Desarrollo de Ejercicio Práctico

Ing. Jeremy Joel Cadena Díaz

Tabla de contenido

Documento de Diseño de Arquitectura: Sistema de Banca por Internet

Resumen Ejecutivo

Este documento presenta una propuesta de arquitectura de soluciones para el nuevo Sistema de Banca por Internet de BP. La solución está diseñada para ser moderna, segura, escalable y resiliente, cumpliendo con los exigentes requisitos del sector financiero. La arquitectura se basa en un enfoque de microservicios desacoplados, desplegados en una infraestructura en la nube (AWS), y documentada visualmente a través del modelo C4 para garantizar la claridad a todos los niveles de la organización. El diseño prioriza una experiencia de usuario fluida, una seguridad de extremo a extremo y la excelencia operativa, abordando todos los requisitos funcionales y no funcionales del ejercicio.

Impulsores Arquitectónicos y Requisitos Clave

La arquitectura propuesta está guiada por los siguientes requisitos funcionales y de calidad:

Requisitos Funcionales:

- Consulta de histórico de movimientos.
- Realización de transferencias a cuentas propias e interbancarias.
- Notificación multicanal obligatoria de todas las transacciones.
- Acceso a través de una aplicación web (SPA) y una aplicación móvil multiplataforma.
- Onboarding de nuevos clientes 100% digital a través de la app móvil, con verificación de identidad por reconocimiento facial.
- Integración con dos sistemas legados para la obtención de datos de clientes (Core Bancario y Sistema Complementario).

Atributos de Calidad (Requisitos NO Funcionales)

- *Alta Disponibilidad (HA) y Tolerancia a Fallos:* El sistema debe permanecer operativo ante fallos de componentes.
- *Recuperación ante Desastres (DR):* Capacidad de recuperar el servicio en una región geográfica diferente en caso de un desastre mayor.
- *Seguridad:* Cumplimiento de los más altos estándares de seguridad bancaria en todas las capas.
- *Escalabilidad y Rendimiento:* Capacidad para manejar picos de demanda y garantizar baja latencia.
- *Monitoreo y Excelencia Operativa:* Visibilidad completa del estado del sistema y capacidad de auto-reparación.
- *Desacoplamiento y Reusabilidad:* Arquitectura modular que facilite futuras evoluciones y la reutilización de componentes.

Arquitectura Propuesta y Justificación

Arquitectura de Aplicación Front-End y Móvil

Para ofrecer una experiencia de usuario moderna y consistente, se proponen las siguientes tecnologías:

- **Aplicación Web (SPA):** Se recomienda el uso de **React.js**.
    - **Justificación:** Su ecosistema maduro, su enfoque basado en componentes reutilizables y la amplia disponibilidad lo hacen ideal para construir una interfaz de usuario compleja y mantenible.
- **Aplicación Móvil Multiplataforma:** Se recomienda **React Native**.
    - **Justificación:** Permite una sinergia total con la aplicación web, compartiendo lógica de negocio (estado, validaciones) y permitiendo que un mismo equipo de desarrollo trabaje en ambas plataformas. Ofrece un rendimiento cercano al nativo y acceso completo a las capacidades del dispositivo (cámara para onboarding, biometría para login).
    - **Alternativa: Flutter** es una excelente opción secundaria, que ofrece un rendimiento nativo superior y una consistencia de UI perfecta entre iOS y Android, ideal para una fuerte identidad de marca.

Arquitectura de Autenticación e Integración con Onboarding

- **Flujo de Autenticación (OAuth 2.0):**
    - **Recomendación:** Se debe configurar el producto de la compañía para que implemente el flujo **Authorization Code Flow con PKCE (Proof Key for Code Exchange)**.
    - **Justificación:** Este es el estándar de seguridad actual para aplicaciones públicas (SPA y móviles), ya que previene ataques de intercepción del código de autorización, garantizando que solo la aplicación que inició el proceso pueda obtener el token de acceso.

- **Integración con Onboarding:**

    1. El flujo de onboarding en la app móvil utilizará una herramienta especializada en KYC (Know Your Customer) como **Onfido**, **Jumio** o una combinación de **AWS Rekognition** con servicios de validación de documentos.
    2. El Onboarding Service orquestará este proceso, enviando las imágenes y videos al proveedor externo para su validación.
    3. Una vez verificado, se creará el usuario en el sistema, permitiéndole establecer sus credenciales y acceder posteriormente mediante contraseña, huella dactilar o reconocimiento facial del dispositivo.

Arquitectura de Acceso a Datos y Backend (Microservicios)

Se propone una arquitectura de microservicios para lograr desacoplamiento y escalabilidad.

- **Patrón General: API Gateway + Microservices**.
    - **API Gateway (AWS API Gateway):** Actúa como punto de entrada único, centralizando la seguridad (validación de JWT), el enrutamiento, el control de tráfico y el versionado de APIs.
    - **Microservicios:** Se han identificado los siguientes servicios para una correcta segmentación de responsabilidades:

        - User Service: Gestiona perfiles, credenciales y preferencias.
        - Accounts Service: Expone los datos de cuentas y movimientos, actuando como capa anti-corrupción frente a los sistemas legados (Core y Complementario).
        - Transfers Service: Orquesta la lógica de transferencias monetarias.
        - Onboarding Service: Maneja el flujo de registro de nuevos clientes.
        - Notifications Service: Se encarga exclusivamente de la comunicación con los proveedores de notificaciones.

- **Arquitectura de Datos:**
    - **Base de Datos Principal (PostgreSQL en AWS RDS):** Almacena los datos propios del dominio de la aplicación (perfiles, estado de transacciones).
    - **Caché (Redis en AWS ElastiCache):** Almacena datos de consulta frecuente (ej. últimos movimientos) para reducir la carga sobre los sistemas legados y mejorar la latencia.

Diseño de Solución de Auditoría

Para cumplir con el requisito de auditoría de manera robusta y escalable, se propone el patrón **Event Sourcing**.

- **Justificación:** En lugar de una base de datos de auditoría tradicional que puede convertirse en un cuello de botella, este patrón garantiza un registro inmutable y completo de cada acción del sistema en forma de eventos.
- **Implementación:**
    1. Los microservicios publican eventos de negocio (TransferenciaIniciada, LoginExitoso, etc.) en un bus de eventos.
    2. **Bus de Eventos (Apache Kafka en AWS MSK):** Actúa como un buffer duradero y desacoplado para los eventos.
    3. Un servicio consumidor procesa estos eventos y los persiste en un **Almacén de Auditoría (Elasticsearch)**, optimizado para búsquedas y análisis.

Integración con Servicios Externos

La arquitectura está diseñada para integrarse de forma segura y resiliente con sistemas externos a través de clientes dedicados dentro de cada microservicio, utilizando patrones como el **Circuit Breaker** para evitar fallos en cascada.

- Onboarding Service -> Servicio de Verificación de Identidad.
- Transfers Service -> Pasarela de Pagos Interbancarios.
- Notifications Service -> Proveedores de Notificaciones (Email, SMS, etc.).
- Accounts Service -> Plataforma Core Bancaria y Sistema de Información Complementario.

Implementación de Requisitos No Funcionales

Alta Disponibilidad (HA), Tolerancia a Fallos y Recuperación ante Desastres (DR)

- **HA y Tolerancia a Fallos:** La solución se desplegará en **múltiples Zonas de Disponibilidad (Multi-AZ)** en AWS. Los microservicios se ejecutarán en contenedores orquestados por **Amazon EKS (Kubernetes)** con auto-escalado y auto-reparación (auto-healing). La base de datos RDS y el clúster de Kafka se configurarán en modo Multi-AZ para conmutación por error automática.
- **DR:** Se realizarán copias de seguridad automáticas y periódicas de la base de datos en una región de AWS geográficamente distante. La infraestructura, definida como código (IaC) con Terraform, puede ser desplegada rápidamente en la región de DR si es necesario.

Conocimientos de Nube (AWS), Seguridad y Manejo de Costos

- **Estrategia Cloud:** Se elige **AWS** por la madurez de sus servicios gestionados, que se alinean perfectamente con la arquitectura propuesta (EKS, RDS, MSK, API Gateway, ElastiCache), reduciendo la carga operativa.
- **Seguridad:** Se implementa un enfoque de defensa en profundidad:
    - **Perímetro: AWS WAF** en el API Gateway para protección contra ataques comunes.
    - **Tránsito:** Cifrado TLS 1.2+ en todas las comunicaciones. **mTLS** para la comunicación entre servicios.
    - **Reposo:** Cifrado de datos en todas las bases de datos y volúmenes de almacenamiento (AWS KMS).
    - **Credenciales:** Uso de **AWS Secrets Manager** para gestionar de forma segura las credenciales de la aplicación.
- **Manejo de Costos:** Se utilizarán instancias reservadas para cargas de trabajo predecibles (Base de Datos, Kafka) y auto-escalado para los microservicios para pagar solo por el cómputo necesario. Se implementará un monitoreo de costos con AWS Budgets.

Implementación de Monitoreo

Se propone una pila de observabilidad completa:

- **Logs:** Centralización de logs de todos los contenedores en Amazon OpenSearch (basado en Elasticsearch).
- **Métricas:** Recolección de métricas de rendimiento con Prometheus y visualización en Grafana.
- **Trazabilidad:** Implementación de trazabilidad distribuida con Jaeger u OpenTelemetry para seguir las peticiones a través de los microservicios, facilitando la depuración.

Conocimientos de Regulaciones Bancarias y Estándares de Seguridad

La arquitectura considera desde su diseño normativas clave del sector financiero:

- **Ley de Protección de Datos Personales (similar a GDPR):** El cifrado de datos en reposo y en tránsito, y una clara segmentación de responsabilidades, ayudan a garantizar la privacidad y seguridad de los datos del cliente.
- **PCI DSS:** Aunque no se almacenen datos de tarjetas, al interactuar con pasarelas de pago, la arquitectura sigue los principios de seguridad de red y control de acceso de este estándar para proteger el entorno de transacciones.

Diagramas de Arquitectura (Modelo C4)

A continuación, se presentan los diagramas que describen la solución en diferentes niveles de abstracción.

Diagrama de Contexto (Nivel C1)

Este diagrama ofrece una vista de alto nivel, mostrando el sistema en su totalidad y cómo interactúa con los usuarios y otros sistemas.

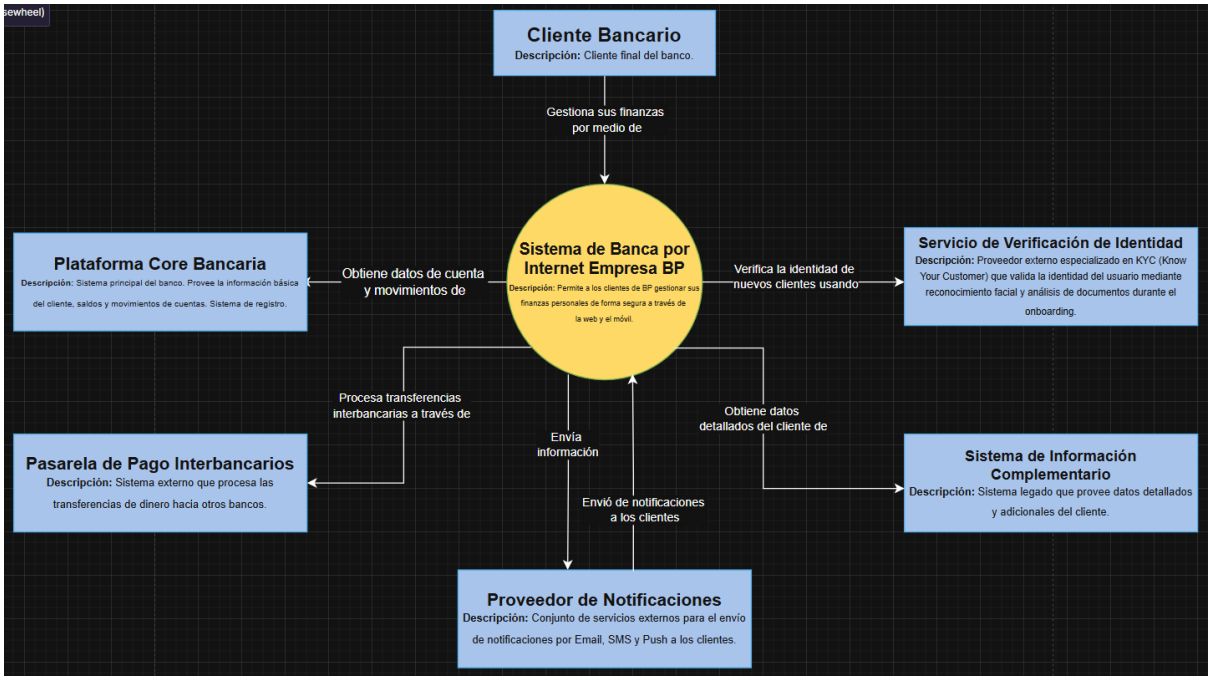


Diagrama de Contenedores (Nivel C2)

Este diagrama descompone el sistema en sus principales contenedores (aplicaciones, servicios, bases de datos), mostrando las responsabilidades y tecnologías de cada uno.

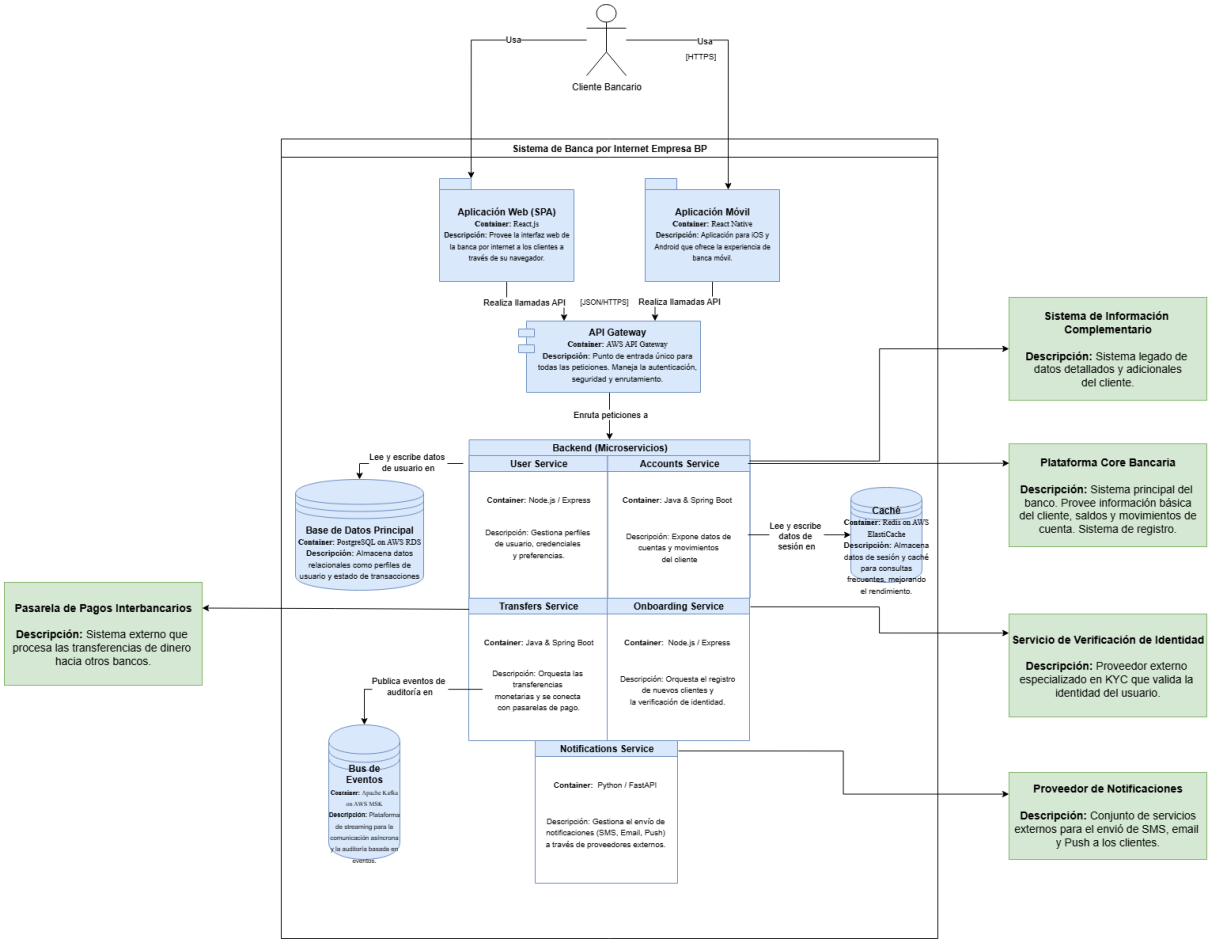
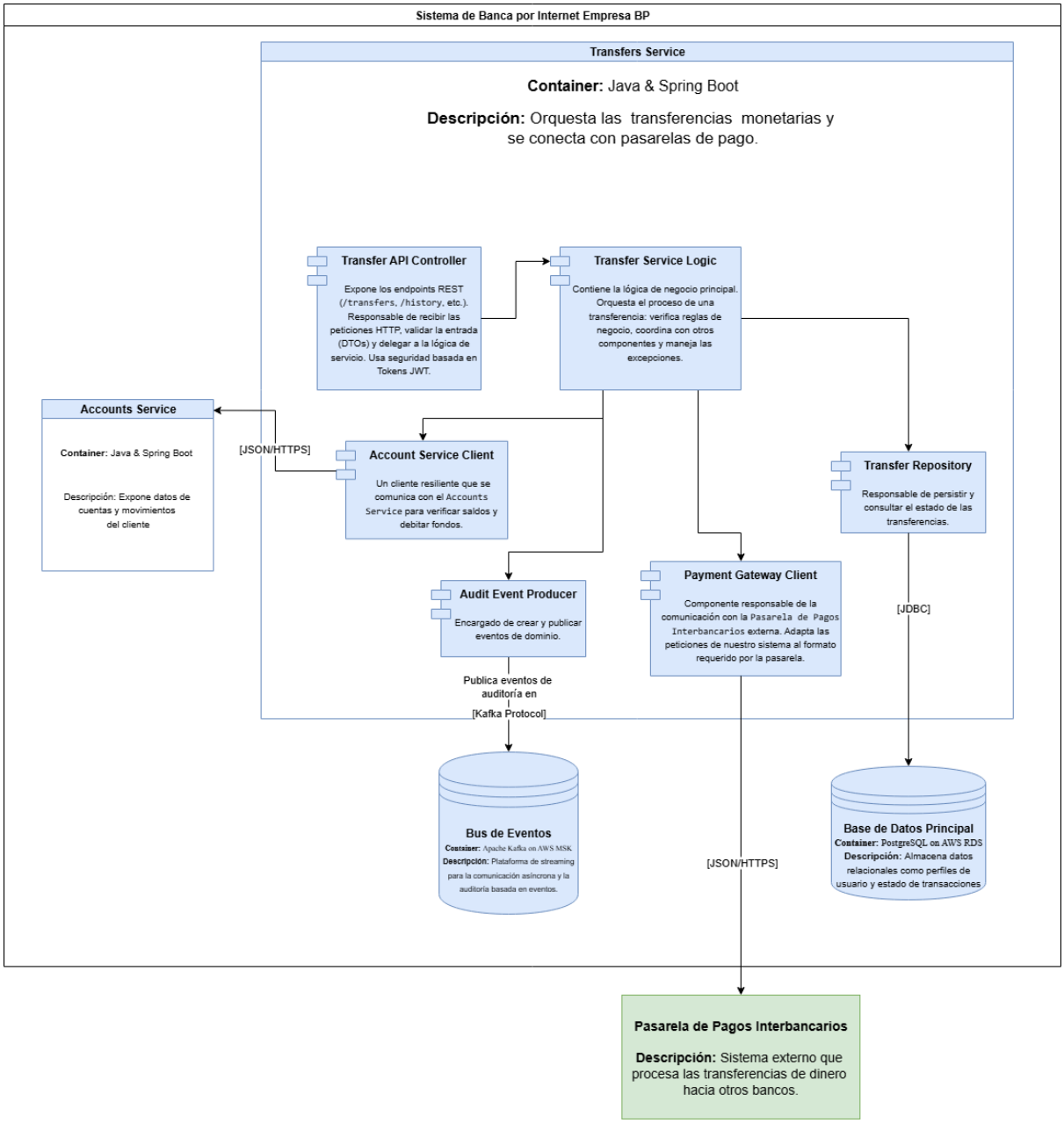


Diagrama de Componentes (Nivel C3)

Este diagrama hace zoom en el Transfers Service para mostrar sus componentes internos y cómo colaboran para cumplir con sus responsabilidades.



Conclusión

La arquitectura de microservicios propuesta, desplegada en AWS y diseñada siguiendo los principios de seguridad, resiliencia y escalabilidad, cumple de manera integral con todos los requisitos del Sistema de Banca por Internet de BP. La solución no solo resuelve las necesidades de negocio actuales, sino que también proporciona una base sólida y flexible para la evolución futura de la plataforma.