BP – Internet Banking

Resumen de arquitectura (C4) y consideraciones del reto

1) Visión general (C1 – Contexto)

- **Actor principal:** Cliente (web/móvil) para consultar movimientos y ejecutar transferencias/pagos.
- **Sistema en foco: BP Internet Banking (IBCore)**.
- **Sistemas externos:**
    - **Core Bancario:** productos, saldos y transacciones históricas.
    - **Redes de pago/clearing:** transferencias interbancarias (ISO 20022/PSP).
    - **IdP OAuth2/OIDC:** autenticación y consentimiento (Authorization Code + PKCE).
    - **Riesgo/Fraude (AML/FRM):** validaciones en línea.
    - **Notificaciones:** email/SMS/push.
    - **KYC biométrico:** onboarding con verificación facial y prueba de vida.
    - **Reguladores (SB/UAFE):** reportes y alertas (ROS).
- **Relaciones clave:** IB se autentica contra IdP, consulta/ordena al Core, ejecuta pagos en la red, valida riesgo, notifica al cliente y reporta a reguladores.

2) Contenedores (C2 – Vista lógica)

Canales

- **SPA Web** y **App Móvil** (OIDC Code + PKCE).

Borde/seguridad

- **API Gateway + WAF:** auth, rate limiting, DDoS, mTLS, routing.
- **BFF Web/Móvil:** composición de vistas y caché por canal.
- **Policy/Enforcement (OPA):** ABAC/RBAC por claims y scopes, políticas en el edge.

Dominio

- **Onboarding/KYC Orchestration:** alta de cliente, binding con IdP y alta de datos.

- **Transferencias:** orquestación (sagas), idempotencia y límites.
- **Movimientos:** consolida históricos y estados.
- **Datos de Cliente:** perfil, productos, beneficiarios.
- **Notificaciones:** plantillas y envíos; suscriptor de eventos.

Datos/Eventos

- **TransfDB (PostgreSQL/Oracle)** como base de negocio de transferencias.
- **CDC (Debezium/Kafka Connect)** captura cambios de TransfDB (WAL/binlog) y publica a **Kafka** (Event Streaming).
- **Read-Models (CQRS):** Redis/DynamoDB para vistas de lectura (saldo, últimos N, beneficiarios).
- **Auditoría/Ledger inmutable:** almacenamiento append-only con retención WORM.

Transversal

- **Observabilidad (OTel + métricas + logs):** trazas, KPIs y tableros.
- **KMS/Vault:** llaves, secretos, firmas JWS.
- **Regulatory Reporting:** consume eventos/auditoría y entrega ROS/reportes.

Notas de diseño

- Tráfico **norte-sur** asegurado por WAF + mTLS; **este-oeste** con mTLS/mesh.
- **CDC desde la BD de Transferencias** para emitir eventos de dominio sin acoplar al servicio.
- **CQRS** para separar escritura (servicios) de lectura (read-models).

3) Componentes del Servicio de Transferencias (C3)

- **API de Transferencias (REST/gRPC):** validación, **idempotencia** por canal, **límites**.
- **Idempotency Manager (Redis):** claves idempotentes y locks con TTL.
- **Limits/Rules Engine:** límites diarios/beneficiario/horarios.
- **Saga/Orchestrator:** pasos de reserva, ejecución, confirmación y **compensaciones**.

- **Adapters:**
    - **Core:** consulta/reserva de saldo.
    - **Payments (ISO 20022/PSP):** instrucción de pago interbancario.
- **Tx Repository:** persistencia de estados de transferencia.
- **Outbox (opcional en C3) + Serializer:** inserta evento dentro de la misma TX y serializa.
- **CDC (externo):** publica cambios a Kafka; **Movimientos** y **Read-Models** los consumen.
- **Audit Append (fallback):** ruta directa a auditoría si ES está degradado.
- **OpenTelemetry:** trazabilidad end-to-end.

Observación: C2 adopta **CDC directo** desde la BD como estrategia base. C3 muestra **Outbox + CDC** para escenarios donde se prefiera la inserción de mensajes transaccionales; el objetivo es converger en **CDC sobre cambios de negocio** y mantener Outbox solo como transición/fallback.

4) Flujos críticos (resumen)

- **Onboarding:** App/SPA → BFF → Orquestador KYC → Proveedor biométrico → IdP (creación/binding) → Servicio de Cliente (alta) → Notificaciones.
- **Login/consentimiento:** OIDC Code + PKCE (navegador/system browser).
- **Consulta de movimientos:** BFF → Movimientos (read-models) → Core si hace falta enriquecimiento.
- **Transferencia interbancaria:**
    1. API valida esquema, límites e idempotencia.
    2. Saga reserva saldo en Core.
    3. Ejecuta instrucción en red de pagos.
    4. Persiste estado; **CDC** emite evento.
    5. Read-models se actualizan, Notificaciones envía aviso, Riesgo/Regulatorio consumen para alertas/reportes.

6. Errores activan **compensaciones** (liberación de reserva, reverso/ajuste).

5) Decisiones de diseño (y por qué)

- **BFF por canal:** desacopla UI, reduce chattiness y permite caché/compresión específicas.
- **CDC + Kafka:** desacopla escritura/lectura, habilita **event-driven** para notificaciones y reportería.
- **CQRS:** consultas rápidas y escalables sin impactar transacciones.
- **Idempotencia + límites:** evita duplicados y controla riesgo operativo/comercial.
- **OPA (ABAC/RBAC):** centraliza políticas y facilita auditoría de decisiones.
- **Ledger WORM:** soporte probatorio y cumplimiento regulatorio.
- **mTLS, PKCE y (opcional) DPoP:** protegen tokens y previenen replay.
- **Fallback de auditoría:** continuidad de evidencia si hay degradación del bus.

6) Requisitos no funcionales (NFR)

- **Seguridad:** cifrado en tránsito/repouso, rotación de secretos, mínimo privilegio, hardening del edge.
- **Disponibilidad:** objetivo ≥ **99,95%**; **RTO** ≤ 15 min; **RPO** ≤ 5 min.
- **Escalabilidad:** horizontal en BFFs y servicios; particionamiento por dominio; caché de lectura.
- **Desempeño:** p95 de transferencia ≤ 700 ms hasta encolado/ack; lectura de saldo p95 ≤ 150 ms.
- **Resiliencia:** timeouts, **circuit breakers**, reintentos con backoff, **bulkheads** y **idempotencia** en operaciones.
- **Observabilidad:** trazas distribuidas, KPIs de negocio (tasa de éxito, STP, reversos), SLOs y alertas.
- **Cumplimiento:** retención WORM, segregación de PII, anonimización para analytics, ROS oportunos.

7) Datos, consistencia y concurrencia

- **Modelo de estados de transferencia** (ej.: created → reserved → sent → confirmed/failed → compensated).
- **Consistencia final** en read-models; **consistencia fuerte** en operaciones críticas (reserva/ejecución).
- **Exactly-once lógico** vía idempotencia + claves/locks y procesamiento **al menos una vez** en consumidores.
- **Esquemas versionados** (compatibilidad hacia atrás en eventos).

8) Pruebas y calidad

- **Unitarias y de dominio** (reglas/estados).
- **Contract testing** (BFF↔servicios, servicios↔Core/PSP).
- **E2E** por flujos críticos (onboarding, transferencia).
- **Performance** (carga y estrés) y **chaos testing** (fallas de red/PSP/Kafka).
- **Seguridad:** SAST, DAST, escaneo de dependencias/containers y pruebas de autorización con OPA.

9) Entrega y operación (DevEx/SRE)

- **CI/CD** con pipelines por servicio, promoción entre ambientes y **feature flags**.
- **Despliegues** blue/green o canary; migraciones de esquema compatibles.
- **Runbooks & playbooks** para incidentes (PSP caído, latencia de Core, backlog de Kafka).
- **KPIs operativos:** latencia p95/p99, tasa de errores, saturación, cola de eventos, éxitos STP, tiempos de ROS.

10) Riesgos y mitigaciones

- **Duplicidad de órdenes:** idempotencia por canal + claves compuestas (canal+cliente+nonce).

- **Intermitencia en PSP/Core:** reintentos con backoff, colas diferidas, compensaciones.
- **Desfase de saldos en lectura:** read-models con **lag** monitoreado y botón de “forzar actualización” contra Core.
- **Degradación del bus: fallback** de auditoría y colas locales transitorias.
- **Políticas mal configuradas:** validación estática y pruebas de autorización en CI; modo “dry-run” en OPA antes de producción.