Sistema de Banca BP

Table of Contents

Resume ejecutivo

El presente documento detalla la arquitectura de solución propuesta para el nuevo sistema de Banca para la entidad BP. El objetivo principal es modernizar la plataforma digital permitiendo a los usuarios consultar movimientos, realizar transferencias y pagos a través de canales Web y Móviles.

La solución se ha diseñado bajo una arquitectura de Microservicios, utilizando el modelo C4 para la documentación. Se han priorizado atributos de calidad como la Alta Disponibilidad (HA), la seguridad mediante estándares OAuth2 y Biometría, y el rendimiento mediante el uso de estrategias de Caché y comunicación asíncrona. Asimismo, se incluye una estrategia de integración con el Core Bancario legado mediante capas de anticorrupción (ACL) para garantizar la escalabilidad y mantenibilidad del sistema.

Introducción y contexto

Contexto del Negocio

En el actual panorama financiero, la transformación digital ha dejado de ser una ventaja competitiva para convertirse en una necesidad operativa. La entidad financiera BP Bank se enfrenta al desafío de modernizar sus canales de atención para satisfacer las demandas de una base de usuarios creciente que exige disponibilidad inmediata, seguridad robusta y una experiencia de usuario fluida.

Actualmente, la entidad opera sobre sistemas centrales (Core Bancario Legacy) y sistemas satélites que, si bien son estables, carecen de la agilidad necesaria para exponer servicios directos a internet de forma segura y escalable.

Definición del Problema

El principal obstáculo tecnológico radica en la rigidez de la infraestructura heredada. Exponer el Core Bancario directamente a internet representaría un riesgo inaceptable de seguridad y rendimiento. Además, la normativa bancaria vigente exige estrictos controles de autenticación (Biometría), notificación transaccional y protección de datos, los cuales no pueden ser implementados fácilmente en las plataformas antiguas.

La ausencia de una arquitectura desacoplada ha dificultado la implementación de nuevos canales digitales (Web y Móvil), generando fricción en la experiencia del cliente y limitando la capacidad del banco para innovar frente a sus competidores.

Objetivos del Proyecto

El objetivo de este proyecto es diseñar y documentar la arquitectura de software para el nuevo sistema de Banca por Internet BP. La solución propuesta debe:

1. **Habilitar Canales Digitales:** Proveer interfaces Web (SPA) y Móviles nativas para consultas, transferencias y pagos.
2. **Garantizar la Interoperabilidad:** Integrar los servicios modernos con el Core Bancario existente y redes interbancarias sin comprometer la estabilidad del sistema legado.
3. Asegurar la Continuidad Operativa: Diseñar una infraestructura tolerante a fallos y desastres, capaz de operar 24/7.
4. **Cumplimiento Normativo:** Implementar estándares de seguridad avanzados como OAuth 2.0 y validación biométrica facial para el *onboarding* y autorización de transacciones.

Requerimientos de calidad

Para garantizar el cumplimiento normativo y la satisfacción del usuario, la arquitectura satisface los siguientes atributos de calidad (NFRs):

- **Alta Disponibilidad (HA):** El sistema está diseñado para operar 24/7. Se logra mediante el despliegue de microservicios redundantes y bases de datos en clúster.
- **Seguridad y Cumplimiento:**
    - Implementación de OAuth 2.0 / OIDC para la gestión de identidades.
    - Uso de Biometría Facial para el *onboarding* y validación de operaciones críticas.
    - Cumplimiento de la Ley de Protección de Datos mediante cifrado de datos en reposo y tránsito (TLS 1.3).
- **Rendimiento y Baja Latencia:** Implementación de Redis para datos de acceso frecuente (clientes VIP) y procesamiento asíncrono para auditoría y notificaciones.
- **Interoperabilidad:** Integración con sistemas legados (Core) y externos (Pasarelas, Reniec) utilizando estándares REST e ISO8583.

Stack tecnológico

El diseño de la arquitectura del sistema "Banca Internet BP" se ha fundamentado en un análisis exhaustivo de los requisitos funcionales y no funcionales. Cada tecnología y patrón seleccionado busca maximizar la robustez, seguridad y escalabilidad de la solución. A continuación, se presenta la matriz de decisiones técnicas, detallando la justificación teórica detrás de cada elección para garantizar que la solución no solo cumpla con las necesidades actuales, sino que sea sostenible y evolutiva en el tiempo.

Decisión	**Tecnología Seleccionada**	**Justificación Teórica (Mínimo 2 razones)**
Frontend Web (SPA)	React (con TypeScript)	1. **Virtual DOM:** React utiliza una representación virtual del DOM para calcular eficientemente los cambios mínimos necesarios en la interfaz. Esto asegura una alta velocidad de respuesta al renderizar listas extensas de movimientos bancarios sin recargar la página. 2. **Arquitectura Basada en Componentes**: Permite encapsular la lógica y el estilo de elementos UI en piezas reutilizables y aisladas, facilitando el mantenimiento y las pruebas unitarias. Además, React ofrece una biblioteca ligera basada en componentes funcionales. Nota: Se descartó Angular debido a que su mecanismo tradicional de detección de cambios puede resultar más pesado en interfaces con alta densidad de datos en tiempo real si no se optimiza manualmente.
Frontend Móvil	Flutter	1. **Rendimiento Nativo (AOT):** Al compilar directamente a código máquina ARM (sin puentes de JavaScript), ofrece un rendimiento de 60fps estable, crítico para la fluidez percibida en aplicaciones financieras. 2. **Seguridad del Código:** El código compilado es binario, lo que dificulta significativamente la ingeniería inversa en comparación con soluciones híbridas basadas en webviews. Nota: Se evaluó React Native como segunda opción, pero se descartó porque Flutter compila a código nativo (ARM) ofreciendo mejor rendimiento en animaciones bancarias que el puente JS de React Native.
Backend	Microservicios (Spring Boot)	1. **Escalabilidad Granular:** Permite asignar más recursos (CPU/RAM) específicamente al servicio de "Transferencias" durante picos de fin de mes, sin desperdiciar recursos en módulos de bajo uso como "Configuración". 2. **Tolerancia a Fallos:** La arquitectura distribuida asegura que un error fatal en un servicio (ej. caída del servicio de notificaciones) no detenga la operación principal del banco (transferencias).

Persistencia Auditoría	MongoDB (NoSQL)	1. **Flexibilidad de Esquema (Schemaless):** Permite almacenar logs con estructuras heterogéneas (JSON) que pueden variar según el tipo de transacción, sin necesidad de realizar costosas migraciones de esquema como en SQL. 2. **Escritura Asíncrona Masiva:** Diseñada para soportar altas cargas de inserción (Write-Heavy), ideal para registrar cada paso del usuario sin latencia perceptible.
Integración Legacy	Patrón ACL (Anti-Corruption Layer)	1. **Aislamiento del Dominio:** Crea una barrera que traduce los modelos de datos anticuados del Core Bancario a modelos modernos, evitando que la deuda técnica del sistema legado "contamine" la nueva arquitectura. 2. **Desacoplamiento:** Si el banco decide cambiar su Core en el futuro, solo será necesario modificar la capa ACL, manteniendo intacta la lógica de los microservicios y frontends.
Caché	Redis	1. **Acceso de Baja Latencia:** Al operar completamente en memoria RAM, ofrece tiempos de respuesta inferiores al milisegundo, cumpliendo con el requisito de acceso inmediato para datos de "Clientes Frecuentes". 2. **Estructuras de Datos Avanzadas:** Permite gestionar sesiones de usuario y contadores de intentos fallidos (para seguridad) con operaciones atómicas nativas, más eficientes que en una base de datos relacional.
Notificaciones	Amazon SNS	**1.** **Desacoplamiento:** Implementa el patrón *Pub/Sub* para que las transacciones no se bloqueen esperando el envío de notificaciones. **2.** **Multicanal:** Permite el envío nativo de SMS (crítico para alertas bancarias) y la integración directa con otros servicios AWS.
	Amazon SES	**1.** **Entregabilidad:** Garantiza que los correos corporativos no caigan en SPAM mediante gestión de reputación y firmas DKIM/SPF. **2.** **Escalabilidad:** Soporta picos masivos de envío sin requerir administración de servidores de correo.
Monitoreo	CloudWatch	**1.** **Centralización de Logs:** En una arquitectura de microservicios (EKS) donde los contenedores son efímeros, es vital centralizar la salida de logs. CloudWatch recolecta todos los registros de aplicaciones y bases de datos en un solo lugar.. **2.** **Disparador de Auto-healing (Resiliencia):** CloudWatch al detectar métricas anómalas (como CPU alta o latencia excesiva), se dispara automáticamente las alarmas que activan el auto-escalado, permitiendo que el sistema se recupere y adapte a la demanda sin intervención humana.

Vista de Arquitectura (Modelo C4)

A continuación, se presentan los diagramas arquitectónicos generados bajo el estándar C4.

Nivel 1: Diagrama de Contexto

El siguiente diagrama ilustra las interacciones del Sistema de Banca BP con los actores y sistemas externos.

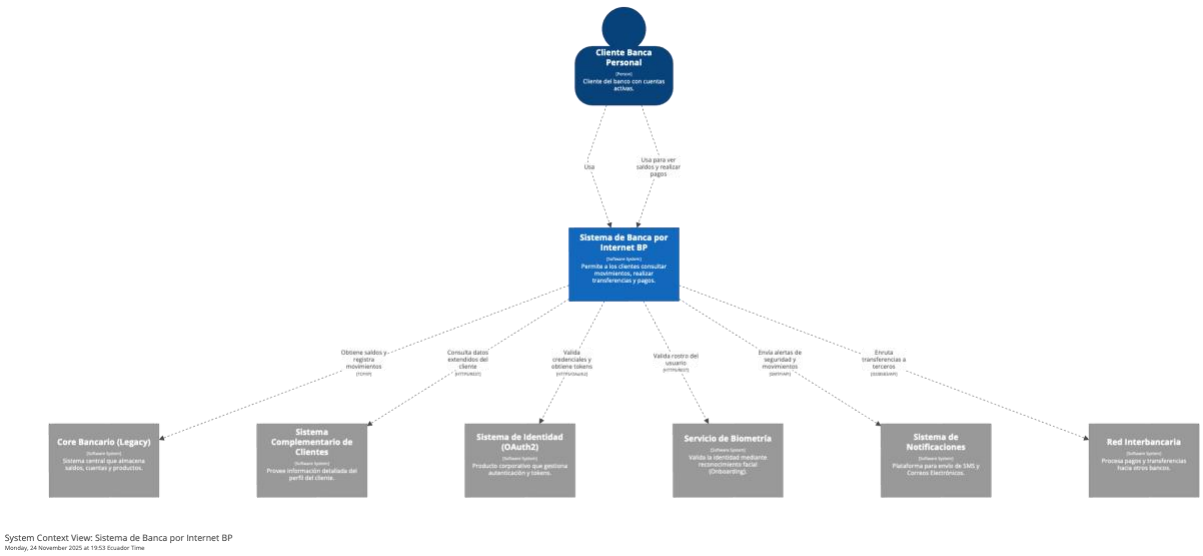


Figura 1: Diagrama de Contexto del Sistema Banca BP.

Descripción de Elementos:

- Sistema Banca BP: Núcleo de la solución.
- Core Bancario: Sistema legado proveedor de fondos y saldos.
- Sistemas Externos: Se integran servicios de Biometría (Seguridad), Notificaciones y conexión interbancaria.

Nivel 2: Diagrama de Contenedores

Este diagrama detalla la arquitectura técnica, mostrando las aplicaciones, servicios y almacenes de datos.

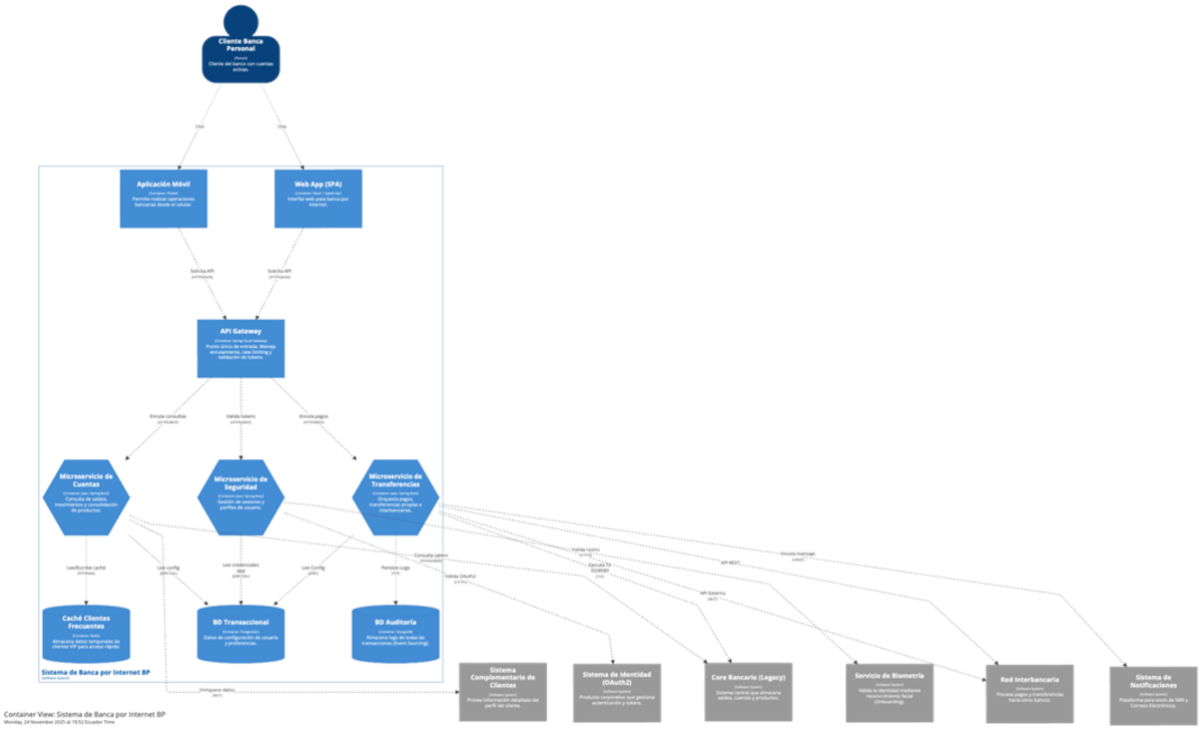


Figura 2: Diagrama de Contenedores y Tecnologías.

Análisis de la Solución:

- Frontend: Se propone una arquitectura híbrida con Flutter (Móvil) y React (Web) consumiendo una API unificada.
- API Gateway: Spring Cloud Gateway actúa como punto único de entrada, gestionando el enrutamiento y con la responsabilidad de seguridad (validación de tokens) de los microservicios.
- Persistencia: Se utiliza la base de datos adecuada para cada problema: PostgreSQL (Transaccional y datos de usuarios), Mongo (Auditoría) y Redis (Caché).

Nivel 3: Diagrama de Componentes

1. Componentes de Transferencias

Se realiza un acercamiento al microservicio más crítico (tranferencias) para demostrar la segmentación de responsabilidades.

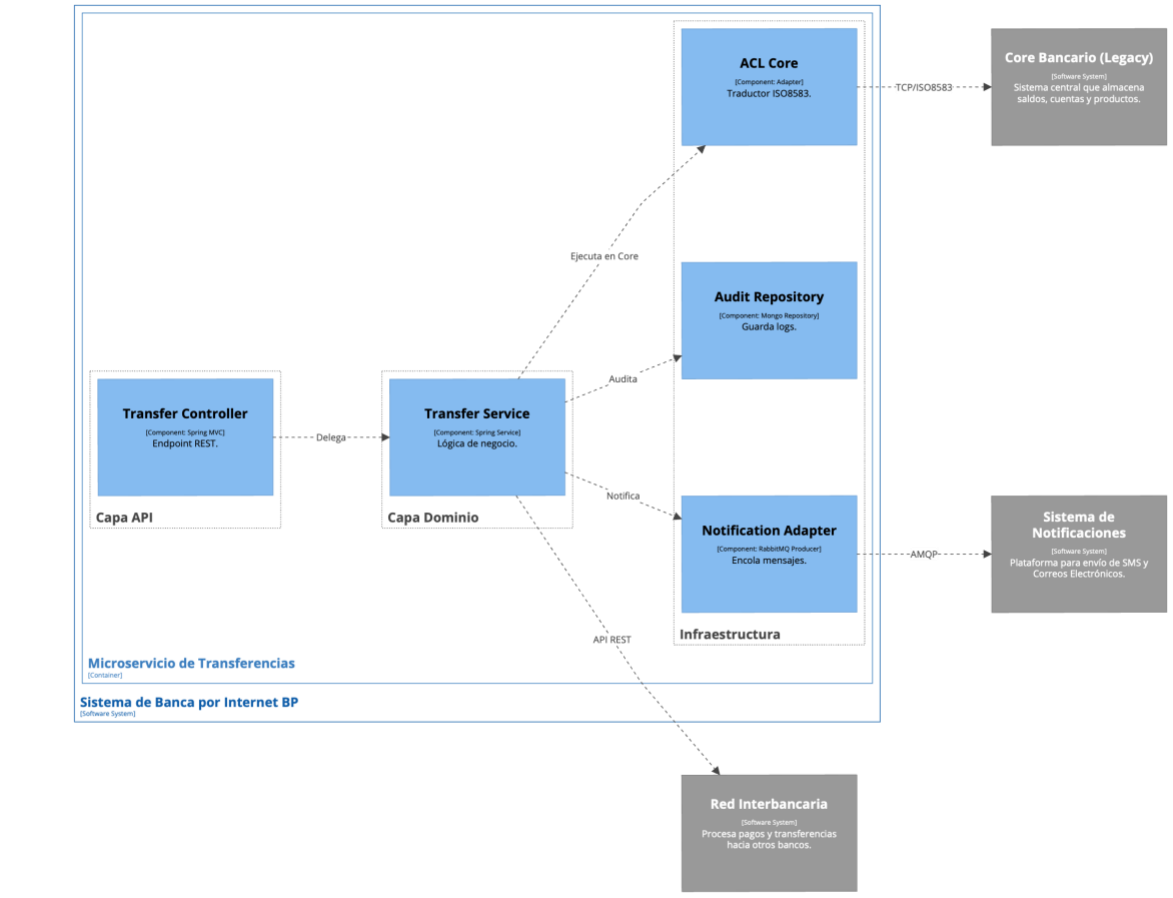


Figura 3: Diseño interno del Microservicio de Transferencias.

Patrones Aplicados:

- Separación de Intereses: El TransferController gestiona la petición HTTP, el TransferService aplica la lógica de negocio y los Repositories gestionan el acceso a datos.
- ACL (Anti-Corruption Layer): El componente ACL Service encapsula la complejidad de la comunicación con el Core Bancario (ISO8583), permitiendo que el resto del servicio trabaje con objetos limpios de Java.
- Comunicación Asíncrona: El NotificationAdapter utiliza colas de mensajería (RabbitMQ) para enviar correos sin bloquear la respuesta al usuario.

2. Componentes de Cuentas

Este componente es responsable de la consulta de saldos y movimientos. Dado que es la operación más frecuente del usuario, su diseño se centra en el rendimiento y la reducción de carga hacia el Core Bancario.

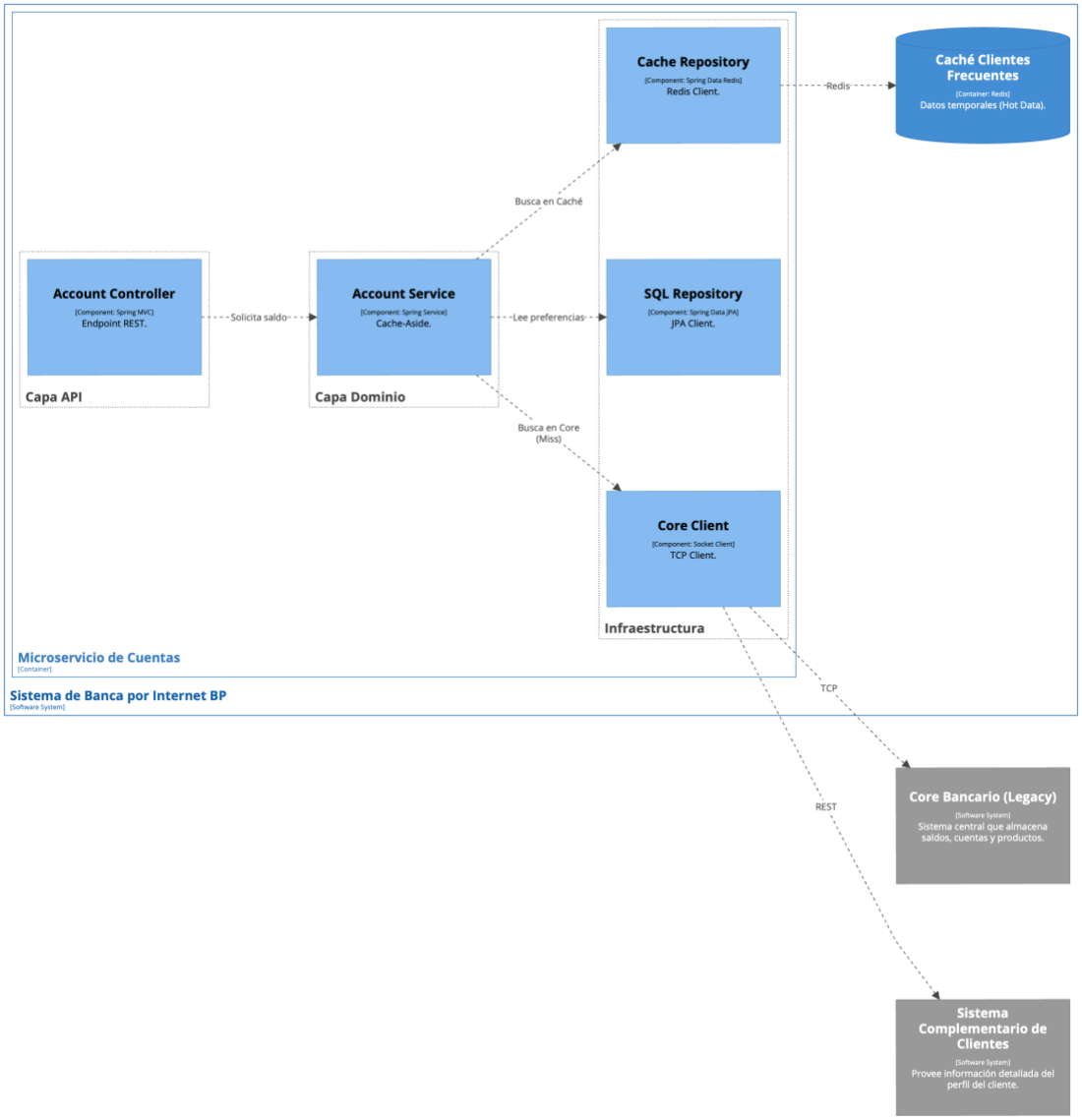


Figura 4: Diseño interno del Microservicio de Cuentas y estrategia de Caché.

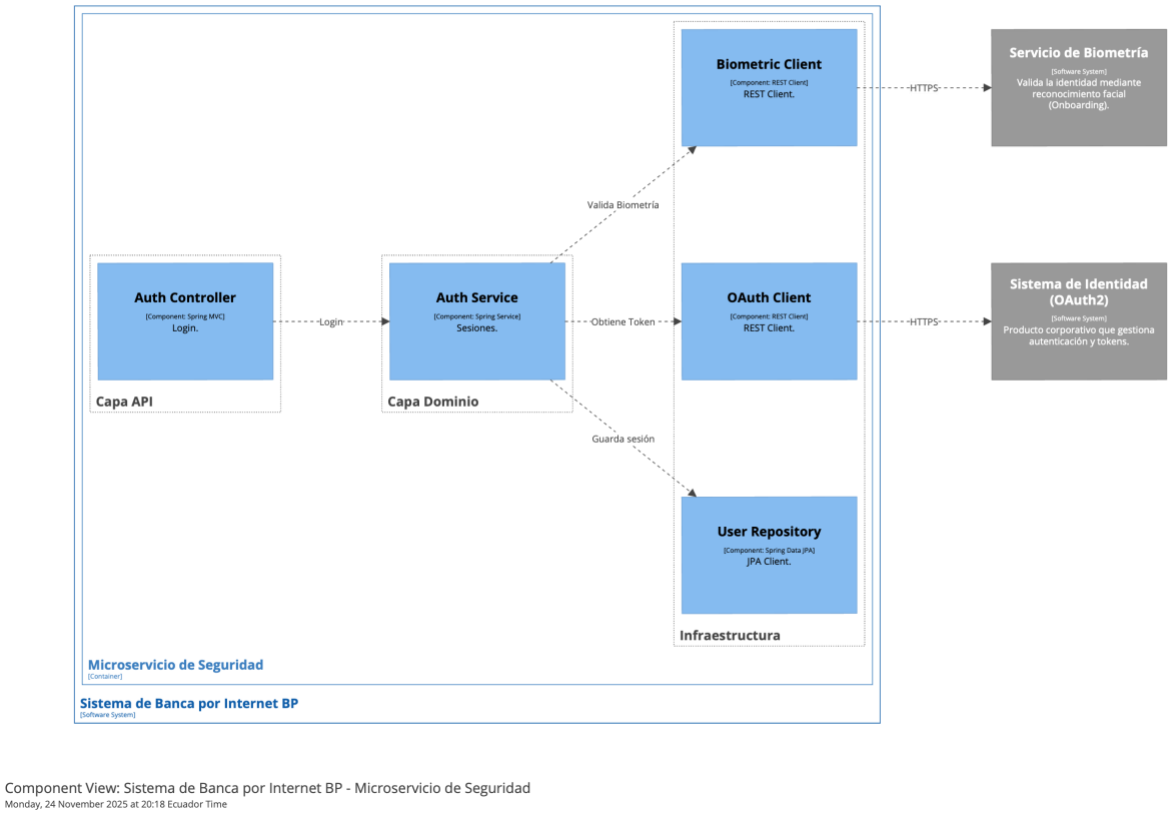
Patrones y Decisiones Aplicadas:

- Patrón Cache-Aside: El AccountService no consulta directamente al Core Bancario en primera instancia. Primero verifica la existencia del dato en el CacheRepository (Redis). Si el dato existe, se retorna en milisegundos. Si no existe, se consulta al Core a través del CoreAdapter y se actualiza la caché para futuras peticiones.

- Gestión de Datos Calientes vs. Fríos: Se utiliza Redis para almacenar datos volátiles y de acceso crítico (saldos), mientras que el SQLRepository se conecta a PostgreSQL para recuperar datos estáticos o de configuración (preferencias de visualización de la cuenta) que no requieren latencia ultra-baja.
- Adaptador de Protocolo: El componente CoreAdapter aísla la complejidad de establecer conexiones TCP/Socket con el mainframe legado, exponiendo una interfaz limpia al resto del dominio.

3. Componentes de Seguridad

Este microservicio orquesta el flujo de inicio de sesión y *onboarding*, integrando múltiples factores de autenticación para cumplir con la normativa de seguridad bancaria.



Component View: Sistema de Banca por Internet BP - Microservicio de Seguridad
Monday, 24 November 2025 at 20:18 Ecuador Time

Figura 5: Diseño interno del Microservicio de Seguridad e Integraciones.

Patrones y Decisiones Aplicadas:

- Orquestación de Servicios Externos: El AuthService actúa como un orquestador que coordina la validación de identidad. No implementa la criptografía del token ni el reconocimiento de imágenes, sino que delega estas responsabilidades a sistemas de OAuthSystem y Biometría a través de clientes HTTP OAuthClienty BiometricClient.

- Para OAuth2: El componente OAuthClient actúa como una fachada que simplifica el flujo de intercambio de credenciales y *refresh tokens* con el servidor de identidad corporativo. De esta manera se ocultando la complejidad del protocolo HTTP/REST.
- Auditoría de Acceso: A través del UserRepository, se registra la fecha y hora de cada intento de acceso exitoso o fallido en la base de datos local (PostgreSQL), permitiendo un rastreo de seguridad independiente de la actividad transaccional.

Recomendación de Flujo Estándar: Para la autenticación tanto en la Web (SPA) como en Móvil, es recomendable implementar el flujo Authorization Code Flow con PKCE (Proof Key for Code Exchange). A diferencia del flujo "Implicit" (que ya está obsoleto), PKCE protege el intercambio de tokens evitando que el código de autorización sea interceptado. De esta manera se cumplen con los estándares de seguridad financiera actuales (OAuth 2.1).

Diseño de infraestructura

Para soportar la arquitectura de microservicios propuesta y cumplir con los requisitos de Alta Disponibilidad (HA), Seguridad Bancaria y Tolerancia a Fallos, se ha diseñado una infraestructura en la nube sobre Amazon Web Services (AWS).

La solución implementa una estrategia Multi-Región (Activo-Pasivo) para garantizar la continuidad del negocio ante desastres geográficos, y una topología de red segregada para la máxima protección de los datos financieros.

Diagrama de Despliegue Global

La siguiente figura ilustra la topología de red, los componentes de cómputo y las estrategias de replicación de datos entre la región primaria (us-east-1) y la región de recuperación (us-west-1).

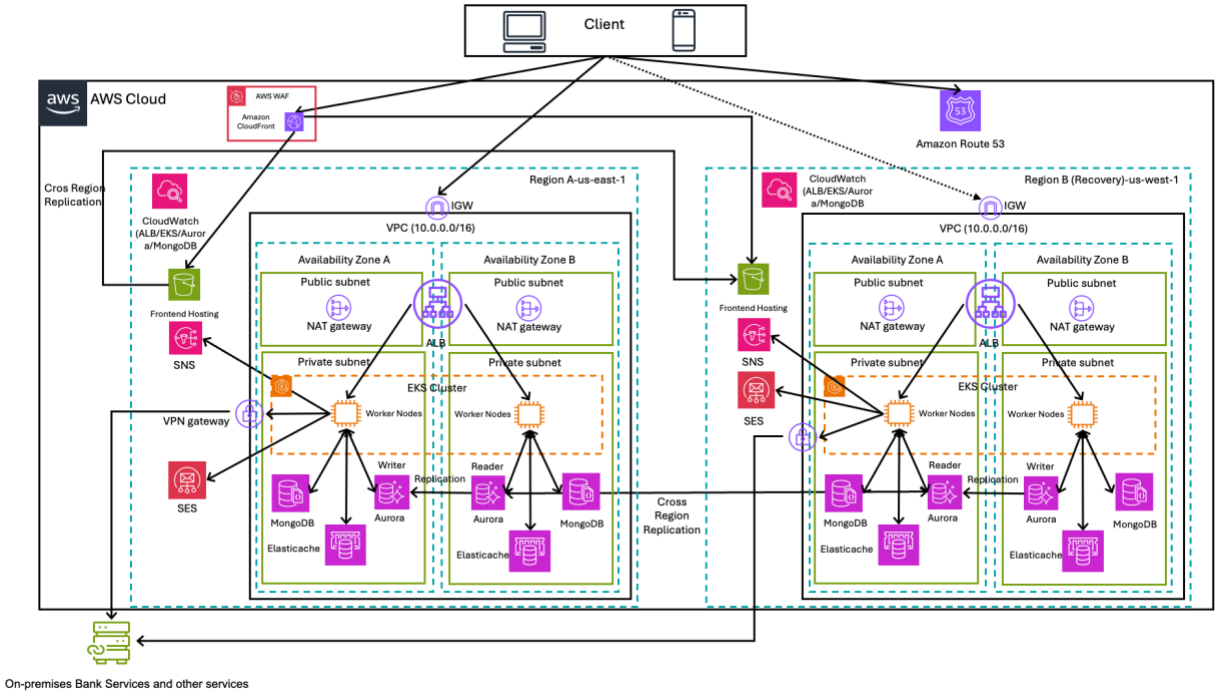


Figura 6: Arquitectura de Infraestructura en AWS con estrategia de Disaster Recovery.

Inventario de Recursos y Justificación Técnica

A continuación, se detalla cada componente de la infraestructura, su rol en el sistema y la justificación de su elección basada en los atributos de calidad del proyecto.

Recurso AWS	Rol en la Arquitectura	Justificación
Amazon Route 53	DNS y Gestión de Tráfico	Actúa como el punto de entrada global. Se eligió por su capacidad de Health Checks y Failover de DNS, lo que permite redirigir el tráfico automáticamente a la Región B si la Región A deja de responder.
AWS WAF	Firewall de Aplicaciones	Implementa la seguridad perimetral. Protege tanto al Frontend (en CloudFront) como al Backend (en el ALB) contra ataques comunes como SQL Injection, XSS y DDoS, cumpliendo con la normativa de seguridad.
Amazon CloudFront	CDN (Content Delivery Network)	Distribuye la aplicación Web (SPA React) desde puntos de presencia cercanos al usuario, garantizando una latencia mínima en la carga de la interfaz, independientemente de la ubicación del cliente.
Amazon S3	Alojamiento Estático	Almacena los archivos compilados del Frontend (HTML/JS/CSS). Es una solución Serverless que escala infinitamente y es mucho más económica y segura que mantener servidores web tradicionales (Nginx/Apache).
VPC & Subnets	Aislamiento de Red	Se utiliza una arquitectura de red segregada. Las subredes públicas solo contienen balanceadores y NATs, mientras que los datos y aplicaciones viven en subredes privadas sin acceso directo a internet, reduciendo la superficie de ataque.
Internet Gateway (IGW)	Puerta de Enlace	Permite la entrada de tráfico legítimo desde internet hacia los balanceadores públicos y la salida controlada de tráfico (vía NAT) para actualizaciones.
NAT Gateway	Salida Segura a Internet	Permite que los microservicios en la red privada descarguen parches o se conecten a APIs externas sin exponerse a recibir conexiones entrantes desde internet.
Application Load Balancer (ALB)	Balanceo de Carga	Distribuye el tráfico entrante entre los nodos de Kubernetes. Se encarga de la terminación SSL/TLS, descargando el trabajo de cifrado de los microservicios.
Amazon EKS	Orquestador de Contenedores	Servicio gestionado de Kubernetes. Permite desplegar, escalar y gestionar los microservicios (Transferencias, Cuentas, etc.) con alta disponibilidad, reiniciando contenedores fallidos automáticamente (Auto-healing).
Amazon Aurora (Global DB)	Base de Datos Relacional	Motor compatible con PostgreSQL. Se eligió la versión Global Database para replicar los datos transaccionales a la región secundaria en menos de 1 segundo, garantizando un RPO (Recovery Point Objective) cercano a cero.
Amazon DocumentDB	Base de Datos NoSQL	Motor compatible con MongoDB. Se utiliza para el almacenamiento masivo y rápido de los logs de auditoría, permitiendo escrituras de alta velocidad sin bloquear el sistema transaccional.

Amazon ElastiCache	Caché en Memoria	Motor de Redis. Almacena sesiones de usuario y datos de consulta frecuente. Su configuración Multi-AZ evita que una caída de zona desconecte a los usuarios activos.
VPN Gateway	Conectividad Híbrida	Establece un túnel seguro (IPsec) entre la nube de AWS y el Data Center físico del banco ("On-Premise"). Es crucial para que la Capa Anticorrupción (ACL) consuma los servicios del Core Bancario legado de forma segura.
Amazon SNS	Bus de Eventos y Notificaciones (Pub/Sub)	Permite que el servicio de Transferencias publique un evento único y que este se distribuya automáticamente a múltiples canales (SMS y Email) de forma asíncrona, sin bloquear la transacción financiera.
Amazon SES	Motor de Envío de Correos Transaccionales	Servicio de alta escala gestionado para el envío de comprobantes y alertas. Garantiza la entregabilidad de los correos (evitando la carpeta de SPAM) mediante gestión de reputación de IP y autenticación DKIM/SPF, vital para la confianza del cliente bancario.
Amazon CloudWatch	Observabilidad Centralizada y Alarmas	

Estrategia de Recuperación ante Desastres (DR)

Dado el requisito de presupuesto ilimitado y alta criticidad, se ha implementado una estrategia de "Active-Passive Hot Standby":

1. Región Primaria (us-east-1): Procesa el 100% del tráfico en condiciones normales.
2. Región de Recuperación (us-west-1): Mantiene una infraestructura idéntica desplegada.
3. Sincronización de Datos:
    - Aurora Global Database: Replica transacciones financieras en tiempo real.
    - S3 Cross-Region Replication (CRR): Mantiene sincronizados los archivos estáticos y documentos.
4. Escenario de Fallo: Si Route 53 detecta que la Región A está caída, actualiza los registros DNS para apuntar a la Región B. Los microservicios en la Región B (que ya están conectados a la réplica de lectura de la BD) promueven la base de datos a escritura y asumen la carga operativa en minutos.

Conectividad Híbrida

El sistema no vive aislado. La integración con el Core Bancario Legado se realiza a través de una conexión VPN Site-to-Site. Esto asegura que el tráfico sensible (consultas de saldo real, ejecución de transacciones en el mainframe) viaje encriptado y nunca transite por la internet pública, cumpliendo con los estándares.

Conclusiones

Tras el análisis de requisitos, el diseño de la arquitectura lógica (Modelo C4) y la definición de la infraestructura física en AWS, se presentan las siguientes conclusiones sobre la solución propuesta:

1. **Desacoplamiento Efectivo mediante Microservicios**: La adopción de una arquitectura de microservicios, orquestada mediante un API Gateway, permite segregar las responsabilidades del negocio. Esto no solo facilita el mantenimiento evolutivo, sino que permite escalar módulos críticos (como Transferencias) independientemente de módulos de bajo tráfico.
2. **Gestión de Deuda Técnica con Patrón ACL:** La implementación de una Capa Anticorrupción (ACL) para el diseño. Este componente actúa como un escudo protector que aísla al nuevo sistema digital de la complejidad y obsolescencia tecnológica que podría tener el Core Bancario. Gracias a esto, el equipo de desarrollo puede trabajar con tecnologías modernas (Java 17, Spring Boot) sin verse limitado por los protocolos legados.
3. **Experiencia de Usuario y Rendimiento:** La estrategia de rendimiento basada en dos pilares Flutter para la compilación nativa en móviles y Redis para el almacenamiento en caché de datos frecuentes garantiza tiempos de respuesta inferiores. Esto soluciona el problema de latencia inherente a las consultas directas contra sistemas bancarios antiguos.
4. **Resiliencia y Alta Disponibilidad:** El diseño de infraestructura en AWS bajo un esquema Multi-Región (Active-Passive) supera los requisitos estándar de disponibilidad. La replicación de datos en tiempo real mediante Aurora Global Database y la capacidad de desvío de tráfico global con Route 53 aseguran que el banco pueda resistir fallos catastróficos regionales sin pérdida de datos transaccionales (RPO cercano a cero).
5. **Seguridad en Profundidad:** La solución no depende de un solo perímetro. La seguridad se ha integrado en múltiples capas: desde el borde (WAF y CloudFront), pasando por la identidad (OAuth2 y Biometría), hasta la infraestructura de red (Subredes Privadas y VPN). Esto garantiza el cumplimiento de las normativas de protección de datos y reduce significativamente la superficie de ataque.

La arquitectura propuesta proporciona a BP Bank una plataforma tecnológica segura y preparada para el futuro, capaz de soportar el crecimiento transaccional en próximos años y facilitar la incorporación de nuevas funcionalidades financieras.