ARQUITECTURA DE SOLUCIONES

EJERCICIO PRÁCTICO DE ARQUITECTURA DE SOLUCIONES

- Cree un documento PDF con la respuesta al ejercicio,
    - El documento debe estar bien organizado y ser fácil de leer.
    - Agregue imágenes para explicar los diagramas. Puedes usar herramientas como http://draw.io/ para generar las imágenes.
    - Cada uno de los diagramas de C4 e un entregable importante. Asegúrese de hacerlos correctos y detallados.
    - Preste también atención a los otros diagramas requeridos.
    - Asegúrese de añadir cualquier texto explicativo que considere necesario.
- Suba el documento como respuesta a este ejercicio. Asegúrese de subirlo dentro del tiempo de resolución
- Adicional, crear un repositorio público en GitHub y subir el PDF a ese repositorio. Colocar la URL al repositorio en los comentarios de este ejercicio.

Recomendaciones MUY valoradas:

Cada decisión arquitectónica en el diseño debe tener una justificación teórica (máximo 2 por cada una) ejemplo: ¿Por qué se decidió por determinados patrones, tecnologías, componentes, protocolos, etc.? ¿Qué opciones evaluó?

Se debe incluir los siguientes diagramas en el diseño de la solución:

- **Diagrama de contexto:** para usuarios no técnicos. Muestra sistemas internos y externos, actores clave, breves descripciones y conexiones explicativas.
- **Diagrama de contenedores:** para técnicos, representa aplicaciones, servicios, bases de datos y mensajería, incluyendo componentes cloud sin mucho detalle. Conexiones con descripciones breves.
- **Diagrama de componentes:** mayor detalle técnico, incluye microservicios, patrones arquitectónicos y protocolos de comunicación con seguridad. Destaca el uso de componentes cloud.

Descripción del ejercicio:

Usted ha sido contratado por una entidad llamada BP como arquitecto de soluciones para diseñar un sistema de banca por internet, en este sistema los usuarios podrán acceder al histórico de sus movimientos, realizar transferencias y pagos entre cuentas propias e interbancarias.

Toda la información referente al cliente se tomara de 2 sistemas, una plataforma Core que contiene información básica de cliente, movimientos, productos y un sistema independiente que complementa la información del cliente cuando los datos se requieren en detalle.

Debido a que la norma exige que los usuarios sean notificados sobre los movimientos realizados, el sistema utilizara sistemas externos o propios de envío de notificaciones, mínimo 2.

Este sistema contara con 2 aplicaciones en el front, una spa y una aplicación móvil desarrollada en un framework multiplataforma (mencione 2 opciones y justifique el porqué de su elección).

Ambas aplicaciones autenticaran a los usuarios mediante un servicio que usa el estándar OAuth 2.0, para el cual no requiere implementar toda la lógica, ya que la compañía cuenta con un producto que puede ser configurado para este fin, sin embargo, debe dar recomendaciones sobre cuál es el mejor flujo de autenticación que debería usar el estándar.

ARQUITECTURA DE SOLUCIONES

Tenga en cuenta que el sistema de onboarding para nuevos clientes en la aplicación móvil usa reconocimiento facial, por tanto, su arquitectura deberá considerarlo como parte del flujo de autorización y autenticación, a partir del onboarding el nuevo usuario podrá ingresar al sistema mediante usuario y clave, huella o algún otro método especifique alguno de los anteriores dentro de su arquitectura, también puede recomendar herramientas de industria que realicen estas tareas y robustezca su aplicación.

El sistema utiliza una base de datos de auditoría que registra todas las acciones del cliente y cuenta con un mecanismo de persistencia de información para clientes frecuentes, para este caso proponga una alternativa basada en patrones de diseño que relaciones los componentes que deberían interactuara para conseguir el objetivo.

Para obtener los datos del cliente el sistema pasa por una capa de integración compuesta por un api Gateway y consume los servicios necesarios de acuerdo con el tipo de transacción, inicialmente usted cuenta con 3 servicios principales , consulta de datos básicos, consulta de movimientos y transferencias que realiza llamados a servicios externos dependiendo del tipo, si considera que debería agregar más servicios para mejorar el rendimiento de su arquitectura o agregar más servicios para mejorar la respuesta de información a sus clientes, es libre de hacerlo.

Consideraciones

Para este reto, mencione aquellos elementos normativos que considere importantes para tener en cuenta para entidades financieras, ejemplo ley de datos personales, seguridad, etc.

Garantice en su arquitectura , alta disponibilidad (HA), tolerancia a fallos, recuperación ante desastres(DR), seguridad y monitoreo, excelencia operativa y auto-healing.

Si lo considera necesario, su arquitectura puede contener elementos de infraestructura de nube, azure o aws, garantice baja latencia, cuenta con presupuesto para esto.

En lo posible plantee una arquitectura desacoplada con elementos y reusables y cohesionados para otros componentes que puedan adicionarse en el futuro.

El modelo debe ser desarrollado bajo modelo c4 (modelo de contexto, modelo de aplicación o contenedor y componentes), describa el modelo de componentes, la infraestructura la puede modelar como usted lo considere usando la herramienta de su preferencia.

Se calificaran los siguientes elementos:

- Que la solución satisfaga los requerimientos y todo cuente con justificación.
- Calidad y profundidad de los diagramas (Contexto, contenedores y componentes)
- Segmentación de responsabilidades y desacoplamiento
- Uso de patrones de arquitectura
- Integración con servicios externos
- Calidad de arquitectura de aplicación front-end y móvil
- Arquitectura de acceso a datos
- Conocimientos de nube (AWS o Azure)
- Manejo de costos
- Arquitectura de autenticación

ARQUITECTURA DE SOLUCIONES

- Arquitectura de integración con onboarding
- Diseño de solución de auditoria
- Conocimiento de regulaciones bancarias y estándares de seguridad
- Implementación de alta disponibilidad y tolerancia a fallos
- Implementación de monitoreo.

Éxitos

ARQUITECTURA DE SOLUCIONES

RESUMEN EJECUTIVO

Objetivo del documento.

Diseñar una arquitectura de soluciones de un sistema de banca por internet, para la entidad bancaria BP, garantizando que las consultas y operaciones bancarias se puedan realizar desde los canales web/móvil, integrando de forma segura con los sistemas de autenticación actuales, realizar la verificación de identidad mediante biometría, registro de auditoría, y la plataforma de integración disponible. Garantizando en su arquitectura, alta disponibilidad (HA), tolerancia a fallos, recuperación ante desastres(DR), seguridad y monitoreo, excelencia operativa y auto-healing.

Alcance del proyecto.

Diseñar una arquitectura de soluciones de un sistema de banca por internet, que permita:

1. Acceder al histórico de los movimientos de los clientes,
2. Realizar transferencias y pagos entre cuentas propias e interbancarias.
3. Acceder a la información de los clientes a 2 sistemas Core e Información de cliente.
4. Notificar a los clientes sobre los movimientos realizados, mediante 2 sistemas externos o propios
5. Debe contar con 2 aplicaciones Front web (SPA) y Móvil desarrollada en un framework multiplataforma
6. Autenticar con el sistema actual que se basa en el estándar OAuth2.0 (recomendar flujo de autenticación más adecuado)
7. La aplicación móvil debe contar con un sistema de onboarding para clientes nuevos
    a. Debe usar reconocimiento facial
    b. Debe considerar el reconocimiento facial como parte del flujo de autorización y autenticación.
8. El usuario debe poder acceder mediante usuario y clave, huella o algún otro método, recomendar aplicaciones de terceros que se puedan usar para este fin
9. El sistema debe registrar auditoria de todas las acciones del cliente y contar con un mecanismo de persistencia de información para clientes frecuentes.
10. Los servicios principales de consulta de datos de cliente, de movimientos y de transferencias deben pasar por una capa de integración expuesta por un apigateway.
11. El servicio de transferencias llama a servicios externos para las operaciones interbancarias.

Beneficios esperados.

- Cumplimiento normativo financiero

  Hay que asegurar que la aplicación cumpla con los requisitos regulatorios, normativos y de cumplimiento aplicables al sector financiero, garantizando trazabilidad, auditoría, confidencialidad y controles operativos.

- Alta disponibilidad (HA) y continuidad operativa

  Diseñar y desplegar la solución con mecanismos que aseguren disponibilidad continua, minimizando tiempos de inactividad y permitiendo operación ininterrumpida ante fallos parciales.

- Tolerancia a fallos y auto-recuperación (auto-healing)

Implementar patrones y capacidades que permitan detectar fallos, aislarlos y recuperarse automáticamente sin afectar la experiencia del usuario ni la integridad del servicio.

- Recuperación ante desastres (DR)

  Garantizar estrategias de respaldo, replicación y recuperación que permitan restaurar la operación en escenarios de desastre, cumpliendo con los RTO/RPO definidos para entidades financieras.

- Seguridad y monitoreo integral

  Incorporar controles de seguridad, observabilidad y monitoreo continuo que permitan detectar anomalías, prevenir incidentes y asegurar la integridad, disponibilidad y confidencialidad de la información.

- Excelencia operativa

  Asegurar prácticas de operación eficientes, automatizadas y auditables que permitan mantener la calidad del servicio, reducir errores humanos y facilitar la gestión del ciclo de vida de la aplicación.

- Despliegue en nube

  Implementar la solución en una plataforma cloud que permita escalabilidad, elasticidad, automatización y cumplimiento de los estándares de resiliencia requeridos por el sector financiero.

- Garantizar baja latencia

  Optimizar la arquitectura, los componentes y la infraestructura para asegurar tiempos de respuesta mínimos, adecuados a los niveles de servicio exigidos por aplicaciones financieras críticas

CONTEXTO Y ALCANCE

Situación actual

Se requiere diseñar un sistema web de banca por internet que permita a los usuarios consultar el historial de sus movimientos, realizar transferencias y efectuar pagos entre cuentas propias e interbancarias. La solución debe integrarse con los mecanismos actuales de autenticación basados en OAuth, con los dos servicios de notificaciones contratados y con los sistemas biométricos utilizados para la verificación de identidad. Además, deberá aprovechar la base de datos de auditoría existente para el registro de actividades y trazabilidad operativa.

Necesidades actuales.

Necesidades funcionales y de integración

La organización requiere una plataforma de banca por internet que permita a los usuarios consultar el histórico de movimientos, realizar transferencias y pagos entre cuentas propias e interbancarias, y acceder a información proveniente de dos sistemas Core. La solución debe integrarse con los mecanismos actuales de autenticación basados en OAuth 2.0, incorporar un flujo de onboarding móvil con reconocimiento facial y soportar múltiples métodos de acceso, incluyendo usuario/clave, biometría y soluciones de terceros. Además, el sistema debe interactuar con dos servicios de notificaciones externos o propios, registrar auditoría completa de

las acciones del cliente y contar con un mecanismo de persistencia especializado para clientes frecuentes.

Necesidades técnicas, de arquitectura y operación

La plataforma debe incluir dos aplicaciones como front, web (SPA) y móvil multiplataforma, y exponer sus servicios principales: (1) consultas de cliente, (2) consulta de movimientos y (3) transferencias. Utilizando una capa de integración gestionada por un API Gateway. El sistema de transferencias debe conectarse con servicios externos para operaciones interbancarias, garantizando seguridad, trazabilidad y consistencia. De igual manera, se requiere una arquitectura capaz de operar de forma confiable en un entorno distribuido, asegurando baja latencia, disponibilidad continua y cumplimiento de los estándares operativos y regulatorios propios del sector financiero.

Consideraciones regulatorias.

El sistema debe cumplir con los estándares de seguridad, privacidad, auditoría, continuidad operativa y prevención de fraude.

Esto incluye autenticación robusta, trazabilidad completa, protección de datos personales, controles AML/KYC, resiliencia ante fallos, monitoreo continuo, integración segura con sistemas internos y externos, y cumplimiento de normativas locales e internacionales aplicables.

1. Seguridad de la información (ISO 27001 / OWASP / Zero Trust)
    - Gestión de identidades y accesos (IAM) con 2FA o MFA.
    - Autenticación y autorización robusta (OAuth2.0 / OIDC).
    - Encriptación en tránsito (TLS 1.2+) y en reposo (AES-256).
    - Gestión segura de secretos.
    - Políticas de mínimo privilegio.
    - Hardening de servidores, contenedores y pipelines.
    - Protección contra ataques aplicando en el desarrollo OWASP Top 10 (XSS, CSRF, SQLi, SSRF, etc.).
2. Auditoría, trazabilidad y no repudio
    - Registro detallado de todas las acciones del cliente y del sistema.
    - Logs inmutables y firmados digitalmente.
    - Correlación de eventos.
    - Conservación de logs.
    - Evidencias para investigaciones, fraudes y conciliaciones.
3. Cumplimiento normativo financiero
    - Prevención de Lavado de Activos (AML)
        - Validación de identidad (KYC).
        - Monitoreo de transacciones sospechosas.
        - Límites transaccionales configurables.
        - Reportes automáticos a organismos reguladores.
    - Conozca a su Cliente (KYC)
        - Verificación biométrica.
        - Validación documental.
        - Pruebas de vida (liveness detection).
        - Actualización periódica de datos.
    - Protección de datos personales
        - Consentimiento explícito.
        - Minimización de datos.
        - Derecho al olvido.

  - Clasificación y gobierno de datos.
  - Transferencia segura de información.
4. Continuidad del negocio y resiliencia (BCP/DRP)
   - Alta disponibilidad (HA).
   - Recuperación ante desastres (DR) con RTO/RPO definidos.
   - Pruebas periódicas de failover.
   - Arquitectura tolerante a fallos (CB, retry, timeout, fallback).
   - Auto-healing y escalamiento automático.
5. Gestión de riesgos tecnológicos.
   - Evaluación de riesgos operativos y cibernéticos.
   - Controles compensatorios.
   - Monitoreo continuo de amenazas.
   - Gestión de vulnerabilidades y parches.
   - Segregación de ambientes (DESARROLLO / PRUEBAS O CERTIFICACIÓN / PRODUCCIÓN).
6. Integración segura con terceros (APIs, Core bancario, biometría, notificaciones)
   - API Gateway con políticas de seguridad.
   - Validación estricta de contratos (schema validation).
   - Rate limiting y throttling.
   - Auditoría de integraciones.
   - Gestión de certificados y rotación automática.
7. Cumplimiento para aplicaciones móviles
   - Almacenamiento seguro
   - Protección contra jailbreak/root.
   - Detección de manipulación o ingeniería inversa.
   - Validación biométrica conforme a estándares FIDO2.
8. Gobierno de datos y calidad de información
   - Catálogo de datos.
   - Linaje y trazabilidad.
   - Políticas de retención y eliminación.
   - Controles de integridad y consistencia.
9. Cumplimiento operativo y de monitoreo
   - Observabilidad (logs, métricas, trazas).
   - Alertas en tiempo real.
   - Detección de anomalías.
   - Monitoreo de fraude.
   - SLA/SLO/SLI definidos y auditables.

Principios de Arquitectura

La arquitectura propuesta se basa en principios de separación de responsabilidades, diseño impulsado por el dominio, seguridad por diseño, integración bajo gobierno, cumplimiento regulatorio financiero, alta disponibilidad, escalabilidad, resiliencia, observabilidad, excelencia operativa y documentación estandarizada.

Estos principios garantizarán que la solución sea segura, auditable, mantenible, escalable y alineada a las exigencias del sector financiero.

1. Separación de responsabilidades (SoC) y Arquitectura Hexagonal
   - El dominio debe permanecer puro, sin dependencias técnicas.
   - Los puertos deben definir *qué* necesita el dominio.
   - Los adapters deben implementar *cómo* se conecta con infraestructura.

- La resiliencia (Timeout, Fallback, Circuit Breaker, Caching) vive exclusivamente en los adapters (Adaptadores).

2. Diseño impulsado por el dominio (DDD)
   - Lenguaje ubicuo en puertos, servicios y modelos.
   - Utilizar Value Objects para evitar primitivos anémicos.
   - Establecer casos de uso explícitos para orquestar lógica de negocio.
   - Repositorios como abstracciones del dominio.
3. Resiliencia como primer factor de diseño
   - Aplicación sistemática de patrones:
     - Circuit Breaker
     - Timeout
     - Retry
     - Fallback
     - Bulkhead
     - Cache Aside
   - Evitar fallas en cascada y garantizar continuidad operativa.
4. Seguridad por diseño (Security by Design)
   - Autenticación basada en OAuth2.0 / OIDC.
   - MFA y biometría como parte del flujo de autorización.
   - Encriptación en tránsito y en reposo.
   - Zero Trust: cada llamada autenticada y autorizada.
   - Gestión segura de secretos.
5. Cumplimiento regulatorio financiero
   - Auditoría completa e inmutable.
   - KYC integrados en los flujos (Onboarding, autenticación, activación de productos).
   - AML integrado en los flujos (transferencias interbancarias, movimientos de alto valor, operaciones repetitivas y sospechosas)
   - Protección de datos personales (privacidad, minimización, retención).
   - Trazabilidad end-to-end para conciliación y no repudio.
6. Alta disponibilidad, tolerancia a fallos y DR
   - Despliegue multi-zona o multi-región para plataformas en cloud.
   - Despliegue en un centro de datos alterno para plataformas en la premisa.
   - RTO/RPO definidos y probados.
   - Auto-healing y escalamiento automático.
   - Infraestructura modelada en C4 Deployment.
7. Integración desacoplada y gobernada
   - API Gateway como punto único de entrada.
   - Validación de contratos (schema validation).
   - Versionado semántico (SemVer) para APIs.
   - Integración con Core bancario mediante puertos y adapters.
8. Observabilidad integral
   - Logs estructurados y trazas distribuidas.
   - Métricas de negocio y técnicas.
   - Alertas proactivas y detección de anomalías.
   - Correlación de eventos para auditoría.
9. Escalabilidad y baja latencia
   - Uso de caching estratégico (Cache Aside).
   - Minimización de hops (saltos o llamadas) entre servicios.
   - Uso eficiente de colas/eventos cuando aplique.
   - Diseño orientado a performance desde el inicio.
10. Excelencia operativa
    - Automatización de despliegues (CI/CD).

- Infraestructura como código.
- Pruebas automatizadas (unitarias, contract testing, integración).
- Monitoreo continuo de SLA/SLO/SLI.

11. Documentación clara y estandarizada
    - Uso de C4 para modelar:
        - Contexto
        - Contenedores
        - Componentes
        - Infraestructura (Deployment)
    - Artefactos institucionales: matrices, checklists, flujos, políticas.

Arquitectura de Alto Nivel

La arquitectura propuesta se basa en un modelo modular y desacoplado, organizado bajo principios de Arquitectura Hexagonal y DDD para mantener un dominio puro, expresivo y libre de dependencias técnicas. La solución se compone de dos aplicaciones frontales, una SPA web y una aplicación móvil multiplataforma, que consumen servicios transaccionales expuestos a través de un API Gateway, el cual actúa como punto único de entrada, permitiendo control de seguridad y gobierno. Los servicios de backend se estructuran como contenedores lógicos independientes que ejecutan funcionalidades clave: consulta de movimientos, transferencias, pagos, información del cliente y onboarding biométrico.

Estos servicios se integran con dos sistemas (1) core bancario e (2) información de clientes, con dos proveedores externos de notificaciones y con mecanismos de autenticación basados en OAuth 2.0, incorporando biometría y validaciones KYC/AML dentro de los flujos críticos.

La arquitectura está diseñada para poder operar en la nube con alta disponibilidad, baja latencia y tolerancia a fallos, aplicando patrones de resiliencia como Circuit Breaker, Timeout, Retry, Fallback y Cache Aside en los adapters de infraestructura. La solución incorpora auditoría completa e inmutable de todas las acciones del cliente, así como un mecanismo especializado de persistencia para clientes frecuentes.

La infraestructura se despliega mediante contenedores orquestados, con observabilidad integral (logs, métricas y trazas distribuidas), monitoreo continuo y controles de seguridad.

La integración con sistemas externos se realiza mediante adapters especializados, mientras que la capa de dominio se mantiene aislada mediante repositorios como abstracciones del negocio.

La arquitectura presentada garantiza cumplimiento regulatorio, escalabilidad, continuidad operativa y una experiencia segura y consistente para los usuarios.

ARQUITECTURA DE SOLUCIONES

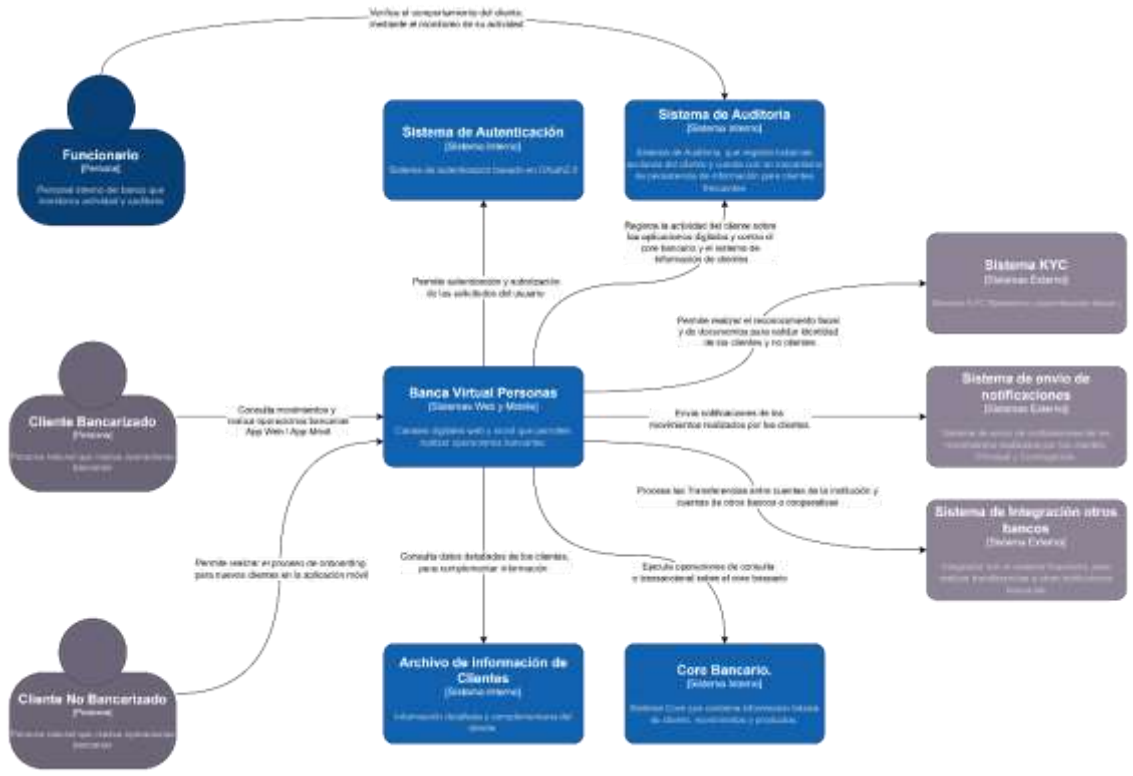


Fig1. Diagrama de contexto del sistema de banca por internet

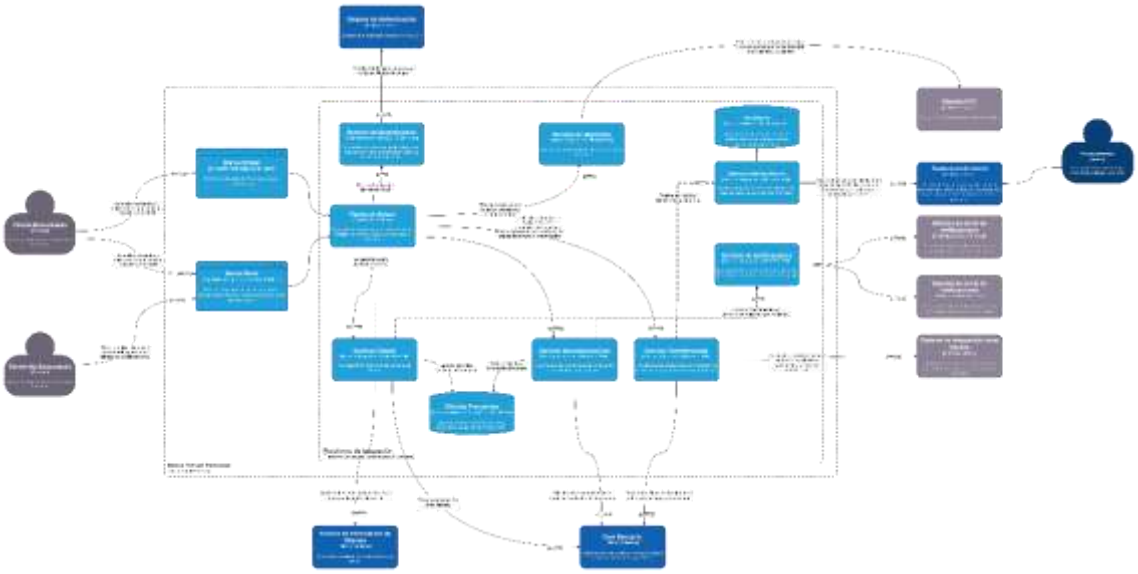


Fig2. Diagrama de contenedores del sistema de banca por internet

Descripción de la solución.

Aplicaciones de Front End

Para el desarrollo de la aplicación web reactiva con SPA, se propone desarrollo en React, porque permite el reúso de componentes, uso de virtual DOM mejora el rendimiento, integración a aplicaciones actuales es más sencilla, su desarrollo declarativo simplifica la lógica y reduce errores; es ideal para desarrollo de SPA (Single Page Applications), compatible con API Rest y GraphQL, además permitirá reutilizar parte de la lógica en las aplicaciones móviles.

ARQUITECTURA DE SOLUCIONES

Para el desarrollo de aplicaciones móviles, hay varias opciones disponibles Xamarin, Ionic, NativeScript, KMM, PWA, etc.

Para este proyecto se consideran los frameworks que sean multiplataforma nativos, que permitan el acceso a APIs nativas de los dispositivos, se toma en cuenta que los equipos usen como estándar una sola tecnología, para que se puedan adaptar rápidamente, y que permita que se puedan reutilizar componentes y lógica de la aplicación web.

Considerando estos elementos contamos con las opciones de React Native, Ionic React o Flutter, trabajar con Flutter para aplicaciones móviles, nos proporciona una interfaz muy elaborada, mejores animaciones y mejor rendimiento, pero implica tener un equipo especializado en Dart y que domine el framework y una curva de aprendizaje más alta.

Tomando en cuenta que uno de los requerimientos es poder realizar autenticación por medios biométricos, utilizando la cámara y aprovechar la biometría del dispositivo, como FaceID y TouchID y la posibilidad de integrarse con servicios de terceros que cumplan esa función; deben cumplirse los siguientes criterios (1) acceso completo a APIs nativas, (2) soporte para librerías de terceros y (3) alto rendimiento. Los que cumplen muy bien con estos criterios son las opciones React Native y Flutter.

Considerando todos los criterios expuestos la opción más recomendable es usar React Native, permitirá aprovechar el conocimiento de los equipos en React, permitirá el reúso de lógica y componentes desarrollados para las aplicaciones web, la curva de aprendizaje es más corta ya que el equipo usa React/JS para las aplicaciones web, aunque la UI cambia, la arquitectura y librerías de JS son totalmente compatibles.

Con respecto al posible uso de servicios de terceros para autenticación biométrica (facial) son varios candidatos, FacePhi (Authentication y Onboarding), Entrust Identity security e Incode.

Ambos SDKs Facephi e incode son compatible con React Native.

FacePhi actualmente tiene mayor presencia en Latinoamérica, proporciona una precisión facial muy alta y biometría de comportamiento para protección contra fraude, es una buena opción si se necesita seguridad de un nivel alto y se recomienda para ser usado como servicio de biometría y verificación de identidad.

Flujo de autenticación a usar con OAuth2.0

El sistema debe usar el flujo OAuth 2.0 Authorization Code con PKCE, este flujo es recomendado para aplicaciones web, móviles y SPAs por su seguridad y compatibilidad con estándares actuales.

El cliente obtiene un authorization code tras autenticación del usuario, lo intercambia por tokens y utiliza el access_token para consumir servicios protegidos a través del API Gateway. El flujo garantiza seguridad, cumplimiento regulatorio, trazabilidad y compatibilidad con MFA, biometría y validaciones KYC/AML.

Como es el flujo para el consumo de servicios

1. La aplicación (web, móvil o SPA) inicia el proceso y redirige al usuario al servidor de autorización con los parámetros necesarios: client_id, redirect_uri, scope, response_type=code y el code_challenge (PKCE).
2. El usuario se autentica, El servidor de autorización valida credenciales, aplica MFA y cualquier política adicional (biometría, riesgo, etc.).

3. Tras autenticarse, el usuario es redirigido a la aplicación con un código temporal de un solo uso (authorization code).
4. La aplicación, intercambia el código por tokens, envía el authorization_code y el code_verifier al servidor de autorización, el servidor devuelve: access_token
5. Cada llamada al API Gateway incluye: Authorization: Bearer < access_token >; con esto el gateway valida firma, expiración, scopes y claims.
6. Si el flujo se configura para renovación del token, la app usa el refresh_token para obtener nuevos access_token sin pedir al usuario que vuelva a autenticarse.

Onboarding para nuevos clientes

La aplicación móvil incorpora un flujo de onboarding que integra reconocimiento facial y verificación documental como parte de la autorización inicial del cliente.

Durante este proceso se valida la identidad mediante servicios biométricos y KYC/AML, se crea el registro del cliente en el core bancario y se emite una identidad digital en el proveedor de identidad (OAuth2.0/OIDC).

A partir del onboarding, el usuario accede al sistema mediante un flujo estándar de autenticación OAuth2.0 utilizando usuario y contraseña, pudiendo habilitar la biometría del dispositivo (huella o reconocimiento facial nativo) como mecanismo de autenticación simplificada, que actúa como factor local para proteger el uso de tokens y sesiones

Fig 3. Componentes Servicio Biometría (integración con el servicio externo)

Visión general del flujo

El "sistema de onboarding con reconocimiento facial" se va a convertir en un módulo de identidad y alta de clientes dentro de tu arquitectura móvil:

Primero, el proceso de Onboarding incluye verificación de identidad (KYC biométrico). El usuario se registra, captura documentos, realiza reconocimiento facial y se valida su identidad contra servicios internos/externos (KYC/AML, verificación documental, listas, etc.).

Segundo, se realiza la emisión de credenciales y factores de autenticación, una vez validado, se crea el usuario en el Identity Provider (IdP) y se registran los métodos de autenticación: usuario/clave, biometría del dispositivo (huella/FaceID), y asociación con el dispositivo, es decir, activar la biometría del dispositivo como forma de desbloqueo local y como factor dentro del flujo OAuth esto permite liberar el uso de un refresh token o un secure storage local.

Tercero, queda habilitado la autenticación normal, a partir de este punto, el usuario ya no usa el flujo de onboarding, simplemente se autentica mediante OAuth2.0/OIDC usando usuario/clave o la biometría del dispositivo.

Mecanismo especializado para clientes frecuentes

En este diseño, el mecanismo especializado de persistencia para clientes frecuentes consiste en una ruta alterna de almacenamiento y de consulta, diseñada para ofrecer baja latencia, mayor

disponibilidad y acceso optimizado a la información de los clientes que realizan operaciones recurrentes o de alto valor.

Este mecanismo se implementa mediante un Strategy Pattern en el dominio, que decide si un cliente debe ser atendido por la persistencia estándar o por la persistencia especializada. La lógica de negocio no debe conocer detalles técnicos, solo debe invocar un repositorio abstracto, y el adapter resuelve la estrategia adecuada.

El Strategy Pattern se define en el dominio como una abstracción del comportamiento, pero sus implementaciones concretas viven en los adaptadores de infraestructura. El dominio decide *qué* estrategia usar; los adaptadores implementan *cómo* se ejecuta.

Esto mantiene dominio limpio, infraestructura desacoplada, resiliencia en los adaptadores, mejora la capacidad de pruebas y claridad conceptual.

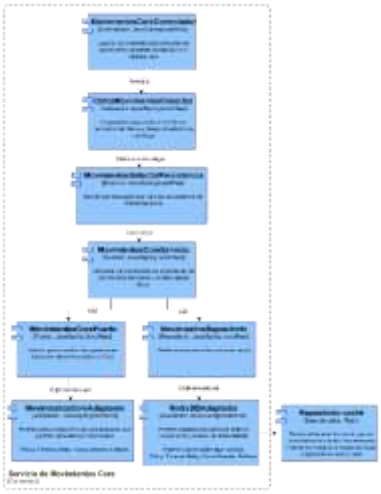


Fig 4. Componentes Servicio Movimientos

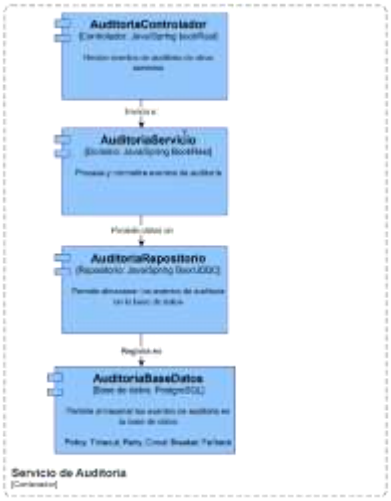
Fig 5. Componentes Servicio Cliente

La persistencia especializada se implementará en la capa de infraestructura, mediante un adapter dedicado, que usa tecnologías para lectura rápida, en este caso Redis. El adapter mantiene datos críticos del cliente (perfil, límites, preferencias, últimos movimientos, tokens de sesión, reglas AML/KYC precalculadas) en un formato accesible.

El diseño incluye políticas de cache-aside, replicación, TTL inteligente y sincronización eventual con la base de datos principal. De esta forma, los clientes frecuentes obtienen respuestas más rápidas, y el sistema mantiene consistencia, auditoría completa y resiliencia ante fallos.

Auditoria del Sistema

El sistema de auditoría es definido como un componente transversal y desacoplado que captura, registra y preserva todas las acciones relevantes ejecutadas por los clientes y por los servicios internos.



Recibe eventos estructurados provenientes de los casos de uso del dominio. Cada evento incluye información mínima obligatoria —identidad del usuario, timestamp, operación realizada, parámetros críticos, resultado, canal, dispositivo y correlación transaccional— garantizando trazabilidad completa y no repudio.

El dominio no conoce detalles técnicos del almacenamiento: solo emite eventos de auditoría a través de un repositorio de auditoría, que es implementado por un adapter especializado en la capa de infraestructura.

Fig 6. Componentes Servicio Auditoria

En la infraestructura, el sistema de auditoría utiliza un mecanismo de persistencia inmutable y seguro, basado en almacenamiento append-only, cifrado en reposo y firmado digitalmente para asegurar integridad.

Los eventos se almacenan en un Audit Log Store optimizado para escritura secuencial.

El sistema soporta retención configurable según normativa, indexación para búsquedas regulatorias, correlación de eventos mediante trace IDs y exportación controlada para procesos de cumplimiento, conciliación y análisis forense.

Servicio de Transferencias

El Servicio de Transferencias es un componente backend que orquesta de forma segura, auditable y resiliente el proceso de mover fondos entre cuentas propias, internas o interbancarias.

Expone un conjunto de endpoints protegidos por OAuth 2.0 a través del API Gateway, y encapsula toda la lógica de negocio mediante casos de uso explícitos

El dominio aplica reglas de negocio como validación de saldo, límites diarios, perfil de riesgo.

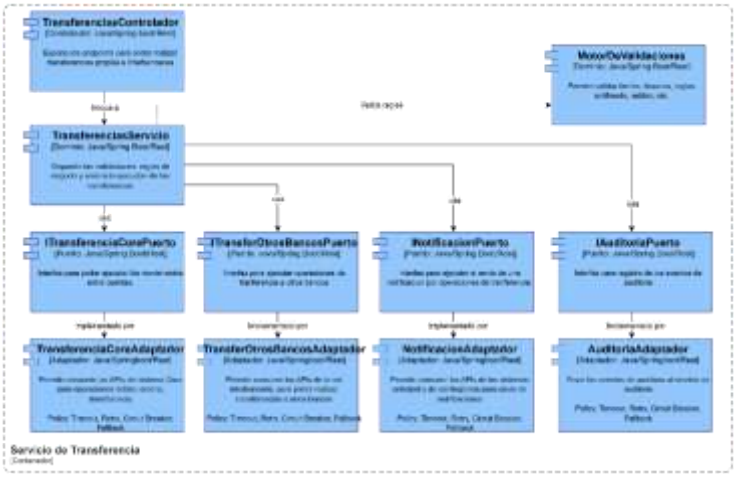


Fig 7. Componentes Servicio Transferencia

El servicio no conoce detalles técnicos de persistencia ni de integración: interactúa únicamente con puertos del dominio manteniendo un modelo limpio.

En la capa de infraestructura, los adapters implementan la integración con el Core bancario, motores de pagos, sistemas interbancarios, servicios de notificaciones y mecanismos de auditoría.

Cada adapter aplica patrones de resiliencia como Circuit Breaker, Timeout, Retry y Fallback para evitar fallas en cascada y garantizar continuidad operativa.

El servicio registra cada paso del flujo en el sistema de auditoría mediante un AuditPort, asegurando trazabilidad completa y cumplimiento regulatorio.

El resultado es un servicio robusto, seguro, auditable y desacoplado, capaz de operar bajo alta demanda y cumplir con los estándares del sector.

Alta Disponibilidad y Tolerancia a Fallos

La arquitectura está diseñada para mantenerse operativa incluso cuando uno de sus componentes falla. Esto se logra porque cada capa del modelo —bases de datos, caché distribuido, mensajería, servicios puede ejecutarse en múltiples instancias y en zonas distintas, evitando depender de un único punto.

Los servicios continúan funcionando porque las plataformas base (base transaccional, mensajería, almacenamiento y monitoreo) están configuradas con replicación y recuperación automática.

Además, el diseño híbrido permite operar con el Core y los sistemas Internos y externos en paralelo, lo que añade una capa adicional de continuidad. Si un componente presenta problemas en uno de los sistemas, otra instancia puede seguir procesando mientras se estabiliza el servicio afectado.

En conjunto, estos elementos aseguran que el ecosistema completo —canales, integraciones, servicios internos y externos— tenga una operación continua, resistente y preparada para recuperarse rápidamente ante fallas.

Observabilidad y Monitoreo

La arquitectura incorpora observabilidad desde la base, apoyándose en un conjunto de servicios nativos de AWS que permiten monitorear el comportamiento de todos los componentes. Los servicios de negocio, los canales digitales, la mensajería, la base transaccional y las integraciones externas reportan métricas y eventos de forma centralizada.

A nivel de monitoreo, Amazon CloudWatch se utiliza para recolectar métricas de infraestructura y de aplicación, generar dashboards operativos y configurar alarmas ante degradaciones de rendimiento, aumento de errores o tiempos de respuesta anómalos. Los logs de aplicaciones y de acceso se centralizan en CloudWatch Logs, y pueden ser indexados en Amazon OpenSearch Service para facilitar búsquedas y análisis más avanzados.

Para el seguimiento detallado de transacciones end-to-end entre BFF, servicios de dominio, y los sistemas interno y externo, se puede utilizar AWS X-Ray, lo que permite identificar con precisión en qué salto del flujo híbrido (Legacy / Digital) se está produciendo una latencia o un fallo. Adicionalmente, eventos relevantes de seguridad y cambios de configuración pueden auditarse mediante AWS CloudTrail, reforzando la trazabilidad de la plataforma.
En conjunto, esta combinación de CloudWatch, OpenSearch, X-Ray y CloudTrail entrega una vista completa y en tiempo casi real del estado del ecosistema, permitiendo una operación proactiva y reduciendo el tiempo de detección y resolución de incidentes.

ARQUITECTURA DE SOLUCIONES

Infraestructura de Despliegue

La infraestructura se despliega sobre AWS utilizando un enfoque completamente administrado y orientado a microservicios. Los contenedores de backend se ejecutan en Amazon ECS sobre AWS Fargate, eliminando la gestión de servidores y permitiendo escalado automático.

Las imágenes se almacenan en Amazon ECR, mientras que la configuración dinámica y los feature flags se gestionan mediante AWS AppConfig. El descubrimiento de servicios se resuelve con AWS Cloud Map, y la comunicación interna se asegura mediante IAM y redes privadas dentro de VPC.

Para datos y eventos, la arquitectura utiliza Amazon DynamoDB como base NoSQL de baja latencia y Amazon MSK para streaming y procesamiento en tiempo real. La capa de API puede exponerse mediante AWS AppSync (GraphQL) o API Gateway, según el caso.

La seguridad perimetral se refuerza con AWS WAF, certificados TLS gestionados por AWS Certificate Manager, y secretos protegidos en AWS Secrets Manager.

La observabilidad se implementa con Amazon CloudWatch para métricas y logs, AWS CloudTrail para auditoría de acciones, y Amazon OpenSearch para análisis avanzado. Finalmente, toda la infraestructura se define como código mediante AWS CloudFormation, garantizando despliegues reproducibles, gobernados y auditables.

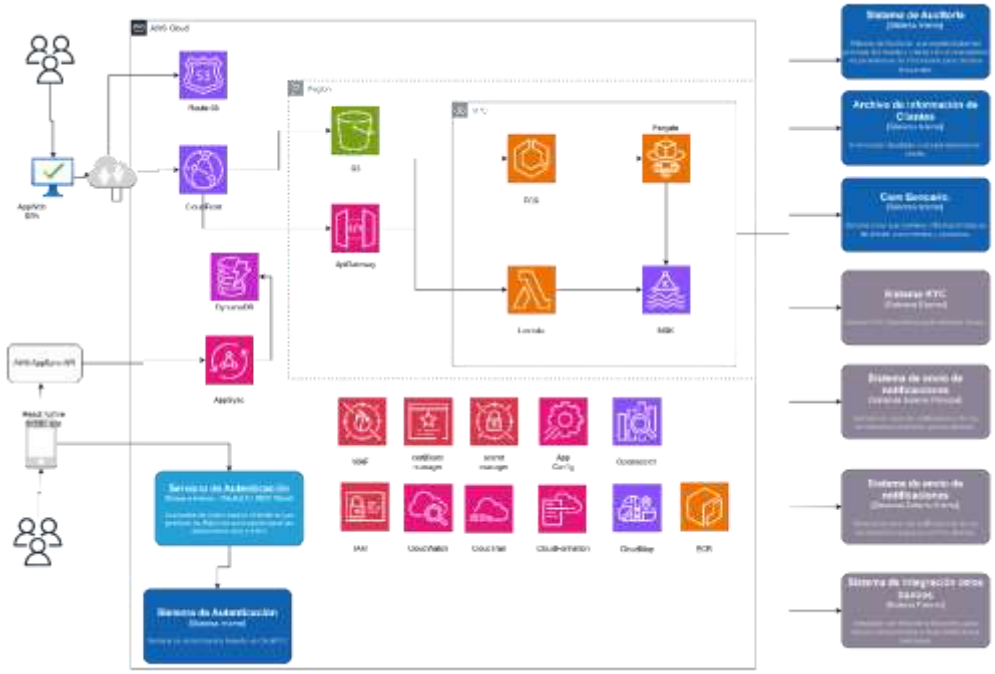


Fig 8. Deployment en AWS

Componentes de la infraestructura

Aplicación WEB

Amazon API Gateway lo ayuda a crear, publicar, mantener, monitorear y proteger las API REST, HTTP y de WebSocket a cualquier escala.

AWS CloudFormation ayuda a configurar los recursos de AWS, aprovisionarlos de manera rápida y coherente y administrarlos durante todo su ciclo de vida en las regiones y Cuentas de AWS.

Amazon CloudFront acelera la distribución del contenido web distribuyéndolo a través de una red mundial de centros de datos, lo que reduce la latencia y mejora el rendimiento.

ARQUITECTURA DE SOLUCIONES

AWS CloudTrail lo ayuda a auditar la gobernanza, el cumplimiento, y el riesgo operativo de la Cuenta de AWS.
Amazon CloudWatch le permite supervisar las métricas de sus recursos de AWS y las aplicaciones que se ponen en marcha en AWS en tiempo real.
Amazon Route 53 es un servicio web de sistema de nombres de dominio (DNS) escalable y de alta disponibilidad.
Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento de objetos basado en la nube que lo ayuda a almacenar, proteger y recuperar cualquier cantidad de datos.

Aplicación Mobile
AWS AppSync
Servicio administrado para crear APIs GraphQL seguras y en tiempo real, permitiendo a apps web/móviles obtener exactamente los datos que necesitan. Puede conectarse a DynamoDB, Lambda, HTTP y otras fuentes de datos.
Amazon DynamoDB
Base de datos NoSQL totalmente administrada, con latencias de milisegundos y escalabilidad automática para cargas de alto rendimiento. Ideal para modelos clave-valor y documentos.

Integración
Amazon MSK (Managed Streaming for Apache Kafka)
Servicio administrado que facilita la ingesta y procesamiento de datos en tiempo real usando Apache Kafka con alta disponibilidad. Permite construir pipelines de streaming y conectar con otros servicios como Lambda o DynamoDB.
Amazon ECS (Elastic Container Service)
Orquestador de contenedores altamente escalable y administrado. Permite ejecutar workloads en EC2 o Fargate con integración nativa a AWS.
AWS Fargate
Motor serverless para ejecutar contenedores sin administrar servidores. Escala automáticamente y se integra con ECS/EKS.
Amazon ECR (Elastic Container Registry)
Registro privado de contenedores totalmente administrado. Permite almacenar, versionar y escanear imágenes Docker.

Seguridad
AWS WAF (Web Application Firewall)
Firewall administrado que protege aplicaciones web contra ataques comunes (SQLi, XSS, bots). Permite reglas personalizadas y filtrado a nivel de capa 7.
AWS Certificate Manager (ACM)
Servicio que gestiona certificados SSL/TLS, automatizando emisión, renovación y despliegue. Facilita cifrado en tránsito sin manejar certificados manualmente.
AWS Secrets Manager
Almacena y rota credenciales, claves API y secretos de forma segura. Permite rotación automática y acceso granular mediante IAM.
AWS IAM (Identity and Access Management)
Sistema centralizado para gestionar identidades, permisos y políticas de acceso. Controla quién puede hacer qué en cada recurso AWS.

Monitoreo y observabilidad
Amazon OpenSearch Service
Motor administrado para búsqueda, análisis y observabilidad basado en OpenSearch/Elasticsearch. Ideal para logs, métricas, dashboards y análisis en tiempo real.

ARQUITECTURA DE SOLUCIONES

Amazon CloudWatch

Plataforma de monitoreo que recolecta métricas, logs y trazas. Permite alarmas, dashboards y observabilidad de aplicaciones e infraestructura.

AWS CloudTrail

Servicio que registra todas las acciones realizadas en la cuenta AWS. Proporciona auditoría, trazabilidad y cumplimiento normativo.

Gestión de cambios

AWS AppConfig

Servicio para gestionar configuraciones dinámicas y feature flags sin redeploy. Permite despliegues controlados, validación y rollback seguro.

AWS CloudFormation

Infraestructura como código para crear, actualizar y versionar recursos AWS mediante plantillas declarativas. Permite despliegues reproducibles y controlados.

AWS Cloud Map

Servicio de descubrimiento de servicios que permite registrar y resolver endpoints dinámicos. Útil para microservicios y arquitecturas distribuidas.

Link: https://github.com/Yesseania-Auqui-Fajardo/Arquitectura-Solucion.git