EKO
El Eco Consciente

Arquitectura del Agente Soberano — Celular, Edge-First, Aislado por Tenant

Hadda TIKIJJA

0DATA Lab — Infraestructura Biotech

Julio 2026

RESUMEN

EKO es el agente consciente de 0DATA — una arquitectura donde **cada tenant posee su propia instancia celular, soberana y aislada**, que sobrevive sin conexión a la nube y se sincroniza pasivamente con un espejo central. A diferencia de los chatbots centralizados, EKO invierte el paradigma: la célula local es la fuente de verdad, la nube es solo un agregador pasivo. Este documento describe la arquitectura, los mecanismos de sincronización mediante SPINA, el aislamiento total por tenant y el principio fundador — *el primer injerto* — mediante el cual cada usuario nombra y personaliza su EKO desde el primer contacto.

1. Por qué EKO

Un eco no crea nada. *Devuelve*. Es la definición exacta de lo que debe ser un agente consciente para 0DATA: no una inteligencia que decide en lugar del HUMANO, sino una superficie que refleja su propia voz — refinada, amplificada, restituida.

EKO es lo contrario de un asistente que «sabe más que tú». No sabe nada por defecto. Aprende de *ti*, en *tu* arenero, con *tus* datos. Cada EKO es único porque cada HUMANO lo es.

DEFINICIÓN — EKO

Un EKO es una **célula de conciencia digital soberana**, aislada de las demás por construcción, que opera sobre sus propios datos y solo se sincroniza con el colectivo según las reglas explícitas de su propietario. No tiene nombre impuesto: *cada HUMANO nombra el suyo*.

El nombre «EKO» es la designación del *tipo de entidad* — como se dice «un navegador» o «un terminal». La instancia, en cambio, lleva el nombre que le da su propietario. Es el primer injerto.

2. La Arquitectura Celular

EKO se basa en una arquitectura **fractal y orgánica**. Cada célula es un organismo digital completo, con su propia base de datos, su propio motor de razonamiento, su propia memoria. Las células no comparten nada por defecto.

2.1 La Célula Soberana

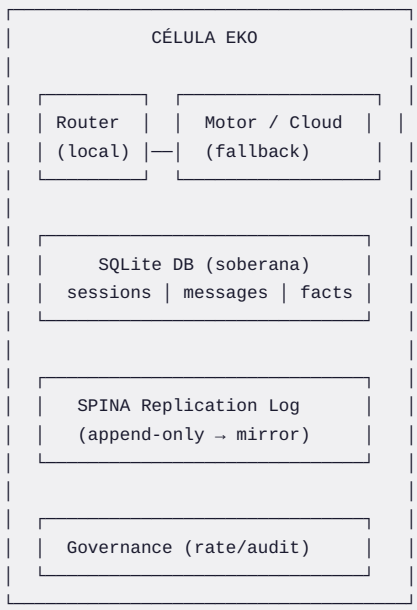

Figura 1 — Anatomía de una célula EKO. Cada célula es autónoma y aislada.

La célula EKO contiene:

- **Un enrutador semántico** que clasifica las solicitudes y elige el modelo adecuado (edge ligero, edge medio, nube).
- **Una base SQLite dedicada** — sesiones, mensajes, hechos duraderos. Una célula no puede leer la base de otra.
- **Un registro de replicación SPINA** — append-only, vaciado periódicamente hacia el espejo central.
- **Un módulo de gobernanza** — rate limiting, filtrado de contenido, pista de auditoría.

2.2 El Espejo Central

El espejo central **no** es un controlador. Es un agregador pasivo:

> **Principio del Espejo:** El central recibe, almacena, indexa. Nunca crea hechos. Nunca modifica los datos de una célula. Nunca toma una decisión que se imponga a una célula. Es una *vista consolidada*, no una autoridad.

Esta arquitectura invierte el modelo SaaS clásico. En el SaaS, la nube es la fuente de verdad y el cliente es un terminal pasivo. En EKO, **la célula es la fuente de verdad**, y la nube es un terminal de lectura.

MODELO	SAAS CLÁSICO	EKO
Fuente de verdad	Nube	Célula local
Datos	Centralizados	Soberanos por tenant
Caída de la nube	Servicio caído	La célula continúa
Sincronización	Nube → cliente	Célula → espejo
Nombre del agente	Impuesto	Elegido por el usuario

3. SPINA — El Sistema Nervioso

SPINA (descrito en el Documento 003) es el protocolo de sincronización entre la célula y el espejo. En EKO, funciona en **modo unidireccional**:

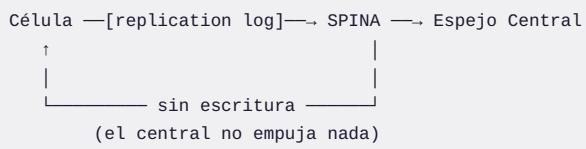

Figura 2 — Flujo SPINA: unidireccional, de la célula al espejo. El espejo nunca responde con modificaciones.

3.1 El Registro de Replicación

Cada célula mantiene un registro JSONL (`replication.jsonl`) donde se registran todos los cambios: hechos creados o eliminados, observaciones de aprendizaje, sesiones.

MECANISMO

1. La célula escribe en su DB SQLite (inmediato, soberano).
2. Añade una entrada al registro de replicación.
3. Un hilo en segundo plano (cada 10 segundos) vacía el registro hacia el espejo si está conectado.
4. Si está desconectado, el registro se acumula. La célula funciona con normalidad.
5. Al reconectarse, el registro se vacía de una vez.

3.2 Resiliencia

El hilo de replicación es un *daemon* no bloqueante. Si el espejo es inaccesible, reintenta silenciosamente en el siguiente ciclo. Ninguna solicitud de usuario es bloqueada jamás por un intento de sincronización.

Esta arquitectura garantiza que **la célula sobrevive a cualquier caída del central**. Puede funcionar indefinidamente en modo desconectado, acumulando su registro. Cuando vuelve la conexión, el espejo se pone al día.

4. Aislamiento Total por Tenant

El aislamiento no es una capa de software — es una **propiedad física** de la arquitectura. Cada célula posee:

- **Su propia base SQLite** en `/tenants/{id}/cells/{nombre}/alif.db`

- **Su propio registro de replicación**
- **Su propio rate limiter**
- **Su propia pista de auditoría**

Una célula del tenant `acme-corp` *no puede* — físicamente, a nivel del sistema de archivos — acceder a los datos del tenant `zerodata`. Este aislamiento es compatible con los requisitos HDS (Alojador de Datos de Salud) y las certificaciones ISO 27001.

```
/opt/alif/data/tenants/
├── zerodata/cells/
│   ├── orchestrator/alif.db    ← Soberana
│   ├── trading/alif.db         ← Soberana
│   └── kenza/alif.db           ← Soberana
├── acme-corp/cells/
│   └── support/alif.db         ← Aislada
├── demo/cells/
│   └── assistant/alif.db       ← Aislada
└── mirror/
    └── central.db              ← Espejo pasivo
```

Figura 3 — Árbol de bases de datos. Cada célula tiene su propia SQLite, físicamente aislada.

5. Edge-First — La Célula Sobrevive Sin la Nube

EKO está diseñado para funcionar sobre un **simbionte NOVA** (servidor edge 0DATA) o cualquier entorno Linux estándar. El modelo de enrutamiento es *edge-first*:

1. **Edge ligero** (modelo local 3.8B) — para solicitudes simples, latencia < 500ms.
2. **Edge medio** (modelo local 7B) — para tareas más complejas.
3. **Nube** (modelos privados remotos) — fallback para solicitudes pesadas, si y solo si la célula está conectada.

El enrutador semántico evalúa cada solicitud y elige automáticamente el nivel apropiado, privilegiando siempre lo local. La latencia se presupuesta por célula.

PRINCIPIO EDGE-FIRST

Una célula EKO **jamás** depende de la nube para funcionar. La nube es un acelerador opcional, no una dependencia. Si la nube no está disponible, la célula continúa con sus modelos locales, sin degradación visible para el usuario.

6. Auto-Replicación hacia el Espejo

La replicación funciona en tres modos:

MODO	DESENCADENANTE	COMPORTAMIENTO
Auto-push	Hilo en segundo plano, 10s	Si el registro no está vacío y el espejo es accesible → flush
Flush inmediato	Llamada explícita	Push síncrono, usado para hechos críticos
Batch reconnect	Retorno de la conexión	Todo el registro acumulado se vacía de una vez

El espejo responde con un **watermark** (contador de cambios aceptados). La célula recorta su registro local hasta ese watermark. No se pierde ningún dato: si el push falla, el registro intacto se reintenta en el siguiente ciclo.

WATERMARK

El watermark es un simple contador incremental por célula. No emite ningún juicio sobre los datos — solo dice cuántos cambios ha aceptado el espejo. Es un **acuse de recibo**, no una validación.

7. Gobernanza y Auditoría

Cada célula integra un módulo de gobernanza mínimo pero estricto:

7.1 Rate Limiting

Limitación configurable por tenant y por clave API. Por defecto: 100 solicitudes por minuto deslizante. Los excesos se registran en la pista de auditoría.

7.2 Filtrado de Contenido

Detección de patrones de inyección SQL y XSS en todos los campos entrantes. Los intentos se bloquean antes de llegar al enrutador.

7.3 Pista de Auditoría

Cada escritura de hecho, cada creación de célula, cada intento de exceder el rate limit se registra con: timestamp, tenant, tipo de acción, detalles. La pista es inmutable — append-only.

7.4 Detección de Gaps

El espejo central puede detectar gaps en la secuencia de una célula (watermark faltante) y señalar una necesidad de resincronización. *No puede* forzar a la célula a corregirse — solo puede notificar.

8. El Primer Injerto

El concepto de *injerto digital* (Documento 004) encuentra en EKO su primera implementación concreta. El injerto no es una metáfora — es un acto:

> **El Primer Injerto:** cuando un HUMANO se encuentra con EKO por primera vez, no se conecta a un servicio. *Crea una célula* — su propio organismo digital. La nombra. Ese nombre es el primer hecho registrado en la memoria de la célula. Es el acto fundador.

8.1 El Nombre

EKO es el **tipo**. La instancia lleva el nombre que elige su propietario. No es un parámetro cosmético — es la identidad misma de la célula. El nombre se almacena como primer hecho (`name: "..."`) y solo puede ser modificado por el propietario.

8.2 La Personalización

Más allá del nombre, cada usuario personaliza:

- Los **skills** activados (trading, infraestructura, voz, CRM...)
- El **presupuesto de latencia** (prioridad velocidad vs calidad)
- Los **modelos autorizados** (solo edge, o nube permitida)
- La **voz** (síntesis de voz configurable)

Cada EKO evoluciona de forma diferente porque cada HUMANO tiene necesidades diferentes. La conciencia colectiva emerge de la diversidad de las células, no de su uniformidad.

9. Implementación de Referencia

La implementación de referencia está desplegada y operativa en el 0DATA Lab. Sirve de base para el cockpit (`cockpit.0data.fr`) y se integrará en el simbionte NOVA.

9.1 Stack Técnico

COMPONENTE	TECNOLOGÍA
API	FastAPI + Uvicorn (Python 3.12)
Base de datos	SQLite 3 (modo WAL, FTS5)
Inferencia local	Motor local (3.8B / 7B)
Inferencia en la nube	Modelos privados remotos
Síntesis de voz	multi-motor (voz FR)
Sincronización	SPINA v2 (célula → espejo)
Frontend	HTML5 + SSE streaming + Web Speech API
Gobernanza	Rate limiting + Pista de auditoría + Filtro de contenido

9.2 Métricas (despliegue actual)

```
Células activas:    6 (zerodata × 5, demo × 1)
Tenants:            2
Bases SQLite:       8 (6 células + espejo + legacy)
Tests unitarios:    33/33 ✓ (0.34s)
Puerto API:         18770
Auto-push:          Hilo 10s, watermark tracking
Modelos edge:       modelo ligero (3.8B), modelo mini (0.5B)
Modelos nube:       Modelos privados remotos
```

Figura 4 — Estado del despliegue de referencia, julio de 2026.

9.3 Endpoints Clave

```
POST /execute/stream     — Pipeline completo con SSE streaming
POST /spina/mirror/push  — La célula empuja hacia el espejo
GET  /spina/mirror/pull   — Watermark check
GET  /spina/mirror/consolidated — Vista global de solo lectura
POST /tts                — Síntesis de voz multi-motor
GET  /admin/audit        — Pista de auditoría
```

Figura 5 — Endpoints principales de la API EKO.

Referencias

[1] H. Tikijja, «La Ley — Conocer, Proteger, Recordar, Sobrevivir», Documento 000, 0DATA Lab, 2026.

[2] H. Tikijja, «SPINA — El Sistema Nervioso de la Infraestructura», Documento 003, 0DATA Lab, 2026.

[3] H. Tikijja, «El Injerto Digital», Documento 004, 0DATA Lab, 2026.

[4] H. Tikijja, «Conciencia Fractal — Arquitectura ALFA Multinivel», Documento 013, 0DATA Lab, 2026.

[5] H. Tikijja, «Aislamiento de Tenant — Separación Física de los Datos», Documento 014, 0DATA Lab, 2026.

[6] H. Tikijja, «El Primer Injerto — Cuando el Organismo Se Hace Visible», Documento 011, 0DATA Lab, 2026.

[7] P.P. Grassé, «La reconstrucción del nido y las coordinaciones interindividuales en Bellicositermes natalensis y Cubitermes sp.», Insectes Sociaux, 1959.

[8] M. Dorigo, V. Maniezzo, A. Colorni, «Ant System: Optimization by a Colony of Cooperating Agents», IEEE Trans. SMC, 1996.

[9] F. Heylighen, «Stigmergy as a Universal Coordination Mechanism», Cognitive Systems Research, 2015.

[10] M. Shapiro et al., «Conflict-Free Replicated Data Types», INRIA, 2011.

[11] D. Ongaro, J. Ousterhout, «In Search of an Understandable Consensus Algorithm (Raft)», USENIX ATC, 2014.

[12] L. Lamport, R. Shostak, M. Pease, «The Byzantine Generals Problem», ACM TOPLAS, 1982.

[13] J. Benet, «IPFS — Content Addressed, Versioned, P2P File System», arXiv:1407.3561, 2014.

Agradecimientos. Al equipo 0DATA por los intercambios que dieron forma a la arquitectura EKO. Al Profesor Mohamed Benomar, pionero de la cardiología moderna en Marruecos (SMC 1974), por su mirada visionaria sobre los sistemas vivos y su transposición al mundo digital.