2.7), facturación del despacho y su configuración. |
| `bookings/[id]/convert-to-intake` · `admin/bookings` (+ `/[id]/reassign`) | POST · GET / POST | Citas reservadas desde la web pública: conversión a intake y reasignación por staff. |

#### 10.2.4 Cuenta y despacho: `me/**`

| Ruta | Métodos | Propósito |
|---|---|---|
| `me/profile` · `me/preferences` · `me/public-profile` · `me/branding` | GET,PATCH · GET,PATCH · GET,PUT · GET,POST,PUT | Perfil del usuario, preferencias, ficha pública del abogado y marca blanca del despacho (logo servido por `branding/[orgId]/logo`). |
| `me/2fa` (+ `/verify`) | GET,POST,PUT,DELETE / POST | Alta, verificación y baja del segundo factor TOTP (política MFA flag-gated). |
| `me/colegiado/challenge` · `verify` · `assisted` | POST · POST · GET,POST | Verificación de colegiado ACA: reto de firma con certificado cualificado, verificación y vía asistida revisada por superadmin. |
| `me/despacho/ai-key` | GET,PUT,DELETE | BYOK: alta/consulta/baja de la clave Anthropic propia del despacho (cifrada AES-256-GCM en `organizations.anthropic_key_enc`). |
| `me/org-subscription/checkout` · `change-plan` · `portal` · `redeem-coupon` · `request-custom` (+ `checkout-custom`) · `ai-overage` | POST (ai-overage: POST,DELETE) | Suscripción del despacho (Modelo B): checkout Stripe, cambio de plan, portal de facturación, cupones, planes a medida y autorización/revocación del exceso de IA. |
| `me/subscription` (+ `/checkout`, `/portal`) · `me/billing-profile` · `me/invoices` · `me/ai-consumption` | GET / POST / POST · GET,PUT · GET · GET | Suscripción individual, datos de facturación, facturas y panel de consumo de IA. |
| `me/team` (+ `/[membershipId]`, `/invitations/[id]`) | GET,POST / PATCH,DELETE / DELETE | Equipo del despacho: memberships, roles e invitaciones (aceptación en `aceptar-invitacion`, respuesta neutra). |
| `me/calendar/availability` · `me/calendar/google/start\|callback\|disconnect` · `me/agenda` (+ `/google`) · `me/booking-settings` · `agenda/feed/[token]` | GET,PUT · GET/—/POST · GET · GET,PUT · GET | Disponibilidad de citas, OAuth con Google Calendar, agenda unificada y feed iCal por token. |
| `me/dsr` (+ `/export`, `/delete`, `/[id]/download`) · `me/notifications/**` · `me/novedades` · `me/documents/upload` | GET / POST / POST / GET · GET,POST · GET · POST | Derechos RGPD del propio usuario (exportación/supresión con estados y caducidad de descarga), notificaciones y subidas personales. |

#### 10.2.5 Administración y superadmin: `admin/**`

Gate de rol administrador de plataforma; el subárbol `admin/superadmin/**` exige el rol superadmin (y la impersonación «ver como» es de solo lectura: cualquier escritura lanza `read_only_impersonation`).

| Ruta | Métodos | Propósito |
|---|---|---|
| `admin/ai-usage` (+ `/by-client`, `/recent`, `/reconciliation`) | GET | Consumo de IA de la plataforma: agregado, por cliente, últimas llamadas y conciliación contra `ai_request_log`. |

| Ruta | Métodos | Propósito |
|---|---|---|
| `admin/users` (+ `/[id]`, `/[id]/roles`) · `users` · `clientes` · `peritos` | GET,POST / PATCH / GET,POST,DELETE · GET · POST · GET | Gestión de usuarios y roles; directorio de usuarios, alta de clientes y directorio de peritos. |
| `admin/dsr` (+ `/[id]`, `/[id]/approve`, `/[id]/reject`) | GET / GET / POST / POST | Cola de solicitudes DSR (RGPD) con aprobación/denegación por staff. |
| `admin/leads` · `admin/analytics` · `admin/invoices` · `admin/internal-coupons` | GET,PATCH · GET · GET · GET,POST,DELETE | Leads de la web, analítica, facturación y cupones internos. |
| `admin/marketing` (+ `/[id]`, `/run`, `/seo`, `/settings`, `/stats`) | GET / PATCH,DELETE / POST / GET / PUT / GET | Cola editorial del marketing autónomo (piezas retenidas por el linter deontológico), ejecución manual, monitor SEO y ajustes. |
| `admin/support/tickets/**` · `admin/suggestions` (+ `/[id]`) · `admin/jurisprudence/seeding` | GET,PATCH,POST · GET / PATCH · GET | Soporte interno, buzón de sugerencias y estado de la siembra de la biblioteca de jurisprudencia. |
| `admin/superadmin/verificaciones` (+ `/[id]/doc`) · `comisiones` · `referral` (+ `/settlements`) · `linkbuilding` (+ `/[id]`, `/[id]/verify`) | GET,PATCH / GET · GET,PATCH · GET,POST,PATCH · GET,POST / PATCH,DELETE / POST | Verificación asistida de colegiados, comisiones, programa de referidos (liquidaciones) y tracker de enlaces externos con verificación real. |

#### 10.2.6 Biblioteca de jurisprudencia, colaboradores y referidos

| Ruta | Métodos | Propósito |
|---|---|---|
| `jurisprudence` · `jurisprudence/[id]` (+ `/file`, `/favorite`, `/link`, `/validate`, `/report`, `/dedup`) · `duplicates` · `taxonomy` | GET,POST · GET / GET / POST / POST,DELETE / POST / POST / PATCH · GET · GET | Biblioteca penal compartida: subida (incl. masiva con rate-limit 500/h), descarga (o enlace a fuente oficial BOE/TC), favoritos, vinculación a casos (`allow_ai_use`), validación, denuncia de contenido y gestión de duplicados. |
| `jurisprudence/search` | POST | Búsqueda IA (RAG): embedding de la consulta → Vectorize → **re-imposición del aislamiento en SQL** (`precedentVisibilityFilter`) → rerank; degrada a palabra clave si el RAG no está configurado. |
| `colaboradores` (+ `/[id]`, `/[id]/approve\|reject\|suspend\|reactivate\|revoke\|resend-invitation`, `/invitations`) | GET / GET / POST ×6 / POST | Programa de colaboradores (E1): ciclo de vida completo de la relación con estados tipados (`collaborator_invalid_state`, 409). |
| `referral/profile` · `alias` · `assets` · `referrals` · `commissions` · `settlements` · `tax-profile` | GET,POST · GET,POST · GET · GET · GET · GET,POST · GET,PUT | Panel del referidor: alta, alias del enlace `/r/{alias}`, materiales, referidos atribuidos, comisiones, liquidaciones y datos fiscales. |
| `marketplace/settings` · `marketplace/proposals/[id]/accept\|decline` | GET,PUT · POST | Marketplace de derivación de casos entre abogados: ajustes y respuesta a propuestas. |

#### 10.2.7 Público, webhooks, auth y salud (portal)

| Ruta | Métodos | Gate | Propósito |
|---|---|---|---|
| `auth/[...nextauth]` | (NextAuth) | — | Autenticación NextAuth (login OTP por email, sesiones JWT). |
| `registro-despacho` · `registro-profesional` · `aceptar-invitacion` | POST | rate-limit + respuesta neutra | Alta de despacho (Modelo B), alta profesional e ingreso por invitación — idempotentes y anti-enumeración: nunca revelan si el email existe. |
| `public/presupuestos/[token]/decide` · `public/facturas/[token]/pdf\|declare-paid` · `public/lawyer-join` | POST · GET/POST · POST | token opaco de un solo recurso + rate-limit | Aceptación pública de presupuestos del CRM (aceptado → pendiente_firma_pago), factura por token y adhesión de abogado invitado. |
| `stripe/webhook` | POST | firma Stripe | Webhook Stripe: verifica `stripe-signature` contra `STRIPE_WEBHOOK_SECRET` (`constructEvent` vía `verifyAndParseWebhookWithId`), deduplica por `event_id` prefijado (`portal:`) en `stripe_webhook_events` antes de ejecutar y *revierte el registro de dedup si el handler falla* para que el reintento de Stripe re-ejecute de verdad. |
| `demo-login` · `health` (+ `/deep`) · `%5Ftest-error` | POST · GET · GET | flag demo / — | Acceso a las cuentas demo (flag `is_demo`), health check superficial y profundo (BD), y ruta de prueba de Sentry. |
| `onboarding/[token]` (+ `/profile`, `/upload-doc`, `/request-signature`, `/submit-for-review`) | GET / POST ×4 | token de invitación | Onboarding del colaborador: datos, documentación, firma del contrato y envío a revisión (`collaborator_onboarding_incomplete` con la lista de lo que falta). |
| `suggestions` · `support/tickets` (+ `/[id]`, `/[id]/messages`) | POST · GET,POST / GET / POST | sesión | Buzón de sugerencias y soporte del usuario del portal. |

## 10.3 App del cliente — `apps/cliente/src/app/api` (67 rutas)

El cliente (rol CLIENT) solo accede a *sus* recursos: el gate de `my/cases/[id]` comprueba la titularidad del caso además del tenant, y la visibilidad de documentos/análisis está filtrada por lo que el abogado ha compartido (`client-visibility`). El grupo `onboarding/[token]` replica el flujo documental y el cuestionario para clientes invitados que aún no han creado cuenta, autenticados solo por el token de invitación.

| Ruta | Métodos | Propósito |
|---|---|---|
| `my/cases` · `my/cases/[id]` | GET · GET | Expedientes propios y detalle con la vista permitida al cliente. |
| `my/cases/[id]/documents` (+ `/upload-init\|upload-complete\|upload-abort`, `/[did]`, `/[did]/download`, `/[did]/status`) | GET,POST / POST ×3 / DELETE / GET / GET | Aportación documental del cliente con el mismo flujo multiparte a R2 que el portal. |
| `my/cases/[id]/questionnaire` (+ `/answers`, `/submit`) | GET / PUT / POST | Cuestionario aclaratorio: lectura, guardado incremental de respuestas y envío definitivo. |
| `my/cases/[id]/analysis/re-run` · `base-context` · `clarifications` · `co-investigados` | POST · GET · GET,POST · GET | Relanzar el análisis con aclaraciones nuevas, contexto base y co-investigados visibles. |
| `my/cases/[id]/messages` (+ `/[mid]`, `/read`) · `perito-messages` | GET,POST / PATCH,DELETE / POST · GET,POST | Mensajería con el abogado y con el perito del caso. |
| `my/cases/[id]/payments/retry` · `peritaje/pay` · `fianza` | POST | Reintentar un cargo fallido, pagar el peritaje y gestionar la fianza. |
| `my/cases/[id]/meeting/[mid]/video-token` · `tasks/[tid]/complete` · `trial-prep` (+ `/simulator`) · `witnesses` · `invite-lawyer` | POST · POST · GET / POST · GET,POST · GET,POST,DELETE | Unirse a la videollamada, completar tareas asignadas, preparación de la declaración (+ simulador), proponer testigos e invitar a su propio abogado externo. |
| `my/bookings` (+ `/slots`, `/[id]/cancel`) | GET,POST / GET / POST | Citas del cliente: reservar sobre la disponibilidad publicada, listar y cancelar. |
| `my/marketplace` (+ `/offer`, `/withdraw`) · `my/peritos` (+ `/suggest`) | GET / POST / POST · GET / POST | Publicar el caso en el marketplace de abogados y sugerir peritos. |
| `me/dsr` (+ `/export`, `/delete`, `/[id]/download`) | GET / POST / POST / GET | Derechos RGPD del cliente (arts. 15/17/20): exportación y supresión con ventana anti-abuso (`dsr_recent_request`) y descarga caducable. |
| `me/payment-method` · `me/invoices` · `me/preferences` · `me/notifications/**` · `me/novedades` · `me/referral/**` · `me/suggestions` · `me/support/tickets/**` | POST · GET · GET,PATCH · GET,POST · GET · GET,POST · POST · GET,POST | Método de pago, facturas, preferencias, notificaciones, novedades, programa de referidos del cliente, sugerencias y soporte. |
| `onboarding/[token]/cases/[id]/documents` (+ `/[did]`, `/[did]/download`) · `questionnaire` (+ `/answers`, `/submit`) | GET,POST / DELETE / GET · GET / PUT / POST | Flujo de invitado por token: aportar documentación y responder el cuestionario antes de tener cuenta. |
| `demo-login` · `auth/[...nextauth]` · `health` (+ `/deep`) | POST · (NextAuth) · GET | Demo, autenticación OTP y salud. |

## 10.4 API pública de captación — `apps/web/src/app/api` y `apps/api` (Hono)

### 10.4.1 `apps/web`: endpoints públicos de la web de marketing

Sin sesión: la protección es rate-limit por IP, tokens opacos por recurso y verificación criptográfica en los webhooks. Todos los textos libres que acaban en IA viajan como datos no confiables (cap. 9 §9.6.1).

| Ruta | Protección | Propósito |
|---|---|---|
| `public/triage` · `public/evaluar` (vía Hono) | rate-limit IP | Evaluador penal público: triaje IA preliminar del relato (lead scoring con `DRAFT_MODEL`). |
| `public/intakes` (+ `/[id]/confirm`) · `public/contact` · `public/leads` (+ `/unsubscribe`) | rate-limit + honeypot | Alta de consultas, contacto, captura de leads del lead-magnet y baja de comunicaciones. |
| `public/intake-uploads/init` (+ `/[id]/complete\|abort\|status`) | token de subida | Aportación documental anónima previa al alta, con el flujo multiparte a R2. |
| `public/booking/init` · `slots` · `[id]/cancel\|reschedule` · `booking/stripe-webhook` | token de reserva / firma Stripe | Reserva de citas (incl. de pago); el webhook Stripe de booking deduplica con su propio prefijo de `event_id` para no colisionar con el del portal. |
| `public/telnyx-ai/init` · `tool` · `post-call` | firma Ed25519 o secreto compartido | Recepcionista telefónica IA (Telnyx AI Assistant): variables dinámicas al inicio de llamada, ejecución de tools (`voice-assistant-tools.ts`) y resumen post-llamada. Autorización preferente por firma Ed25519 de Telnyx sobre `timestamp\|rawBody` (`apps/web/src/lib/telnyx-verify.ts`, clave pública en `TELNYX_PUBLIC_KEY`, anti-replay 10 min) con fallback de secreto compartido por cabecera o query (`?key=`, porque el subsistema de variables dinámicas no envía cabeceras). Gotcha operativo: Telnyx envía los parámetros no rellenados como `null` y anidados/serializados — el handler los limpia y desenvuelve (`unwrap`) antes de validar con Zod. |
| `public/search` · `public/jurisprudencia` | rate-limit | Buscador del sitio y consulta pública de la enciclopedia/jurisprudencia. |

| Ruta | Protección | Propósito |
|---|---|---|
| `internal/lead-followup` · `internal/leads/nurture` · `internal/voice-reminders` | secreto interno | Tareas programadas: seguimiento de leads, nurturing y recordatorios de voz. |

#### 10.4.2 `apps/api`: microservicio Hono

`apps/api/src/server.ts` monta una app Hono ligera (con middleware `cors` y `logger`, servida por `@hono/node-server`) con cinco grupos de rutas montados vía `app.route()`:

| Montaje | Módulo | Propósito |
|---|---|---|
| `/api/health` | `apps/api/src/routes/health.ts` | Salud del microservicio. |
| `/api/evaluar` | `routes/evaluar.ts` | Backend del evaluador penal público (triaje IA del relato). |
| `/api/whatsapp` · `/api/whatsapp-telnyx` | `routes/whatsapp.ts`, `routes/whatsapp-telnyx.ts` | Webhooks del canal WhatsApp; la variante Telnyx verifica la firma Ed25519 sobre el cuerpo crudo y es la implementación de referencia que `apps/web/src/lib/telnyx-verify.ts` replica (las apps Next no pueden importar de `apps/api`). |
| `/api/internal/lead-followup` | `routes/lead-followup.ts` | Disparador interno del seguimiento de leads. |

El servicio comparte `packages/ai`, `packages/db` y `packages/email` con el resto del monorepo, por lo que aplica las mismas convenciones de seguridad de prompts y tarificación del capítulo 9; incluye además el `email-worker.ts` de envíos diferidos.

> ℹ **Rutas legadas y de prueba.** `%5Ftest-error` (portal y cliente) existe solo para verificar la captura de Sentry en producción. Los duplicados de sincronización de iCloud (`route 2.ts`, `route 3.ts`) no son rutas: Next solo compila `route.ts`.

## 10.5 Checklist para añadir una ruta nueva

Condensado de las convenciones anteriores, el orden canónico que sigue toda ruta nueva del portal o del cliente (y que revisan el lint de gates y los tests de aislamiento cross-org):

1. **Envolver** el handler con `withErrorHandler` y exportar solo los métodos HTTP reales (`export const GET = withErrorHandler(async (req, ctx) => …)`).
2. **Gate primero**: si la ruta cuelga de `cases/[id]`, `intakes/[id]` o `causas/[id]`, llamar a `requireCaseAccess` / `requireIntakeAccess` antes de cualquier consulta, con la lista explícita de roles (incluir `LAWYER` para el Modelo B *solo* a través del gate). El gate ya entra en el contexto BYOK.
3. **Rate-limit** con `consume()` si el endpoint es público, de registro, de subida o dispara IA.
4. **Validar** body y params con Zod y lanzar `ValidationError`; los importes con `.nonnegative()` (no `.positive()`, que rompe los ceros legítimos).
5. **Errores tipados**, nunca `NextResponse.json({error},…)` a mano para condiciones de negocio: si el caso no existe para ese tenant es `NotFoundError` (no 403), y las respuestas que tocan identidades deben ser neutras.
6. **Transacción** para escrituras multi-tabla y correlativas con `MAX+1`; en migraciones nuevas, respetar la colación `utf8_general_ci` de `cases.id` para las FK (errno 150).
7. **Coste de IA**: si la ruta genera con el modelo, elegirlo con `pickModel` / `bestLegalReasoningModel`, registrar la llamada en `ai_request_log` con el modelo de `resp.model` y persistir el resultado versionado.
8. **Verificar en vivo** tras desplegar (doble verificación): la ruta responde en el dominio correcto y con el contrato esperado.

> ✓ **Invariantes de la superficie.** (1) Ningún handler construye errores a mano: lanza tipado y `mapError` responde. (2) Ninguna ruta bajo `[id]` toca la BD sin su gate (`requireCaseAccess` / `requireIntakeAccess`) — el filtro `org_id` + 404-antes-que-403 impide IDOR y enumeración. (3) Todo webhook verifica firma criptográfica (Stripe `constructEvent`, Telnyx Ed25519) sobre el cuerpo crudo y deduplica por id de evento. (4) Toda mutación multi-paso va en `db.transaction`. (5) Los endpoints públicos por token nunca exponen ids internos ni existencia de cuentas.

# 11
# El portal del abogado

> *El portal (`apps/portal`, dominio del despacho) es la superficie de trabajo de los profesionales: donde el abogado gestiona expedientes, dirige al Letrado IA, redacta escritos, consulta jurisprudencia, coordina causas con varios investigados, tramita firmas eIDAS, opera el CRM comercial y administra su despacho como tenant del Modelo B. Este capítulo recorre esa superficie módulo a módulo — dashboard, detalle de caso y sus 16 pestañas, Escritos v2, Claves del Caso, Causas, Jurisprudencia, Firma electrónica, CRM, gestión del despacho, superadmin, programa de colaboradores y onboarding del abogado — documentando componentes, endpoints, máquinas de estados y gobernanza reales.*

## 11.1 Roles, scoping y contexto de tenant

Todo el portal se sirve tras la barrera de sesión de Auth.js y un gate de rol. El árbol `/portal` exige `requireRole` con el conjunto {`ADMIN`, `COORDINADOR`, `COLABORADOR`, `LAWYER`}; un `CLIENT` o `PERITO` es rechazado antes de tocar la base de datos. Sobre ese gate de rol se superpone el *scoping multi-tenant*: cada expediente pertenece a una organización (`cases.org_id`) y el contexto activo (`getActiveContext`) restringe todas las consultas al `org_id` del profesional. Los roles de menor privilegio aplican además un scoping `restrictToOwn`: el `COLABORADOR` y el abogado no titular solo ven los expedientes en los que constan como letrado o co-letrado (tabla `case_collaborators`), nunca la cartera completa del despacho.

> ⚠ **Rol sin gate = IDOR.** El bug sistémico del Modelo B: añadir el rol `LAWYER` a una ruta sin que ésta pase por `requireCaseAccess` / `requireIntakeAccess` abre acceso a recursos de otros tenants por id. Toda ruta nueva bajo `cases/[id]`, `intakes/[id]` o `causas/[id]` debe empezar por su gate; hay lint de gates y tests de aislamiento cross-org que lo verifican.

## 11.2 El dashboard del profesional

La raíz `apps/portal/src/app/portal/page.tsx` es el panel de aterrizaje tras el login. Compone un saludo dinámico según la hora y el nombre del profesional, y ramifica según el rol y el estado de onboarding.

### 11.2.1 Cabecera, onboarding y banners condicionales

- **Onboarding del abogado (rol `LAWYER`)**: si el despacho aún no ha completado su alta, el dashboard muestra `DemoShowcaseCard` (acceso al caso demo ya trabajado) y `LawyerOnboarding` (pasos pendientes: verificación de colegiado opcional, primer caso, datos fiscales).
- **Banner de superadmin**: si el usuario es superadmin (doble gate, §11.11), aparece un banner de acceso a la consola de plataforma.
- **Banner de registro incompleto**: `RegistroIncompletoBanner` persiste hasta que el despacho completa datos fiscales y (opcionalmente) verificación de colegiado.

### 11.2.2 Los cuatro KPIs

Bajo la cabecera se pintan cuatro tarjetas de indicadores con cifras reales calculadas contra la BD dentro del scoping del tenant:

| # | KPI | Cálculo |
|---|---|---|
| 1 | Expedientes activos | Casos no cerrados dentro del `org_id` (con `restrictToOwn` para colaborador/no titular) |
| 2 | Consultas sin asignar | `intakes` entrantes pendientes de asignación de letrado |
| 3 | Plazos en 7 días | Plazos procesales con vencimiento dentro de la próxima semana |
| 4a | Tareas comerciales (rol `LAWYER`) | Tareas del CRM asignadas al profesional |
| 4b | Uso de IA del mes (staff) | Consumo de IA del despacho en el periodo actual |

La cuarta tarjeta es condicional al rol: para el abogado del despacho (`LAWYER`) muestra sus tareas comerciales; para el staff de plataforma (`ADMIN` / `COORDINADOR`) muestra el consumo de IA del mes.

### 11.2.3 Widgets

El cuerpo del dashboard agrupa seis widgets accionables: *expedientes prioritarios* (casos con plazos o urgencias), *análisis IA reciente* (últimos análisis de caso completados), *próximos plazos*, *agenda de hoy* (citas y reuniones), *consultas entrantes* (leads del CRM sin gestionar) y *uso de IA del mes*. Todos respetan el scoping del tenant y el `restrictToOwn` del rol.

## 11.3 El detalle de caso: `portal/casos/[id]`

El componente `apps/portal/src/components/portal/cases/CaseDetail.tsx` es el centro de gravedad del portal. Su cabecera concentra los controles de estado del expediente y su cuerpo organiza dieciséis pestañas en seis grupos temáticos con color.

### 11.3.1 Cabecera del expediente

- **Número de expediente copiable** (clic para copiar al portapapeles).
- `PosicionClienteSelect` — define si el cliente es defensa o acusación; invierte automáticamente el enfoque de todos los generadores de IA.
- `SituacionProcesalControls` — situación de libertad/detención/prisión provisional, que gobierna los escritos recomendados por fase (§11.5.3).
- `PenalCategorySelect` — categoría penal del asunto.
- `StatusPill` — estado del caso (`recibido`, `en_analisis`, `propuesta_enviada`, `en_curso`, `en_seguimiento`, `cerrado`), coloreado por tono (info/warn/ok).
- Pastillas de estado: *acceso del cliente*, *hoja de encargo* (`hoja_encargo_status`).
- `ResendClientAccess` — botón para reenviar el acceso al cliente (§11.13).

### 11.3.2 Las 16 pestañas (`TAB_GROUPS`)

La constante `TAB_GROUPS` define seis grupos, cada uno con su color y su conjunto de pestañas. La agrupación guía visualmente al abogado de lo descriptivo (Resumen) a lo operativo (Mensajes):

| Grupo (color) | Pestaña | Componente | Contenido / endpoints |
|---|---|---|---|
| Resumen *(dorado #9c7b3f)* | general | ResumenTab | Datos del caso, cliente, estado procesal, hitos |
| Caso y prueba *(azul #3f6493)* | documentos | DocumentsTab | Subida multipart a R2, análisis IA por documento; `cases/[id]/documents/**` |
| | grabaciones | RecordingsTab | Grabaciones de reuniones (Plaud), consentimiento, transcripción Whisper |
| | cuestionario | QuestionnaireTab | Preguntas propuestas por IA por rondas; `cases/[id]/questionnaire/**` |
| | testigos | WitnessesTab | Gestión de testigos del caso |
| | peritos | ExpertsTab | Peritos y peritajes vinculados |
| | valoracion | ValoracionTab («Mi criterio») | Criterio del letrado, ponderación de la prueba |
| Análisis IA *(violeta #6d5b9e)* | analisis-ia | CaseAnalysisTab | Informe de IA del caso completo; `cases/[id]/analysis` |
| | jurisprudencia | JurisprudenciaTab | Precedentes vinculados al caso, Motor Favorable |
| | chat-ia | ChatExpedienteTab | Chat conversacional sobre el expediente (SSE); `cases/[id]/chat/**` |
| Escritos *(verde #3f7d5f)* | escritos | EscritosTab | Catálogo v2 de ~28 herramientas de redacción; `cases/[id]/escritos/[kind]` |
| Vistas y juicios *(magenta #8a4b6b)* | preparacion | PreparacionVistasTab | Preparación de vista/juicio, simulador de interrogatorio |
| Seguimiento *(ámbar #c2772e)* | actividad | ActividadTab | Bitácora de eventos del caso (`case_events`) |
| | reunion | ReunionTab | Notas y agenda de reuniones |
| | checklist | ChecklistTab | Lista de comprobación del expediente |
| Mensajes *(teal #2f8080)* | mensajes | MensajesTab | Mensajería con el cliente; polling y ventana de edición |

### 11.3.3 `DocumentsTab` — documentos y análisis por documento

La subida sigue el patrón de tres fases hacia R2 (init/subida directa/complete) documentado en el capítulo de API, con validación de extensión y coincidencia MIME en servidor. Cada documento se analiza individualmente con IA y su tarjeta refleja el estado del análisis mediante una máquina de estados:


Fig. 11.1 — Estados del análisis por documento con refresco por polling.

El componente hace *polling* con SWR cada 3 segundos mientras haya documentos en `pending`/`processing`, de modo que el abogado ve avanzar el análisis sin recargar. Además del análisis de ficheros, existe la ruta `/analyze-text` para pegar texto suelto (una diligencia, un correo) y analizarlo sin subir un fichero.

### 11.3.4 `QuestionnaireTab` — cuestionario dirigido por IA

La IA propone preguntas al cliente por rondas (hasta un tope de 100 preguntas por caso) para cerrar lagunas del relato. La pestaña ofrece un filtro con chips `all` / `answered` / `pending` y un contador, para que el abogado vea de un vistazo qué queda por responder. El cuestionario es una pieza de evidencia crítica: sus respuestas alimentan el análisis y los escritos (cap. de completitud de evidencia).

### 11.3.5 `CaseAnalysisTab` — el informe de IA del caso

Es el análisis integral del expediente. Se compone de varias secciones renderizadas a partir de una salida estructurada:

| Sección | Contenido |
|---|---|
| `ResumenEjecutivo` | Síntesis del caso para lectura rápida |
| `Diagnostico` | Calificación jurídica, elementos del tipo, riesgos |

| Sección | Contenido |
|---|---|
| `Estrategia` | Líneas de defensa/acusación recomendadas |
| `Timeline` | Cronología reconstruida de los hechos |
| `PartyGraph` | Grafo de partes e intervinientes (ReactFlow) |
| `PrognosisTrend` | Evolución del pronóstico a lo largo de las revisiones |
| `Plazos` | Plazos procesales detectados |
| `ClarificationsPanel` | Aclaraciones del cliente incorporadas al re-análisis |
| `JurisprudenciaFavorable` | Precedentes que favorecen la posición del cliente (Motor JUR-02) |
| `EvidenceCoverageBanner` | Cobertura de evidencia: qué documentos/respuestas se han considerado |

El análisis tiene su propia máquina de estados: `pending` → `processing` → `done`, con `failed` ante error y `stale` cuando entra nueva evidencia que invalida el análisis previo y conviene relanzarlo.

> ℹ **Cobertura de evidencia.** El `EvidenceCoverageBanner` es la materialización de la regla de completitud de evidencia: ningún informe se da por bueno sin nombrar qué documentos, respuestas del cliente y peritajes se han leído y considerado; si algo falta o falló, se anuncia y se ofrece relanzar.

#### 11.3.6 `ChatExpedienteTab` y `PreparacionVistasTab`

`ChatExpedienteTab` abre una conversación en lenguaje natural sobre el expediente completo; las respuestas se sirven por Server-Sent Events token a token. `PreparacionVistasTab` prepara la vista o el juicio (`kind` = `vista` / `juicio`): genera guiones, incluye el `InterrogationSimulator` (simulador de interrogatorio para ensayar preguntas y respuestas) y permite exportar el material a PDF y DOC.

## 11.4 Escritos v2

`EscritosTab` es el arsenal de redacción del Letrado IA. Reúne alrededor de 28 herramientas cuya presentación se adapta a la posición procesal del cliente.

#### 11.4.1 Catálogo adaptado a la posición del cliente

Según `posicionCliente` (defensa o acusación) se muestra uno de dos catálogos: `SECCIONES_DEFENSA` (7 secciones) o `SECCIONES_ACUSACION` (6 secciones). Un buscador filtra en vivo el catálogo de herramientas por nombre. La cabecera del catálogo destaca en dorado el bloque «Empieza aquí — Claves del caso».

#### 11.4.2 «Empieza aquí» y recomendados por fase

El bloque dorado usa una `EscritoCard` con `kind=claves_del_caso` y el componente `ClavesAcciones`, que convierte las acciones sugeridas por el análisis en tareas del caso. A continuación, el catálogo recomienda escritos según la fase procesal, con una regla dura: *la libertad manda*. Si el cliente está detenido o en prisión, esos escritos suben al primer plano por encima de la fase de instrucción:

| Situación / fase | Escritos recomendados |
|---|---|
| Detenido | `habeas_corpus` + `oposicion_prision` |
| Prisión provisional | `solicitud_libertad` + `recurso_auto_prision` |
| Preprocesal / policial | Escritos de fase inicial |
| Instrucción / diligencias previas | Proposición de prueba, oposición, personación |
| Citado | Preparación de declaración |
| Juicio oral | Escrito de defensa, conclusiones |
| Sentencia | Recursos (reforma, apelación, casación) |
| Ejecución | Escritos de ejecución |

El `EvidenceCoverageBanner` vuelve a aparecer aquí: no se redacta sobre evidencia sin considerar.

#### 11.4.3 El registro `GENERATORS` y el pipeline de generación

El endpoint `apps/portal/src/app/api/cases/[id]/escritos/[kind]/route.ts` mantiene un registro `GENERATORS` con 40 `ALLOWED_KINDS`, cada uno mapeado a su función `generate-*.ts` del paquete `@jbhlegal/ai`. El pipeline de un `POST` (generar un escrito) es una secuencia de gates y control de calidad:


Fig. 11.2 — Pipeline de generación de un escrito con auditoría y autocorrección.

La generación persiste en `case_drafts` (estado `generating` → `ready`) y cada versión en `case_draft_versions`. La *auditoría IA* (`runAndPersistAudit`, skill `legal-document-audit`, con 1 reintento) revisa el escrito con triple mirada (letrado, fiscal, juez); si el resultado no es «apto», entra el bucle de autocorrección `refineLegalDraft` antes de marcar `ready`.

Los dos gates de entrada del pipeline no son intercambiables. `assertOrgCanWrite` resuelve la pregunta comercial —¿este tenant tiene derecho a generar trabajo?— y es donde se aplica la lógica de «primer caso gratis» y de suscripción vencida; un despacho sin plan válido recibe aquí el 402 con el upsell. `assertAiBudget` resuelve la pregunta de consumo —¿queda presupuesto de IA en el tramo actual?— y es donde se materializa el bloqueo por fallo de cargo. Solo si ambos pasan se marca el borrador como `generating` y se invoca al modelo; ese orden garantiza que nunca se consume cómputo de IA por un tenant que no puede pagarlo. La selección de modelo respeta la doctrina de «cada opción su campeón»: las Claves del Caso van siempre al modelo de máximo razonamiento (`bestLegalReasoningModel()`), mientras que el resto de escritos usa `pickModel`, que pondera el *kind* del escrito y el *tier* del despacho para elegir entre los modelos disponibles.

El versionado no es un lujo: en un escrito penal, poder reconstruir qué texto se generó, qué corrigió la auditoría y qué editó el abogado a mano es una garantía de trazabilidad. Cada `PATCH` crea una fila en `case_draft_versions`, de modo que la historia completa del documento —de la primera generación al escrito validado— queda accesible. El gate humano del `PUT` (CR-3) es el punto donde el sistema deja de suponer que un texto de IA es entregable: nada se considera revisado hasta que un profesional colegiado lo valida explícitamente.

### 11.4.4 Versionado y gate humano

- `PATCH` — edita el escrito y crea una nueva versión (versionado en `case_draft_versions`).
- `PUT` — valida el escrito: es el **gate humano CR-3**, el punto donde el abogado da el visto bueno final antes de considerar el escrito revisado.

> ✓ **Grounding de citas.** Toda cita legal (artículo o sentencia) generada pasa por el cierre de la restricción crítica C1: el LLM no inventa jurisprudencia, se apoya en candidatos reales de la biblioteca y marca lo no verificado con `[VERIFICAR]`. La auditoría y el gate humano cierran el círculo.

## 11.5 Claves del Caso

La herramienta estratégica `kind=claves_del_caso` es la joya del arsenal: un análisis con prompt de 14 secciones que orienta la estrategia del caso. Su enfoque se auto-invierte según `posicionCliente` (defensa o acusación). Su contenido es **solo para el abogado** — nunca visible para el cliente. Usa siempre el modelo de máximo razonamiento `bestLegalReasoningModel()`, configurable por la variable de entorno `BEST_LEGAL_REASONING_MODEL`. El componente `ClavesAcciones` convierte las acciones que sugiere el análisis en tareas concretas del expediente.

## 11.6 Causas: coordinación de multi-investigado

El módulo de causas (`portal/causas/[id]`, gate `requireCausaAccess`) modela un asunto penal común que agrupa a varios investigados. Una *causa* agrega N investigados; cada investigado es una fila de `cases` con `causa_id`. Esto permite al despacho llevar de forma coordinada una defensa múltiple sin mezclar los expedientes individuales.

### 11.6.1 Modelo de datos

| Tabla | Columnas / campos relevantes |
|---|---|
| `causas` | `causaNum`, `caratula`, `area`, `status` |
| `causa_analyses` | `inconsistenciasJson`, `corroboracionesJson`, `conflictosInteresJson`, `posturaPorInvestigadoJson`, `tesisConjunta` |
| `causa_core_questions` | Preguntas núcleo de la causa |
| `causa_drafts` | Escritos a nivel de causa |

### 11.6.2 Seudonimización y detector de incoherencias

Antes de enviar nada a Anthropic, la causa aplica *seudonimización de PII* (`redact-pii.ts`): los nombres reales de los investigados se sustituyen por «Investigado N» y se restauran en la salida. Es una obligación reforzada por el art. 10 del RGPD (datos penales) y una defensa en profundidad que se suma a la política ZDR (Zero Data Retention) del proveedor.

> ⚠ **Datos penales = categoría especial.** La seudonimización no es opcional en causas: la comparación entre investigados es precisamente donde más PII confluye. El detector de contradicciones (`detect-comparativa-contradicciones.ts`) opera **siempre con Opus** y siempre con los investigados seudonimizados, buscando incoherencias entre las versiones de unos y otros.

Los generadores de causa producen documentos de coordinación: `acuerdo-defensa-conjunta`, `estrategia-coordinada` y `conflicto-interes` (que evalúa si el despacho puede defender simultáneamente a varios investigados sin conflicto).

## 11.7 Jurisprudencia: biblioteca compartida y Motor Favorable

El módulo `portal/jurisprudencia` es una biblioteca de precedentes compartida globalmente entre los despachos, con aislamiento estricto por caso.

### 11.7.1 Visibilidad y aislamiento

La tabla `legal_precedents` tiene un campo `visibility` con tres valores: `private_org` (privado del despacho), `pending_common` (propuesto para la biblioteca común) y `common` (compartido globalmente). El vínculo entre un precedente y un caso concreto se registra en `legal_precedent_links`; el aislamiento por caso se reimpone siempre en SQL, de modo que ver la biblioteca común nunca filtra qué precedentes usa otro despacho en sus casos.

### 11.7.2 Búsqueda IA por RAG

El endpoint `api/jurisprudence/search` implementa una búsqueda semántica RAG:

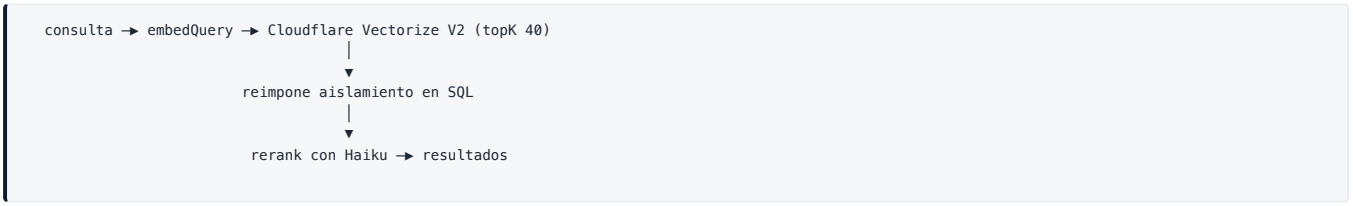


Fig. 11.3 — Canalización RAG de búsqueda de jurisprudencia.

Los embeddings usan `voyage-law-2` (1024 dimensiones) de Voyage o, alternativamente, `bge-m3` de Cloudflare; el índice vive en Vectorize y el reranking final lo hace Haiku. La deduplicación de precedentes usa *simhash*.

El orden de la canalización es defensivo por diseño. La búsqueda vectorial devuelve un `topK` generoso (40) precisamente porque el paso siguiente —reimponer el aislamiento en SQL— *reduce* ese conjunto: filtra los precedentes que el despacho no tiene derecho a ver, de modo que un índice vectorial compartido nunca se convierte en una fuga de qué precedentes usa otro bufete. Solo sobre el conjunto ya aislado actúa el reranking con Haiku, que reordena por relevancia semántica fina. La asignación de modelos por etapa es económica: embeddings y reranking usan modelos baratos y rápidos (Voyage/Cloudflare y Haiku), y el modelo caro (Opus) se reserva para el análisis del caso, no para tareas de recuperación.

### 11.7.3 Motor de Jurisprudencia Favorable (JUR-02)

El motor `jurisprudencia-favorable.ts` (JUR-02) es un módulo crítico: dado el perfil del caso y un conjunto de *candidatos reales* de la biblioteca, devuelve los precedentes que favorecen a la posición del cliente, cada uno con su `lineaDefensa`, el `motivo` por el que favorece y una `relevancia` (0–100).

> ✓ **Grounding C1.** El LLM solo *puntúa* candidatos reales; nunca inventa una sentencia. Solo completa con datos de CENDOJ el ECLI y la materia de precedentes que ya existen. En los escritos (`JURIS_ESCRITO_KINDS`) las citas resultantes se marcan con `[VERIFICAR]` hasta contraste.

El módulo se completa con un motor de deduplicación, colecciones, favoritos y reportes de la biblioteca.

## 11.8 Firma electrónica eIDAS

La app de firmas (`apps/firmas`, dominio `firmas.jbhasesorialegal.com`) y el paquete `packages/signing` implementan la firma electrónica avanzada. El servicio central es `signature-service.ts`.

### 11.8.1 Flujo de firma

1. `verifySigningToken` valida el JWT del enlace de firma (firmado con `SIGNING_JWT_SECRET`, cuyo valor es out-of-band en `shared/*.env` del servidor).
2. Validaciones de consentimiento: el firmante acepta los términos y teclea su nombre, que debe coincidir con el `signerNameSnapshot` registrado.
3. Integridad del documento: se recomputa el SHA-256 del PDF y se verifica contra `documentHashSha256`; ante *mismatch* la solicitud se revoca (el documento cambió tras emitir el enlace).
4. Reclamación atómica del estado: la transición `pending` / `viewed` → `signed` se hace con una escritura condicional que solo procede si `affectedRows === 1`, evitando dobles firmas.

El servicio soporta dos modos: `case` (firma vinculada a un expediente, típicamente la hoja de encargo) y `collaborator` (firma de un colaborador).

### 11.8.2 Hoja de encargo

Cada caso lleva `cases.hoja_encargo_status` con valores `pendiente` / `firmada` / `no_aplica`. La firma de la hoja de encargo puede acompañarse de un *setup fee* opcional (checkout de 500€) según el modelo de negocio.

### 11.8.3 Modelo de datos de firma

| Tabla | Campos |
|---|---|
| `signature_requests` | `tokenJti` (unique), `status` (`pending` / `viewed` / `signed` / `revoked` / `expired`), `documentHashSha256`, `expiresAt` |
| `signatures` | Evidencias: `signedAt`, `signerName`, `signerEmail`, `ipAddress`, `userAgent`, `documentHashSha256` |

### 11.8.4 Hoja de evidencia y encaje eIDAS

Al firmar, `mergeWithEvidence` (`packages/signing/src/merge.ts`) anexa al PDF una **HOJA DE EVIDENCIA DE FIRMA ELECTRÓNICA** con el hash del documento, el nombre y email del firmante, su IP y user-agent, el `jti` del token y el `signedAt`. El resultado es una firma electrónica avanzada bajo el art. 25 del Reglamento (UE) 910/2014 (eIDAS), sin necesidad de OTP por SMS.

La verificación del hash es la pieza que da valor probatorio a todo el esquema. El `documentHashSha256` se calcula sobre el PDF en el momento de emitir la solicitud de firma y se recomputa en el momento de firmar; si difieren, el documento fue alterado entre ambos instantes y la solicitud se revoca automáticamente en lugar de firmarse. Esto cierra el ataque de «cambiar el contrato tras enviarlo a firmar». La reclamación atómica del estado (`affectedRows === 1`) cierra el ataque complementario —dos pulsaciones simultáneas del botón de firmar—: solo la primera transición `pending` / `viewed` → `signed` tiene efecto, la segunda encuentra la fila ya firmada y no produce una segunda firma. El `tokenJti` único por solicitud impide reutilizar un enlace de firma, y el `expiresAt` lo caduca pasado su plazo.

> ℹ **eIDAS art. 25.** La firma electrónica avanzada no se deniega efecto jurídico ni admisibilidad como prueba por el mero hecho de ser electrónica. La hoja de evidencia anexa (hash + identidad + IP/UA + sello temporal) es lo que sostiene su valor probatorio.

## 11.9 CRM comercial: `/portal/crm`

El CRM (`requireCrmContext`) es el panel comercial del despacho. Opera un pipeline kanban sobre la tabla `intakes` (enfoque B: cada intake es un lead), con ficha de lead 360º, entrantes, presupuestos, tareas, facturas e informes.

### 11.9.1 Máquina de estados del pipeline

`status-machine.ts` define los estados permitidos del lead. El tablero tipo Pipedrive (`board-service.ts`, `BOARD_PAGE_SIZE` = 30, cabecera con total y suma de valor) los presenta como columnas:

| Estado | Significado |
|---|---|
| `pending` | Entrante sin procesar (se muestra como «nuevo» en el tablero) |
| `nuevo` | Lead nuevo asignado |
| `contactado` | Primer contacto realizado |
| `en_revision` | En estudio |
| `cualificado` | Lead cualificado |
| `cita_agendada` | Cita concertada |
| `presupuesto_enviado` | Presupuesto emitido |
| `pendiente_firma_pago` | Presupuesto aceptado, pendiente de firmar/pagar |
| `contratado` | Convertido en cliente (existe `caseId`) |
| `descartado` | Lead perdido |

### 11.9.2 Gates de realidad

`stage-gates.ts` impide mentir al pipeline: una etapa avanzada exige que exista la realidad que representa. `presupuesto_enviado` exige un presupuesto realmente enviado; `pendiente_firma_pago` exige un presupuesto aceptado; `contratado` exige un `caseId`. La promoción de lead a intake/caso (`promote-lead.ts`) es idempotente.

### 11.9.3 Presupuestos con aceptación pública

Los presupuestos (`crm_presupuestos`) tienen su propia máquina de estados: `borrador` → `enviado` → `visto` → `aceptado` / `rechazado`, con caducidad perezosa (`caducado`). El número se calcula `MAX+1` dentro de una transacción. `markSent` genera un token cuyo hash (`sha256`) se guarda en `public_token_hash`; el cliente acepta el presupuesto sin sesión en `/presupuesto/[token]` vía `api/public/presupuestos/[token]/decide`. Al **aceptar**, el intake pasa a `pendiente_firma_pago` y se crea automáticamente una tarea de alta «enviar hoja de encargo» con vencimiento a 24h.

#### 11.9.4 Facturas de honorarios

Las facturas de honorarios (`lawyer_invoices`) representan el cobro directo del abogado a su cliente — *no* pasan por JBH — con la comisión de plataforma registrada en snapshot en el momento de la factura.

#### 11.9.5 IA del CRM

`ia.ts` (`runLeadIa`) dota al pipeline de inteligencia:

- `classifyIntake` — detecta urgencias procesales y crea automáticamente una tarea de prioridad alta con vencimiento a 4h.
- *Siguiente mejor acción* — recomienda el próximo paso comercial.
- Borradores de email/WhatsApp (`generate-crm-message.ts`): `CRM_MESSAGE_KINDS` = `primer_contacto` / `pedir_documentacion` / `seguimiento` / `reactivacion`. Todo borrador pasa por `lintBannedPhrases` (compliance: no prometer resultados, distinguir plataforma de despacho).

#### 11.9.6 Automatizaciones y reportes

El proceso pm2 `jbhlegal-crm-automations` (cron `15 * * * *`, cada hora al minuto 15) ejecuta cuatro reglas:

| Regla | Acción |
|---|---|
| R1 — SLA | Lead entrante sin atender 24h → alerta/tarea |
| R2 — Presupuesto | Seguimiento del presupuesto enviado sin respuesta a 72h |
| R3 — Lead frío | Lead sin actividad 14 días → reactivación |
| R4 — Reality-sync | Si el caso ya existe, sincroniza el lead a `contratado` |

Los informes (`report-service.ts`) ofrecen análisis de cohorte y embudo, con previsión de valor calculada como valor × probabilidad por etapa.

Los gates de realidad merecen énfasis porque atacan un vicio clásico de los CRM: el *optimismo del pipeline*, donde los leads se arrastran a etapas avanzadas para inflar la previsión sin que exista la realidad subyacente. Aquí no se puede mover un lead a `presupuesto_enviado` sin un presupuesto realmente emitido, ni a `contratado` sin un `caseId`. La consecuencia es que la previsión de valor del embudo se apoya en hechos verificables, no en la esperanza del comercial. La regla R4 de las automatizaciones (reality-sync) refuerza esto en la dirección inversa: si un caso ya existe pero el lead sigue en una etapa anterior, el sistema lo promueve solo a `contratado` con traza del motivo, evitando que el tablero mienta por omisión.

La aceptación pública de presupuestos por token es el mecanismo que convierte una conversación comercial en un contrato sin fricción: el cliente recibe un enlace, ve el presupuesto sin necesidad de cuenta, y al aceptar dispara toda la maquinaria posterior —cambio de etapa, tarea de «enviar hoja de encargo» a 24h, y más adelante la promoción del lead a expediente con copia de los honorarios acordados. El hash del token en `public_token_hash` (nunca el token en claro) protege el enlace igual que en el resto de flujos públicos del sistema.

## 11.10 Gestión del despacho (Modelo B)

El Modelo B convierte a cada despacho en un tenant con su propia suscripción, equipo, marca y consumo de IA. Las páginas viven bajo `portal/me`.

#### 11.10.1 Modelo de datos multi-tenant

| Tabla | Campos clave |
|---|---|
| `organizations` | `type` (`jbh_platform` / `law_firm`), `branding_json` |
| `memberships` | `role` (`OWNER` / `FIRM_ADMIN` / `LAWYER` / `STAFF` / `COLLABORATOR` / `PERITO`) |
| `organization_subscriptions` | `plan` (`profesional` / `despacho` / `firma`), `status` (`none` / `trialing` / `active` / `past_due` / `canceled`) |

#### 11.10.2 Las páginas `me/*`

| Página | Función |
|---|---|
| `me/despacho` | Suscripción (3 planes), Stripe Checkout + Customer Portal, `OrgAiConsumptionPanel`, `OrgByokPanel`, `InternalCouponBox`, `MoneyCompass` |
| `me/equipo` | Miembros e invitaciones del equipo |
| `me/facturacion` | Datos fiscales del despacho |
| `me/marca` | Branding (logo, color primario) |
| `me/supervision` | Panel del titular multi-asiento |
| `me/verificacion` | Verificación de colegiado (§11.13.4) |

#### 11.10.3 IA medida (metered) y primer caso gratis

El paquete `packages/payments/ai-budget.ts` implementa el cobro de IA por consumo. Un despacho paga su suscripción más el consumo de IA a coste, facturado en tramos de 100€ (`AI_USAGE_THRESHOLD_CENTS` = 10000). El importe facturable aplica un margen: `billableAiCents = ceil(raw × AI_BILLING_MARGIN)` con `AI_BILLING_MARGIN` = 1.35 (+35%). Un fallo de cargo bloquea la IA.

> ⚠ **Nunca desglosar «coste + margen».** El margen del 35% está integrado en `billableAiCents`; el marketing y la UI presentan el cobro como «por uso», nunca como coste más margen.

El *primer caso gratis* se implementa en `guardOrgWrite`: en plan gratuito el despacho puede crear y trabajar su primer expediente; el segundo exige suscripción. El anti-abuso «un caso por NIF» actúa como cuarta capa de defensa contra la creación repetida de cuentas.

El diseño de tramos de 100€ (`AI_USAGE_THRESHOLD_CENTS` = 10000) convierte un consumo continuo y granular de IA en una facturación predecible y espaciada: en lugar de micro-cargos por cada llamada al modelo, el consumo se acumula y se cobra al cruzar cada umbral. Esto reduce la carga sobre Stripe y da al despacho una factura legible. El bloqueo por fallo de cargo es la contrapartida: si un cobro de exceso falla, la IA se detiene hasta que el despacho repara su método de pago, protegiendo a la plataforma de acumular deuda incobrable de cómputo. El margen del 35% (`AI_BILLING_MARGIN` = 1.35) se aplica sobre el coste bruto para calcular `billableAiCents`, pero es un detalle de implementación que jamás se expone: hacia fuera el modelo es «pago por uso», sin desglosar coste y margen.

#### 11.10.4 Cupones internos y BYOK

- **Cupones internos** (`internal_coupons`): el código se guarda como `code_hash` salado (nunca en claro); soportan un modo compartido con filas hija. El secreto de firma es `INTERNAL_COUPON_SECRET` (valor out-of-band).
- **BYOK** (`api/me/despacho/ai-key`): el despacho puede traer su propia clave de Anthropic; el endpoint la valida contra Anthropic, la guarda cifrada con AES-256-GCM en `organizations`, y `enterOrgAiContext` hace que todo el flujo de IA de sus casos herede esa clave.

## 11.11 Superadmin

La consola de plataforma (`portal/admin/superadmin`) está protegida por un doble gate: `isSuperadmin` exige rol `ADMIN` *y* pertenencia a una allowlist de emails. Toda acción queda auditada.

| Módulo | Función |
|---|---|
| Stats | Consola de métricas de plataforma |
| Verificaciones de colegiado | ASISTIDAS: `colegiado_assisted_requests` (`nif_hash` + `nif_last3`), documento en R2, SLA 24h hábiles |
| Monitor de actividad | Quién entra, qué consume cada despacho/caso |
| «Ver como» (impersonación) | SOLO LECTURA: `startImpersonation`, cookie firmada; `assertNotImpersonating` lanza `ReadOnlyImpersonationError` (403) ante cualquier escritura |
| Marketing / SEO | Monitor de Search Console (GSC) |
| Link-building | Tracker con `verify`: fetch server-side, busca backlinks, evalúa dofollow/indexabilidad; NO crea enlaces |
| Users · Soporte · Analytics · AI-usage · Comisiones | Gestión transversal de plataforma |

> ⚠ **Impersonación sin escritura.** «Ver como» es estrictamente de lectura: el superadmin ve el portal como lo ve el despacho, pero cualquier mutación es interceptada por `assertNotImpersonating` y devuelve `read_only_impersonation` (403). Nunca se escribe en nombre de un tenant.

## 11.12 Programa de colaboradores / referidos (E1)

Bajo el flag `REFERRAL_PROGRAM_ENABLED`, el panel `/portal/programa-colaboradores` gestiona referidos y resellers. El modelo de datos es contable y a prueba de manipulación:

| Tabla | Rol |
|---|---|
| `referral_profiles` | Perfil del referidor |
| `referral_aliases` | Alias públicos (ruta `/r/{alias}`) |
| `referral_links` | Enlaces de referido |
| `referral_attributions` | Atribución first-touch, una por sujeto |
| `referral_commissions` | Comisiones en cuarentena 30 días |
| `referral_ledger` | Libro append-only; los saldos se derivan, no se editan |

Los niveles de comisión van del 5% al 12% según volumen. La ruta pública `/r/{alias}` siembra la cookie `jbh_ref` (90 días) y aplica atribución first-touch. El dinero está bloqueado hasta que se abran los gates (`REFERRAL_PAYOUTS_ENABLED`).

## 11.13 Enviar acceso al cliente y visibilidad

El botón «Enviar acceso al cliente» y su enlace one-time cierran el puente entre el trabajo del abogado y el área del cliente.

### 11.13.1 El endpoint de invitación

`api/cases/[id]/client-invitation` (gate `requireCaseAccess` + `assertOrgCanWrite`; el `COLABORADOR` recibe 403) ramifica según el estado del alta del cliente:

- **Alta no completada** → `regenerateClientInvitation`: token con TTL de 7 días, URL `/invitacion?t=`.
- **Alta completada** → enlace directo one-time (`client_access_links`, hash `sha256`, 7 días, un solo uso), URL `/acceso?t=`.

### 11.13.2 «Qué ve tu cliente»

El componente `ClientVisibilityPanel` (`case-client-visibility.ts`) da al abogado interruptores por sección y por caso para decidir qué ve el cliente. La política por defecto separa el trabajo del letrado de la colaboración con el cliente:

| Nace OCULTO (trabajo del letrado) | Nace VISIBLE (colaboración) |
|---|---|
| analisis, informe, preparacion, testigos, cronologia, pasos | documentos, cuestionario, mensajes |

ℹ **El criterio del letrado no se filtra.** El análisis de IA, las Claves del Caso y la preparación de vistas nacen ocultos: son el criterio interno del abogado. El cliente solo ve lo que aporta valor a su relación (sus documentos, el cuestionario que debe rellenar, los mensajes) hasta que el letrado decide compartir más.

### 11.13.3 Onboarding del abogado

Desde 2026-07-17 el alta del despacho no exige colegiación obligatoria. El registro (`registro-despacho`) permite operar y trabajar el *primer caso sin verificar* (`assertOrgCanWrite`), con un `RegistroIncompletoBanner` persistente que recuerda completar el perfil. Existe además `/portal/demo`: un caso ficticio ya trabajado (`DEMO_SHOWCASE_CASE_ID`, con `noindex`) para que el abogado vea el producto en marcha antes de subir su primer expediente real.

### 11.13.4 Verificación criptográfica de colegiado

La verificación en `/portal/me/verificacion` acepta dos vías: firmar con Autofirma o subir un fichero (migración `0121`). El paquete `packages/colegiado` realiza la verificación criptográfica del certificado ACA (Abogacía): validación de `cms`, cadena de confianza (`chain`), revocación (`revocation`) y confianza (`trust`). Cuando la verificación es asistida, la solicitud entra en el flujo de superadmin (`colegiado_assisted_requests`, §11.11) con SLA de 24h hábiles.

La decisión de hacer la colegiación *opcional* para empezar (desde 2026-07-17) es de producto: exigir verificación antes de dejar probar el producto elevaría demasiado la barrera de entrada. El equilibrio se logra con capas: el despacho puede registrarse y trabajar su primer caso sin verificar (`assertOrgCanWrite` lo permite), pero un banner persistente (`RegistroIncompletoBanner`) recuerda el paso pendiente, y la verificación —cuando llega— es criptográficamente rigurosa (validación completa del certificado ACA, no una simple foto del carné). En el flujo asistido, el superadmin coteja el documento subido con el `nif_hash` y los tres últimos dígitos del NIF (`nif_last3`) que se almacenan sin exponer el NIF completo, aplicando minimización de datos incluso en el proceso de verificación.

El caso demo (`/portal/demo`, `DEMO_SHOWCASE_CASE_ID`) cumple una doble función. Comercialmente, deja que un abogado dubitativo vea el Letrado IA trabajando sobre un expediente ficticio completo —con su análisis, sus escritos, su jurisprudencia— antes de comprometer un caso real. Técnicamente, se sirve con `noindex` para que ese contenido ficticio nunca contamine los buscadores. El showcase se apoya en un caso construido para exhibir capacidades concretas, incluida la diligencia de la IA al detectar incoherencias en el relato.

✓ **Separación dura de responsabilidades.** El portal mantiene tres separaciones invariantes: (1) tenants aislados por `org_id` con gates en toda ruta `[id]`; (2) trabajo del letrado oculto al cliente por defecto; (3) datos penales seudonimizados antes de salir a Anthropic. Ninguna feature nueva debe romper estas tres líneas.

# 12

# El área del cliente

> *La app del cliente (`apps/cliente`) es la ventana que el particular tiene sobre su propio expediente penal: entra sin contraseña mediante un código de un solo uso, se da de alta guiado por un asistente, firma su hoja de encargo, responde el cuestionario, aporta documentos, conversa con su letrado y ejerce sus derechos RGPD por autoservicio. Este capítulo documenta el login OTP, el onboarding por token, el dashboard «Mis casos» y su detalle gobernado por la visibilidad que decide el abogado, la mensajería, los pagos con doble consentimiento, los tours guiados, el autoservicio de derechos del interesado y la distinción entre los modelos de negocio A y B.*

El área del cliente parte de una premisa distinta a la del portal del abogado: el usuario no es un profesional que gestiona una cartera, sino un particular —a menudo bajo estrés procesal— que sigue un único asunto que le concierne personalmente. De ahí tres decisiones transversales que recorren toda la app: entrada sin contraseña (nada que recordar ni que se filtre), interfaz que muestra *solo* lo que el letrado decide compartir, y acompañamiento explícito (tours, mensajes de error accionables, un solo aviso urgente a la vez). Toda la app se sirve con `robots: noindex`: ninguna pantalla del cliente debe aparecer en un buscador. El acceso está gobernado por tres vías de entrada mutuamente excluyentes —el código OTP por email, la invitación de onboarding y el enlace de acceso one-time— y protegido por un middleware que, además del gate de sesión, bloquea toda escritura de las cuentas demo.

## 12.1 Login OTP: entrada sin contraseña

El cliente no tiene contraseña. La configuración `packages/auth/src/client-config.ts` emite un código de 6 dígitos por email (`randomInt(100000, 1000000)`) con TTL de 30 minutos. Es deliberadamente un *código que se teclea*, no un magic link.

> ⚠ **Por qué código y no magic link.** Los escáneres de seguridad de correo (antivirus corporativos, proxies) pre-visitan los enlaces de los correos entrantes. Un magic link se consumiría solo antes de que el cliente lo pulse. Un código de 6 dígitos que el usuario teclea en la pantalla de verificación es inmune a ese pre-consumo.

### 12.1.1 Flujo y anti-enumeración

El flujo es `/cliente/login` (introduce email) → `/cliente/login/verify?email=` (teclea el código). La respuesta es neutra ante enumeración: siempre «Si tu email está registrado, recibirás un código», sin revelar si la cuenta existe y sin redirigir al registro. El endpoint OTP tiene rate-limit de 12 intentos por cada 10 minutos por IP.

### 12.1.2 Sesión

La sesión usa la estrategia *database* de Auth.js con duración de 30 días deslizante (`SESSION_MAX_AGE_DAYS`), refrescada como muy pronto cada 24h de actividad (`updateAge`) para no arrastrar un rol o permiso obsoleto. La cookie es `__Secure-authjs.session-token`. La estrategia *database* (frente a JWT) es deliberada: al vivir la sesión en la BD, revocarla es instantáneo (borrar la fila) y el permiso efectivo se re-lee del servidor, de modo que un cambio de acceso —por ejemplo, un caso que el letrado deja de compartir— surte efecto sin esperar a que expire un token firmado. El `updateAge` de 24h evita, además, escribir en la tabla de sesiones en cada petición: solo se prolonga la caducidad una vez al día por sesión activa.

### 12.1.3 Cuenta demo

Las cuentas demo (`users.is_demo`) son de solo lectura: un middleware bloquea todo método mutante (`POST` / `PUT` / `PATCH` / `DELETE`) con 403. Permiten enseñar el producto sin riesgo de alterar datos.

### 12.1.4 La ruta `/acceso`: enlace one-time

Cuando el abogado envía un acceso directo (cap. 11 §11.13), el cliente aterriza en `/acceso?t=`. El token se contrasta contra `client_access_links` (hash `sha256`, validez 7 días, un solo uso). El consumo usa un *patrón intersticial anti-prefetch* (evita que el navegador o un escáner consuman el enlace por prefetch) y una escritura atómica que solo procede si `affectedRows === 1`. Al consumirlo, se crea una sesión de 30 días.


Fig. 12.1 — Las tres vías de entrada del cliente: código OTP, invitación y acceso one-time.

## 12.2 Onboarding por token

El alta guiada se dispara desde una invitación. La tabla `client_invitations` guarda el `token_hash` (`sha256`), con TTL de 7 días, estado (`pending` / `consumed` / `revoked` / `expired`) y un contador `attempts` anti-fuerza-bruta.

### 12.2.1 Bienvenida y asistente de 4 pasos

El cliente entra por `/invitacion?t=` (pantalla de bienvenida con la marca del despacho) y pasa a `/onboarding?t=`, donde el `OnboardingWizard` lo guía por cuatro pasos, reanudable, con barra de progreso con la marca del despacho:

| Paso | Contenido |
|---|---|
| 1 — Datos | Nombre, DNI, teléfono, dirección + checkbox RGPD obligatorio |
| 2 — Hoja | Genera y firma la hoja de encargo (firma electrónica simple, eIDAS art. 25.1); registra IP + navegador + hash; **consume la invitación** |
| 3 — Cuestionario | Responde el cuestionario del caso (post-firma) |
| 4 — Documentación | Aporta documentos (post-firma) |

Los pasos posteriores a la firma se sirven por rutas autenticadas por token: `/api/onboarding/{token}/cases/{caseId}/...`. La firma de la hoja de encargo en el paso 2 es lo que consume la invitación, de modo que un abandono previo se puede reanudar.

```
/invitacion?t=  (bienvenida, marca del despacho)
     │
     ▼
/onboarding?t=   OnboardingWizard  (reanudable, barra de progreso)
     │
     ├─ 1 Datos ......... nombre/DNI/tel/dirección + ☑ RGPD (obligatorio)
     ├─ 2 Hoja .......... firma eIDAS 25.1  ──►  CONSUME invitación
     ├─ 3 Cuestionario .. (post-firma, /api/onboarding/{token}/cases/{id})
     └─ 4 Documentación . (post-firma, subida simple 25 MB)
```

Fig. 12.x — Asistente de alta del cliente: la firma del paso 2 consume la invitación.

El orden de los pasos no es casual. Los datos identificativos y el consentimiento RGPD van primero porque son requisito para todo lo demás; la hoja de encargo va antes que el cuestionario y la documentación porque establece la relación profesional y, al firmarse, cierra la invitación y crea la relación de confianza bajo la que se recogerán después los datos sensibles del caso. El checkbox RGPD del paso 1 es *obligatorio*: sin marcarlo no se avanza, y su marca queda registrada como base del tratamiento.

> ℹ **Firma electrónica simple vs. avanzada.** La hoja de encargo del onboarding usa firma electrónica *simple* del art. 25.1 eIDAS (aceptación con registro de IP, navegador y hash del documento). La firma de solicitudes vinculadas al caso desde el portal usa el flujo avanzado con hoja de evidencia (cap. 11 §11.8).

### 12.2.2 Anatomía del token y defensa contra abuso

El token nunca se almacena en claro: la fila de `client_invitations` guarda solo su `token_hash` (`sha256`), de modo que ni un volcado de la base de datos revela enlaces vivos. La máquina de estados del token es estricta y monótona: una vez `consumed`, `revoked` o `expired` no vuelve a `pending`. El contador `attempts` se incrementa en cada intento fallido de validación y, superado un umbral, invalida el token para frenar el rastreo por fuerza bruta del espacio de tokens. El TTL de 7 días acota la ventana de riesgo: pasado ese plazo la invitación caduca aunque nadie la haya usado, y el abogado debe regenerarla desde el portal (cap. 11 §11.13.1).

> ℹ **Reanudable, pero de un solo uso.** El asistente es reanudable mientras la invitación siga `pending`: el cliente puede cerrar y volver, y retoma en el paso donde estaba. La invitación se consume en el paso de firma (paso 2); a partir de ahí el cliente ya tiene sesión y continúa por rutas autenticadas, no por el token de invitación.

## 12.3 «Mis casos»

El dashboard `/cliente/casos` muestra los expedientes del cliente, siempre con doble scoping: por `clientUserId` y con `deletedAt` nulo (un caso con soft-delete no existe para el cliente).

- Con **un solo caso**, redirige directamente al detalle.
- Con **dos o más**, muestra un dashboard con KPIs reales: expedientes activos, cuestionarios pendientes, pasos pendientes, próximas citas.

### 12.3.1 Detalle del caso y visibilidad decidida por el letrado

El detalle (`CaseDashboard`, datos de `GET /api/my/cases/[id]`) es la contraparte del panel del abogado, pero **solo muestra lo que el letrado decidió compartir**: sus pestañas se derivan de `data.visibility` (los interruptores de `ClientVisibilityPanel`, cap. 11 §11.13.2). Las pestañas posibles son: `analisis`, `cuestionario`, `documentos`, `testigos`, `preparacion-vista`, `peritaje`, `mensajes`, `activar`.

| Componente | Función |
|---|---|
| `PreliminaryAnalysisCard` | Análisis preliminar; polling cada 15s; *locked* si falta el pago |
| `QuestionnaireForm` | Cuestionario del caso con autoguardado |
| `ClientWitnessesSection` | Testigos que aporta el cliente |
| `TasksList` | Próximos pasos / tareas del cliente |
| `MilestonesTimeline` | Cronología y fases del procedimiento |

| Componente | Función |
|---|---|
| `DocumentsSection` | Documentos compartidos y aportados |
| `MessagesSection` | Mensajería con el letrado |
| *Aside*: `LawyerCard`, `HojaEncargoCard`, `MeetingCard` | Letrado asignado, estado de la hoja de encargo, próxima reunión |

El `PreliminaryAnalysisCard` merece detalle: presenta al cliente el análisis preliminar de su caso y hace polling cada 15 segundos mientras el análisis se está generando en segundo plano, de modo que la tarjeta transita sola de «analizando» a «listo» sin recarga. Cuando el modelo de negocio exige pago (Modelo A) y el setup fee aún no se ha abonado, la tarjeta aparece *locked*: el cliente ve que existe un análisis pero su contenido queda tras el muro del pago, un gancho comercial honesto que no oculta la existencia del trabajo pero sí su detalle hasta la activación. La `MilestonesTimeline` traduce las fases del procedimiento a una cronología legible para un no-jurista, y el `MeetingCard` del aside integra la próxima cita (con enlace de videollamada cuando aplica). El `HojaEncargoCard` refleja en todo momento el estado de la relación contractual (`pendiente` / `firmada`).

### 12.3.2 La visibilidad como contrato de datos

El objeto `data.visibility` que devuelve `GET /api/my/cases/[id]` no es solo una guía de renderizado: es el *contrato de datos* del endpoint. El servidor recorta la respuesta a lo que el letrado ha marcado visible antes de enviarla, de forma que las secciones ocultas no viajan al navegador siquiera como payload latente. Esto es esencial en un expediente penal: el análisis de IA, las Claves del Caso o la preparación de la vista son el criterio interno del abogado y nacen ocultos (cap. 11 §11.13.2); si el cliente inspeccionara la respuesta de red, no encontraría esos datos porque el backend nunca los incluye mientras el interruptor esté apagado.

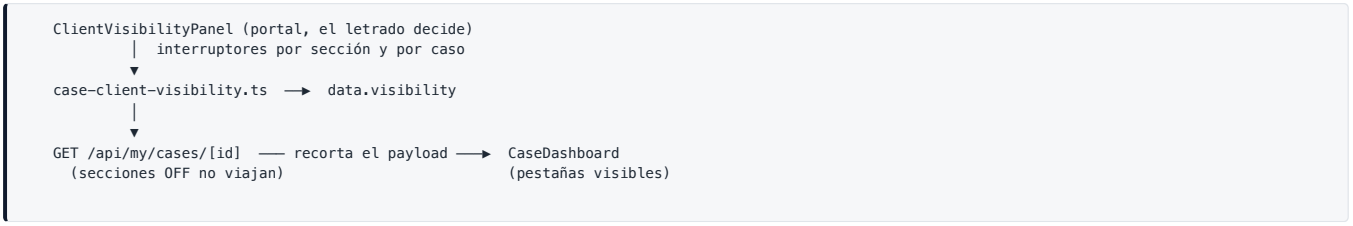


Fig. 12.2 — La decisión del letrado se propaga como recorte del payload, no solo como ocultación en la UI.

## 12.4 Documentos

`DocumentsSection` alinea su validación de tipos con el backend (`packages/storage/limits.ts`: extensiones `pdf`, `docx`, `jpg`, `png`, `zip`, `xlsx`…). La subida es multipart hacia R2 con hasta 500 MB (y 25 MB en la subida simple del onboarding). El `upload-init` valida la extensión y la coincidencia MIME, sanea el nombre y sube al bucket de cuarentena con URLs presignadas de vida corta (TTL 3600s).

> ⚠ **Errores accionables.** Las subidas fallan con mensajes que dicen qué hacer: `file_too_large` (supera el tope), `mime_mismatch` («el contenido no coincide con la extensión; vuelve a exportarlo como PDF»)… No son códigos crípticos: guían al cliente a resolverlo.

Los documentos se analizan con IA en vivo, con feedback visible («Leyendo X/Y págs», y ante fallo «⚠ No se pudo procesar» con `role="alert"`). La sección separa claramente «Compartidos por el despacho» de «Lo que has subido tú».

La subida a cuarentena antes que al bucket definitivo es una salvaguarda deliberada: el fichero del cliente aterriza primero en un bucket de *quarantine*, se valida (extensión, coincidencia MIME por sniffing de contenido en servidor, tamaño) y solo entonces se promueve. Un fichero cuyo contenido real no case con su extensión (`mime_mismatch`) nunca llega a considerarse un documento válido del expediente. Las URLs presignadas de 3600s de vida acotan la ventana en que el navegador puede escribir en R2; caducada, hay que reiniciar la subida. Este mismo patrón de tres fases y validación en servidor es el que usa el portal del abogado (cap. 11 §11.3.3), compartido a través de `packages/storage`.

| Error de subida | Causa | Mensaje accionable |
|---|---|---|
| `file_too_large` | Supera el tope de bytes | Indica el tamaño máximo admitido |
| `mime_mismatch` | Contenido ≠ extensión declarada | «Vuelve a exportarlo como PDF» |
| `unsupported_mime_type` | Extensión no admitida | Lista las extensiones válidas |

## 12.5 Mensajes

`MessagesSection` es la mensajería bidireccional con el letrado. Hace polling cada 30 segundos. La edición y el borrado están limitados a una ventana de 5 minutos, verificada tanto en cliente como en servidor:

| Operación | Regla |
|---|---|
| Editar (`PATCH`) | 1–5000 caracteres, rate 20/h, ventana 5 min; marca `editedAt` |
| Borrar (`DELETE`) | Soft-delete, solo mensajes propios, ventana 5 min |

Ambas operaciones registran eventos en `caseEvents`, dejando trazabilidad de qué se editó o borró. La doble comprobación de la ventana de 5 minutos —en cliente para ocultar los botones y en servidor para rechazar la mutación— es imprescindible: la validación de cliente es solo cortesía de UX, la del servidor es la

que hace cumplir la regla. El borrado es *soft-delete* (marca, no purga física) y solo alcanza a los mensajes propios del cliente; nunca puede tocar los del letrado. El límite de 1–5000 caracteres y el rate de 20 ediciones/hora acotan tanto el abuso como los errores accidentales de pegado masivo.

> ℹ **Polling, no websockets.** La mensajería y las tarjetas de análisis usan *polling* (30s los mensajes, 15s el análisis preliminar) en lugar de conexiones persistentes. Es una decisión de infraestructura: los procesos Next del cliente corren tras el Apache de Plesk, donde una conexión larga chocaría con el `ProxyTimeout`; el polling es robusto frente a ese entorno y suficiente para la cadencia de una conversación abogado-cliente.

## 12.6 Tareas, aclaraciones y cuestionario

- **Tareas**: `TasksList` lista los próximos pasos; el cliente las completa con `POST tasks/[tid]/complete` (solo si es el propietario de la tarea).
- **Aclaraciones**: `POST clarifications` permite al cliente matizar el análisis por secciones (`debilidad`, `fortaleza`, `recomendacion`, `estrategia`, `diagnostico`, `cuestionario`, `documento`, `pronostico`, hasta 20000 caracteres). Estas aclaraciones se incorporan al re-análisis del caso (cap. 11 §11.3.5, `ClarificationsPanel`).
- **Cuestionario**: `questionnaire/answers` autoguarda las respuestas; `questionnaire/submit` las envía.

Las aclaraciones son el canal por el que el cliente corrige o matiza el análisis de la IA sin tener que escribir un mensaje suelto: al vincularse a una sección concreta (una debilidad que la IA señaló pero que el cliente puede explicar, un dato del diagnóstico que matizar), entran de forma estructurada en el siguiente re-análisis del caso y aparecen en el `ClarificationsPanel` del lado del abogado (cap. 11 §11.3.5). El límite de 20000 caracteres permite explicaciones extensas —un relato completo de contexto— sin degradar el rendimiento del re-análisis. El autoguardado del cuestionario protege al cliente de perder respuestas largas por un cierre accidental de pestaña; el `submit` explícito marca el momento en que el letrado puede considerar completo ese bloque de evidencia.

> ✓ **El cliente participa en la calidad del análisis.** Aclaraciones y cuestionario no son formularios muertos: alimentan el bucle de re-análisis del expediente. La regla de completitud de evidencia (cap. 11 §11.3.5) garantiza que ninguna respuesta del cliente quede sin considerar antes de dar por bueno un informe o un escrito.

## 12.7 Pagos: setup fee con doble consentimiento

En el Modelo A, el cliente directo abona una tarifa de activación (setup fee) de 500€. La `FianzaActivationCard` y el endpoint `POST fianza` gobiernan el cobro, que exige **dos consentimientos independientes**:

| Consentimiento | Qué acepta |
|---|---|
| `acceptedConditions` | Las condiciones del servicio |
| `immediateExecutionWaiver` | Renuncia al desistimiento por ejecución inmediata (art. 103 TRLGDCU) |

Ambos se registran en `caseFianzaConsents` con IP y `TERMS_VERSION`. El cobro crea una Stripe Checkout Session (con `automatic_tax`, presentada como «500€ + impuestos»), con protección anti-doble-cargo; el pago se registra en `casePayments` (`kind` = `setup_fee`) y el estado pasa a `setupFeeStatus` = `awaiting_payment`.

El versionado del consentimiento (`TERMS_VERSION`) es lo que hace defendible el cobro: no basta con que el cliente pulse «acepto», hay que poder acreditar *qué* texto aceptó y *cuándo* (IP + versión). La renuncia al desistimiento (art. 103 TRLGDCU) es un consentimiento separado precisamente porque la ley lo exige de forma expresa e informada para servicios que empiezan a ejecutarse antes de que expire el plazo de desistimiento; agruparlo con las condiciones generales lo viciaría. El `automatic_tax` de Stripe calcula el impuesto según la ubicación del cliente, de ahí la fórmula «500€ + impuestos» en lugar de un total fijo.

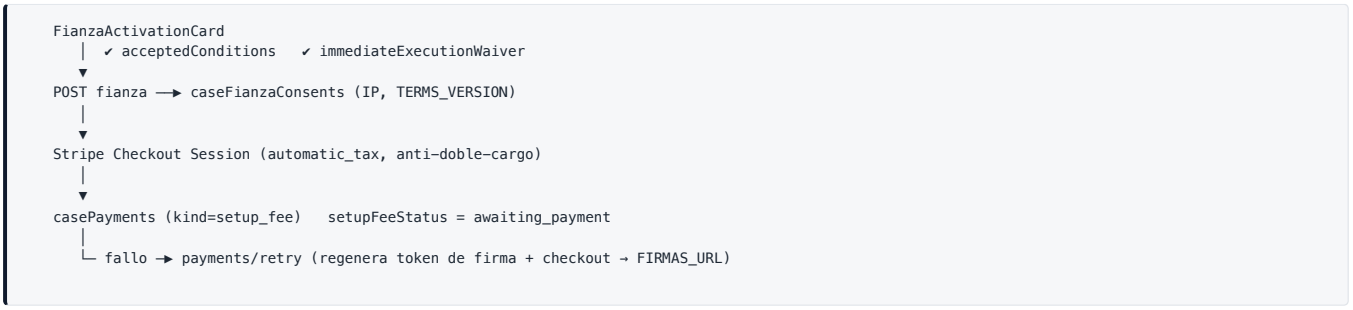


Fig. 12.3 — Flujo del setup fee con doble consentimiento y reintento.

### 12.7.1 Diferencia A vs. B en el pago

El Modelo A crea un Stripe customer con tarjeta guardada off-session, necesaria para cobrar el consumo de IA al propio cliente. El Modelo B no crea tarjeta del cliente: el consumo de IA lo paga el despacho (cap. 11 §11.10.3).

### 12.7.2 Reintento y banners

Si el pago falla, `payments/retry` regenera el token de firma y el checkout hacia `FIRMAS_URL`. La UI muestra un solo aviso urgente a la vez con `PaymentBanner`, y prioriza la firma sobre el pago: el `SignatureBanner`, alimentado por `pendingSignatureRequests`, enlaza a `FIRMAS_URL` antes de reclamar el pago.

ℹ **Un solo banner urgente.** Para no abrumar al cliente, la pantalla muestra como mucho un aviso crítico. El orden de prioridad es firma > pago: primero se pide firmar, y solo después de firmado se reclama el cobro.

## 12.8 Tours guiados

El cliente se orienta con tours anclados a la interfaz. `clienteTour.ts` (`CLIENTE_TOURS`, v4) y el componente `OnboardingTour` se anclan a selectores `[data-onboarding]` y a anclas internas (`#cuestionario`, `#documentos`, `#mensajes`). El tour salta la primera vez que se visita cada pantalla, y siempre queda disponible un botón «Ver tutorial». Las cuentas demo tienen su propio `DemoTour`.

El anclaje por selectores estables (`[data-onboarding]`) desacopla los tours del marcado concreto: los pasos apuntan a atributos de datos, no a clases de estilo que cambian con cada rediseño, de modo que reorganizar la UI no rompe silenciosamente el recorrido. El versionado (`v4`) permite que, cuando el producto cambia lo suficiente como para justificar volver a enseñar una pantalla, el tour se vuelva a mostrar una vez a quien ya lo había visto en una versión anterior, sin repetirlo en cada visita. La disponibilidad permanente del botón «Ver tutorial» respeta al usuario que descartó el tour la primera vez pero lo necesita más tarde, sin obligarle a buscar en ajustes.

ℹ **Un producto para no-expertos.** El público del área del cliente no es técnico ni necesariamente jurídico. El coste de una pantalla mal entendida no es solo fricción: puede ser un cuestionario sin responder o un documento clave sin aportar, con impacto real en el procedimiento penal. Por eso el acompañamiento (tours, errores accionables, priorización de un único aviso) es una decisión de producto de primer orden, no un adorno.

### 12.8.1 La API `/api/my`: la superficie del cliente

Toda la funcionalidad del cliente cuelga del prefijo `/api/my` (en `apps/cliente/src/app/api`), separado por diseño del `/api` del portal: son dos aplicaciones distintas con dominios distintos, y el cliente nunca alcanza un endpoint del profesional. Las rutas comparten las convenciones transversales del monorepo —`withErrorHandler` con mapeo de errores tipados, validación Zod en el borde, rate limiting por clave, subidas de fichero en tres fases— documentadas en el capítulo de API. El gate dominante aquí no es el rol de portal sino la propiedad: cada ruta bajo `my/cases/[id]` comprueba que el caso pertenece al `clientUserId` de la sesión y no está borrado, devolviendo 404 (no 403) ante un id ajeno para no filtrar su existencia.

| Grupo de rutas | Propósito |
|---|---|
| `my/cases/[id]` | Detalle del caso recortado por visibilidad |
| `my/cases/[id]/documents/**` | Subida y descarga mediada de documentos |
| `my/cases/[id]/questionnaire/**` | Autoguardado y envío del cuestionario |
| `my/cases/[id]/messages` · `clarifications` · `tasks/**` | Mensajería, aclaraciones y tareas del cliente |
| `my/cases/[id]/fianza` · `payments/retry` | Setup fee y reintento de pago (Modelo A) |
| `my/dsr/export` · `my/dsr/delete` · `my/dsr/[id]/download` | Autoservicio de derechos del interesado |
| `onboarding/{token}/**` | Rutas del asistente de alta autenticadas por token |

## 12.9 Autoservicio de derechos (DSR)

El cliente ejerce sus derechos del interesado sin escribir correos, desde `/cliente/me/dsr` (`DsrSelfService`). Las solicitudes se registran en `dsr_requests`, con plazo legal de 1 mes y contacto del DPO en `dpo@jbhasesorialegal.com`.

| Derecho | Endpoint | Comportamiento |
|---|---|---|
| Acceso / portabilidad (art. 15 / 20) | `POST me/dsr/export` | Genera un ZIP; se descarga con un token de vida 72h |
| Supresión (art. 17) | `POST me/dsr/delete` | Requiere aprobación admin (`pending_review`); anonimiza pero conserva firmas eIDAS y pagos |

⚠ **La supresión no borra las obligaciones legales.** El derecho de supresión anonimiza los datos personales del cliente, pero conserva lo que la ley obliga a retener: las firmas electrónicas eIDAS (valor probatorio) y los registros de pago (obligaciones fiscales/contables). El borrado total del rastro legal no procede.

El diseño del autoservicio distingue los dos derechos por su naturaleza. El de acceso/portabilidad es *self-service* puro: el cliente lo dispara, el sistema arma un ZIP con sus datos y lo entrega mediante un token de descarga de vida corta (72h) —caducado, la descarga falla con error tipado y hay que regenerarla— sin intervención humana. La supresión, en cambio, no se ejecuta sola: entra en estado `pending_review` y requiere aprobación de un administrador, porque hay que valorar caso por caso qué se anonimiza y qué se retiene por obligación legal. Ambos flujos quedan registrados en `dsr_requests` con su marca temporal, de modo que el despacho puede acreditar el cumplimiento del plazo de un mes que fija el RGPD.

## 12.10 Ausencia de PWA

ℹ **No hay PWA.** La app del cliente no es una Progressive Web App: no existe `manifest` ni service worker. Solo se declara `themeColor` y `robots: noindex` (el área del cliente nunca se indexa). Cualquier referencia a instalación o modo offline sería incorrecta.

La ausencia de service worker es coherente con la naturaleza del producto: un expediente penal exige datos frescos y consistentes con el servidor, no una capa de caché offline que pudiera mostrar un análisis o un mensaje desactualizado. El `themeColor` y el `noindex` son la única concesión a los metadatos de plataforma; todo lo demás se resuelve con renderizado del servidor y polling. Si en el futuro se quisiera notificar al cliente fuera de la sesión, el canal natural sería el correo (como el propio OTP), no una notificación push que requeriría el service worker inexistente.

## 12.11 Modelo A vs. Modelo B

La distinción de modelo de negocio del cliente se determina por `cases.orgId`, y condiciona el pago, la tarjeta y la marca que ve el cliente:

| Aspecto | Modelo A (cliente directo) | Modelo B (cliente de despacho) |
|---|---|---|
| Organización | Bajo `JBH_ORG_ID` | Bajo el despacho (`org_id` del bufete) |
| Coste | 500€/caso + consumo de IA | El consumo de IA lo paga el despacho |
| Tarjeta del cliente | Sí — Stripe customer off-session | No — sin tarjeta del cliente |
| Marca | Neutra JBH (champán) | Marca del despacho + «tecnología de JBH» |

El branding del despacho procede de `organizations.brandingJson` (`logoUrl`, `primaryColor`, `clientVisibleFields`). Con marca del despacho, la interfaz del cliente muestra su logotipo y color, con la coletilla «tecnología de JBH»; sin marca, aparece la identidad neutra champán de JBH.

La consecuencia práctica de que el modelo se determine por `cases.orgId` es que un mismo usuario podría, en teoría, ser cliente directo de JBH en un caso y cliente de un despacho en otro: el modelo se resuelve por expediente, no por persona. Esto explica por qué las decisiones de pago y branding se toman a nivel de caso. En el Modelo B, el cliente nunca ve un cobro de IA: su relación económica es con el despacho, y el consumo de IA es un coste interno del bufete (cap. 11 §11.10.3); por eso no se le pide tarjeta. En el Modelo A, la tarjeta off-session es imprescindible porque JBH cobra el consumo de IA directamente al particular, además del setup fee inicial.

ℹ **«Tecnología de JBH», no «servicio de JBH».** La coletilla que acompaña al branding del despacho es deliberada en su literalidad: el despacho presta el servicio jurídico bajo su marca; JBH aporta la tecnología. Esta distinción —plataforma frente a despacho— es la misma que vigila el linter de frases del CRM (cap. 11 §11.9.5) y sostiene el encaje deontológico del producto.

⚠ **Los datos fiscales del despacho nunca llegan al cliente.** El branding expuesto al cliente se limita a logo, color y campos visibles declarados (`clientVisibleFields`). Los datos fiscales de la organización (CIF, cuentas, facturación) NUNCA se envían al área del cliente.

✓ **El cliente ve exactamente lo que debe ver.** El área del cliente combina tres controles: scoping por `clientUserId` + `deletedAt`, visibilidad por sección decidida por el letrado, y separación de branding que oculta los datos fiscales del despacho. El resultado es un portal donde el particular sigue su caso sin acceder al criterio interno del abogado ni a datos ajenos.

# 13
# Voz, mensajería, email y marketing autónomo

> *JBH Legal atiende y capta clientes por cuatro canales que no requieren intervención humana continua: una recepcionista telefónica de IA (Telnyx AI Assistant), un bot de WhatsApp sobre la Cloud API de Meta, un asistente de email entrante que sondea las bandejas del despacho, y un sistema de correo saliente transaccional (Resend). Sobre ellos opera una fábrica de contenido de marketing que cada mañana genera artículos de blog, publicaciones sociales y briefs de Reel para tres audiencias, siempre pasando por un linter deontológico que prohíbe prometer resultados. Este capítulo documenta la arquitectura, las tablas, las variables de entorno y los gotchas reales de cada canal, además del subsistema de SEO técnico que amplifica el contenido publicado.*

## 13.1 Telefonía: Telnyx AI Assistant

La recepcionista telefónica y el «Abogado IA» de voz están construidos sobre **Telnyx AI Assistant**. Este es el sistema en producción; el proveedor anterior (VAPI) sobrevive únicamente como legado residual (§13.1.7). Telnyx orquesta la llamada (STT, LLM, TTS, telefonía) y delega en JBH tres webhooks HTTP que viven en `apps/web/src/app/api/public/telnyx-ai/`: `init/`, `tool/` y `post-call/`. La lógica de negocio de las herramientas es agnóstica del proveedor y reside en `apps/web/src/lib/voice-assistant-tools.ts`, de modo que el mismo catálogo de tools sirve para Telnyx hoy y serviría para otro orquestador mañana.

La separación entre orquestación (Telnyx) y negocio (JBH) es la decisión arquitectónica clave de este canal. Telnyx se encarga de todo lo que es difícil y específico de la telefonía en tiempo real —transcribir voz a texto con baja latencia, decidir cuándo hablar, sintetizar la respuesta con una voz natural, gestionar interrupciones y silencios— mientras que JBH conserva el control de lo que importa jurídicamente: quién puede consultar qué, qué se persiste y qué no, y qué información sale por el altavoz. Que la migración desde VAPI se resolviera sustituyendo webhooks sin reescribir la lógica de las diez herramientas es la prueba de que esa frontera está bien trazada.


Fig. 13.1 — Los tres webhooks de Telnyx AI Assistant y su materialización en la tabla `leads`.

### 13.1.1 Webhook `init`: variables dinámicas en <1 s

El endpoint `init/route.ts` responde a `assistant.initialization`: Telnyx lo llama al descolgar y espera las *variables dinámicas* con las que personaliza el saludo. Debe responder en menos de un segundo, así que hace una única consulta ligera por el número llamante. Devuelve, entre otras: `caller_registrado` (booleano), `nombre` y `tratamiento` del contacto si consta, un `saludo` con la marca escrita fonéticamente («Jota Be ache» para que el TTS no deletree «JBH»), y `es_horario_laboral` para que el asistente ajuste el guion (ofrecer cita frente a dejar recado). Si el número no está registrado, devuelve los valores neutros y el asistente trata la llamada como de un desconocido.

### 13.1.2 Webhook `tool`: ejecución de herramientas

El endpoint `tool/route.ts` ejecuta la herramienta que el LLM del asistente decide invocar en cada turno. El nombre de la herramienta llega por query string (`?name=…`) y el número del llamante por la cabecera `X-Caller-Number`. Telnyx anida los argumentos de la tool dentro de sobres de payload, de modo que la ruta **desanida hasta cinco niveles** buscando el objeto de argumentos real antes de validarlo. Cada herramienta se ejecuta con un tope de `TOOL_TIMEOUT_MS = 7000` y bajo una política *never-throw*: si algo falla o se agota el tiempo, se devuelve un resultado degradado pero válido para que el asistente pueda continuar la conversación en voz sin cortar la llamada.

> ⚠ **Gotcha: Telnyx envía `null` explícito.** Telnyx rellena los campos de argumentos que el LLM no ha aportado con `null` literal, no los omite. Un schema Zod con `.optional()` admite `undefined` pero **rechaza** `null`, por lo que `safeParse` fallaría en herramientas con parámetros opcionales. La solución en `voice-assistant-tools.ts` es limpiar el objeto antes de validar con `Object.fromEntries(...)` filtrando toda clave cuyo valor sea `null`, `undefined` o la cadena vacía. VAPI omitía esos campos, así que este saneo apareció al migrar a Telnyx.

### 13.1.3 Catálogo de 10 herramientas de voz

El archivo `voice-assistant-tools.ts` expone diez herramientas agnósticas de proveedor. La siguiente tabla las enumera con su cometido y su efecto lateral.

| Herramienta | Cometido | Efecto / dato que devuelve |
|---|---|---|
| `identificar_llamante` | Resolver quién llama a partir del número o de datos aportados | Contacto registrado / desconocido; base para el resto de tools |
| `consultar_estado_caso` | Consultar la logística del expediente | Requiere OTP; devuelve estado/abogado/cita/gestiones, nunca contenido |
| `verificar_codigo_caso` | Validar el OTP de 6 dígitos enviado al email | Habilita la consulta de estado si el código es correcto |
| `registrar_consulta` | Anotar el motivo de una consulta nueva | Crea/actualiza un registro en `leads` |
| `enviar_enlace_portal` | Enviar por email el enlace de acceso al portal | Dispara una plantilla de Resend |
| `proponer_horarios` | Ofrecer huecos disponibles de agenda | Lee disponibilidad (motor de agenda, §13.1.7) |
| `agendar_cita` | Reservar una cita en un hueco propuesto | Crea el evento; confirma por email |
| `cancelar_cita` | Anular una cita existente | Elimina el evento y notifica |
| `dejar_mensaje_abogado` | Recoger un mensaje para el letrado del caso | Persiste el mensaje asociado al lead |
| `recado_humano` | Escalar a una persona / recoger recado general | Registra recado y marca para seguimiento humano |

### 13.1.4 Verificación de firma: Ed25519 con fallback

Todo webhook de Telnyx se autentica en `apps/web/src/lib/telnyx-verify.ts` mediante la función `authorizedTelnyx()`. El método preferente es la firma **Ed25519**: Telnyx firma el cuerpo crudo con su clave y envía la firma en `telnyx-signature-ed25519` junto con `telnyx-timestamp`; JBH verifica con la clave pública `TELNYX_PUBLIC_KEY` y aplica una ventana anti-replay de 10 minutos sobre el timestamp. Si la verificación Ed25519 no es aplicable (configuración de webhook por secreto), cae a un **secreto compartido** `TELNYX_AI_WEBHOOK_SECRET`. Ambos valores son secretos y viven out-of-band (`shared/*.env` del servidor).

### 13.1.5 OTP de expediente: el asistente nunca ve el caso

La consulta de estado por voz está protegida por un OTP para no exponer la logística de un expediente a quien simplemente conoce un número de teléfono. El flujo emite un código de **6 dígitos al email registrado** del contacto, lo almacena hasheado con `sha256` en `verification_tokens`, con validez de 10 minutos y un solo uso. Al validar con `verificar_codigo_caso`, el helper `fetchCaseLogistics` comunica **únicamente** el estado del expediente, el abogado asignado, la próxima cita y las gestiones pendientes.

> ✓ **Barrera dura de confidencialidad.** `fetchCaseLogistics` jamás devuelve el contenido del caso (hechos, documentos, estrategia, análisis de IA). La recepcionista de voz conoce si hay cita el martes, pero no puede leer ni describir el expediente. El OTP al email cierra además la vía de suplantación por *caller ID* falseado.

El diseño del OTP al email —y no un SMS al número que llama— es intencionado en un contexto de datos penales. El identificador de llamada es trivialmente falsificable, así que confiar en el número entrante como prueba de identidad expondría la logística de un expediente a cualquiera que conozca el teléfono de un cliente. El correo registrado, en cambio, es un canal que el atacante no controla con solo marcar un número. El límite de un solo uso y la ventana de 10 minutos acotan además la utilidad de un código interceptado, y el hasheo `sha256` en `verification_tokens` significa que ni siquiera un volcado de esa tabla revela los códigos en claro.

### 13.1.6 Materialización en `leads`, no en «calls»

No existe una tabla `calls`. Toda llamada se materializa en la tabla `leads` mediante el webhook `post-call`, que extrae de forma defensiva `summary`, `transcript`, `duration` y el motivo de colgado. Las columnas relevantes de `leads` son `source`, `phone`, `summary` y `transcript` (ambas `MEDIUMTEXT`), `durationSec`, `endedReason`, `vapiCallId` (nombre heredado del proveedor anterior, hoy contiene el identificador de la llamada Telnyx) y `status`. La operación es idempotente por `vapiCallId`: reintentos de Telnyx sobre el mismo `post-call` hacen UPSERT y no duplican el lead.

La grabación de audio está desactivada; lo que se persiste es la transcripción y el resumen que Telnyx entrega en `post-call`. La infraestructura de voz se compone de dos asistentes configurados en Telnyx —«JBH Recepción» (español) y «JBH Abogado IA»— con voces de **ElevenLabs** (modelo `turbo_v2_5`), STT **Deepgram Flux** y supresión de ruido **krisp**. En la web, el widget de voz usa `@telnyx/ai-agent-widget` parametrizado con `TELNYX_AGENT_ID`.

### 13.1.7 VAPI: legado residual

El proveedor anterior dejó rastros que conviene conocer para no confundirlos con el sistema vivo:

| Resto de VAPI | Estado |
|---|---|
| Documento `VAPI_TELEFONO.md` | Obsoleto. El webhook `/api/public/vapi` que describe ya no existe. |
| Columna `leads.vapi_call_id` | Reutilizada como identificador de llamada Telnyx (nombre heredado, no se renombró). |
| `vapi-agenda.ts` | Vivo y en uso: es el motor de agenda sobre Google Calendar. Nombre heredado; lo consumen `proponer_horarios` / `agendar_cita`. |
| Salientes `voice-reminders` / `lead-followup` | En **simulación**. Se activan con `VOICE_REMINDERS_ENABLED=1` y un número emisor `+34` pendiente de provisionar. |

## 13.2 WhatsApp: Meta Cloud API

El bot de WhatsApp vive en `apps/api` (framework Hono). Expone dos webhooks: el principal `/api/whatsapp` contra la Cloud API de Meta y un alternativo `/api/whatsapp-telnyx` para el caso de que WhatsApp se sirva vía Telnyx.

El webhook principal implementa el protocolo de Meta en dos verbos. El `GET` responde al *handshake* de verificación comparando el `hub.verify_token` con `WHATSAPP_VERIFY_TOKEN` y devolviendo el `hub.challenge`. El `POST` recibe los mensajes: valida la firma HMAC de la cabecera `x-hub-signature-256` con `WHATSAPP_APP_SECRET` sobre el cuerpo crudo, y garantiza idempotencia registrando el `wa_message_id` en la tabla `whatsapp_events` (un mensaje reentregado no se procesa dos veces). El webhook alternativo `/api/whatsapp-telnyx` usa en cambio firma Ed25519, como el resto de webhooks de Telnyx.

La conversación sigue una **máquina de estados de identificación + OTP por email** equivalente a la de la voz: el bot identifica al contacto, y para operaciones sensibles exige el OTP enviado al correo registrado. El razonamiento del asistente usa el modelo `FLAGSHIP_MODEL` con *grounding* estricto sobre el dossier del caso (responde «No consta en el expediente» antes que inventar). Las tablas de soporte son `whatsapp_links` (vínculo número↔contacto), `whatsapp_events` (idempotencia y log de eventos) y `case_chat_threads` / `case_chat_messages` (hilos y mensajes del chat de caso).

> ℹ **Cuenta Meta provisional.** La integración de WhatsApp opera contra una cuenta de Meta provisional. El código está completo (verificación, firma, idempotencia, máquina de estados), pero la cuenta de negocio definitiva es un frente de configuración pendiente.

## 13.3 Email entrante: el worker asistente

El correo entrante lo atiende un proceso pm2 dedicado, `jbhlegal-email-worker`, cuyo código es `apps/api/src/email-worker.ts`. Sondea por IMAP (librerías `imapflow` para el protocolo y `mailparser` para el parseo de MIME) las bandejas de `soporte@`, `info@` y `despacho@`. La conexión usa `IMAP_HOST = mail.jbhasesorialegal.com` en el puerto 993 (IMAPS) y sondea con periodicidad `EMAIL_POLL_MS = 60000` (un minuto). El worker **solo arranca su lógica si** `EMAIL_ASSISTANT_ENABLED=1`; con la variable ausente o distinta, el proceso queda inerte.


Fig. 13.2 — Flujo del worker de email entrante: sondeo IMAP, clasificación y auto-respuesta.

Por cada mensaje nuevo, el worker clasifica el **departamento** destino: soporte técnico, información comercial o despacho. Para el correo dirigido al despacho invoca `runAssistantTurn`, el mismo turno de asistente jurídico que alimenta otros canales. Después auto-responde al remitente con un disclaimer explícito de que la contestación la ha generado una IA, no un abogado.

El *gating* por `EMAIL_ASSISTANT_ENABLED` merece una nota operativa: un worker de email que auto-responde es un sistema con potencial de bucle (dos autorespondedores conversando entre sí) y de fuga de tono inadecuado si una clasificación falla. Mantenerlo tras un interruptor explícito permite desplegar el código en producción con la funcionalidad apagada, activarla de forma controlada y desactivarla al instante si algo va mal, sin necesidad de un nuevo despliegue. El sondeo cada 60 segundos es un compromiso entre responsividad percibida (un cliente que escribe no espera respuesta en el mismo segundo, pero sí en pocos minutos) y presión sobre el servidor IMAP.

> ⚠ **El disclaimer de IA no es opcional.** Toda respuesta automática del worker de email —igual que la del bot de WhatsApp o la recepcionista de voz— debe dejar claro al destinatario que la ha generado una inteligencia artificial y no un letrado colegiado. En un despacho, confundir al ciudadano sobre si está recibiendo asesoramiento jurídico de un profesional tiene implicaciones deontológicas; el disclaimer preserva la frontera entre información asistida por IA y asesoramiento legal.

### 13.3.1 Tickets in-portal: un sistema independiente

No debe confundirse el worker de email con el sistema de tickets del portal, que es un canal distinto y no depende del worker. Los tickets viven en las tablas `support_tickets` y `support_ticket_messages` y conectan a cliente o abogado con el superadmin. Cada ticket tiene una categoría (`tecnico`, `facturacion`, `consulta`, `otro`) y recorre los estados `abierto` → `en_curso` → `resuelto` → `cerrado`.

## 13.4 Email saliente: Resend

El correo saliente transaccional lo emite el paquete `packages/email`, un wrapper HTTP directo sobre la API de **Resend** (archivo `resend.ts`). Autentica con `RESEND_API_KEY` y remite por defecto desde `DEFAULT_FROM = noreply@jbhasesorialegal.com`. En desarrollo, `MOCK_EMAIL=1` escribe los correos a disco en lugar de enviarlos, lo que permite probar plantillas sin gastar cuota ni molestar a destinatarios reales. Existe además una vía SMTP alternativa (`smtp.ts`).

El catálogo supera las **40 plantillas transaccionales**, cada una un módulo en `packages/email/src/templates/`. La tabla agrupa las familias principales.

| Familia | Plantillas (ejemplos) | Disparador |
|---|---|---|
| Acceso | `magic-link`, `client-otp`, `member-invite`, `lawyer-invite` | Login sin contraseña, OTP, invitación de miembro/letrado |
| Caso / IA | `questionnaire-ready`, `deadline-reminder` | Cuestionario listo, recordatorio de plazo procesal |
| Agenda | `booking-*` (confirmación, recordatorio, cancelación) | Reserva/cancelación de cita |
| Pagos | `payment-*` | Cargos, recibos e incidencias de facturación |
| Firma | `signature-*` | Solicitud, recordatorio y confirmación de firma eIDAS |
| CRM | `crm-*` | Presupuestos, seguimiento de leads, automatizaciones |
| Colaboradores | `collaborator-*` | Onboarding y comunicaciones del programa de referidos |
| RGPD | `dsr-*` | Acuse y entrega de derechos del interesado (acceso, supresión…) |
| Notificaciones | `notification-digest` | Resumen periódico de novedades del expediente |

## 13.5 Marketing autónomo: la fábrica diaria de contenido

El sistema de marketing autónomo es un proceso pm2 (`jbhlegal-marketing-worker`) que corre una sola vez al día por cron (`0 8 * * *` en UTC, `autorestart false`). Es un *one-shot idempotente por día UTC*: si algo lo relanza el mismo día, detecta que ya trabajó y no duplica contenido. El entrypoint es `apps/portal/scripts/marketing-worker-entry.ts` y la lógica `apps/portal/src/lib/marketing-worker.ts`.

### 13.5.1 Flujo diario

El worker recorre las tres audiencias objetivo —**cliente**, **abogado** y **perito**— en *round-robin*, de manera que cada día rota el foco. Para la audiencia del día:

1. **Claim-the-day**: antes de gastar un solo token de LLM, reclama el día en la base de datos. Si ya está reclamado, aborta. Esto evita trabajo duplicado y coste doble ante reejecuciones.
2. **Tema del día**: genera un tema, deduplicado contra los 40 temas más recientes para no repetirse.
3. **Artículo de blog**: redacta el artículo. Si pasa el linter deontológico, se **auto-publica** con `status = published` en la ruta `/insights/<slug>`.
4. **Un post por canal social habilitado**: genera una pieza adaptada por cada red activada en los ajustes.
5. **Brief de Reel** (solo si Instagram está habilitado): produce un guion de Reel y lo encola con `status = needs_recording`. **Nunca** auto-publica un Reel: requiere grabación humana.

El resultado son unas **~15 piezas al día**. Cada pieza pasa por `marketingSafetyLint` antes de guardarse, y el coste de IA se imputa en `ai_request_log` con `scope = marketing`, de modo que el gasto de contenido es visible y separable del gasto de casos.

El patrón *claim-the-day-antes-del-LLM* es la salvaguarda económica del sistema. Reclamar el día en la base de datos es una operación barata; generar quince piezas con modelos de lenguaje no lo es. Al reclamar primero, una segunda ejecución accidental (un reinicio de pm2, un cron duplicado, un despliegue que relance el proceso) se detiene antes de gastar un solo token, en lugar de descubrir a posteriori que se ha pagado dos veces por el contenido de un día. La idempotencia por día UTC convierte al worker en seguro de reejecutar, propiedad valiosa en un proceso automático que corre desatendido.

La rotación round-robin entre audiencias reparte el foco a lo largo de la semana en lugar de saturar un solo perfil. Un despacho penal capta por tres vías distintas —el ciudadano que busca defensa (cliente), el abogado que podría revender la plataforma (abogado) y el perito que colabora en causas (perito)—, y cada una responde a un mensaje y un tono diferentes. Generar cada día para una sola audiencia produce, a lo largo del tiempo, un cuerpo de contenido equilibrado sin que el sistema tenga que producir quince piezas por audiencia y día, lo que sería inasumible en coste y en ruido para los seguidores.

> ℹ **Watchdog independiente.** Un segundo cron (`0 13 * * *`) actúa de vigía: si el worker de las 08:00 no dejó rastro de haber corrido, envía una alerta por email. El entrypoint es `apps/portal/scripts/marketing-watchdog-entry.ts`. Así, un fallo silencioso del cron no pasa un día entero desapercibido.

### 13.5.2 Modelo de datos

| Tabla | Cardinalidad | Contenido |
|---|---|---|
| `marketing_settings` | Singleton | Canales habilitados, audiencias, parámetros globales del sistema |
| `marketing_topics` | Histórico | Temas generados; base del dedup contra los 40 últimos |
| `content_pieces` | Una fila por pieza | Canal, cuerpo, estado; el blog vivo se lee de aquí |

El campo `channel` de `content_pieces` toma valores `blog`, `linkedin`, `instagram`, `facebook`, `google_business`, `twitter` y `reel`. El campo `status` recorre `draft`, `needs_review`, `published`, `failed`, `rejected` y `needs_recording` (este último exclusivo de los Reels). El blog público de `/insights` se sirve leyendo las filas con `channel = blog` y `status = published`.


Fig. 13.3 — Ciclo diario del marketing autónomo y su watchdog.

### 13.5.3 Conectores de redes sociales

Cada red tiene un *publisher* en `apps/portal/src/lib/marketing/channels/`. El registro `ALL_CHANNELS` enumera los seis canales sociales y `getPublisher(channel)` devuelve el conector o `null`. Cada conector se activa solo si tiene sus credenciales.

| Canal | Variables de entorno | Nota |
|---|---|---|
| Facebook | `META_ACCESS_TOKEN`, `META_FB_PAGE_ID` | Publicación en la página |
| Instagram | `META_IG_USER_ID` | Requiere imagen alojada en R2 |
| LinkedIn | `LINKEDIN_ACCESS_TOKEN`, `LINKEDIN_ORG_URN` | Publicación como organización |
| Google Business | `GBP_*` | Publicaciones del perfil de empresa |
| X / Twitter | `X_API_*` | OAuth 1.0a |

Todos los valores anteriores son secretos y viven out-of-band. Un canal sin sus variables configuradas simplemente no se publica; el worker no falla por ello.

### 13.5.4 El linter deontológico: `marketing-banned.ts`

La pieza de cumplimiento crítica es `packages/ai/src/marketing-banned.ts`, cuya función `lintBannedPhrases` examina todo texto de marketing con coincidencia insensible a mayúsculas y a acentos. Prohíbe cualquier promesa de resultado, porque la deontología de la abogacía impide garantizar el éxito de un asunto. Entre las frases vetadas: `garantizamos` (y variantes `te/le/os garantiz…`), `resultado garantizado`, `éxito garantizado`, `absolución garantizada`, `absolución asegurada`, `100%`, `mejor abogado` y `sin riesgo`.

> ⚠ **«Garantías» no es «garantizar».** El linter distingue el verbo de compromiso («garantizamos un resultado», prohibido) del sustantivo legítimo («garantías procesales», «garantías constitucionales», permitido). Por eso veta `garantizamos` pero no `garantías` a secas: un contenido que hable de las garantías del investigado en el proceso penal debe poder publicarse. Las frases y el copy aprobados se registran en `docs/marketing/frases-aprobadas.md`.

## 13.6 SEO técnico en `apps/web`

El contenido que produce el marketing solo rinde si los buscadores —y los LLM— pueden descubrirlo. La web pública (`apps/web`) implementa el andamiaje de SEO técnico que lo amplifica.

### 13.6.1 Sitemap dinámico

El archivo `apps/web/src/app/sitemap.ts` genera el sitemap combinando fuentes estáticas y dinámicas: las páginas fijas, la lista de servicios, las landings, las páginas de SEO local por ciudad, la **matriz delito × ciudad** (una URL por combinación de tipo penal y localidad), las entradas de la enciclopedia penal con su `lastModified` real, y las entradas de blog leídas dinámicamente de las `content_pieces` publicadas.

### 13.6.2 `robots.ts` y bienvenida a los crawlers de IA

`apps/web/src/app/robots.ts` permite `/` y deniega `/api` y `/agendar`. Además, mantiene una lista `AI_CRAWLERS` con los rastreadores de IA a los que se da la **bienvenida explícita** —`GPTBot`, `ClaudeBot`, `PerplexityBot`, entre otros— porque aparecer en las respuestas de los asistentes de IA es un objetivo de captación deliberado (AEO/GEO). Se complementa con los ficheros `llms.txt` y `llms-full.txt`, que describen el sitio en el formato pensado para consumo por modelos de lenguaje.

### 13.6.3 La enciclopedia penal: goteo programado

La enciclopedia penal (`apps/web/src/lib/enciclopedia.ts`) es un depósito de entradas que se publican por **goteo programado**, sin necesidad de un despliegue por cada tanda. El mecanismo combina tres piezas: un campo `publicarDesde` por entrada (fecha a partir de la cual es pública), ISR con `revalidate = 3600` (una hora) y `dynamicParams`. Así, un backlog de entradas ya escritas y fechadas se libera solo al llegar su fecha; una entrada cuya fecha aún no ha llegado devuelve **404** hasta ese momento. El helper `getBacklogInfo` actúa de vigía del backlog pendiente.

✓ **Publicación sin fricción.** El goteo de la enciclopedia y la auto-publicación del blog significan que el sitio crece en contenido indexable a diario sin intervención de ingeniería: ni un `git push` ni un redeploy son necesarios para que aparezcan nuevas URLs. El sitemap las recoge (`lastModified` real de la enciclopedia, blog leído de `content_pieces`) y los buscadores y crawlers de IA las descubren.

El goteo por `publicarDesde` resuelve un problema práctico de un despacho que no tiene un equipo editorial a jornada completa: permite escribir un lote grande de entradas de enciclopedia de una vez y programar su aparición escalonada, de forma que el sitio proyecte una cadencia de publicación constante —señal positiva para los buscadores— sin exigir un despliegue por cada entrada ni un redactor que pulse «publicar» cada mañana. Que las entradas futuras devuelvan 404 hasta su fecha, y no un borrador visible, evita indexar contenido a medio cocer o filtrar antes de tiempo material planificado.

### 13.6.4 Datos estructurados

Múltiples páginas de `apps/web` inyectan JSON-LD (datos estructurados de schema.org) para que los buscadores comprendan la naturaleza del contenido —servicios jurídicos, artículos, negocio local, FAQ— y lo presenten con resultados enriquecidos. Junto al sitemap, el `robots.ts` permisivo con IA, los ficheros `llms.*` y la enciclopedia por goteo, el JSON-LD cierra un andamiaje de SEO orientado tanto a los buscadores clásicos como a los asistentes de IA.

La estrategia de fondo es doble: **SEO clásico** (aparecer en los resultados de Google mediante sitemap, datos estructurados, contenido fresco y SEO local por ciudad) y **AEO/GEO** —optimización para motores de respuesta y para IA generativa— mediante la bienvenida explícita a los crawlers de IA y los ficheros `llms.txt` / `llms-full.txt`. La segunda vertiente reconoce un cambio de comportamiento del usuario: cada vez más personas plantean su problema jurídico a un asistente de IA antes que a un buscador, y aparecer citado en esas respuestas es una fuente de captación tan real como el primer resultado de Google. La matriz delito × ciudad materializa la *long tail* de esa captación: una URL específica para «delito X en ciudad Y» capta la consulta exacta que un ciudadano preocupado escribe o dicta, sea a Google o a un LLM.

ℹ **Cierre del capítulo.** Los cuatro canales de contacto (voz, WhatsApp, email entrante, email saliente) comparten dos invariantes de diseño: la identidad se verifica siempre por OTP al email antes de tocar información sensible, y ningún canal expone jamás el contenido del expediente —solo su logística. La fábrica de marketing añade una tercera: nada se publica sin pasar el linter deontológico, porque en un despacho penal prometer un resultado no es solo mal marketing, es una infracción.

# 14
# Operaciones, calidad y runbooks

> *Este capítulo cierra la documentación con la cara operativa del sistema: cómo se prueba (una suite de más de mil tests sobre vitest), cómo se vigila en producción (endpoints de salud, Better Stack, Sentry y métricas Prometheus del worker de IA), cómo se respalda y recupera (backup nocturno cifrado a R2 y clave privada de DR fuera del servidor), qué runbooks documentan cada operación delicada, cómo funciona la cola del* `ai-worker` *y por qué nunca se reinicia en vuelo, y qué gotchas operativos e incidencias recurrentes conviene conocer. Termina con una valoración del estado de madurez del producto y su deuda técnica conocida.*

## 14.1 Testing

El marco de pruebas es **vitest** de forma unánime: existen 13 archivos `vitest.config` en el monorepo y **cero** de jest. Cada paquete y app corre su propia suite. El comando por paquete es `pnpm --filter @jbhlegal/<x> exec vitest run`.

La filosofía de pruebas del proyecto no persigue una cobertura homogénea sino *densidad donde duele*: las barreras de aislamiento multi-tenant, la lógica de dinero (cupones, pagos, herencias) y el motor de IA concentran la mayoría de los tests, mientras que las capas puramente presentacionales (ui, video) apenas necesitan unos pocos. El paquete `api` figura con cero archivos de test porque su superficie —los webhooks de WhatsApp y de Telnyx— se ejercita indirectamente desde las pruebas de integración de las apps que los consumen y desde la validación de firmas, que sí está cubierta.

### 14.1.1 Cobertura por paquete

El reparto de archivos de test refleja dónde se concentra el riesgo: el portal del abogado (donde vive la lógica multi-tenant y de casos) y el paquete de IA acumulan la mayoría.

| Paquete / app | Archivos de test | Paquete / app | Archivos de test |
|---|---|---|---|
| portal | 124 | signing | 7 |
| packages/ai | 81 | ui | 7 |
| web | 25 | google-calendar | 6 |
| cliente | 17 | storage | 5 |
| ai-worker | 16 | firmas | 5 |
| herencias | 14 | booking | 4 |
| payments | 14 | peritos | 3 |
| email | 13 | logger | 3 |
| db | 9 | colegiado | 2 |
| auth | 8 | video | 1 |
| signatures | 7 | api | 0 |

### 14.1.2 Tests destacados: las barreras que no pueden romperse

En un sistema multi-tenant que custodia datos penales, no todos los tests tienen el mismo peso. Un fallo cosmético en un componente de UI es una molestia; una regresión que permita a una organización leer los casos de otra es una brecha de confidencialidad con consecuencias legales. Por eso el proyecto trata un subconjunto de tests como barreras duras: verifican los invariantes de aislamiento y de control de acceso, y su verde es condición innegociable para desplegar. Estos tests no comprueban una funcionalidad, sino la *ausencia* de una fuga: intentan deliberadamente el acceso cruzado entre organizaciones y exigen que el sistema lo rechace.

Ciertos tests protegen invariantes de seguridad y aislamiento cuya regresión sería crítica en un sistema multi-tenant con datos penales:

| Test | Qué garantiza |
|---|---|
| `case-access-cross-org` | Un usuario de una organización no accede a casos de otra (aislamiento por `org_id`) |
| `causa-access` / `intake-access` | El control de acceso a causas y a intakes respeta el rol y la pertenencia |
| `superadmin` | Los privilegios de superadmin están correctamente acotados |
| `colegiado-gating` | El gating por verificación de colegiado del letrado se aplica |
| `internal-coupons-shared` | La lógica compartida de cupones internos es consistente |
| `uploads` | La validación de subidas de documentos |
| `marketing-banned` | El linter deontológico veta las frases prohibidas (§13.5.4) |

| Test | Qué garantiza |
|---|---|
| `mfa-policy` | La política de MFA y su mecanismo anti-lockout |

#### 14.1.3 Estado verificado y baseline de tests pre-rotos

El estado de las suites principales, verificado en la última pasada:

| Suite | Archivos | Tests | Estado |
|---|---|---|---|
| portal | 123 | 764 | Verde |
| web | 25 | 156 | Verde |
| cliente | 17 | 81 | Verde |
| packages/ai | 81 | — | 7 tests pre-rotos conocidos |

> ⚠ **7 tests pre-rotos en `packages/ai`.** Hay siete tests de `packages/ai` que fallan en runtime por entorno o mocks, no por lógica de producto. **No están marcados `skip`**: fallan de verdad al ejecutar. Antes de atribuir una regresión a un cambio propio, hay que comparar contra el baseline (`git stash` del cambio y reejecución): si esos siete ya fallaban antes, no son culpa del cambio. Es deuda conocida (§14.7), no una regresión nueva.

La existencia de un baseline explícito de fallos conocidos es una decisión de disciplina de ingeniería: en un monorepo grande, un test que falla por un mock desactualizado del entorno contamina la señal y lleva a atribuir a un cambio inocente una rotura que ya existía. Al fijar el baseline —los siete tests de `packages/ai`— el equipo puede seguir mereciendo un veredicto binario («¿introduje una regresión?») pese a que el árbol no esté al 100 % verde. La alternativa (marcarlos `skip`) se descartó a propósito: un test en `skip` se olvida, mientras que uno que falla en rojo mantiene la presión de repararlo.

#### 14.1.4 Typecheck y e2e

El chequeo de tipos es estricto y forma parte del pipeline: `turbo run typecheck` a nivel de monorepo, o `pnpm --filter <x> exec tsc --noEmit` por paquete. TypeScript en modo estricto es la primera línea de defensa contra clases enteras de error (accesos a propiedades inexistentes, `null` no comprobado, incompatibilidades de forma entre módulos), y el hecho de que forme parte del pipeline —no de un chequeo opcional del editor— garantiza que ningún despliegue arrastre un error de tipos. Las pruebas de extremo a extremo usan **Playwright**; el comando `test:e2e` depende del `build` (necesita la app compilada antes de arrancar el navegador), lo que las convierte en pruebas contra la app real servida, no contra componentes aislados.

## 14.2 Monitorización

#### 14.2.1 Endpoints de salud

Las apps con cara pública (portal, cliente, firmas, peritos) exponen dos endpoints de salud. El *shallow* `api/health` responde 200 con `{ok, app, version, uptime}` sin tocar dependencias —ideal para un ping frecuente. El *deep* `api/health/deep` comprueba las dependencias críticas y devuelve **503 si alguna falla**:

| Dependencia | Comprobación |
|---|---|
| Base de datos | `SELECT 1` con timeout de 2 s |
| Resend | La clave tiene forma `re_…` |
| Stripe | Consulta de `balance` con timeout de 2 s |

La distinción entre el chequeo *shallow* y el *deep* es deliberada. Un monitor de uptime que golpea el endpoint deep en cada ping añadiría carga a la base de datos, a Resend y a Stripe cada pocos segundos, y una latencia transitoria de un tercero podría teñir de rojo un servicio que en realidad está sano de cara al usuario. Por eso el ping frecuente usa el shallow (barato, sin dependencias) y el deep se reserva para comprobaciones espaciadas o para diagnóstico manual cuando ya hay sospecha de un problema. Los timeouts de 2 segundos en las comprobaciones de base de datos y Stripe evitan que un tercero lento cuelgue el propio endpoint de salud: si la dependencia no responde en 2 s, se considera caída y se devuelve 503, que es exactamente la señal que el operador necesita.

#### 14.2.2 Better Stack y escalado

La vigilancia externa la lleva **Better Stack** con 8 monitores de uptime más un *heartbeat* del worker de compliance (con una gracia de 26 horas: si no late en ese margen, alerta). El escalado de una alerta va a Telegram, al bot `@JBHLegalOpsBot`, y a email. El patrón de *heartbeat* —a diferencia de un monitor de uptime que sondea desde fuera— sirve para vigilar procesos que no tienen cara HTTP y que deben ejecutarse periódicamente: el worker de compliance «late» al terminar su ejecución diaria, y si Better Stack no recibe ese latido dentro de la ventana de gracia de 26 horas (holgura sobre las 24 h del ciclo diario), asume que el proceso ha dejado de correr y escala. Es la misma idea que el watchdog del marketing (§13.5.1), aplicada a la infraestructura.

#### 14.2.3 Sentry y logs

La captura de errores es **Sentry** (`@sentry/node`), configurado en `sentry.*.config.ts`, `instrumentation.ts` y, en el worker de IA, `sentry.ts`. Se activa con `SENTRY_DSN`; **sin DSN es no-op** y los logs (pino) van a stdout. Los logs por proceso se escriben en `~/logs` y son accesibles con `pm2 logs`. Existe además un monitor de actividad de superadmin para auditar acciones privilegiadas.

Que Sentry sea no-op sin DSN es una propiedad deliberada de degradación segura: el mismo binario corre en desarrollo y en producción, y en desarrollo —donde no se quiere ruido de telemetría hacia un servicio externo— la simple ausencia de `SENTRY_DSN` desactiva el envío sin ramas condicionales dispersas por el código. Los errores siguen registrándose por pino a stdout, capturados por pm2 en `~/logs`, de modo que nunca se pierden; lo único que cambia es si se

agregan además en Sentry para su triaje. El monitor de actividad de superadmin cumple una función distinta y complementaria: no vigila la salud técnica sino la gobernanza, dejando traza auditable de las acciones más privilegiadas del sistema, que en un entorno con datos penales es tan importante como la disponibilidad.

### 14.2.4 Salud y métricas del `ai-worker`

El worker de IA no tiene cara HTTP de app, así que expone su propio servidor de salud (`health-server.ts`) en el puerto **9700**. La ruta `/health` responde 200 `ok` si el último sondeo (`lastPollAt`) fue hace menos de `STALE = 60 s`, y 503 si está *stale* o apagándose. La ruta `/metrics` publica métricas Prometheus (`ai_worker_jobs_done_total`, entre otras) para dashboards.

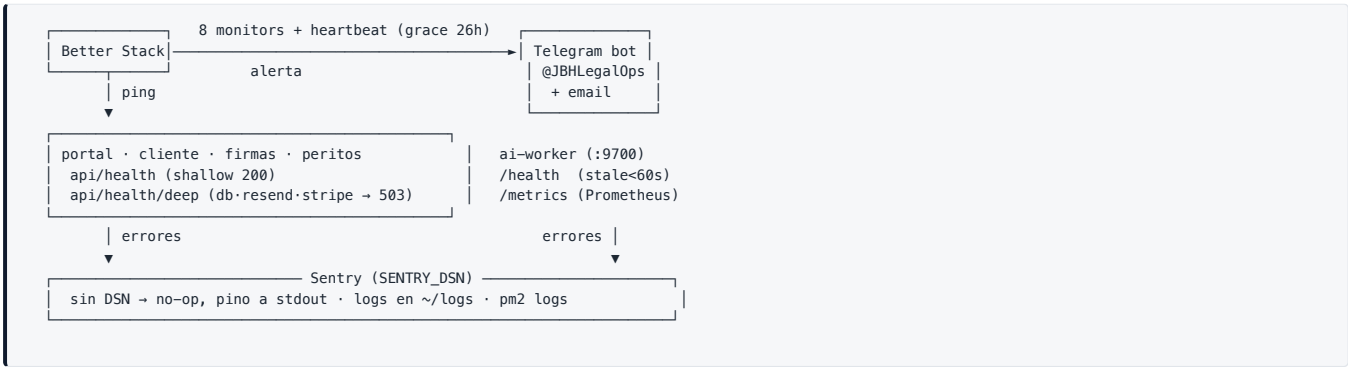


Fig. 14.1 — Malla de observabilidad: salud, uptime externo, escalado y captura de errores.

## 14.3 Backup y recuperación ante desastres

El detalle de retención y cumplimiento RGPD del backup se trata en el capítulo 8; aquí se documenta la operativa.

La tarea Plesk **1038** (cron `30 2 * * *`, 02:30 UTC) ejecuta `jbh-nightly-backup.sh` seguido de `jbh-r2.mjs`, que suben a R2 (bucket `[bucket principal reservado]`) bajo dos prefijos: `db-backups/` (volcado `mysqldump | gzip`) y `secrets-backup/` (los 10 ficheros `shared/*.env` cifrados de forma asimétrica). La retención es de 30 días. Existe también un backup on-demand, verificado.

El resultado es auditable **sin SSH**: el log legible se publica en `/_next/static/[ruta reservada].txt` y debe terminar en `DONE` con `db http=200` y `secrets http=200`; cualquier otra cosa indica un backup fallido. La elección de publicar el log de backup como un fichero estático servido por la web no es casual: el acceso SSH al VPS está restringido por IP (§ capítulo 2), de modo que un operador no siempre puede entrar por consola a comprobar si el backup de anoche terminó. Un `curl` a esa URL desde cualquier sitio responde la última línea del log, y ver `DONE` con dos `http=200` confirma en un vistazo que tanto el volcado de base de datos como el cifrado de secretos subieron correctamente a R2.

### 14.3.1 Secuencia de recuperación

La recuperación ante un desastre completo (pérdida del VPS) sigue la lógica del runbook 12 y se apoya en que las dos piezas irremplazables viven fuera del servidor:

1. **Cloudflare** conserva la configuración de DNS, zonas y el bucket R2 con los backups; se accede con la cuenta (fuera del servidor).
2. Se descargan de R2 el último volcado `db-backups/` y los `secrets-backup/` cifrados.
3. Con la **clave privada de DR** (`jbh-dr-private.pem`, en el gestor de contraseñas / iCloud, nunca en el servidor) se descifran los secretos con `openssl smime`.
4. Se reconstruye el árbol de la aplicación, se restauran los `shared/*.env` descifrados y se importa el volcado de MariaDB.

> ℹ **Asimetría deliberada.** Cifrar con clave pública en el servidor y descifrar solo con la privada fuera de él significa que un atacante que comprometa el VPS puede *escribir* backups pero no *leer* los secretos ya respaldados: no tiene la clave privada. Es la propiedad que convierte el backup de secretos en algo seguro de subir a un almacenamiento de terceros como R2.

> ⚠ **La clave privada de DR nunca está en el servidor.** Los secretos se cifran con la clave *pública* alojada en el servidor; la clave *privada* vive únicamente en `~/Documents/jbh-recovery/jbh-dr-private.pem` (y en el gestor de contraseñas). El descifrado se hace localmente con `openssl smime`. Sin esa clave privada, los secretos respaldados son irrecuperables. Dos cosas quedan deliberadamente fuera del servidor y son imprescindibles para un DR completo: la **cuenta de Cloudflare** y la **clave privada de DR**. El procedimiento completo está en el runbook 12.

## 14.4 Runbooks

Los procedimientos operativos delicados están documentados en `docs/runbooks/`. Cada uno cubre una operación repetible con pasos verificables.

| Runbook | Cubre |
|---|---|
| `01-deploy` | Procedimiento de despliegue |
| `02-credentials` | Gestión de credenciales |
| `04-portal-secrets` | Secretos del portal |

| Runbook | Cubre |
|---|---|
| `05-cliente-secrets` | Secretos de la app de cliente |
| `06-firmas-secrets` | Secretos de la app de firmas |
| `07-stripe-secrets` | Secretos de Stripe |
| `08-colaboradores-onboarding` | Alta del programa de colaboradores |
| `09-paginas-legales` | Publicación de páginas legales |
| `10-observability` | Observabilidad (salud, Sentry, Better Stack) |
| `11-dsr-rgpd` | Derechos del interesado (DSR) RGPD |
| `12-disaster-recovery` | Recuperación ante desastres |
| `12-letrado-ia-cuestionario` | Cuestionario del Letrado IA |
| `12-notifications` | Sistema de notificaciones |
| `13-icloud-build-corruption` | Corrupción de la caché de build en iCloud |
| `13-ssh-ed25519-migration` | Migración de claves SSH a Ed25519 |
| `14-encargados-y-evidencia-rgpd` | Encargados del tratamiento y evidencia RGPD |
| `2026-phase16-booking-rollout` | Despliegue de la fase 16 de reservas |
| `mariadb-backup` | Backup de MariaDB |
| `mariadb-encryption-at-rest` | Cifrado en reposo de MariaDB (TDE) |

Los runbooks siguen una convención de prefijo numérico que agrupa por tema y por fase de madurez del proyecto: la serie 04–07 documenta los secretos de cada app (portal, cliente, firmas, Stripe), la serie 11–14 cubre los procesos de cumplimiento y recuperación (DSR RGPD, disaster recovery, notificaciones, encargados y evidencia), y los runbooks con prefijo de año o de fase (`2026-phase16-booking-rollout`) documentan despliegues concretos. Los dos runbooks de MariaDB sin prefijo numérico —`mariadb-backup` y `mariadb-encryption-at-rest`— son procedimientos de base de datos transversales. El valor de un runbook no es que exista, sino que sus pasos sean verificables: cada operación delicada (rotar un secreto, ejecutar un DSR, recuperar de un desastre) tiene un procedimiento escrito que un segundo operador puede seguir sin conocimiento tácito, lo que reduce el factor bus del sistema.

## 14.5 Operación del `ai-worker`

El worker de IA procesa los análisis pesados a través de una cola en la tabla `document_analysis_jobs`. Sus columnas de control son `status` (`pending` / `processing` / `done` / `failed`), `scope`, `attempts`, `max_attempts`, `priority`, `process_after`, `locked_at` y `locked_by`. Este diseño de cola-en-tabla evita añadir una pieza de infraestructura (Redis, un broker de mensajes) al stack: la propia MariaDB, que ya es la fuente de verdad del sistema, sirve de cola, con la ventaja de que el estado de un job es consultable con SQL corriente y sobrevive a un reinicio del worker sin depender de la durabilidad de un tercer servicio.

### 14.5.1 Claim atómico sin `SKIP LOCKED`

El reclamo de un job (`claim.ts`) es de dos fases: primero un `SELECT` escoge un candidato, luego un *compare-and-swap* atómico `UPDATE ... SET status='processing' WHERE status='pending'`. Solo gana el runner cuyo `UPDATE` devuelve `affectedRows = 1`; los demás reintentan con otro candidato. Este patrón evita depender de `SKIP LOCKED`, lo que mantiene la compatibilidad con MariaDB 10.3.

> ℹ **Barrido de zombies.** `sweepStuckJobs` corre cada 5 minutos y revierte a `pending` los jobs atascados en `processing` más de `STUCK_TIMEOUT_MS = 10 min` (por ejemplo, si un runner murió a media tarea). Reclamar zombies es una operación solo de base de datos: no requiere reiniciar el proceso.

### 14.5.2 Concurrencia y chunking

La concurrencia la fija `WORKER_CONCURRENCY` (por defecto 4, acotado al rango 1–8): un pool de runners que reclaman y procesan jobs en paralelo, con backoff exponencial con jitter en los reintentos. El jitter —una perturbación aleatoria del retardo de reintento— evita el efecto rebaño en el que varios runners que fallaron a la vez reintentan sincronizados y vuelven a saturar el recurso; al desincronizarlos, el sistema se recupera de forma más suave. Cada job corre bajo `runWithOrgAi`, que aplica la clave BYOK de la organización si la tiene (de modo que el consumo de IA de un despacho con clave propia se cobra a su cuenta de Anthropic, no a la de JBH). Los PDFs se transcriben por trozos: `PDF_TRANSCRIBE_PAGES_PER_CHUNK = 5` páginas por chunk, con `PDF_TRANSCRIBE_CONCURRENCY = 4` chunks en paralelo. El producto de `WORKER_CONCURRENCY` por `PDF_TRANSCRIBE_CONCURRENCY` determina el número real de llamadas de visión simultáneas contra Anthropic, por lo que subir ambos a la vez multiplica la presión y puede provocar timeouts en documentos escaneados grandes; ajustarlos es un ejercicio de equilibrio, no de maximización.

> ⚠ **No reiniciar el `ai-worker` en vuelo.** Un `restart` / `SIGINT` /deploy durante un análisis en curso **aborta ese análisis y lo reencola**: se pierde el trabajo (y el coste de IA ya gastado). Antes de tocar el worker, verificar `/health`: si el último resultado es `no-job` y `jobsDone` está estable, no hay nada en vuelo. Un documento grande puede tardar minutos **a 0 % de CPU** porque está esperando la respuesta de red de Anthropic; eso no significa que esté colgado. La regla es mirar el log real y la base de datos antes de reiniciar, nunca reiniciar por impaciencia.

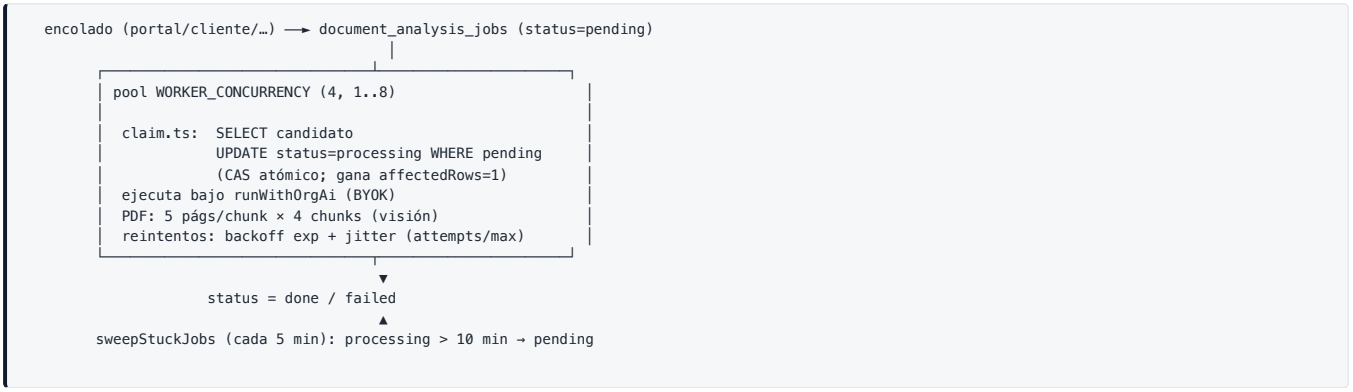


Fig. 14.2 — Cola del `ai-worker`: claim atómico, pool concurrente y barrido de zombies.

#### 14.5.3 Por qué un job lento no es un job colgado

La confusión operativa más cara con el `ai-worker` es interpretar la lentitud como cuelgue. Un documento penal grande —un atestado de decenas de páginas, un auto extenso— puede tardar quince minutos en analizarse, y durante ese tiempo el proceso aparece al 0 % de CPU porque no está calculando: está esperando la respuesta de red de Anthropic, que a su vez está leyendo y razonando sobre un texto largo. Un operador que confunda ese estado con un bloqueo y reinicie el worker no acelera nada; al contrario, aborta el análisis en curso, lo reencola desde cero y hace que el trabajo tarde el doble, además de duplicar el coste de IA de las páginas ya procesadas. La disciplina correcta es diagnosticar antes de actuar: consultar `/health` (si `lastResult` refleja actividad reciente y `jobsDone` no está congelado, el worker vive), leer el log de salida real del proceso y mirar el estado del job en `document_analysis_jobs`. Solo si hay evidencia positiva de bloqueo —no la mera ausencia de progreso visible— se justifica intervenir, y aun entonces reclamar zombies vía `sweepStuckJobs` es preferible a un reinicio, porque no aborta lo que esté en vuelo.

## 14.6 Incidencias y gotchas operativos

La siguiente tabla recoge las clases de incidencia recurrentes y su resolución. Conocerlas ahorra horas de diagnóstico erróneo. Muchas nacen del entorno concreto de despliegue —un VPS Plesk compartido, la caché de build sincronizada por iCloud, Cloudflare delante del origen y un WAF ModSecurity agresivo—, de modo que no son bugs del producto sino fricciones del sustrato sobre el que corre.

| Síntoma / situación | Realidad y acción |
|---|---|
| `ai-worker` «no avanza» | Casi siempre falsa alarma: espera de red a 0 % CPU. No reiniciar en vuelo (§14.5.2); verificar `/health` y la BD primero. |
| Workers cron en estado «stopped» | Normal entre ejecuciones: son one-shot (`autorestart false`). Aparecen «stopped» hasta el siguiente cron. |
| WAF ModSecurity da 404 a un POST | El WAF bloquea POST sin cuerpo. Enviar `body:'{}'` con `content-type: application/json`. |
| JOINs que devuelven vacío o error | Colación: usar `utf8mb4_general_ci`. Mezclar con `unicode_ci` rompe los JOINs por colaciones incompatibles. |
| Crash de webpack `WasmHash` | Caché de build en iCloud (symlinks): no borrarla a pelo (deja symlinks colgando). Recuperar con el procedimiento del runbook 13-icloud. |
| `TypeError reading 'length'` en build | `.next` corrupta: `rm -rf .next` y reconstruir. |
| Cambios que no aparecen tras deploy | Cloudflare cachea `/_next/static` por ruta. Verificar por `BUILD_ID`; usar nombres únicos para el readback. |
| Espacio en disco escaso en el VPS | El disco del VPS es compartido con otros proyectos; vigilar antes de operaciones grandes. |
| El scheduler no reinicia procesos | El deploy real es por SSH; el scheduler es no-op para un `restart` de pm2. Desplegar por SSH. |
| Swap de build en deploy | El swap se hace en `dirname(pm_exec_path)`, no en el cwd del proceso. |

Tres de estas incidencias merecen contexto adicional por su recurrencia. La primera, el **WAF que responde 404 a un POST sin cuerpo**: ModSecurity, configurado a nivel de servidor por el proveedor, interpreta ciertas peticiones sin payload como sospechosas y las bloquea con un 404 que despista, porque parece que la ruta no existe cuando en realidad existe pero fue rechazada; la cura es enviar siempre un cuerpo, aunque sea `{}`, con su `content-type`. La segunda, la **caché de build en iCloud**: el árbol del proyecto se sincroniza por iCloud, que ocasionalmente deja archivos de `node_modules` o `.next` como *placeholders* no descargados, y un build sobre ese estado produce el críptico crash de `WasmHash` de webpack; nunca debe borrarse esa caché a mano, sino recuperarse con el procedimiento del runbook. La tercera, el **caché de Cloudflare sobre** `/_next/static`: Cloudflare cachea los assets estáticos por ruta, así que tras un deploy hay que verificar que el `BUILD_ID` nuevo está sirviéndose realmente —no basta con que el deploy no diera error— consultando un asset con nombre único del build nuevo.

## 14.7 Deuda conocida y decisiones pendientes

Un inventario honesto de lo que queda abierto, con la naturaleza de cada frente (decisión de producto frente a trabajo técnico):

| Frente | Estado | Naturaleza |
|---|---|---|
| MFA obligatorio | `MFA_ENFORCED` apagado; mecanismo anti-lockout listo | Decisión de producto, no bloqueo técnico |
| Cifrado en reposo de MariaDB (TDE) | Pendiente (runbook `mariadb-encryption-at-rest`); requiere plugin + reinicio | Mitigado: R2 ya cifra los backups |

| Frente | Estado | Naturaleza |
|---|---|---|
| Restos de VAPI | Columna `vapi_call_id`, libs `vapi-agenda` / `voice-reminders` | Limpieza cosmética; funcionan bajo nombre heredado |
| 7 tests pre-rotos en `packages/ai` | Fallan por entorno/mocks, no por lógica | Deuda de test; comparar contra baseline |
| Duplicados locales de iCloud | Archivos `… 2.ts` en local; no en git | Higiene local, sin impacto en producción |

Ninguno de estos frentes es un riesgo latente sin dueño. El MFA obligatorio está listo a nivel de mecanismo —incluido el anti-lockout que evita dejar fuera a un usuario legítimo— y solo espera la decisión de producto de activarlo, un juicio de fricción frente a seguridad que corresponde al negocio, no a la ingeniería. El cifrado en reposo de MariaDB (TDE) está mitigado en la práctica: los backups, que son el vector realista de fuga de datos en reposo, ya viajan cifrados a R2, de modo que el TDE cerraría el hueco de un acceso directo al disco del VPS —un escenario de menor probabilidad— a cambio de un plugin y un reinicio que el runbook `mariadb-encryption-at-rest` documenta. Los restos de VAPI y los duplicados locales de iCloud son higiene, no riesgo. Y los siete tests pre-rotos son deuda de test acotada, con baseline conocido, que no enmascara la señal de regresión.

> ✓ **Estado de madurez para producción.** El sistema está **APTO y en producción, operando de forma estable**. Los frentes que la última auditoría marcó como críticos están cerrados: se añadieron los 7 `INSERT` con `orgId` faltantes que amenazaban el aislamiento multi-tenant, las suites principales (portal 764 tests, web 156, cliente 81) están en verde, el pipeline de migraciones es verificable, y todo texto de usuario que entra en un prompt viaja envuelto por `wrapUntrusted`. La deuda restante es explícita, acotada y en su mayoría decisión de producto (MFA) o mitigada por otra capa (TDE frente a cifrado de R2), no riesgo latente sin dueño. Con la observabilidad, el backup nocturno cifrado con clave privada fuera del servidor, y los runbooks que documentan cada operación delicada, JBH Legal reúne las condiciones operativas para gestionar datos penales en producción con las garantías que exige el art. 10 del RGPD.