Solución de Arquitectura

El presente documento expone la arquitectura propuesta para el desarrollo e implementación de un **sistema de banca por internet** para la entidad **BP**, diseñado para ofrecer a sus clientes una experiencia digital moderna, segura y confiable. Esta solución permitirá a los usuarios consultar el histórico de sus movimientos financieros, realizar transferencias y pagos, así como gestionar sus productos bancarios a través de múltiples canales (web y móvil).

La arquitectura ha sido concebida bajo los lineamientos del **modelo C4** (Contexto, Contenedores y Componentes), complementada con vistas de infraestructura, seguridad, cumplimiento normativo y operación. El diseño incorpora principios de **alta disponibilidad (HA)**, **resiliencia**, **seguridad de la información**, **eficiencia operativa** y **escalabilidad horizontal**, alineándose con las mejores prácticas en arquitectura de soluciones en la nube y con los requisitos del sector financiero.

Adicionalmente, se contemplan elementos normativos clave como la **Ley 1581 de 2012** sobre protección de datos personales en Colombia, la **Circular Básica Jurídica (CBJ) de la SFC**, y estándares internacionales como **ISO/IEC 27001**, **PCI-DSS** y **OWASP**, asegurando así una solución alineada a los marcos regulatorios y de seguridad exigidos.

El sistema está pensado para ser **desacoplado, modular y extensible**, soportando no solo los requerimientos actuales, sino también futuras funcionalidades que la entidad pueda incorporar. Se han utilizado tecnologías y patrones de diseño ampliamente adoptados en la industria, garantizando una base sólida, confiable y evolutiva.

Este documento servirá como guía técnica y estratégica para los equipos de desarrollo, infraestructura y seguridad, así como para los stakeholders del proyecto, y será la base para una implementación controlada, medible y alineada con los objetivos de la entidad BP.

Diagrama de Contexto

Justificación de Decisiones Arquitectónicas

1. Separación entre Aplicación Web (SPA) y Aplicación Móvil

Implementar dos aplicaciones frontales: una SPA para acceso vía navegador y una app móvil multiplataforma.

Justificación:

- Permite **optimizar la experiencia de usuario** en cada canal, aplicando buenas prácticas de UX para cada tipo de dispositivo.
- Facilita el uso de **tecnologías específicas** y eficientes en cada canal (ej. Angular o React para la SPA; Flutter o React Native para la app móvil).
- Soporta **escalabilidad del canal de atención**, al permitir futuras extensiones como asistentes virtuales, chatbots, etc.

2. Uso de un API Gateway como capa de integración

Centralizar las solicitudes desde los canales hacia los servicios backend a través de un API Gateway.

Justificación:

- Habilita un **punto centralizado de control**, aplicando políticas de seguridad (OAuth2, CORS, rate limiting).
- Permite una **observabilidad mejorada**, registrando métricas, trazas y errores por canal o cliente.
- Facilita la **gestión de versiones y enrutamiento dinámico** sin afectar directamente a los clientes.

3. Interacción con Sistemas Core y Complementarios

Integrarse con dos sistemas existentes: el core bancario y un sistema complementario.

Justificación:

- Respeta el principio de **reuse over rebuild**, maximizando el valor de sistemas existentes sin duplicar lógicas ni datos.

- Permite implementar una **arquitectura orientada a servicios (SOA)**, consumiendo solo lo necesario por tipo de operación.
- Mitiga el riesgo de dependencia total de una sola fuente, facilitando la **resiliencia del sistema** si uno de los orígenes falla.

4. Uso de Servicio de Autenticación basado en OAuth2.0

Autenticación y autorización delegadas en un servicio OAuth2.0 corporativo.

Justificación:

- Estándar ampliamente adoptado que permite **autenticación segura y escalable**, cumpliendo normativas como PCI-DSS y ISO 27001.
- Facilita la **implementación de flujos específicos** como Authorization Code Flow (seguro para apps móviles y web), integrando múltiples factores (MFA).
- Permite **interoperabilidad con otros sistemas internos o externos**, como portales de autogestión, evitando credenciales duplicadas.

5. Implementación de Canales de Notificación múltiples

Integrar mínimo dos canales de notificación (email, push, SMS).

Justificación:

- Cumple con **regulaciones financieras de notificación de movimientos** en tiempo real (por ejemplo, Circular Básica Jurídica SFC en Colombia).
- Garantiza **redundancia y cobertura**: si un canal falla (push), se envía por otro (email o SMS).
- Mejora la **experiencia del cliente**, brindando tranquilidad y trazabilidad de operaciones sensibles.

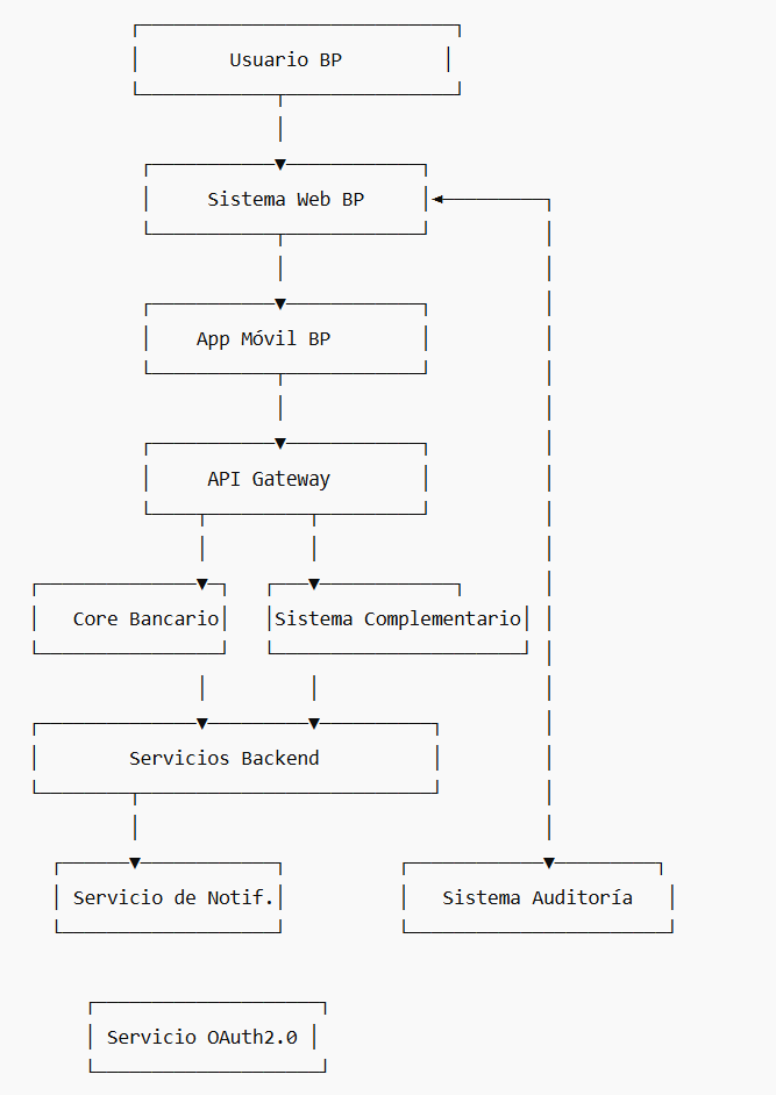
6. Sistema de Auditoría externo

Registrar todas las acciones del cliente en un sistema de auditoría dedicado.

Justificación:

- Permite cumplir con estándares de **auditoría exigidos por organismos financieros**, como SOX, Basel II o la Superintendencia Financiera de Colombia.
- Desacopla la lógica operativa del registro histórico, mejorando el **rendimiento transaccional** del sistema principal.
- Habilita una trazabilidad completa para análisis forense, antifraude o litigios.

Diagrama



Elemento	Tipo	Descripción
Usuario Cliente BP	Persona	Usuario final del sistema de banca por internet. Accede desde app móvil o navegador web.
Sistema de Banca Web	Sistema principal	Aplicación SPA donde el cliente consulta sus movimientos, transfiere dinero, y gestiona cuentas.
App Móvil BP	Sistema principal	Aplicación móvil multiplataforma para realizar las mismas funciones que el portal web.
Core Bancario	Sistema externo	Sistema principal con datos de productos, movimientos y cliente.
Sistema Complementario	Sistema externo	Provee detalles adicionales del cliente cuando se requiere más información.
Servicio de Autenticación (OAuth2.0)	Sistema externo o propio	Sistema de autenticación federado. Soporta login con usuario/clave, huella o reconocimiento facial.
API Gateway	Sistema de integración	Capa que enruta las solicitudes desde el front a los servicios backend.
Sistemas de Notificación	Sistemas externos/propios	Mínimo 2 canales (ej. correo electrónico y notificaciones push/SMS) usados para avisar movimientos y operaciones importantes al usuario.
Sistema de Auditoría	Sistema secundario	Almacena las acciones realizadas por el cliente en el sistema para trazabilidad y control normativo.
Administrador TI	Persona (opcional)	Usuario interno que puede acceder a monitoreo y configuración del sistema, incluyendo logs y trazas de auditoría.

Diagrama de Contenedores

Justificación de Decisiones Arquitectónicas

1. Separación entre SPA Web y App Móvil como contenedores independientes

- Permite una **optimización independiente por canal**, adaptando la arquitectura a las capacidades del dispositivo (por ejemplo, biometría en móviles).
- Facilita **ciclos de despliegue separados** y pruebas diferenciadas para cada canal, mejorando la agilidad del desarrollo.
- Garantiza **resiliencia**: una caída en uno de los canales no afecta al otro, alineándose con principios de tolerancia a fallos.

2. Uso de un API Gateway como contenedor de entrada

- Permite aplicar **políticas de seguridad transversales** (auth, rate limiting, CORS, logging) en un punto centralizado.
- Reduce la complejidad del backend al aislar la lógica de enrutamiento y autenticación, fomentando el **principio de separación de responsabilidades**.
- Aumenta la **observabilidad** del sistema al exponer métricas detalladas del uso de los endpoints (tiempo de respuesta, errores, frecuencia).

3. Arquitectura basada en servicios (backend desacoplado)

- Cada servicio (ej. transferencias, movimientos, datos básicos) está **alineado a una capacidad de negocio**, fomentando la modularidad y la mantenibilidad.
- Facilita el **escalamiento horizontal** independiente por tipo de carga (ej. transferencias puede escalar más que consultas básicas).
- Habilita una transición futura a **microservicios**, incluso si actualmente se implementa en un monolito modular.

4. Servicio de Autenticación federado (OAuth2.0 como contenedor externo)

Justificación:

- Evita la implementación interna de complejidades de autenticación, reduciendo la **superficie de ataque** y riesgos de cumplimiento normativo.

- Permite implementar **flujos seguros** como Authorization Code + PKCE, recomendados por el IETF para apps móviles y web SPA.
- Garantiza compatibilidad con **MFA y SSO** en caso de expansión hacia usuarios internos o canales externos.

5. Servicio de Onboarding biométrico como contenedor separado

- Aisla la lógica especializada de validación facial y antifraude, facilitando el **cumplimiento de la Ley de Habeas Data y protección de datos biométricos.**
- Permite **integración con terceros** certificados en biometría (ej. Azure Face API), sin acoplar la lógica a la aplicación móvil.
- Mejora la **experiencia de usuario** al reducir pasos de registro y validar identidad sin documentos físicos.

6. Servicios de Notificaciones desacoplados

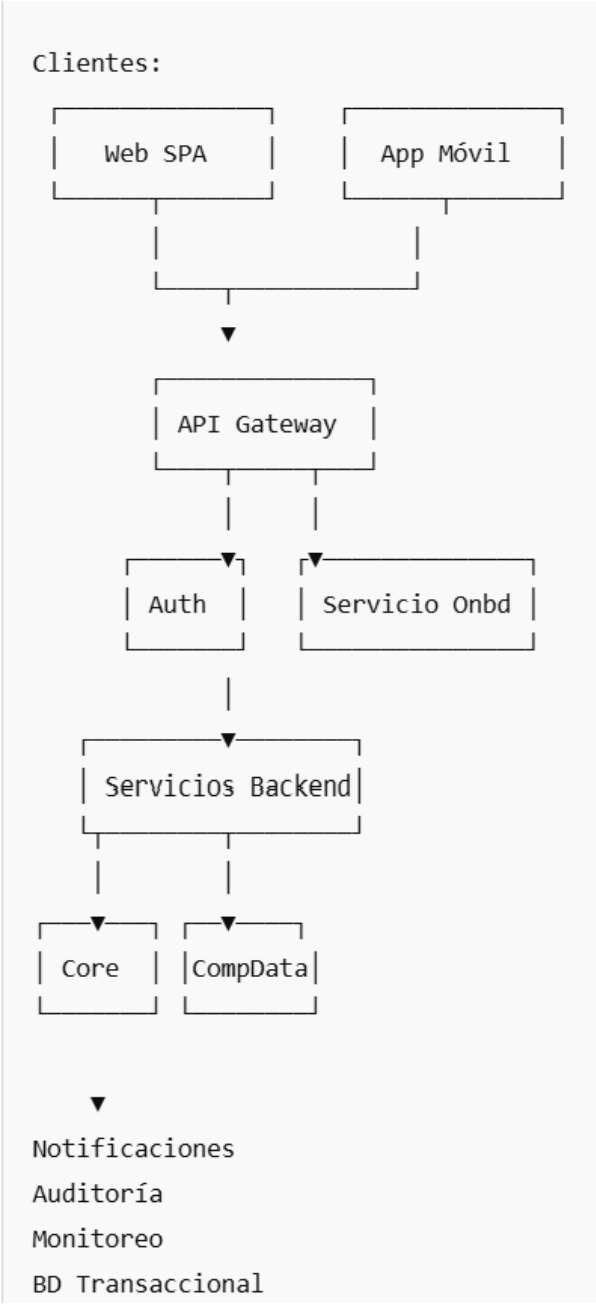
- Permiten una arquitectura orientada a eventos, habilitando el patrón **event-driven** para acciones como movimientos o alertas de seguridad.
- La separación mejora la **resiliencia y capacidad de reintento** en canales fallidos (push, email, SMS).
- Cumple con **requisitos regulatorios** de entidades financieras en cuanto a notificación oportuna de transacciones.

7. Bases de Datos separadas (Transaccional y Auditoría)

- Mejora la **performance del sistema transaccional**, evitando sobrecarga por escrituras frecuentes de logs de auditoría.
- La base de auditoría puede escalarse con tecnologías orientadas a series de eventos o NoSQL, optimizando el **costo por volumen**.
- Facilita **retención histórica segura** para cumplir con normativas como SARLAFT, Basilea II, SOX o SFC.

8. Contenedor de Monitoreo independiente

Justificación:

- Garantiza la **observabilidad del sistema en producción**, con alertas y tableros en tiempo real.
- Permite **automatizar acciones correctivas (auto-healing)** en caso de fallos detectados.
- Refuerza los pilares de **excelencia operativa**, uno de los principios clave en arquitecturas modernas como las de AWS Well-Architected o Azure CAF.
- Diagrama
- 

Inventario de contenedores

Contenedor	Tipo	Descripción técnica
Web SPA (Angular/React)	Front-end	Aplicación web de banca en línea. Consume servicios vía API Gateway.
App Móvil (Flutter/ReactNative)	Front-end	Aplicación móvil multiplataforma. Se conecta vía API Gateway. Incluye onboarding y biometría.
API Gateway (Azure API Mgmt / AWS API Gateway)	Integración	Expone servicios del sistema, aplica políticas de seguridad, versionamiento, trazas, rate limiting.
Backend de Servicios	Back-end (microservicios o monolito modular)	Maneja lógica de negocio para operaciones bancarias. Incluye 3 servicios principales: Consulta básica, Movimientos, Transferencias.
Servicio de Notificaciones	Back-end externo/propio	Gestiona el envío de notificaciones (correo, push, SMS). Puede ser Twilio, Firebase, SendGrid, o solución interna.
Servicio de Autenticación (OAuth2.0)	Externo/Corporativo	Proveedor de identidad (IDP) que autentica clientes y entrega tokens JWT.
Servicio de Onboarding Biométrico	Servicio externo o integrado	Verifica identidad mediante reconocimiento facial. Puede usarse FaceTec, Microsoft Azure Face API, etc.
Base de Datos Transaccional	Data	Almacena datos operativos: sesiones, tokens, transacciones temporales.
Base de Datos de Auditoría	Data	Registra logs de acciones del usuario, transferencias, accesos y eventos importantes.
Core Bancario	Sistema externo	Fuente principal de información financiera (productos, movimientos).

Contenedor	Tipo	Descripción técnica
Sistema Complementario	Sistema externo	Enriquecimiento de datos del cliente.
Plataforma de Monitoreo	Observabilidad	Stack de monitoreo (Ej: Prometheus, Grafana, Azure Monitor, CloudWatch).

Recomendaciones Tecnológicas por Contenedor

Contenedor	Tecnología Recomendada
Web SPA	Angular 17 o React 18
App Móvil	Flutter o React Native
API Gateway	Azure API Management / AWS API Gateway
Backend	.NET 8 / Node.js / Spring Boot
Autenticación	Keycloak / Auth0 / Azure AD B2C
Notificaciones	Firebase / Twilio / SendGrid
Auditoría	MongoDB / PostgreSQL / ElasticSearch
Monitoreo	Prometheus + Grafana / Azure Monitor
Onboarding biométrico	FaceTec / Azure Face API / iBeta

- Las apps cliente se comunican exclusivamente a través del **API Gateway**.
- El **backend** se comunica con el **Core Bancario** y el **sistema complementario** por servicios REST/SOAP.
- Las operaciones son **auditadas** en la base de datos de auditoría.
- **Notificaciones** se disparan desde el backend tras eventos críticos.
- La autenticación usa **Authorization Code Flow** en OAuth2.0.

Diagrama de Componentes

Decisiones Arquitectónicas

1. Separación entre Controllers, Services y Repositories

- Promueve el **principio de separación de responsabilidades (SRP)**, donde cada capa tiene una función específica: exposición (Controller), lógica de negocio (Service) y persistencia (Repository).
- Facilita la **testabilidad** del sistema, permitiendo pruebas unitarias y de integración aisladas por capa.
- Mejora la **escalabilidad y mantenibilidad** del sistema en el tiempo, permitiendo cambios sin impactos cruzados.

2. Uso de Servicios de Negocio (CustomerService, TransferService, NotificationService)

- Centralizan la lógica funcional, evitando duplicación de reglas y flujos dentro de los controllers o adaptadores.
- Permiten la **orquestación de flujos complejos**, como validación, ejecución y notificación de transferencias en un solo punto lógico.
- Alinean la solución con una **arquitectura por capacidades de negocio**, que puede evolucionar a microservicios.

3. Aplicación del Patrón Adapter para integración con sistemas externos

- Permite **desacoplar el backend** de la implementación específica de los sistemas externos (Core, sistema complementario, OAuth2, biometría).
- Facilita la **sustitución o extensión** de un proveedor externo sin impactar el resto del sistema.
- Aplica principios de diseño SOLID (en especial el Principio de Inversión de Dependencias), favoreciendo la inyección de dependencias y pruebas simuladas (mocks).

4. Diseño del NotificationService con múltiples adaptadores

- Implementa el patrón **Strategy o Chain of Responsibility**, permitiendo el envío por diferentes canales en función de prioridad o disponibilidad.
- Mejora la **resiliencia** del sistema frente a fallos en uno de los canales de comunicación (ej: SMS, email, push).
- Permite cumplir con **requerimientos regulatorios**, que exigen redundancia y trazabilidad en notificaciones de transacciones.

5. Uso de AuditService y AuditRepository separados

- Separa el manejo de auditoría del resto de la lógica de negocio, lo cual es crucial en sectores financieros donde se exige **no modificar registros históricos**.
- Optimiza la persistencia en la base de auditoría, permitiendo usar esquemas NoSQL o particionamiento por volumen.
- Apoya el cumplimiento de normativas como **Basilea II, SOX, Circular 052 SFC**, donde los eventos deben ser completos, inviolables y almacenados por años.

6. Integración con OAuth2 a través de AuthClient

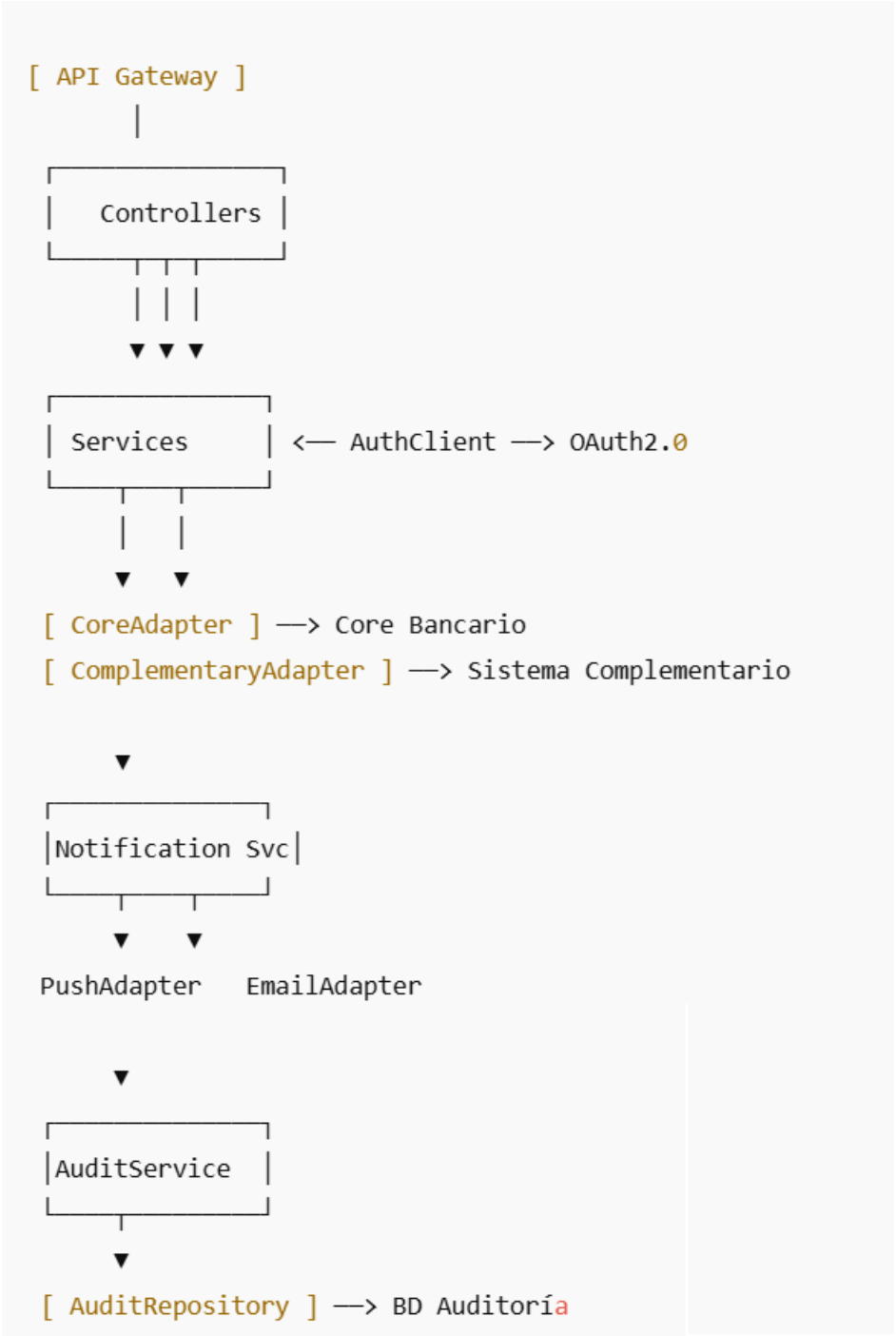
Justificación:

- Evita acoplar el backend a un proveedor de identidad específico, habilitando la posibilidad de utilizar Keycloak, Azure AD B2C, Auth0, etc.
- Permite implementar **autenticación delegada** segura, sin exponer credenciales del usuario ni manejar tokens sensibles dentro de la lógica central.

7. Cliente Biométrico desacoplado (BiometricClient)

- Encapsula la lógica de comunicación con el proveedor biométrico, permitiendo reemplazarlo si cambian las políticas de onboarding o se habilita un proveedor nacional.
- Refuerza el cumplimiento de la **Ley de Protección de Datos (Ley 1581 de 2012 en Colombia)**, que requiere tratamiento especial de datos biométricos.

Diagrama



Estándares y Patrones aplicados

- **Controller-Service-Repository**: separa capas de presentación, lógica y persistencia.

- **Adapter Pattern**: facilita integración desacoplada con sistemas externos.
- **Single Responsibility**: cada componente tiene una función clara.
- **Audit Trail Pattern**: registro consistente y seguro de eventos de usuario.
- **Service Composition**: CustomerService y TransferService orquestan múltiples adaptadores.

Inventario de Componentes

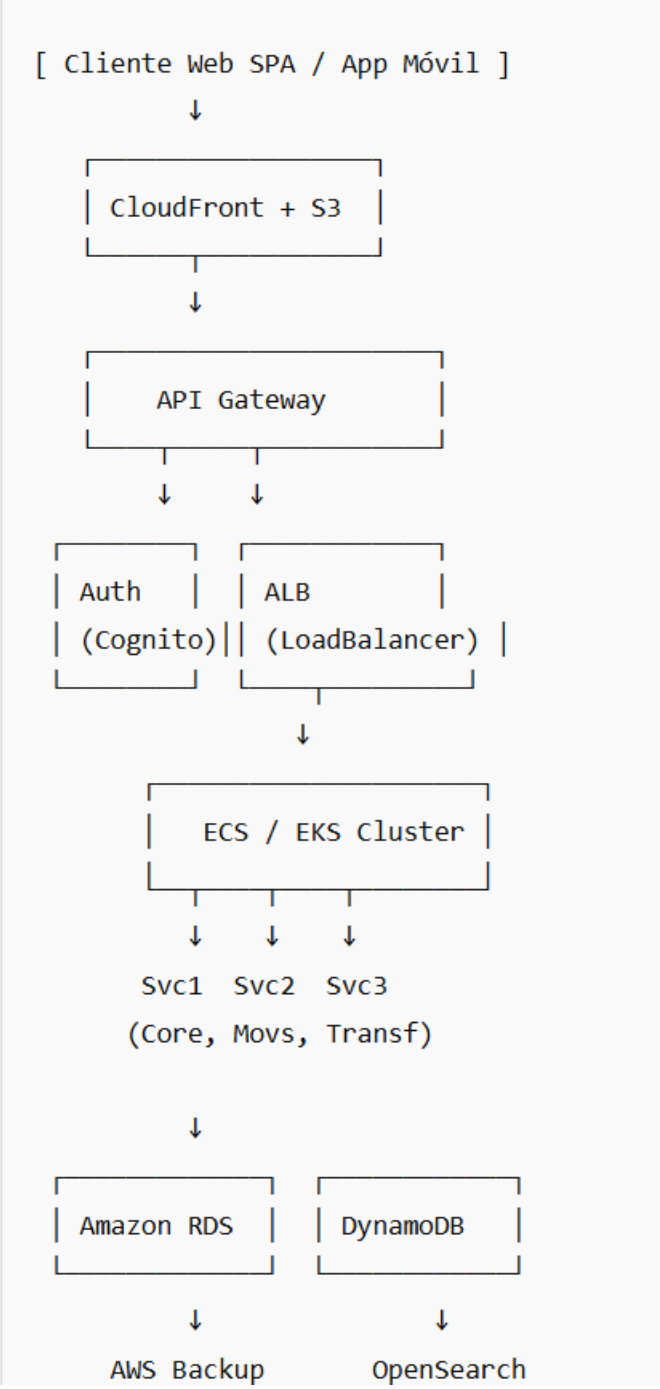
Mostrar cómo está estructurado internamente el backend, destacando los componentes responsables de manejar flujos funcionales como transferencias, movimientos, autenticación, notificaciones y auditoría.

Componente	Tipo / Rol	Descripción funcional
AuthController	Controller REST	Expone endpoints públicos para autenticación de usuario y validación de tokens (con OAuth2.0).
CustomerController	Controller REST	Expone endpoints para consulta de datos básicos del cliente.
TransactionController	Controller REST	Permite consultar movimientos y realizar transferencias.
NotificationService	Servicio interno	Orquesta envío de notificaciones usando adaptadores.
AuditService	Servicio interno	Registra en la base de datos de auditoría todas las acciones del usuario.
TransferService	Servicio de negocio	Implementa lógica de negocio de validaciones, límites y reglas para transferencias.
CustomerService	Servicio de negocio	Orquesta la consulta al Core y al Sistema Complementario para componer la información del cliente.
CoreAdapter	Componente de integración	Adaptador que consume los servicios del Core Bancario.
ComplementaryAdapter	Componente de integración	Adaptador para consumir el sistema complementario.

Componente	Tipo / Rol	Descripción funcional
NotificationAdapterEmail	Componente de integración	Encargado del canal de notificación por correo electrónico (ej. SendGrid).
NotificationAdapterPush	Componente de integración	Canal de notificación push (ej. Firebase Cloud Messaging).
AuditRepository	Persistencia	Encapsula la lógica de guardado de registros de auditoría.
TransactionRepository	Persistencia	Encapsula persistencia de transacciones en base temporal.
AuthClient	Cliente externo (OAuth2.0)	Se comunica con el servidor OAuth2 para validar tokens, intercambiar credenciales, etc.
BiometricClient	Cliente externo	Cliente para integración con el proveedor de onboarding biométrico (FaceTec / Azure Face API).

Vista de Infraestructura en AWS

Modelar cómo se despliega la solución sobre AWS, garantizando disponibilidad, escalabilidad, monitoreo, seguridad y cumplimiento normativo.



Capa	Servicio AWS	Función
Canal web/móvil	Amazon CloudFront + S3	Distribución global de la SPA (React/Angular) y delivery de assets.
Balanceo y acceso	AWS API Gateway + ALB	Expone APIs RESTful, desacopla clientes, enruta tráfico hacia servicios backend.
Servicios backend	Amazon ECS Fargate o EKS (Kubernetes)	Contenedores escalables que ejecutan los microservicios.
Datos operacionales	Amazon RDS (PostgreSQL) o Aurora	Base de datos relacional para transacciones del sistema.
Auditoría	Amazon DynamoDB o OpenSearch Service	Base de datos NoSQL/analítica para logs de acciones de usuario.
Notificaciones	Amazon SNS + Amazon SES	Canales de notificación multiuso (email, SMS, push).
Seguridad y claves	AWS IAM, KMS, Cognito (o IDP externo)	Gestión de identidad, permisos y cifrado de claves y tokens.
Almacenamiento seguro	AWS S3 con políticas privadas	Almacenamiento de fotos faciales del onboarding biométrico y archivos relacionados.
Monitoreo y DR	Amazon CloudWatch + AWS Backup + Multi-AZ	Logs, métricas, backups automáticos, alertas y recuperación ante fallos.

Vista de Seguridad en AWS

1. Identidad y Autenticación

- **Usuarios finales**: autenticación OAuth2.0 implementada con **Amazon Cognito** o **IDP externo** (Auth0, Azure AD B2C), usando Authorization Code Flow + PKCE.
- **Servicios internos**: permisos mínimos (principio de **least privilege**) usando **IAM Roles** por servicio.

2. Protección de APIs y comunicaciones

- API Gateway protege todos los endpoints con:
  - Autenticación JWT
  - WAF (Web Application Firewall)
  - Rate Limiting / Throttling
- Todo el tráfico usa **TLS 1.3**
- Token storage en frontend bajo política **secure, HttpOnly, SameSite=strict**

3. Cifrado en tránsito y en reposo

- RDS, DynamoDB, S3 cifrados con **AWS KMS**
- Todos los tokens, imágenes biométricas y archivos sensibles se almacenan **cifrados con claves rotadas automáticamente**

4. Auditoría y trazabilidad

- Todo evento relevante es:
  - Registrado en **DynamoDB/OpenSearch**
  - Emitido hacia **CloudTrail y CloudWatch Logs**
  - Notificado vía **SNS** a los equipos responsables (seguridad, auditoría, TI)

5. Recuperación ante desastres (DR)

- Despliegue en **multi-AZ**
- Backups automáticos con **AWS Backup**
- Replicación cruzada en otra región si se requiere (cross-region replication)