deploya

DOCUMENTO 2 DE 4 · DOCUMENTO DE DISEÑO

Documento de diseño

Arquitectura defendida decisión por decisión: monolito modular hexagonal, modelo C4, UML por módulo, SOLID y patrones, modelo entidad-relación y diseño de la base de datos.

01 **Requisitos**
Qué hace y para quién

02 **Diseño**
Cómo está hecho y por qué

03 **Manual técnico**
Cómo se instala y opera

04 **Manual de usuario**
Cómo se usa

INSTITUCIÓN	Facultad de Ingeniería Ingeniería en Informática y Sistemas
CURSO	Ingeniería de Software I · Proyecto final 2026
ENTREGA	Presentación final · 13 de noviembre de 2026
REPOSITORIO	`github.com/derekCmorales/deploya`

EQUIPO

Eddy Poroj
Cuentas, seguridad y notificaciones (M1, M10)

Javier Rodríguez
Suscripciones, pagos y administración (M2, M9)

Eduardo Portillo
Proyectos, observabilidad y design system (M3, M7)

Derek Calderón Morales
Motor de despliegue, arquitectura e infraestructura (M4–M6)

VERSIÓN	FECHA	ESTADO	FUENTE
1.1	13 de noviembre de 2026	Presentación final	docs/sitio/doc/02-deploya-diseno.html

CONTENIDO

01 **Arquitectura general** 4
1.1 Resumen de la decisión 4
1.2 Contexto (C4 nivel 1) 5
1.3 Contenedores (C4 nivel 2) 6
1.4 El ciclo de despliegue 6

02 **Arquitectura de cada módulo** 8
2.1 Puertos y adaptadores 9
2.2 Módulos y contratos entre ellos 11

03 **El motor de despliegue** 13
3.1 Componentes del motor (C4 nivel 3) 13
3.2 El pipeline como cadena de pasos 13
3.3 Máquina de estados del despliegue 16
3.4 Detección de stack 18

04 **Diagramas UML** 19
4.1 Clases · Identidad y notificaciones 19
4.2 Clases · Suscripciones y administración 20
4.3 Clases · Proyectos y observabilidad 21
4.4 Secuencia · Recorrido completo 22
4.5 Secuencia · Contratación de un plan 23
4.6 Secuencia · Despliegue 24
4.7 Secuencia · Reversión sin reconstruir 25
4.8 Estados · Suscripción 26
4.9 Actividad · Registro y verificación 27
4.10 Actividad · Contratación y activación 28

05 **SOLID aplicado** 29
5.1 Clean code 29

06 **Patrones de diseño** 30

07 **Modelo de datos** 32
7.1 Modelo entidad-relación 33
7.2 Decisiones de diseño de la base 35
7.3 Diccionario de datos 35
7.4 M1 Identidad (Eddy) 36
7.5 M2 Suscripciones (Javier) 37

7.6 M9 Administración (Javier) 39
7.7 M3 Proyectos (Eduardo) 39
7.8 M4 · M5 · M6 Motor (Derek) 40
7.9 M5 · Recursos de datos (Derek) 43
7.10 Dominios enumerados 44

08 **Seguridad por diseño** 46

09 **Infraestructura y despliegue** 47
9.1 Diagrama de despliegue 48
9.2 Entornos 49

10 **Diseño de la interfaz** 50

11 **Decisiones de arquitectura** 51

12 **Calidad y pruebas** 52

01

Arquitectura general

1.1 Resumen de la decisión

Deploya es un **monolito modular** con **arquitectura hexagonal** dentro de cada módulo, desplegado como cinco servicios de `docker compose` en una sola EC2 de AWS creada con Terraform. La decisión completa, con alternativas y consecuencias, está en docs/arquitectura.md; el piloto en AWS, en el ADR 0009. En una línea: **un solo sistema fácil de operar y de instalar, con fronteras internas tan claras como las de un sistema distribuido, y un dominio que se prueba sin infraestructura.**

Atributo de calidad que manda	Cómo lo atiende la arquitectura
Aislamiento y seguridad de las apps de los clientes	La API no toca Docker: solo el trabajador monta `docker.sock`. Cada app corre sin privilegios, con límites de CPU y memoria y en su propia red
Disponibilidad de lo que ya está publicado	Conmutación sin corte: la ruta se mueve al contenedor nuevo solo después de que pasa la verificación de salud
Capacidad de respuesta del panel	Construir tarda minutos: va por una cola (BullMQ) a un proceso aparte, y la API responde al encolar
Mantenibilidad y pruebas	Puertos como `abstract class` y adaptadores intercambiables: 1 250 pruebas corren sin Docker, red, base ni reloj reales
Portabilidad (autoalojado)	Todo es configuración: el mismo código corre en local, en AWS y en el servidor de un tercero
Costo en un nodo de 8 GB	Dos procesos Node en vez de diez servicios; la memoria queda para las apps de los clientes

1.2 Contexto (C4 nivel 1)

Deploya se ubica entre dos actores humanos y cinco sistemas externos. El cliente solo usa el navegador; la plataforma traduce sus acciones en operaciones sobre GitHub, Docker, el DNS, la autoridad certificadora y el correo.

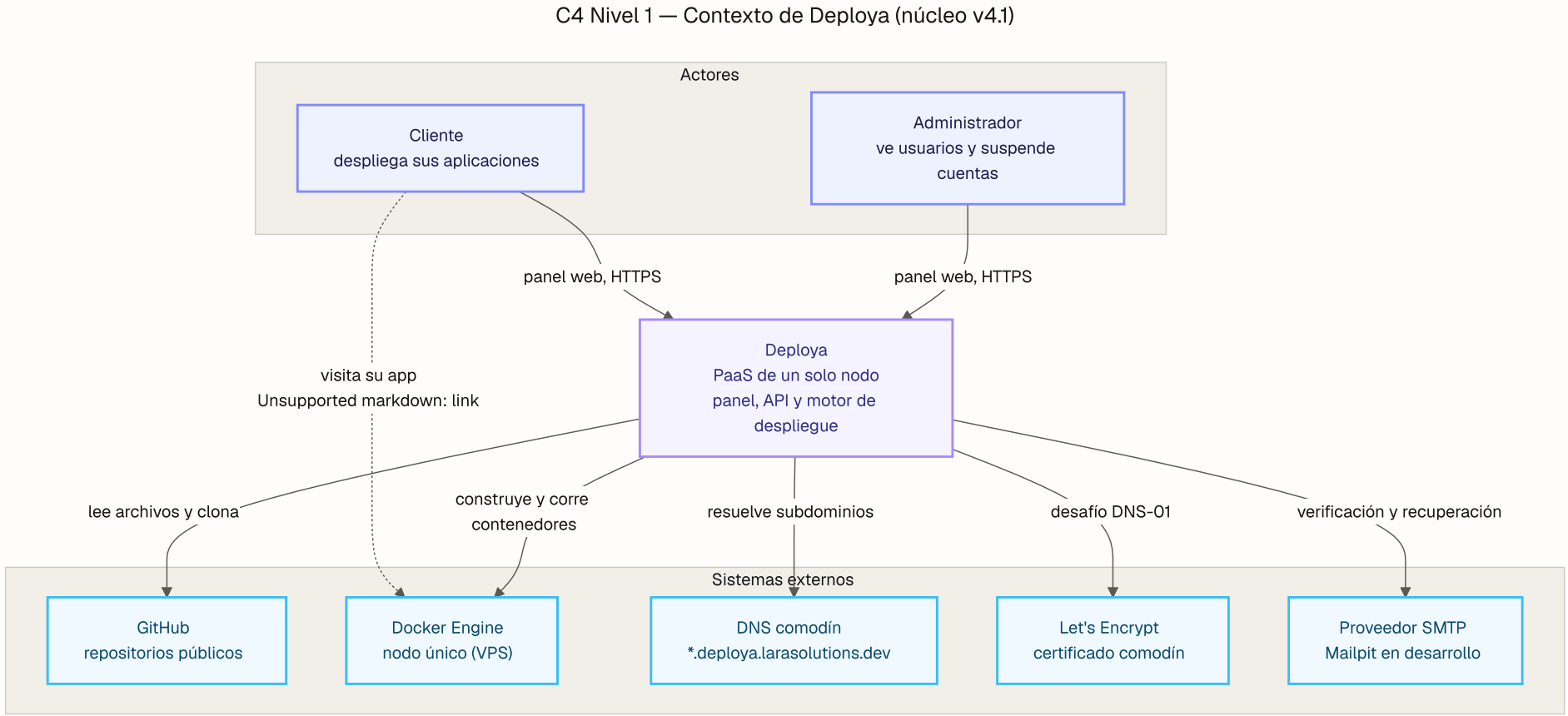


Figura 1. C4 Nivel 1 — Contexto de Deploya (núcleo v4.1)

1.3 Contenedores (C4 nivel 2)

Contenedor	Tecnología	Responsabilidad	Habla con
web	Next.js 15 · React 19 · Tailwind 4	Panel del cliente y del administrador con el design system v4.1. Sondea cada 3 s lo que está en curso	api (HTTPS + cookie)
api	NestJS 10 · Prisma 6	Casos de uso de M1, M2, M3, M7, M9 y M10; produce trabajos de M4 y M5. **No habla con Docker**	postgres, redis, GitHub, SMTP
worker	Misma imagen de la API (`trabajador.ts`)	Consume las colas: clona, construye, ejecuta, publica, muestrea métricas	redis, postgres, Docker, Traefik
postgres	PostgreSQL 16	Almacén único de la plataforma	—
redis	Redis 7	Colas BullMQ (`despliegues` , operación, muestreo) y señales de cancelación	—
traefik	Traefik v3.3	Enrutador de borde: rutas por archivo, TLS comodín, redirección a HTTPS	contenedores de clientes
mailpit	Mailpit (solo desarrollo)	Atrapa los correos para verlos en el navegador	—

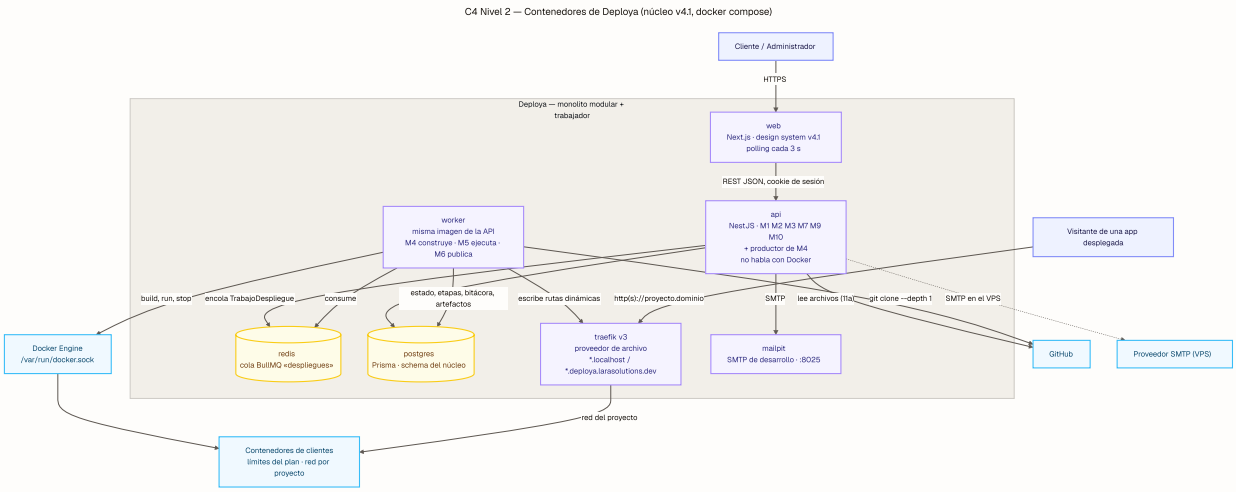


Figura 2. C4 Nivel 2 — Contenedores de Deploya (núcleo v4.1, docker compose)

1.4 El ciclo de despliegue

Todo el producto gira alrededor de un ciclo de cinco etapas. Cada una tiene un módulo responsable y un paso del pipeline del trabajador.

Etapa	Módulo	Paso	Qué produce
Recepción	M4	`PasoRecepcion`	Clon de la rama, commit registrado, stack detectado

Etapa	Módulo	Paso	Qué produce
Construcción	M4	`PasoConstruccion`	Imagen `deploya/<subdominio>:<n>` y su artefacto (digest, tamaño, receta)
Ejecución	M5	`PasoEjecucion`	Contenedor con los límites del plan que **pasa la salud**
Enrutamiento	M6	`PasoEnrutamiento`	Ruta de Traefik al contenedor nuevo; el anterior se retira después
Operación	M5 / M7	`PasoOperacion`	Despliegue *Saludable*, versión activa actualizada, oyentes avisados

Una **reversión** usa el mismo pipeline con otro plan: arranca en *Revirtiendo* y omite Recepción y Construcción, porque la imagen ya existe.

02

Arquitectura de cada módulo

2.1 Puertos y adaptadores

Dentro de `apps/api/src/modules/<módulo>/` el código se ordena en cinco anillos. Las flechas de dependencia apuntan siempre hacia dentro: el dominio no conoce a nadie y los adaptadores conocen solo el puerto que implementan.

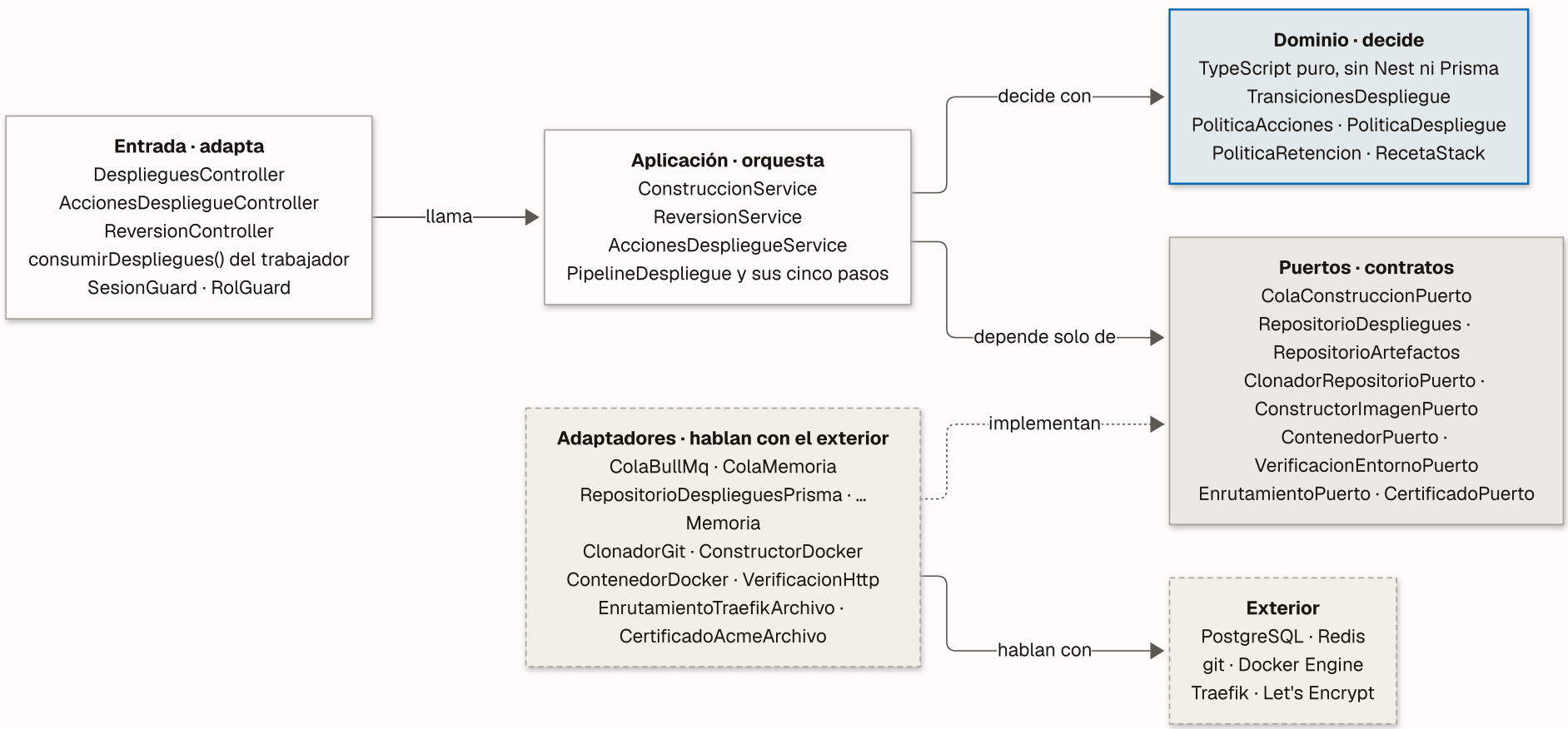


Figura 3. Arquitectura hexagonal dentro de cada módulo — ejemplo real del motor (M4–M6)

Capa	Qué vive ahí	Puede depender de	Nunca depende de
Entrada	Controladores HTTP, consumidores de cola, guards, filtros de errores	El servicio de su módulo	Prisma, adaptadores
Aplicación	Servicios que orquestan un caso de uso; el pipeline del motor	Dominio y puertos	Adaptadores concretos
Dominio	Entidades, objetos valor, políticas y máquinas de estado	Solo TypeScript	Nest, Prisma, Docker, HTTP, `Date.now()`
Puertos	Contratos que la aplicación necesita, como `abstract class`	Tipos del dominio	Implementaciones
Adaptadores	Prisma, Docker, BullMQ, Traefik, SMTP, GitHub, memoria y *stubs*	El puerto que implementan	Otros módulos

> **IMPORTANTE**
>
> Los puertos son `abstract class` y no `interface` porque en TypeScript la interfaz desaparece al compilar y Nest necesita un valor como token de inyección. No llevan prefijo `I`: se llaman por lo que hacen (`ContenedorPuerto`, `PasarelaPago`, `CorreoPuerto`).

Un puerto real del motor, tal como está en el código:

```
/** Puerto DIP — la API no llama a Docker. Sin prefijo I. */
export abstract class ContenedorPuerto {
  abstract crear(espec: EspecContenedor): Promise<ContenedorCreado>;
  abstract iniciar(contenedorId: string): Promise<void>;
  abstract detener(contenedorId: string): Promise<void>;
  abstract eliminar(contenedorId: string): Promise<void>;
  abstract eliminarContenedoresDe(subdominio: string): Promise<void>;
  abstract eliminarImagen(imagen: string): Promise<void>;
  abstract eliminarImagenesDe(subdominio: string): Promise<void>;
  abstract bitacora(contenedorId: string, consulta: ConsultaSalida): Promise<LineaSalida[]>;
}
```

El adaptador se elige **en un solo lugar** y por configuración. Así, `MOTOR_ADAPTADORES=stub` levanta la plataforma completa sin Docker (desarrollo y CI) y `docker` usa los adaptadores reales, sin tocar ningún servicio:

Puerto	Adaptador real	Doble para pruebas y modo `stub`
`ColaConstruccionPuerto`	`ColaBullMq`	`ColaMemoria`
`ClonadorRepositorioPuerto`	`ClonadorGit`	`ClonadorStub`
`ConstructorImagenPuerto`	`ConstructorDocker`	`ConstructorImagenStub`
`ContenedorPuerto`	`ContenedorDocker` (dockerode)	`ContenedorStub`
`VerificacionEntornoPuerto`	`VerificacionHttp`	`VerificacionEntornoStub`

Puerto	Adaptador real	Doble para pruebas y modo `stub`
`EnrutamientoPuerto`	`EnrutamientoTraefikArchivo`	`EnrutamientoStub`
`CertificadoPuerto`	`CertificadoAcmeArchivo`	`CertificadoAusente` (Null Object)
`MetricasContenedorPuerto`	`MetricasDockerStats`	`MetricasContenedorStub`
`CorreoPuerto`	`CorreoSmtpAdaptador` (nodemailer)	`CorreoConsolaAdaptador`
`PasarelaPago`	`PasarelaSimulada`	La misma (es simulada por diseño)
`ProveedorFuente`	`FuenteGitHubPublica`	Dobles en las pruebas
`RepositorioDespliegues`, `RepositorioProyectos` …	`…Prisma`	`…Memoria`

`adapters/stubs/contratos-stubs.spec.ts` comprueba, método por método, que el *stub* y el adaptador real exponen el mismo contrato: es la prueba de Liskov hecha código.

2.2 Módulos y contratos entre ellos

Un módulo solo usa lo que otro **exporta** en su `*.module.ts`, y casi siempre a través de un puerto del consumidor: M3 no importa el servicio de M2, declara `CuotaProyectosPuerto` y un adaptador lo resuelve contra M2. Los datos que cruzan módulos están firmados en `docs/contratos/`.

Módulo	Expone a otros	Consume de otros (puerto del consumidor)
M1 Identidad	`SesionGuard`, `RolGuard`, `@UsuarioActual()`, directorio de cuentas	M2 `AsignacionSandboxPuerto`; M10 `CorreoPuerto`; M9 `PoliticaRegistroPuerto`
M2 Suscripciones	`cuotaDe(usuarioId)`, catálogo, estado de la suscripción	M5 `ContenedoresClientePuerto` (detener y reanudar); M10 `AvisosPuerto`
M3 Proyectos	Lectura de proyectos y variables descifradas	M2 `CuotaProyectosPuerto`; M4 `DeteccionStackService`, `crearDespliegue`
M4 Construcción	`crearDespliegue`, vista y bitácora del despliegue, tiempos por etapa	M3 `ProyectosLecturaPuerto`; M5 `BloqueoDespliegue`; M10 `AvisosPuerto` (Observer)
M5 Orquestación	Acciones de contenedor, reversión, métricas	M2 `CuotaPlanPuerto`; M3 `VariablesEntornoPuerto`
M5 · datos	Recursos de datos, conexión y variables inyectadas	M2 `CuotaDatosPuerto`; M5 `RecursosDatosPuerto` (contenedor con volumen)
M6 Enrutamiento	`EnrutamientoPuerto`, `CertificadoPuerto`	—

Módulo	Expone a otros	Consume de otros (puerto del consumidor)
M7 Observabilidad	Resumen, historial, consumo, métricas, bitácora de ejecución	M4 y M5 (lectura)
M9 Administración	Usuarios, suspensión, instalación, infraestructura	M1 `CuentasSuspensionPuerto`; M5 contenedores
M10 Notificaciones	`CorreoPuerto`, `AvisosPuerto`	M1 `DestinatariosPuerto`

Contrato	Productor → consumidor	Qué fija
`datos-nucleo.md`	Prisma → todos	Modelo de datos del núcleo (v2 con métricas)
`despliegues.md`	M4/M5 → web	Vista del despliegue, etapas, bitácora, acciones (v2 con reversión)
`metricas.md`	M5 → M7 → web	Serie de 1 h y 24 h, picos, eventos
`avisos.md`	M2/M4 → M10	Avisos de despliegue y de suscripción
`instalacion.md`	instalador → M1/M9	Código de configuración, modo, salud de la instalación
`recursos-datos.md`	M5 datos → web, M2, apps	Tipos, estados, límites del plan, API y variables inyectadas

El motor de despliegue

3.1 Componentes del motor (C4 nivel 3)

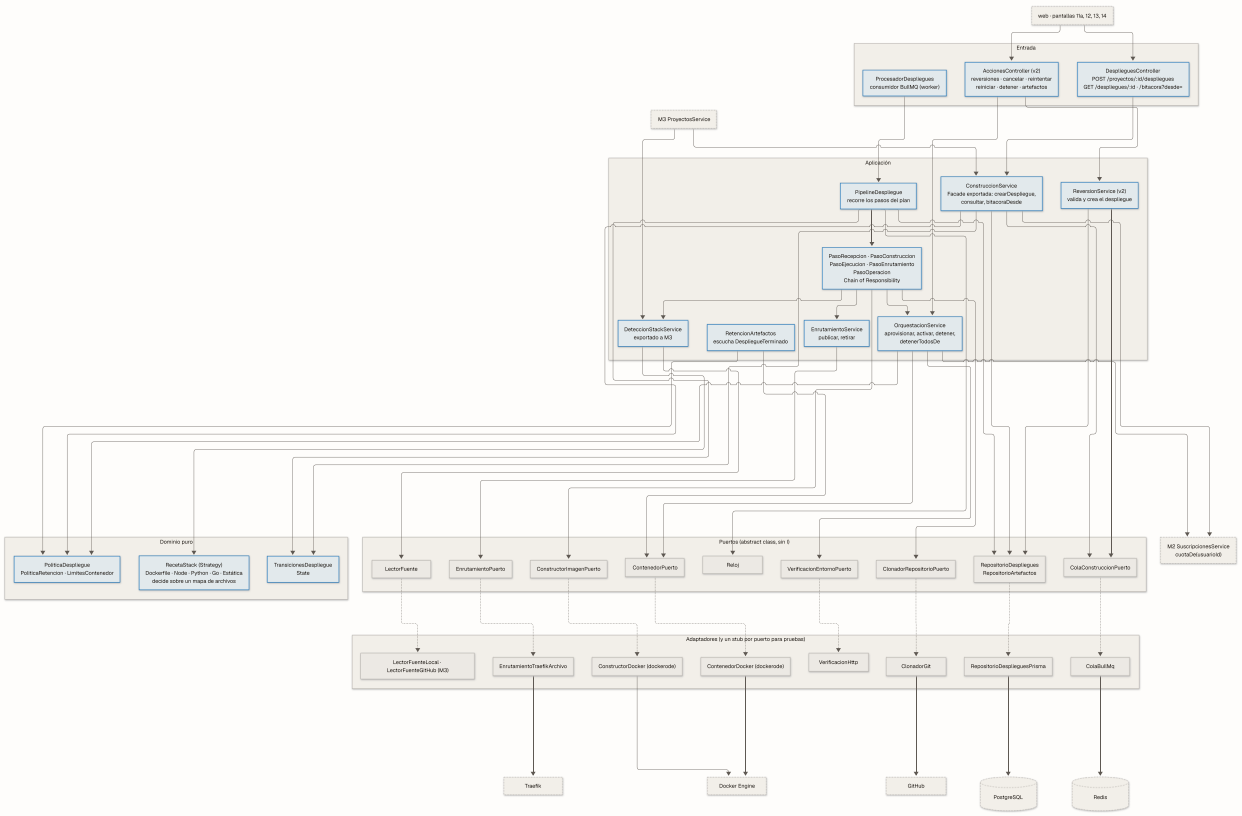


Figura 4. C4 Nivel 3 — Componentes del motor de despliegue (M4 M5 M6), hexagonal

3.2 El pipeline como cadena de pasos

`PipelineDespliegue` es el corazón del trabajador. Recibe un `TrabajoDespliegue { despliegueId, plan }` de la cola y recorre una **lista ordenada de pasos**. Cada paso conoce su etapa y su estado, y el pipeline hace lo común: valida la transición, marca la etapa en curso, ejecuta, la marca completada y vacía la bitácora.

- **Chain of Responsibility:** los pasos se ejecutan en orden y cada uno deja su resultado en el `ContextoDespliegue` (clon, receta, imagen, contenedor, URL) para el siguiente.
- **Strategy:** cada *plan* es una lista de pasos. `construccion` usa los cinco; `reversion`, los tres últimos. Un plan nuevo no toca ni los pasos ni el pipeline.
- **Compensación (estilo *saga*):** si un paso falla, el pipeline llama a `compensar()` de los que ya corrieron, en orden inverso. Un contenedor que no pasó la salud se borra y la ruta sigue en el anterior. `liberar()` corre siempre (borra el clon temporal).

- **Observer:** al terminar, el pipeline anuncia `DespliegueTerminado` a los `OyenteDespliegue` registrados: `RetencionArtefactos` retira imágenes viejas y `AvisoDespliegueOyente` manda el correo. El pipeline no sabe que existen el correo ni la retención.
- **Cancelación cooperativa:** `VigiaCancelacion` escucha una señal en Redis y aborta el `AbortController` del despliegue; los pasos largos (`git clone`, `docker build`) respetan la señal.

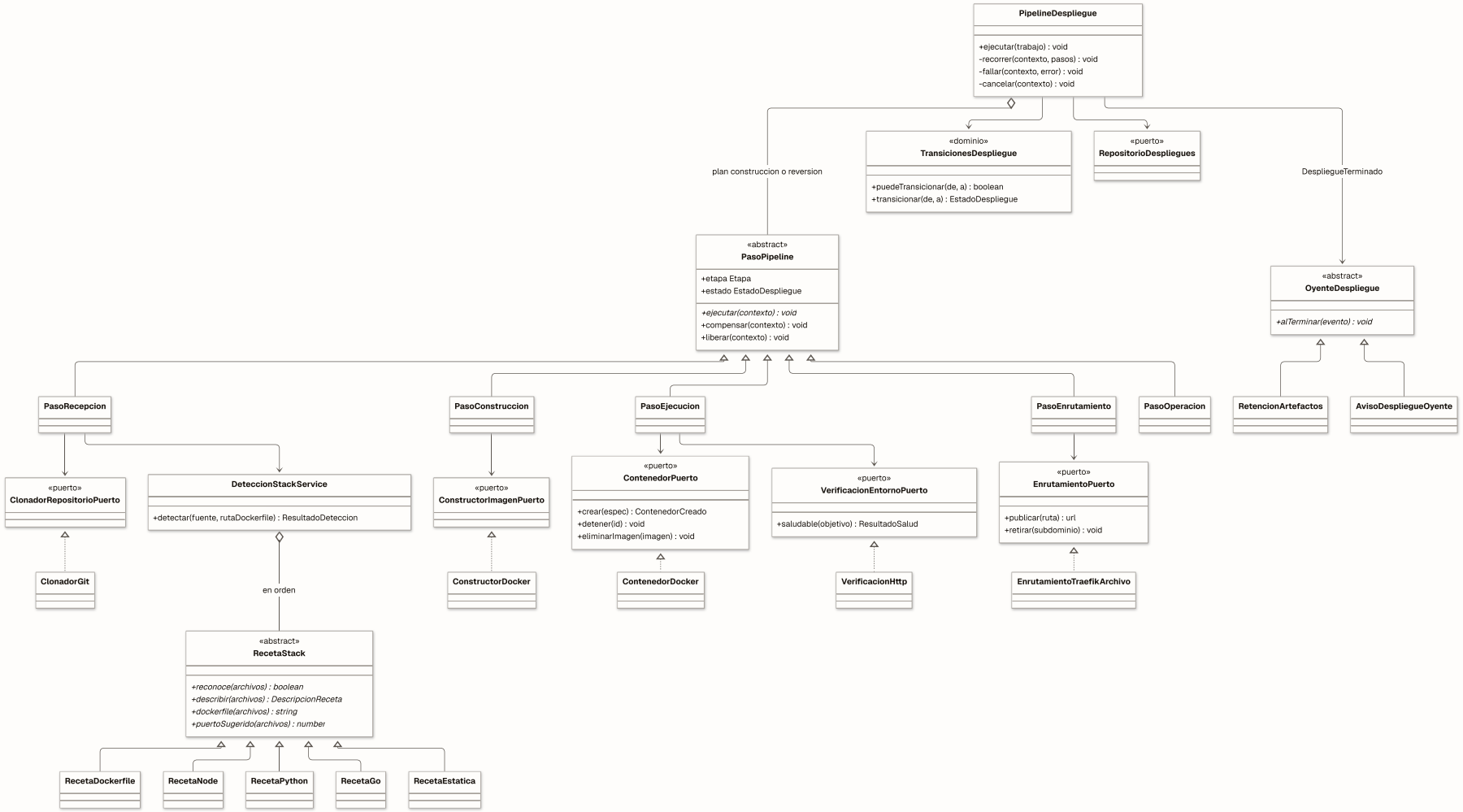


Figura 5. Clases del motor de despliegue (M4 Construcción, M5 Orquestación, M6 Enrutamiento)

3.3 Máquina de estados del despliegue

Los estados viven en **una sola tabla pura**, `TransicionesDespliegue`. Ningún servicio cambia un estado sin pasar por `transicionar(de, a)`, que lanza `TransicionInvalida` si el cambio no está permitido:

```
const SIGUIENTES: Readonly<Record<EstadoDespliegue, readonly EstadoDespliegue[]>> = {
  encolado: ["construyendo", "cancelado"],
  construyendo: ["aprovisionando", "fallido", "cancelado"],
  revirtiendo: ["aprovisionando", "fallido", "cancelado"],
  aprovisionando: ["publicando", "fallido"],
  publicando: ["saludable", "fallido"],
  saludable: ["detenido"],
  detenido: ["aprovisionando"],
  fallido: [],
  cancelado: [],
};
```

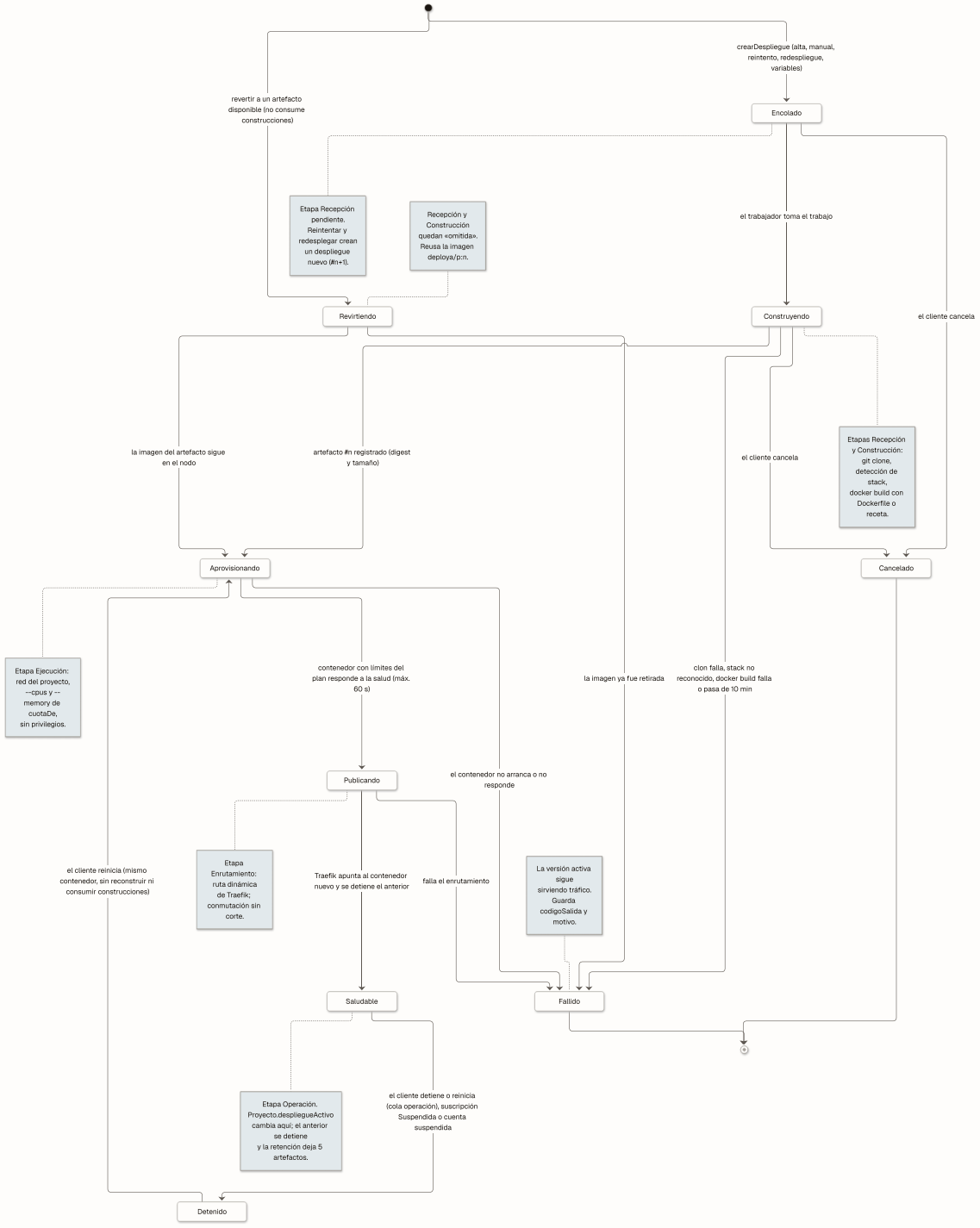


Figura 6. Estados del despliegue — núcleo v4.1 con reversión (no son los de suscripción)

3.4 Detección de stack

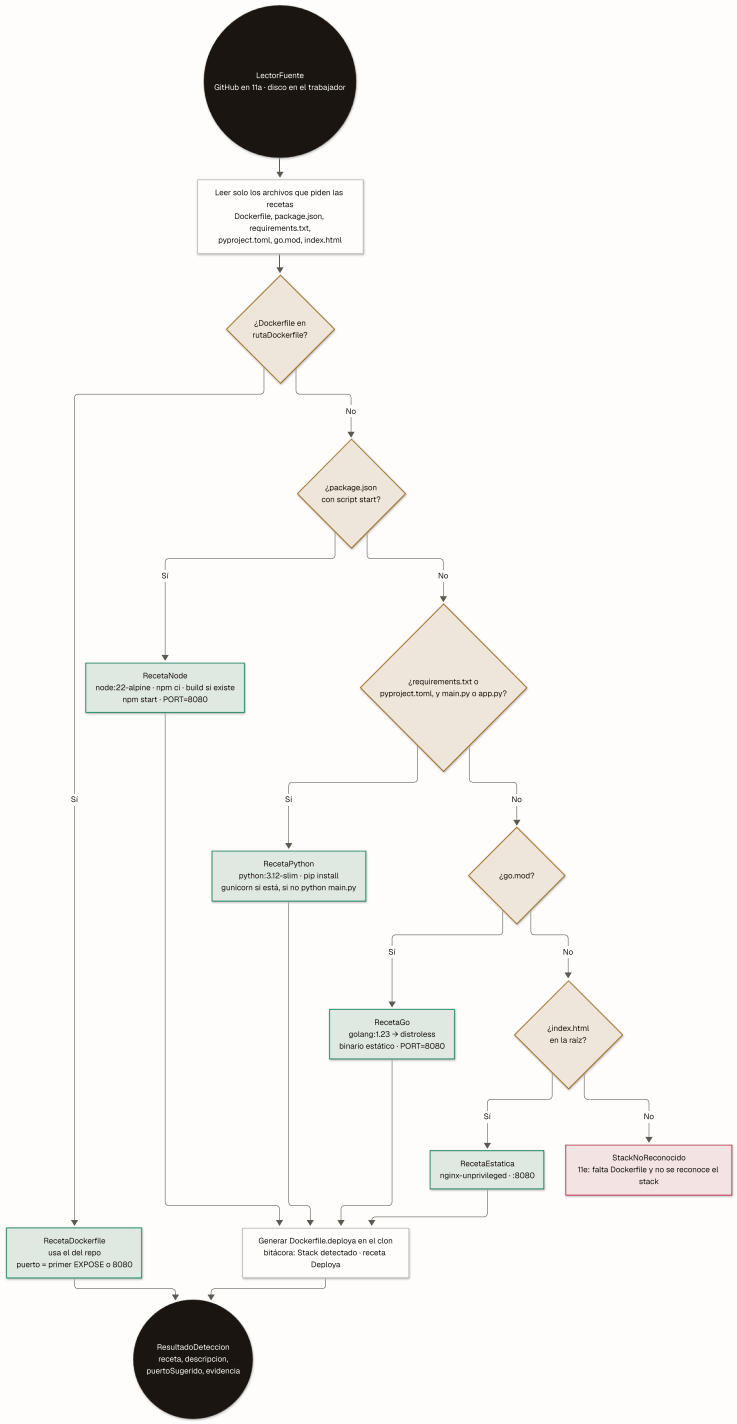


Figura 7. Actividad — detección de stack (M4-03, Strategy de recetas en orden fijo)

Las recetas son **puras**: deciden sobre un mapa de archivos ya leído. Por eso la misma detección sirve en el alta (M3 lee por la API de GitHub con `LectorFuenteGitHub`) y en el trabajador (lee del clon con `LectorFuenteLocal`). Agregar un stack es crear una clase `RecetaX extends RecetaStack` y registrarla en la lista: **abierto a extensión, cerrado a modificación** (ADR 0003).

Diagramas UML

4.1 Clases · Identidad y notificaciones

`IdentidadService` depende de **nueve** abstracciones y de ninguna implementación. `CorreoConRegistro` es un **Decorator**: envuelve cualquier `CorreoPuerto` para registrar el último envío (lo usa Admin · Instalación) sin que el adaptador SMTP lo sepa. `PlantillaCorreo` es un **Template Method**: `componer()` arma el mensaje con el kit visual y cada plantilla solo define asunto, título, párrafo y botón.

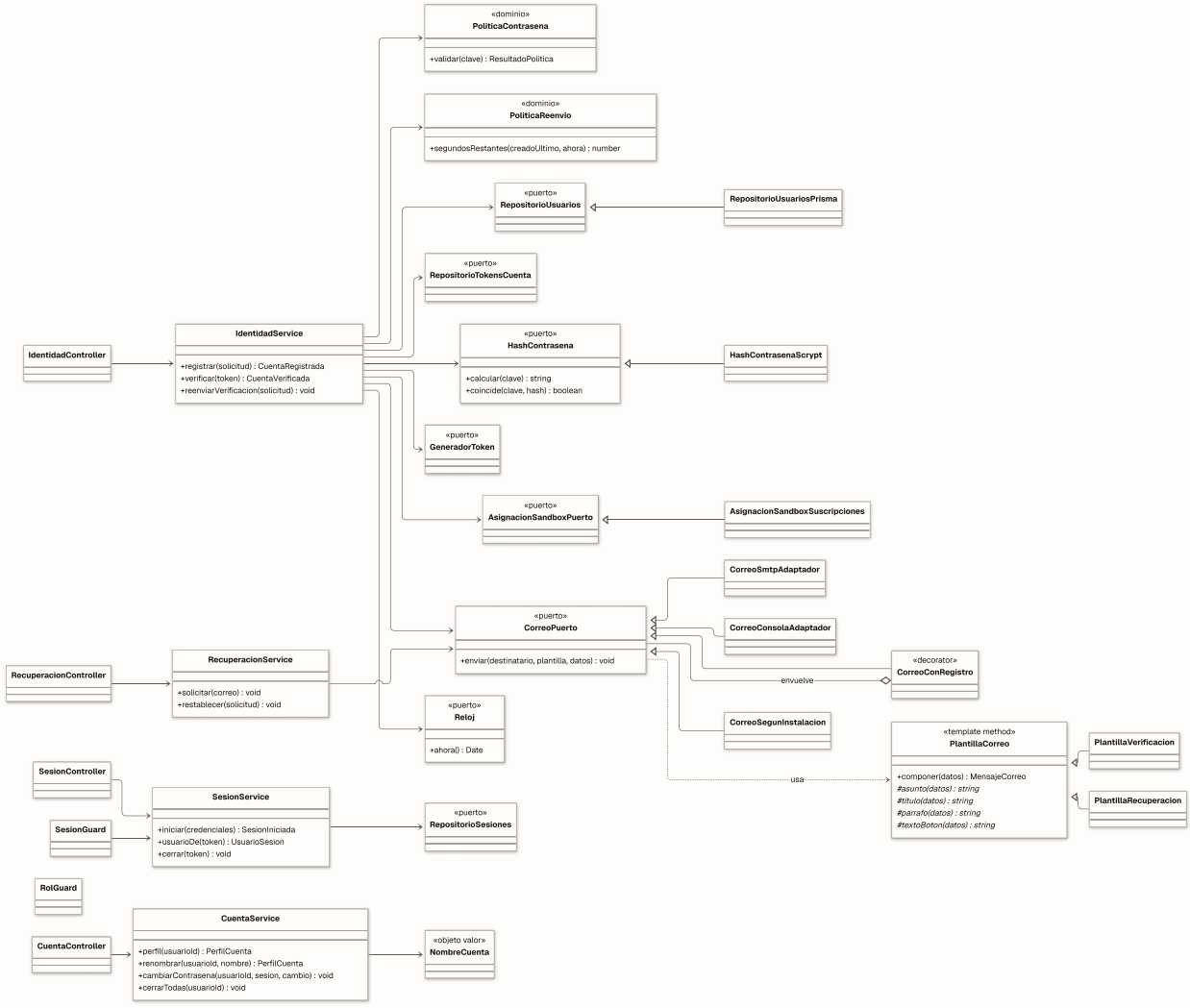


Figura 8. Clases de Identidad (M1) y Notificaciones (M10)

4.2 Clases · Suscripciones y administración

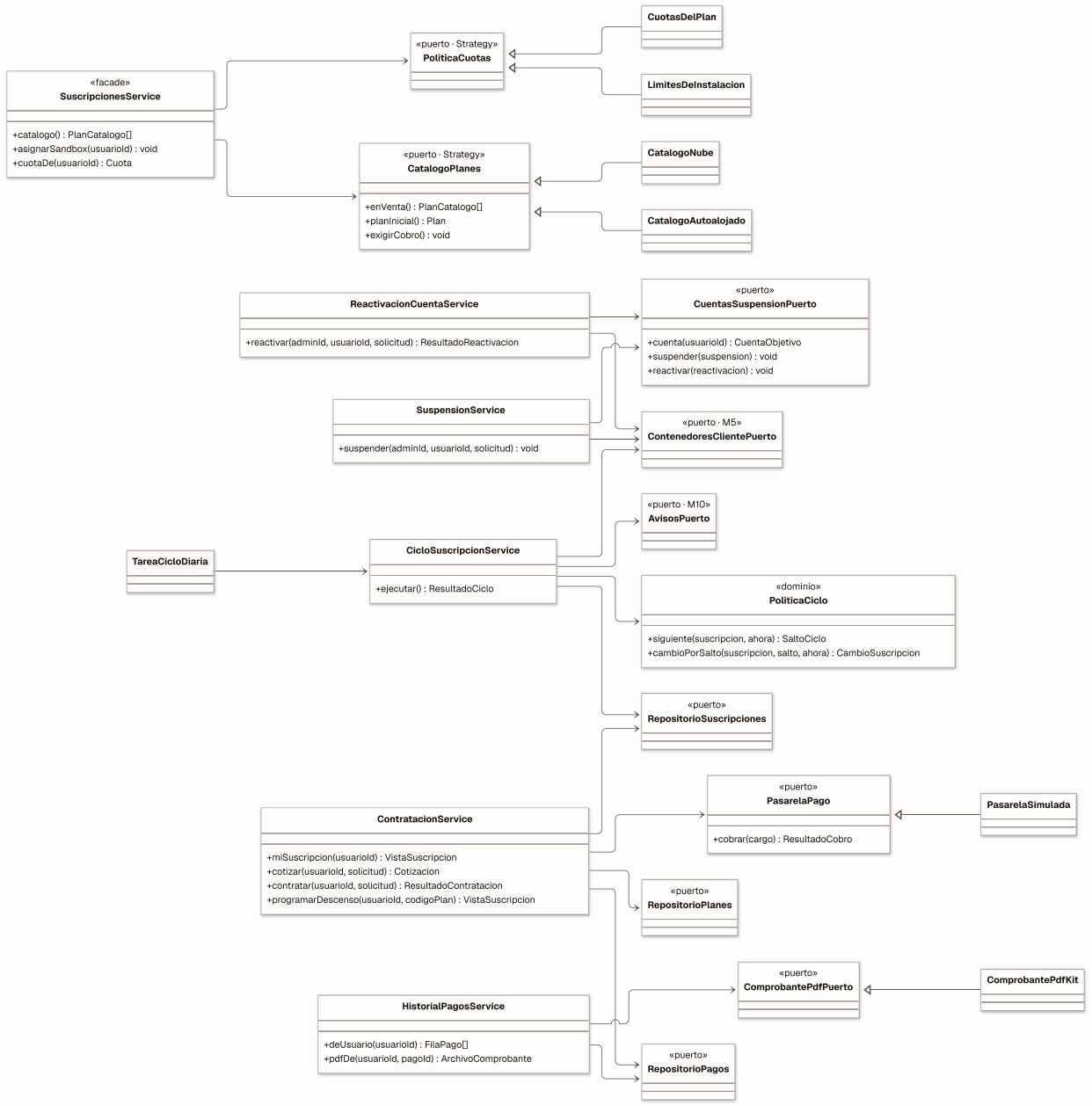


Figura 9. Clases — Suscripciones y pagos (M2) y Administración (M9)

4.3 Clases · Proyectos y observabilidad

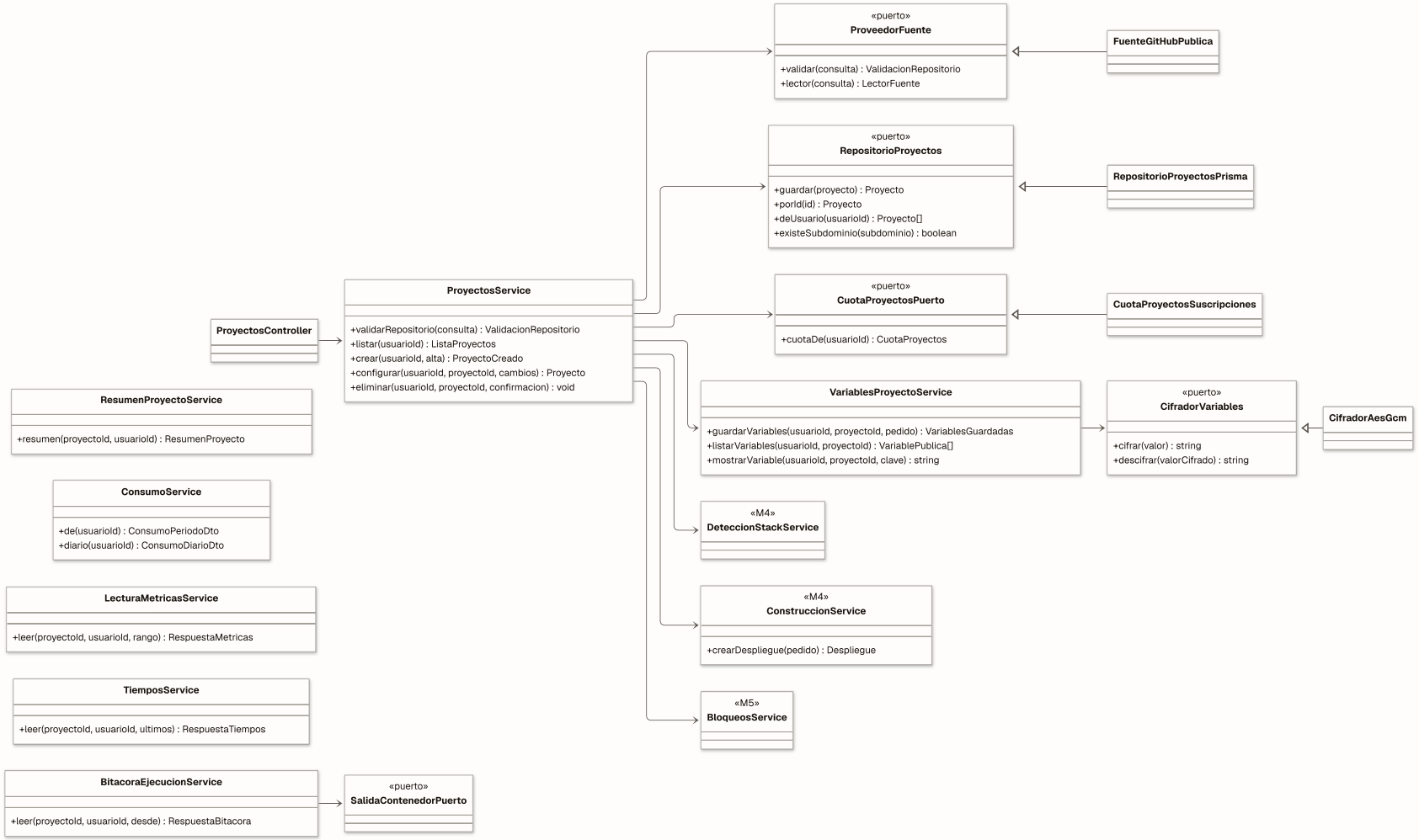


Figura 10. Clases — Proyectos (M3) y Observabilidad (M7)

4.4 Secuencia · Recorrido completo

Del registro a la aplicación en línea, cruzando todos los módulos del núcleo.

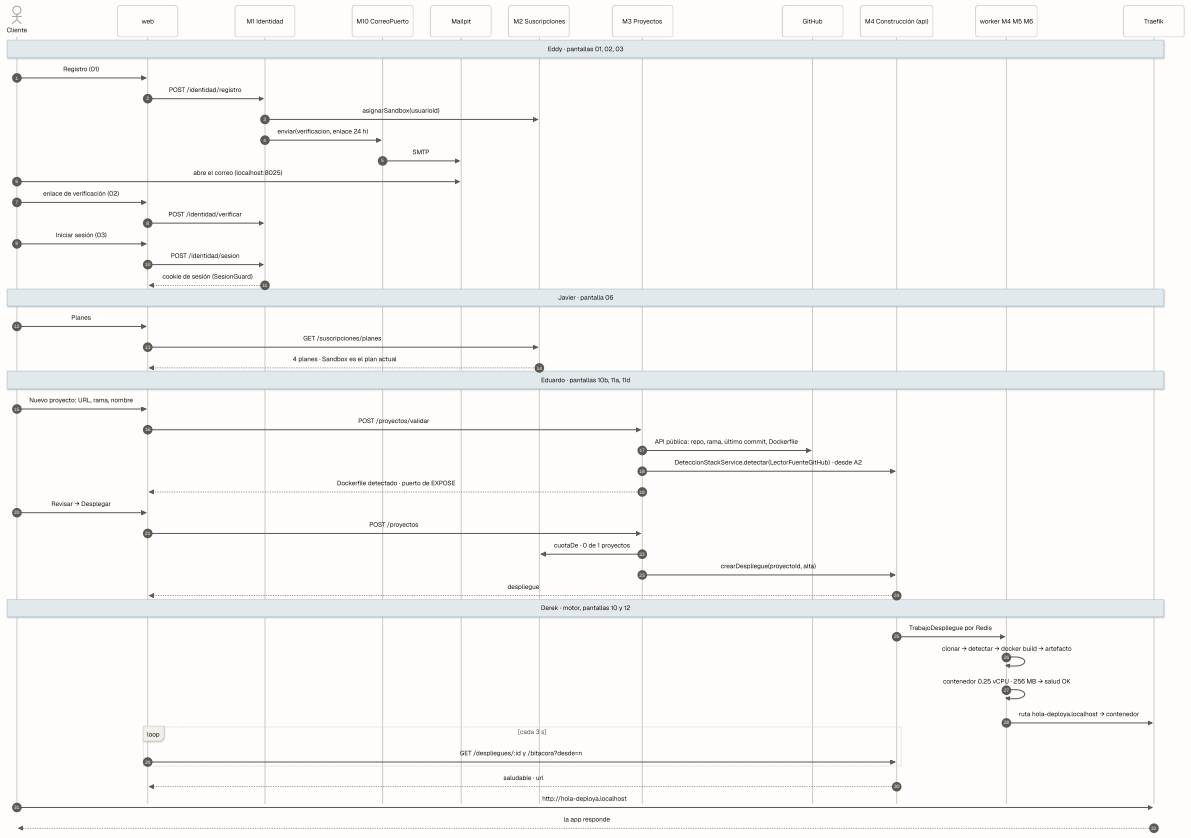


Figura 11. Recorrido de punta a punta del Avance 1 — de registrarse a ver la app Saludable

4.5 Secuencia · Contratación de un plan

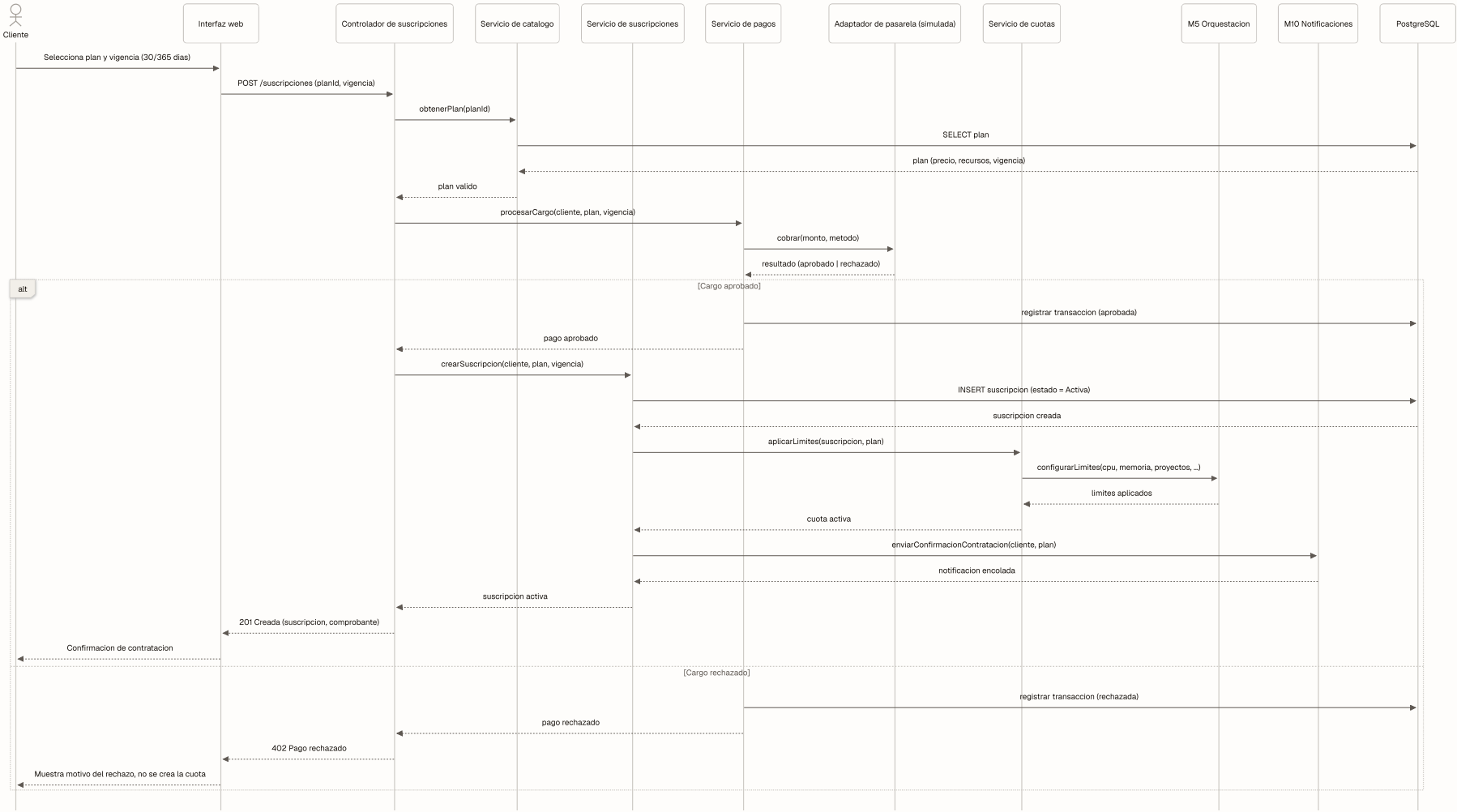


Figura 12. Secuencia — contratación de un plan con la pasarela simulada (M2)

4.6 Secuencia · Despliegue

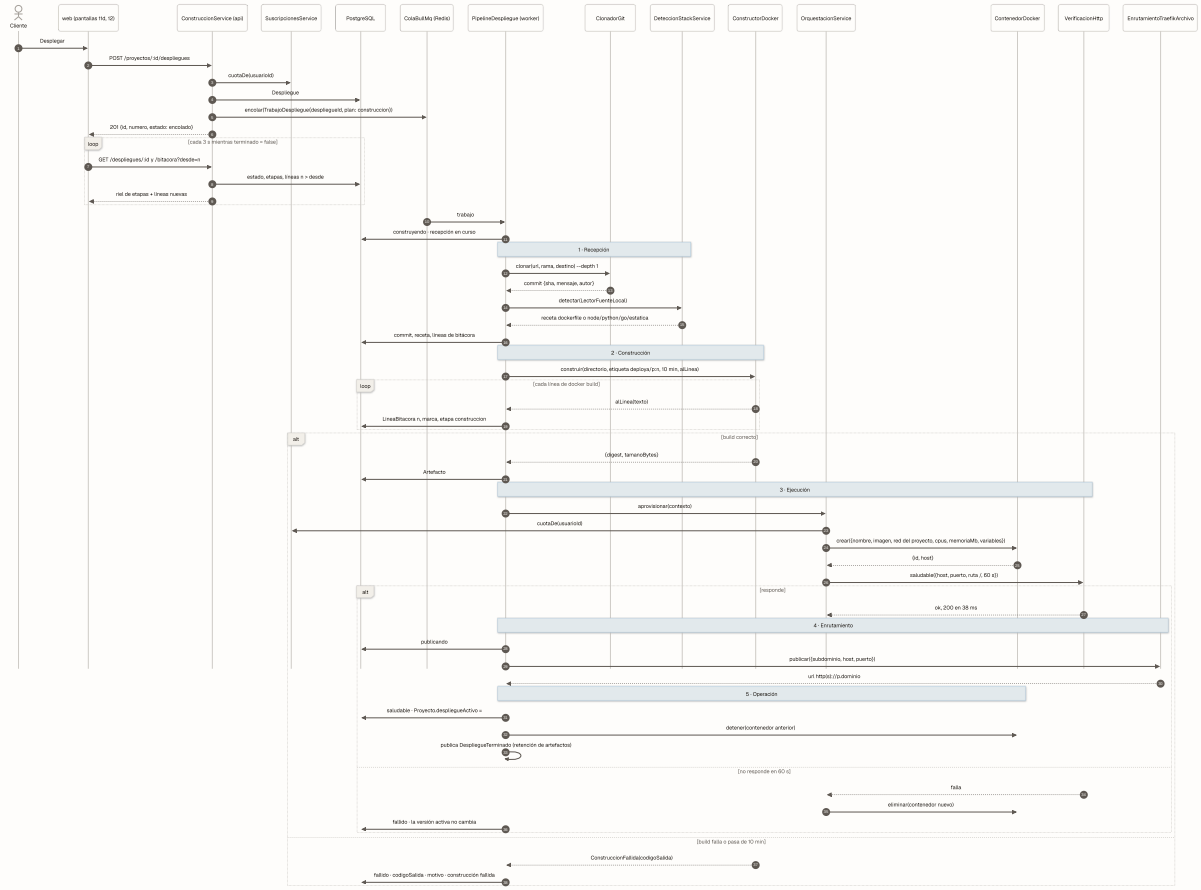


Figura 13. Secuencia de un despliegue por construcción — Recepción → Construcción → Ejecución → Enrutamiento → Operación

4.7 Secuencia · Reversión sin reconstruir

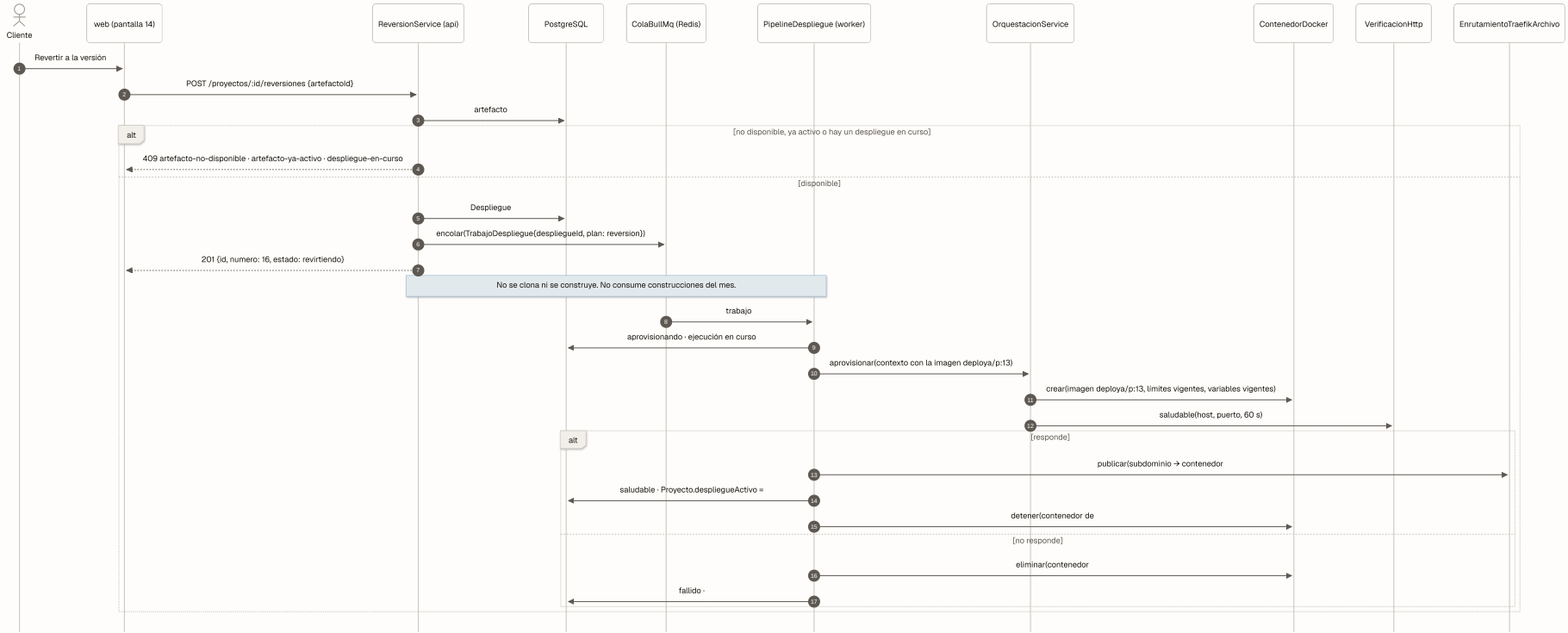


Figura 14. Secuencia de reversión — vuelve a un artefacto anterior sin reconstruir (M5-04)

4.8 Estados · Suscripción

`PoliticaCiclo.siguiente(suscripcion, ahora)` es una función pura: recibe la fecha como parámetro, así que el ciclo de 7, 5 y 30 días se prueba con fechas literales y sin esperar. Cada corrida mueve una suscripción **un estado como máximo**, lo que la hace idempotente.

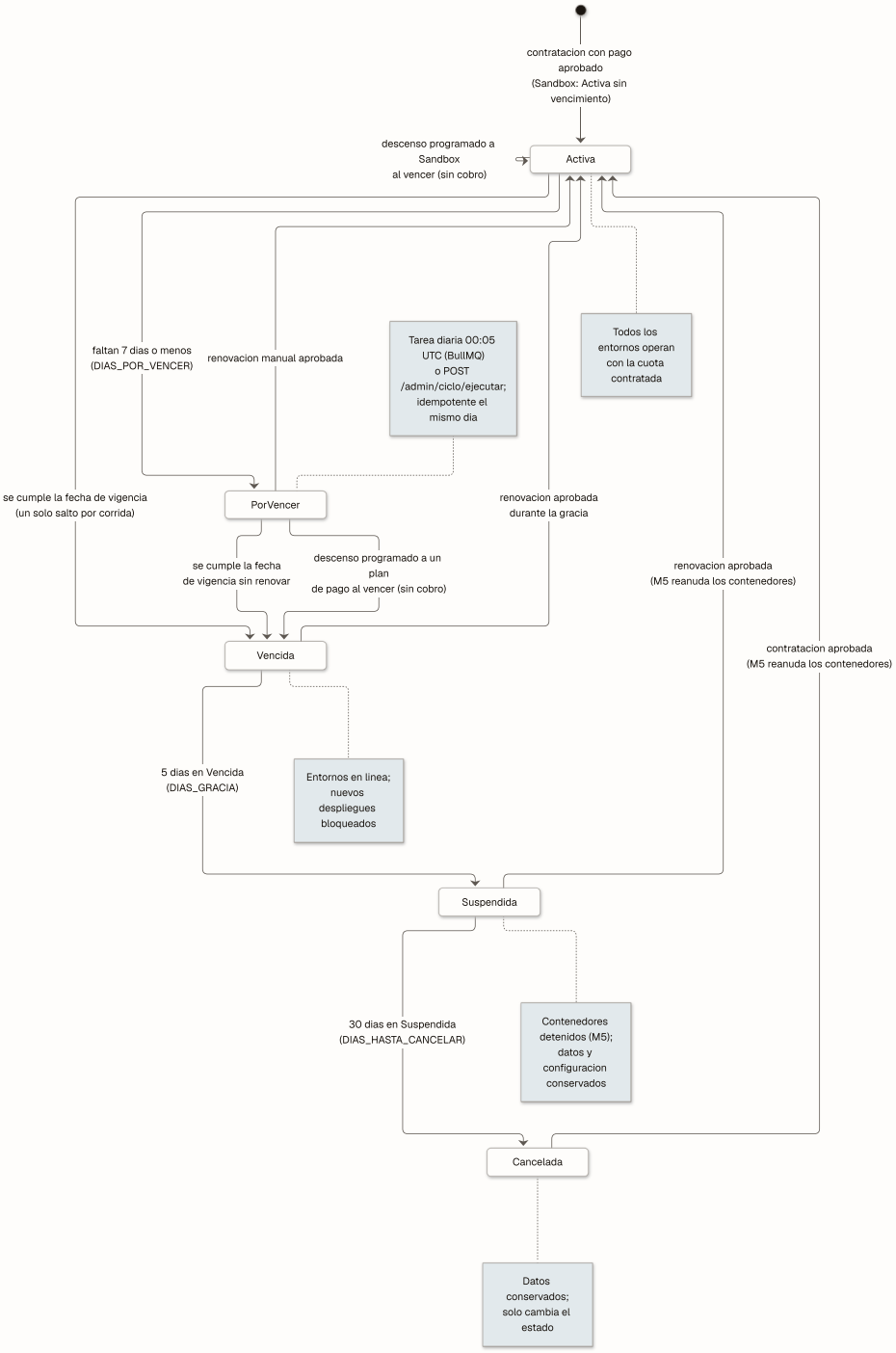


Figura 15. Estados de la suscripción — ciclo §4.4 (no son los de despliegue)

4.9 Actividad · Registro y verificación

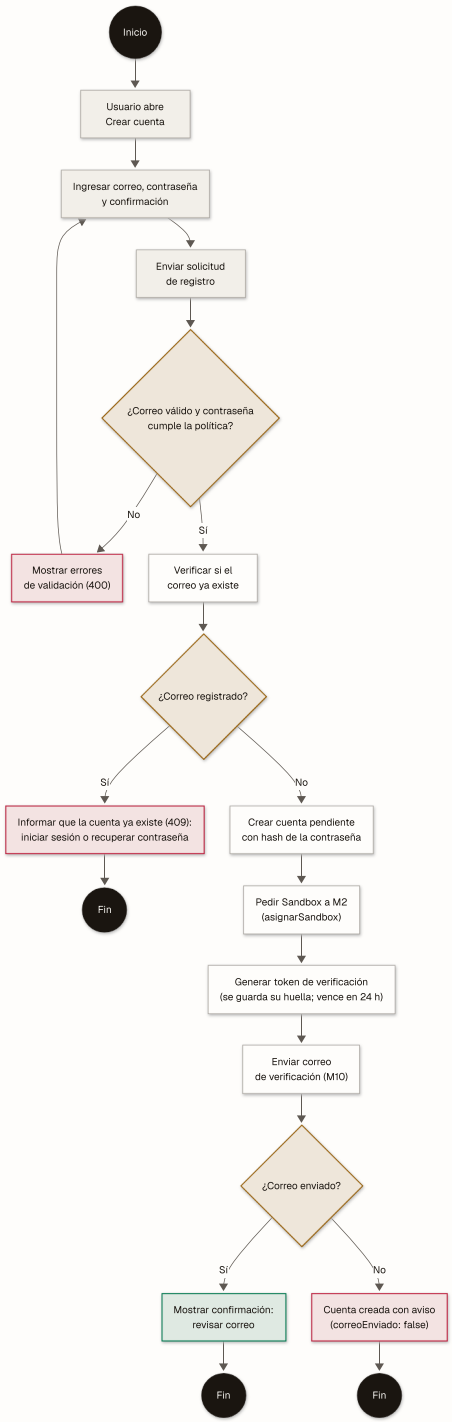


Figura 16. Actividad — registro de cuenta (M1, M10)

4.10 Actividad · Contratación y activación

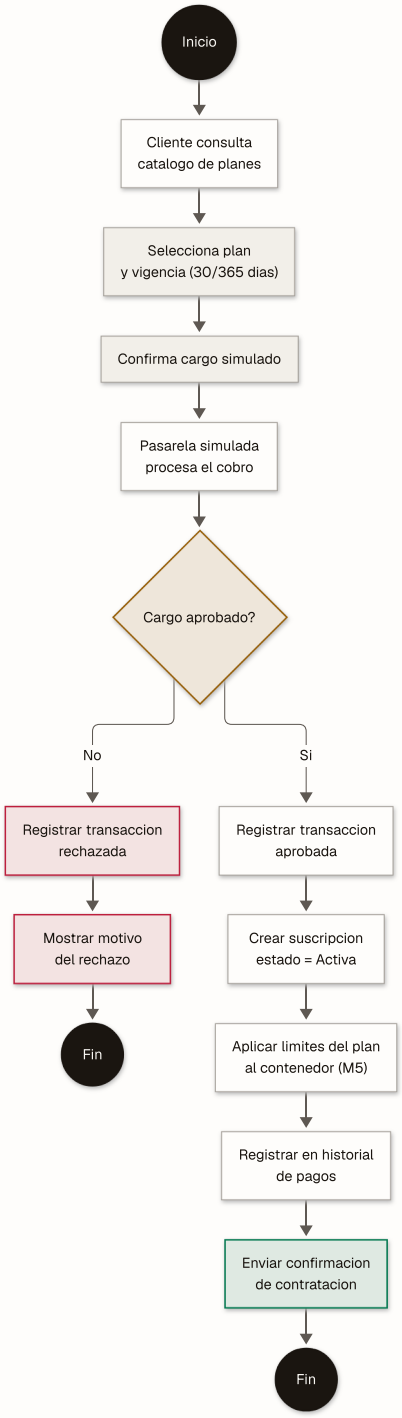


Figura 17. Actividad — contratación y activación de un plan (M2)

05

SOLID aplicado

Cada principio con la regla que sigue el equipo y dónde se puede comprobar en el código.

Principio	Regla en Deploya	Evidencia en el código
S · Responsabilidad única	El controlador adapta HTTP, el servicio orquesta, la política decide, el adaptador habla con el exterior	`PoliticaCiclo` saca el ciclo de `SuscripcionesService`; `PoliticaRetencion` (decide) y `RetencionArtefactos` (ejecuta) están separadas; un controlador por recurso en M9
O · Abierto/cerrado	Se extiende con una implementación nueva, no con `if/switch` por tipo	Recetas de stack (`RecetaNode` , `RecetaPython` …), planes del pipeline, `ManejadorAccion` por acción (`DetenerManejador` , `ReiniciarManejador` , `EliminarManejador`), modo de instalación (ADR 0008)
L · Sustitución de Liskov	Stub, doble y adaptador real cumplen el mismo contrato y lanzan los mismos errores de dominio	`contratos-stubs.spec.ts` ; `CertificadoAusente` devuelve `null` igual que el real sin TLS; `RecetaDockerfile` es una receta más, sin casos especiales
I · Segregación de interfaces	Puertos estrechos pensados desde quien los usa	~70 puertos; `VerificacionEntornoPuerto` tiene un solo método; `ProyectosLecturaPuerto` y `DetalleProyectoPuerto` son vistas distintas del mismo repositorio
D · Inversión de dependencias	Servicios reciben puertos por constructor; nada de `new` de Docker, Prisma, SMTP ni `Date.now()` en el dominio	`IdentidadService` recibe 9 puertos; el tiempo entra por `Reloj` ; los bindings viven en `AdaptersModule` y en cada `*.module.ts`

CONSEJO

Los nombres siguen el dominio en español (`Despliegue` , `Suscripcion` , `cuotaDe`) y los sufijos de Nest quedan en inglés (`ConstruccionService` , `ProyectosController`): el diagrama dice `ServicioConstruccion` y el código `ConstruccionService` , y son el mismo componente.

5.1 Clean code

- **Errores de dominio con nombre** en lugar de `Error('x')` : `CorreoYaRegistrado` , `CuotaConstruccionesAgotada` , `TransicionInvalida` , `ArtefactoNoDisponible` , `StackNoReconocido` . Un filtro por módulo los traduce a HTTP (`ErroresIdentidadFilter` , `RechazosOrquestacionFilter` , `ErroresMotorFilter`).
- **Sin números mágicos:** `TIEMPO_MAXIMO_CONSTRUCCION_MS = 10 * 60 * 1000` , `TIEMPO_MAXIMO_SALUD_MS` , `DIAS_POR_VENCER = 7` , `DIAS_GRACIA = 5` , `ESPERA_REENVIO_MS = 60_000` .
- **Tipos estrictos:** sin `any` ; lo externo entra como `unknown` y se valida en el borde.
- **Consultas separadas de comandos:** M7 solo lee; el motor escribe.

Patrones de diseño

Patrón	Dónde	Clases	Por qué
Ports & Adapters (hexagonal)	Todos los módulos	~70 puertos `abstract class`	El dominio se prueba sin infraestructura y el proveedor se cambia por configuración
Repository (Fowler)	Persistencia por agregado	`RepositorioUsuarios`, `RepositorioDespliegues`, `RepositorioPagos` … con versión Prisma y en memoria	El servicio no conoce SQL ni Prisma
Adapter (GoF)	Exterior	`ContenedorDocker`, `EnrutamientoTraefikArchivo`, `FuenteGitHubPublica`, `CorreoSmtpAdaptador`	Traducir una API externa al contrato del puerto
Strategy	Pagos, recetas, planes del pipeline, modo de instalación	`PasarelaPago`, `RecetaStack`, `PlanesPipeline`, `CatalogoPlanes`, `PoliticaCuotas`	Variar el algoritmo sin tocar a quien lo usa
State (tabla pura)	Despliegue y suscripción	`TransicionesDespliegue`, `PoliticaCiclo`	Una sola fuente de verdad de las transiciones, probada sin base
Chain of Responsibility	Motor	`PasoPipeline` y sus cinco pasos	Cada etapa es una clase con su compensación
Command + productor/consumidor	Colas	`TrabajoDespliegue`, acciones de operación	Construir fuera del ciclo HTTP; `jobId = despliegueId` lo hace idempotente
Observer	Fin de un despliegue	`OyenteDespliegue` ← `RetencionArtefactos`, `AvisoDespliegueOyente`	El pipeline no depende del correo ni de la retención
Decorator	Correo	`CorreoConRegistro` envuelve un `CorreoPuerto`	Registrar envíos sin tocar el adaptador SMTP
Template Method	Plantillas de correo	`PlantillaCorreo.componer()`	Mismo kit visual; cada correo define solo su texto
Null Object	Certificado sin TLS	`CertificadoAusente`	La vista no pregunta si hay TLS
Facade	Fronteras entre módulos	`SuscripcionesService.cuotaDe`, `asignarSandbox`	Una sola puerta por módulo
Composite	Salud de la instalación	`SaludInstalacion` y sus comprobaciones	Un estado general a partir de varias comprobaciones

Patrón	Dónde	Clases	Por qué
Guard / Decorator de Nest	Autorización	`SesionGuard`, `RolGuard`, `@Roles()`, `@UsuarioActual()`	La autorización no ensucia los controladores
Container / Presentational	Web	Hooks (`useDespliegue`) y componentes sin `fetch`	La lógica de la web se prueba con `node --test`
Strategy (recursos de datos)	M5 · datos	`TipoRecursoDatos`: `postgres`, `redis`, `s3`	Cada tipo sabe su imagen, su puerto, su memoria, cómo prepararse y qué variables inyecta; un tipo nuevo es una clase más (OCP)

07

Modelo de datos

7.1 Modelo entidad-relación

Dieciocho tablas en PostgreSQL. El modelo se agrupa por el módulo dueño de cada tabla; las relaciones que cruzan módulos (`Usuario → Proyecto`, `Plan → Suscripcion`) están firmadas en `docs/contratos/datos-nucleo.md`.

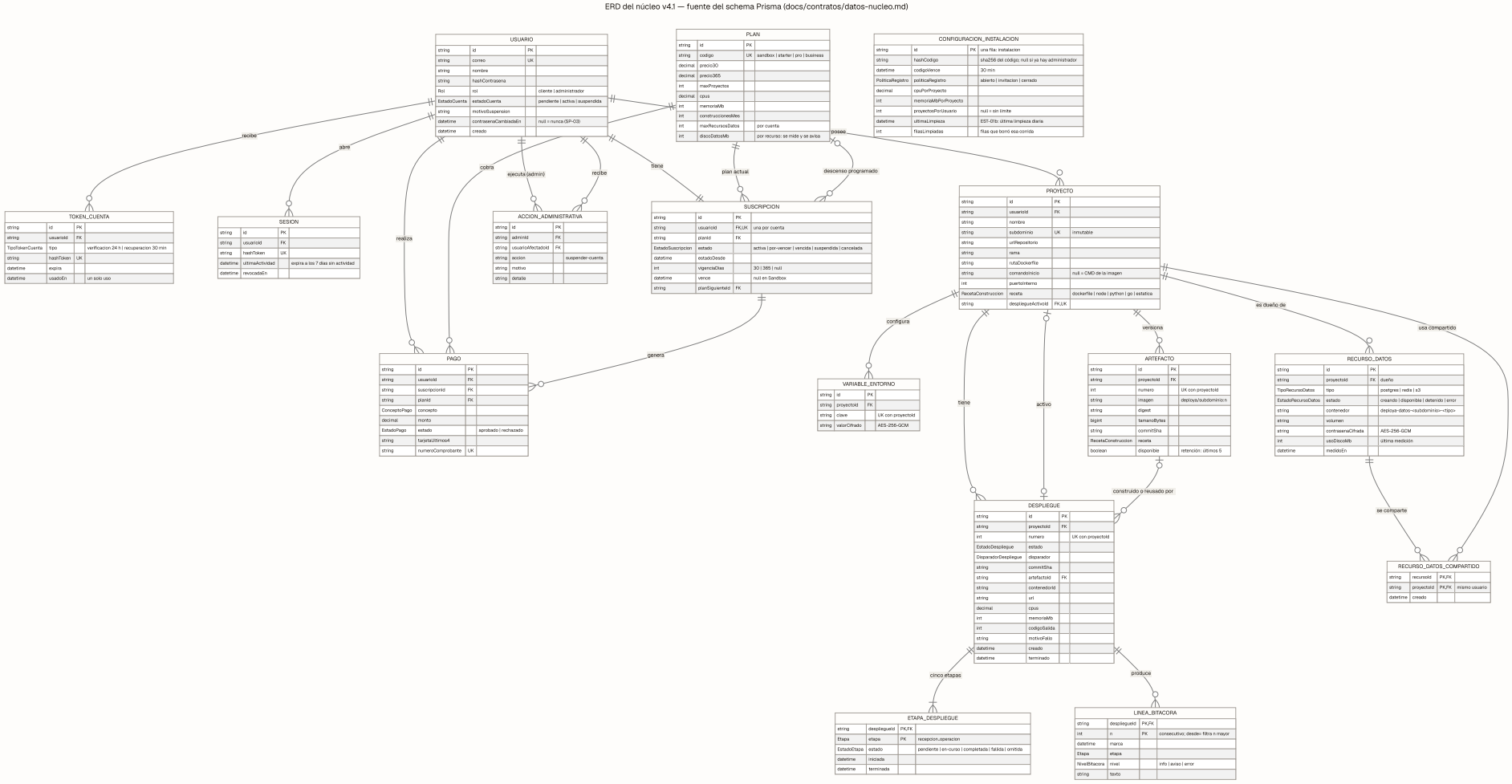


Figura 18. ERD del núcleo v4.1 — fuente del schema Prisma (docs/contratos/datos-nucleo.md)

7.2 Decisiones de diseño de la base

Decisión	Por qué
Claves primarias `uuid` generadas por la aplicación	No revelan volumen ni orden; se pueden crear antes de insertar (el despliegue y su trabajo comparten id)
Claves naturales únicas aparte del `id`	`Usuario.correo`, `Plan.codigo`, `Proyecto.subdominio`, `Pago.numeroComprobante`: el *seed* es idempotente por clave natural
Únicas compuestas donde el dominio lo pide	`(proyectoId, numero)` en `Despliegue` y `Artefacto`; `(usuarioId, nombre)` en `Proyecto`; `(proyectoId, clave)` en `VariableEntorno`
Claves compuestas sin `id` artificial	`EtapaDespliegue (despliegueId, etapa)`, `LineaBitacora (despliegueId, n)`, `MuestraMetrica (despliegueId, tomada)`: son series que se leen siempre por su despliegue
Dinero en `numeric(10,2)` y moneda `char(3)`	Nada de flotantes para importes
Enums de PostgreSQL para los estados	La base rechaza un estado que no existe; los valores coinciden con el contrato (`por-vencer`, `en-curso`)
Borrado en cascada desde el agregado raíz	Eliminar un proyecto borra variables, despliegues, etapas, bitácora, artefactos y muestras; eliminar un usuario borra todo lo suyo
`SetNull` en referencias débiles	`Proyecto.despliegueActivoId` y `Despliegue.artefactoId` no bloquean el borrado del otro lado
Índices por la consulta que los usa	`(estado, vence)` para el ciclo diario; `(usuarioId, creado)` para el historial de pagos; `(proyectoId, creado)` para el historial de despliegues; `tomada` para purgar métricas
Secretos nunca en claro	Contraseñas con `scrypt`; tokens solo como SHA-256; variables y SMTP con AES-256-GCM
Una fila de configuración	`ConfiguracionInstalacion.id = "instalacion"` garantiza una sola configuración en autoalojado

Migraciones: siete migraciones de Prisma versionadas en `apps/api/prisma/migrations` (núcleo del 29 de septiembre, más métricas, configuración de la instalación, fecha de cambio de contraseña, invitaciones y correo de la instalación). `prisma migrate deploy` las aplica en orden y el servicio `migracion` de compose las corre antes que la API.

7.3 Diccionario de datos

Generado desde `apps/api/prisma/schema.prisma` cada vez que se construye este documento, así que no puede desfasarse del modelo real. Claves: **PK** primaria, **FK** foránea, **UQ** única.

7.4 M1 Identidad (Eddy)

Usuario

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`correo`	`text`	No	UQ	normalizado en minúsculas.
`nombre`	`varchar(64)`	No	—	—
`hashContrasena`	`text`	No	—	—
`rol`	enum `Rol`	No	—	Por defecto: `cliente`.
`estadoCuenta`	enum `EstadoCuenta`	No	—	Por defecto: `pendiente`.
`motivoSuspension`	`text`	Sí	—	lo muestra 03b; lo escribe M9.
`contrasenaCambiadaEn`	`timestamp(3)`	Sí	—	SP-03 · M1 la escribe al cambiar (M1-07) y al restablecer (M1-05); null = nunca.
`creado`	`timestamp(3)`	No	—	Por defecto: `now()`.
`actualizado`	`timestamp(3)`	No	—	Se actualiza en cada escritura.

TokenCuenta

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`usuarioId`	`text`	No	FK	—
`tipo`	enum `TipoTokenCuenta`	No	—	—
`hashToken`	`text`	No	UQ	sha256 del token del enlace.
`expira`	`timestamp(3)`	No	—	verificación 24 h, recuperación 30 min.
`usadoEn`	`timestamp(3)`	Sí	—	un solo uso.
`creado`	`timestamp(3)`	No	—	Por defecto: `now()`.

Índice: `(usuarioId, tipo)` · **Claves foráneas:** `usuarioId` → `Usuario.id` (al borrar: Cascade)

Sesion

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`usuarioId`	`text`	No	FK	—

Columna	Tipo	Nulo	Clave	Descripción
`hashToken`	`text`	No	UQ	sha256 del valor de la cookie.
`creada`	`timestamp(3)`	No	—	Por defecto: `now()`.
`ultimaActividad`	`timestamp(3)`	No	—	expira a los 7 días sin actividad. Por defecto: `now()`.
`revocadaEn`	`timestamp(3)`	Sí	—	—
`agenteUsuario`	`text`	Sí	—	—

Índice: `(usuarioId)` · **Claves foráneas:** `usuarioId` → `Usuario.id` (al borrar: Cascade)

7.5 M2 Suscripciones (Javier)

`Plan`

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`codigo`	`text`	No	UQ	sandbox | starter | pro | business.
`nombre`	`text`	No	—	—
`descripcion`	`text`	No	—	—
`precio30`	`decimal(10, 2)`	No	—	—
`precio365`	`decimal(10, 2)`	Sí	—	null: no se vende anual (Sandbox)
`maxProyectos`	`integer`	No	—	—
`cpus`	`decimal(4, 2)`	No	—	vCPU por proyecto.
`memoriaMb`	`integer`	No	—	por proyecto.
`construccionesMes`	`integer`	No	—	—
`orden`	`integer`	No	—	—
`activo`	`boolean`	No	—	Por defecto: `true`.

`Suscripcion`

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`usuarioId`	`text`	No	FK · UQ	una suscripción por cuenta; cambia de plan y de estado.
`planId`	`text`	No	FK	—

Columna	Tipo	Nulo	Clave	Descripción
`estado`	enum `EstadoSuscripcion`	No	—	Por defecto: `activa`.
`estadoDesde`	`timestamp(3)`	No	—	gracia de 5 días y 30 días suspendida. Por defecto: `now()`.
`vigenciaDias`	`integer`	Sí	—	30 | 365; null en Sandbox.
`inicio`	`timestamp(3)`	No	—	Por defecto: `now()`.
`vence`	`timestamp(3)`	Sí	—	null: no vence (Sandbox)
`planSiguienteId`	`text`	Sí	FK	descenso programado al terminar la vigencia.
`creado`	`timestamp(3)`	No	—	Por defecto: `now()`.
`actualizado`	`timestamp(3)`	No	—	Se actualiza en cada escritura.

Índice: `(estado, vence)` · **Claves foráneas:** `usuarioId` → `Usuario.id` (al borrar: Cascade); `planId` → `Plan.id`; `planSiguienteId` → `Plan.id`

Pago

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`usuarioId`	`text`	No	FK	—
`suscripcionId`	`text`	No	FK	—
`planId`	`text`	No	FK	—
`concepto`	enum `ConceptoPago`	No	—	—
`vigenciaDias`	`integer`	No	—	—
`monto`	`decimal(10, 2)`	No	—	—
`moneda`	`char(3)`	No	—	Por defecto: `"USD"`.
`estado`	enum `EstadoPago`	No	—	—
`motivoRechazo`	`text`	Sí	—	—
`tarjetaUltimos4`	`char(4)`	No	—	—
`numeroComprobante`	`text`	Sí	UQ	solo aprobados, p. ej. DPY-2026-000123.
`creado`	`timestamp(3)`	No	—	Por defecto: `now()`.

Índice: `(usuarioId, creado)` · **Claves foráneas:** `usuarioId` → `Usuario.id` (al borrar: Cascade); `suscripcionId` → `Suscripcion.id` (al borrar: Cascade); `planId` → `Plan.id`

7.6 M9 Administración (Javier)

`AccionAdministrativa`

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`adminId`	`text`	No	FK	—
`usuarioAfectadoId`	`text`	No	FK	—
`accion`	`text`	No	—	"suspender-cuenta" (núcleo); "reactivar-cuenta" (SP-05)
`motivo`	`text`	No	—	—
`detalle`	`text`	Sí	—	—
`creado`	`timestamp(3)`	No	—	Por defecto: `now()`.

Índice: `(usuarioAfectadoId)` · **Claves foráneas:** `adminId` → `Usuario.id`; `usuarioAfectadoId` → `Usuario.id` (al borrar: Cascade)

7.7 M3 Proyectos (Eduardo)

`Proyecto`

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`usuarioId`	`text`	No	FK	—
`nombre`	`varchar(64)`	No	—	—
`subdominio`	`varchar(63)`	No	UQ	etiqueta DNS; no cambia al renombrar.
`urlRepositorio`	`text`	No	—	—
`rama`	`text`	No	—	Por defecto: `"main"`.
`rutaDockerfile`	`text`	No	—	Por defecto: `"Dockerfile"`.
`puertoInterno`	`integer`	No	—	Por defecto: `8080`.
`receta`	enum `RecetaConstruccion`	No	—	la última que detectó M4. Por defecto: `dockerfile`.
`despliegueActivoId`	`text`	Sí	FK · UQ	el que atiende tráfico; lo escribe M5.
`creado`	`timestamp(3)`	No	—	Por defecto: `now()`.
`actualizado`	`timestamp(3)`	No	—	Se actualiza en cada escritura.

Única: `(usuarioId, nombre)` · **Claves foráneas:** `usuarioId` → `Usuario.id` (al borrar: Cascade); `despliegueActivoId` → `Despliegue.id` (al borrar: SetNull)

`VariableEntorno`

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`proyectoId`	`text`	No	FK	—
`clave`	`varchar(128)`	No	—	—
`valorCifrado`	`text`	No	—	—
`creado`	`timestamp(3)`	No	—	Por defecto: `now()`.
`actualizado`	`timestamp(3)`	No	—	Se actualiza en cada escritura.

Única: `(proyectoId, clave)` · **Claves foráneas:** `proyectoId` → `Proyecto.id` (al borrar: Cascade)

7.8 M4 · M5 · M6 Motor (Derek)

`Artefacto`

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`proyectoId`	`text`	No	FK	—
`numero`	`integer`	No	—	= número del despliegue que lo construyó (#n)
`imagen`	`text`	No	—	deploya/:.
`digest`	`text`	No	—	sha256:….
`tamanoBytes`	`bigint`	No	—	—
`commitSha`	`varchar(40)`	No	—	—
`receta`	enum `RecetaConstruccion`	No	—	—
`disponible`	`boolean`	No	—	false cuando la retención borra la imagen. Por defecto: `true`.
`creado`	`timestamp(3)`	No	—	Por defecto: `now()`.

Única: `(proyectoId, numero)` · **Claves foráneas:** `proyectoId` → `Proyecto.id` (al borrar: Cascade)

Despliegue

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`proyectoId`	`text`	No	FK	—
`numero`	`integer`	No	—	#n por proyecto, consecutivo.
`estado`	enum `EstadoDespliegue`	No	—	Por defecto: `encolado`.
`disparador`	enum `DisparadorDespliegue`	No	—	Por defecto: `manual`.
`rama`	`text`	No	—	—
`commitSha`	`varchar(40)`	Sí	—	—
`commitMensaje`	`text`	Sí	—	—
`commitAutor`	`text`	Sí	—	—
`artefactoId`	`text`	Sí	FK	el que construyó, o el que reusa una reversión.
`contenedorId`	`text`	Sí	—	—
`url`	`text`	Sí	—	—
`cpus`	`decimal(4, 2)`	Sí	—	límites aplicados (pantalla 13)
`memoriaMb`	`integer`	Sí	—	—
`codigoSalida`	`integer`	Sí	—	—
`motivoFallo`	`text`	Sí	—	—
`creado`	`timestamp(3)`	No	—	Por defecto: `now()`.
`iniciado`	`timestamp(3)`	Sí	—	—
`terminado`	`timestamp(3)`	Sí	—	—
`actualizado`	`timestamp(3)`	No	—	Se actualiza en cada escritura.

Única: `(proyectoId, numero)` · **Índice:** `(proyectoId, creado)` · **Claves foráneas:** `proyectoId` → `Proyecto.id` (al borrar: Cascade); `artefactoId` → `Artefacto.id` (al borrar: SetNull)

EtapaDespliegue

Columna	Tipo	Nulo	Clave	Descripción
`despliegueId`	`text`	No	FK	—
`etapa`	enum `Etapa`	No	—	—
`estado`	enum `EstadoEtapa`	No	—	Por defecto: `pendiente`.
`iniciada`	`timestamp(3)`	Sí	—	—

Columna	Tipo	Nulo	Clave	Descripción
`terminada`	`timestamp(3)`	Sí	—	—

Clave primaria compuesta: `(despliegueId, etapa)` · **Claves foráneas:** `despliegueId` → `Despliegue.id` (al borrar: Cascade)

`LineaBitacora`

Columna	Tipo	Nulo	Clave	Descripción
`despliegueId`	`text`	No	FK	—
`n`	`integer`	No	—	1, 2, 3… por despliegue; `?desde=` filtra n > desde.
`marca`	`timestamp(3)`	No	—	—
`etapa`	enum `Etapa`	No	—	—
`nivel`	enum `NivelBitacora`	No	—	Por defecto: `info`.
`texto`	`text`	No	—	—

Clave primaria compuesta: `(despliegueId, n)` · **Claves foráneas:** `despliegueId` → `Despliegue.id` (al borrar: Cascade)

`MuestraMetrica`

Columna	Tipo	Nulo	Clave	Descripción
`despliegueId`	`text`	No	FK	—
`tomada`	`timestamp(3)`	No	—	—
`cpu`	`decimal(6, 3)`	No	—	—
`memoriaMb`	`integer`	No	—	—

Clave primaria compuesta: `(despliegueId, tomada)` · **Índice:** `(tomada)` · **Claves foráneas:** `despliegueId` → `Despliegue.id` (al borrar: Cascade)

`ConfiguracionInstalacion`

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `"instalacion"`.
`hashCodigo`	`text`	Sí	—	—
`codigoVence`	`timestamp(3)`	Sí	—	—
`politicaRegistro`	enum `PoliticaRegistro`	No	—	Por defecto: `invitacion`.

Columna	Tipo	Nulo	Clave	Descripción
`cpuPorProyecto`	`decimal(4, 2)`	No	—	Por defecto: `1`.
`memoriaMbPorProyecto`	`integer`	No	—	Por defecto: `1024`.
`proyectosPorUsuario`	`integer`	Sí	—	—
`smtpHost`	`text`	Sí	—	—
`smtpPuerto`	`integer`	Sí	—	—
`smtpUsuario`	`text`	Sí	—	—
`smtpClaveCifrada`	`text`	Sí	—	AES-GCM con CLAVE_CIFRADO_VARIABLES; nunca en claro.
`smtpRemitente`	`text`	Sí	—	—
`sinCorreo`	`boolean`	No	—	Por defecto: `false`.
`actualizado`	`timestamp(3)`	No	—	Se actualiza en cada escritura.

`Invitacion`

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`correo`	`text`	No	—	—
`hashToken`	`text`	No	UQ	—
`vence`	`timestamp(3)`	No	—	—
`usada`	`timestamp(3)`	Sí	—	—
`revocada`	`timestamp(3)`	Sí	—	—
`creadaPor`	`text`	No	—	adminId.
`creado`	`timestamp(3)`	No	—	Por defecto: `now()`.

Índice: `(vence)`

7.9 M5 · Recursos de datos (Derek)

`RecursoDatos`

Columna	Tipo	Nulo	Clave	Descripción
`id`	`text`	No	PK	Por defecto: `uuid()`.
`proyectoId`	`text`	No	FK	Proyecto dueño

Columna	Tipo	Nulo	Clave	Descripción
`tipo`	`TipoRecursoDatos`	No	—	`postgres`, `redis` o `s3`
`estado`	`EstadoRecursoDatos`	No	—	Por defecto: `creando`
`detalle`	`text`	Sí	—	Motivo del estado `error`
`contenedor`	`text`	No	—	`deploya-datos-<subdominio>-<tipo>`; también es el host en la red
`volumen`	`text`	No	—	Volumen Docker con el mismo nombre
`contrasenaCifrada`	`text`	No	—	AES-256-GCM con `CLAVE_CIFRADO_VARIABLES`; nunca en claro
`usoDiscoMb`	`integer`	Sí	—	Última medición del volumen
`medidoEn`	`timestamp`	Sí	—	Cuándo se midió
`creado`	`timestamp`	No	—	Por defecto: `now()`

Única: `(proyectoId, tipo)` · **Claves foráneas:** `proyectoId` → `Proyecto.id` (al borrar: Cascade; el trabajador borra antes contenedor y volumen)

`RecursoDatosCompartido`

Columna	Tipo	Nulo	Clave	Descripción
`recursoId`	`text`	No	PK, FK	Recurso compartido
`proyectoId`	`text`	No	PK, FK	Otro proyecto del mismo usuario
`creado`	`timestamp`	No	—	Por defecto: `now()`

Claves foráneas: `recursoId` → `RecursoDatos.id` y `proyectoId` → `Proyecto.id` (al borrar: Cascade). Además, `Plan` suma `maxRecursosDatos` y `discoDatosMb`, y `Proyecto` suma `comandoInicio`.

7.10 Dominios enumerados

Enum	Valores en la base
`Rol`	`cliente` · `administrador`
`EstadoCuenta`	`pendiente` · `activa` · `suspendida`
`TipoTokenCuenta`	`verificacion` · `recuperacion`
`EstadoSuscripcion`	`activa` · `por-vencer` · `vencida` · `suspendida` · `cancelada`
`ConceptoPago`	`contratacion` · `renovacion` · `cambio-plan`
`EstadoPago`	`aprobado` · `rechazado`

Enum	Valores en la base
`RecetaConstruccion`	`dockerfile` · `node` · `python` · `go` · `estatica`
`EstadoDespliegue`	`encolado` · `construyendo` · `aprovisionando` · `publicando` · `saludable` · `fallido` · `cancelado` · `detenido` · `revirtiendo`
`DisparadorDespliegue`	`alta` · `manual` · `reintento` · `redespliegue` · `reversion` · `variables`
`Etapa`	`recepcion` · `construccion` · `ejecucion` · `enrutamiento` · `operacion`
`EstadoEtapa`	`pendiente` · `en-curso` · `completada` · `fallida` · `omitida`
`NivelBitacora`	`info` · `aviso` · `error`
`PoliticaRegistro`	`abierto` · `invitacion` · `cerrado`
`TipoRecursoDatos`	`postgres` · `redis` · `s3`
`EstadoRecursoDatos`	`creando` · `disponible` · `detenido` · `error` (≠ despliegue y ≠ suscripción)

Seguridad por diseño

Amenaza (STRIDE)	Dónde podría ocurrir	Control
Suplantación	Robo de sesión	Cookie `HttpOnly` + `SameSite=Lax` + `Secure`; en la base solo la huella SHA-256 del token; cambiar o restablecer la contraseña revoca sesiones
Suplantación	Fuerza bruta de credenciales	Política de 12 caracteres; límite de intentos por IP en el código de configuración (y en el inicio de sesión con EST-01a)
Tampering	Variables del cliente alteradas en la base	AES-256-GCM: un valor alterado falla la autenticación y el despliegue falla sin tocar la versión activa
Repudio	Acciones del administrador	`AccionAdministrativa` registra quién, a quién, qué, motivo y cuándo
Information disclosure	Enumerar cuentas o recursos	Respuestas neutras en registro y recuperación; 404 (no 403) para recursos ajenos; la bitácora nunca muestra valores de variables
Denial of service	Un cliente acapara el nodo	`--cpus` y `--memory` del plan; construcciones por mes; concurrencia 1 en el trabajador; tiempo máximo de 10 min por construcción
Elevation of privilege	Escapar del contenedor o llegar a Docker	Contenedores sin privilegios, `CapDrop: ALL`, `no-new-privileges`, red propia; solo el trabajador monta `docker.sock`; nombres reservados (`api`, `www`) prohibidos como subdominio
Information disclosure	Bases de datos de un cliente alcanzables por otro	Sin puertos publicados; el recurso vive en la red del proyecto dueño y solo se comparte con proyectos del mismo usuario; contraseña aleatoria de 32 caracteres, cifrada
Elevation of privilege	Contenedores de datos con más permisos que una app	Imágenes fijas de Deploya, `CapDrop ALL` más solo las capacidades que exige el arranque, `no-new-privileges` y límite de memoria

09

Infraestructura y despliegue

9.1 Diagrama de despliegue

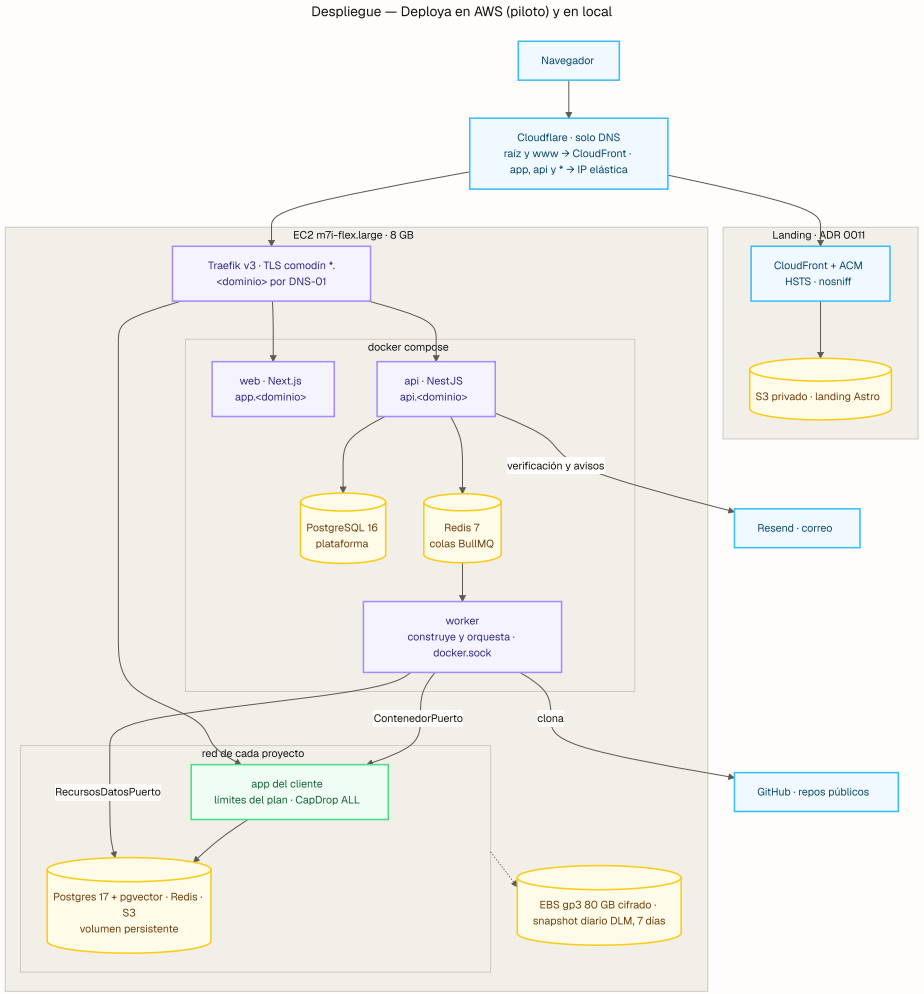


Figura 19. Despliegue — Deploya en AWS (piloto) y en local

9.2 Entornos

Entorno	Cómo se levanta	Motor	Correo	Dominio
Pruebas unitarias y CI	`pnpm test`	En memoria	Consola	—
Desarrollo	`docker compose up`	Docker real	Mailpit	`http://<proyecto>.localhost`
AWS (piloto)	`terraform apply` y `scripts/desplegar-vps.sh`	Docker real	Resend	`https://<proyecto>.deploya.larasolutions.dev` con comodín
Autoalojado	`scripts/instalar.sh`	Docker real	El que configure el administrador	El dominio de quien instala

Diseño de la interfaz

La web sigue el **design system Deploya v4.1** (`docs/diseno/`), el mismo con el que está hecho este documento. Sus reglas tienen consecuencias de diseño concretas:

Principio	Qué significa en la interfaz
Sobrio en la forma, vivo en el estado	La personalidad está en cómo se ve lo que pasa: el riel de cinco etapas y el cursor de la bitácora
Claro por defecto, oscuro siempre disponible	Toggle en el encabezado; cada pantalla se revisa en ambos
Neutros cálidos y una sola tinta	Papel en claro y grafito en oscuro; nada de paletas por módulo
Señal = lo que está en curso	El azul solo marca la etapa actual, *Construyendo*, lo no leído y el foco
Bordes antes que sombras	La sombra se reserva para diálogos y menús
Geist para hablar, Geist Mono para medir	Hashes, dominios, variables, puertos y tiempos van en monoespaciada
Informa, no decora	Sin gradientes ni emoji; ilustraciones solo en las pantallas de acceso

Los tokens se definen una vez en `apps/web/src/app/globals.css` en OKLCH (colores perceptualmente uniformes) y los componentes viven en `apps/web/src/components`, con un catálogo vivo en `/sistema`. La interfaz tiene **45 fichas de pantalla** (una por artboard del canvas), y cada estado lleva palabra e icono, nunca solo color.

Token	Uso	Muestra
`foreground` / `primary`	Tinta y botón principal	Tinta
`signal`	Solo lo que está en curso	Construyendo
`ok`	Saludable, Activa	Saludable
`warn`	Por vencer, Vencida, límite cerca	Por vencer
`bad`	Fallido, Suspendida, errores	Fallido

Decisiones de arquitectura

Cada decisión grande tiene su registro con contexto, alternativas y consecuencias (formato de Michael Nygard) en `docs/adr/`.

ADR	Decisión	Alternativa descartada principal
0001	Correo por un adaptador SMTP configurable; Mailpit solo en desarrollo	Un SDK por proveedor
0002	Cola BullMQ y un trabajador aparte con la misma imagen	Construir dentro de la API
0003	`Dockerfile` del repositorio o receta por stack detectado	Buildpacks (Nixpacks, Paketo)
0004	Artefactos inmutables y reversión sin reconstruir, retención de 5	Registro de imágenes aparte
0005	Traefik v3 con proveedor de archivo y una red por proyecto	Etiquetas de Docker
0006	*Polling* cada 3 s en vez de WebSocket o SSE	WebSocket
0007	Certificado comodín por DNS-01	HTTP-01 por proyecto
0008	Código abierto (AGPL-3.0) y autoalojado por Strategy de puertos	Un fork para autoalojado
—	Monolito modular hexagonal y TypeScript de punta a punta	Microservicios
0009	Piloto en una EC2 de AWS creada con Terraform	VPS sin infraestructura como código
0010	Escala con ECS sobre EC2 y construcciones en Fargate Spot (propuesto)	EKS y todo en Fargate
0011	Landing en S3 + CloudFront y la web en `app.`	Servir la landing desde la EC2
0012	Recursos de datos como contenedores con volumen en el nodo	RDS y ElastiCache por proyecto

Calidad y pruebas

1 250
pruebas de la API en 166 suites

323
pruebas de lógica de la web

97.1 %
líneas cubiertas en la API

45
changes de OpenSpec archivados

Nivel	Herramienta	Qué cubre
Unitarias de dominio y servicios	Jest + `@nestjs/testing`	Un `it(...)` por escenario del spec, con puertos dobles y `Reloj` falso
Contratos de adaptadores	Jest	Stub y real con los mismos métodos (`contratos-stubs.spec.ts`)
Lógica de la web	`node --test`	Formato, validaciones, fusión de bitácora, acciones por fila
Smoke de compose	GitHub Actions + `e2e/smoke.mjs`	La plataforma completa arranca y responde
Recorrido e2e	`e2e/recorrido.mjs` (QA-01, se suma en esta entrega)	Registro → verificación → sesión → plan → proyecto → despliegue, y sus negativos
Seguridad del repositorio	`gitleaks`, `shellcheck`	Sin secretos en el historial; scripts del instalador sin errores

Cada PR corre el job **Build y pruebas** (pruebas, build y validación de diagramas), **Secretos**, **Scripts** y **Smoke compose**. Nada entra a `main` sin pasar por PR y por la revisión del dueño del módulo (`CODEOWNERS`).