

Digital Integration Hub

Arquitectura orientada a microservicios para banca

El patrón que desacopla los canales digitales del core bancario: anatomía, consistencia, casos de uso, anti-patrones y ruta de implementación. Con el primer cuerpo de referencia en español.



CAMILO BERNAL

Arquitecto de Soluciones · Banca y Servicios Financieros

Edición 1.0 · Agosto de 2026 · camilobernal.dev

Sobre este libro

Este documento nació de una constatación incómoda: **no existe literatura seria sobre Digital Integration Hub en español**. Ni un libro, ni un paper académico revisado, ni un curso. El término lo acuñó Gartner alrededor de 2019 y su corpus completo comparte tres características: está en inglés, lo produce mayoritariamente el fabricante que vende el producto, y es descriptivo antes que prescriptivo. Te dice qué es un DIH; rara vez cómo construirlo, cuándo no construirlo, y qué te va a salir mal.

Este texto intenta llenar ese vacío para el contexto que conozco: banca latinoamericana, cores heterogéneos, presupuestos reales y reguladores que no aceptan «consistencia eventual» como respuesta en una auditoría.

A quién está dirigido

A arquitectos de solución y de empresa, líderes técnicos, ingenieros de plataforma y responsables de modernización de core en instituciones financieras. Asume familiaridad con microservicios, mensajería asíncrona y el vocabulario básico de sistemas distribuidos. No asume experiencia previa con grids en memoria ni con CDC.

Qué no encontrarás aquí

Ni recomendaciones de fabricante ni comparativas de licenciamiento. Los productos se mencionan como categorías con criterios de decisión, no como veredictos. El precio, el soporte regional y la madurez de un producto cambian más rápido de lo que envejece un PDF.

Sobre el uso de inteligencia artificial

Este libro fue escrito **con asistencia de inteligencia artificial**. Lo digo al principio y no en letra pequeña, porque la alternativa —no decirlo— me parece indefendible en un texto que dedica un capítulo entero a la degradación honesta.

El reparto fue este. La concepción, el alcance, la estructura y el criterio técnico son míos, y provienen de haber visto este patrón nacer y fracasar en instalaciones reales. La redacción, la organización del material, la investigación de fuentes y la composición tipográfica se produjeron en colaboración con un modelo de lenguaje. Cada afirmación técnica, cada cifra y cada recomendación fueron revisadas y validadas por mí, y la responsabilidad editorial sobre el resultado es enteramente mía.

Menciono el método por dos razones. La primera es de higiene intelectual: si vas a construir una arquitectura bancaria apoyándote en este texto, mereces saber cómo se produjo. La segunda es que el método es parte del argumento. Un libro de casi ochenta páginas sobre un patrón sin literatura en español, producido en días en lugar de meses, es una demostración práctica de lo que estas herramientas hacen posible cuando quien las dirige sabe qué está buscando.

Notas de método

Todo dato regulatorio colombiano proviene de fuentes públicas del Banco de la República y la Superintendencia Financiera, citadas en el capítulo 25. Las cifras de negocio son órdenes de magnitud típicos de banca retail mediana en la región, no benchmarks auditados: úsalas para dimensionar una conversación, nunca para sustentar un caso de negocio sin medir tu propia instalación.

Título Digital Integration Hub: arquitectura orientada a microservicios para banca
Autor Camilo Bernal – Arquitecto de Soluciones, BFSI
Edición 1.0 – Agosto de 2026
Idioma Español (LatAm). Código y nomenclatura técnica en inglés.
Uso Documento de trabajo para evangelización técnica y diseño de arquitectura.

C o p y l e f t

Licencia



ATRIBUCIÓN — COMPARTIRIGUAL 4.0 INTERNACIONAL

Esta obra se distribuye bajo licencia **Creative Commons Atribución-CompartirIgual 4.0 Internacional (CC BY-SA 4.0)**, una licencia copyleft: la cláusula *CompartirIgual* obliga a que toda obra derivada se distribuya bajo esta misma licencia, de modo que la libertad concedida no pueda ser retirada aguas abajo.

Eres libre de

- **Compartir** — copiar y redistribuir el material en cualquier medio o formato.
- **Adaptar** — remezclar, transformar y construir a partir del material para cualquier finalidad, **incluso comercial**.

El licenciante no puede revocar estas libertades mientras cumplas los términos.

Bajo las siguientes condiciones

- **Atribución (BY)** — debes reconocer la autoría, enlazar a la licencia e indicar si realizaste cambios. Puedes hacerlo de cualquier forma razonable, pero no de manera que sugiera que el autor te respalda a ti o al uso que haces.
- **CompartirIgual (SA)** — si remezclas, transformas o construyes a partir de esta obra, debes distribuir tu contribución bajo la misma licencia que el original.
- **Sin restricciones adicionales** — no puedes aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que la licencia permite.

Cadena de atribución sugerida

«Digital Integration Hub: arquitectura orientada a microservicios para banca», por Camilo Bernal (camilobernal.dev), edición 1.0, 2026. Licenciado bajo CC BY-SA 4.0. <https://creativecommons.org/licenses/by-sa/4.0/deed.es>

Texto legal completo de la licencia: creativecommons.org/licenses/by-sa/4.0/legalcode.es

Alcance y excepciones

La licencia cubre **el texto, las tablas, los diagramas y el diseño editorial** de esta obra. No se extiende a:

- **Marcas y nombres de terceros.** BIAN, ISO 20022, SWIFT, Bre-B, y los nombres de productos y fabricantes citados pertenecen a sus respectivos titulares. Su mención en este texto es nominativa y descriptiva, con fines de identificación e interoperabilidad.
- **Normas y estándares citados.** Los documentos de BIAN, ISO 20022, del Banco de la República y de la Superintendencia Financiera se rigen por sus propias condiciones de uso. Aquí solo se describen y se referencian.
- **Los distintivos de licencia.** Los símbolos que encabezan esta página son una representación tipográfica de los identificadores **CC**, **BY** y **SA**, compuesta con la tipografía de esta obra. No son los botones oficiales de Creative Commons, cuya marca pertenece a esa organización y puede obtenerse en creativecommons.org. El símbolo copyleft que los acompaña —la «C» invertida dentro de un círculo— es de dominio público y aquí está compuesto con la tipografía de esta obra.

Sobre el código fuente de la obra

El código que genera este documento —plantillas HTML, hoja de estilos y script de composición— puede liberarse por separado bajo una licencia copyleft de software, como **GNU GPLv3** o **AGPLv3**. CC BY-SA no está diseñada para software y Creative Commons desaconseja usarla con ese fin.

VERIFICACIÓN PREVIA A LA PUBLICACIÓN

Una licencia Creative Commons es **irrevocable**: una vez publicada la obra bajo CC BY-SA, no se puede retirar esa concesión para las copias ya distribuidas. Antes de publicar, confirma que ostentas los derechos patrimoniales sobre el texto —en particular, que no se trata de una obra por encargo o producida en el marco de una relación laboral cuyo contrato asigne la titularidad al empleador—. No soy abogado y esto no es asesoría legal; es la comprobación que conviene hacer antes de firmar algo irreversible.

Contenido

APERTURA

- Prólogo: el vacío que estás leyendo 8
- Cómo leer este libro 9

PARTE I · FUNDAMENTOS

- 01 Qué es realmente un DIH 11
- 02 Qué no es un DIH 13
- 03 Por qué la banca es el caso canónico 14

PARTE II · ANATOMÍA

- 04 La arquitectura de referencia en cuatro capas 17
- 05 Capa 1 — Sincronización 19
- 06 Capa 2 — Estado 22
- 07 Capa 3 — Cómputo 24
- 08 Capa 4 — Exposición 25

PARTE III · LA CONVERSACIÓN DIFÍCIL

- 09 Taxonomía de frescura 28
- 10 Patrones de escritura 30
- 11 Garantías operacionales 32
- 12 CAP, PACELC y el gerente de riesgo 34

PARTE IV · EL DOMINIO

- 13 BIAN: el DIH como materialización de Service Domains 37
- 14 ISO 20022 como canónico del backbone 39
- 15 Caso Colombia: Bre-B, DICE y el directorio de llaves 40

PARTE V · CASOS DE USO

- 16 Los ocho casos, ordenados por retorno 43
- 17 Coexistencia y migración de core 45

PARTE VI · LO QUE SALE MAL

- 18 Catálogo de anti-patrones 48
- 19 Seguridad y cumplimiento 51

PARTE VII · EJECUCIÓN

20	Roadmap por fases	54
21	NFR y observabilidad	56
22	FinOps del DIH	58
23	Modelo operativo y organización	59

PARTE VIII · APRENDER

24	Plan de estudio de ocho semanas	61
25	Bibliografía comentada	62
26	Glosario EN→ES	64

ANEXOS

A	Plantilla de taxonomía de frescura	67
B	Checklist de revisión de arquitectura	68
C	Matriz de decisión tecnológica	70
D	Preguntas para el comité de negocio	71
E	Plantilla de ADR para decisiones de DIH	72

Apertura

Prólogo: el vacío que estás leyendo

Hay un patrón de arquitectura que la mitad de los bancos de la región ya está construyendo sin saber cómo se llama, y que la otra mitad va a necesitar en los próximos treinta y seis meses.

Lo he visto nacer tres veces con tres nombres distintos. Una vez fue «la capa de caché corporativa». Otra, «el ODS operacional». La tercera, «el proyecto de alivio del core». En los tres casos el problema era idéntico: la app móvil crecía más rápido que la capacidad del sistema de registro para responderle, la factura del mainframe subía en línea recta, y alguien en un comité preguntó por qué consultar un saldo cuesta lo que cuesta.

Gartner le puso nombre al montaje en 2019: **Digital Integration Hub**. El nombre importa menos que la disciplina que impone. Un DIH bien construido es una decisión de arquitectura sobre *qué verdad se sirve a quién y con cuánto retraso*. Un DIH mal construido es un caché distribuido que un día empieza a mentir sobre el dinero de los clientes y nadie se entera hasta que el cliente reclama.

La diferencia entre los dos no es la tecnología. He visto DIH excelentes sobre Redis y desastres sobre productos de seis cifras. La diferencia es si alguien tuvo, antes de escribir la primera línea de código, una conversación honesta con el negocio sobre tolerancia a la desactualización del dato. Ese es el capítulo 9 de este libro, y si solo vas a leer un capítulo, lee ese.

Por qué este libro está en español

Porque el noventa y cinco por ciento de las decisiones de arquitectura en banca latinoamericana se discuten en español, en comités donde la mitad de los asistentes no lee inglés técnico con fluidez, y donde el arquitecto que traduce mal un concepto de consistencia acaba de introducir un riesgo operativo que nadie va a detectar en revisión.

Hay una razón adicional, menos noble y más honesta: el vacío es la oportunidad. Nadie ha escrito el documento de referencia sobre este patrón para nuestro contexto. Ni los fabricantes, ni la academia, ni las consultoras. Escribirlo es la forma más rápida que conozco de entender algo de verdad.

UNA ADVERTENCIA SOBRE EL RESTO DE LA LITERATURA

Cuando termines este libro y vayas al material en inglés, vas a encontrar mucho contenido corporativo que describe el DIH como si fuera una categoría de producto que se compra. No lo es. Es un patrón que se construye, y la mayor parte del trabajo —modelo canónico, taxonomía de frescura, reconciliación, gobierno de esquema— no viene en ninguna licencia. Lee ese material por el vocabulario y desconfía de la promesa llave en mano.

Apertura

Cómo leer este libro

El libro está ordenado como se construye un DIH, no como se estudia. Si vas a leerlo completo, el orden natural funciona. Si tienes menos tiempo, hay tres rutas.

RUTA	CAPÍTULOS	QUÉ OBTIENES
Decisión ejecutiva ≈ 45 min	1, 2, 3, 9, 18, 20, 22	Qué es, por qué la banca lo necesita, cuánto cuesta, cómo fracasa y en qué orden se construye. Suficiente para defender o rechazar una iniciativa en comité de arquitectura.
Diseño técnico ≈ 3 h	4 a 12, 17, 21	La anatomía completa y el tratamiento de consistencia. Es el núcleo del libro y lo que necesitas para producir un documento de arquitectura defendible.
Contexto de dominio ≈ 1 h	3, 13, 14, 15, 16, 19	Cómo el patrón aterriza en BIAN, ISO 20022, pagos inmediatos y el marco regulatorio colombiano. Para quien ya sabe de arquitectura pero no de banca.

Convenciones tipográficas

- Las **cajas azules** contienen contexto que amplía el argumento principal sin ser necesario para seguirlo.
- Las **cajas ámbar** son advertencias: cosas que he visto salir mal en producción.
- Las **cajas con borde completo** contienen reglas duras. No son sugerencias; son las condiciones bajo las cuales el patrón funciona.
- La terminología técnica se mantiene en inglés cuando la traducción al español está en disputa o degrada el significado. El glosario del capítulo 26 documenta cada decisión.

SOBRE LAS CIFRAS DE ESTE LIBRO

Cada número de negocio —porcentajes de reducción de carga, latencias objetivo, volúmenes— es un orden de magnitud típico, no un benchmark. Los cito para calibrar expectativas y estructurar conversaciones. Tu instalación es distinta. **Mide antes de prometer.**

Parte I

Fundamentos

Qué es el patrón, qué no es, y por qué la banca es el único sector donde las cinco presiones que lo justifican aparecen simultáneamente.

Capítulo 01

Qué es realmente un DIH

Un Digital Integration Hub es un modelo de lectura corporativo, materializado y gobernado, que sirve consultas de alta concurrencia con latencia de decenas de milisegundos desde una copia sincronizada de los sistemas de registro, mientras las escrituras siguen respetando la autoridad transaccional del core.

Esa es la definición que uso en comités de arquitectura y la que sostiene el resto del libro. Vale la pena desarmarla pieza por pieza, porque cada palabra excluye algo.

ELEMENTO DE LA DEFINICIÓN	QUÉ IMPLICA Y QUÉ EXCLUYE
Modelo de lectura	El DIH sirve consultas. Las escrituras van a otra parte. Esto es CQRS elevado a escala organizacional. Excluye la idea de un hub que «también escribe» por defecto.
Corporativo	Trasciende una aplicación. Varios canales y varios equipos consumen la misma vista. Excluye el caché privado de un microservicio, que es una decisión local y no necesita gobierno.
Materializado	El dato está precalculado y desnormalizado en la forma en que se consume, no computado al vuelo. Excluye la capa de agregación sin estado —un BFF que hace fan-out a seis servicios no es un DIH.
Gobernado	Tiene modelo canónico versionado, dueño, SLA de frescura publicado y reconciliación. Excluye el caché improvisado, que es exactamente el mismo montaje sin ninguna de estas cuatro cosas.
Alta concurrencia	Dimensionado para el pico del canal digital, no para el promedio del core. Es la razón económica del patrón.
Copia sincronizada	El dato es derivado. Siempre existe una fuente autoritativa en otro lado, y siempre es reconstruible. Excluye que el DIH sea sistema de registro de nada.
Autoridad transaccional	El dinero se mueve donde está el libro mayor. El DIH refleja el movimiento; no lo decide. Excluye el write-behind sobre datos financieros.

El linaje del patrón

El DIH no es una invención. Es la convergencia de cinco líneas que llevaban dos décadas madurando por separado, y su valor está precisamente en el ensamblaje.

LÍNEA	ORIGEN	APORTACIÓN AL DIH
CQRS	Greg Young, ~2010	La separación conceptual entre el modelo que escribe y el modelo que lee, con esquemas distintos optimizados para cada lado.
Vista materializada	Bases de datos, años 80	La idea de precalcular y persistir el resultado de una consulta costosa. El DIH la lleva del ámbito de una base al ámbito de una organización.
In-Memory Data Grid	Finales de los 90	El sustrato técnico: almacenamiento distribuido en memoria con particionamiento, réplicas y cómputo colocado.
Change Data Capture	Replicación de bases, años 90	El mecanismo de sincronización de bajo impacto que hace viable copiar el core sin tocarlo ni pedirle permiso.
Operational Data Store	Data warehousing, años 90	El antecedente directo. El ODS ya agregaba datos operacionales de varios sistemas; le faltaban la latencia, la elasticidad y el enfoque de API.

Si llevas veinte años en esto, la sensación de *déjà vu* es correcta y no invalida el patrón. Lo nuevo no son las piezas: es que el volumen del canal digital y el modelo de costo de la nube hicieron económicamente obligatorio un montaje que antes era un lujo de banca de inversión.

NOTA TERMINOLÓGICA

Vas a encontrar el mismo patrón bajo otros nombres según quién lo venda: *operational data hub*, *real-time data layer*, *data access layer*, *smart cache*, *API data fabric*. Cuando alguien te presente uno de estos, hazle las cuatro preguntas de gobierno: ¿cuál es el modelo canónico?, ¿quién es el dueño?, ¿cuál es el SLA de frescura publicado?, ¿cómo se reconcilia contra el sistema de registro? Si no hay respuesta a las cuatro, te están vendiendo un caché con otro nombre.

Capítulo 02

Qué no es un DIH

La forma más rápida de entender el patrón es delimitarlo contra las seis cosas con las que se confunde. Cada confusión produce un modo de fallo distinto y predecible.

NO ES	POR QUÉ SE CONFUNDE	DIFERENCIA ESENCIAL Y MODO DE FALLO
ESB	Ambos «integran» sistemas y se dibujan en el centro del diagrama.	El ESB es <i>sin estado</i> y orquesta; el DIH es <i>con estado</i> y materializa. Fallo: si construyes el DIH con mentalidad de ESB acabas poniendo lógica de negocio y transformación en la capa central, y reinventas el cuello de botella organizacional que los microservicios vinieron a eliminar.
API Gateway	Ambos están delante de los sistemas y los canales les hablan a ellos.	El gateway aplica política, autenticación y enrutamiento en el borde; no tiene el dato. El DIH tiene el dato. Fallo: confundirlos lleva a activar caché en el gateway y llamarlo DIH, sin modelo canónico ni invalidación coherente.
iPaaS	Ambos conectan sistemas heterogéneos con conectores.	El iPaaS mueve datos <i>entre</i> aplicaciones en flujos; el DIH sirve datos a canales en consultas. Fallo: usar un iPaaS como capa de consulta produce latencias de cientos de milisegundos y costos por transacción que escalan linealmente con el tráfico.
Data Lakehouse	Ambos consolidan datos de múltiples fuentes en un lugar.	El lakehouse es analítico: optimizado para escaneos amplios con latencia de minutos u horas. El DIH es operacional: optimizado para acceso por clave con latencia de milisegundos. Fallo: servir la app móvil desde el lakehouse. Funciona en la demo y muere en el pico.
Caché distribuido	Comparten literalmente la misma tecnología subyacente.	El caché es un <i>detalle de implementación</i> del DIH. El DIH añade modelo canónico, sincronización dirigida por eventos, SLA de frescura, reconciliación y gobierno. Fallo: el más común de todos. Ver capítulo 18, anti-patrón 1.
Data Mesh	Ambos hablan de dominios y de datos como producto.	Data Mesh es un modelo <i>organizacional</i> ; el DIH es un patrón <i>técnico</i> . Son complementarios: cada dominio puede publicar su producto de datos operacional en el DIH. Fallo: tratarlos como alternativas y elegir uno.

REGLA DURA

Si tu «DIH» no tiene un documento que responda estas cuatro preguntas, no es un DIH: **¿cuál es el modelo canónico y quién lo versiona?, ¿cuál es el SLA de frescura por clase de dato?, ¿cómo se detecta y corrige la divergencia contra el sistema de registro?, ¿qué pasa cuando la sincronización se retrasa?** Es un caché, y va a fallar como fallan los cachés: silenciosamente y en el peor momento.

Capítulo 03

Por qué la banca es el caso canónico

Ningún otro sector combina las cinco presiones simultáneamente. Retail tiene picos pero no tiene mainframe. Telecom tiene volumen pero no tiene regulador de liquidez. Salud tiene regulación pero no tiene pagos inmediatos. La banca las tiene todas a la vez.

Presión 1 — La economía del mainframe

Este es el caso de negocio número uno y el único que sobrevive intacto a un comité financiero. El core bancario factura por capacidad consumida: MIPS, MSU, o el equivalente en licenciamiento del proveedor. Cada consulta de saldo desde la app móvil es dinero quemado en un dato que casi nunca cambió desde la consulta anterior.

La aritmética que uso para abrir la conversación:

Clientes retail activos digitalmente	5.000.000
Consultas de saldo / cliente / mes	12
Consultas de movimientos / cliente / mes	4

Transacciones de lectura al core / mes	80.000.000
Proporción sobre el total de operaciones	~85%
Proporción cuyo dato NO cambió desde la consulta inmediatamente anterior	~90%

Esa última línea es el argumento completo. Nueve de cada diez veces el core hace un trabajo costoso para devolver exactamente lo que devolvió la vez anterior. El DIH no reduce el trabajo del core: **elimina el trabajo redundante**, que es una categoría distinta y mucho más fácil de defender.

CÓMO PRESENTAR ESTO SIN EXAGERAR

La métrica honesta no es «ahorro» sino *reducción de transacciones de lectura enviadas al sistema de registro*. El ahorro monetario depende de tu contrato, de si el core es propio o licenciado, y de si el hardware liberado se puede desprovisionar o simplemente queda ocioso. Prometer ahorro sin haber leído el contrato de licenciamiento es la forma más rápida de perder credibilidad en la segunda revisión de presupuesto.

Presión 2 — Las ventanas batch

El core cierra. Cierre diario, conciliación, fin de mes, fin de año. Durante esas ventanas el sistema de registro está indisponible o degradado para consulta transaccional. El canal digital, en cambio, no puede cerrar: el cliente que abre la app a las 11:40 de la noche no sabe ni le importa que el banco esté conciliando.

Un DIH permite operar en modo sustitutivo durante la ventana: consulta normal, operaciones de bajo riesgo permitidas dentro de límites conservadores, y encolamiento de lo demás para procesamiento al reabrir. Es la diferencia entre una app que dice «servicio no disponible» durante noventa minutos cada noche y una que simplemente funciona.

Presión 3 — Picos no negociables

Quincena, día de pago, temporada de compras, devolución de impuestos, viernes por la tarde. El core se dimensiona para el promedio porque escalarlo es caro y lento; el canal digital sufre el pico. La relación entre pico y promedio en banca retail latinoamericana suele estar entre 4:1 y 12:1 según el día.

El DIH absorbe el pico elásticamente —añadir nodos a un grid es cuestión de minutos— sin que el core se entere de que hubo pico. Es desacoplamiento de capacidad, no solo de datos.

Presión 4 — Pagos inmediatos

Bre-B en Colombia, Pix en Brasil, SPEI en México, FedNow, UPI. Todos comparten el mismo perfil de exigencia: resolución de directorio de llaves o alias, validación de límites, verificación de cuenta destino y scoring antifraude, todo en sub-segundo, veinticuatro horas al día, todos los días del año, sin ventana de mantenimiento.

Los cores diseñados en los años ochenta y noventa no fueron construidos para eso, y ninguna cantidad de optimización los va a llevar ahí. El DIH no es una mejora en este escenario: es condición de posibilidad. El capítulo 15 desarrolla el caso colombiano en detalle.

Presión 5 — Open Banking y terceros

Exponer APIs a agregadores y proveedores terceros implica asumir un SLA contractual frente a un actor sobre el que no tienes control operativo. Si un agregador decide consultar el saldo de sus cien mil usuarios cada quince minutos, y esa consulta llega directo al core, acabas de darle a un tercero la capacidad de degradar tu canal propio.

El DIH interpone una capa que puedes dimensionar, tarifar y limitar independientemente, sin que el comportamiento del tercero toque nunca el sistema de registro.

EL RESUMEN PARA EL COMITÉ

El DIH no es un proyecto de tecnología. Es la decisión de que **el canal digital deje de depender de la disponibilidad, la capacidad y la ventana operativa del sistema de registro**, a cambio de asumir explícitamente y gobernar una cantidad medida de desactualización del dato. Toda la ingeniería de este libro existe para hacer que esa cantidad sea conocida, acotada y auditable.

Parte II

Anatomía

Las cuatro capas del patrón, con las decisiones de diseño que definen si el resultado sirve consultas en veinte milisegundos o en doscientos.

Capítulo 04

La arquitectura de referencia en cuatro capas

Todo DIH que he visto funcionar tiene estas cuatro capas, aunque las llamen de otra forma o alguna esté implícita. Cuando falta una, el problema que resolvía esa capa reaparece disperso en las otras tres.



Arquitectura de referencia. Las flechas de datos suben desde el sistema de registro; las de consulta bajan desde los consumidores. Las escrituras transaccionales no aparecen en este diagrama: van directo del canal al core (capítulo 10).

Las tres reglas estructurales

Primera: el flujo de datos y el flujo de control van en direcciones opuestas. Los datos suben del core al DIH por sincronización asíncrona. Las consultas bajan del canal al DIH de forma síncrona. Los dos caminos nunca se cruzan en la ruta caliente. Si tu diseño

tiene una consulta que dispara una sincronización, acabas de reintroducir el acoplamiento que el patrón venía a eliminar.

Segunda: cada capa solo conoce a su vecina inmediata. La Capa 4 no sabe si el dato vino de Redis o de Cosmos DB. La Capa 2 no sabe si el consumidor es la app o un agregador. Esta disciplina es lo que te permite cambiar el motor de estado en la fase 3 sin tocar cuarenta consumidores.

Tercera: el modelo canónico vive en la frontera entre la Capa 1 y la Capa 2. Es el punto de mayor apalancamiento de todo el diseño y el que más equipos se saltan. El capítulo 5 explica por qué saltárselo condena el proyecto.

Capa 1 — Sincronización

Aquí se gana o se pierde el proyecto. Todo lo demás es ingeniería conocida; esto es donde el patrón toca el legacy, y el legacy no colabora.

Los cuatro mecanismos

MECANISMO	LATENCIA TÍPICA	IMPACTO EN EL CORE	CUÁNDO USARLO
CDC sobre log transaccional	100 ms – 5 s	Prácticamente nulo	El caso por defecto cuando el core no emite eventos de negocio. Lee el redo/binlog sin consultar tablas. Herramientas: Debezium, GoldenGate, InfoSphere CDC, Qlik Replicate.
Eventos de dominio	50 ms – 1 s	Nulo	Cuando el core es moderno y publica eventos de negocio (Mambu, Thought Machine, 10x). Infinitamente superior al CDC : transporta semántica, no filas.
Puente MQ	200 ms – 3 s	Bajo	Cuando ya existe integración por colas. IBM MQ sigue siendo el sistema nervioso de medio sistema financiero mundial. Se construye un puente MQ → Kafka.
Snapshot / bootstrap	Horas	Alto y planificado	Carga inicial, reconstrucción tras incidente, y alta de un dominio nuevo. Se subestima siempre.

En la práctica se combinan. Un DIH bancario típico usa snapshot para la carga inicial, CDC para los cambios de datos maestros, eventos de dominio para los productos que ya migraron a core moderno, y un puente MQ para los sistemas satélite que nadie va a modernizar en cinco años.

Por qué el CDC crudo no basta

El CDC entrega filas del esquema del legacy. Si propagas esas filas tal cual a la Capa 2, acabas de trasplantar el acoplamiento en vez de romperlo: el día que migres el core, migras también el DIH y los cuarenta consumidores que se acostumbraron a ver `CTA_SL-DO_DISP_VLR`.

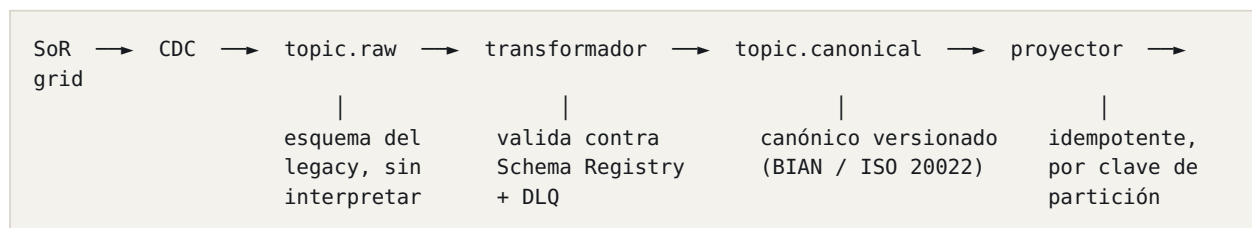
REGLA DURA

Entre el CDC y el motor de estado debe existir siempre una **capa de transformación a modelo canónico**. No es opcional, no es una optimización posterior, y no se puede añadir después sin reescribir a los consumidores. Es la frontera que hace que el DIH sobreviva al core que lo alimenta.

En banca ese canónico no hay que inventarlo. Para pagos, **ISO 20022** ya define las estructuras y está gobernado, versionado y entendido por todo el ecosistema. Para el

modelo de dominio, **BIAN** da el vocabulario y los límites. Los capítulos 13 y 14 desarrollan ambos.

Anatomía del pipeline de ingesta



Cinco decisiones que definen la calidad de este pipeline:

UNO — IDEMPOTENCIA OBLIGATORIA

El CDC entrega *al menos una vez*. Vas a recibir el mismo cambio dos veces, y en un failover vas a recibir un bloque entero repetido. Cada proyección debe ser idempotente por identificador de transacción, LSN u offset. Sin esto, un failover duplica movimientos y el saldo materializado deja de cuadrar.

DOS — ORDENAMIENTO POR PARTICIÓN

Kafka solo garantiza orden dentro de una partición. Si particionas por `accountId`, todos los eventos de esa cuenta llegan en orden y el saldo se puede reconstruir aplicándolos secuencialmente. Si particionas por otra cosa, no. Esta decisión se toma una vez y es carísima de revertir.

TRES — COLA DE MENSAJES FALLIDOS

Un evento que no se puede transformar no puede bloquear la partición ni descartarse en silencio. Va a una DLQ con alerta, y existe un procedimiento documentado de reproceso. La DLQ vacía es una métrica de salud; la DLQ que nadie mira es una bomba de tiempo.

CUATRO — REGISTRO DE ESQUEMAS CON COMPATIBILIDAD FORZADA

El día que un DBA del core añada una columna, tu pipeline debe fallar ruidosamente o adaptarse correctamente, nunca servir datos parciales en silencio. Schema Registry con política de compatibilidad hacia atrás es la única defensa práctica.

CINCO — EL BOOTSTRAP ES UN PROYECTO

Cargar cuarenta millones de cuentas con su histórico de movimientos no es «correr un script». Requiere paralelización, control de presión sobre el core, punto de corte coherente con el inicio del CDC —para no perder ni duplicar cambios en la transición— y un plan de reintento. Presupuéstalo como fase propia.

EL ERROR DE LA TRANSICIÓN SNAPSHOT → CDC

El fallo clásico: se toma el snapshot a las 02:00 y se arranca el CDC a las 06:00. Los cambios de esas cuatro horas se perdieron para siempre y nadie lo detecta hasta la primera reconciliación, meses después. La secuencia correcta es **arrancar el CDC primero**, dejar los eventos acumulándose en el topic, tomar el snapshot después, y reproducir el topic desde el punto de consistencia del snapshot confiando en la idempotencia de la proyección.

Capa 2 — Estado

La decisión tecnológica de fondo y, sobre todo, la decisión de modelado que determina si el DIH responde en veinte milisegundos o en doscientos.

Categorías de motor

CATEGORÍA	EJEMPLOS	LATENCIA P99	CUÁNDO ELEGIRLA
IMDG puro	Hazelcast, GridGain / Apache Ignite, Oracle Coherence	1 – 10 ms	Latencia extrema, cómputo colocado con el dato, necesidad de transacciones y consultas en memoria. La opción más potente y la más exigente en operación.
DIH empaquetado	GigaSpaces Smart DIH, Hazelcast Platform	1 – 15 ms	Cuando quieres CDC, grid y motor de consulta como producto integrado y estás dispuesto a pagar por reducir el trabajo de ensamblaje.
Clave-valor gestionado	Redis Enterprise, Azure Managed Redis, Aerospike	1 – 8 ms	Perfil de acceso mayoritariamente por clave, operación delegada al proveedor, equipo pequeño. El punto de partida sensato para la mayoría de bancos.
NoSQL distribuido	Cosmos DB, ScyllaDB, Cassandra, DynamoDB	5 – 30 ms	Volumen que hace prohibitivo mantener todo en RAM, requisitos multirregión activo-activo, tolerancia a decenas de milisegundos.
Estado de streaming	Kafka Streams + ksqldb, Apache Flink	5 – 50 ms	Cuando el evento es primario y el estado es puramente derivado. Elegante conceptualmente, más difícil de operar y de consultar por clave arbitraria.

CÓMO ELEGIR SIN CAER EN LA DEMO

Toda comparativa de fabricante te va a enseñar latencias de laboratorio con un dato de cien bytes. Tu prueba de concepto debe usar **tu agregado real**, con su tamaño real, bajo tu concurrencia de pico, con réplicas activas y con el CDC corriendo en paralelo. La diferencia entre el número de la demo y el número de tu PoC suele ser de un orden de magnitud. El anexo C contiene la matriz de decisión que uso.

El modelado es más importante que el motor

He visto DIH excelentes sobre Redis y desastres sobre productos que cuestan seis cifras al año. La diferencia casi siempre está en cuatro decisiones de modelado.

UNO — DESNORMALIZA POR CASO DE USO, NO POR ENTIDAD

La unidad de almacenamiento es el **agregado** en el sentido de Domain-Driven Design: el conjunto de datos que se consulta y se invalida junto. La pantalla de detalle de cuenta ne-

cesita titular, saldo, últimos movimientos y estado del producto. Eso es un objeto, no cuatro tablas.

La objeción previsible es la duplicación. La respuesta es que en un modelo de lectura la duplicación no es deuda, es el diseño: no hay anomalías de actualización porque nadie actualiza el DIH directamente. Todo se reprojeta desde el evento.

DOS — LA CLAVE DE PARTICIÓN MANDA

Elegir mal la clave de partición es la única decisión de este libro que no se puede corregir incrementalmente. Determina si una consulta se resuelve en un nodo o requiere un *scatter-gather* sobre todo el cluster, que es la muerte de la latencia p99.

CASO DE USO	CLAVE DE PARTICIÓN	CONSECUENCIA
Vista 360 del cliente	<code>customerId</code>	Todos los productos del cliente en el mismo nodo. Una llamada, cero red interna.
Detalle y movimientos de cuenta	<code>accountId</code>	Movimientos colocados con la cuenta. Cálculo de saldo local.
Directorio de llaves de pago	<code>hash(llave)</code>	Distribución uniforme, resolución en un salto. Ver capítulo 15.
Features antifraude	<code>customerId</code>	Todas las features del sujeto juntas para evaluar en una sola lectura.

TRES — COLOCACIÓN POR AFINIDAD

Cuenta, movimientos y titular en la misma partición permiten resolver la unión localmente dentro de un nodo. Sin afinidad, esa misma consulta se convierte en tres saltos de red y triplica la latencia. Todos los motores serios lo soportan bajo algún nombre: *affinity key*, *partition key*, *hash tag*.

CUATRO — ALMACENAMIENTO POR NIVELES

La memoria es cara. La política que uso como punto de partida en banca retail:

NIVEL	CONTENIDO	LATENCIA	CRITERIO
Memoria	SalDOS, datos maestros, últimos 90 días de movimientos	1 – 5 ms	Cubre el 95 % de las consultas reales del canal.
SSD local / disco	Movimientos de 90 días a 24 meses	10 – 40 ms	Consultado con poca frecuencia pero de forma predecible.
Sistema de registro	Histórico profundo, certificaciones, extractos antiguos	200 ms – 2 s	Read-through bajo demanda, con expectativa de latencia comunicada al usuario.

Sin esta política, el proyecto que nació para reducir la factura del core acaba siendo el que disparó el presupuesto de infraestructura. Es el anti-patrón 4 del capítulo 18 y el capítulo 22 desarrolla el modelo de costo completo.

Capa 3 — Cómputo

Aquí viven los microservicios de dominio. La regla que gobierna toda esta capa cabe en una frase: lleva el cómputo al dato, no el dato al cómputo.

El principio de colocación

Calcular el saldo disponible de una cuenta requiere el saldo contable, las retenciones vigentes, los cheques en canje y las autorizaciones de tarjeta pendientes. Hay dos formas de hacerlo:

ENFOQUE	QUÉ OCURRE	COSTO
Traer el dato	El microservicio pide los cuatro conjuntos al grid, los serializa por red y calcula en su propio proceso.	Cuatro viajes de red, serialización de posiblemente cientos de registros, latencia dominada por transferencia.
Llevar el cómputo	El microservicio envía una función al nodo que posee la partición; el nodo calcula localmente y devuelve un número.	Un viaje de red, un número de retorno. Un orden de magnitud menos de latencia y de ancho de banda.

Los IMDG lo exponen como *entry processor*, *compute task* o función del lado servidor. Redis lo permite mediante scripts y funciones. Es una de las razones principales para elegir un IMDG sobre un almacén clave-valor simple.

EL LÍMITE DEL CÓMPUTO COLOCADO

El código que corre dentro del grid comparte proceso con el almacenamiento. Una función que bloquea, que lanza una excepción no controlada o que consume memoria sin límite **tumba el nodo y con él su partición**. Reglas: sin llamadas de red, sin acceso a disco, sin bucles no acotados, con límite de tiempo explícito, y probado bajo carga antes de producción. El cómputo colocado es un bisturí, no un martillo.

Descomposición en microservicios

Los servicios de esta capa se organizan por *bounded context*, no por entidad técnica. Un servicio de Posición, uno de Movimientos, uno de Directorio de Pagos, uno de Perfil. Cada uno es dueño de sus proyecciones en la Capa 2 y del contrato que expone hacia arriba.

Lo que **no** debe vivir en esta capa: lógica de negocio autoritativa. El DIH no decide si un pago se aprueba; sirve los datos con los que otro sistema lo decide. En el momento en que alguien pone una regla de negocio con consecuencia contable en la capa de cómputo del DIH, has convertido el modelo de lectura en un sistema de registro por accidente. Es el anti-patrón 2.

Capa 4 — Exposición

La capa más visible, la más fácil de construir y la que más rápido se degrada si no se gobierna el contrato desde el primer día.

Un BFF por canal, no un BFF universal

La app móvil, la banca web, el IVR y un agregador tercero tienen necesidades incompatibles: tamaño de carga útil, granularidad, tolerancia a latencia, modelo de autenticación y cadencia de versionado. Un único BFF que sirva a los cuatro acaba siendo un objeto de respuesta con noventa campos donde cada consumidor usa doce distintos.

CANAL	PROTOCOLO TÍPICO	CARACTERÍSTICA DOMINANTE DEL CONTRATO
App móvil	REST o GraphQL	Carga útil mínima, tolerancia a red inestable, versionado lento porque el usuario no actualiza la app.
Banca web	REST o GraphQL	Cargas más ricas, versionado rápido porque el despliegue es del banco.
IVR y sucursal	REST o gRPC	Pocos campos, latencia crítica, altísima sensibilidad a errores.
TPP / Open API	REST estandarizado	Contrato regulatorio o de mercado, versionado formal, cuotas y tarificación explícitas.

El campo que casi nadie incluye y siempre se necesita

Toda respuesta servida desde el DIH debe incluir la marca de tiempo de la última sincronización del dato y, cuando aplique, la clase de frescura. No es un detalle de depuración: es el mecanismo que permite al consumidor y al auditor saber qué verdad está viendo.

```
{
  "accountId": "0012938471",
  "availableBalance": { "amount": 4820500, "currency": "COP" },
  "_meta": {
    "sourceSystem": "CORE-DEP",
    "freshnessTier": "T1",
    "asOf": "2026-08-20T14:32:07.412Z",
    "lagMillis": 840,
    "degraded": false
  }
}
```

Cuando el lag supera el umbral de la clase, `degraded` pasa a `true` y el canal tiene información suficiente para decidir: mostrar un aviso, ocultar la operación de transferencia, o hacer read-through al core. El capítulo 11 desarrolla esta degradación honesta.

REGLA DURA

Un DIH que sirve datos sin declarar su frescura está mintiendo por omisión. El consumidor asume que el dato es actual porque nada le dice lo contrario, y cuando resulte que no lo era, la conversación con el regulador va a ser sobre por qué el banco no sabía qué estaba mostrando. **La marca de frescura es un requisito de auditoría, no una comodidad de ingeniería.**

Parte III

La conversación difícil

Los proyectos de DIH no mueren por problemas técnicos. Mueren porque nadie acordó con el negocio, antes de construir, cuánta desactualización es aceptable y para qué dato.

Taxonomía de frescura

Si solo vas a leer un capítulo de este libro, lee este. La taxonomía de frescura es el artefacto que separa un DIH gobernado de un caché con relaciones públicas.

El error de fondo es tratar la frescura como una propiedad del sistema. No lo es: es una propiedad **de cada dato en cada caso de uso**. El saldo mostrado en la pantalla de inicio y el saldo evaluado al autorizar una transferencia son el mismo número con requisitos radicalmente distintos.

Los dos modos de fallo simétricos:

- **Tratar todo como crítico.** Todo hace read-through al core, el DIH no reduce carga, y el proyecto no tiene caso de negocio. Muere en la revisión de presupuesto.
- **Tratar todo como tolerante.** Alguien autoriza una operación contra un saldo que ya no existe. Muere en la auditoría, y peor.

Las cuatro clases

CLASE	TOLERANCIA	EJEMPLOS BANCARIOS	ESTRATEGIA DE SINCRONIZACIÓN Y DE SERVICIO
T0 Estricta	Cero	Saldo al autorizar un pago o retiro. Límite disponible al aprobar una transacción. Estado de bloqueo judicial de una cuenta.	Read-through síncrono al sistema de registro, o reserva de fondos atómica en el DIH con confirmación posterior (capítulo 10). Nunca se sirve desde la proyección sin más.
T1 Segundos	< 5 s	Saldo mostrado en la app. Estado de una transferencia en curso. Directorio de llaves de pago. Features antifraude de alta rotación.	CDC o eventos de dominio en streaming continuo. Se sirve desde la proyección con marca de frescura. Degradación automática si el lag supera el umbral.
T2 Minutos	< 5 min	Movimientos históricos. Estado de cartera y mora. Puntos de fidelidad. Posición consolidada de inversiones.	CDC en micro-lotes o proyección diferida. Se sirve directo sin verificación. Reconciliación diaria.
T3 Horas o días	< 24 h	Datos demográficos del cliente. Catálogo de productos y tarifas. Segmentación comercial. Condiciones contractuales.	ETL programado nocturno. Reconstrucción completa periódica en lugar de sincronización incremental.

Cómo se produce el artefacto

No es un ejercicio de arquitectura. Es una sesión con negocio, riesgo operativo y cumplimiento, y el entregable lleva firma. La secuencia que uso:

1. **Inventario de datos por pantalla y por operación.** No por tabla ni por entidad: por lo que el usuario ve y por lo que el sistema decide. El mismo campo aparece varias veces con clases distintas, y eso es correcto.
2. **Pregunta única por cada uno:** *«Si este dato tuviera treinta segundos de retraso, ¿qué puede salir mal, para quién, y cuánto cuesta?»* Esa pregunta traduce un concepto de sistemas distribuidos a un lenguaje que riesgo operativo sabe evaluar.
3. **Clasificación y firma.** Cada dato queda en una clase, con dueño de negocio nombrado. La firma no es burocracia: es lo que te protege cuando alguien pregunte, dos años después, por qué el sistema mostró lo que mostró.
4. **Traducción a requisito técnico.** Cada clase se convierte en un umbral de lag monitorizado, con alerta y con comportamiento de degradación definido.

EL DESCUBRIMIENTO HABITUAL

En todas las sesiones que he facilitado, el negocio clasifica inicialmente casi todo como T0 —«el saldo tiene que estar exacto»— y luego, al desarrollar la pregunta del punto 2, la mayoría baja a T1 o T2. El movimiento típico es que menos del cinco por ciento de los datos permanece en T0, y ese cinco por ciento son casi siempre puntos de decisión transaccional, no pantallas. Ese hallazgo es el que desbloquea el caso de negocio.

REGLA DURA

Ningún dato entra al DIH sin clase de frescura asignada y dueño de negocio nombrado. El proceso de alta de un dato nuevo incluye esa clasificación como paso obligatorio, no como documentación posterior. La plantilla está en el anexo A.

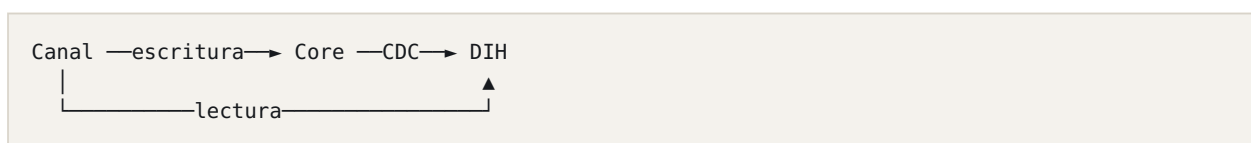
Capítulo 10

Patrones de escritura

El DIH es un modelo de lectura, pero los usuarios escriben. Cómo se maneja esa asimetría define la experiencia del canal y el perfil de riesgo del sistema.

Patrón A — Camino de comando separado

La escritura va directo al sistema de registro. El DIH se entera después, por el evento de sincronización. Es simple, seguro y correcto por construcción.



Su único defecto es real y visible: el usuario transfiere, la pantalla se recarga, y el saldo sigue mostrando el valor anterior porque el evento aún no llegó. Es el problema de *read-your-own-writes*, y hunde la percepción de calidad de la app aunque el sistema esté funcionando exactamente como se diseñó.

Patrón B — Escritura optimista con reconciliación

La solución al problema anterior. La operación va al core y, en paralelo, se escribe en el DIH una entrada marcada como pendiente, con tiempo de vida corto.

1. El canal envía la transferencia al core y recibe confirmación de aceptación.
2. El servicio de dominio escribe en el DIH un movimiento **PENDING** con TTL de, digamos, sesenta segundos, y ajusta el saldo proyectado.
3. El usuario ve inmediatamente su operación reflejada, marcada visualmente como en proceso.
4. Llega el evento de CDC con la operación confirmada. La proyección la promueve a **SETTLED** y elimina la entrada pendiente por idempotencia.
5. Si el TTL vence sin evento, la entrada pendiente desaparece, el saldo vuelve al valor proyectado del core, y se emite una alerta de reconciliación.

Es más complejo, y es lo que hacen todas las aplicaciones bancarias que se sienten rápidas. El punto crítico es el paso 5: la entrada pendiente **debe** tener TTL. Sin él, un fallo de sincronización deja un movimiento fantasma permanente en la vista del cliente.

Patrón C — Write-behind

El DIH acepta la escritura, responde al cliente y propaga al sistema de registro de forma asíncrona. Es el patrón de mayor rendimiento y el de mayor riesgo.

REGLA DURA

Write-behind **nunca** para datos financieros. Nunca para saldos, movimientos, autorizaciones, límites ni nada con consecuencia contable. Válido exclusivamente para datos sin impacto en el libro mayor: preferencias de usuario, marcado de notificación leída, estado de sesión, telemetría de uso, favoritos, apodos de contactos. La pregunta de control es directa: *si este dato se pierde en un fallo de nodo, ¿alguien pierde dinero o el banco incumple algo?*

Patrón D — Reserva de fondos en el grid

El patrón avanzado, necesario para pagos inmediatos de alto volumen donde consultar el core en la ruta caliente no es viable. El DIH mantiene el saldo con un bloque de fondos reservados y opera atómicamente sobre él.

PASO	COMPORTAMIENTO
Reserva	Operación atómica en el grid: verifica disponible, decrementa y crea una reserva identificada con TTL. Todo dentro de la partición, sin red.
Envío	La orden se envía al core y a la red de pagos con el identificador de reserva como clave de idempotencia.
Confirmación	El evento de liquidación libera la reserva y aplica el movimiento definitivo.
Compensación	Si la operación se rechaza o el TTL vence, la reserva se revierte y el disponible se restituye. Es una saga, con todo lo que eso implica.

Requisitos no negociables para este patrón: atomicidad garantizada por el motor dentro de la partición, TTL en toda reserva, log de auditoría inmutable de cada reserva y cada compensación, y reconciliación con el core en cada ciclo de liquidación. Es el único caso donde el DIH toca dinero, y por eso el capítulo 19 le dedica controles específicos.

Capítulo 11

Garantías operacionales

Cuatro mecanismos que convierten una copia de datos en un sistema del que se puede responder ante un auditor.

Idempotencia

Ya tratada en el capítulo 5, pero conviene insistir porque es la garantía de la que dependen todas las demás. Toda proyección debe poder aplicarse dos veces con el mismo resultado. La clave de idempotencia viene del origen —identificador de transacción del core, LSN, offset de Kafka— y nunca se genera en el DIH.

Ordenamiento

Solo existe orden dentro de una partición. Ese hecho gobierna el diseño completo de la Capa 1. Si dos eventos que deben aplicarse en orden pueden caer en particiones distintas, el diseño está mal y ninguna cantidad de código lo va a arreglar.

Reconciliación activa

El mecanismo que más se omite y el más importante. Un proceso periódico toma una muestra —aleatoria y estratificada— de entidades del DIH, consulta el sistema de registro, y compara.

DIMENSIÓN	PRÁCTICA RECOMENDADA
Frecuencia	Continua para T1 con muestra pequeña; diaria completa para T2 y T3 usando totales agregados o sumas de verificación.
Muestreo	Estratificado: cuentas de alto saldo, cuentas de alta transaccionalidad y muestra aleatoria del resto. El error no se distribuye uniformemente.
Método	Comparación de suma de verificación sobre campos materiales, no de objeto completo. Comparar todo es caro e introduce falsos positivos por diferencias de formato.
Umbral	Tasa de divergencia objetivo por debajo del 0,001 %. Cualquier divergencia en un dato T0 o T1 es un incidente, no una métrica.
Remediación	Reproyección automática de la entidad divergente y registro del evento. Divergencia recurrente en el mismo dominio es un defecto de pipeline, no un caso aislado.

Confía, pero verifica. Un DIH sin reconciliación activa no es un sistema confiable con un riesgo pequeño; es un sistema que está mintiendo con una probabilidad que nadie ha medido.

Degradación honesta

Cuando el lag de sincronización supera el umbral de una clase de frescura, el DIH debe cambiar de comportamiento de forma explícita y observable. Las tres opciones legítimas, en orden de preferencia:

1. **Informar.** Servir el dato marcando `degraded: true` con su `asOf`, y que el canal muestre el aviso correspondiente.
2. **Desviar.** Hacer read-through al sistema de registro para los datos afectados, aceptando la latencia mayor mientras dure la degradación.
3. **Rechazar.** Para operaciones T0, negar la operación con un código de error específico en lugar de decidir sobre datos que se sabe caducos.

La cuarta opción —seguir sirviendo el dato caduco como si nada— no es una opción. Es el modo de fallo que convierte un incidente técnico en un evento de riesgo operativo reportable.

Capítulo 12

CAP, PACELC y el gerente de riesgo

La teoría de sistemas distribuidos es corta, tiene cuarenta años y explica por qué este patrón funciona. Vale la pena poder enunciarla en dos minutos frente a alguien que no es ingeniero.

Lo que dice CAP y lo que no

El teorema establece que un sistema distribuido no puede garantizar simultáneamente consistencia, disponibilidad y tolerancia a particiones de red. Como las particiones de red no son opcionales —ocurren— la elección real es entre consistencia y disponibilidad *durante* una partición.

El malentendido común es leerlo como una elección permanente de arquitectura. No lo es: es una elección que se toma por operación y durante el fallo. El sistema de registro elige consistencia: si no puede garantizar el saldo, rechaza. El DIH elige disponibilidad: sirve la última verdad conocida y declara su antigüedad.

PACELC, que es lo que realmente importa

La extensión de Daniel Abadi añade la parte que CAP omite: *y si no hay partición, la elección es entre latencia y consistencia*. Esa es la decisión que un DIH toma miles de veces por segundo en operación normal.

Si hay Partición → elegir entre Disponibilidad y Consistencia
En caso contrario → elegir entre Latencia y Consistencia

Un DIH es, en notación PACELC, un sistema **PA/EL**: prioriza disponibilidad durante la partición y latencia en operación normal. El sistema de registro que lo alimenta es **PC/EC**. La arquitectura completa combina los dos deliberadamente, y la taxonomía de fresca es exactamente el documento que dice qué dato se sirve desde cuál.

Cómo se explica esto en un comité

Un banco no tiene un saldo. Tiene un saldo contable en el libro mayor, un saldo disponible después de retenciones, un saldo que el cajero automático conoce y un saldo que el cliente ve en la app. Nunca han sido el mismo número y nunca lo serán. Lo único que este proyecto cambia es que por primera vez vamos a saber, medir y publicar cuánto se diferencian.

— La frase con la que he desbloqueado más comités de riesgo

Ese argumento funciona porque es verdad y porque invierte la carga de la objeción. El estado actual no es «todo consistente»; es «inconsistente sin instrumentación». El DIH no introduce la inconsistencia: la hace visible y acotada.

Parte IV

El dominio

Cómo el patrón aterriza en los estándares que ya gobiernan la banca —BIAN e ISO 20022— y en el caso colombiano de pagos inmediatos.

Capítulo 13

BIAN: el DIH como materialización de Service Domains

Si tu organización ya trabaja con BIAN, el DIH deja de ser un concepto nuevo y se convierte en algo más simple: la materialización física de las vistas de lectura de los Service Domains que ya tienes definidos.

BIAN descompone el negocio bancario en dominios de servicio con límites y responsabilidades definidas. Sus *Semantic APIs* distinguen entre acciones que consultan y acciones que modifican. Esa distinción mapea casi uno a uno con la separación del DIH.

SERVICE DOMAIN	MATERIALIZACIÓN EN EL DIH	CLASE DE FRESCURA TÍPICA Y CLAVE
Current Account	Agregado de saldo, estado y movimientos por cuenta	T1 para saldo, T2 para movimientos. Clave: <code>accountId</code>
Savings Account	Equivalente al anterior con reglas de producto propias	T1 / T2. Clave: <code>accountId</code>
Customer Reference Data Management	Vista 360 del cliente: identificación, contacto, segmento	T3. Clave: <code>customerId</code>
Party Reference Data Directory	Directorio de contrapartes, alias y llaves de pago	T1. Clave: <code>hash(llave)</code>
Product Directory	Catálogo de productos, tarifas y condiciones	T3. Clave: <code>productId</code>
Customer Position	Posición consolidada multi-producto del cliente	T2. Clave: <code>customerId</code>
Payment Order	Estado y trazabilidad de órdenes en vuelo	T1. Clave: <code>paymentId</code>
Card Authorization	Autorizaciones pendientes y límites disponibles	T0 / T1. Clave: <code>cardId</code>
Customer Credit Rating	Score interno y features de comportamiento	T2. Clave: <code>customerId</code>
Fraud Evaluation	Features operacionales para scoring en tiempo real	T1. Clave: <code>customerId</code>

La regla de federación que BIAN te regala

La disciplina de dominios aporta algo más valioso que el vocabulario: el criterio para saber dónde poner las fronteras. **Un DIH por dominio de negocio, no un DIH monolítico corporativo.**

Esa frontera es lo que evita el error que convierte el patrón en un ESB con memoria RAM: un equipo central dueño de todo, con una cola de nueve meses para añadir un campo, que reproduce exactamente las patologías organizacionales que los microservicios vinieron a resolver. El capítulo 23 desarrolla el modelo operativo federado.

Capítulo 14

ISO 20022 como canónico del backbone

El modelo canónico del DIH no hay que inventarlo para todo lo que sea pagos. Ya existe, está gobernado por un organismo internacional, versionado, y entendido por cada contraparte con la que tu banco va a hablar.

ISO 20022 define un diccionario de datos y un conjunto de esquemas de mensaje para el dominio financiero. Los que aparecen con más frecuencia en el pipeline de un DIH bancario:

FAMILIA	MENSAJES TÍPICOS	USO EN EL CONTEXTO DEL DIH
pain	pain.001, pain.002	Iniciación de pago desde el cliente y reporte de estado. Alimenta la proyección de órdenes en vuelo.
pacs	pacs.008, pacs.002, pacs.004	Liquidación interbancaria, estado y devolución. Es el evento que confirma o revierte una reserva de fondos.
camt	camt.052, camt.053, camt.054	Reportes de movimientos y notificaciones de abono/cargo. Fuente natural para la proyección de movimientos.
acmt	acmt.023, acmt.024	Verificación de identidad y de cuenta. Relevante para directorios de alias.

LA DECISIÓN PRÁCTICA

No inventes un canónico propio si ISO 20022 cubre el dominio. Un canónico interno tiene tres costos que se subestiman: hay que gobernarlo, hay que documentarlo para cada equipo nuevo, y hay que traducirlo en cada frontera con el exterior. ISO 20022 ya pagó esos tres costos. **Para dominios no cubiertos** —perfil de cliente, catálogo comercial, features de riesgo— sí necesitas canónico propio, y ahí BIAN da la estructura.

UN MATIZ SOBRE EL PESO DEL FORMATO

ISO 20022 en XML es verboso. Usarlo como formato de serialización en un topic de alto volumen tiene un costo real en ancho de banda y en CPU de deserialización. La práctica común es adoptar el *modelo semántico* de ISO 20022 —los nombres, la estructura, los tipos, los códigos— y serializarlo en Avro o Protobuf en el backbone interno, reservando el XML para las fronteras externas donde el estándar es contractual.

Capítulo 15

Caso Colombia: Bre-B, DICE y el directorio de llaves

Bre-B es, para el arquitecto latinoamericano, el mejor caso de estudio disponible sobre por qué el patrón DIH dejó de ser opcional. Y está documentado en español, gratis, por el Banco de la República.

Qué es y cómo está construido

Bre-B es el sistema de pagos inmediatos interoperado de Colombia, en operación plena desde octubre de 2025. Su diseño descansa en dos componentes centrales: el **DICE**, un directorio que permite enrutar pagos usando llaves o alias —número de celular, correo, documento— en lugar del número de cuenta, y el **MOL**, que asegura la liquidación en tiempo real con dinero de banco central.

Sobre esa base el esquema impone estándares operativos que definen directamente los requisitos no funcionales de cualquier participante: mensajería **ISO 20022**, un tiempo máximo de veinte segundos por transferencia, límite por operación expresado en unidades de valor básico, y reglas comunes de códigos QR y de experiencia de usuario.

Las cifras que definen el problema de arquitectura

MAGNITUD	VALOR A ENERO DE 2026
Entidades vinculadas	218, incluyendo 26 bancos, 153 cooperativas y 4 SEDPE
Llaves registradas en el DICE	99 millones, acumuladas entre julio de 2025 y enero de 2026
Clientes representados	33,9 millones de personas
Comercios representados	2,9 millones
Ventana temporal por transferencia	20 segundos extremo a extremo

Fuente: documento técnico de Bre-B, Banco de la República, febrero de 2026.

Por qué esto es un problema de DIH y no de core

Un directorio de noventa y nueve millones de llaves, consultado en la ruta caliente de cada pago, dentro de una ventana de veinte segundos que además debe cubrir validación de límites, antifraude, enrutamiento y liquidación. Y disponible las veinticuatro horas, todos los días, sin ventana de mantenimiento.

Ningún core bancario tradicional resuelve eso. La resolución de llave debe ocurrir en una capa de datos operacional, en memoria, particionada por hash de la llave, con réplicas y con sincronización continua. Eso es literalmente la Capa 2 de este libro.

El propio esquema regulatorio lo hace explícito: los participantes deben incorporar procedimientos que garanticen la **sincronización de llaves en los directorios federa-****dos** cuando participan en varios sistemas de pago, además de control de parámetros de tiempo y de límites máximos por transacción. Sincronización de estado operacional replicado entre organismos independientes, con garantías de consistencia acordadas: el problema exacto que resuelve el patrón.

Diseño de referencia para el participante

CANAL DE PAGO

App / web / QR → solicitud de pago por llave



RESOLUCIÓN (T1, SUB-50 MS)

DIH: directorio local de llaves particionado por hash(llave)
Devuelve entidad destino, cuenta enmascarada y nombre del titular



VALIDACIÓN (T0 / T1, SUB-100 MS)

DIH: límites del cliente, features antifraude precalculadas
Reserva de fondos atómica sobre saldo proyectado



LIQUIDACIÓN

Orden ISO 20022 (pacs.008) → SPBVI → MOL
Confirmación (pacs.002) → libera reserva y aplica movimiento

La resolución y la validación —los dos pasos sensibles a latencia— ocurren íntegramente en el DIH. El core solo participa en la liquidación, que es donde debe estar la autoridad contable. De los veinte segundos de presupuesto, el DIH consume menos de doscientos milisegundos y deja el resto para la red interbancaria.

EJERCICIO RECOMENDADO

Descarga el documento técnico de Bre-B —referencia completa en el capítulo 25— y léelo con ojos de arquitecto de DIH en lugar de con ojos de analista de negocio. Cada requisito operativo del esquema se traduce directamente en un requisito no funcional de tu capa de datos. Es el mejor ejercicio de diseño disponible en español y sale gratis.

Parte V

Casos de uso

Ocho aplicaciones del patrón ordenadas por retorno sobre esfuerzo, y un capítulo aparte para la que justifica el proyecto entero.

Los ocho casos, ordenados por retorno

Este orden no es arbitrario. Es el que produce valor demostrable antes de que se agote la paciencia del patrocinador, que es el recurso más escaso de cualquier programa de modernización.

#	CASO	ESFUERZO	RETORNO	MÉTRICA DE ÉXITO
01	Saldos y movimientos	Bajo	Muy alto	% de reducción de lecturas enviadas al core
02	Posición consolidada / Cliente 360	Medio	Alto	p99 de la pantalla de inicio; número de llamadas backend por carga
03	Coexistencia y migración de core	Alto	Muy alto	Consumidores que requieren cambio por ola de migración
04	Pagos inmediatos	Alto	Muy alto	p99 de resolución de llave; tasa de timeout del esquema
05	Fraude en tiempo real	Medio	Alto	Latencia de recuperación de features; cobertura de features en línea
06	Open Banking / TPP	Medio	Medio	Cumplimiento de SLA contractual; aislamiento del core
07	Stand-in de tarjetas	Alto	Medio	Autorizaciones cursadas durante indisponibilidad del autorizador
08	Onboarding y KYC	Bajo	Medio	Costo de consultas repetidas a burós y listas

01 — Saldos y movimientos

El clásico y la fase uno de todo DIH bancario del mundo. Proyección de saldo y movimientos por cuenta, servida a la app y a la web. Reducciones del setenta al noventa por ciento del tráfico de lectura al core son el rango habitual, y es el caso que financia todo lo demás. Empieza por un solo producto —cuenta de ahorros— y una sola pantalla.

02 — Posición consolidada y Cliente 360

Agrega cuentas, tarjetas, créditos, inversiones y seguros procedentes de cinco o seis sistemas distintos, y los sirve en una sola llamada de decenas de milisegundos. Sin DIH esto solo se logra con un fan-out síncrono a seis backends, cuya latencia es la del más lento y cuya disponibilidad es el producto de las seis. Es el caso donde el patrón hace algo que sencillamente no se puede hacer de otra forma.

03 — Coexistencia y migración de core

Tiene capítulo propio. Ver capítulo 17.

04 — Pagos inmediatos

Desarrollado en el capítulo 15 con el caso de Bre-B. Resolución de directorio, validación de límites, enriquecimiento de datos del beneficiario y verificación antifraude dentro de la ventana del esquema.

05 — Fraude y decisionamiento en tiempo real

El DIH como *feature store* operacional. Las features de comportamiento —gasto promedio de treinta días, geografía habitual, velocidad transaccional, desviación respecto al patrón del segmento— se precálculan en la Capa 3 y quedan disponibles para el modelo de scoring en menos de diez milisegundos.

UN MATIZ QUE SE PASA POR ALTO

El feature store operacional y el feature store analítico deben calcularse con **la misma lógica**, o el modelo entrenado offline y el modelo evaluado en línea ven mundos distintos. Ese desfase se llama *training-serving skew* y degrada silenciosamente la precisión del modelo. La solución es una única definición de feature con dos rutas de materialización, no dos implementaciones que se parecen.

06 — Open Banking y terceros

Cumplir SLA contractual frente a agregadores sin exponer el core. Añade tarificación y cuotas por consumidor, que son mucho más fáciles de implementar en una capa que controlas que en un mainframe.

07 — Stand-in processing de tarjetas

Autorizar transacciones dentro de límites conservadores cuando el autorizador principal está indisponible. Requiere el patrón de reserva de fondos del capítulo 10, un modelo de riesgo explícito sobre cuánto se está dispuesto a autorizar a ciegas, y aprobación formal de riesgo. Alto valor y alta responsabilidad.

08 — Onboarding y KYC

Agregación de listas restrictivas, consultas a buró y validación de identidad, con almacenamiento del resultado de las consultas costosas. El retorno viene de dejar de pagar dos veces por la misma consulta a un proveedor externo. Cuidado con la vigencia regulatoria del dato almacenado: una consulta de lista restrictiva tiene fecha de caducidad legal, no técnica.

Coexistencia y migración de core

Este solo caso justifica el DIH en cualquier programa de modernización de core. Si tienes una migración en marcha o en el horizonte, este capítulo es el argumento completo.

El problema real de una migración de core

Nadie migra un core de golpe. Se migra por olas: primero un producto, luego un segmento, luego una región. Durante dieciocho o treinta y seis meses coexisten dos sistemas de registro, y el problema no es técnico sino combinatorio: **cada uno de los cuarenta consumidores del core tiene que saber, para cada cliente, cuál de los dos sistemas es el autoritativo.**

La solución ingenua es poner lógica de enrutamiento en cada consumidor. Eso significa cuarenta implementaciones de la misma regla, cuarenta oportunidades de divergencia, y cuarenta despliegues coordinados cada vez que una ola de migración avanza. Es la razón por la que las migraciones de core se retrasan años.

El DIH como capa de abstracción de la migración

LOS 40 CONSUMIDORES

Consultan al DIH. No saben ni les importa dónde vive la cuenta.



DIH — MODELO CANÓNICO ÚNICO

Una sola forma del dato, independiente del origen
Contrato estable durante toda la migración

▲ la decisión de ruta vive AQUÍ, una sola vez

CAPA 1 — SINCRONIZACIÓN DUAL

CDC desde el core legacy + Eventos desde el core nuevo
Transformación a canónico común · Tabla de estado de migración por cuenta



DOS SISTEMAS DE REGISTRO COEXISTIENDO

Core legacy (cuentas no migradas) Core nuevo (cuentas migradas)

La regla de enrutamiento se implementa **una vez**, en la capa de sincronización. Los consumidores nunca se enteran de que hubo una migración. Cuando la última cuenta migra, se apaga el CDC del legacy y no hay que tocar ni un consumidor.

Secuencia de una ola de migración con DIH

FASE	ACCIÓN	DETALLE
1	Preparación	Las cuentas de la ola se cargan en el core nuevo en modo espejo. Ambos cores reciben las mismas operaciones; el legacy sigue siendo autoritativo.
2	Comparación en sombra	El DIH ingesta de ambos orígenes en proyecciones separadas y compara continuamente. Toda divergencia es un defecto de migración detectado antes de afectar a nadie.
3	Corte	La tabla de estado de migración marca las cuentas de la ola como pertenecientes al core nuevo. El cambio es de configuración, no de código, y es reversible.
4	Observación	Se mantiene el CDC del legacy durante un período de gracia. Si aparece un problema, la reversión es cambiar la marca de vuelta.
5	Cierre	Se apaga la ingesta del legacy para esas cuentas y se libera la capacidad correspondiente.

EL ARGUMENTO EN UNA FRASE

Sin DIH, una migración de core es un proyecto de N consumidores multiplicado por M olas de migración. Con DIH, es un proyecto de M olas y un solo punto de enrutamiento. **La complejidad pasa de multiplicativa a aditiva**, y esa es la diferencia entre una migración de dieciocho meses y una de cuatro años.

PATRÓN RELACIONADO

Esto es *Strangler Fig* —la higuera estranguladora de Martin Fowler— aplicado a la capa de datos en lugar de a la capa de aplicación. El DIH es el andamiaje que permite que el sistema nuevo crezca alrededor del viejo hasta reemplazarlo, sin un momento de corte total.

Parte VI

Lo que sale mal

Doce anti-patrones que he visto en producción, y el conjunto de controles de seguridad y cumplimiento que el patrón exige en un entorno regulado.

Capítulo 18

Catálogo de anti-patrones

Cada uno de estos lo he visto. Están ordenados por frecuencia, no por gravedad. El primero es responsable de más fracasos que los otros once juntos.

01 — EL CACHE GLORIFICADO

Síntoma: el sistema funciona bien seis meses y luego aparece la primera divergencia que le cuesta dinero a un cliente.

Causa raíz: se construyó la infraestructura sin ninguno de los cuatro elementos de gobierno: modelo canónico, SLA de frescura, reconciliación y dueño.

Remedio: las cuatro preguntas del capítulo 2, respondidas por escrito, antes de la primera línea de código.

02 — EL DIH QUE SE CONVIRTIÓ EN SISTEMA DE REGISTRO

Síntoma: tras un reinicio del cluster, un dato desapareció y no se puede reconstruir.

Causa raíz: alguien escribió en el grid un dato que no existe en ningún sistema de registro. Suele empezar por algo inocente: un apodo de contacto, una preferencia, un estado de proceso.

Remedio: regla dura — si un dato no puede reconstruirse desde un sistema de registro, no pertenece al DIH sin persistencia declarada, respaldo y dueño explícito. Y si lo aceptas, deja de ser «solo un modelo de lectura» y hereda todos los requisitos de un SoR.

03 — EL BIG BANG

Síntoma: dieciocho meses de proyecto, cero valor entregado, y un comité que empieza a preguntar.

Causa raíz: «carguemos todo el core al grid». Costo de memoria astronómico y ninguna entrega intermedia que justifique continuar.

Remedio: el roadmap del capítulo 20. Un caso de uso, una métrica, doce semanas.

04 — IGNORAR EL COSTO DE LA MEMORIA

Síntoma: el proyecto que nació para reducir la factura del core disparó el presupuesto de infraestructura.

Causa raíz: sin políticas de TTL, expulsión y niveles de almacenamiento, el grid crece monótonamente.

Remedio: FinOps desde el diseño, no como auditoría posterior. Capítulo 22.

05 — PROPAGAR EL ESQUEMA DEL LEGACY

Síntoma: el día que se migra el core hay que reescribir el DIH y los cuarenta consumidores.

Causa raíz: se omitió la capa de transformación a canónico y el CDC alimentó el grid con las filas del legacy tal cual.

Remedio: capítulo 5. No es recuperable a posteriori sin reescribir consumidores.

06 — EL DIH MONOLÍTICO COMPARTIDO

Síntoma: nueve meses de cola para añadir un campo. Los equipos empiezan a construir cachés propios por su cuenta.

Causa raíz: un equipo central dueño de todo. Se reinventó el ESB con sus patologías organizacionales intactas.

Remedio: federación por dominio con un *paved road* que permita a cada equipo publicar su vista sin pasar por el centro. Capítulo 23.

07 — SIN GOBIERNO DE EVOLUCIÓN DE ESQUEMA

Síntoma: el DIH sirve datos parcialmente vacíos y nadie lo detecta durante semanas.

Causa raíz: un cambio en el core alteró el esquema, el pipeline lo absorbió en silencio.

Remedio: Schema Registry con compatibilidad forzada y fallo ruidoso.

08 — SEGURIDAD POSTERGADA

Síntoma: la auditoría pregunta quién puede leer el mapa de saldos y no hay respuesta.

Causa raíz: el grid contiene datos de todos los clientes y se trató como infraestructura interna de confianza.

Remedio: capítulo 19 completo.

09 — LA RECONCILIACIÓN QUE NADIE MIRA

Síntoma: existe un job de reconciliación, produce un reporte, y el reporte va a una carpeta que nadie abre.

Causa raíz: se implementó como cumplimiento de un requisito, no como control operativo.

Remedio: la divergencia debe generar alerta accionable con dueño y procedimiento, no un informe.

10 — DEGRADACIÓN SILENCIOSA

Síntoma: durante un incidente de sincronización el sistema siguió sirviendo datos de hace cuarenta minutos como si fueran actuales.

Causa raíz: no se implementó comportamiento de degradación ni marca de frescura en la respuesta.

Remedio: capítulos 8 y 11.

11 — LÓGICA DE NEGOCIO EN LA CAPA DE DATOS

Síntoma: una regla que decide si una operación se permite vive dentro del DIH, y nadie sabe si es la misma regla que aplica el core.

Causa raíz: conveniencia. Estaba cerca del dato y se puso ahí.

Remedio: el DIH sirve datos; otros sistemas deciden. Sin excepciones para reglas con consecuencia contable.

12 — OPTIMIZAR LA LATENCIA MEDIA

Síntoma: el tablero muestra 12 ms de promedio y los usuarios se quejan de lentitud.

Causa raíz: la media oculta la cola. El p99 está en 900 ms por *scatter-gather* en una clave mal particionada.

Remedio: los objetivos se fijan en p99 y p99.9, medidos en el borde y no en el grid. Capítulo 21.

Seguridad y cumplimiento

El grid contiene, en claro y en memoria, los datos financieros de todos los clientes del banco. Tratarlo como infraestructura interna de confianza es el error de seguridad más caro que se puede cometer en este patrón.

Principios

Zero Trust aplica **dentro** del grid, no solo en el perímetro. Cada nodo autentica a cada nodo. Cada servicio que lee autentica y autoriza contra un mapa específico. La pertenencia a la red no otorga acceso a los datos.

CONTROL	IMPLEMENTACIÓN ESPERADA
Cifrado en tránsito	TLS mutuo entre nodos del cluster y entre clientes y cluster. Rotación automatizada de certificados. Sin excepciones para tráfico «interno».
Cifrado en reposo	Aplicado a la persistencia del grid, a los niveles en disco y a los topics del backbone. Claves gestionadas en HSM o servicio de claves, nunca en configuración.
Tokenización	El número de tarjeta completo nunca entra al DIH. Se almacena token y últimos cuatro dígitos. Es requisito PCI DSS, no una buena práctica opcional.
Minimización de datos	Solo los campos que los casos de uso definidos requieren. Cada campo adicional es superficie de exposición y costo de memoria.
Control de acceso	RBAC por mapa o espacio de nombres. El servicio de movimientos no lee el mapa de perfiles. Principio de menor privilegio aplicado a nivel de estructura de datos.
Trazabilidad	Registro de acceso a datos sensibles con identidad del servicio consumidor y propósito. Retención acorde a la política del banco.
Segregación de entornos	Ni un dato productivo en entornos de desarrollo o prueba. Enmascaramiento o datos sintéticos, con verificación automatizada.
Continuidad	Plan de recuperación probado, no documentado. RPO y RTO declarados y verificados en simulacro periódico.

Marco normativo aplicable en Colombia

MARCO	IMPLICACIÓN DIRECTA SOBRE EL DIH
PCI DSS	Prohíbe almacenar datos de autenticación sensibles y exige protección del PAN. Determina qué puede y qué no puede entrar al grid en el dominio de tarjetas.
Ley 1581 de 2012 y decretos reglamentarios	Régimen de protección de datos personales: finalidad declarada, autorización del titular, y derecho de supresión. El derecho de supresión implica que el DIH debe poder borrar de forma verificable, incluida la persistencia y los respaldos.
Circulares de la Superintendencia Financiera	Riesgo operativo, continuidad del negocio, gestión de la seguridad de la información y requisitos sobre computación en la nube. Definen buena parte de tus objetivos de disponibilidad y de tus obligaciones de reporte de incidentes.
Regulación de SPBVI	Para participantes de pagos inmediatos, define parámetros de tiempo, límites y obligaciones de sincronización de directorios. Ver capítulo 15.

EL REQUISITO QUE ROMPE DISEÑOS

El derecho de supresión del titular obliga a borrar de forma verificable, incluyendo réplicas, niveles de disco, respaldos y topics del backbone con retención larga. Un backbone de eventos configurado con retención infinita e inmutable es incompatible con ese derecho si transporta datos personales. La solución habitual es compactación por clave con borrado por *tombstone*, más cifrado por titular con destrucción de clave —*crypto-shredding*— para lo que no se puede reescribir. **Esto se diseña al principio o se paga carísimo después.**

Parte VII

Ejecución

Cómo se construye por fases, qué se mide, cuánto cuesta y cómo se organiza el equipo para que el patrón sobreviva a su primer año.

Capítulo 20

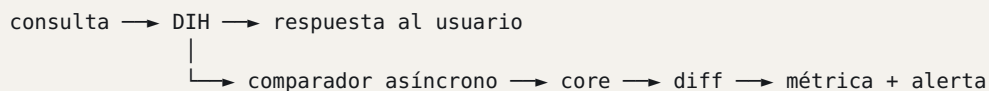
Roadmap por fases

Cinco fases con criterio de salida explícito. Ninguna avanza sin cumplir el criterio de la anterior, y la fase 1 entrega valor medible antes del tercer mes.

FASE	DURACIÓN	ENTREGABLES	CRITERIO DE SALIDA
0 Funda- ción	4 – 6 semanas	Taxonomía de frescura firmada. Selección tecnológica con PoC de latencia y costo real. Canónico del primer dominio. ADR de las decisiones estructurales.	El documento de taxonomía tiene firma de negocio y riesgo. La PoC midió el agregado real bajo concurrencia de pico.
1 Un caso, una mé- trica	8 – 12 semanas	Salos y movimientos de un producto. Pipeline CDC completo con DLQ y Schema Registry. Reconciliación básica. Observabilidad del lag.	Reducción medida de lecturas al core con el tráfico cortado, tras dos a cuatro semanas de comparación en sombra sin divergencias materiales.
2 Consoli- dación	3 – 6 meses	Segundo y tercer dominio. Reconciliación automatizada con alerta. Degradación honesta implementada. Lag como métrica de tablero ejecutivo.	Tres dominios en producción. Tasa de divergencia bajo umbral durante un trimestre completo. Un incidente de degradación gestionado correctamente.
3 Federa- ción	6 – 12 meses	Propiedad por dominio. Plantillas y camino pavimentado. El equipo de plataforma pasa de constructor a habilitador. Catálogo de vistas publicadas.	Un equipo de dominio publicó una vista nueva de principio a fin sin intervención del equipo de plataforma.
4 Escritu- ra y mul- tirregión	Variable	Reserva de fondos. Stand-in processing. Replicación activo-activo geográfica.	Prueba de conmutación regional ejecutada con tráfico real y RPO verificado.

La técnica de la comparación en sombra

Es el mecanismo que convierte el corte de tráfico de un acto de fe en un hecho medido. Durante dos a cuatro semanas, cada consulta se resuelve por los dos caminos: el DIH sirve la respuesta y, de forma asíncrona, un proceso consulta el core y compara.



Se corta cuando la tasa de divergencia material se mantiene bajo umbral durante el período completo. Los primeros días siempre aparecen diferencias, y casi todas son de formato —redondeo, zona horaria, orden de movimientos— no de valor. Esas también hay que corregirlas: un consumidor que recibe la fecha en otro huso horario tiene un defecto real.

SOBRE LA FASE 0

La tentación permanente es saltársela para «empezar a construir ya». No lo hagas. Seis semanas de fase 0 evitan seis meses de reescritura, y el entregable más valioso —la taxonomía de frescura— no lo puede producir el equipo técnico solo. Necesita calendario de negocio, y ese calendario tarda en conseguirse.

Capítulo 21

NFR y observabilidad

Los requisitos no funcionales que exijo en cualquier especificación de DIH, y la métrica que debe estar en el tablero ejecutivo aunque el resto no lo esté.

INDICADOR	OBJETIVO TÍPICO	NOTA DE MEDICIÓN
Latencia p50	< 20 ms	Medida en el borde de la Capa 4, no dentro del grid. Al usuario le importa el extremo a extremo.
Latencia p99	< 100 ms	Es el número que se compromete. La media no se compromete nunca.
Latencia p99.9	< 250 ms	Revela problemas de partición y de recolección de basura que el p99 esconde.
Lag de sincronización p99	< 2 s para T1	La métrica de salud del DIH. Si solo puedes tener un gráfico, que sea este.
Disponibilidad del canal	99,99 %	Incluyendo la ventana de cierre del core. Ese es el punto del patrón.
RPO	0	El sistema de registro es la verdad; todo es reconstruible. Si tu RPO no es cero, tienes datos que solo viven en el DIH — ver anti-patrón 2.
RTO	< 5 min	Mediante persistencia y recarga paralela. Se verifica en simulacro, no se declara en un documento.
Tasa de divergencia	< 0,001 %	Sobre muestreo estratificado. Cualquier divergencia en T0 o T1 es incidente.
Profundidad de DLQ	0 en estado estable	Con alerta a partir del primer mensaje y procedimiento de reproceso documentado.
Costo por millón de consultas	Comparado contra el costo del core	La métrica que sostiene el presupuesto año tras año. Capítulo 22.

Qué instrumentar

Observability-first significa que la instrumentación es parte del diseño, no un añadido. Los cuatro planos que necesitas desde el primer día:

- **Trazas distribuidas** con contexto propagado desde el borde hasta el nodo del grid. Sin esto, diagnosticar un p99 alto es adivinar.
- **Métricas de pipeline:** lag por topic y por partición, tasa de eventos, latencia de transformación, tasa de error, profundidad de DLQ.
- **Métricas de grid:** ocupación de memoria por mapa, distribución de particiones, tasa de expulsión, latencia de operación por tipo, salud de réplicas.
- **Métricas de negocio:** divergencias detectadas, eventos de degradación, proporción de consultas servidas desde cada nivel de almacenamiento.

EL GRÁFICO ÚNICO

Si el comité ejecutivo solo va a mirar un número de este sistema, que sea el **lag de sincronización p99 contra su umbral**. Es la única métrica que traduce directamente a «¿está el banco mostrando la verdad?», y es comprensible sin conocimiento técnico.

FinOps del DIH

El patrón nace de un argumento económico. Si no se gobierna su propio costo, el argumento se invierte en el segundo año y el proyecto se vuelve indefendible.

Estructura de costo

COMPONENTE	PESO TÍPICO	PALANCA DE CONTROL
Memoria del grid	45 – 65 %	Políticas de nivel y TTL. Es la palanca dominante y la más ignorada.
Cómputo de nodos	15 – 25 %	Dimensionamiento por pico real medido, con autoescalado donde el motor lo permita.
Backbone de eventos	10 – 20 %	Retención por topic. Un topic con retención infinita cuesta para siempre.
Licenciamiento de CDC	5 – 15 %	Número de orígenes y volumen. Debezium sobre Kafka Connect elimina esta línea a cambio de operación propia.
Egreso y replicación	3 – 8 %	Topología multizona y multirregión. Crece rápido y sin avisar.

La comparación que sostiene el presupuesto

Costo unitario del core = costo mensual de capacidad / transacciones de lectura atendidas
 Costo unitario del DIH = costo mensual total DIH / consultas servidas
 Retorno DIH = (unitario_core - unitario_DIH) × consultas desviadas - costo fijo DIH

Este cálculo se rehace cada trimestre y se publica. Cuando el numerador se degrada — porque el grid creció sin control— la conversación ocurre antes de que alguien la abra en un comité de recorte presupuestal.

LA TRAMPA DEL AHORRO NO REALIZADO

Reducir el ochenta por ciento de las lecturas al core no reduce la factura del core en un ochenta por ciento si el contrato es por capacidad contratada y no por consumo, o si la capacidad liberada no se puede desprovisionar. **Lee el contrato de licenciamiento antes de prometer.** Cuando el ahorro directo no es realizable, el argumento sigue siendo válido pero cambia de forma: se convierte en capacidad de crecimiento sin ampliar el core, que es un ahorro diferido y perfectamente defendible si se presenta como tal.

Capítulo 23

Modelo operativo y organización

La arquitectura sobrevive o muere según cómo se organice el equipo. El DIH es especialmente sensible porque su modo de fallo organizacional —la centralización— es también su configuración inicial natural.

La trayectoria correcta

ETAPA	CONFIGURACIÓN DEL EQUIPO	RIESGO DOMINANTE
Fases 0 – 1	Un equipo central construye todo. Es correcto y necesario: hay que descubrir el patrón antes de poder enseñarlo.	Ninguno todavía. La centralización es apropiada mientras se aprende.
Fase 2	El equipo central construye con uno o dos equipos de dominio en modalidad de acompañamiento. Empieza a extraerse la plantilla.	Quedarse aquí. Es cómodo y funciona lo suficiente como para no doler durante un año.
Fase 3	Equipos de dominio publican sus propias vistas. El equipo central mantiene la plataforma, las plantillas, la observabilidad y el gobierno del canónico.	Federación sin gobierno: cada dominio inventa su propio canónico y se pierde la interoperabilidad.

Qué queda en el centro y qué se federa

RESPONSABILIDAD	DUEÑO	RAZÓN
Plataforma de grid y backbone	Central	Economía de escala y experticia operativa concentrada.
Gobierno del modelo canónico	Central	La interoperabilidad entre dominios se pierde si cada uno decide por su cuenta.
Plantillas y camino pavimentado	Central	Es el producto que el equipo central le vende internamente a los demás.
Observabilidad y SLO de plataforma	Central	Visión transversal necesaria para detectar problemas sistémicos.
Proyecciones y vistas del dominio	Dominio	Quien conoce el negocio modela el agregado. Nadie más puede hacerlo bien.
Clase de frescura de sus datos	Dominio + negocio	Es una decisión de riesgo de negocio, no de arquitectura.
Contrato de API de su vista	Dominio	Acoplado a los consumidores del dominio, no a la plataforma.

El objetivo del equipo de plataforma no es construir el DIH. Es llegar al punto en que ya no tenga que construirlo nadie más que los dueños de cada dominio.

Parte VIII

Aprender

Una ruta de ocho semanas, la bibliografía disponible con su nivel de calidad declarado, y el glosario que necesitas para leer el corpus en inglés sin traducirlo mal.

Capítulo 24

Plan de estudio de ocho semanas

Seis horas semanales, con un entregable propio por semana. El entregable importa más que la lectura: este patrón no se aprende leyendo.

SEMANA	FOCO	ENTREGABLE PROPIO
1	Patrones fundacionales: CQRS, vista materializada, cache-aside	Tabla comparativa de los tres con criterio explícito de elección
2	Domain-Driven Design: agregados, bounded contexts, lenguaje ubicuo	Modelo canónico del agregado «Cuenta» con su frontera justificada
3	Comunicación asíncrona y eventos de integración	Diagrama del flujo CDC → canónico → grid con puntos de fallo marcados
4	Consistencia: CAP, PACELC, sagas, compensación	Taxonomía de frescura completa para un dominio real de tu banco
5	CDC práctico y streaming	PoC funcionando: Postgres → Debezium → Kafka → Redis, con proyección idempotente
6	Particionamiento, colocación por afinidad, niveles de almacenamiento	Diseño de clave de partición para vista 360, con análisis de distribución
7	Dominio: Bre-B, ISO 20022, marco regulatorio	Arquitectura de un directorio de llaves sobre DIH con NFR derivados del esquema
8	Strangler Fig y coexistencia de core	Diagrama C4 de coexistencia legacy ↔ core nuevo vía DIH

LA SEMANA 5 NO ES NEGOCIABLE

El momento en que entiendes este patrón de verdad no llega leyendo. Llega el día que ves tu primer evento de CDC aplicarse fuera de orden, revisas tu saldo materializado, y descubres que está mal. Ese día aprendes más que en quinientas páginas. Monta la PoC aunque sea con contenedores en tu portátil.

Capítulo 25

Bibliografía comentada

Con el nivel de calidad y el sesgo de cada fuente declarados. La escasez en español es real; la estrategia es estudiar los pilares en español y leer el corpus específico del patrón en inglés con criterio.

EN ESPAÑOL — GRATUITO Y DE CALIDAD

FUENTE	CALIDAD	PARA QUÉ SIRVE
Azure Architecture Center — catálogo de patrones de diseño learn.microsoft.com/es-es/azure/architecture/patterns/	Alta	El recurso más denso en español. CQRS, vista materializada, cache-aside, event sourcing, fragmentación, tabla de índices, saga, higuera estranguladora, publicador-suscriptor, interruptor de circuito, BFF. Es la columna vertebral conceptual del DIH.
Microservicios de .NET: arquitectura para aplicaciones en contenedor — Microsoft learn.microsoft.com/es-es/dotnet/architecture/microservices/	Alta	Aproximadamente 300 páginas gratuitas con aplicación de referencia. Cubre DDD, agregados, bus de eventos, CQRS con consultas simplificadas, consistencia eventual y resiliencia. El PDF descargable solo está en inglés; la versión web sí está en español.
Documento técnico de Bre-B — Banco de la República, febrero de 2026 banrep.gov.co/publicaciones e investigaciones documentos técnicos	Muy alta	El mejor caso de estudio de arquitectura de pagos inmediatos disponible en español y en nuestra jurisdicción. Diseño, DICE, MOL, estándares operativos, cifras de adopción. Léelo con ojos de arquitecto.
Minisito y normatividad de SPBVI — Banco de la República banrep.gov.co/es/bre-b	Alta	Reglas técnicas y de usabilidad, manual de uso, parámetros de tiempo y límites. Fuente normativa directa para derivar NFR.
CEMLA — publicaciones sobre sistemas de pago	Media-alta	Investigación en español sobre pagos inmediatos e interoperabilidad en América Latina. Contexto regional comparado.
Superintendencia Financiera de Colombia — circulares de riesgo operativo, continuidad y seguridad de la información	Alta	Aburrido y obligatorio. Define buena parte de tus objetivos de disponibilidad y tus obligaciones de reporte.

ADVERTENCIA LINGÜÍSTICA SOBRE MICROSOFT LEARN

Buena parte del contenido en español es traducción automática. Vas a encontrar «coherencia» donde debe decir *consistencia*, y calcos como «contenerizadas». No te fíes de la terminología traducida: fíate de los diagramas y de la lógica. Cuando una frase chirríe, cambia /es-es/ por /en-us/ en la URL y contrasta. El glosario del capítulo 26 documenta las traducciones correctas.

EN INGLÉS — EL CORPUS ESPECÍFICO DEL PATRÓN

FUENTE	SESGO	CÓMO LEERLA
Gartner — definición y notas de investigación sobre Digital Integration Hub	Analista	Origen del término. Útil para el vocabulario y para citar en comités. Requiere suscripción para el material completo.
Hazelcast — <i>What is a Digital Integration Hub</i> y material de fundamentos	Fabricante	Buena explicación conceptual del patrón como capa de datos con acceso centralizado a fuentes dispares. Gratuito. Descarta las secciones de producto.
GigaSpaces — blog de arquitectura DIH	Fabricante fuerte	El material más extenso disponible y el más sesgado. Reconoce explícitamente que otros fabricantes ofrecen un enfoque de «hágalo usted mismo» frente a su producto integrado. Léelo por los diagramas y desconfía de la comparativa.
Documentación de Debezium — capítulos conceptuales de CDC y garantías de entrega	Proyecto abierto	Aproximadamente treinta páginas que dan el ochenta por ciento del criterio sobre sincronización. La lectura técnica con mejor relación valor-esfuerzo de toda la lista.
Pat Helland — <i>Life Beyond Distributed Transactions</i>	Ninguno	Doce páginas. El mejor texto que existe sobre por qué este patrón funciona. Si lees un solo paper, este.
Eric Brewer — <i>CAP Twelve Years Later</i>	Ninguno	El autor del teorema corrigiendo las malinterpretaciones más comunes. Corto y clarificador.
Daniel Abadi — PACELC	Ninguno	La extensión que introduce el compromiso latencia-consistencia en ausencia de partición, que es la decisión real de un DIH.
BIAN — Service Landscape y Semantic APIs	Consortio	Registro gratuito. El vocabulario de dominio y el criterio de federación. Sin versión en español que valga la pena.
ISO 20022 — catálogo de mensajes y diccionario de datos	Organismo	Acceso libre en iso20022.org. Fuente del canónico para todo lo que sea pagos.
Confluent Developer — fundamentos de Kafka y patrones dirigidos por eventos	Fabricante moderado	Cursos gratuitos de buena calidad técnica. El material conceptual es sólido y transferible a otros backbones.

Capítulo 26

Glosario EN→ES

Vas a tener que leer en inglés. Este glosario documenta la traducción correcta y la traducción automática frecuente que hay que evitar, porque en este dominio una palabra mal traducida introduce un riesgo operativo.

INGLÉS	ESPAÑOL CORRECTO	EVITAR Y POR QUÉ
Consistency	Consistencia	<i>Coherencia</i> — en español técnico «coherencia» apunta a ausencia de contradicción lógica, no a la propiedad de sistemas distribuidos. La confusión es sistemática en Microsoft Learn.
Eventual consistency	Consistencia eventual	<i>Coherencia final</i> — además de la anterior, «final» sugiere un estado terminal garantizado en un plazo, que no es lo que el término significa.
Staleness	Desactualización, caducidad del dato	No tiene traducción de una palabra. Preferir la perífrasis a un calco.
System of Record (SoR)	Sistema de registro, sistema autoritativo	<i>Sistema de registros</i> — el plural cambia el sentido de «registro» de función a colección.
Read model	Modelo de lectura, vista de consulta	—
In-Memory Data Grid	Malla de datos en memoria	<i>Rejilla de datos</i> — «rejilla» evoca una estructura física, no una topología distribuida.
Change Data Capture	Captura de cambios de datos	En la práctica se usa la sigla CDC sin traducir. Es aceptable y preferible.
Write-behind	Escritura diferida	Traducciones literales. Mantener el término en inglés en documentación técnica.
Read-through	Lectura pasante	Igual que el anterior: preferible mantenerlo en inglés y definirlo una vez.
Backpressure	Contrapresión	—
Throughput	Rendimiento, caudal	Cuidado: «rendimiento» también traduce <i>performance</i> . Desambiguar en contexto o mantener el inglés.
Latency percentile	Percentil de latencia	—
Stand-in processing	Procesamiento sustitutivo	Es término de industria de tarjetas. Suele dejarse en inglés.
Shard / Sharding	Fragmento / fragmentación	Aceptable, aunque «partición» es más claro en contexto de grid.
Affinity key	Clave de afinidad	—
Materialized view	Vista materializada	—

INGLÉS	ESPAÑOL CORRECTO	EVITAR Y POR QUÉ
Bounded context	Contexto delimitado	La traducción es correcta pero poco usada. En la práctica se mantiene en inglés.
Strangler Fig	Higuera estranguladora	Traducción correcta y usada por Microsoft Learn. Conviene mencionar el original la primera vez.
Dead Letter Queue	Cola de mensajes fallidos	La sigla DLQ es de uso corriente y preferible.
Idempotency	Idempotencia	—
Crypto-shredding	Destrucción por clave criptográfica	Sin traducción establecida. Definirlo la primera vez y usar el inglés.
Training-serving skew	Desfase entre entrenamiento y servicio	Sin traducción establecida en la industria.

A n e x o s

Plantillas de trabajo

Cinco artefactos listos para usar en una sesión de arquitectura. Copiar, llenar, firmar.

Anexo A

Plantilla de taxonomía de frescura

Una fila por dato y por caso de uso. El mismo campo aparece varias veces con clases distintas y eso es correcto. La columna de dueño de negocio no es opcional.

DATO	CASO DE USO	CLASE	TOLERANCIA	DUEÑO DE NEGOCIO	ORIGEN	DEGRADACIÓN
Saldo disponible	Pantalla de inicio	T1	5 s	Gerencia de canales	CDC core depósitos	Mostrar aviso con marca de tiempo
Saldo disponible	Autorización de transferencia	T0	0	Riesgo operativo	Read-through al core	Rechazar la operación
Movimientos	Extracto en pantalla	T2	5 min	Gerencia de canales	CDC core depósitos	Servir con marca de tiempo
Datos de contacto	Perfil del cliente	T3	24 h	Gerencia de clientes	ETL nocturno CRM	Servir sin aviso

Anexo B

Checklist de revisión de arquitectura

Para usar en revisión de diseño antes de aprobar una iniciativa de DIH o una vista nueva. Toda respuesta negativa requiere justificación escrita, no explicación verbal.

GOBIERNO

1. ¿Existe un modelo canónico documentado y versionado para cada dominio ingerido?
2. ¿Cada dato tiene clase de frescura asignada y dueño de negocio nombrado?
3. ¿La taxonomía de frescura está firmada por negocio y por riesgo operativo?
4. ¿Está definido quién aprueba el alta de un dato nuevo en el DIH?
5. ¿Existe un catálogo consultable de las vistas publicadas y sus contratos?

SINCRONIZACIÓN

1. ¿Toda proyección es idempotente, con clave de idempotencia proveniente del origen?
2. ¿El particionamiento garantiza orden para los eventos que lo requieren?
3. ¿Existe DLQ con alerta y procedimiento de reproceso documentado?
4. ¿Hay Schema Registry con política de compatibilidad forzada?
5. ¿La transición snapshot → CDC está diseñada sin ventana de pérdida?
6. ¿Existe capa de transformación a canónico, separada del CDC crudo?
7. ¿El procedimiento de reconstrucción total está documentado y probado?

ESTADO Y MODELADO

1. ¿La clave de partición corresponde al patrón de acceso dominante?
2. ¿Los datos que se consultan juntos están colocados por afinidad?
3. ¿Existen políticas de TTL, expulsión y niveles de almacenamiento?
4. ¿Se midió la distribución de particiones para detectar claves calientes?
5. ¿Hay algún dato que solo exista en el DIH? (si sí, ver anti-patrón 2)

CONSISTENCIA Y OPERACIÓN

1. ¿Existe reconciliación activa con muestreo estratificado?
2. ¿La divergencia genera alerta accionable con dueño, no un reporte?
3. ¿Está implementado el comportamiento de degradación por clase?
4. ¿Las respuestas incluyen marca de frescura y bandera de degradación?
5. ¿El lag de sincronización está monitorizado por topic y por partición?
6. ¿Se fijaron objetivos en p99 y p99.9, medidos en el borde?

7. ¿Existe algún camino de escritura write-behind sobre datos financieros?
8. ¿Toda reserva de fondos tiene TTL y compensación probada?

SEGURIDAD Y CUMPLIMIENTO

1. ¿TLS mutuo entre nodos y entre clientes y cluster?
2. ¿Cifrado en reposo en persistencia, niveles de disco y topics?
3. ¿El PAN está tokenizado, sin excepciones?
4. ¿RBAC por mapa o espacio de nombres, con menor privilegio?
5. ¿Existe registro de acceso a datos sensibles?
6. ¿El derecho de supresión es ejecutable incluyendo respaldos y topics?
7. ¿Se verificó que no hay datos productivos en entornos no productivos?
8. ¿RPO y RTO fueron verificados en simulacro, no solo declarados?

ECONOMÍA Y ORGANIZACIÓN

1. ¿Se midió el costo unitario del DIH contra el del sistema de registro?
2. ¿Se leyó el contrato de licenciamiento del core antes de prometer ahorro?
3. ¿Existe revisión trimestral publicada del modelo de costo?
4. ¿Hay un plan explícito para pasar de equipo central a modelo federado?
5. ¿Existe camino pavimentado para que un dominio publique sin el centro?

Anexo C

Matriz de decisión tecnológica

Se puntúa cada criterio de 1 a 5 según la importancia para tu caso, y se evalúa cada candidato. El resultado no decide por ti; hace explícito el compromiso que estás aceptando.

CRITERIO	PESO (1-5)	QUÉ OBSERVAR EN LA PRUEBA DE CONCEPTO
Latencia p99 con agregado real		Con tu tamaño de objeto, tu concurrencia de pico y las réplicas activas. No con el dato de la demo.
Cómputo colocado		¿Soporta funciones del lado servidor con aislamiento y límite de tiempo? Determina la viabilidad del capítulo 7.
Modelo de consulta		¿Solo por clave, o índices secundarios y consulta tipo SQL? Condiciona cuántas proyecciones necesitas.
Atomicidad en partición		Requisito duro si vas a implementar reserva de fondos.
Niveles de almacenamiento		¿Soporta memoria + disco de forma transparente? Es la palanca principal de costo.
Replicación multizona y multi-región		Topología, modo de consistencia y costo de egreso asociado.
Tiempo de recuperación		¿Cuánto tarda en recargar tu volumen real tras caída total? Es tu RTO efectivo.
Integración con CDC		¿Conectores nativos o hay que construirlos? Impacta directamente el cronograma.
Observabilidad nativa		Métricas por partición y por mapa, exportables a tu plataforma existente.
Controles de seguridad		mTLS entre nodos, RBAC granular, cifrado en reposo, registro de auditoría.
Modelo operativo		¿Gestionado o autoadministrado? ¿Tienes el equipo para operarlo a las tres de la mañana?
Costo total a tres años		Licencia + infraestructura + operación. Con tu proyección de crecimiento, no con la actual.
Soporte regional		Zona horaria, idioma y experiencia previa del proveedor en banca de tu país.
Estrategia de salida		¿Qué cuesta migrar a otro motor en tres años? El canónico te protege; el modelo de consulta propietario no.

Anexo D

Preguntas para el comité de negocio

Las preguntas de la sesión de descubrimiento de la fase 0. Están formuladas para que negocio y riesgo puedan responderlas sin conocimiento técnico, y para que las respuestas se traduzcan directamente en requisitos.

SOBRE TOLERANCIA Y RIESGO

1. Si este dato tuviera treinta segundos de retraso, ¿qué puede salir mal, para quién, y cuánto cuesta?
2. ¿Existe alguna decisión automática que se tome usando este dato? ¿Cuál y con qué consecuencia?
3. ¿Hay una obligación regulatoria o contractual sobre la actualidad de este dato?
4. Si el sistema tuviera que elegir entre mostrar un dato de hace un minuto o no mostrar nada, ¿qué prefieres?

SOBRE EL ESTADO ACTUAL

1. ¿Cuántas veces al mes consulta un cliente típico esta información?
2. ¿Cuál es la relación entre el día pico y el día promedio?
3. ¿Qué pasa hoy con esta consulta durante la ventana de cierre del core?
4. ¿Cuántas quejas al mes se relacionan con lentitud o indisponibilidad de esta función?

SOBRE EL COSTO Y EL CONTRATO

1. ¿El licenciamiento del core es por capacidad contratada o por consumo?
2. Si liberamos capacidad, ¿es desprovisionable o queda ociosa?
3. ¿Hay una ampliación de capacidad del core presupuestada en los próximos veinticuatro meses?

SOBRE PROPIEDAD

1. ¿Quién es el dueño de negocio de este dato y firma su clasificación?
2. ¿Quién decide, y en cuánto tiempo, cuando la reconciliación detecta una divergencia?
3. ¿Quién autoriza que el sistema opere en modo degradado y bajo qué criterio?

Anexo E

Plantilla de ADR

Registro de decisión de arquitectura, adaptado a las decisiones típicas de un DIH. Una decisión por documento, inmutable una vez aceptada; si cambia, se emite una nueva que supersede la anterior.

```
# ADR-0000: [Título en una línea, en imperativo]

## Estado
Propuesta | Aceptada | Superseded by ADR-XXXX | Deprecada
Fecha: AAAA-MM-DD   Autor: [nombre]   Revisores: [nombres]

## Contexto
Qué situación motiva la decisión. Restricciones reales: regulatorias,
de contrato, de equipo, de plazo. Sin adjetivos.

## Decisión
Qué se decide, en voz activa. "Usaremos X para Y."

## Clase de frescura afectada
T0 | T1 | T2 | T3 – y qué datos concretos.

## Consecuencias
### Positivas
### Negativas
### Riesgos aceptados y su mitigación

## Alternativas consideradas
Cada una con la razón concreta del descarte. Si no hubo
alternativas evaluadas, dilo – es información valiosa.

## Impacto en el modelo canónico
¿Introduce, modifica o versiona alguna estructura del canónico?

## Impacto en consumidores
¿Requiere cambio en algún consumidor? ¿Es compatible hacia atrás?

## Cómo se verifica
Qué métrica o prueba demuestra que la decisión fue correcta,
y en qué plazo se revisa.
```

LAS CINCO DECISIONES QUE SIEMPRE NECESITAN ADR

La elección del motor de estado. La clave de partición de cada dominio. El mecanismo de sincronización por origen. La política de niveles de almacenamiento. Y cualquier excepción a las reglas duras de este libro —especialmente si alguien propone write-behind sobre un dato financiero o almacenar en el DIH algo que no existe en un sistema de registro.

• • •

Things are only impossible until they're not.

— Jean-Luc Picard