ARQUITECTURA DE SOLUCION-SISTEMA DE BANCA DIGITAL BP

DISEÑA BAJO MODELO C4

Prueba técnica-Arquitectura de soluciones

Autor:

Alex Fabricio Anchundia Mero

Fecha:

22/11/2025

Versión:

1.0

Contenido

1. RESUMEN EJECUTIVO

El presente documento describe la arquitectura propuesta para el sistema de Banca Digital de BP. El objetivo principal es diseñar una solución moderna, segura y escalable que soporte las operaciones esenciales del cliente: consulta de movimientos, transferencias, pagos, notificaciones y un proceso de onboarding basado en biometría facial.

La arquitectura se fundamenta en microservicios desacoplados, un API Gateway que centraliza seguridad y ruteo, y mecanismos robustos de autenticación (OAuth2 + PKCE). También se plantean componentes como auditoría inmutable, un modelo de cache para consultas frecuentes (CQRS) y la integración directa con los sistemas Core bancarios y el Sistema Complementario.

El diseño se representa utilizando el modelo C4 (Contexto, Contenedores y Componentes), lo que permite visualizar claramente cómo interactúan los usuarios, servicios y sistemas externos. Además, la arquitectura está preparada para ejecutarse en AWS o Azure, garantizando alta disponibilidad, seguridad y capacidad de escalar según la demanda.

Esta arquitectura garantiza:

- Seguridad bancaria de nivel industrial.
- Escalabilidad horizontal.
- Alta disponibilidad.
- Integración transparente con sistemas Core.
- Mantenimiento futuro y extensibilidad.

2. REQUERIMIENTOS

2.1 Requerimientos Funcionales

- Consulta de datos básicos del cliente.
- Consulta de movimientos financieros.
- Realización de transferencias (propias e interbancarias).
- Recepción de notificaciones sobre movimientos y transacciones.
- Onboarding biométrico para nuevos clientes.
- Acceso a detalles de productos bancarios.

- Integración con sistemas Core bancarios y complementarios.
- Registro de auditoría de todas las acciones del cliente.

2.2 Requerimientos No Funcionales

- Alta disponibilidad (99.9%+).
- Baja latencia para consultas frecuentes.
- Seguridad: TLS 1.3, OAuth2, OIDC, 2FA, cifrado en reposo.
- Cumplimiento normativo: protección de datos personales, PCI DSS, ISO 27001.
- Observabilidad: monitoreo, alarmas, dashboards.
- Tolerancia a fallos y auto-recuperación.
- Escalabilidad horizontal.
- Arquitectura desacoplada y cohesionada.
- Manejo eficiente de costos.
- Persistencia de auditoría inmutable.

3. ARQUITECTURA GENERAL

La arquitectura se basa en microservicios independientes que se comunican a través de un API Gateway. Cada microservicio cumple una función específica y se integra con sistemas internos y externos sin afectar a los demás. Este enfoque permite escalar componentes críticos, mejorar mantenibilidad y reducir el impacto de fallos.

La autenticación se gestiona mediante OAuth2 Authorization Code, con PKCE para la aplicación móvil. La información de clientes frecuentes se sirve desde un modelo de lectura optimizado (CQRS), mientras que la auditoría se almacena en una base inmutable diseñada para cumplir lineamientos financieros.

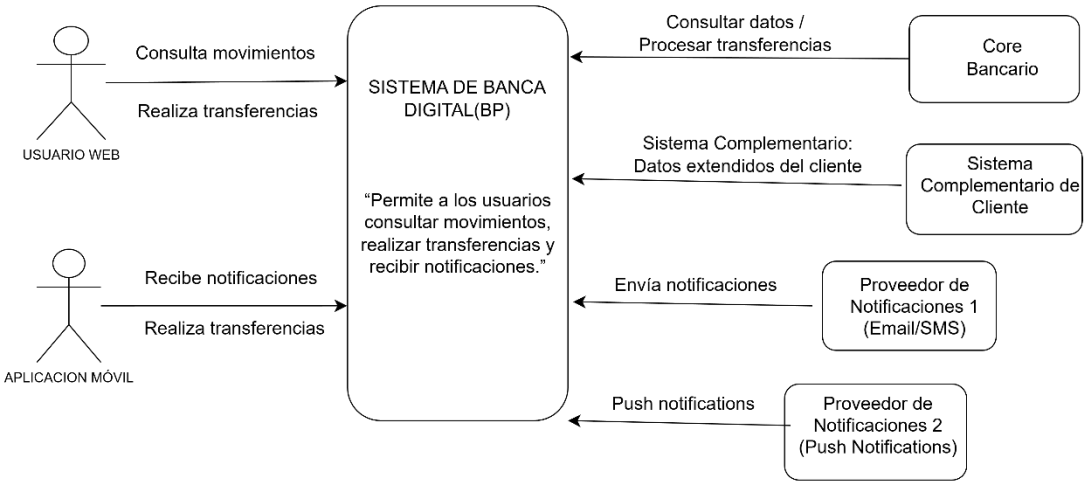
La solución se integra directamente con:

- El Core Bancario
- El Sistema Complementario
- Proveedores de notificaciones
- Servicios biométricos

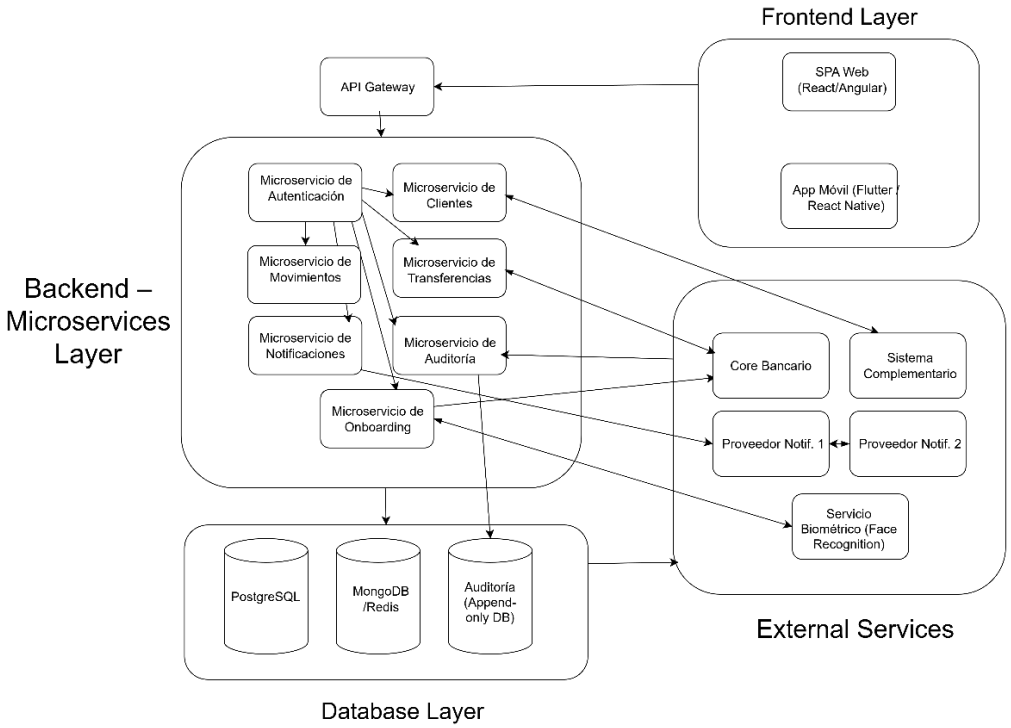
Los diagramas detallados del modelo C4 describen el sistema a nivel de contexto, contenedores y componentes clave.

4. MODELO C4

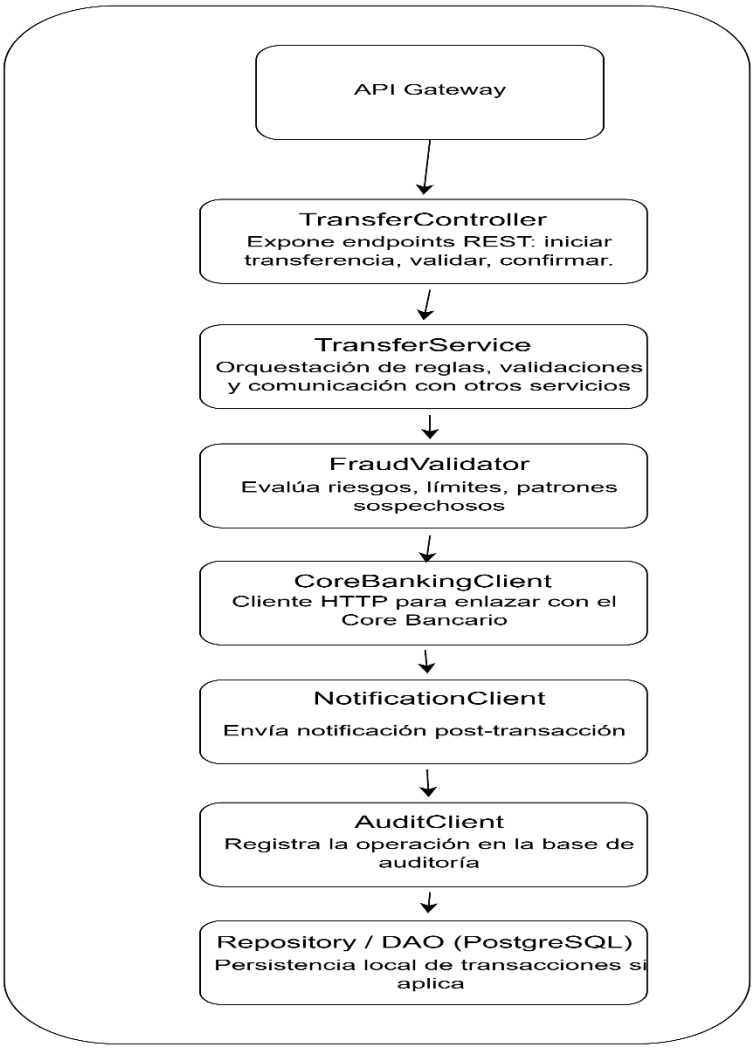
4.1-DIAGRAMA DE CONTEXTO



4.2-DIAGRAMA DE CONTENEDORES



4.3-DIAGRAMA DE COMPONENTES



5. Decisiones Arquitectónicas

Las siguientes decisiones arquitectónicas se tomaron considerando la naturaleza del dominio bancario, la necesidad de seguridad reforzada y la escalabilidad del sistema. Cada decisión está acompañada de una justificación técnica y operacional.

5.1 Uso de Microservicios

Decisión:

Adoptar una arquitectura basada en microservicios para segmentar la solución en dominios funcionales claros.

Justificaciones:

1. Permite escalar de forma independiente servicios críticos como transferencias y movimientos sin afectar el resto de la plataforma.
2. Reduce el impacto de fallos y facilita el despliegue continuo, permitiendo actualizaciones sin interrumpir la operación general.

5.2 API Gateway como punto de entrada

Decisión:

Utilizar un API Gateway centralizado para recibir todas las solicitudes desde SPA y aplicación móvil.

Justificaciones:

1. Centraliza validación de tokens, protección por rate limiting y control de acceso, evitando duplicación de lógica en microservicios.
2. Simplifica la exposición externa permitiendo que los clientes solo consuman una URL unificada.

5.3 Autenticación mediante OAuth2 Authorization Code + PKCE

Decisión:

Implementar OAuth2 Authorization Code, utilizando PKCE en el aplicativo móvil.

Justificaciones:

1. PKCE evita ataques de interceptación en aplicaciones móviles donde no existe un backend propio para proteger secretos.
2. OAuth2 es el estándar de la industria, compatible con proveedores como Cognito, Azure AD B2C o Keycloak.

5.4 Persistencia de datos mediante CQRS

Decisión:

Separar la persistencia entre un modelo transaccional (PostgreSQL) y un modelo de lectura rápido (Redis o MongoDB).

Justificaciones:

1. Reduce la carga del Core Bancario al almacenar los datos más consultados en un read model local.
2. Mejora la latencia en consultas de movimientos y datos del cliente.

5.5 Base de Auditoría Inmutable

Decisión:

Crear un repositorio append-only exclusivamente para auditorías y eventos críticos.

Justificaciones:

1. Cumple normas y auditorías bancarias que exigen trazabilidad sin alteración.
2. Facilita análisis de fraude y permite auditorías regulatorias claras.

5.6 Integración directa con Core Bancario y Sistema Complementario

Decisión:

Separar la integración en clientes específicos (CoreBankingClient, ComplementaryClient).

Justificaciones:

1. Aísla cambios futuros en sistemas externos sin afectar la lógica de dominio.
2. Permite aplicar patrones de resiliencia como retry, timeout y circuit breaker.

5.7 Uso de Cache para Clientes Frecuentes

Decisión:

Mantener un almacenamiento temporal para sesiones activas y datos repetidos.

Justificaciones:

1. Evita consultas repetitivas al Core Bancario, reduciendo costos y latencia.
2. Aumenta la respuesta de la plataforma en operaciones cotidianas del cliente.

5.8 Onboarding Biométrico con Servicio Especializado

Decisión:

Externalizar la verificación facial y reconocimiento de documentos a un proveedor de IA en la nube.

Justificaciones:

1. Los servicios biométricos requieren modelos entrenados y certificaciones de seguridad difíciles de mantener internamente.
2. Garantiza precisión, soporte continuo y compliance con normas de verificación de identidad.

5.9 Infraestructura Cloud-Ready (AWS/Azure)

Decisión:

Diseñar la arquitectura para nube pública utilizando contenedores administrados y servicios gestionados.

Justificaciones:

1. Permite crecer bajo demanda con costos controlados mediante autoscaling.
2. Facilita la resiliencia multi-AZ, indispensable para plataformas bancarias.

6. Arquitectura de Autenticación

La autenticación del sistema se basa en OAuth2 bajo el flujo Authorization Code. Para la aplicación móvil se utiliza PKCE, mientras que la SPA web usa el flujo clásico con cliente confidencial a través del backend de autenticación.

El **API Gateway** actúa como primer filtro, verificando tokens de acceso en cada solicitud. El servicio de autenticación delega la gestión de identidades a un Identity Provider (por ejemplo, Cognito, Azure AD B2C o Keycloak), lo que permite:

- Soportar multifactor (MFA) cuando sea requerido.
- Integrarse con directorios corporativos si la organización lo necesita.

- Centralizar recuperación de contraseñas, bloqueo de cuentas y políticas de contraseña.

El flujo básico es:

1. El usuario inicia sesión desde la SPA o la app móvil.
2. El navegador/app es redirigido al proveedor de identidad.
3. Tras autenticarse, se devuelve un *authorization code*.
4. El cliente intercambia ese código por un *access token* (y, si aplica, un *refresh token*).
5. El token se incluye en cada llamada hacia el API Gateway, que lo valida antes de redirigir al microservicio correspondiente.

Este enfoque permite desacoplar la lógica de negocio de la gestión de identidades, manteniendo la seguridad alineada con estándares de la industria.

7. Arquitectura de Onboarding Biométrico

El proceso de onboarding está pensado para nuevos clientes que desean abrir productos o habilitar su acceso digital sin acudir físicamente a una agencia.

El flujo propuesto es:

1. Desde la app móvil, el usuario inicia el proceso de alta.
2. Se le solicita capturar una selfie y fotografías de su documento de identidad.
3. El **Microservicio de Onboarding** envía estas imágenes a un servicio biométrico en la nube (por ejemplo, AWS Rekognition o Azure Face API).
4. El servicio biométrico valida:
    - Coincidencia entre selfie y foto del documento.
    - Calidad y vigencia del documento.
5. Si la validación es exitosa, el onboarding crea el cliente en el Core Bancario y registra la operación en la base de auditoría.
6. Finalmente, se habilita la cuenta de acceso digital y se envía una notificación de confirmación.

Este diseño mantiene la lógica de negocio de onboarding dentro de un microservicio especializado, mientras delega el reconocimiento facial y la verificación de documentos a plataformas que ya cuentan con modelos entrenados, certificaciones y evolución continua.

8. Arquitectura de Auditoría

La auditoría es un componente clave en un entorno bancario. Se propone un **Microservicio de Auditoría** dedicado, con una base de datos de tipo *append-only*, diseñada exclusivamente para escritura.

Se registran, al menos, los siguientes eventos:

- Inicios y cierres de sesión.
- Intentos fallidos de autenticación.
- Transferencias creadas, aprobadas o rechazadas.
- Cambios de datos personales y configuración sensible.
- Operaciones realizadas durante el onboarding.

Cada evento incluye:

- Identificador del usuario.
- Marca de tiempo (timestamp).
- Tipo de operación.
- Resultado (éxito/error).
- Metadatos relevantes (IP, dispositivo, canal).

La arquitectura evita modificar o borrar registros de auditoría. Cualquier corrección se registra como un nuevo evento, preservando el historial completo. Esto facilita revisiones internas, auditorías externas y análisis forense en caso de incidentes.

9. Arquitectura de Persistencia de Datos

Se plantea un modelo de persistencia dividido en tres capas:

1. **Base transaccional (PostgreSQL o Azure SQL / Aurora)**
   Almacena la información de dominio: cuentas, clientes, productos, transacciones y configuraciones esenciales.
2. **Modelo de lectura (CQRS) con Redis, MongoDB o CosmosDB**
   Almacena información preprocesada para acelerar consultas frecuentes (por ejemplo, últimos movimientos, saldos consolidados o perfiles de cliente).
   Se actualiza de forma asíncrona cuando se ejecutan operaciones en el Core o en la base transaccional.
3. **Base de auditoría**
   Diseñada únicamente para escritura secuencial, optimizada para lectura analítica posterior.

Este enfoque permite:

- Mantener la integridad y consistencia en la base transaccional.
- Reducir la latencia de consultas frecuentes usando el modelo de lectura.
- Separar las necesidades de auditoría del resto de la operación.

10. Integración con Sistemas Externos

El sistema se integra con varios componentes externos:

- **Core Bancario:**
  Responsable de saldos, movimientos, débitos, créditos y administración de productos.
  La integración se hace mediante un cliente especializado (CoreBankingClient) por cada microservicio que lo requiera (por ejemplo, Movimientos y Transferencias).
- **Sistema Complementario de Cliente:**
  Provee información extendida del cliente (datos adicionales, segmentación, campañas, etc.).
  Un cliente específico consulta este sistema cuando se necesita enriquecer la vista del usuario.
- **Proveedores de notificaciones (Email, SMS, Push):**
  El Microservicio de Notificaciones se integra con al menos dos proveedores,

para evitar depender de un solo canal. Si uno falla, se utiliza el otro como fallback.

- **Servicios biométricos:**
  Utilizados por el Microservicio de Onboarding para validación facial y documental.

Para todas estas integraciones se recomienda:

- Establecer timeouts razonables.
- Implementar reintentos limitados.
- Usar circuit breakers y colas asíncronas cuando la operación no requiera respuesta inmediata.

Esto mejora la resiliencia del sistema frente a fallos o degradación en servicios externos.

11. Infraestructura en la Nube (AWS/Azure)

La arquitectura está pensada para ejecutarse en una nube pública, aprovechando servicios gestionados para reducir carga operativa y mejorar la resiliencia.

11.1 Opción AWS

- **Cómputo:**
  - Amazon ECS Fargate o EKS para desplegar los microservicios en contenedores.
- **API & Networking:**
  - Amazon API Gateway como punto de entrada para SPA y app.
  - Elastic Load Balancer para distribuir tráfico entre instancias.
- **Datos:**
  - Amazon Aurora PostgreSQL para datos transaccionales.
  - DynamoDB o Elasticache Redis como modelo de lectura/cache.
  - S3/Glacier para almacenamiento de archivos de onboarding y auditoría histórica.
- **Seguridad:**
  - AWS Cognito para gestión de usuarios, OAuth2 y PKCE.
  - AWS WAF + Shield para protección perimetral.

- IAM para control de permisos.

- **Monitoreo:**
  - CloudWatch (logs, métricas, alarmas).
  - X-Ray para trazas distribuidas.

11.2 Opción Azure

- **Cómputo:**
  - Azure Kubernetes Service (AKS) para orquestación de microservicios.
  - Alternativamente, App Services para microservicios gestionados.
- **API & Networking:**
  - Azure API Management para exponer APIs, aplicar políticas y seguridad.
  - Application Gateway + WAF para enrutamiento y protección.
- **Datos:**
  - Azure SQL Database para datos transaccionales.
  - CosmosDB o Redis Cache para el modelo de lectura.
  - Blob Storage para archivos de onboarding y almacenamiento frío.
- **Seguridad:**
  - Azure AD B2C para OAuth2/OIDC y PKCE.
  - Key Vault para gestión de secretos.
  - Defender for Cloud para monitoreo de amenazas.
- **Monitoreo:**
  - Azure Monitor + Log Analytics para dashboards y alertas.
  - Application Insights para trazas y telemetría.

En ambos casos, la arquitectura se apoya en zonas de disponibilidad, autoscaling y backups automáticos para garantizar continuidad del servicio.

12. Alta Disponibilidad y Recuperación ante Desastres

Para garantizar la continuidad del negocio se consideran los siguientes elementos:

- **Despliegue multi-AZ:**
  Los servicios críticos y las bases de datos se despliegan en al menos dos zonas de disponibilidad distintas.
- **Balanceo de carga:**
  Un Application Load Balancer distribuye el tráfico entre las instancias de los microservicios.
- **Autoscaling:**
  Se habilita escalado automático basado en CPU, memoria o métricas de negocio (por ejemplo, número de transacciones por minuto).
- **Backups y snapshots:**
  La base de datos transaccional y la auditoría cuentan con copias de seguridad periódicas y pruebas regulares de restauración.
- **Plan de recuperación ante desastres (DR):**
  Se define un RPO (Recovery Point Objective) y RTO (Recovery Time Objective) claros. En escenarios más exigentes, se puede habilitar replicación cross-region para restaurar el servicio en otra región cloud en caso de un incidente mayor.

13. Monitoreo y Observabilidad

La observabilidad es crucial para detectar problemas antes de que impacten a los clientes.

La propuesta incluye:

- **Logs centralizados:**
  Todos los microservicios envían logs a una plataforma central (CloudWatch, Log Analytics, ELK, etc.).
- **Métricas de negocio y técnicas:**
  - Tiempo de respuesta por endpoint.
  - Tasa de errores.
  - Número de transferencias por minuto.
  - Cantidad de intentos fallidos de login.
- **Trazabilidad distribuida:**
  Cada solicitud lleva un *correlation ID*, lo que permite seguirla a través de varios microservicios.

- **Alertas y dashboards:**
  Se configuran paneles de monitoreo y alertas en tiempo real (correo, Slack, Teams) para eventos críticos, como:
  - Incremento abrupto de errores 5xx.
  - Latencias elevadas en el Core Bancario.
  - Caídas de algún proveedor de notificaciones.

Este enfoque permite reaccionar rápido, identificar cuellos de botella y mejorar la experiencia del usuario.

14. Manejo de Costos

Aunque se trata de un sistema bancario con alta criticidad, es importante optimizar costos sin comprometer la calidad del servicio.

Algunas decisiones para controlar costos:

- **Autoscaling:**
  Las instancias de microservicios crecen y disminuyen según demanda, evitando sobredimensionar la infraestructura.
- **Uso de servicios gestionados:**
  Servicios como API Gateway, bases de datos administradas y colas de mensajes reducen el esfuerzo operativo.
- **Almacenamiento por niveles:**
  La auditoría antigua y archivos de onboarding se mueven a almacenamiento frío o de bajo costo (Glacier, Archive, etc.).
- **Cache efectiva:**
  Disminuir las llamadas al Core Bancario y a la base transaccional reduce el uso de recursos y por tanto el costo final.
- **Monitoreo de consumo:**
  Revisar periódicamente métricas de uso, ajustar tamaños de instancias y políticas de escalado según el comportamiento real.

Conclusiones

La arquitectura propuesta aborda los principales retos de una plataforma de banca digital moderna: seguridad, disponibilidad, integración con sistemas legados y capacidad de crecimiento.

Mediante el uso de microservicios, un API Gateway, autenticación basada en estándares, un modelo de persistencia optimizado y una capa robusta de auditoría, el sistema puede evolucionar sin comprometer la estabilidad. La preparación para despliegue en AWS o Azure abre la puerta a una operación resiliente, escalable y alineada con las mejores prácticas del mercado.

En conjunto, el diseño busca un balance entre simplicidad, robustez y capacidad de adaptación, ofreciendo una base sólida para seguir incorporando nuevos productos, canales y funcionalidades en el futuro.