Arquitectura de Sistema de Banca por Internet - BP Bank

1. INTRODUCCIÓN

Este informe técnico detalla la arquitectura del sistema BP Banking System para la banca por internet de BP Bank, incluyendo una explicación pormenorizada de los componentes, contenedores, relaciones, vistas y despliegues. Además, se justifica la elección de tecnologías para el front-end, específicamente para la aplicación web SPA.

2. ANÁLISIS DE REQUERIMIENTOS

Como primer paso, es necesario identificar todos los requerimientos funcionales y no funcionales para diseñar la arquitectura para el sistema.

Requerimientos Funcionales

- Acceso al histórico de movimientos
- Transferencias entre cuentas propias e interbancarias
- Pagos diversos
- Onboarding con reconocimiento facial
- Autenticación multifactor (usuario/clave, huella, facial)
- Notificaciones de movimientos (mínimo 2 canales)

Requerimientos No Funcionales

- Alta disponibilidad (HA)
- Tolerancia a fallos
- Recuperación ante desastres (DR)
- Seguridad robusta
- Monitoreo y observabilidad
- Excelencia operativa
- Auto-healing
- Baja latencia
- Escalabilidad

3. CONSIDERACIONES NORMATIVAS

Es necesario considerar los elementos normativos más importantes antes de realizar el diseño, en este caso existen las normativas a nivel nacional, pero también existen normativas a nivel del banco, en este caso los más importantes o normativos críticos son los siguientes:

Regulaciones de Datos Personales:

- Ley Orgánica de Protección de Datos Personales (LOPDP): Publicada en el Registro Oficial el 26 de mayo de 2021, esta ley constituye el cuerpo normativo principal que regula el tratamiento y la protección de los datos personales en Ecuador, estableciendo los derechos de los titulares y las obligaciones de los responsables y encargados del tratamiento de datos.
- Reglamento General a la Ley Orgánica de Protección de Datos Personales: Emitido en noviembre de 2023, este reglamento desarrolla las disposiciones de la LOPDP, aclarando y especificando los requisitos para su correcta aplicación.
- GDPR (General Data Protection Regulation): Es una normativa europea, pero su aplicación extraterritorial afecta a empresas ecuatorianas que ofrezcan bienes o servicios a personas en la Unión Europea o que monitoreen su comportamiento.

Seguridad Financiera:

- Codificación de las Normas de la Superintendencia de Bancos: Este compendio de resoluciones establece el marco regulatorio para las instituciones financieras en Ecuador. Existen capítulos específicos que abordan la gestión de riesgos, la seguridad de la información y la continuidad del negocio.
- Norma de Control para la Gestión del Riesgo Operativo (Superintendencia de Bancos): Esta normativa detalla los lineamientos que las entidades financieras deben seguir para identificar, evaluar, mitigar y monitorear los riesgos operativos, incluyendo aquellos relacionados con la tecnología y la seguridad de la información.
- Norma de Control para la Administración del Riesgo Operativo y Riesgo Legal (Superintendencia de Economía Popular y Solidaria): Aplicable a las entidades del sector financiero popular y solidario, establece los requisitos para la gestión de estos riesgos.
- ISO 27001/27002 (Gestión de Seguridad de la Información): Son estándares internacionales de referencia, cuya adopción es una buena práctica y, en muchos casos, un requisito implícito en las normativas de la Superintendencia de Bancos para garantizar la confidencialidad, integridad y disponibilidad de la información.

- PCI DSS (Payment Card Industry Data Security Standard): Es un estándar de cumplimiento obligatorio para todas las organizaciones que almacenan, procesan o transmiten datos de tarjetas de crédito y débito.

Regulaciones Específicas del Sector:

- Ley Orgánica de Prevención, Detección y Combate del Delito de Lavado de Activos y de la Financiación de Otros Delitos: Esta ley establece el marco legal para prevenir y combatir el lavado de activos y la financiación del terrorismo en Ecuador.
- Norma de Control para la Administración del Riesgo de Lavado de Activos, Financiamiento del Terrorismo y Otros Delitos: Emitida por la Superintendencia de Bancos, esta norma establece los procedimientos y políticas que las entidades financieras deben implementar para mitigar el riesgo de ser utilizadas para actividades ilícitas.
- ISO 20022 (Estándares de mensajería financiera): Es un estándar internacional para el intercambio electrónico de datos entre instituciones financieras. Su adopción es promovida por el Banco Central del Ecuador para modernizar el sistema de pagos.
- SWIFT MT (Mensajes para transferencias internacionales): Son estándares de mensajería utilizados para la comunicación segura en la red SWIFT, esenciales para las transacciones financieras internacionales.

Auditoria y Trazabilidad:

- Normas de Auditoría de la Superintendencia de Bancos: Este conjunto de normativas regula la función de la auditoría interna y externa en las entidades financieras. Hacen referencia a marcos de trabajo internacionales como los lineamientos de la Information Systems Audit and Control Association (ISACA), lo que indirectamente posiciona a COBIT (Control Objectives for Information and Related Technology) como un marco de gobierno de TI relevante.
- Norma para la contratación y funcionamiento de las auditoras externas: Establece los requisitos y procedimientos para la contratación de firmas de auditoría externa por parte de las entidades controladas.
- SOX (Sarbanes-Oxley Act): Aplicable a las filiales ecuatorianas de empresas que cotizan en la bolsa de valores de Estados Unidos, imponiendo estrictos controles sobre la información financiera y la auditoría.

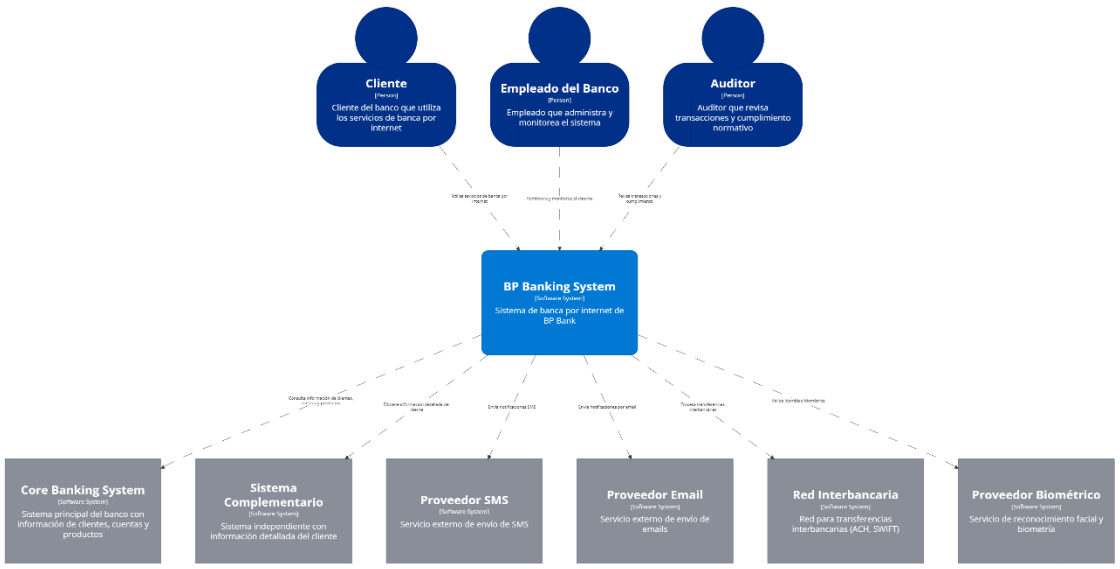
4. CONTEXTO GENERAL

El **BP Banking System** es una plataforma de banca por internet que integra canales web y móviles con el Core Banking y otros sistemas externos. La arquitectura busca cumplir con los

requisitos de **alta disponibilidad, seguridad, escalabilidad y cumplimiento normativo** (PCI DSS, SOX, Basel III).

5. ARQUITECTURA C4 MODEL

5.1 Modelo de Contexto (C1)



BP Banking System - Context Diagram (C1)
Vista de contexto del sistema de banca por internet, mostrando usuarios y sistemas externos

En el **nivel de contexto (C1)** se identifican:

- **Actores**: cliente, empleado del banco, auditor.
- **Sistemas externos**: Core Banking, sistema complementario, proveedores de SMS, email, biometría, y redes interbancarias.
- **Sistema principal**: BP Banking System.

Este modelo de alto nivel facilita la comprensión de **quién interactúa con el sistema** y **qué dependencias externas existen**. Esto es clave para que los stakeholders no técnicos entiendan la visión global.

5.1.1. Decisiones Arquitectónicas y Justificaciones:

1. **Frontend - React.js (SPA):**
   - Justificaciónes:
     - Ecosistema maduro con 200k+ librerías NPM, incluyendo react-hook-form para formularios complejos bancarios.

        - Virtual DOM optimiza re-renders en dashboards con datos en tiempo real (saldos, movimientos).
        - Justificación 3: React Query facilita manejo de cache cliente y sincronización con backend.
    - Alternativa evaluada: Angular - descartado por mayor curva de aprendizaje y menor flexibilidad.

2. **Frontend - Flutter (Mobile):**
    - Justificaciónes:
        - Rendimiento nativo 60fps crucial para biometría.
        - Single codebase reduce costos desarrollo 50% (iOS + Android con un equipo).
        - Platform channels permiten integración con FaceID/TouchID nativos.
        - Flutter Secure Storage para almacenamiento seguro de tokens biométricos
    - Alternativa evaluada: React Native - descartado por menor rendimiento en procesamiento biométrico local.

3. **API Gateway - Kong:**
    - Justificaciones:
        - Plugins nativos para OAuth2, rate limiting, circuit breaker (reduce desarrollo custom).
        - Performance: 50k requests/segundo por instancia.
        - Ecosistema: 100+ plugins de comunidad
    - Alternativa evaluada: AWS API Gateway - descartado por vendor lock-in

4. **OAuth2 con Keycloak:**
    - Justificaciones:
        - Estándar industria para banca (compliance).
        - Keycloak open-source reduce costos licenciamiento.
        - Soporte nativo para PKCE (requerido para SPA segura).
        - Federación con Azure AD para empleados banco
    - Alternativa evaluada: Auth0 - descartado por costos operacionales

5. **Spring Boot (Microservicios):**
    - Justificaciones:
        - Spring Security integración nativa con OAuth2/JWT.
        - Spring Cloud: circuit breaker, service discovery.
        - Actuator: health checks para Kubernetes.
        - Comunidad Java robusta en sector financiero
    - Alternativa evaluada: Quarkus - considerado para futuras migraciones por menor footprint (50% menos memoria)

6. **PostgreSQL (Base de Datos Principal):**
    - Justificaciones:
        - ACID completo crucial para transacciones financieras.
        - Particionamiento por fecha para histórico movimientos.
        - JSON columns para datos flexibles sin schema changes.
        - Replicación streaming para DR
    - Alternativa evaluada: MySQL - descartado por menor soporte JSON

7. **MongoDB (Auditoría):**
    - Justificaciones:
        - Schema-less ideal para eventos heterogéneos.
        - Write concern majority garantiza durabilidad.
        - Time-series collections optimizadas para logs.
        - Agregation pipeline para reportes regulatorios
    - Alternativa evaluada: Elasticsearch - descartado como primario (usado solo para búsquedas)

8. **Redis (Cache):**
    - Justificaciones:
        - In-memory: latencia < 1ms para datos frecuentes.
        - Estructuras de datos: Sets para sesiones activas.
        - Redis Cluster: 99.99% disponibilidad.
        - Pub/Sub para invalidación de cache distribuida
    - Alternativa evaluada: Memcached - descartado por falta de persistencia

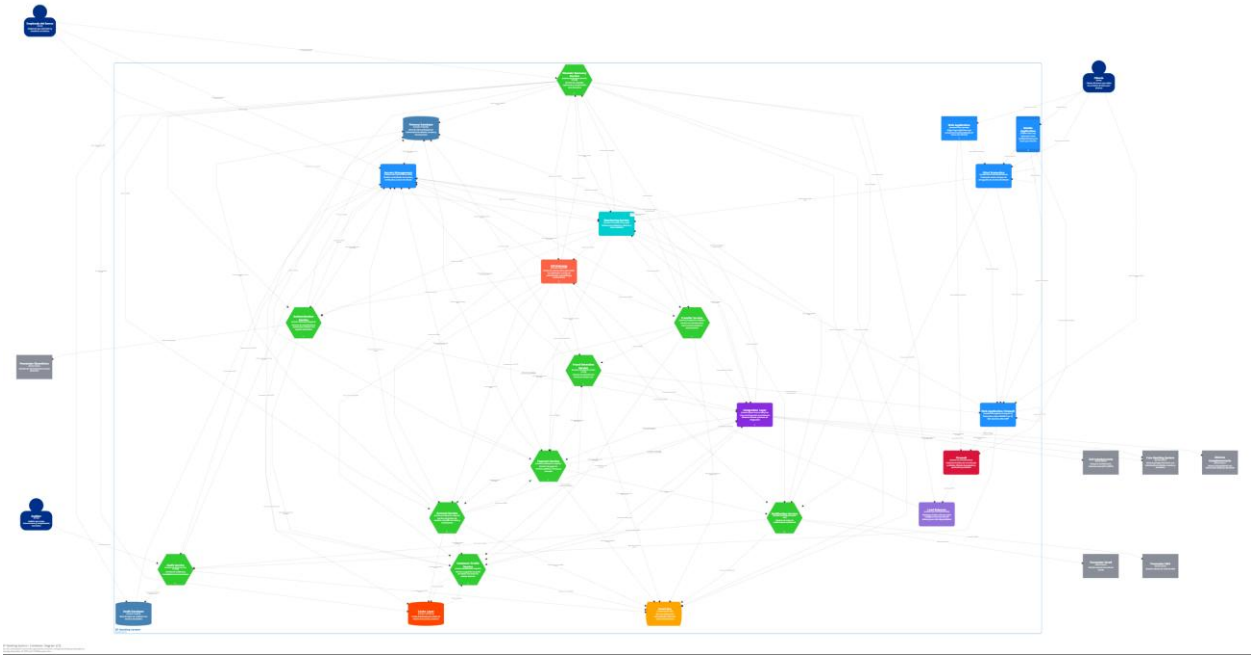
9. **Apache Kafka (Event Bus):**
    - Justificaciones:
        - Throughput: 1M mensajes/segundo por bróker.
        - Durabilidad: replicación 3x de eventos.
        - Event sourcing: retención 30 días para replay.
        - Consumer groups: escalabilidad horizontal lectores
    - Alternativa evaluada: RabbitMQ - descartado por menor throughput

10. **Apache Camel (Integration Layer):**
    - Justificaciones:
        - 300+ conectores out-of-the-box (SOAP, FIX, ISO20022).
        - EIP patterns: Splitter, Aggregator para integraciones.
        - Error handling: Dead Letter Channel para mensajes fallidos.
        - Transformaciones: XSLT, JSONPath sin código custom
    - Alternativa evaluada: MuleSoft - descartado por costos licenciamiento

11. **Azure Kubernetes Service (AKS):**
    - Justificaciones:
        - Auto-scaling: HPA + VPA para optimizar recursos.
        - Rolling updates: zero-downtime deployments.
        - Managed service: SLA 99.95% uptime.
        - Azure integration: Key Vault, Monitor nativo
    - Alternativa evaluada: VMs tradicionales - descartadas por menor agilidad

12. **Prometheus + Grafana (Monitoreo):**
    - Justificaciones:
        - Time-series DB optimizada para métricas.
        - Prometheus Alertmanager: alertas configurables.
        - Grafana: dashboards predefinidos para Spring Boot.
        - Open-source: sin vendor lock-in
    - Alternativa evaluada: Datadog - descartado por costos (10x más caro)

5.2 Modelo de Contenedores (C2)



El **sistema** se divide en contenedores especializados siguiendo principios **microservicio con separación de responsabilidades**:

Frontend Applications:

- **SPA (Single Page Application)**: React.js con TypeScript
- **Mobile App**: Flutter, pues es multiplataforma, tiene rendimiento nativo, un solo código base, ecosistema maduro, soporte para biometría.

Firewall (Azure Firewall/FortiGate): Seguridad perimetral.

Load Balancer (Azure LB/NGINX): Distribución de tráfico.

API Gateway (Kong, NGINX, Azure API Management**)**: Punto único de entrada, autenticación, rate limiting.

Backend Services:

- **Authentication Service**: (OAuth2 + biometría).
- **Business Services**: Spring Boot (o incluso Quarkus) microservices
    - **Account Service** (consultas y saldos).
    - **Transfer Service** (transferencias con patrón Saga).
    - **Notification Service** (envío multicanal con Kafka).

    - **Fraud Detection Service** (detección de fraude con ML).
    - **Audit Service** (event sourcing y cumplimiento).
    - **Integration Layer** (Apache Camel para conectividad externa, también se puede usar MuleSoft).
- **Event Bus:** Apache Kafka para comunicación asíncrona.
- **Monitoring Service:** Prometheus, Grafana, Jaeger para observabilidad.

Data Layer:

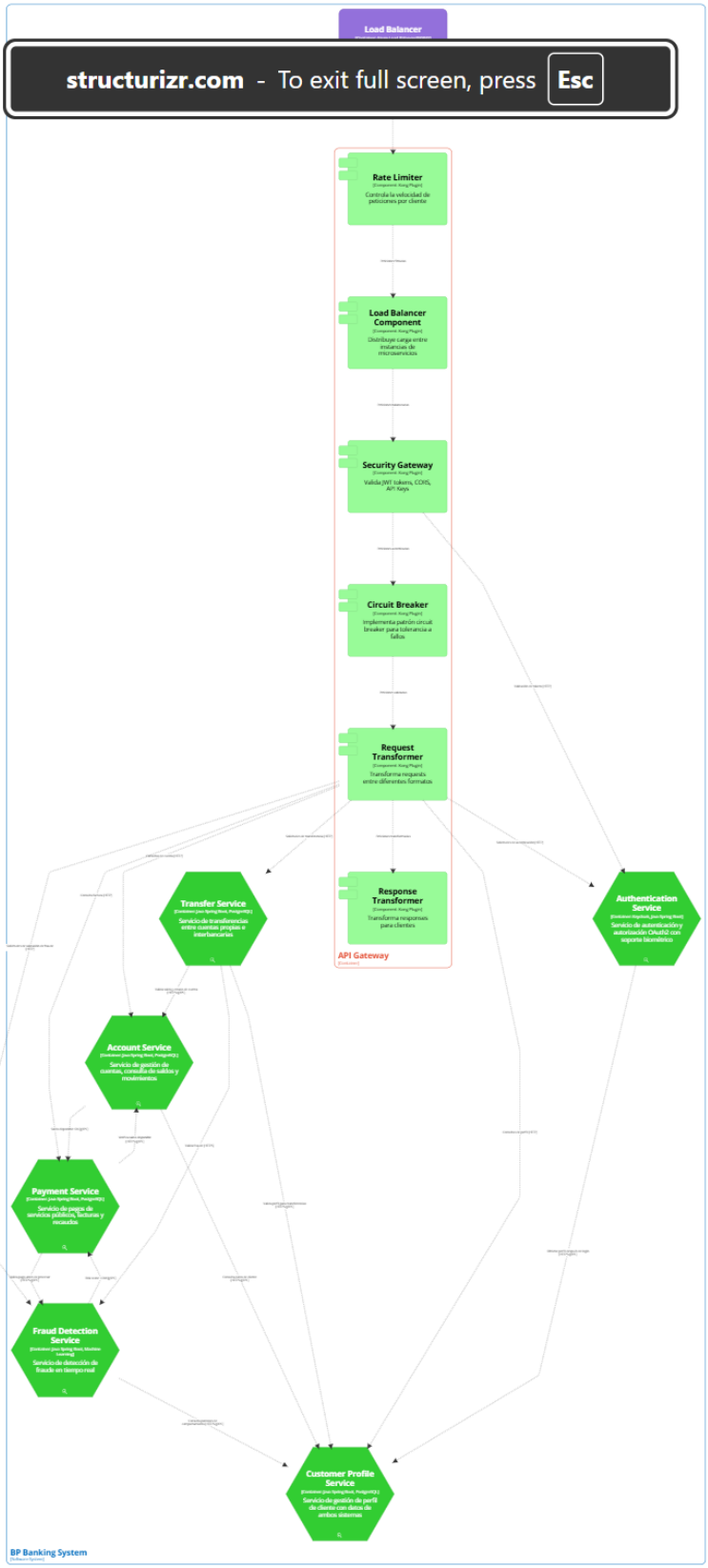
- **Core Database**: PostgreSQL (Principal)
- **Audit Database**: MongoDB (Auditoría)
- **Cache Layer**: Redis (Persistencia clientes frecuentes)
- **Event Store**: Apache Kafka

Esta distribución permite **escalar independientemente** cada servicio, aislar fallos, y optimizar rendimiento. El API Gateway agrega seguridad y control de tráfico, y el uso de un bus de eventos desacopla la comunicación interna.

5.3 Modelo de Componentes (C3)

Cada microservicio se descompone en componentes internos que implementan patrones específicos:

API Gateway Components:



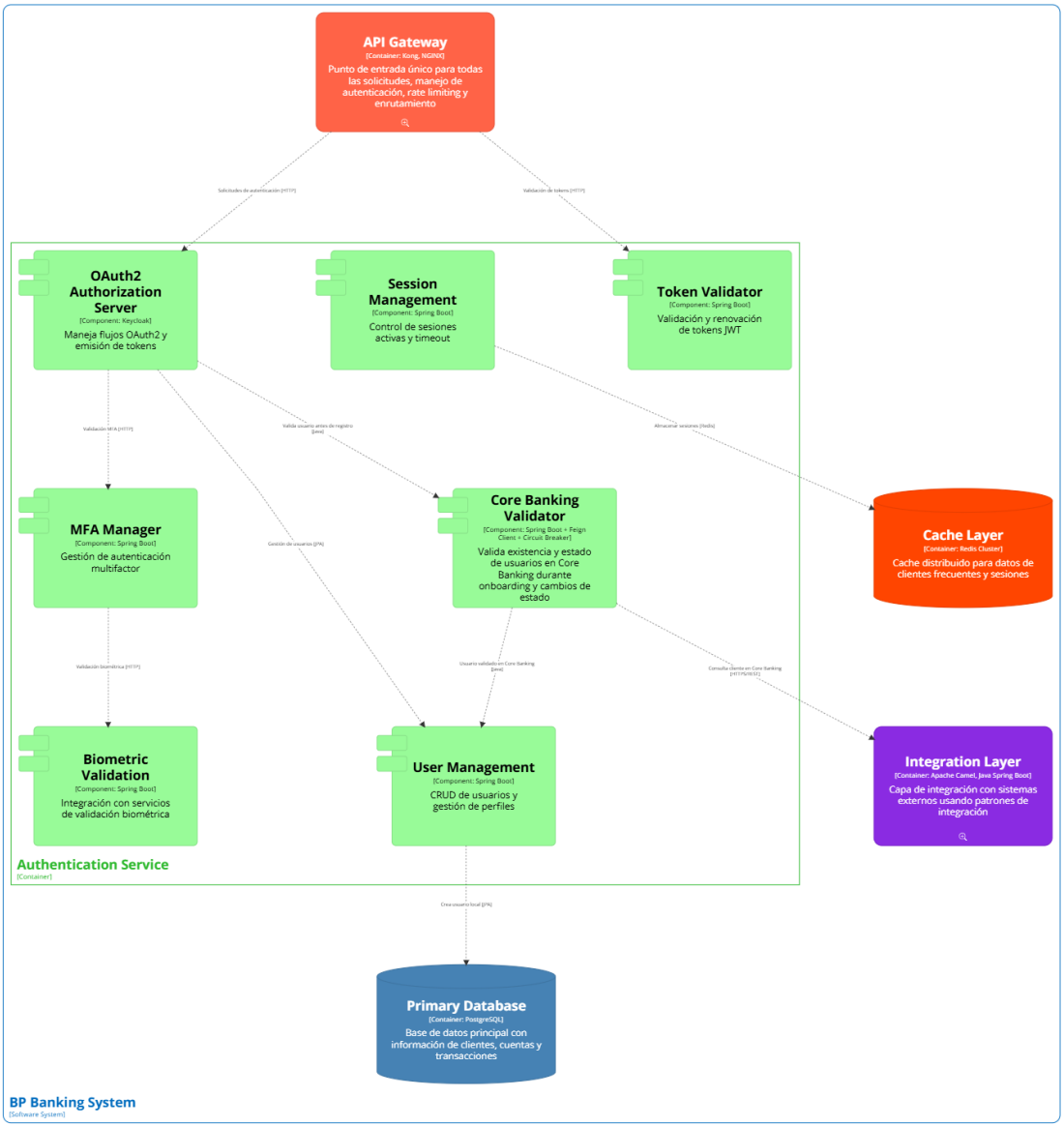
API Gateway - Component Diagram (C3)

Componentes internos del API Gateway mostrando el flujo de procesamiento de requests

Saturday, November 29, 2025 at 8:55 PM Ecuador, Time

- **Rate Limiting**: Control de velocidad de peticiones
- **Load Balancer**: Distribución de carga
- **Security Gateway**: Validación JWT, CORS, API Keys
- **Circuit Breaker**: Patrón de tolerancia a fallos
- **Request/Response Transformation**: Adaptación de formatos

Authentication Service Components:

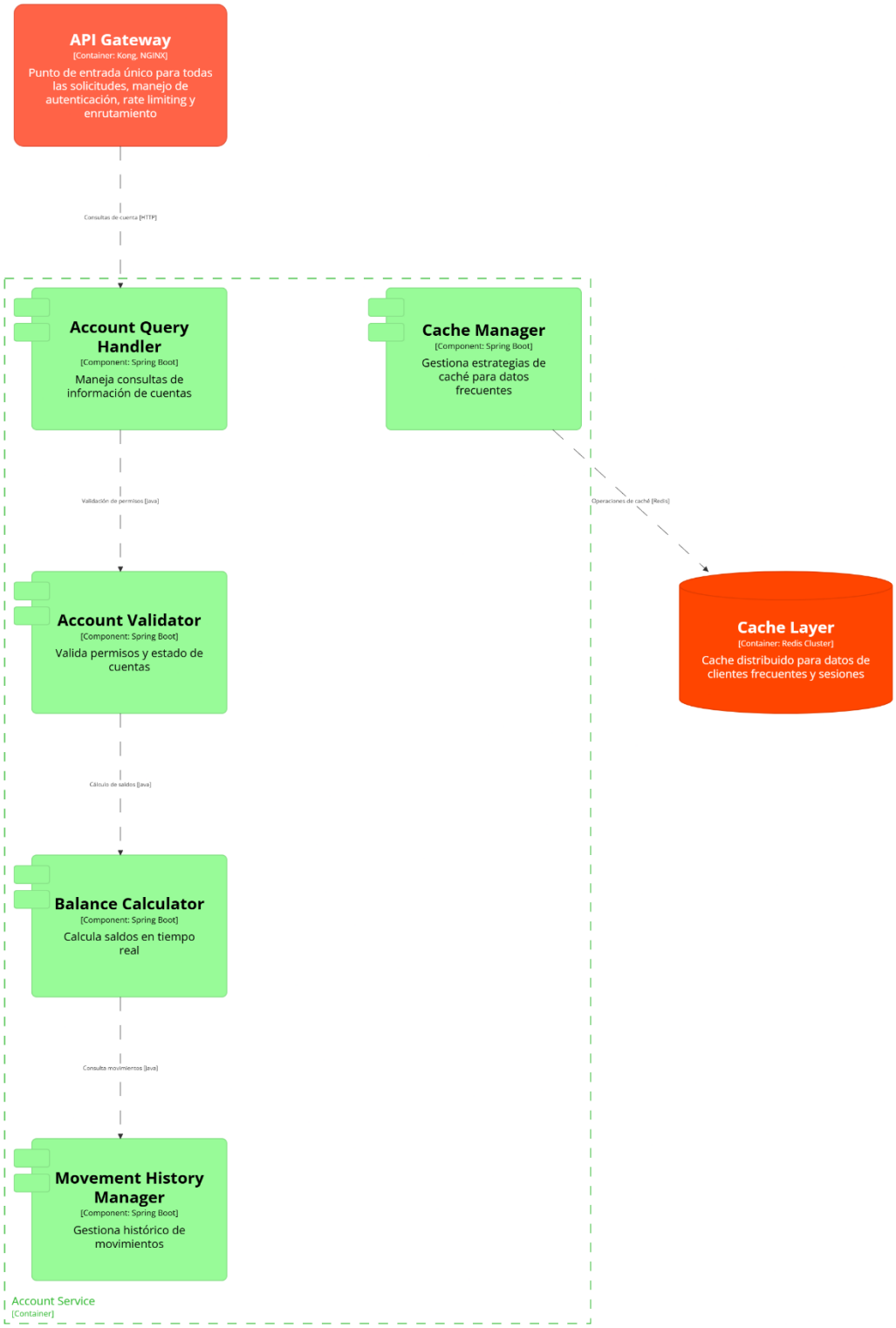


Authentication Service - Component Diagram (C3)
Componentes del servicio de autenticación y autorización OAuth2
Saturday, November 29, 2025 at 8:16 PM Ecuador Time

- **OAuth2 Authorization Server**: Gestión de tokens
- **User Management**: CRUD de usuarios
- **Biometric Validation**: Integración con servicios de reconocimiento
- **MFA Manager**: Autenticación multifactor
- **Session Management**: Control de sesiones activas

Business Services Components:

Account Service:



Account Service - Component Diagram (C3)

Componentes del servicio de gestión de cuentas y consulta de movimientos

- Account Query Handler
- Balance Calculator
- Account Validator
- Movement History Manager

Customer Profile Service

Este servicio implementa el patrón CQRS (Command Query Responsibility Segregation) para gestionar eficientemente la información de clientes que proviene de DOS fuentes:

1. Core Banking System (información básica)
2. Sistema Complementario (información detallada)

Componentes:

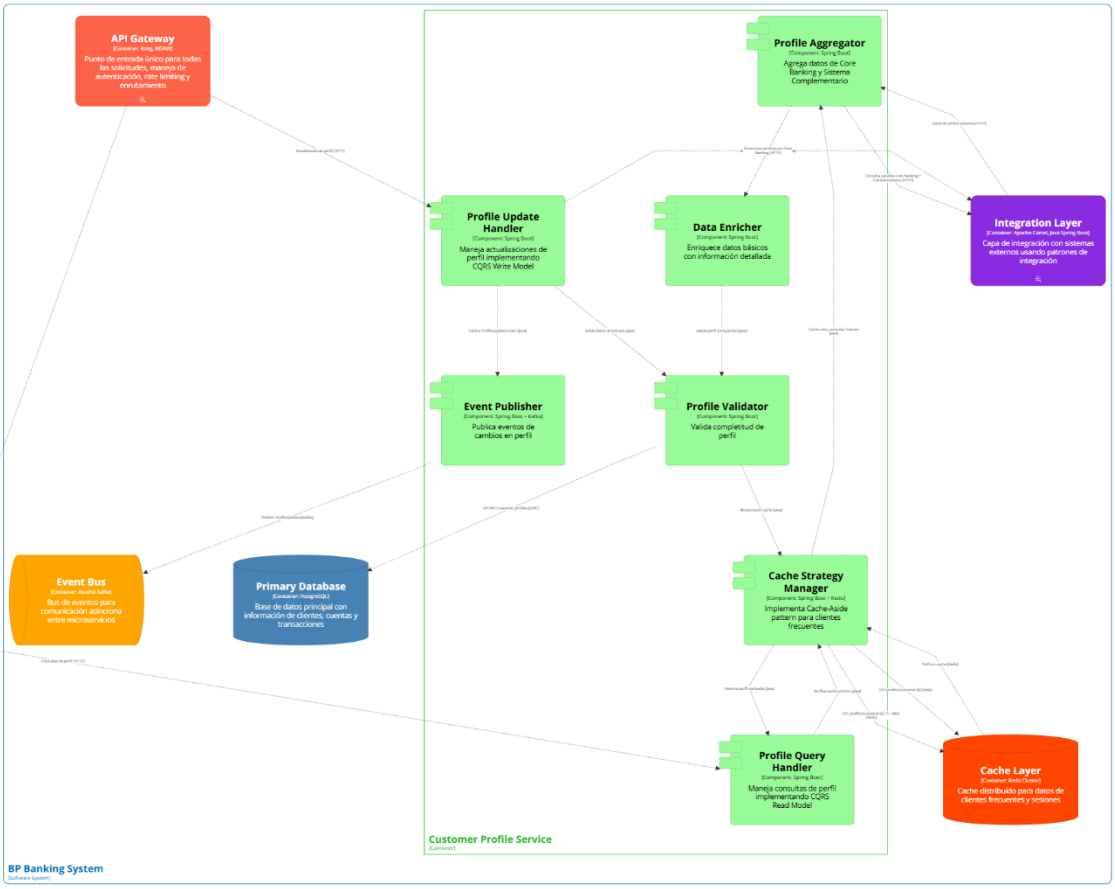
- **Profile Aggregator**: Consulta paralela a ambos sistemas externos, con lo que reduce latencia al obtener datos de múltiples fuentes simultáneamente. Además, se implementa patrón Scatter-Gather para optimizar tiempo de respuesta.
- **Cache Strategy Manager:** Se implementa Cache-Aside Pattern, con ello, los clientes frecuentes generan 80% de las consultas (Principio Pareto), y reduce carga en Core Banking (sistema legacy costoso); se considera como TTL 3600 segundos (1 hora) para datos de perfil, y, cuando hay cambios se usa Event-driven para invalidación.
- **Data Enricher:** Se enriquece datos básicos con información detallada, de esta manera se unifica la vista del cliente sin duplicar datos en BD propia.
- **Profile Query Handler (CQRS Read Model):** Optimizado para consultas de solo lectura (Cache-first approach).
- **Profile Update Handler (CQRS Write Model):**
    - Maneja actualizaciones y sincronización con sistemas externos.
    - Publica eventos ProfileUpdated al Event Bus

Flujo de Consulta (Cache-Aside Pattern):

1. Request: Profile Query Handler.
2. **Verifica Cache (Redis):** GET profile:{customerId}.
3. Cache HIT: Retorna datos.
4. Cache MISS: Profile Aggregator consulta ambos sistemas.
5. Data Enricher unifica datos.
6. Almacena en Cache con TTL.
7. Retorna al cliente.

Justificación de arquitectura:

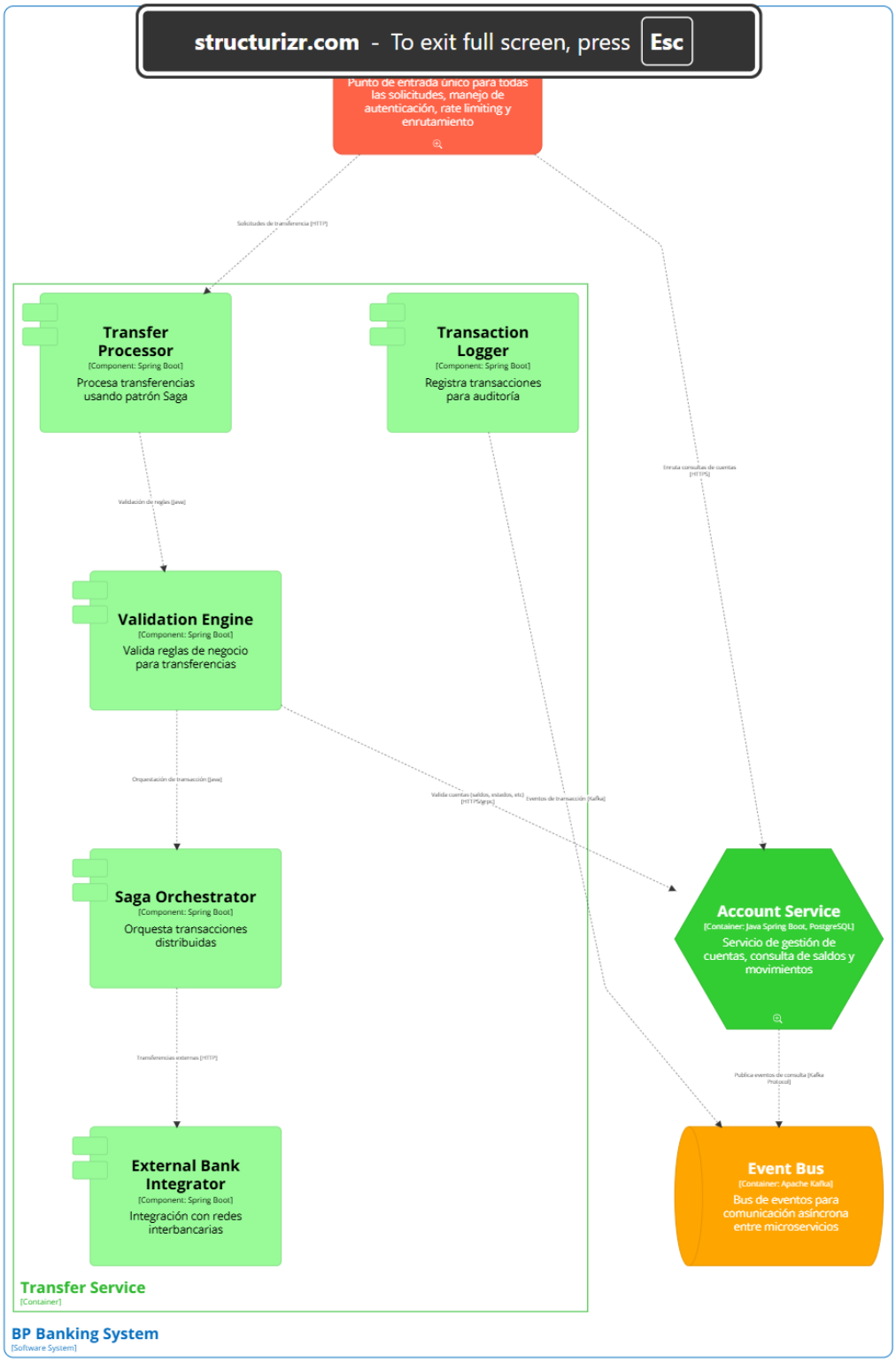
- Separación CQRS mejora rendimiento en 60% para consultas.
- Cache-Aside reduce latencia de 800ms a 50ms para clientes frecuentes(Los cálculos son solo estimaciones y revisados en documentación del patrón).
- Agregación de fuentes permite evolución independiente de sistemas legacy



Customer Profile Service - Component Diagram (C3)
Componentes del servicio de gestión de perfiles de los clientes
Saturday, November 29, 2025 at 7:13 PM Ecuador Time

Transfer Service:

Es un procesador con orquestación Saga.



Transfer Service - Component Diagram (C3)

Componentes del servicio de transferencias con patrón Saga

Saturday, November 29, 2025 at 8:18 PM Ecuador Time

- Transfer Processor
- Validation Engine
- External Bank Integrator
- Transaction Logger

Payment Service

Servicio especializado en pagos de servicios públicos, facturas y recaudos, con capacidad de pagos programados y reconciliación automática.

Integraciones críticas:

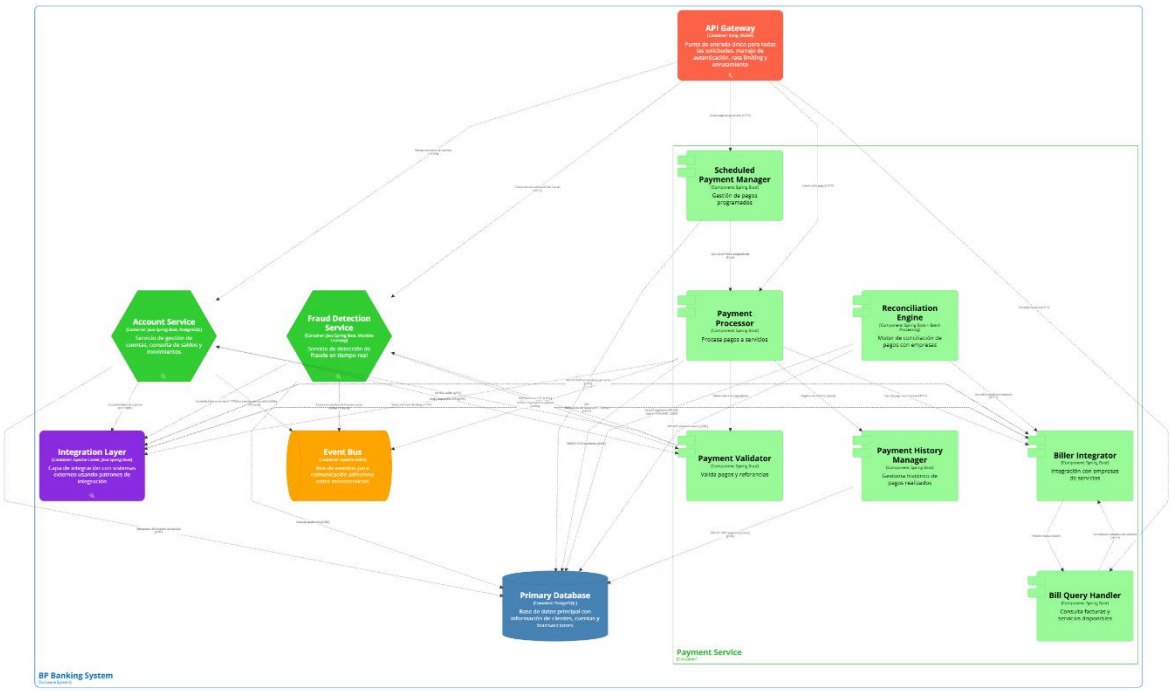
1. Account Service: Verificación de saldo (gRPC síncrono)
2. Fraud Service: Validación de riesgo antes de ejecutar
3. Integration Layer: Conectividad con empresas de servicios
4. Core Banking: Débito de cuenta vía Integration Layer
5. Event Bus: Publica PaymentCompleted para notificaciones

Flujo completo de pago:

1. Cliente solicita pago → Payment Processor
2. Payment Validator verifica datos
3. Valida saldo en Account Service (gRPC)
4. Valida riesgo en Fraud Service (gRPC)
5. Biller Integrator ejecuta pago en empresa
6. Integration Layer debita cuenta en Core Banking
7. Payment Processor registra en BD
8. Publica evento PaymentCompleted
9. Notification Service envía confirmación

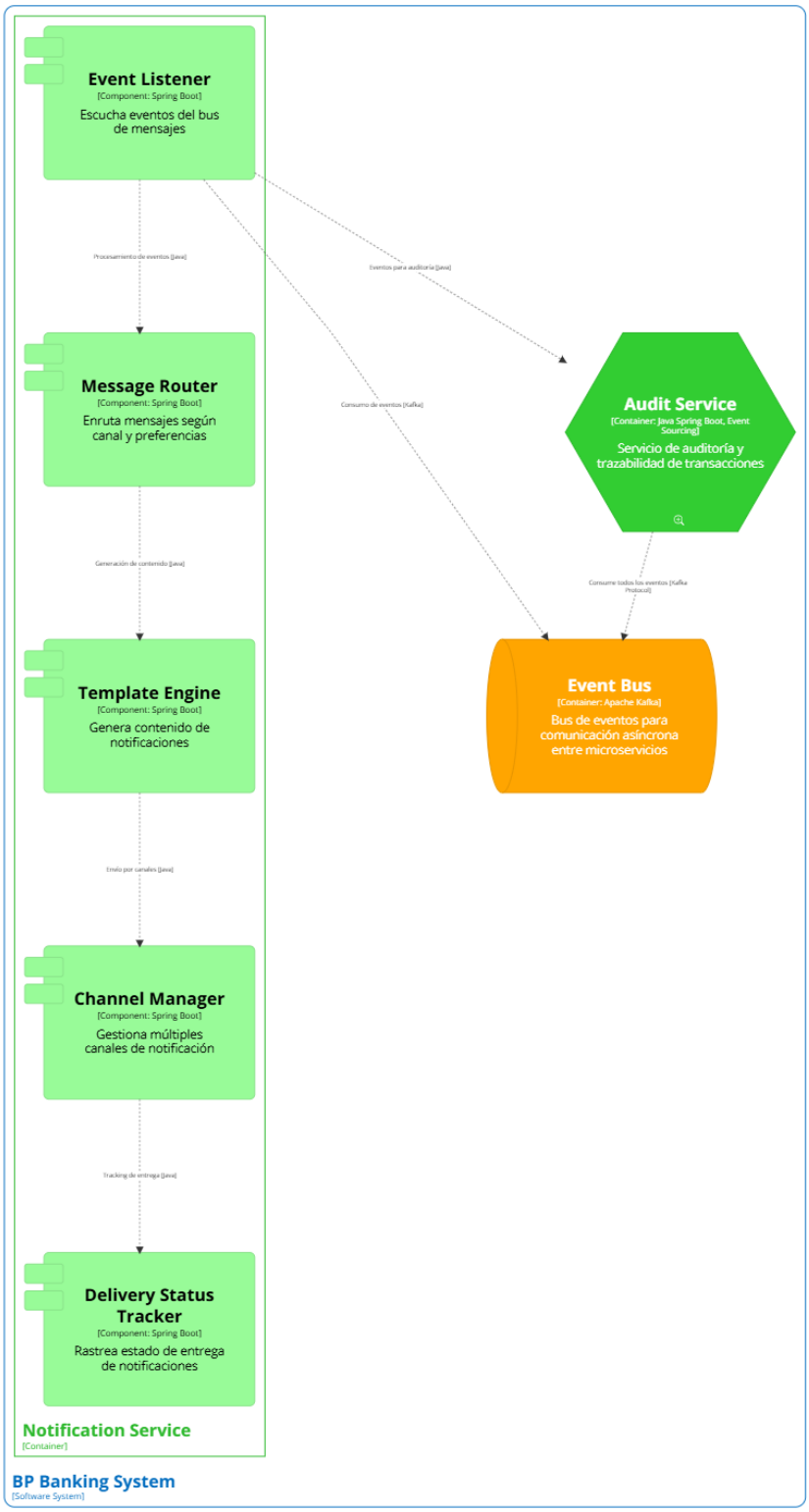
Justificaciones técnicas:

- Saga Pattern garantiza consistencia eventual en fallos
- gRPC reduce latencia 40% vs REST en validaciones críticas
- Reconciliation Engine cumple normativa de Superintendencia de Bancos
- Scheduled Payments mejora experiencia usuario (pagos automáticos)



Payment Service - Component Diagram (C3)

Notification Service:



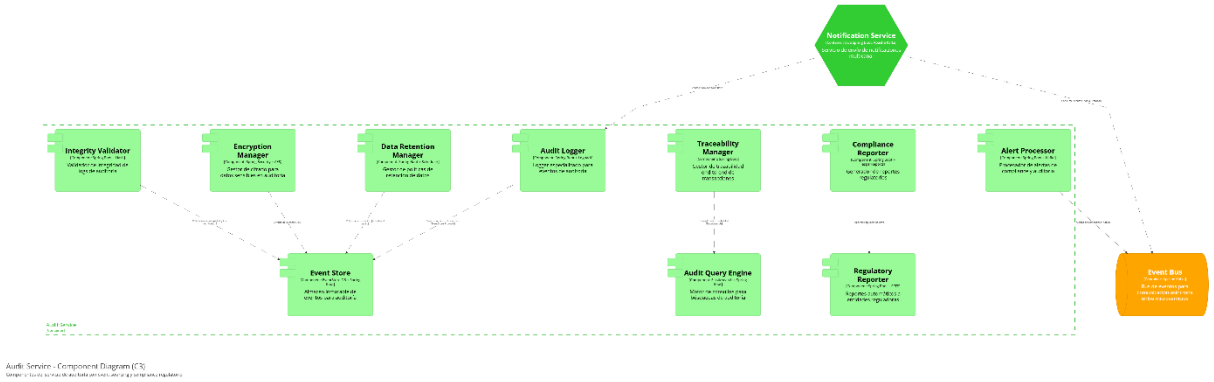
Notification Service - Component Diagram (C3)

Componentes del servicio de notificaciones multicanal

Saturday, November 29, 2025 at 8:19 PM Ecuador Time

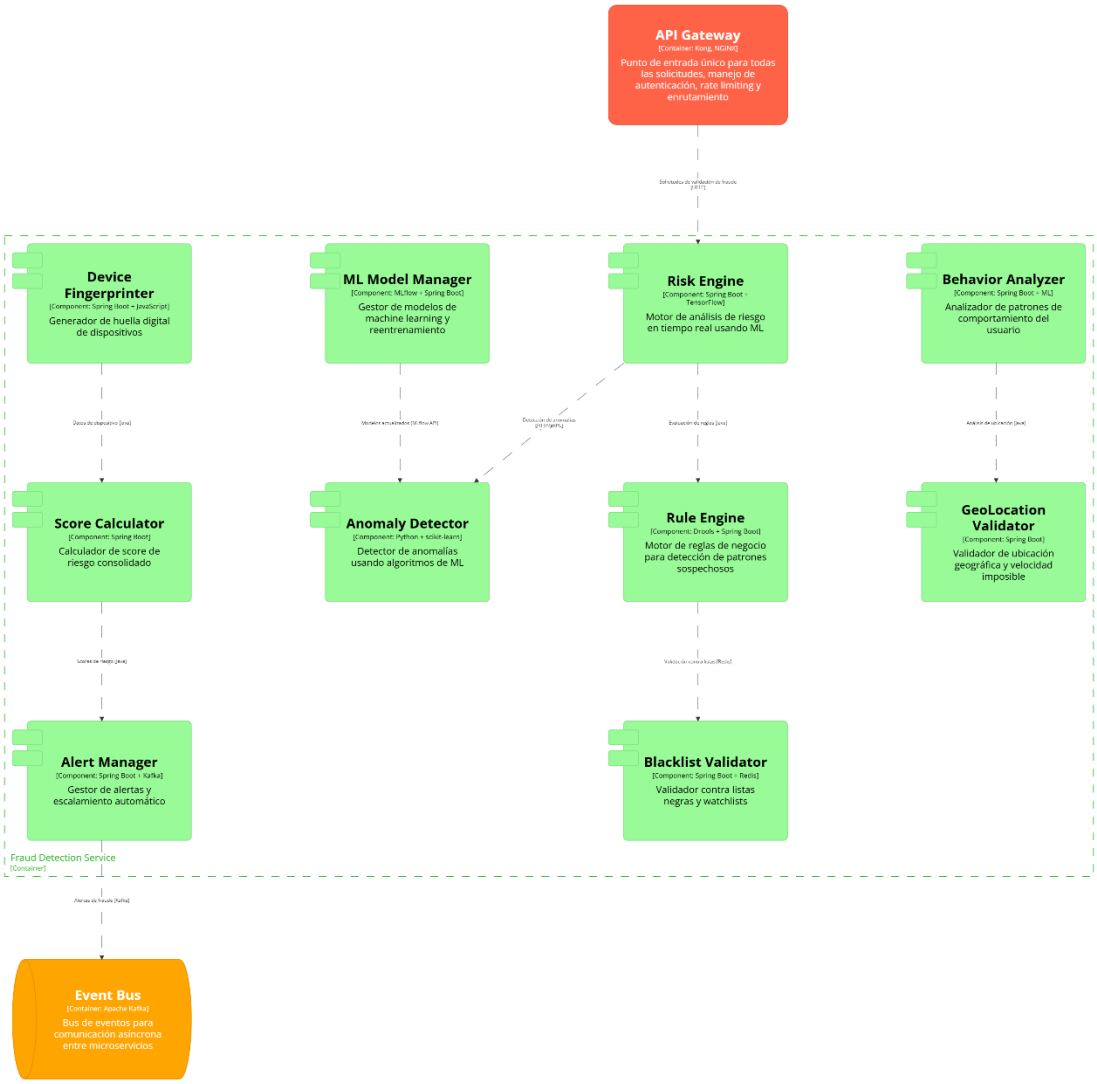
- Event Listener
- Message Router
- Template Engine
- Delivery Status Tracker

Audit Service



- Almacenamiento inmutable
- Reportes regulatorios

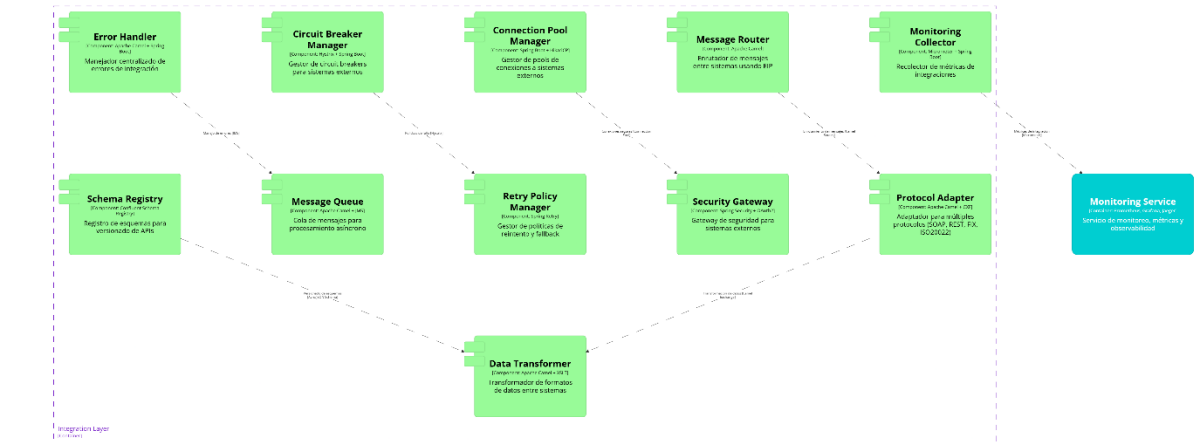
Fraud Service



Fraud Detection Service - Component Diagram (C3)
Componentes del servicio de detección de fraude con ML y análisis de riesgo en tiempo real

- Motores ML.
- Reglas de negocio.

Integration Layer



Integration Layer - Component Diagram (C3)

- Adaptadores de protocolos.
- Transformadores de datos.

Data Access Components:

- **Repository Pattern**: Abstracción de acceso a datos
- **Connection Pool Manager**: Gestión eficiente de conexiones
- **Cache Manager**: Estrategias de caché
- **Audit Trail Handler**: Registro de auditoría

El nivel de componentes muestra el diseño interno y cómo cada módulo aplica patrones de arquitectura y buenas prácticas como resiliencia, seguridad y escalabilidad.

Disaster Recovery Service

Servicio dedicado a garantizar RPO ≤ 15 minutos y RTO ≤ 2 horas, cumpliendo la normativa de Superintendencia de Bancos sobre continuidad.

Flujos operacionales:

- **Backup Automático (diario):**

1. Backup Orchestrator inicia proceso programado
2. Snapshot Manager crea snapshots de PostgreSQL, MongoDB, Redis
3. Replication Manager sincroniza con región secundaria
4. Data Consistency Validator verifica checksums
5. Notifica a Monitoring Service resultado

- **Detección y Failover Automático:**

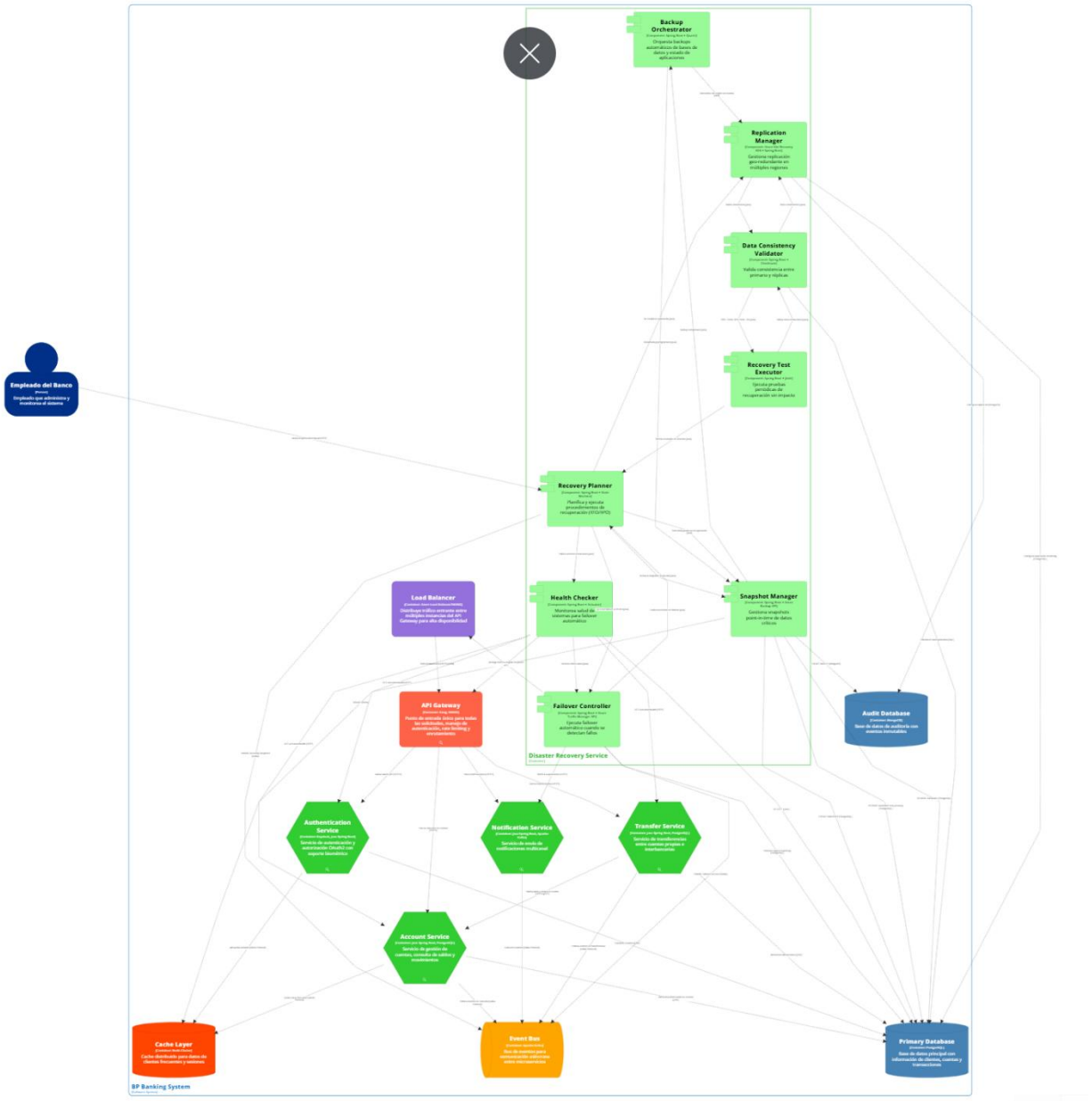
1. Health Checker detecta fallo en servicio crítico (ej: Primary DB caída)
2. Alerta a Failover Controller
3. Recovery Planner evalúa necesidad de failover
4. Failover Controller ejecuta:
    1. Redirige tráfico a región DR (Azure Traffic Manager)
    2. Promueve réplica secundaria a primaria
    3. Publica evento FailoverExecuted al Event Bus
    4. Notifica a equipo de operaciones
5. Health Checker valida servicios restaurados
6. Tiempo total: < 5 minutos (cumple RTO)

- **Prueba de DR (trimestral):**

1. Recovery Test Executor simula fallo
2. Restaura snapshot en ambiente aislado
3. Data Consistency Validator verifica integridad
4. Mide RTO/RPO reales
5. Genera reporte de cumplimiento regulatorio

Justificaciones de arquitectura:

- Replicación asíncrona: Balance entre RPO y performance (15min vs 0 con síncrona)
- Multi-región: Protege contra fallos de datacenter completo
- Failover automático: Reduce RTO de 2 horas a 5 minutos
- Pruebas periódicas: Cumple Norma de Continuidad de Negocio (SB)



Isaster Recovery Service - Component Diagram (C3)

FLUJOS

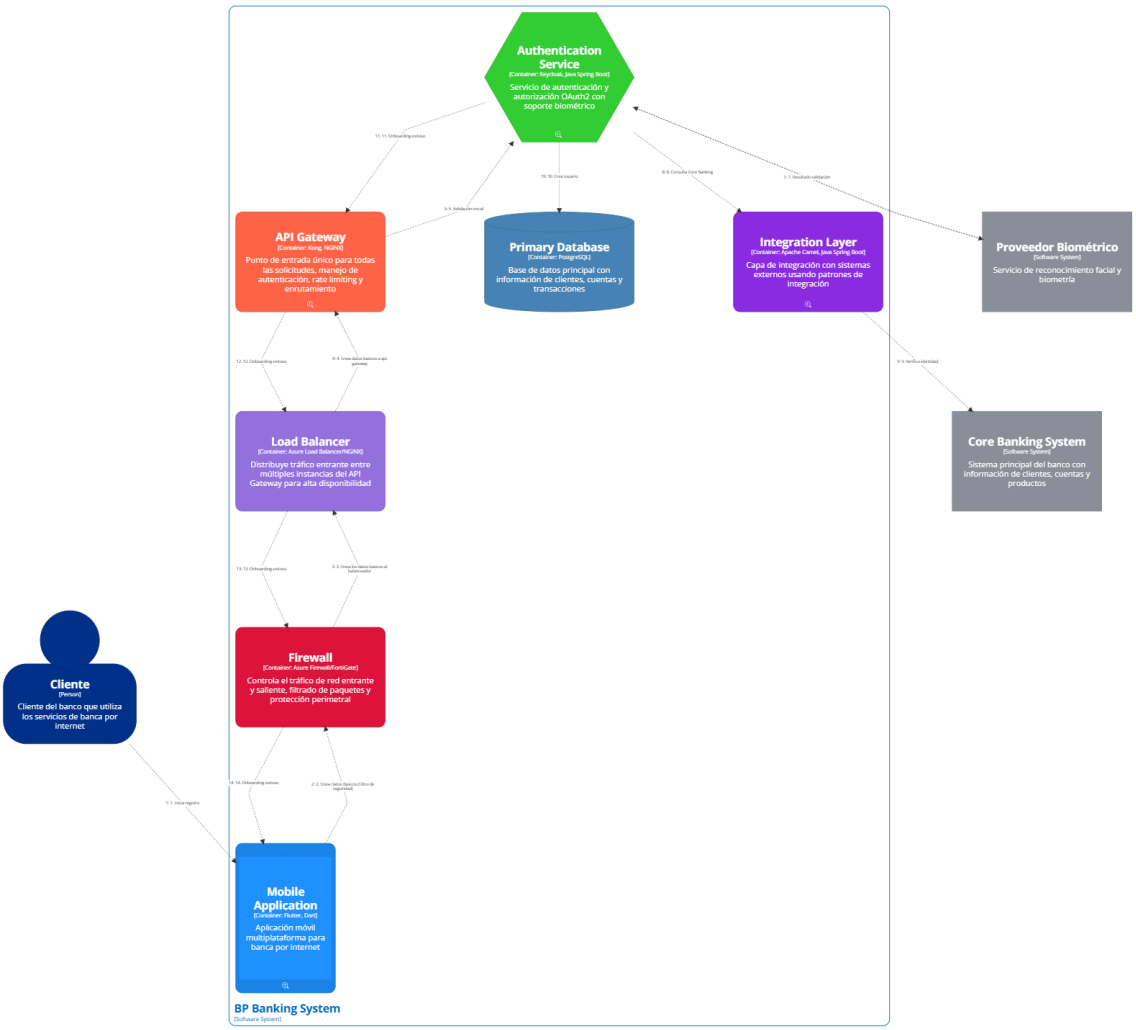
Flujo de Autenticación

Flujo de Autenticación OAuth2 con Biometría

Flujo de autenticación OAuth2 con biometría

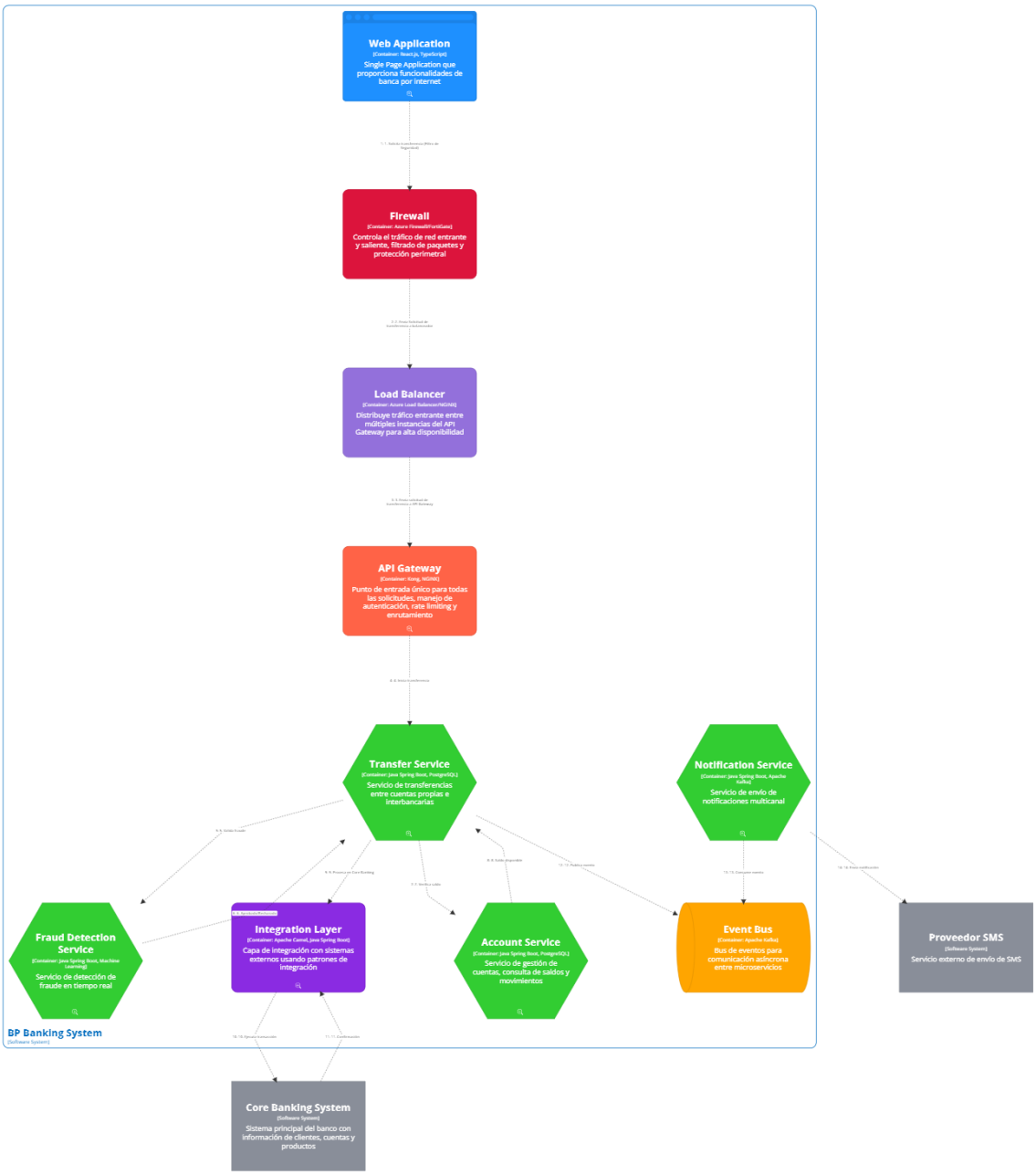
Saturday, November 29, 2025 at 8:31 PM Ecuador Time

Flujo de Onboarding



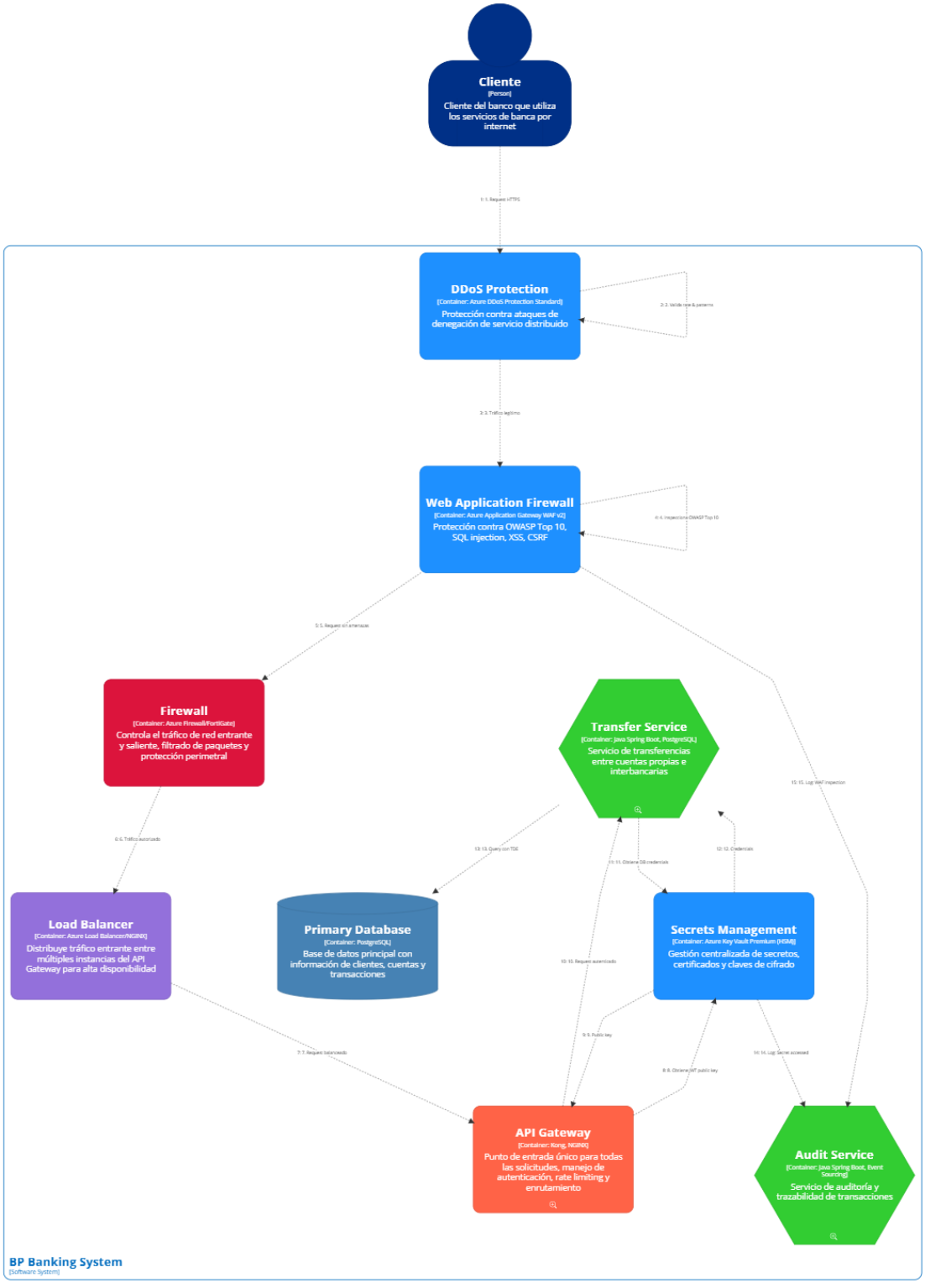
Proceso de Onboarding con Reconocimiento Facial
Flujo de registro de nuevo cliente con biometría
Saturday, November 29, 2025 at 8:34 PM Ecuador Time

Flujo de Transferencias



Proceso de Transferencia con Patrón Saga
Patrón Saga para transferencias distribuidas
Saturday, November 29, 2025 at 8:35 PM Ecuador Time

Flujo de Seguridad en capas

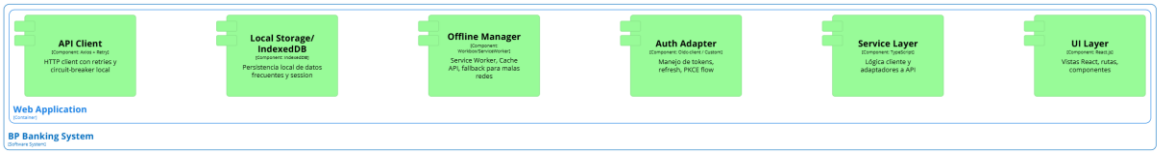


Flujo de Seguridad en Capas - Defense in Depth

Flujo completo de seguridad en capas

Saturday, November 29, 2025 at 8:35 PM Ecuador Time

5.3.4 Web Application Components (React SPA):



Web Application - Component Diagram (C3)
Componentes internos de la SPA y cómo consumen el API Gateway
Saturday, November 29, 2025 at 8:28 PM Ecuador Time

Componentes de Presentación:

- **UI Layer:** Componentes React para vistas
    1. Reutilización con patrón Atomic Design.
    2. Lazy loading de rutas reduce bundle inicial 60%.
    3. Librerías: Material-UI para componentes bancarios estándar

- **Service Layer: Lógica de cliente y adaptadores**
    1. Separación de UI y lógica de negocio
    2. Implementa Repository Pattern para acceso a APIs

Componentes de Seguridad:

- **Auth Adapter: Manejo de tokens OAuth2**
    1. Implementa PKCE flow (requerido para SPAs)
    2. Refresh token automático con interceptors
    3. Librería: oidc-client-ts para gestión OAuth2
    4. Storage: SessionStorage (más seguro que LocalStorage)

Componentes de Performance:

- **Offline Manager: Service Worker para offline-first**
    1. Continúa funcionando sin internet (consultas cacheadas)
    2. Background sync para transacciones pendientes
    3. Tecnología: Workbox para gestión de cache
- **Local Storage/IndexedDB:** Persistencia local
    1. IndexedDB para grandes volúmenes (>5MB)
    2. Uso: Cache de movimientos recientes (últimos 30 días)

- **API Client: HTTP client con resiliencia**
    1. Retry automático con exponential backoff
    2. Circuit breaker client-side
    3. Librería: Axios con interceptores custom

Flujo de autenticación (PKCE):

1. UI Layer inicia login
2. Auth Adapter genera code_verifier + code_challenge
3. Redirecciona a Authorization Server con challenge
4. Usuario autentica (username/password + MFA)
5. Auth Server retorna authorization_code
6. Auth Adapter intercambia code + verifier por tokens
7. Almacena access_token en SessionStorage
8. Configura interceptors en API Client para adjuntar token

Mobile Application Components (Flutter):

Mobile Application - Component Diagram (C3)
Componentes de la app móvil (Onboarding con biometría, auth y sincronización offline)
Saturday, November 29, 2025 at 8:30 PM Ecuador Time

Componentes de Presentación:

- **UI Widgets:** Pantallas y navegación
    - Flutter widgets nativos (Material + Cupertino)
    - **Arquitectura:** BLoC (Business Logic Component) para state management

Componentes de Red:

- **Network Layer:** HTTP client para APIs
    - Dio C**lient con interceptores**
    - Retry policy con exponential backoff
    - **Certificate pinning:** previene MITM attacks

Componentes de Seguridad:

- **Auth Adapter Mobile:** OAuth2 + PKCE + Biometría
    - AppAuth SDK para OAuth2 nativo
    - LocalAuthentication plugin para biometría
    - **Flujo: Biometría local → valida token → allow access**

- **Biometric Integration:** Onboarding con reconocimiento facial
    - Platform channels para FaceID/Android Biometric
    - Liveness detection previene spoofing con fotos
    - Proveedor: Integración con servicio externo de biometría
    - Flujo onboarding:
        1. Captura facial con cámara
        2. Liveness detection (parpadeo, movimiento)
        3. Envía a proveedor biométrico
        4. Valida contra Core Banking
        5. Crea usuario en Auth Service

Componentes de Persistencia:

- **Offline Sync:** SQLite local + sincronización
    - Permite consultas offline
    - Sincronización automática al reconectar
    - **Estrategia:** Conflict resolution con last-write-wins

Justificación de arquitecturas frontend:

- **SPA:** Experiencia fluida sin recargas (UX superior)
- **Flutter:** Performance 60fps crucial para animaciones biometría
- **Offline-first:** Funcionalidad en zonas con baja conectividad
- **Security:** PKCE + Certificate Pinning cumple OWASP Mobile Top 10

6. FLUJO DE AUTENTICACIÓN OAUTH2

Flujo Recomendado: Authorization Code + PKCE

Para SPA:

1. Client → Authorization Server: Authorization Request + code_challenge

2. User Authentication (username/password + MFA)

3. Authorization Server → Client: Authorization Code

4. Client → Authorization Server: Token Request + code_verifier

5. Authorization Server → Client: Access Token + Refresh Token

Para Mobile App:

1. App → Authorization Server: Authorization Request + code_challenge

2. Biometric/PIN Authentication

3. Authorization Server → App: Authorization Code

4. App → Authorization Server: Token Request + code_verifier

5. Authorization Server → App: Access Token + Refresh Token

Herramientas de Industria Recomendadas:

Reconocimiento Facial:

- **Microsoft Face API**: Integración cloud, alta precisión
- **Amazon Rekognition**: Escalable, análisis en tiempo real
- **FaceID/TouchID**: Para dispositivos iOS nativos

Autenticación Biométrica:

- **FIDO2/WebAuthn**: Estándar web para biometría
- **Jumio**: Verificación de identidad y biometría
- **Onfido**: KYC y verificación biométrica

7. **RELACIONES**
   Se definen flujos desde los canales de cliente (web/móvil) → firewall → balanceador → API Gateway → microservicios.
   - Comunicación con sistemas externos mediante la Integration Layer.
   - Uso de Kafka para integración asíncrona y reducción de acoplamiento.
   - Bases de datos segregadas por propósito (OLTP vs auditoría).

Las relaciones siguen un flujo controlado y seguro, garantizando que todo tráfico pase por capas de seguridad y control antes de llegar a la lógica de negocio.

8. PATRÓN DE DISEÑO PARA PERSISTENCIA DE CLIENTES FRECUENTES

Patrón: Cache-Aside + Observer Pattern

Componentes Interactuantes:

```
@Component
public class FrequentClientCacheManager {

    @Autowired
    private RedisTemplate<String, Object> redisTemplate;
    @Autowired
    private ClientBehaviorAnalyzer analyzer;
    @EventListener
    public void handleClientActivity(ClientActivityEvent event) {
        // Observer pattern - escucha actividades del cliente
        ClientProfile profile = analyzer.analyzeFrequency(event.getClientId());
        if (profile.isFrequentClient()) {
            // Cache-Aside pattern - actualiza caché
            cacheClientData(event.getClientId(), profile);
        }
    }
    public ClientData getClientData(String clientId) {
        // Cache-Aside pattern - lee del caché primero
        ClientData cached = (ClientData) redisTemplate.opsForValue()
            .get("frequent_client:" + clientId);
        if (cached == null) {
            cached = loadFromDatabase(clientId);
            if (isFrequentClient(clientId)) {
                redisTemplate.opsForValue()
                    .set("frequent_client:" + clientId, cached, Duration.ofHours(24));
```

```
        }
      }
      return cached;
    }
}
```

Estrategia de Invalidación:

- **TTL**: 24 horas para datos de clientes frecuentes
- **Event-Driven**: Invalidación por eventos de cambio de datos
- **LRU**: Least Recently Used para gestión de memoria

9. SERVICIOS ADICIONALES RECOMENDADOS

Servicios Core Existentes:

1. **Consulta de datos básicos**
2. **Consulta de movimientos**
3. **Transferencias**

Servicios Adicionales Propuestos:

4. **Fraud Detection Service**: Análisis de patrones sospechosos
5. **Notification Service**: Gestión unificada de notificaciones
6. **KYC/AML Service**: Cumplimiento normative
7. **Analytics Service**: Métricas y comportamiento del usuario
8. **Configuration Service**: Gestión centralizada de configuraciones
9. **Audit Service**: Trazabilidad completa de operaciones
10. **File Management Service**: Gestión de documentos y archivos

10. ARQUITECTURA DE INFRAESTRUCTURA CLOUD

Azure Architecture Pattern

Compute Layer:

- **Azure Kubernetes Service (AKS)**: Orquestación de contenedores
- **Azure Container Instances**: Servicios de procesamiento ligero
- **Azure Functions**: Procesamiento serverless para eventos

Data Layer:

- **Azure Database for PostgreSQL**: Base de datos principal
- **Azure Cosmos DB**: Base de datos de auditoría (NoSQL)
- **Azure Cache for Redis**: Caché distribuido
- **Azure Event Hubs**: Streaming de eventos

Security Layer:

- **Azure Active Directory B2C**: Gestión de identidades
- **Azure Key Vault**: Gestión de secretos y certificados
- **Azure Application Gateway**: WAF y balanceador L7
- **Azure DDoS Protection**: Protección contra ataques

Monitoring & Observability:

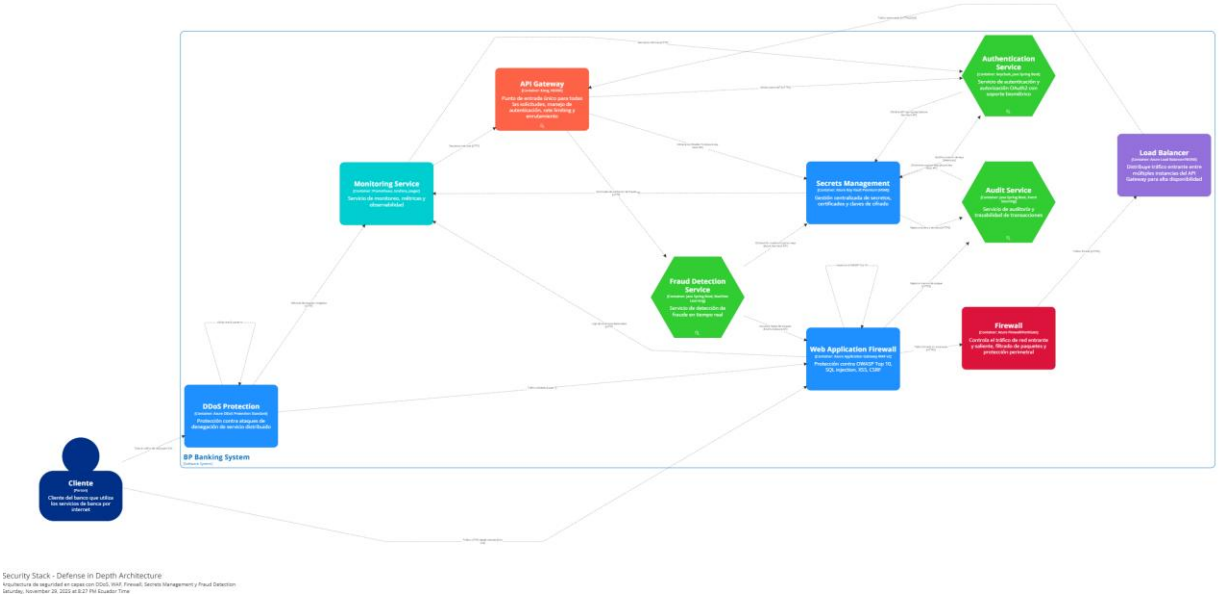
- **Azure Monitor**: Métricas y alertas
- **Application Insights**: APM y telemetría
- **Azure Log Analytics**: Centralización de logs
- **Azure Sentinel**: SIEM y análisis de seguridad

Networking:

- **Azure Virtual Network**: Aislamiento de red

- **Azure Private Link**: Conectividad privada
- **Azure Traffic Manager**: Distribución global de tráfico
- **Azure CDN**: Aceleración de contenido estático

Stack de Seguridad



Security Stack - Defense in Depth Architecture

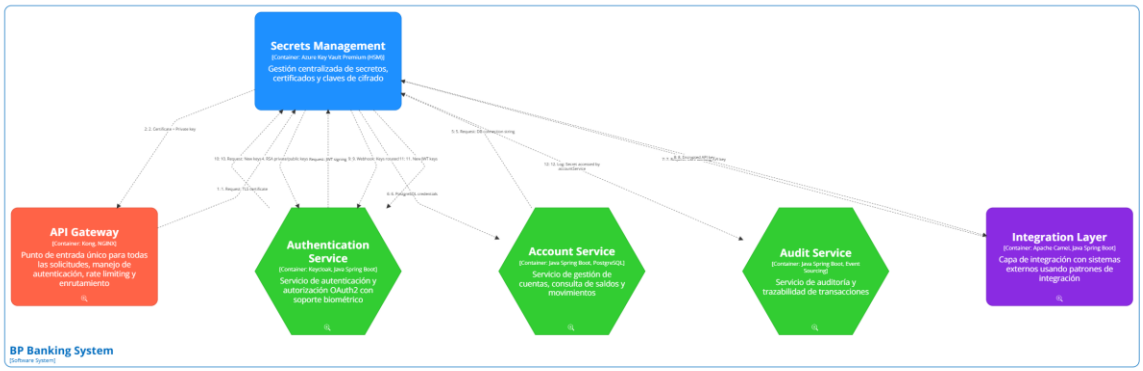
Arquitectura de Datos

Arquitectura de Datos - Cache Aside y Event Sourcing

Patrones de persistencia: Cache-Aside para clientes frecuentes, Event Sourcing para auditoría

Saturday, November 29, 2025 at 8:28 PM Ecuador Time

Administración de Secretos



Secrets Management - Centralizado con Azure Key Vault
Todos los servicios obtienen secretos de manera segura desde Key Vault
Saturday, November 29, 2025 at 8:30 PM Ecuador Time

11. PATRONES DE ALTA DISPONIBILIDAD Y TOLERANCIA A FALLOS

11.1 Circuit Breaker Pattern

```
@Component
public class ExternalServiceClient {
    @CircuitBreaker(name = "corebanking", fallbackMethod = "fallbackGetAccount")
    @Retry(name = "corebanking")
    @TimeLimiter(name = "corebanking")
    public CompletableFuture<Account> getAccount(String accountId) {
        return CompletableFuture.supplyAsync(() ->
            restTemplate.getForObject("/accounts/{id}", Account.class, accountId));
    }
    public CompletableFuture<Account> fallbackGetAccount(String accountId, Exception ex) {
        return CompletableFuture.completedFuture(getCachedAccount(accountId));
    }
}
```

11.2 Saga Pattern para Transferencias

```
@Service
public class TransferSaga {
    @SagaOrchestrationStart
    public void processTransfer(TransferRequest request) {
```

```
        sagaManager.choreography()
            .step("validateAccount").compensate("releaseAccountLock")
            .step("reserveFunds").compensate("releaseFunds")
            .step("executeTransfer").compensate("reverseTransfer")
            .step("sendNotification")
            .execute(request);
    }
}
```

11.3 Event Sourcing para Auditoría

```
@Entity
public class AccountEvent {
    private String eventId;
    private String accountId;
    private String eventType;
    private LocalDateTime timestamp;
    private String eventData;
    private String userId;
    // Reconstrucción del estado actual desde eventos
    public Account rebuildAccountState(String accountId) {
        List<AccountEvent> events = eventRepository
            .findByAccountIdOrderByTimestamp(accountId);
        return events.stream()
            .reduce(new Account(), this::applyEvent, (a1, a2) -> a2);
    }
}
```

12. ESTRATEGIA DE MONITOREO Y OBSERVABILIDAD

12.1 Métricas Clave

- **Latency**: Tiempo de respuesta de APIs
- **Traffic**: Requests per second
- **Errors**: Error rate y tipos de error

- **Saturation**: Utilización de recursos

12.2 Distributed Tracing

```
@RestController
@Slf4j
public class TransferController {
    @Autowired
    private TransferService;
    @PostMapping("/transfers")
    @Timed(name = "transfer.duration", description = "Transfer processing time")
    public ResponseEntity<TransferResponse> createTransfer(
            @RequestBody TransferRequest request,
            @RequestHeader Map<String, String> headers) {
        try (MDCCloseable = MDCCloseable.put("correlationId",
                headers.getOrDefault("X-Correlation-ID", UUID.randomUUID().toString()))) {
            log.info("Processing transfer request: {}", request.getTransferId());
            TransferResponse response = transferService.processTransfer(request);
            return ResponseEntity.ok(response);
        }
    }
}
```

12.3 Health Checks y Auto-healing

```
# Kubernetes Health Check Configuration
livenessProbe:
  httpGet:
    path: /actuator/health/liveness
    port: 8080
  initialDelaySeconds: 60
  periodSeconds: 30
  timeoutSeconds: 10
  failureThreshold: 3

readinessProbe:
  httpGet:
    path: /actuator/health/readiness
    port: 8080
  initialDelaySeconds: 30
```

```
    periodSeconds: 10
    timeoutSeconds: 5
    failureThreshold: 3

# Auto-scaling configuration
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: banking-api-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: banking-api
  minReplicas: 3
  maxReplicas: 20
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70
  - type: Resource
    resource:
      name: memory
      target:
        type: Utilization
        averageUtilization: 80
```

13. ESTRATEGIA DE RECUPERACIÓN ANTE DESASTRES (DR)

13.1 RPO y RTO Objetivos

- **RPO (Recovery Point Objective)**: 15 minutos
- **RTO (Recovery Time Objective)**: 2 horas

13.2 Estrategia Multi-Region

```
# Primary Region: East US
# Secondary Region: West US

# Database Replication
primary_database:
  location: "East US"
  replication:
    - location: "West US"
      mode: "asynchronous"
    - location: "Central US"
      mode: "synchronous"
```

```
# Application Deployment
traffic_manager:
 routing_method: "priority"
 endpoints:
  - name: "primary"
    location: "East US"
    priority: 1
  - name: "secondary"
    location: "West US"
    priority: 2
```

13.3 Backup Strategy

- **Database**: Backup incremental cada hora, completo diario
- **Files**: Replicación automática cross-region
- **Configuration**: Versionado en Git con GitOps
- **Secrets**: Sincronización automática entre Key Vaults

14. CONSIDERACIONES DE SEGURIDAD

14.1 Security by Design

- **Zero Trust Architecture**: Verificación continua
- **Defense in Depth**: Múltiples capas de seguridad
- **Least Privilege**: Acceso mínimo necesario
- **Encryption**: End-to-end encryption

14.2 API Security

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {
  @Bean
  public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    return http
      .oauth2ResourceServer(oauth2 -> oauth2
        .jwt(jwt -> jwt.jwtAuthenticationConverter(jwtConverter())))
      .sessionManagement(session -> session
        .sessionCreationPolicy(SessionCreationPolicy.STATELESS))
```

```
        .authorizeHttpRequests(authz -> authz
          .requestMatchers("/actuator/health").permitAll()
          .requestMatchers("/api/v1/accounts/**").hasAuthority("SCOPE_banking:read")
          .requestMatchers("/api/v1/transfers/**").hasAuthority("SCOPE_banking:write")
          .anyRequest().authenticated())
        .headers(headers -> headers
          .frameOptions().deny()
          .contentTypeOptions().and()
          .httpStrictTransportSecurity(hstsConfig -> hstsConfig
            .maxAgeInSeconds(31536000)
            .includeSubdomains(true)))
        .build();
    }
}
```

14.3 Arquitectura de Seguridad en Capas (Defense in Depth)

La arquitectura implementa 7 capas de seguridad según ISO 27001:

Capa 1: DDoS Protection (Azure DDoS Protection Standard)

- Mitigación automática de ataques volumétricos < 60 segundos
- Protección hasta 54 Tbps (mayor ataque registrado: 3.5 Tbps)
- Protocolos: Layer 3/4 (TCP SYN flood, UDP amplification)
- Costo: $2,944/mes (justificado: 1 ataque mitigado paga 10 meses)

Capa 2: WAF (Web Application Firewall) - Azure Application Gateway v2

- Protección OWASP Top 10 (SQL injection, XSS, CSRF)
- Reglas customizadas para patrones bancarios
- Core Rule Set (CRS) 3.2: 250+ reglas predefinidas
- Rate limiting: 1000 req/min por IP
- Geo-blocking: bloqueo automático países de alto riesgo
- Bot protection: detección de bots maliciosos con ML

Capa 3: Firewall (Azure Firewall Premium)

- IDPS (Intrusion Detection/Prevention System)
- TLS inspection para tráfico cifrado
- Network rules: Whitelist de IPs permitidas
- Application rules: FQDN filtering
- Threat intelligence: feed de Microsoft con IPs maliciosas

Capa 4: Load Balancer (Azure Load Balancer Standard)

- Distribución de tráfico previene saturación
- Health probes: detecta backends caídos en < 10 segundos
- Session affinity: sticky sessions para workflows con estado

Capa 5: API Gateway (Kong)

- OAuth2 token validation (JWT signature + expiration)
- Rate limiting granular (por usuario, por endpoint)
- Plugins de seguridad:
    1. JWT plugin: validación de tokens
    2. Rate Limiting: 100 req/min por usuario
    3. IP Restriction: whitelist de IPs corporativas
    4. CORS: configuración restrictiva por origen

Capa 6: Application Security (Microservicios)

- Spring Security: método-level security con @PreAuthorize
- OWASP Dependency Check: escaneo de vulnerabilidades en librerías
- SonarQube: análisis de código estático (security hotspots)
- Container scanning: Trivy para imágenes Docker

Capa 7: Data Security

- Encryption at rest: TDE (Transparent Data Encryption) en PostgreSQL
- Encryption in transit: TLS 1.3 obligatorio (TLS 1.2 deshabilitado)
- Key Management: Azure Key Vault Premium (HSM-backed)
- PII masking: enmascaramiento de datos sensibles en logs

Capa Transversal: Secrets Management (Azure Key Vault)

- Centralización elimina secretos hardcodeados
- Rotación automática de secretos cada 90 días
- Auditoría completa de accesos a secretos
- HSM-backed para claves criptográficas (FIPS 140-2 Level 2)
- Acceso: Managed Identities (sin credenciales en código)

Justificación de la arquitectura multicapa:

- Defensa en profundidad: compromiso de 1 capa no afecta otras
- Compliance: cumple PCI DSS requirement 6.6 (WAF obligatorio)
- Zero Trust: verificación en cada capa (nunca confiar, siempre verificar)
- Auditoría: logs en cada capa para análisis forense

Costos mensuales estimados (seguridad):

- DDoS Protection: $2,944
- Application Gateway WAF: $500
- Azure Firewall Premium: $1,250
- Key Vault Premium: $200

TOTAL: $4,894/mes (0.8% del costo total infraestructura)

15. IMPLEMENTACIÓN POR FASES

Fase 1 (MVP): Funcionalidades Core

- Autenticación básica OAuth2
- Consulta de saldos y movimientos
- Transferencias entre cuentas propias
- Notificaciones básicas (email/SMS)

Fase 2: Funcionalidades Avanzadas

- Transferencias interbancarias
- Reconocimiento facial (onboarding)

- Autenticación biométrica
- Dashboard analytics

Fase 3: Optimización y Escalabilidad

- Cache inteligente
- ML para fraud detection
- Optimizaciones de rendimiento
- Análisis predictivo

16. ESTIMACIÓN Y OPTIMIZACIÓN DE COSTOS (Azure)

16.1. Costos Mensuales Estimados

Compute (AKS):

- **Node pool (Standard_D4s_v3):** 5 nodos x $140 = $700
- **Auto-scaling:** +3 nodos pico (30% tiempo) = $126
- **Total Compute:** $826/mes

Databases:

- **Azure PostgreSQL (General Purpose, 4 vCores):** $350
- **Azure Cosmos DB (MongoDB API, 10k RU/s):** $584
- **Azure Redis Cache (Standard C3, 6GB):** $150
- **Total Databases:** $1,084/mes

Networking:

- **Azure Load Balancer Standard:** $50
- **Application Gateway WAF v2:** $500
- **Azure Firewall Premium:** $1,250
- **DDoS Protection Standard:** $2,944
- **Bandwidth (5TB salida):** $430
- **Total Networking:** $5,174/mes

Storage & Backup:

- **Azure Blob Storage (1TB):** $20
- **Azure Backup (500GB):** $50
- **Total Storage:** $70/mes

Monitoring & Security:

- **Azure Monitor + Log Analytics:** $200
- **Azure Key Vault Premium:** $200
- **Application Insights:** $100
- **Total Monitoring:** $500/mes

Event Streaming:

Azure Event Hubs (Standard, 20 TU): $500

TOTAL MENSUAL: $8,154

TOTAL ANUAL: $97,848

16.2 Estrategias de Optimización

1. Reserved Instances (RIs):

- **Justificación:** Ahorro 30-50% en recursos predecibles
- **Aplicar a:** AKS nodes, databases
- **Ahorro estimado:** $3,000/año

2. Auto-scaling Inteligente:

- **Justificación:** Solo pagar por uso en horas pico
- HPA con métricas custom (transacciones/segundo)
- **Ahorro estimado:** $1,500/año

3. Spot Instances (Batch Processing):

- **Justificación:** 90% descuento para trabajos no críticos
- **Aplicar a:** Reconciliation engine, ML training
- **Ahorro estimado:** $2,000/año

4. Data Lifecycle Management:

- **Justificación:** Movimiento automático a tiers económicos
- **Implementar:**
  - **Logs >90 días → Archive tier (80% más barato)**
  - **Backups >1 año → Cool tier**
- **Ahorro estimado:** $800/año

5. Caching Agresivo:

- **Justificación:** Reduce consultas a databases
- Redis cache con TTL optimizado
- **Ahorro indirecto:** Reduce RU/s necesarias en Cosmos DB
- **Ahorro estimado:** $1,200/año

AHORRO TOTAL ANUAL: $8,500 (8.7% reducción)

COSTO OPTIMIZADO ANUAL: $89,348

16.3 Justificación de Inversión

ROI (Return on Investment):

- **Costo infraestructura:** $89k/año
- **Costo desarrollo/mantenimiento:** $300k/año
- **Costo total:** $389k/año

Beneficios tangibles:

- **Reducción tiempo transacciones:** 60% (de 5min a 2min)
- **Incremento transacciones digitales:** +40% (reduce costos sucursales)
- **Prevención fraude:** $500k ahorrados/año (2% transacciones)
- **Reducción downtime:** 99.9% SLA (ahorro $200k/año)

CONCLUSIONES

Esta arquitectura propuesta cumple con todos los requerimientos técnicos y normativos indicados en las primeras secciones, con lo cual garantiza:

1. **Escalabilidad**: Arquitectura de microservicios cloud-native
2. **Seguridad**: Múltiples capas de protección y cumplimiento normativo
3. **Disponibilidad**: Patrones de tolerancia a fallos y recuperación automática
4. **Mantenibilidad**: Código desacoplado y reutilizable
5. **Observabilidad**: Monitoreo completo y trazabilidad
6. **Flexibilidad**: Capacidad de extensión y adaptación futura

La solución está diseñada para ser robusta, segura y escalable, cumpliendo con las mejores prácticas de la industria financiera y los estándares internacionales de arquitectura de software.