Ecosistema de Integración Multicore y Open Finance

Autor: Iván Lima

Rol: Arquitecto de Integración

Fecha: 18/02/2026

Índice

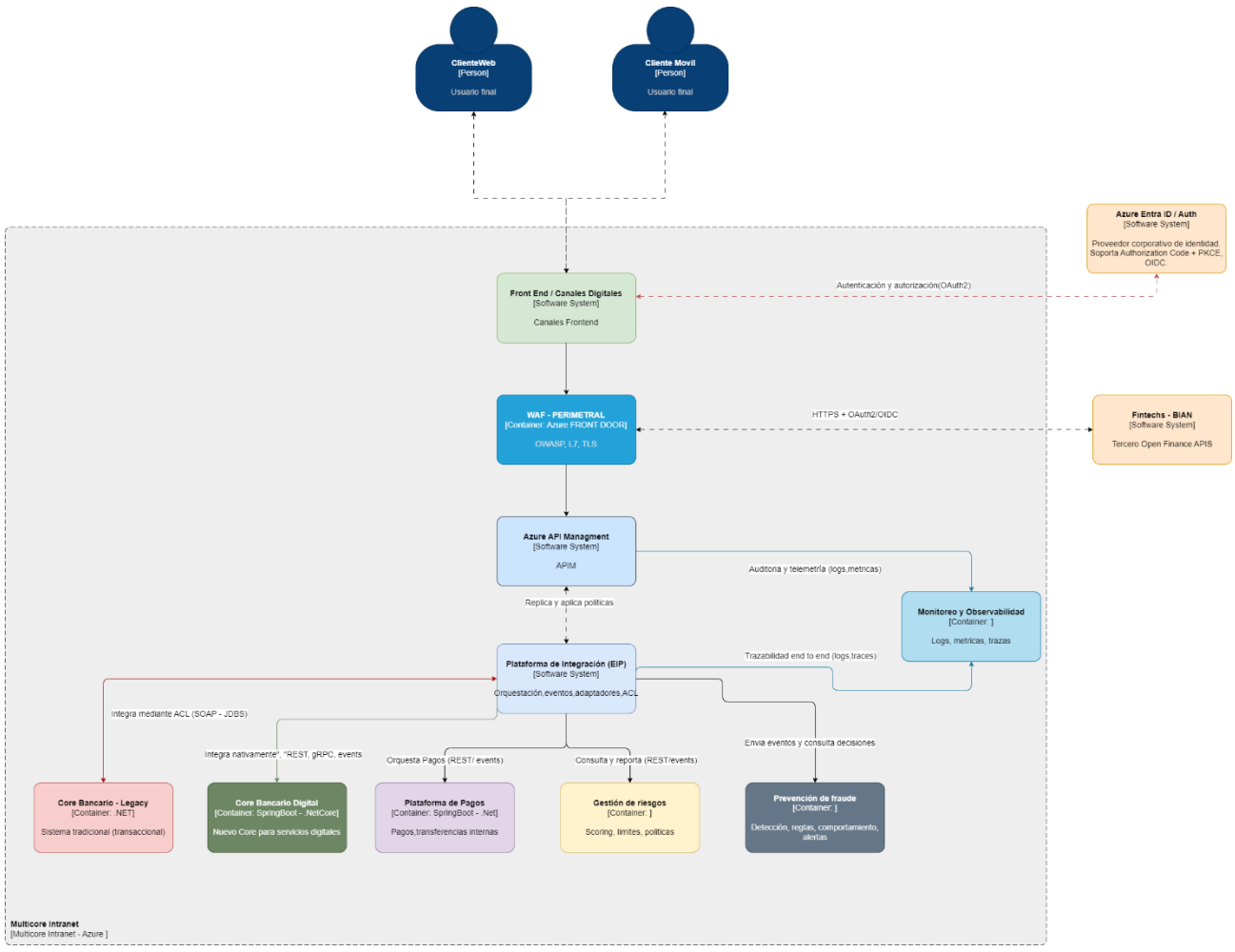
Contenido

Resumen Ejecutivo

Se propone una arquitectura de integración para un banco en transición hacia un modelo multicore y Open Finance, manteniendo continuidad operativa, seguridad regulatoria y baja latencia. La solución posiciona una Plataforma de Integración Empresarial (EIP) detrás de una capa perimetral (WAF) y un API Gateway (APIM), con orquestación síncrona hacia cores y propagación asíncrona de efectos secundarios mediante Event-Driven Architecture soportada por patrón Outbox.

La estrategia de modernización minimiza riesgo operativo aplicando Strangler Fig y ruteo inteligente dentro de la EIP, permitiendo convivir con un Core Legacy (SOAP/WSDL) y un Core Digital (REST) sin que los consumidores cambien su forma de integración.

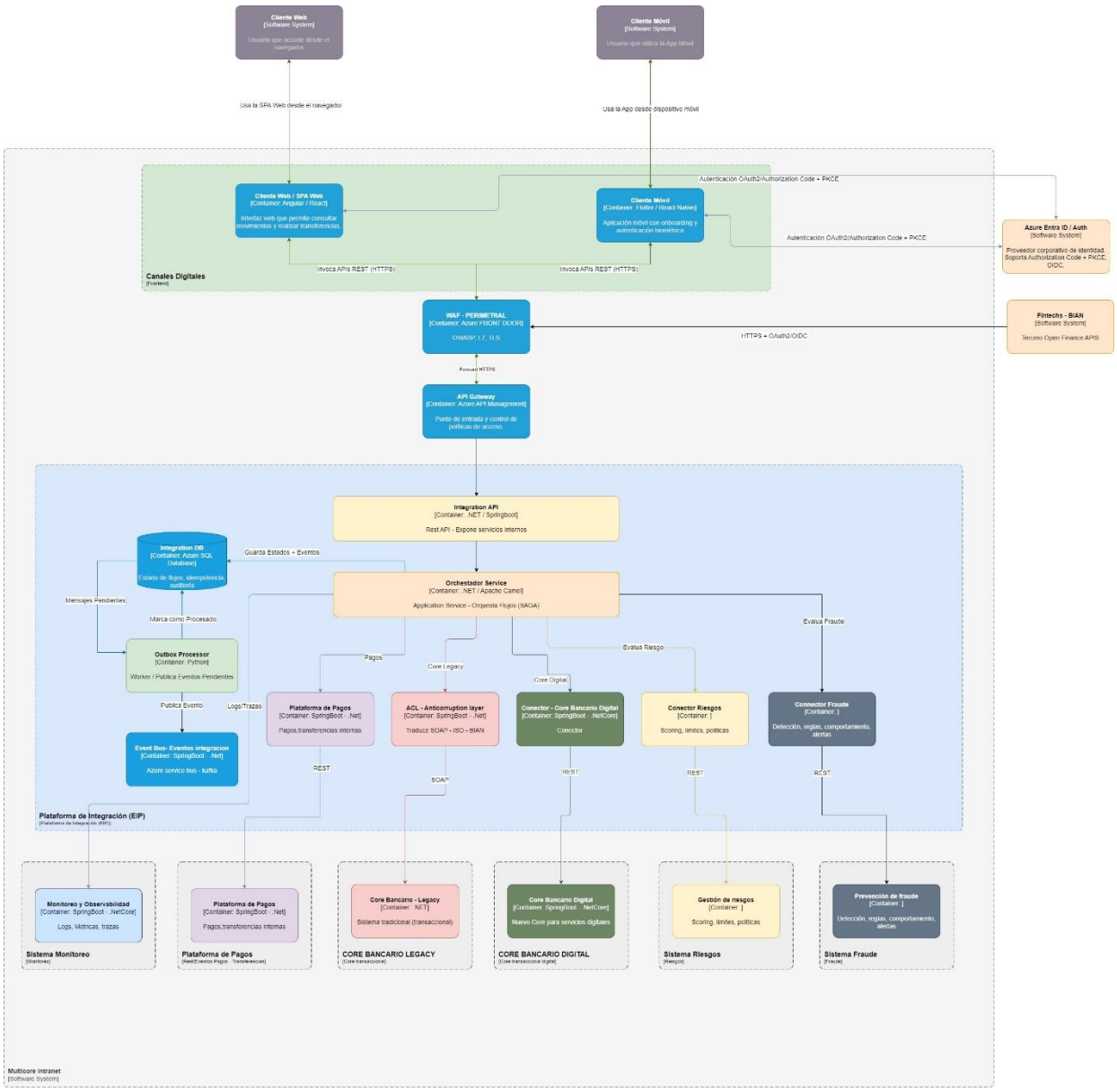
Modelo C4 – Arquitectura General



Lectura técnica del C1

- **WAF (Azure Front Door + WAF)**: primera barrera L7 para todo tráfico público (Web/Móvil/Fintech).
- **APIM**: gobierno y control de acceso (OAuth2/OIDC, scopes, rate limits, versionamiento, policy enforcement).
- **EIP**: capa de integración/orquestación que contiene la lógica multicore, ACL y conectores.
- **Core Legacy (SOAP/WSDL)**: transacciones críticas síncronas vía ACL.
- **Core Digital (REST)**: transacciones críticas síncronas vía conector.
- **Event Bus**: propagación asíncrona (notificación, auditoría, analítica, sincronización satélite).

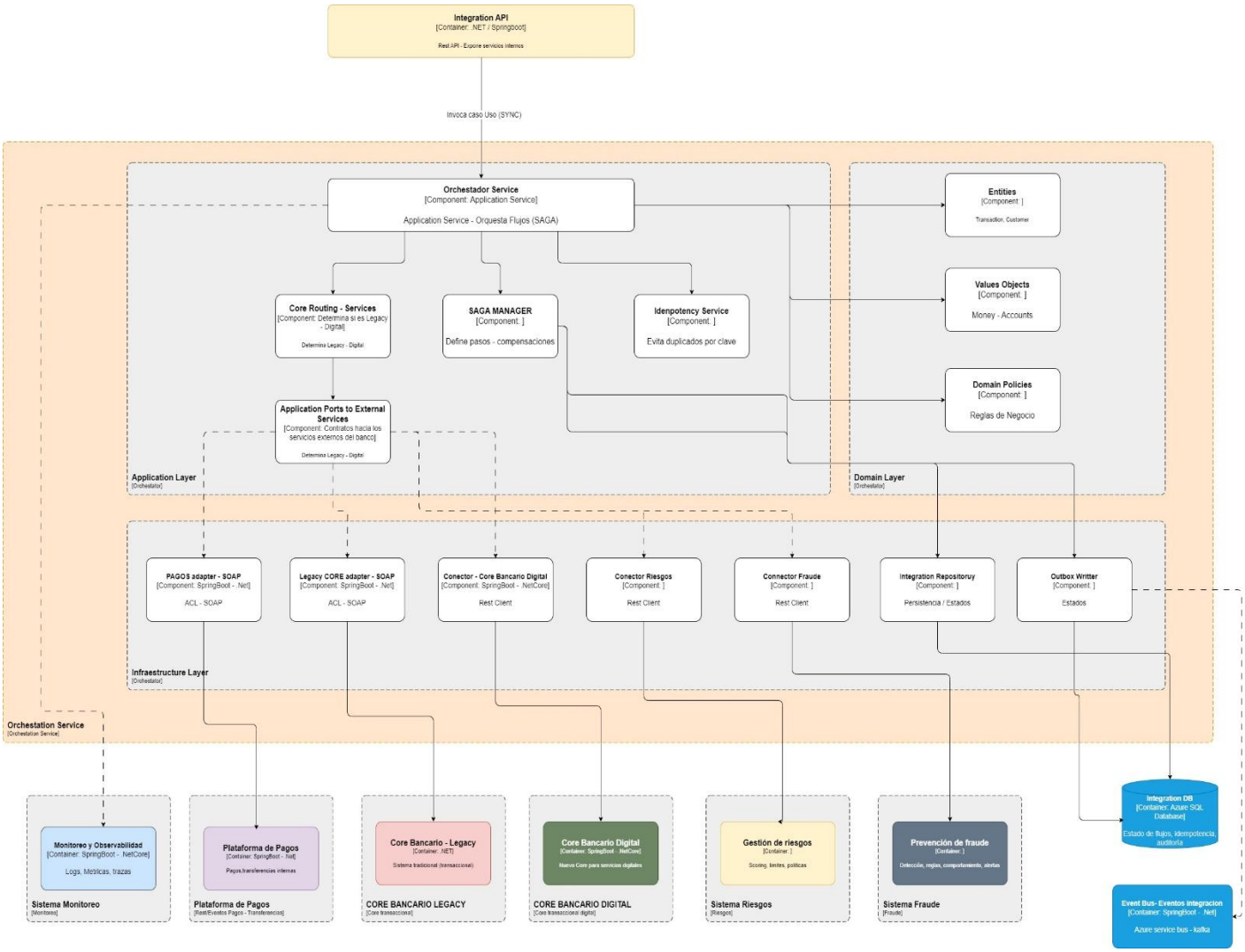
C4 Nivel 2 – Contenedores



Lectura técnica del C2

- **Integration API** expone endpoints de negocio (API First) sin acoplarse a la decisión multicore.
- **Orchestrator Service** concentra lógica de aplicación (Saga), routing multicore, idempotencia, resiliencia y escritura de Outbox.
- **Integration DB** guarda estado de saga/operación, auditoría funcional e Outbox.
- **Outbox Processor** publica eventos a Event Bus con garantías de reintento y sin duplicidad.
- **Conectores** (Legacy/Digital/Pagos/Riesgo/Fraude) encapsulan protocolos, timeouts, mapeos y políticas técnicas.
- **Event Bus** desacopla consumidores secundarios y permite reintentos/DLQ.

C4 Nivel 3 – Componentes



Lectura técnica del C3

- **Clean Architecture**: Application define puertos (interfaces), Infrastructure implementa conectores/adaptadores, Domain concentra reglas.
- **CoreRoutingService** decide destino (legacy/digital) basado en reglas de migración/configuración.
- **SagaManager** mantiene consistencia distribuida y registra estados transitorios.
- **OutboxWriter** garantiza publicación eventual (DB como source of truth).

Patrones de Integración y Tecnologías

API Gateway Pattern (APIM)

Dónde: APIM, detrás de WAF.
Por qué: centraliza enforcement de políticas transversales sin contaminar el dominio:

- **AuthN/AuthZ**: validación JWT, scopes, audience, issuer.
- **Quotas**: por subscription/fintech/producto.
- **Transformaciones ligeras**: headers de correlación, normalización básica.
- **Versionamiento**: /v1, /v2 y deprecación planificada.
- **Protección de backend**: límite de tamaño de payload, validación de content-type, bloqueo de métodos no permitidos.

Tecnologías: Azure API Management + Azure Entra ID.

Anti-Corruption Layer (ACL) hacia Legacy (SOAP/WSDL)

Dónde: Infrastructure Layer, "Legacy Core Adapter".
Por qué:

- Evita que contratos WSDL, códigos de error legacy y modelos XML contaminen el modelo moderno.
- Permite mapear: **DTO moderno -> request SOAP** y **response SOAP -> modelo interno**.
- Unifica manejo de errores: timeouts, faults SOAP, errores funcionales (códigos legacy) hacia un catálogo interno de errores.

Mecanismos técnicos:

- Serialización XML controlada (schemas).
- Catálogo de errores: LEGACY_TIMEOUT, LEGACY_FUNCTIONAL_REJECT, LEGACY_SCHEMA_ERROR.
- Normalización de campos sensibles (enmascaramiento) antes de loguear

Saga Pattern (Orquestación síncrona)

Dónde: Orchestrator (Application Layer).
Por qué:

- La operación financiera requiere **consistencia fuerte** y confirmación al cliente.
- Se ejecutan pasos síncronos: validaciones, consulta de saldos/cuentas, ejecución de débito/crédito, confirmaciones.
- Cuando hay pasos que fallan después de acciones previas, se modelan **compensaciones** (si aplica por dominio).

Outbox Pattern + Event-Driven (Consistencia eventual controlada)

Dónde: Integration DB (tabla Outbox) + Outbox Processor.
Por qué:

- Evita inconsistencia “commit DB OK pero publish event falla”.
- La DB es repositorio de verdad; los eventos son propagación.
- Outbox Processor:
    - Polling (ej: cada 1-5s) o scheduling.
    - Publica y marca “Published”.
    - Backoff exponencial cuando falla.
    - Si excede reintentos, marca Failed y levanta alerta.

Event Bus:

- Topics por dominio: payments.*, accounts.*, risk.*.
- **DLQ** habilitada.
- **Duplicate detection** habilitado usando MessageId = OutboxId.
- CorrelationId propagado end-to-end.

Circuit Breaker, Timeout, Retry (Resiliencia)

Dónde: Conectores (Infrastructure Layer).

Por qué:

Proteger la EIP ante degradación de dependencias externas.

Mecanismo:

- Header Idempotency-Key obligatorio en operaciones monetarias.
- Persistencia de resultado por key + endpoint + tenant.
- Si llega key repetida, devolver el resultado previo.

Idempotency Pattern

Dónde: Orchestrator + DB/Redis
Por qué: evitar doble procesamiento.

Estrategia Multicore

Ruteo multicore

No vive en APIM. Vive en EIP en CoreRoutingService.

Motivo: el destino depende de reglas de negocio y estado de migración (cliente/producto/cuenta), no de infraestructura.

Criterios técnicos de enrutamiento (implementables)

- **Reglas por producto**: producto X migrado al Core Digital, producto Y permanece en legacy.
- **Reglas por cliente**: segmento/flag de migración del cliente.
- **Reglas por cuenta**: prefijo/rango, o catálogo “Account-to-Core”.
- **Feature flags**: canary gradual por porcentaje, controlado por configuración.

Cómo se implementa

- Tabla/servicio de configuración: CoreRoutingRules (DB/Redis).
- Cache local/Redis para baja latencia.
- Auditoría del ruteo: se registra targetCore por transacción.

IAM: Autenticación, Autorización y Control de Acceso.

Consumidores externos (Web/Móvil/Fintech)

- **OAuth2/OIDC** con Azure Entra ID.
- **SPA/Móvil**: Authorization Code + PKCE.
- **Fintech**: Client Credentials (según estándar del programa) y, preferiblemente, **mTLS**.

Autorización (APIM)

- Validación de:
    - issuer
    - audience
    - expiry
    - scopes
    - claims críticos
- Scopes por producto, ejemplo:
    - openfinance.accounts.read
    - openfinance.payments.initiate
- Rate limit por client_id y por endpoint crítico.

Autorización interna (EIP y conectores)

- **Managed Identity** o certificados internos (ideal) entre APIM/EIP/servicios.
- RBAC en AKS (menor privilegio).
- Secretos en Key Vault (si aplica).
- Separación de roles:
    - runtime
    - deploy
    - auditoría

Seguridad y Cumplimiento

Ley Orgánica de Protección de Datos Personales (LOPDP) – Implementación práctica

Se aplican controles alineados a principios de protección de datos:

- **Minimización**: solo atributos necesarios por operación.
- **Finalidad**: cada API documenta propósito y retención.
- **Seguridad**:
    - TLS 1.2+
    - cifrado en reposo (TDE/Storage encryption)
- **Acceso**:
    - RBAC
    - segregación de ambientes
- **Auditoría**:
    - registro de accesos a datos sensibles
    - trazabilidad por CorrelationId
- **Privacidad por diseño**:
    - logs sin PII (enmascaramiento)
    - tokenización o hashing donde aplique

Clasificación de datos

Los datos se clasifican en:

- Datos públicos
- Datos internos
- Datos confidenciales
- Datos sensibles (PII financiera)

Los datos sensibles:

- Nunca se almacenan en logs.
- Se tokenizan.
- Se cifran en tránsito y reposo.
- Se limitan por principio de menor privilegio.

Estrategia de APIs internas y externas

Externas (Open Finance)

- REST + JSON.
- OpenAPI 3.x.
- Versionamiento.
- mTLS (recomendado) + OAuth2.
- Consentimiento (si el programa lo requiere) como capa adicional.

Internas

- Sync REST para pasos críticos.
- Async events para efectos secundarios.

Seguridad perimetral (WAF)

- OWASP ruleset.
- Bot protection.
- Request body inspection y size limits.
- Geo/IP restrictions (solo si el programa lo requiere).
- Integración con SIEM.

Gobierno (APIs y microservicios)

Gobierno de APIs (APIM)

- Catálogo y productos por dominio.
- Políticas base obligatorias:
    - auth
    - rate limit
    - correlation id injection
    - request size limit
- Ciclo de vida:
    - Draft
    - Published
    - Deprecated
    - Retired
- Revisión de seguridad antes de publicar (WAF/APIM policies).

Gobierno de microservicios

- Convenciones:
    - logging estructurado
    - tracing
    - métricas
    - health endpoints

- CI/CD con quality gates.
- ADRs por decisión crítica (routing, outbox, core mapping).
- SLOs: latencia, errores, disponibilidad.

Alta Disponibilidad (HA) y Recuperación ante Desastres (DR)

HA por componente

- **WAF/Front Door**: servicio global, tolerancia regional.
- **APIM**: modo multi-zona (y multi-región si es requerido por criticidad).
- **AKS**:
    - node pools multi-zona
    - réplicas mínimas por servicio (>=2)
    - readiness/liveness probes
    - HPA basado en CPU/RPS
- **DB**:
    - HA nativa (zona redundante / business critical)
    - failover automático
- **Event Bus**:
    - namespaces con redundancia
    - DLQ y reintentos controlados

Definición de RPO y RTO en función del presupuesto

Los objetivos de Recuperación ante Desastres (RPO/RTO) dependen directamente del presupuesto y criticidad del sistema.

Existen tres posibles escenarios de implementación:

Escenario 1 – Alta Disponibilidad Regional (Costo Medio)

- Base de datos en modo Business Critical con replicación síncrona entre zonas.
- AKS desplegado en múltiples zonas de disponibilidad.
- Failover automático dentro de la misma región.

Objetivos estimados:

- RPO ≈ 0 segundos (replicación síncrona).
- RTO < 5 minutos.

Escenario 2 – DR Activo-Pasivo Multi-Región (Costo Alto)

- Replicación geográfica automática de base de datos.
- Clúster AKS preconfigurado en región secundaria.
- Sin tráfico activo hasta evento de desastre.
- Failover manual o semiautomático mediante Front Door / DNS.

Objetivos estimados:

- RPO < 5 minutos.
- RTO < 30 minutos.

Escenario 3 – DR Básico (Costo Controlado)

- Backups automáticos con retención.
- Infraestructura secundaria no activa.
- Recuperación mediante restore manual.

Objetivos estimados:

- RPO según frecuencia de backup (ej. 15–60 minutos).
- RTO varias horas.

En función del presupuesto del proyecto y criticidad regulatoria, se recomienda como mínimo:

- Base de datos en HA (zona redundante).
- AKS multi-zona.
- Replicación geográfica hacia sitio contingente.

Esto garantiza continuidad operativa sin sobrecostos innecesarios.

Plan de Migración Gradual (Strangler Fig) – minimización de riesgo

Fases

1. **Fase 0 – Baseline**: APIM/WAF, contratos API, observabilidad, correlación.
2. **Fase 1 – EIP como fachada**: canales apuntan a APIs unificadas (sin cambiar UX).
3. **Fase 2 – Routing multicore**: CoreRoutingService inicia con reglas simples por producto.
4. **Fase 3 – Migración por capacidades**: mover productos/operaciones a Core Digital gradualmente.
5. **Fase 4 – Optimización**: reducir dependencia del legacy, desactivar rutas legacy por producto.
6. **Fase 5 – Retiro controlado**: legacy sólo para los dominios no migrados o retiro progresivo por capacidad funcional.

Mecanismos de control de riesgo

- Feature flags y canary por porcentaje.
- Métricas por core:
    - latencia p95
    - tasa de errores
    - timeouts
- Rollback de routing por configuración sin despliegue.
- Auditoría de decisiones de ruteo por transacción.

Observabilidad y Monitoreo (técnico)

Objetivos de Nivel de Servicio (SLO) y Latencia

Con el fin de garantizar baja latencia y estabilidad operativa en un entorno bancario, se definen los siguientes Objetivos de Nivel de Servicio (SLO):

Disponibilidad

- Disponibilidad objetivo del servicio: ≥ 99.8% mensual.
- Disponibilidad del API Gateway: ≥ 99.95%.
- Disponibilidad de la base de datos: según tier seleccionado (Business Critical).

Latencia

Para operaciones financieras críticas (sin considerar latencia externa del Core):

- p95 < 300 ms para validaciones internas.
- p99 < 500 ms en condiciones normales.
- Timeout máximo end-to-end: 5 segundos.

Para operaciones que dependen de Core Legacy:

- Timeout configurable (2–5 segundos).
- Circuit Breaker activado ante degradación sostenida.

Métricas Clave Monitoreadas

- Latencia p50, p95 y p99 por endpoint.
- Latencia diferenciada por Core (Legacy vs Digital).
- Tasa de error por tipo (timeout, reject funcional, error técnico).
- Tamaño de backlog de Outbox.
- Profundidad de DLQ en Event Bus.

Gestión de Incidentes

- Alertas automáticas ante:
    - Incremento sostenido de p95.
    - Circuit Breaker abierto.
    - Aumento de mensajes en DLQ.

- Escalamiento automático según severidad.
- **Logs estructurados** con campos:
    - correlationId
    - transactionId
    - targetCore
    - idempotencyKey
    - resultCode
- **Tracing distribuido** (OpenTelemetry):
    - traceparent propagado desde APIM a EIP.
- **Métricas**:
    - RPS por endpoint
    - latencia p95/p99 por core
    - errores por tipo (timeout, circuit open, reject)
    - backlog de outbox pending
    - DLQ Depth

Los SLO definidos guían la configuración de HA y escalamiento automático, garantizando que la arquitectura mantenga latencia y disponibilidad dentro de umbrales aceptables.

Conclusión

La arquitectura propuesta cumple requerimientos de integración, seguridad, cumplimiento y continuidad operativa. El modelo multicore se implementa de forma controlada en la EIP, manteniendo operaciones críticas síncronas y propagando efectos secundarios mediante Event-Driven Architecture con Outbox. Se definen mecanismos concretos de IAM, gobierno, HA/DR y migración gradual, aptos para un entorno bancario regulado y de baja latencia.